

«ИНФОРКОМ»

# ZX-РЕВЮ 1993

Выпуск 01-12

VERSION 1.00

PDF version by NUK, km, Василий Кормилицын, SVA-SM, Kendo Anoubis.

Feedback: [nuk@mail.ru](mailto:nuk@mail.ru)

|                                                                 |                |           |                       |
|-----------------------------------------------------------------|----------------|-----------|-----------------------|
| ZX-PEBЮ № 1-2'1993 "Third Impact"                               |                | Completed |                       |
| ZX-PEBЮ № 3-4'1993 "Return of Jedi"                             |                | Completed |                       |
| ZX-PEBЮ № 5-6'1993 "Hellraiser"                                 |                | Completed |                       |
| ZX-PEBЮ № 7-8'1993 "Clone War"                                  |                | Completed |                       |
| ZX-PEBЮ № 9-10'1993 "Alien Invasion"                            |                | Completed |                       |
| ZX-PEBЮ № 11-12'1993 "Avalon"                                   |                | Completed |                       |
| Спектрум в школе                                                | 1 <sup>1</sup> | Completed | by NUK                |
| Применение ассемблера для создания<br>быстроработающих программ | 9              | Completed | by NUK                |
| Защита программ                                                 | 13             | Completed | by km                 |
| Секреты TR-DOS                                                  | 21             | Completed | by km                 |
| Русификация резидентной процедурой                              | 28             | Completed | Василий Кормилицын    |
| Расчет резонансных характеристик                                | 30             | Completed | Василий Кормилицын    |
| Нестандартная загрузка                                          | 31             | Completed | Василий Кормилицын    |
| Резидентный отладчик машинного кода                             | 33             | Completed | by km                 |
| Программа "Теневой контур"                                      | 35             | Completed | by km                 |
| Конференция (Joe Blade-1)                                       | 38             | Completed | by NUK                |
| MYTH. History in the making                                     | 41             | Completed | by NUK                |
| The Dark Wheel                                                  | 42             | Completed | by NUK                |
| Применение ассемблера для создания<br>быстроработающих программ | 48             | Completed | by NUK                |
| Спектрум в школе                                                | 45             | Completed | by NUK                |
| Стилизованные шрифты                                            | 46             | Completed | by Kendo Anoubis      |
| Forum (Компрессия и декомпрессия)                               | 59             | Completed | by NUK                |
| Методы Билла Гилберта                                           | 64             | Completed | by NUK                |
| Sinclair Logo                                                   | 69             | Completed | by NUK                |
| AFRICAN SEEDS                                                   | 75             | Completed | by NUK                |
| CASH FLOW                                                       | 79             | Completed | by NUK                |
| Ответный Удар Кракса (Joe Blade-2)                              | 82             | Completed | by NUK                |
| The Dark Wheel                                                  | 84             | Completed | by NUK                |
| Спектрум в школе                                                | 89             | Completed | by km                 |
| Sinclair Logo                                                   | 90             | Completed | by NUK                |
| Применение ассемблера для создания<br>быстроработающих программ | 97             | Completed | by km                 |
| Музыкальные и звуковые эффекты                                  | 101            | Completed | by Kendo Anoubis      |
| Forum                                                           | 104            | Completed | by Василий Кормилицын |
| Опыт работы с BETA-BASIC 1.8                                    | 116            | Completed | by km                 |
| Сортировка русифицированных символьных строк                    | 116            | Completed | by Kendo Anoubis      |
| Просмотр защищенных программ                                    | 117            | Completed | by NUK                |
| Какие компьютеры мы выбираем                                    | 119            | Completed | by Василий Кормилицын |
| Распахнутые двери или открытый занавес?                         | 121            | Completed | by NUK                |
| Strategic Games                                                 | 122            | Completed | by NUK                |
| Советы экспертов. International Soccer                          | 123            | Completed | by NUK                |
| Сделайте сами. Block Buster                                     | 125            | Completed | by NUK                |
| Последняя Схватка (Joe Blade-3)                                 | 127            | Completed | by NUK                |
| Спектрум в школе                                                | 133            | Completed | by NUK                |
| Sinclair Logo                                                   | 134            | Completed | by Василий Кормилицын |
| Применение ассемблера для создания<br>быстроработающих программ | 139            | Completed | by NUK                |
| Forum                                                           | 146            | Completed | by km                 |
| Применение статического генератора случайных чисел              | 154            | Completed | by Kendo Anoubis      |
| SWORDS&SORCERY                                                  | 157            | Completed | by km                 |
| HEAVY ON THE MAGIC                                              | 158            | Completed | by km                 |
| Читатель - читателю. Аналог "Морского Боя"                      | 160            | Completed | by NUK                |
| Советы экспертов. Fighter Bomber                                | 161            | Completed | by NUK                |
| Советы экспертов. Mercenary                                     | 162            | Completed | by km                 |

<sup>1</sup> Номер страницы в оригинальном издании

|                                                                 |     |           |                       |
|-----------------------------------------------------------------|-----|-----------|-----------------------|
| Сделайте сами. DICEY                                            | 166 | Completed | by NUK                |
| Стратегия Капитана Кренона (Rebel Star). Часть 1.               | 171 | Completed | by NUK                |
| Авторская программа                                             | 175 | Completed | by NUK                |
| Спектрум в школе                                                | 177 | Completed | by NUK                |
| Sinclair Logo                                                   | 180 | Completed | by Василий Кормилицын |
| Применение ассемблера для создания<br>быстроработающих программ | 187 | Completed | by NUK                |
| IS-DOS                                                          | 190 | Completed | by NUK                |
| Компьютер и звук                                                | 195 | Completed | by NUK                |
| К вопросу о стандартизации                                      | 198 | Completed | by NUK                |
| Спессу + Астрология                                             | 201 | Completed | by NUK                |
| Adventure Games                                                 | 210 | Completed | by NUK                |
| Стратегия Капитана Кренона (Rebel Star). Часть 2.               | 216 | Completed | by NUK                |
| Спектрум в школе                                                | 221 | Completed | by NUK                |
| Sinclair Logo                                                   | 222 | Completed | by NUK                |
| Применение ассемблера для создания<br>быстроработающих программ | 227 | Completed | by SVA-SM, NUK        |
| Wham! The Music Box. Новые возможности                          | 235 | Completed | by SVA-SM, NUK        |
| Нестандартная загрузка                                          | 244 | Completed | by SVA-SM             |
| Исправленные ошибки ПЗУ                                         | 246 | Completed | by SVA-SM             |
| Adventure Building System                                       | 247 | Completed | by SVA-SM, NUK        |
| Forum                                                           | 253 | Completed | by SVA-SM, NUK        |
| Советы Экспертов. The Double                                    | 258 | Completed | by NUK                |
| Советы Экспертов. Urban Upstart                                 | 260 | Completed | by NUK                |
| Битва Усаги (Samurai Warrior)                                   | 262 | Completed | by NUK                |

## Содержание

|                                                                                        |            |
|----------------------------------------------------------------------------------------|------------|
| <b>СПЕКТРУМ В ШКОЛЕ.....</b>                                                           | <b>9</b>   |
| ПРОГРАММА "МОЗАИКА" .....                                                              | 9          |
| <b>ПРИМЕНЕНИЕ АССЕМБЛЕРА ДЛЯ СОЗДАНИЯ БЫСТРОРАБОТАЮЩИХ ПРОГРАММ .....</b>              | <b>25</b>  |
| ПРЕДИСЛОВИЕ .....                                                                      | 25         |
| 1. ВЫВОД НА ЭКРАН.....                                                                 | 26         |
| <b>ЗАЩИТА ПРОГРАММ .....</b>                                                           | <b>33</b>  |
| 1.1 Процедуры в машинных кодах, реализующие загрузку программ с магнитофона .....      | 33         |
| ГЛАВА 2. СТАНДАРТНЫЕ ПРОЦЕДУРЫ ЗАПИСИ ИНФОРМАЦИИ НА МАГНИТНУЮ ЛЕНТУ .....              | 40         |
| ГЛАВА 3. МЕТОДЫ ЗАЩИТЫ ОТ КОПИРОВАНИЯ ДЛЯ ПЭВМ "СПЕКТРУМ" .....                        | 45         |
| 1. Методика создания оригинальной процедуры записи информации на магнитную ленту. .... | 45         |
| 2. Методика создания оригинальной процедуры чтения информации с магнитной ленты. ....  | 47         |
| <b>СЕКРЕТЫ TRDOS .....</b>                                                             | <b>51</b>  |
| КАРТА ПАМЯТИ ОЗУ. ....                                                                 | 51         |
| СИСТЕМНЫЕ ПЕРЕМЕННЫЕ TR-DOS. ....                                                      | 54         |
| ФОРМАТ ЗАПИСИ НА ДИСК.....                                                             | 56         |
| ВЫПОЛНЕНИЕ КОМАНД TR-DOS ИЗ МАШИННОГО КОДА.....                                        | 58         |
| Программа "DIR". ....                                                                  | 61         |
| ПРИЕМЫ ЗАЩИТЫ ОТ КОПИРОВАНИЯ.....                                                      | 62         |
| Программа "OPEN". ....                                                                 | 63         |
| <b>FORUM .....</b>                                                                     | <b>65</b>  |
| РУСИФИКАЦИЯ РЕЗИДЕНТНОЙ ПРОЦЕДУРОЙ.....                                                | 65         |
| РАСЧЕТ РЕЗОНАНСНЫХ ХАРАКТЕРИСТИК.....                                                  | 70         |
| НЕСТАНДАРТНАЯ ЗАГРУЗКА. ....                                                           | 72         |
| <b>ПРОФЕССИОНАЛЬНЫЙ ПОДХОД.....</b>                                                    | <b>77</b>  |
| ОТЛАДЧИК МАШИННОГО КОДА "TRACER" .....                                                 | 77         |
| <b>МАЛЕНЬКИЕ ХИТРОСТИ .....</b>                                                        | <b>82</b>  |
| ПРОГРАММА "ТЕНЕВОЙ КОНТУР" .....                                                       | 82         |
| <b>КОМПЬЮТЕРНАЯ НОВЕЛЛА .....</b>                                                      | <b>87</b>  |
| КОНФЕРЕНЦИЯ .....                                                                      | 88         |
| <b>СОВЕТЫ ЭКСПЕРТОВ .....</b>                                                          | <b>94</b>  |
| MYTH. HISTORY IN THE MAKING .....                                                      | 94         |
| <b>THE DARK WHEEL.....</b>                                                             | <b>99</b>  |
| ГЛАВА 5.....                                                                           | 99         |
| ГЛАВА 6.....                                                                           | 101        |
| <b>СПЕКТРУМ В ШКОЛЕ.....</b>                                                           | <b>105</b> |
| <b>МАЛЕНЬКИЕ ХИТРОСТИ .....</b>                                                        | <b>108</b> |
| СТИЛИЗОВАННЫЕ ШРИФТЫ .....                                                             | 108        |
| <b>ПРИМЕНЕНИЕ АССЕМБЛЕРА ДЛЯ СОЗДАНИЯ БЫСТРОРАБОТАЮЩИХ ПРОГРАММ .....</b>              | <b>111</b> |
| 2. Команды PLOT, DRAW и CIRCLE .....                                                   | 112        |
| PLOT .....                                                                             | 112        |
| DRAW x,y. ....                                                                         | 123        |
| DRAW x,y,a .....                                                                       | 124        |
| CIRCLE x,y,r .....                                                                     | 124        |
| 3. СЧЕТ .....                                                                          | 125        |
| <b>FORUM .....</b>                                                                     | <b>131</b> |
| ДРУГОЙ ВАРИАНТ КОМПРЕССИИ.....                                                         | 135        |
| <b>ПРОФЕССИОНАЛЬНЫЙ ПОДХОД.....</b>                                                    | <b>140</b> |
| МЕТОДЫ БИЛЛА ГИЛЬБЕРТА. ....                                                           | 140        |
| Современные разработки Билла Гильберта.....                                            | 140        |

|                                                                           |            |
|---------------------------------------------------------------------------|------------|
| Iron Man .....                                                            | 141        |
| Sim City .....                                                            | 146        |
| <b>SINCLAIR LOGO .....</b>                                                | <b>153</b> |
| 1. ЧТО ТАКОЕ "ЛОГО" .....                                                 | 153        |
| "Черепашка" .....                                                         | 153        |
| ЛОГО и БЕЙСИК .....                                                       | 154        |
| Диалекты ЛОГО .....                                                       | 155        |
| Как пользоваться этой книгой .....                                        | 155        |
| 2. ПЕРВЫЕ ШАГИ .....                                                      | 156        |
| Редактирование .....                                                      | 159        |
| Сохранение программ на кассете .....                                      | 160        |
| Обслуживание памяти .....                                                 | 160        |
| Печать текста .....                                                       | 161        |
| 3. ПЕРЕМЕННЫЕ .....                                                       | 162        |
| Арифметика .....                                                          | 163        |
| <b>AFRICAN SEEDS .....</b>                                                | <b>165</b> |
| <b>CASH-FLOW .....</b>                                                    | <b>173</b> |
| <b>КОМПЬЮТЕРНАЯ НОВЕЛЛА .....</b>                                         | <b>179</b> |
| ОТВЕТНЫЙ УДАР КРАКСА .....                                                | 179        |
| <b>THE DARK WHEEL .....</b>                                               | <b>184</b> |
| ГЛАВА 7 .....                                                             | 187        |
| ГЛАВА 8 .....                                                             | 192        |
| <b>СПЕКТРУМ В ШКОЛЕ .....</b>                                             | <b>195</b> |
| <b>SINCLAIR LOGO .....</b>                                                | <b>197</b> |
| Команды "Черепашки" .....                                                 | 197        |
| Спирали .....                                                             | 198        |
| Объекты ЛОГО .....                                                        | 200        |
| Виды процедур .....                                                       | 204        |
| ГЛАВА 4. СЛОВА И СПИСКИ .....                                             | 205        |
| Выбор элементов списка .....                                              | 205        |
| Ввод слов и списков .....                                                 | 207        |
| Пустой список .....                                                       | 208        |
| Создание и слияние списков .....                                          | 208        |
| Работа со словами .....                                                   | 209        |
| <b>МАЛЕНЬКИЕ ХИТРОСТИ .....</b>                                           | <b>211</b> |
| <b>ПРИМЕНЕНИЕ АССЕМБЛЕРА ДЛЯ СОЗДАНИЯ БЫСТРОРАБОТАЮЩИХ ПРОГРАММ .....</b> | <b>212</b> |
| 4. СЛУЧАЙНЫЕ ЧИСЛА .....                                                  | 212        |
| 5. ОБСЛУЖИВАНИЕ КЛАВИАТУРЫ .....                                          | 215        |
| 5.1. Использование системной переменной LAST_KEY .....                    | 216        |
| 5.2 Опрос клавиатуры, как внешнего порта: IN A,(C) .....                  | 218        |
| 6. МУЗЫКАЛЬНЫЕ И ЗВУКОВЫЕ ЭФФЕКТЫ .....                                   | 220        |
| Интегральная форма записи действительных чисел .....                      | 220        |
| Упакованная форма записи действительных чисел .....                       | 221        |
| <b>FORUM .....</b>                                                        | <b>224</b> |
| <b>ВОЗВРАЩАЯСЬ К НАПЕЧАТАННОМУ .....</b>                                  | <b>249</b> |
| Опыт работы с BETA-BASIC 1.8 .....                                        | 249        |
| Сортировка русифицированных символьных строк .....                        | 250        |
| <b>ЧИТАТЕЛЬ - ЧИТАТЕЛЮ .....</b>                                          | <b>251</b> |
| ПРОСМОТР ЗАЩИЩЕННЫХ ПРОГРАММ .....                                        | 251        |
| КАКИЕ КОМПЬЮТЕРЫ МЫ ВЫБИРАЕМ .....                                        | 255        |
| НОВЫЕ РОКES .....                                                         | 258        |
| РАСПАХНУТЫЕ ДВЕРИ ИЛИ ОТКРЫТЫЙ ЗАНАВЕС? .....                             | 259        |

|                                                                                       |            |
|---------------------------------------------------------------------------------------|------------|
| <b>STRATEGIC GAMES.....</b>                                                           | <b>261</b> |
| ANNALS OF ROME.....                                                                   | 261        |
| GALLIPOLI .....                                                                       | 262        |
| <b>СОВЕТЫ ЭКСПЕРТОВ .....</b>                                                         | <b>264</b> |
| EMLYN HUGHES INTERNATIONAL SOCCER .....                                               | 264        |
| <b>СДЕЛАЙТЕ САМИ.....</b>                                                             | <b>270</b> |
| BLOCK BUSTER.....                                                                     | 270        |
| <b>КОМПЬЮТЕРНАЯ НОВЕЛЛА .....</b>                                                     | <b>274</b> |
| ПОСЛЕДНЯЯ СХВАТКА .....                                                               | 274        |
| <b>СПЕКТРУМ В ШКОЛЕ.....</b>                                                          | <b>285</b> |
| <b>SINCLAIR LOGO .....</b>                                                            | <b>288</b> |
| ГЛАВА 5 ПОВТОРЯЮЩИЕСЯ ОПЕРАЦИИ.....                                                   | 288        |
| Условия.....                                                                          | 288        |
| Математические условия. ....                                                          | 288        |
| Рекурсия. ....                                                                        | 290        |
| Генератор случайной последовательности. ....                                          | 293        |
| ГЛАВА 6. ЧЕРЕПАШЬЯ ГРАФИКА-2. ....                                                    | 295        |
| <b>ПРИМЕНЕНИЕ АССЕМБЛЕРА ДЛЯ СОЗДАНИЯ БЫСТРОРАБОТАЮЩИХ ПРОГРАММ .....</b>             | <b>301</b> |
| 7. ДВИЖЕНИЕ.....                                                                      | 301        |
| Перемещение знакоместа переднего плана экрана.....                                    | 301        |
| Перемещение изображения заднего плана экрана .....                                    | 303        |
| <b>НАША ПРЕЗЕНТАЦИЯ .....</b>                                                         | <b>315</b> |
| 5.6 Расчленение стрингов. ....                                                        | 315        |
| <b>FORUM .....</b>                                                                    | <b>317</b> |
| <b>ПРОФЕССИОНАЛЬНЫЙ ПОДХОД.....</b>                                                   | <b>333</b> |
| ПРИМЕНЕНИЕ СТАТИЧЕСКОГО ГЕНЕРАТОРА СЛУЧАЙНЫХ ЧИСЕЛ НА ПРИМЕРЕ ПРОГРАММЫ "ELITE" ..... | 333        |
| <b>ВОЗВРАЩАЯСЬ К НАПЕЧАТАННОМУ .....</b>                                              | <b>339</b> |
| SWORDS & SORCERY.....                                                                 | 339        |
| HEAVY ON THE MAGIC .....                                                              | 341        |
| <b>ЧИТАТЕЛЬ ЧИТАТЕЛЮ .....</b>                                                        | <b>345</b> |
| <b>СОВЕТЫ ЭКСПЕРТОВ .....</b>                                                         | <b>348</b> |
| FIGHTER BOMBER.....                                                                   | 348        |
| Загрузка игры .....                                                                   | 348        |
| Управление самолетом .....                                                            | 349        |
| Приборная доска.....                                                                  | 350        |
| Полет.....                                                                            | 350        |
| Военные миссии .....                                                                  | 351        |
| MERCENARY.....                                                                        | 352        |
| Управление игрой. ....                                                                | 353        |
| Работа с лентой. ....                                                                 | 355        |
| Экран программы .....                                                                 | 355        |
| Описание транспортных средств и предметов. ....                                       | 355        |
| <b>СДЕЛАЙТЕ САМИ.....</b>                                                             | <b>361</b> |
| DICEY.....                                                                            | 361        |
| <b>КОМПЬЮТЕРНАЯ НОВЕЛЛА .....</b>                                                     | <b>371</b> |
| СТРАТЕГИЯ КАПИТАНА КРЕНОНА. ....                                                      | 371        |
| Часть 1. Лунная база "Дельта". ....                                                   | 371        |
| <b>АВТОРСКАЯ ПРОГРАММА .....</b>                                                      | <b>380</b> |
| <b>СПЕКТРУМ В ШКОЛЕ.....</b>                                                          | <b>385</b> |

|                                                                         |            |
|-------------------------------------------------------------------------|------------|
| <b>SINCLAIR LOGO .....</b>                                              | <b>392</b> |
| 7. И СНОВА ОБ ОБРАБОТКЕ СПИСКОВ. ....                                   | 392        |
| Уровни процедур. ....                                                   | 394        |
| Глобальные и локальные переменные. ....                                 | 395        |
| Команда OUTPUT для рекуррентных процедур. ....                          | 396        |
| Вложенные списки. ....                                                  | 397        |
| Процедуры создания списков. ....                                        | 398        |
| Сортировка чисел. ....                                                  | 399        |
| Самообучающаяся игра. ....                                              | 402        |
| 8. НЕКОТОРЫЕ МАТЕМАТИЧЕСКИЕ ОПЕРАЦИИ. ....                              | 404        |
| ИСТИНА и ЛОЖЬ (TRUE и FALSE) ....                                       | 405        |
| Возведение в степень и извлечение корня. ....                           | 406        |
| Поиск простых чисел. ....                                               | 406        |
| Тригонометрические операции. ....                                       | 407        |
| <b>ПРИМЕНЕНИЕ АССЕМБЛЕРА ДЛЯ СОЗДАНИЯ БЫСТРАБОТАЮЩИХ ПРОГРАММ .....</b> | <b>410</b> |
| 8. КОМАНДЫ ATTRIBUTE, SCREEN\$ и POINT .....                            | 410        |
| 8.1. ATTRIBUTE (строка, столбец). ....                                  | 410        |
| 8.2 SCREEN\$ .....                                                      | 411        |
| 8.3 POINT .....                                                         | 413        |
| 9. ПРИНТЕР .....                                                        | 414        |
| 9.1. COPY .....                                                         | 414        |
| 9.2 LPRINT .....                                                        | 414        |
| 9.3 LLIST .....                                                         | 415        |
| <b>СТРАНИЦА IS-DOS .....</b>                                            | <b>416</b> |
| Первое знакомство с iS-DOS. ....                                        | 418        |
| <b>ПРОФЕССИОНАЛЬНЫЙ ПОДХОД.....</b>                                     | <b>426</b> |
| КОМПЬЮТЕР И ЗВУК .....                                                  | 426        |
| <b>ЧИТАТЕЛЬ - ЧИТАТЕЛЮ .....</b>                                        | <b>433</b> |
| К ВОПРОСУ О СТАНДАРТИЗАЦИИ .....                                        | 433        |
| СПЕССУ + АСТРОЛОГИЯ .....                                               | 441        |
| Что такое гороскоп. ....                                                | 441        |
| Основной расчет. ....                                                   | 443        |
| Календарь. ....                                                         | 444        |
| Ввод исходных данных. ....                                              | 445        |
| Астрологические дома. ....                                              | 445        |
| "Дебют". ....                                                           | 446        |
| Апология Бейсика. ....                                                  | 449        |
| Параметры орбит. ....                                                   | 451        |
| Положения Солнца и планет. ....                                         | 451        |
| Луна. ....                                                              | 453        |
| Расчет астрологических домов по системе В.Коха. ....                    | 455        |
| Управляющая программа. ....                                             | 455        |
| <b>СДЕЛАЙТЕ САМИ.....</b>                                               | <b>458</b> |
| ADVENTURE GAMES .....                                                   | 458        |
| Введение. ....                                                          | 458        |
| Структура системы. ....                                                 | 458        |
| Листинги. ....                                                          | 459        |
| Работа с листингами. ....                                               | 468        |
| Головная программа ABS. ....                                            | 469        |
| Возможные перемещения. ....                                             | 470        |
| Расположение объектов. ....                                             | 471        |
| Загрузчик. ....                                                         | 471        |
| Демонстрационная программа. ....                                        | 471        |
| <b>КОМПЬЮТЕРНАЯ НОВЕЛЛА .....</b>                                       | <b>472</b> |
| СТРАТЕГИЯ КАПИТАНА КРЕНОНА .....                                        | 472        |
| Часть 2. Столкновение. ....                                             | 472        |
| <b>СПЕКТРУМ В ШКОЛЕ.....</b>                                            | <b>482</b> |

|                                                                           |            |
|---------------------------------------------------------------------------|------------|
| <b>SINCLAIR LOGO .....</b>                                                | <b>484</b> |
| ГЛАВА 9. ВВОД И ВЫВОД ИНФОРМАЦИИ .....                                    | 484        |
| ЗВУК.....                                                                 | 489        |
| ГЛАВА 10. ЕСЛИ ЧТО-ТО НЕ ПОЛУЧАЕТСЯ.....                                  | 490        |
| <b>ПРИМЕНЕНИЕ АССЕМБЛЕРА ДЛЯ СОЗДАНИЯ БЫСТРОРАБОТАЮЩИХ ПРОГРАММ .....</b> | <b>496</b> |
| 10. ПЕРЕВОД БЕЙСИК-ПРОГРАММЫ В МАШИННЫЕ КОДЫ.....                         | 496        |
| Процедура HIT.....                                                        | 508        |
| Процедура END GAME .....                                                  | 510        |
| ПРОЦЕДУРА HOME.....                                                       | 512        |
| <b>ЧИТАТЕЛЬ - ЧИТАТЕЛЮ .....</b>                                          | <b>518</b> |
| WHAM! THE MUSIC BOX. НОВЫЕ ВОЗМОЖНОСТИ .....                              | 518        |
| Универсальный загрузчик.....                                              | 519        |
| Основной файл "m. box". .....                                             | 520        |
| Как "бороться" с COPY.....                                                | 526        |
| Компиляция. ....                                                          | 527        |
| Кодировка мелодии. ....                                                   | 529        |
| Дополнительные функции "WHAM". ....                                       | 529        |
| Нотный стан. ....                                                         | 530        |
| Рекомпиляции.....                                                         | 531        |
| Транспонирование мелодии.....                                             | 532        |
| Сокращение программы. ....                                                | 533        |
| Изменение кодовых блоков. ....                                            | 534        |
| <b>ВОЗВРАЩАЯСЬ К НАПЕЧАТАННОМУ .....</b>                                  | <b>536</b> |
| НЕСТАНДАРТНАЯ ЗАГРУЗКА .....                                              | 536        |
| ИСПРАВЛЕННЫЕ ОШИБКИ ПЗУ.....                                              | 542        |
| <b>ADVENTURE GAMES .....</b>                                              | <b>544</b> |
| "ADVENTURE BUILDING SYSTEM" .....                                         | 544        |
| Управление адвентюрной игрой. ....                                        | 547        |
| Анализатор текста.....                                                    | 548        |
| Таблица расположения объектов.....                                        | 550        |
| Таблица состояния флагов. ....                                            | 550        |
| Демонстрационная программа.....                                           | 551        |
| Обработка глаголов. ....                                                  | 552        |
| Примеры расшифровки подпрограмм. ....                                     | 553        |
| Кодовый блок программы.....                                               | 554        |
| Создание собственной игры.....                                            | 555        |
| С чего практически начать? .....                                          | 555        |
| Скорость работы и объём памяти. ....                                      | 555        |
| Структура подпрограмм. ....                                               | 556        |
| <b>FORUM .....</b>                                                        | <b>557</b> |
| ПРОБЛЕМЫ ELITE .....                                                      | 557        |
| Советы и секреты. ....                                                    | 564        |
| <b>СОВЕТЫ ЭКСПЕРТОВ .....</b>                                             | <b>569</b> |
| THE DOUBLE.....                                                           | 569        |
| Приложение. СЛОВАРЬ ЭКСПЕРТОВ .....                                       | 574        |
| URBAN UPSTART.....                                                        | 576        |
| <b>КОМПЬЮТЕРНАЯ НОВЕЛЛА .....</b>                                         | <b>581</b> |
| БИТВА УСАГИ.....                                                          | 581        |





"ИНФОРКОМ" 123019, Москва, Г-19, а/я 16

## СПЕКТРУМ В ШКОЛЕ

Для тех, кто использует компьютеры семейства "ZX-Spectrum" в школьных учреждениях, мы подготовили сегодня программу "МОЗАИКА". Интересным, на наш взгляд, в ней является то, что это программа двойного назначения. Ее можно применять на уроках информатики как в старших классах, так и в младших.

Более опытные ученики смогут набрать текст этой программы и отладить ее, а для младших школьников она явится замечательным первым шагом в компьютеризацию, поскольку поможет легко наладить процесс общения с компьютером и сделать его радостным.

У нас есть положительный опыт работы с этой программой шестилетних детей, которые без труда освоили всю необходимую технику управления.

Не лишним также будет отметить и тот факт, что применима эта программа не только в школе, но и конечно дома. В этом случае она доставит много приятных минут и детям и их родителям. Не пожалейте несколько утренних часов. Ваш сын или дочь на всю жизнь запомнят этот воскресный день.

Дополнительный интерес в этой программе представляет и то, что при ее наборе Вам предстоит ознакомиться со многими профессиональными приемами, например с подгрузкой своего знакогенератора, с организацией перехода по ошибкам ON ERROR GO TO, с работой шестнадцатиричного загрузчика, с организацией в программе нулевой строки, с заполнением этой строки блоком кодов REM FILL, с адаптацией программы под БЕТА-ДИСК-интерфейс и т.д. и т.п. Все эти вопросы здесь последовательно разобраны, но Вам могут (если Вы пожелаете) пригодиться ранее выпущенные нами сборники "ZX-PEBIO" за прошлые годы.

"ИНФОРКОМ" предоставляет Вам возможность заказать и получить по почте сборники "ZX-PEBIO-91" и "ZX-PEBIO-92", выполненные в виде аккуратных книжек стандартного формата. Наш адрес Вам известен.

Те, кто не хочет связываться с почтой, смогут приобрести их лично на нашем корпункте по адресу: г. Москва, ул. Новый Арбат (бывший просп. Калинина), д.2, о/с Г-19. На первом этаже Вы найдете нас по вывеске "ZX-PEBIO".

## ПРОГРАММА "МОЗАИКА"

В магазинах игрушек продается игра под названием "МОЗАИКА". Кто в детстве не играл в нее? В панель с отверстиями вставляются разноцветные фишки, которые образуют рисунок. Сегодня вниманию читателей предлагается компьютерный вариант этой игры. Программа может использоваться не только в младших классах школы, но даже в дошкольных учреждениях.

По существу это простейший в управлении графический редактор. Освоить профессиональный графический редактор, такой, как, например "ARTIST" или "ARTSTUDIO", ребенок не в состоянии, а программа "МОЗАИКА" поможет ему приобщиться к "компьютерной графике". В то же время, эта программа приучит ребенка пользоваться функциональными клавишами переключения цвета чернил, бумаги, режима работы "графика - текст" и т.п., то есть будет пройден "курс подготовки" перед освоением в будущем более сложных программ.

Для того, чтобы в деталях понять многие моменты, используемые в этой программе, следует обратиться к циклу ряда статей, опубликованных под рубрикой "Профессиональный подход" в ZX-РЕВЮ в 1992 году (см. стр. 113-116, 193-202, 241-248). Однако, если у Вас нет возможности ознакомиться с этими материалами, или нет желания вдаваться в подробности работы программы, Вы можете получить готовую программу, если будете строго придерживаться указаний по набору Бейсик программы и блоков кодов.

Теперь несколько слов о том, что же может эта программа. Для простоты пользования в программе не применяется полностью графический режим, при помощи которого можно рисовать линии, окружности и т.д. Применяется только псевдографика, позволяющая печатать какой-либо символ в то знакоместо экрана, где находятся курсор. Символьный набор же сделан такой, что при нажатии буквенных клавиш, печатаются всевозможные символы: кружочки (это прежде всего, как в обычной игре "Мозаика"), а также квадратики, треугольнички, крестики и т.д., (насколько хватает фантазии) по количеству буквенных клавиш.

Изображение строится, например, из кружочков, как в игре "Мозаика" из круглых фишек, но в отличие от прототипа, можно использовать "фишки" разной формы и к тому же не беспокоиться о том, что фишек нужного цвета не хватит для желаемого рисунка.

Символьный набор для игры построен таким образом, что переключившись в режим CAPS LOCK, можно печатать буквы русского алфавита. Этот режим нужен не для того, чтобы использовать программу в качестве текстового редактора, а для того, чтобы можно было подписать рисунок.

Для перемещения курсора по экрану используются функциональные клавиши перемещения курсора, а при помощи других функциональных клавиш производится переключение цвета чернил, бумаги, бордюра, включение и выключение режимов повышенной яркости и инверсии. При распределении этих функции между клавишами, ориентир был сделан на удобство пользования для клавиатуры "SPECTRUM+" или что то же самое, отечественного компьютера "ДЕЛЬТА-С". (Хотя это распределение Вы можете изменить на свой вкус.)

Краткая инструкция по пользованию программой всегда находится в памяти компьютера и в любой момент может быть вызвана на экран при нажатии BREAK, при этом изображение с экрана запоминается и после просмотра инструкции возвращается обратно на экран.

В программе предусмотрена возможность сохранения рисунка на внешнем носителе (магнитная лента, дискета) и загрузки рисунка с внешнего носителя. Предусмотрена также возможность оперативного сохранения рисунка в специальном буфере в памяти компьютера. С этим буфером можно производить следующие манипуляции: переслать рисунок с экрана в буфер, вызвать рисунок из буфера на экран, а также обменять содержимое экрана и буфера между собой.

После того, как работа над рисунком закончена, можно насладиться результатами своего труда под музыкальное сопровождение компьютера, нажав ENTER.

Для дальнейшей работы, при необходимости, может быть произведена очистка экрана.

Теперь, когда мы кратко ознакомились с возможностями программы "МОЗАИКА", можно переводить к подробному описанию программы. Она набрана на Бейсике, но в ней используются фрагменты в машинных кодах, в частности те, о которых говорилось в "ZX-РЕВЮ", см. выше.

Учитывая тот факт, что у многих пользователей имеется "БЕТА-ДИСК интерфейс", программа построена таким образом, что адаптация ее под "БЕТА-ДИСК" сводится к минимуму. Об этом будет подробный разговор в конце статьи.

Кроме Бейсик-файла и файла в машинных кодах для работы программы необходимо подготовить например, при помощи "ARTSTUDIO", заставку-картинку, на которой будет содержаться краткая инструкция по клавишам управления программой. Форма, цвет, размер шрифта, окружающее оформление - произвольное, пусть его Вам подскажет Ваша фантазия, но там кроме названия программы должна быть следующая информация:

## КЛАВИШИ УПРАВЛЕНИЯ:

|          |                      |
|----------|----------------------|
| [BREAK]  | -Вызов инструкции    |
| [E.MODE] | -Смена цвета чернил  |
| [GRAPH]  | -Изменение яркости   |
| [EDIT]   | -Смена цвета бумаги  |
| [INV.V.] | -Инверсия вкл./выкл. |
| [TR. V.] | -Смена цвета бордюра |
| [DELETE] | -Забой символа       |
| [ENTER]  | -Просмотр рисунка    |
| [C.LOCK] | -Режим графика/текст |

Буквенные клавиши - граф.символы

[CS]+буквенные клавиши, а также

[SS]+2,0,Y,U,D - русские буквы

|      |        |                      |
|------|--------|----------------------|
| STOP | [SS]+A | - Очистка экрана     |
| <=   | [SS]+Q | - Запоминание экрана |
| >=   | [SS]+E | - Экран из буфера    |
| <>   | [SS]+W | - Обмен экран/буфер  |
|      | [SS]+I | - Загрузка экрана    |
|      | [SS]+S | - Запись экрана      |

Подготовив файл с этой картинкой-инструкцией, запишите его на магнитофон под именем "mozaik \$" CODE 16384,6912.

Кроме этого надо подготовить еще один кодовый файл для работы программы. Это будет "mozaik" CODE 63824,1584. Сюда входят следующие коды:

Во-первых, это символьный набор, содержащий русские буквы и графические образы, причем кодам с 64 по 95 (заглавные латинские буквы) соответствуют русские буквы, а кодам с 96 по 127 (строчные латинские буквы) соответствуют графические символы. Русские буквы в символьном наборе следует расположить в соответствии с тем, как подписана Ваша клавиатура. Можно, например, в соответствии с кодами КОИ-7: "ЯВЕРТЫ...", а можно в соответствии с клавиатурой пишущей машинки: "ЙЦУКЕН...". Поскольку программа не претендует на стыковку с принтером в литерном режиме, то возможен любой вариант. Из-за того, что на "Спектруме" меньше клавишей, чем на пишущей машинке, часто применяется вариант со смещенным верхним рядом "ЦУКЕНГ...". В общем, как Вам нравится, так и делайте. Для примера ниже приведен символьный набор в кодах КОИ-7 (имеется в виду только соответствие заглавным латинским буквам, так как на месте строчных букв находятся графические символы "МОЗАИКИ"). Переставить русские буквы в соответствии с тем, как отмаркированы клавиши Вашего компьютера очень просто при помощи программы "ARTSTUDIO". В ней для этого предусмотрена специальная возможность. Надо загрузить предлагаемый символьный набор в "ARTSTUDIO", затем в текстовом режиме, начиная из левого верхнего угла, напечатать 32 русские буквы в соответствии с требуемым новым порядком расположения, а затем, задав окно вокруг этих напечатанных букв скопировать их с экрана в символьный набор, начиная с символа "@", пользуясь операцией CAPTURE FONT меню FONT EDITOR.

Сразу же за символьным набором следует блок UDG-графики. Правда, используется в программе всего лишь один символ UDG - "A", однако нет гарантии, что в дальнейшем программа не будет усовершенствоваться, поэтому зарезервирован полный объем UDG-блока, 168 байт.

Кроме символьного набора и UDG-блока в кодовом файле расположен блок кодов "ON ERROR GO TO" (см. "ZX-РЕВЮ" N 5-6 за 1992 г. стр. 113). Затем идут два небольших блока, длиной по 32 байта каждый, воспроизводящие звуковые эффекты, (см. "ZX-РЕВЮ" N 11-12

за 1992 г. стр. 241).

Далее расположено несколько блоков кодов, осуществляющих переброску из экрана в память или из памяти на экран. Для простой переброски используется команда LDIR, и тогда процедура переброски будет выглядеть следующим образом:

```
LD    HL, 16384    адрес экрана
LD    BC, 6912     длина кодов
LD    DE, 50000    адрес буфера
LDIR
RET
```

Такой блок кодов имеет длину 12 байт и может быть загружен в любое место памяти. Разумеется, при переброске экрана в буфер, содержимое последнего будет уничтожено, как и содержимое экрана будет уничтожено при переброске буфера на экран. Для того, чтобы обменять содержимое экрана и буфера без уничтожения обоих, применяется другая процедура. Она взята из программы "SUPERCODE" (N 27):

```
LD    DE, #C350    (1)
LD    HL, #4000
LD    B, #1B       (2)
PUSH  BC           (5)
LD    B, #00       (4)
LD    A, (HL)      (5)
PUSH  AF
LD    A, (DE)      (6)
LD    (HL), A      (7)
POP   AF           (8)
LD    (DE), A
INC   HL           (9)
INC   DE
DJNZ  #AB0B        (10)
POP   BC           (11)
DJNZ  #AB08        (12)
RET
```

Блок кодов имеет длину 25 байт. Его можно загружать в любое место памяти. Адрес загрузки является адресом старта. Адрес начала дисплейного файла экрана задан в регистре HL, а начало буфера - в регистре DE (1). Основу блока составляет процедура обмена содержимого одной пары ячеек экрана и буфера (5)-(9). Сначала в аккумулятор загружается содержимое ячейки экрана (5) и отправляется для хранения на стек. Затем в аккумулятор загружается содержимое соответствующей ячейки буфера (6) и из него перегружается в ячейку экрана (7). После этого в аккумулятор со стека считывается запомненное значение экранной ячейки (8) и записывается в буфер. После завершения обмена ячеек содержимое регистров HL и DE увеличиваются на единицу (9) для перехода к следующей паре ячеек.

Для полной переброски дисплейного файла вместе с атрибутами надо эту последовательность действий повторить 6912 раз. Это осуществляется двумя циклами (один внутри другого). Внутренний цикл (4)-(10) организован с максимально возможным числом повторений (в счетчик цикла - регистр B записывается 0, что соответствует 256 повторениям). Внешний цикл должен содержать, таким образом 5912/256 -27 (или #1B) повторений. Внешний цикл - (2)-(12). Так как и во внутреннем и во внешнем циклах счетчиком числа повторений является регистр B, то для того, чтобы значение регистра B не терялось при переходе от внешнего цикла ко внутреннему, оно запоминается на стеке (3) и снимается для контроля при переходе к внешнему циклу (11).

Далее - возврат в вызывающую программу.

Перед первым запуском этого блока кодов необходимо позаботиться о том, чтобы в буфере находились атрибуты белого экрана. Иначе, если буфер содержал, например, нули, то после переброски экран станет черный.

При однократном выполнении этого блока кодов происходит смена изображения экрана и буфера. При следующем выполнении восстанавливаются прежние изображения. Это используется, например, для вызова на экран, инструкции и сохранения при этом рисунка.

Для организации разнообразных обменов между экраном и буферами есть два варианта. Первый: иметь один блок кодов для простой переброски и один блок кодов для обмена экрана и буфера. А перед выполнением требуемой переброски подставлять в нужные ячейки памяти (при помощи Бейсика) соответствующие числа, задавая параметры HL и DE. Но учитывая небольшую длину кодовых блоков, их можно заготовить такое количество, какое потребуется для всевозможных перебросок (подготовка исходных параметров при помощи Бейсика съест больше памяти). Поэтому в программе используются несколько блоков для переброски.

Еще в кодовом файле "mozaik" CODE находится машиннокодовый музыкальный фрагмент, воспроизводящий знакомую мелодию. Он подготовлен при помощи музыкального редактора "WHAM".

Для того, чтобы Вы могли полностью воспроизвести у себя программу "МОЗАИКА", ниже приведены все используемые в программе блоки кодов.

Рассмотрим распределение памяти при работе программы "МОЗАИКА ". RAMTOP при старте программы устанавливается равным 49999.

|                                                                                          |       |
|------------------------------------------------------------------------------------------|-------|
| Буфер, зарезервированный<br>для запоминания экрана<br>6912 байт                          | 50000 |
| Буфер, зарезервированный<br>для хранения титульной заставки<br>(инструкции)<br>6912 байт | 56912 |
| Символьный набор<br>768 байт                                                             | 63824 |
| Блок UDG-графики<br>168 байт                                                             | 64592 |
| Блок ON ERROR GO TO<br>73 байта                                                          | 64760 |
| Звук SOUND 1<br>32 байта                                                                 | 64833 |
| Звук SOUND 2<br>32 байта                                                                 | 64865 |
| Процедура для переброски<br>экрана в буфер<br>12 байт                                    | 64897 |
| Процедура для переброски<br>буфера на экран<br>12 байт                                   | 64909 |
| Процедура для обмена<br>экрана и буфера<br>25 байт                                       | 64921 |
| Процедура для переброски<br>инструкции (заставки)<br>из памяти на экран<br>12 байт       | 64946 |
| Обмен экрана и заставки<br>25 байт                                                       | 64958 |
| Музыкальный фрагмент<br>525 байт                                                         | 64983 |
|                                                                                          | 65508 |

Теперь, когда мы разобрались в общих чертах с кодами, можно приступить к описанию Бейсиковского файла программы "МОЗАИКА". Текстовые сообщения на русском языке в листинге программы напечатаны по-русски, хотя так на самом деле не выглядят. Это сделано только для удобочитаемости программы. По-русски они будут выглядеть после переключения символьного набора при работе программы.

```
1 GO TO 100
2 BORDER 7: PAPER 7: INK 0: CLEAR 49999: POKE 23739,111: LOAD "mozaik $"CODE 16364
3 LOAD "mozaik"CODE 63834
4 RANDOMIZE USR 64946: GO SUB 7: GO SUB 20: CLS : RANDOMIZE USR 64897: RUN
5 SAVE "MOZAIK" LINE 2: SAVE "mozaik $"CODE 56912,6912: SAVE "mozaik"CODE 63824,1684
6 VERIFY "MOZAIK": VERIFY "mozaik $"CODE : VERIFY "mozaik"CODE : GO TO 9999
7 POKE 23675,80: POKE 23676,252: RETURN : REM udg
8 POKE 23606,80: POKE 23607,248: RETURN : REM rus
9 POKE 23606,0: POKE 23607,50: RETURN : REM lat
10 POKE 64813,PEEK 23670: POKE 64813,PEEK 23671: RANDOMIZE USR 64760: RETURN : REM err
20 GO SUB 8: PRINT AT 1,0: INK 7: PAPER 1: BRIGHT 1: FLASH 1;"      нажмите любую клавишу
..

22 BEEP .1,26: BEEP .1,20
24 RANDOMIZE 29: GO SUB 10: PAUSE 0
29 RETURN
30 BORDER 7: INPUT ;: PRINT PAPER 7: INVERSE 0: AT 0,0;"
39 RETURN
40 IF x>31 THEN LET x=0: LET y=y+1
42 IF x<0 THEN LET x=31: LET y=y-1
44 IF y>21 THEN LET y=1
46 IF y<1 THEN LET y=21
49 RETURN
50 GO SUB 30
52 PRINT AT 0,0:" введите имя (макс.10 символов) "
54 GO SUB 9: INPUT LINE f$: IF f$="" THEN LET f$="$"
56 GO SUB 30
59 RETURN
60 GO SUB 8: PRINT AT 0,0: INK 7: PAPER 2: BRIGHT 1: FLASH 1;"      ошибка !
   ": BEEP 1,0: PAUSE 0
69 RETURN
70 RANDOMIZE 60: GO SUB 10: LOAD f$ CODE 16384
79 RETURN
80 RANDOMIZE 60: GO SUB 10: SAVE f$ CODE 16384,6912
89 RETURN
100 LET y=20: LET x=3: LET bb=3: LET p=7: LET i=2: LET b=1: LET f=0: LET v=0
1000 RANDOMIZE 1000: GO SUB 10:BORDER bb: INPUT ;: GO SUB 8
1010 PRINT PAPER 7: INK 0: BRIGHT 0: INVERSE 0:AT 0,0;" чернила- яркость- бумага- "
1020 PRINT INK i: PAPER p: BRIGHT b: INVERSE v; AT 0,10: CHR$ 143 ; AT 0,29;" "; INK 7:AT
   0,20: INVERSE 1;" ";
1100 GO SUB 40
1110 PRINT OVER 1: PAPER 8: INK 8: BRIGHT 8: FLASH f: INVERSE v;AT y,x;"A";CHR$ 8;: REM A-
   GRAPH
1200 RANDOMIZE 2000: GO SUB 10:PAUSE 0: LET i$=INKEY$
1210 PRINT OVER 1: PAPER 8: INK 8: BRIGHT 8: FLASH 0: INVERSE v; "A";CHR$ 8;: REM A-GRAPH
1300 RANDOMIZE 1000: GO SUB 10
1310 IF CODE INKEY$=8 THEN LET x=x-1: GO TO 1100
1320 IF CODE INKEY$=9 THEN LET x=x+1: GO TO 1100
1330 IF CODE INKEY$=10 THEN LET y=y+1: GO TO 1100
1340 IF CODE INKEY$=11 THEN LET y=y-1: GO TO 1100
1350 IF CODE INKEY$=12 THEN LET x=x-f: GO SUB 40: PRINT AT y,x;" ": RANDOMIZE USR 64865: GO
   TO 1100
1360 IF INKEY$=" " THEN PRINT AT y,x;" ": RANDOMIZE USR 64833:LET x=x+1: GO TO 1100
1365 IF INKEY$="!" AND p+1=0 THEN GO TO 9999
1370 IF i$=" AND " THEN LET i$="[: REM TOKEN
1380 IF i$=" STEP " THEN LET i$="\": REM TOKEN
1390 IF i$=" OR " THEN LET i$="]": REM TOKEN
1400 IF i$=" " AND i$<="z" THEN PRINT PAPER p: INK i: BRIGHT b: INVERSE v;i$;: RANDOMIZE USR
   64833: LET x=x+f: GO TO 1100
1410 IF i$=" NOT " THEN GO SUB 50: GO SUB 80: REM TOKEN
```

```

1420 IF i$=" AT " THEN GO SUB 50: GO SUB 70: REM TOKEN
1430 IF i$="<=" THEN GO SUB 30: RANDOMIZE USR 64897: PRINT AT 0,0;
      "          экран в памяти          ": BEEP 0.1,26: PAUSE 50
1440 IF i$=">=" THEN GO SUB 30:RANDOMIZE USR 64909: BEEP 0.1,20
1450 IF i$="<>" THEN GO SUB 30: RANDOMIZE USR 64921: BEEP 0.1,26: BEEP 0.1,20
1460 IF i$=" STOP " THEN CLS
1470 IF CODE i$=15 THEN LET b=b=0
1480 IF CODE i$=5 THEN LET v=v=0
1490 IF CODE i$=6 THEN LET f=f=0: POKE 23658,8*(2-f)
1500 IF CODE i$=14 THEN LET i=(1+1)*(i<7)
1510 IF CODE i$=7 THEN LET p=(p+1)*(p<7)
1520 IF CODE i$=4 THEN LET bb=(bb+1)*(bb<7)
1530 IF CODE i$=13 THEN GO SUB 30: FOR c=0 TO 50: NEXT c: RANDOMIZE USR 64983
1999 GO TO 1000
2000 BORDER 7: INPUT ;
2010 RANDOMIZE USR 64958: BEEP .1,36: BEEP .1,20
2020 RANDOMIZE 2030: GO SUB 10:PAUSE 0
2030 RANDOMIZE USE 64958: BEEP .1,26: BEEP .1,20
2040 BORDER bb: INPUT ;: GO TO 1200
9999 GO SUB 9: BORDER 7: POKE 23658,16

```

Переменные, применяемые в программе:

y - вертикальная координата курсора (строка)  
 x - горизонтальная координата курсора (колонка)  
 bb - цвет бордюра  
 p - цвет бумаги (PAPER)  
 i - цвет чернил (INK)  
 b - яркость (BRIGHT)  
 f - мигание (FLASH)  
 v - инверсия (INVERSE)  
 i\$ - нажатая клавиша

Автостарт программы происходит со строки 2. Здесь задаются цвета экрана и резервируется место для работы блока кодов программы. Кроме этого, при помощи POKE отменяется вывод на экран сообщений типа: "Bytes:" при загрузке блоков кодов. Первый загружаемый блок - это заставка-инструкция, и благодаря этому POKE при загрузке следующего кодового файла (в строке 3) экран не будет испорчен. Этот прием уже был описан на страницах "РЕВЮ".

В строке 4 заставка-инструкция при помощи блока машинных кодов (смотри распределение памяти выше) перебрасывается в буфер для хранения инструкции (в адрес 56912). Далее подпрограмма GO SUB 7 производит переключение блока UDG на загруженные коды. Выполняется подпрограмма GO SUB 20, выводящая на экран табличку "нажмите любую клавишу" (место для вывода таблички назначьте такое, чтобы она не уничтожала информации на экране). Звуковой сигнал говорит о том, что загрузка программы завершена и она ждет команды начала работы. Программа останавливается на PAUSE 0 в строке 24, а непосредственно перед паузой конструкция: RANDOMIZE 29: GO SUB 10 задает параметр для работы кодового блока "ON ERROR GO TO", что означает переход в случае ошибки (нажатии BREAK) на строку с номером 29. То есть при нажатии любой клавиши, в том числе и BREAK, будет выполнен возврат на строку 4. При этом происходит следующее. Очищается экран и в этот момент содержимое этого чистого экрана засылается в буфер экрана (в адрес 50000), подготавливая таким образом атрибуты изображения в буфере в случае последующего обмена экрана и буфера, чтобы экран после обмена не выглядел черным. Теперь подготовка завершена и происходят старт программы точно так же, как Вы будете это делать после остановки программы - командой RUN.

По команде RUN происходит старт программы с первой строки, которая переадресует на строку 100. В строках до 100 расположены различные подпрограммы; их назначение будет рассмотрено в процессе описания. В строке 100 присваиваются первоначальные значения переменным, используемым в программе. После этого начинается собственно

выполнение основной части программы.

Строка 1000 является базовой, на нее может быть сделан переход из различных мест программы, поэтому в начале этой строки задается переход на нее же в случае ошибки, которая может возникнуть при работе программы. Далее задается цвет бордюра и происходит окрашивание в цвет бордюра двух системных строк экрана. Они для создания рисунка недоступны. Подпрограмма GO SUB 8 производит переключение символьного набора на тот, который загружен, с графическими образами и русскими буквами.

Строки 1010 и 1020 выводят в верхней части экрана индикаторную строку, в которой отображается текущее состояние цветов чернил и бумаги, а также значение яркости. Индикаторная строка также недоступна для создания рисунка.

Строка 1100 также является базовой. На нее также происходят переходы из различных мест программы. На нее замыкается малый цикл опроса клавишей в который входят опрос курсорных клавишей, клавиши забор, пробела и буквенных клавишей. Поскольку быстродействие Бейсика невелико, такой прием позволил реализовать вполне приемлемое быстродействие при перемещении курсора и печатании символов.

Подпрограмма GO SUB 40 - это контроль положения курсора. При перемещении курсора, при достижении правого края экрана происходит переход на левый край но на следующей строке экрана (строка 40 программы). При достижении левого края - курсор перескакивает на правый край на предыдущую строку экрана (строка 42). При вертикальных перемещениях курсора изменения номера колонки не происходит (строки 44,46).

Строка 1110 выводит на экран курсор. Его изображение закреплено за символом "A" UDG-графики. Цвет курсора определяется теми атрибутами, которые установлены для этого знакоместа. Если там ничего еще не нарисовано, то это белая бумага и черные чернила. Однако, если в этом знакоместе уже напечатан какой либо символ, то цвет курсора будет зависеть от атрибутов этого знакоместа. Кроме того, курсор печатается в режиме OVER 1, поэтому он накладывается на символ, находящийся в этом знакоместе. Вид курсора будет меняться в зависимости от включенного режима инверсии, а также от параметра мигания. Забегая вперед, скажу, что мигание курсора свидетельствует о переключении из режима "графика" в режим "текст". Или иными словами, это включение режима CAPS LOCK.

В строке 1200 происходит ожидание нажатия клавиши. Так как это может быть клавиша BREAK, то на этот случай задается переход на строку 2000, где происходит вывод на экран инструкции. Это делается так. В строке 2000 устанавливается белый цвет бордюра и двух нижних строк экрана, затем (2010) при помощи кодового блока происходит обмен содержимого экрана и буфера инструкции, затем звуковой сигнал и пауза для ожидания нажатия любой клавиши (2020). Но непосредственно перед паузой - установка блока ON ERROR GO TO на следующую после паузы строку. Теперь при нажатии на любую клавишу, в том числе и на клавишу BREAK, произойдет переход на строку 2030, то есть будет продолжено выполнение программы. Еще раз происходит обращение к кодовому блоку, в результате чего происходит восстановление рисунка на экране, а инструкция отправляется назад в свой буфер. Далее - звуковой сигнал, восстановление цвета бордюра и двух нижних строк экрана. Затем возврат в то место программы, откуда был сделан переход на 2000-ю строку, то есть на строку 1200.

Если в строке 1200 нажата не клавиша BREAK, то продолжается выполнение программы со строки 1210. Здесь еще раз происходит печать курсора в режиме OVER 1, в результате чего восстанавливается прежнее изображение в этом знакоместе.

Со строки 1300 начинается определение действий в зависимости от нажатой клавиши. Строки 1310-1340 определяют перемещение курсора.

Строка 1350 - удаление символа в текущем знакоместе. Причем в зависимости от того, в каком режиме находится программа: в графическом или в текстовом, забор действует по-разному. В графическом режиме удаляется символ в позиции курсора, а в текстовом - символ слева от курсора, так привычнее при печати текста. Текущий режим "графика" или "текст" определяется по миганию курсора. Если показатель мигания  $f=0$ , то режим "графика", а если  $f=1$ , то режим "текст". В зависимости от этого происходит или нет изменение текущей координаты "x" курсора. Удаление символа происходит путем печати



пробела в позиции с текущими координатами, что сопровождается соответствующим звуком.

Строка 1360 реагирует на нажатие пробела. В этом случае, как и при удалении символа, происходит печать пробела в знакоместе с текущими координатами, что также приводит к удалению прежнего символа, однако независимо от режима (мигающий курсор или нет) происходит увеличение на единицу горизонтальной координаты курсора. Это может оказаться полезно при "расчистке" небольших участков рисунка в режиме "графика", а в режиме "текст" привычно печатается пробел. (Если Вы считаете, что это слишком запутанно, то просто удалите строку 1360.)

Строка 1365 является "жучком", позволяющим остановить программу для внесения каких-либо изменений в процессе отладки. Для остановки надо ввести символ "!" (SYMBOL SHIFT + 1). Чтобы остановка не происходила самопроизвольно в процессе работы при попытке напечатать этот символ, для остановки программа требуется дополнительное условие: чтобы переменные, определяющие цвета чернил и бумаги имели нулевые значения (черные чернила и черная бумага - то есть ситуация, бессмысленная при обычной работе). В случае соблюдения этих условий программа переходит на строку 9999, в которой происходит переключение на символьный набор ПЗУ Спектрума, установка белого бордюра, а также принудительное отключение регистра CAPS LOCK для удобства редактирования.

Строки 1370-1390 нужны для того, чтобы можно было печатать символы, которые обычно можно напечатать в расширенном режиме EXTEND MODE. В символьном наборе КОИ-7 русские буквы "Ш" и "Щ" расположены на месте "[" и "]", а последние набираются в режиме EXT.MODE. Поэтому используется прием, позволяющий вводить эти символы при помощи SYMBOL SHIFT вместо "AND" и "OR" соответственно. Аналогично вводится "Э" вместо "STEP", так как эта буква находится на месте символа "\". (Этот прием, кстати, позаимствован из "АРТСТУДИИ".) Если же Вы будете использовать символьный набор с другим расположением русских букв, то продумайте вопрос набора тех букв, которые не помещаются на клавишах с 26-ю латинскими буквами.

Строка 1400 выполняет печать графических символов и русских букв. Печать сопровождается специфическим звуком, воспроизводимым блоком в кодах. В том случае, если программа находится в режиме "текст", то очевидно, что после печати очередного символа, курсор должен переместиться на следующее знакоместо. При построении рисунка по-видимому такого перемещения не должно быть, так как заранее не известно, в какую сторону должен быть сдвинут курсор для построения рисунка (может быть, вверх или вниз). Поэтому в режиме "графика" перемещения курсора не происходит. Режим определяется по миганию курсора (параметр f). Вопрос о том, должен ли курсор перемещаться или нет является в общем-то спорным, поэтому, если Вы хотите, чтобы курсор всегда после печати перемещался на одну позицию вправо, то в строке 1400 вместо:  $x=x+f$  подставьте:  $x=x+1$ .

Строки, связанные с нажатием клавишей, с 1300 по 1400 заканчиваются все одинаково: закликиванием на строку 1100. Это, так называемый, малый цикл опроса клавишей. Это, как уже говорилось, сделано для ускорения реагирования программы на нажатие основных клавишей. Со строки 1410 начинается расширенный цикл опроса. В конце этих строк нет перехода на базовую строку. Переход происходит а самом конце расширенного цикла: в строке 1999 программа закликивается на строку 1000.

Строка 1410 обеспечивает запись экрана на магнитную ленту при нажатии SYMBOL SHIFT+"S" (на клавише ключевое слово SAVE). Вначале производится ввод имени файла (GO SUB 50). Вы можете не задавать никакого имени, а просто нажать ENTER, тогда файл будет записан с именем "\$". Можно отменить ввод имени и вернуться к работе программы, если при вводе имени нажать "КУРСОР ВНИЗ". (В обычной Бейсик-программе в таком случае происходит остановка программы по ошибке, но в программе с блоком ON ERROR GO TO происходит переход на заданную строку. Строка для перехода по ошибке была задана последний раз в строке 1300, то есть произойдет переход на строку 1000.) После ввода имени выполняется подпрограмма записи GO SUB 80. В этой подпрограмме блок ON ERROR GO TO устанавливается на строку 60. Это позволит Вам прервать в случае необходимости запись и вернуться к работе через подпрограмму с адреса 60. Это вывод на экран таблички

"ОШИБКА" и звуковой сигнал.

RETURN в конце этой подпрограммы вернет на строку 1420.

Строка 1420 обеспечивает загрузку экрана с магнитной ленты при нажатии SYMBOL SHIFT + "I" (ключевое слово INPUT). Привычнее, наверное, был бы вариант с клавишей "L", или "J" (ключевое слово LOAD), однако эти варианты соответствуют символам, печатаемым в текстовом режиме: "=" и "-". Если все же Вы считаете, что клавиша "I" неудобна, то можно попробовать компромиссный вариант. Вместо 1420 надо будет новую строку расположить внутри малого цикла опроса клавишей:

```
1395 IF i$="=" AND f=0 THEN GO SUB 50: GO SUB 70: GO TO 1000
```

При этом, правда, загрузка будет возможна только в режиме "графика". Зато в режиме "текст" сохранится возможность печатать "=". Выбирайте, что удобнее.

Строка 1430 позволяет сохранять экран в специальном буфере (с адреса 50000). Переброска производится при помощи машиннокодového блока. В процессе работы над рисунком полезно периодически делать запоминание экрана. Строка 1440 производит противоположное действие - вызов экрана из памяти. Например, после каких-то изменений в рисунке, захочется восстановить предыдущий вариант. Строка 1450 производит обмен экрана и буфера без потери изображения обоих. Есть возможность сравнить два варианта рисунка и выбрать лучший.

Строка 1460 выполняет очистку экрана при нажатии SYMBOL SHIFT + "A" (ключевое слово STOP).

Строки 1470-1520 позволяют изменять яркость, инверсию, переключать режим "графика" - "текст", изменять цвет чернил, бумаги и бордюра. Для наглядности выражение типа  $b=b=0$  можно было бы записать так:  $b=(b=0)$ . То, что стоит в скобках, является логическим выражением, оно равно единице, если  $b=0$ . Таким способом достигается переключательный эффект изменения яркости, инверсии и режима "графика" - "текст" при нажатии соответствующих клавиш. При переключении цвета чернил, бумаги и бордюра ситуация похожа. При нажатии на клавишу, цвет (например чернил - 1) изменяется от 0 до 7, а при следующем нажатии ( $i=8$ ) логическое выражение ( $i<7$ ) станет равно 0, что вызывает повторение цикла изменения цвета, начиная с  $i=0$ .

Строка 1530 воспроизводит "музыкальное сопровождение". Циклическое проигрывание мелодии происходит до тех пор, пока не будет нажата какая-нибудь клавиша. Поэтому, если чуть-чуть задержать руку на клавише ENTER, то мелодия прервется на первом же звуке. Для устранения этого недостатка введен замедляющий пустой цикл FOR-NEXT (в отличие от PAUSE, которая в данном случае не поможет, так как прекращается как раз при нажатии клавиши).

Строка 1999 завершает цикл опроса клавишей, передавая управление на базовую строку 1000. С этого момента работа программы повторяется.

Теперь, когда мы покончили с Бейсиком, можно переходить к конкретным действиям по набору программы. Все коды, которые требуются для работы "МОЗАИКИ", (то есть то, что расположено с адреса 63824 и входит в файл "mozaik" CODE), приведены ниже. Число в конце каждой строки после двоеточия - это контрольная сумма строки. Она поможет Вам при наборе этого блока кодов.

```
F950:00 00 00 00 00 00 00 00 :49
F958:00 00 10 10 10 00 10 00 :91
F960:00 00 28 28 00 00 00 00 :A9
F968:00 00 28 7C 28 7C 28 00 :D1
F970:00 10 3C 50 38 14 78 10 :D9
F978:00 00 64 68 10 2C 4C 00 :C5
F980:00 10 28 10 34 48 34 00 :71
F988:00 00 08 10 00 00 00 00 :99
F990:00 00 08 10 10 10 08 00 :C9
F998:00 00 20 10 10 10 20 00 :01
F9A0:00 00 28 10 7C 10 28 00 :85
F9A8:00 00 10 10 7C 10 10 00 :5D
F9B0:00 00 00 00 00 10 10 20 :E9
F9B8:00 00 00 00 7C 00 00 00 :2D
```

```
F9C0:00 00 00 00 00 30 30 00 :19
F9C8:00 00 04 08 10 20 40 00 :3D
F9D0:00 00 38 4C 54 64 38 00 :3D
F9D8:00 00 10 30 10 10 38 00 :69
F9E0:00 00 38 04 18 20 3C 00 :89
F9E8:00 00 38 04 18 04 38 00 :71
F9F0:00 00 18 28 48 7C 08 00 :F5
F9F8:00 00 38 20 38 04 38 00 :BD
FA00:00 00 38 40 78 44 38 00 :66
FA08:00 00 3C 04 08 10 10 00 :6A
FA10:00 00 38 44 38 44 38 00 :3A
FA18:00 00 38 44 3C 04 38 00 :06
FA20:00 00 00 10 00 00 10 00 :3A
FA28:00 00 00 10 00 00 10 20 :62
```

FA30:00 00 08 10 20 10 08 00 :7A  
FA38:00 00 00 7C 00 7C 00 00 :2A  
FA40:00 00 20 10 08 10 20 00 :A2  
FA48:00 00 18 24 08 00 08 00 :8E  
FA50:00 00 4C 52 72 52 4C 00 :F8  
FA58:00 00 38 44 7C 44 44 00 :D2  
FA60:00 00 78 40 78 44 78 00 :46  
FA68:00 00 48 48 48 48 7C 04 :02  
FA70:00 00 1C 24 24 24 7E 42 :B2  
FA78:00 00 7C 40 78 40 7C 00 :62  
FA80:00 00 7C 54 54 7C 10 00 :2A  
FA88:00 00 3C 20 20 20 20 00 :3E  
FA90:00 00 44 28 10 28 44 00 :72  
FA98:00 00 44 4C 54 64 44 00 :1E  
FAA0:00 10 44 4C 54 64 44 00 :36  
FAA8:00 00 44 48 70 48 44 00 :2A  
FAB0:00 00 1C 24 24 24 44 00 :76  
FAB8:00 00 44 6C 54 44 44 00 :3E  
FAC0:00 00 44 44 7C 44 44 00 :46  
FAC8:00 00 38 44 44 44 38 00 :FE  
FAD0:00 00 7C 44 44 44 44 00 :56  
FAD8:00 00 3C 44 3C 24 44 00 :F6  
FAE0:00 00 78 44 44 78 40 00 :92  
FAE8:00 00 38 44 40 44 38 00 :1A  
FAF0:00 00 7C 10 10 10 10 00 :A6  
FAF8:00 00 44 44 3C 04 38 00 :F2  
FB00:00 00 54 54 38 54 54 00 :83  
FB08:00 00 78 44 78 44 78 00 :F3  
FB10:00 00 40 40 78 44 78 00 :BF  
FB18:00 00 42 42 72 4A 72 00 :C5  
FB20:00 00 38 44 08 44 38 00 :1B  
FB28:00 00 44 54 54 54 7C 00 :DF  
FB30:00 00 38 04 1C 04 38 00 :BF  
FB38:00 00 44 54 54 54 7E 02 :F3  
FB40:00 00 44 44 3C 04 04 00 :07  
FB48:00 00 60 20 38 24 38 00 :57  
FB50:00 00 18 24 70 20 7C 00 :93  
FB58:FF FF FF 7F 7F 3F 1F 07 :B3  
FB60:00 00 00 FF FF 00 00 00 :59  
FB68:AA 55 AA 55 AA 55 AA 55 :5F  
FB70:FF 7F 3F 1F 0F 07 03 01 :61  
FD78:01 01 07 0F 1F 3F 7F FF :69  
FB80:FF FE FC F8 F0 E0 C0 80 :7C  
FB88:03 07 0F 0F 0F 0F 07 03 :D3  
FB90:FF FF 7E 3C 00 00 00 00 :43  
FB98:00 00 00 F8 F8 18 18 18 :CB  
FBA0:18 18 18 1F 1F 00 00 00 :21  
FBA8:18 18 18 F8 F8 00 00 00 :DB  
FBB0:18 18 18 FF FF 00 00 00 :F1  
FBB8:18 18 18 F8 F8 18 18 18 :33  
FBC0:18 18 18 1F 1F 18 18 18 :89  
FBC8:00 00 00 FF FF 18 18 18 :09  
FBD0:18 18 18 FF FF 18 18 18 :59  
FBD8:07 1F 3F 7F 7F FF FF FF :33  
FBE0:80 C0 E0 F0 F8 FC FE FF :DC  
FBE8:FF FF FF FE FE FC F8 E0 :B0  
FBF0:00 00 00 00 3C 7E FF FF :A3  
FBF8:00 00 00 1F 1F 18 18 18 :79  
FC00:18 18 18 18 18 18 18 18 :BC  
FC08:E0 F8 FC FE FE FF FF FF :D1  
FC10:FF FF FF FF FF FF FF FF :04  
FC18:C0 E0 F0 F0 F0 F0 E0 C0 :14  
FC20:3C 7E FF FF FF FF 7E 3C :8C  
FC28:00 0E 08 30 08 08 0E 00 :88  
FC30:00 08 08 08 08 08 08 00 :5C

FC38:00 70 10 0C 10 10 70 00 :50  
FC40:00 14 28 00 00 00 00 00 :78  
FC48:3C 42 99 A1 A1 99 42 3C :B4  
FC50:FF 81 81 99 99 81 81 FF :80  
FC58:00 7C 42 7C 42 42 7C 00 :8E  
FC60:00 3C 42 40 40 42 3C 00 :D8  
FC68:00 78 44 42 42 44 78 00 :60  
FC70:00 7E 40 7C 40 40 7E 00 :A4  
FC78:00 7E 40 7C 40 40 40 00 :6E  
FC80:00 3C 42 40 4E 42 3C 00 :06  
FC88:00 42 42 7E 42 42 42 00 :4C  
FC90:00 3E 08 08 08 08 3E 00 :28  
FC98:00 02 02 02 42 42 3C 00 :5A  
FCA0:00 44 48 70 48 44 42 00 :66  
FCA8:00 40 40 40 40 40 7E 00 :62  
FCB0:00 42 66 5A 42 42 42 00 :74  
FCB8:00 42 62 52 4A 46 42 00 :7C  
FCC0:00 3C 42 42 42 42 3C 00 :3C  
FCC8:00 7C 42 42 7C 40 40 00 :C0  
FCD0:00 3C 42 42 52 4A 3C 00 :64  
FCD8:00 7C 42 42 7C 44 42 00 :D6  
FCE0:00 3C 40 3C 02 42 3C 00 :14  
FCE8:00 FE 10 10 10 10 10 00 :32  
FCF0:00 42 42 42 42 42 3C 00 :72  
FCF8:CD 7C 00 3B 3B E1 01 0F :A4  
FD00:00 09 EB 2A 3D 5C 73 23 :4A  
FD08:72 C9 3B 3B CD 8E 02 7B :8E  
FD10:FE FF 20 F8 3A 3A 5C FE :F0  
FD18:FF 28 21 FE 07 28 1D FE :A5  
FD20:08 28 19 3C 32 81 5C FD :AE  
FD28:36 00 FF 21 D0 07 22 42 :B6  
FD30:5C AF 32 44 5C FD CB 01 :D3  
FD38:FE C3 7D 1B 33 33 C3 03 :BA  
FD40:13 0E 01 06 28 21 64 00 :12  
FD48:C5 11 01 00 E5 CD B5 03 :86  
FD50:E1 11 1E 00 ED 5A C1 10 :75  
FD58:EF 3E 02 0C 41 B8 20 E3 :8C  
FD60:C9 0E 01 06 28 21 E8 03 :6F  
FD68:C5 11 01 00 E5 CD B5 03 :A6  
FD70:E1 11 18 00 ED 52 C1 10 :87  
FD78:EF 3E 02 0C 41 B8 20 E3 :AC  
FD80:C9 21 00 40 01 00 1B 11 :D4  
FD88:50 C3 ED B0 C9 21 50 C3 :32  
FD90:01 00 1B 11 00 40 ED B0 :97  
FD98:C9 11 50 C3 21 00 40 06 :E9  
FDA0:1B C5 06 00 7E F5 1A 77 :87  
FDA8:F1 12 23 13 10 F6 C1 10 :B5  
FDB0:F0 C9 21 00 40 01 00 1B :E3  
FDB8:11 50 DE ED B0 C9 11 50 :BB  
FDC0:DE 21 00 40 06 1B C5 06 :E8  
FDC8:00 7E F5 1A 77 F1 12 23 :EF  
FDD0:13 10 F6 C1 10 F0 C9 21 :91  
FDD8:E0 FE 22 F2 FD 21 62 FF :46  
FDE0:22 F6 FD F3 CD 1E FE CD :9B  
FDE8:8E 02 1C 28 F7 FB C9 0A :7E  
PDF0:12 07 EF FE E1 FE 71 FF :42  
PDF8:63 FF EB 5E 23 56 13 1A :46  
FE00:FE 40 28 12 72 2B 73 C9 :4F  
FE08:7E C6 0C 5F 16 00 21 AB :97  
FE10:FE 19 66 2E 01 C9 23 5E :04  
FE18:23 56 2B 2B 18 E1 21 F2 :F1  
FE20:FD CD FB FD 32 EF FD 21 :1F  
FE28:F6 FD CD FB FD 32 F0 FD :FD  
FE30:21 EF FD CD 08 FE CB 13 :EC  
FE38:DA E1 FE E5 21 F0 FD CD :AF

|         |                          |         |                          |
|---------|--------------------------|---------|--------------------------|
| FE40:08 | FE D1 7C 3D 20 04 7A :6C | FF18:29 | 0B 29 06 29 0A 29 0A :E0 |
| FE48:3D | 28 42 3A FA FD 4F 06 :73 | FF20:29 | 29 29 1E 29 12 29 1B :37 |
| FE50:00 | 3A F1 FD 08 3A F1 FD :A6 | FF28:29 | 12 29 19 29 11 29 11 :18 |
| FE58:DD | 62 16 10 00 00 08 1D :E0 | FF30:29 | 29 29 01 29 0B 29 03 :0B |
| FE60:D3 | FE 20 17 DD 5C AA 08 :51 | FF38:29 | 0B 29 06 29 0A 29 0A :00 |
| FE68:2D | C2 82 FE D3 FE 6C AA :BC | FF40:29 | 29 29 1E 29 12 29 1B :57 |
| FE70:10 | EA 0C C2 5E FE C9 61 :BC | FF48:29 | 12 29 19 29 11 29 11 :38 |
| FE78:64 | 61 6D 28 FE 08 2D CA :CD | FF50:29 | 29 29 01 29 0B 29 03 :2B |
| FE80:6C | FE D3 FE 00 00 10 D4 :9D | FF58:29 | 0B 29 06 29 0A 29 0A :20 |
| FE88:0C | C2 5E FE C9 3A FA FD :AA | FF60:29 | 40 00 29 29 29 29 12 :7E |
| FE90:2F | 4F C5 F5 06 00 E5 21 :D2 | FF68:29 | 12 29 29 29 29 29 12 :81 |
| FE98:00 | 00 CB 2E CB 2E CB 2E :81 | FF70:29 | 12 29 29 29 29 29 12 :89 |
| FEA0:00 | E1 10 F2 0D C2 96 FE :E4 | FF78:29 | 29 29 12 29 29 29 11 :90 |
| FEA8:F1 | C1 C9 FF F0 E3 D7 CB :95 | FF80:29 | 11 29 29 29 29 29 11 :97 |
| FEB0:C0 | B4 AB A1 97 90 88 80 :9D | FF88:29 | 11 29 29 29 29 29 11 :9F |
| FEB8:79 | 72 6C 66 60 5B 56 51 :D5 | FF90:29 | 11 29 29 29 29 29 11 :A7 |
| FEC0:4C | 48 44 40 3D 39 36 33 :B5 | FF98:29 | 29 29 11 29 29 29 12 :B0 |
| FEC8:30 | 2D 2B 28 26 24 22 20 :02 | FFA0:29 | 12 29 29 29 29 29 0A :B1 |
| FED0:1E | 1C 1B 19 18 17 15 14 :94 | FFA8:29 | 29 29 0A 29 29 29 0B :B2 |
| FED8:13 | 12 11 10 0F 0E 0D 0C :52 | FFB0:29 | 0B 29 29 29 29 29 11 :C1 |
| FEE0:01 | 0F 0D 06 29 0A 29 0A :67 | FFB8:29 | 29 29 11 29 29 29 12 :D0 |
| FEE8:29 | 0F 0D 06 29 0A 29 0A :97 | FFC0:29 | 12 29 29 29 29 29 0A :D1 |
| FEF0:29 | 0F 0D 06 29 0A 29 03 :98 | FFC8:29 | 29 29 0A 29 29 29 0B :D2 |
| FEF8:29 | 0A 29 01 29 0B 29 0B :BB | FFD0:29 | 0B 29 29 29 29 29 11 :E1 |
| FF00:29 | 0F 0D 01 29 0B 29 0B :AD | FFD8:29 | 29 29 11 29 29 29 12 :F0 |
| FF08:29 | 0F 0D 01 29 0B 29 0B :B5 | FFE0:29 | 12 29 40 00 00 00 00 :83 |
| FF10:29 | 0F 0D 01 29 0B 29 03 :B5 |         |                          |

Для набора кодового файла "mozaik" CODE 63824,1684 можно использовать любую программу-монитор, которая есть под рукой. Однако для того, чтобы избежать ошибок, которые обязательно возникнут при таком количестве набираемых цифр, удобно будет воспользоваться программой для ввода, опубликованной в "ZX-РЕВЮ" N 3 за 1991 г. на стр. 59 в статье "ZX-MODEM", оформив числовые данные в виде строк DATA. Учитывая то, что не все имеют возможность ознакомиться с этой статьей, мы рекомендуем Вам воспользоваться несколько измененным вариантом программы для ввода кодов. В приведенном ниже варианте данные вводятся непосредственно при помощи INPUT. (Возможно, так удобнее). Кроме того, периодически, вводя "S", Вы имеете возможность сохранить на ленте то, что уже введено. Приведенные выше коды программы "МОЗАИКА" напечатаны вразрядку, да еще с разделителями ":". Это сделано для удобочитаемости данных. Вам при наборе надо будет вводить их подряд, например для первой строки:

```
F95000000000000000000049 [ENTER]
```

Никаких пробелов или двоеточий вводить не надо. Ошибки, допущенные при вводе, будут отмечены звуковым сигналом, после чего Вам будет предложено повторить ввод.

```
1 GO TO 100
2 LOAD "mozaik" CODE
4 GO TO 0
5 SAVE "INPUT" LINE 2: STOP
100 BORDER 7: PAPER 7: INK 0: CLEAR 29999
110 DIM a(10)
120 DEF FN A(a$)=(CODE a$(1)-48-(7 AND a$(1)"9"))*16+(CODE a$(2)-48-(7 AND a$(2)"9"))
1000 POKE 23658,8: INPUT "DATA:"; LINE b$
1010 IF h$ = "S" THEN GO TO 2000
1020 LET a$=h$
1030 LET sum=0
1040 FOR i=1 TO 2
1050 LET b$=a$(2*i-1 TO 2*i)
1060 LET a(i)=FN A(b$)
1070 NEXT i
1080 LET add=a(1)*256+a(2)
1090 LET sum=a(1)+a(2)
1100 FOR i=3 TO 10
```

```

1110 LET b$=h$(2*i-1 TO 2*i)
1120 LET a(i)=FN A(b$)
1130 LET sum=sum+a(i)
1140 POKE add, a(i)
1150 LET add=add+1
1160 NEXT i
1170 LET b$=h$(21 TO )
1180 LET cs=FN A(b$)
1190 LET cs1=sum-256*INT (sum/256)
1200 IF cs<>cs1 THEN PRINT a$; INVERSE 1;" ERROR ! ": BEEP 1,0:LET add=add-8: GO TO 1000
1210 PRINT a$( TO 4): GO TO 1000
2000 SAVE "mozaik" CODE 63824,1684
2010 VERIFY "mozaik"CODE
2020 CLS : PRINT a$( TO 4): GO TO 1000

```

Набрав эту Бейсик-программу, сделайте RUN 5 для записи ее на магнитную ленту (автостарт со 2-й строки обеспечит в дальнейшем автоматическую подгрузку уже частично набранного кодового файла). После этого запустите программу: RUN и начинайте ввод кодового файла "mozaik" CODE. При перерывах в работе сохраняйте его на магнитной ленте, вводя "S".

Теперь последовательность, в которой следует набирать всю программу "МОЗАИКА". Сначала надо заготовить картинку-заставку при помощи графического редактора. Затем набрать приведенный выше кодовый файл. Теперь можно приступить к набору Бейсик-файла. Набирая 10 строку, подставьте в ее начало RETURN: это отключит пока блок кодов ON ERROR GO TO, блокирующий остановку программы. В строке 30 в кавычках набрано 32 пробела. Набирая текст на русском языке, учитывайте, какой русский символьный набор будет применяться в Вашей программе. Можно рекомендовать такой подход. Загрузите уже набранный кодовый файл "mozaik" CODE, затем, непосредственно перед набором русского текста, переключите символьный набор командой GO SUB 8, теперь Вы будете видеть набираемый текст таким, каким он будет на экране. Обратное переключение символьного набора - GO SUB 9. Обратите внимание: в строках 1110 и 1210 в кавычках вводится символ "А", используя графический регистр. В строках 1370-1390 а также 1410-1460 в кавычках вводятся токены ключевых слов.

После того, как набор Бейсик-файла будет закончен, сделайте RUN 2 и загрузите заготовленные заранее кодовые файлы. После этого программа начнет работать. После того, как Вы убедитесь, что ошибок нет или устраните имеющиеся, можно убрать RETURN из строки 10. Запустите программу: RUN и посмотрите, как она будет вести себя при нажатии клавиши BREAK. Для остановки программы используйте "жучок" в строке 1365: как говорилось выше, если установить чёрный цвет чернил и черный цвет бумаги, то программу можно остановить, введя восклицательный знак (SYMB.SHIFT+1). Затем можно выгрузить полностью готовую программу на магнитную ленту, выполнив RUN 5.

Теперь пришло время несколько слов сказать о том, как адаптировать программу под "БЕТА-ДИСК интерфейс". Прежде всего, надо изменить строки 2 и 3 Бейсик-файла, добавив перед ключевым словом "LOAD" префикс:

```
RANDOMIZE USR 15619: REM :
```

Кроме того, надо будет изменить подпрограммы, связанные с загрузкой и выгрузкой экранов, начинающиеся со строк 70 и 80. БЕТА-вариант этих подпрограмм:

```

70 LET err=USR 15619 : REM : LOAD f$ CODE 16384
72 IF err<>0 THEN GO SUB 60
79 RETURN
80 RANDOMIZE USR 15619: REM : ERASE f$ CODE
82 LET err=USR 15619: REM : SAVE f$ CODE 16384,6912
84 IF err<>0 THEN GO SUB 60
89 RETURN

```

В подпрограмме записи файла сначала производится удаление предыдущей версии файла на диске с тем именем, которое было задано, а затем происходит запись экрана в новый файл с таким именем. В случае ошибки чтения-записи происходит вызов

подпрограммы GO SUB 60, которая выводит предупредительную табличку. Для простоты программы не предусмотрена верификация. Вы можете сами организовать ее, если захотите. Кроме того, если предварительное стирание файла с заданным именем Вас не устраивает, то просто исключите строку 80. В результате, в том случае, если на диске уже имеется файл с таким именем, то записи экрана на диск не произойдет, а будет выдано сообщение об ошибке подпрограммой GO SUB 60.

В строке 52 вместо "макс. 10 символов" подставьте: "макс. 8 символов". В строке 54 удалите все, начиная с IF ... до конца строки.

На этом адаптацию под диск можно считать законченной.

А теперь дополнительная информация для тех, кто занят творческим поиском, а не просто копирует готовую программу.

Нам почему-то гораздо симпатичнее вариант, когда все блоки кодов, используемые в программах, находятся в нулевой строке, в области за оператором REM. В частности, тогда полностью исключается несанкционированная остановка программы, так как блок кодов ON ERROR GO TO, располагаясь внутри Бейсик программы, может быть инициирован сразу же после старта программы. Это обстоятельство, дополненное грамотно продуманными приемами защиты, создаст больше хлопот для взломщиков. Программа "МОЗАИКА", конечно, не претендует на такую серьезную защиту, но речь сейчас о том, какие методы Вы можете использовать в своих разработках, которые, возможно, как раз потребуют хорошей защиты. Еще одно обстоятельство в пользу кодов в нулевой строке. Такая программа гораздо быстрее загружается (это относится как к магнитофонному, так и к дисковому вариантам). А это уже существенное преимущество для пользователя. Все определяется только тем, "стоит ли овчинка выделки", то есть какими трудами это дается программисту. Попробуем показать, что в нашем случае эти дополнительные эти труды совсем незначительны.

При разработке "МОЗАИКИ" мы исходили из того, что программа должна быть легко адаптируема под "БЕТА-ДИСК интерфейс". О путях решения этой проблемы уже говорилось в "ZX-РЕВЮ" N 9-10 за 1992 г. стр. 197. Но там была ситуация, когда все блоки кодов в нулевой строке были перемещаемыми в памяти. Проблема была только в том, чтобы обеспечить правильное обращение к ним из Бейсика. В программе "МОЗАИКА" используется неперемещаемый блок кодов, воспроизводящий мелодию! Если даже его скомпилировать при помощи музыкального редактора "WHAM" для расположения в нулевой строке магнитофонного варианта, тогда при адаптации под "БЕТА-ДИСК" он окажется смещенным на 112 байт и не будет работать.

Для универсальности программы применяется следующий прием. После загрузки программы все рабочие блоки кодов из нулевой строки перебрасываются в отведенное для них место (с адреса 63824) и работают уже там. (Программа "МОЗАИКА" небольшая, с количеством свободной памяти проблем нет.)

В начале нулевой Бейсик-строки находится процедура, осуществляющая переброску кодового блока заданной длины, следующего непосредственно за этой процедурой, на заданный адрес. Этот способ Вы можете использовать и в других своих программах. Независимо от того, где располагается сама процедура, адрес, откуда будет произведена переброска, всегда берется следующим за концом процедуры. Вот она:

|             |                             |           |     |
|-------------|-----------------------------|-----------|-----|
| 5CD0 CD7C00 | CALL                        | #007C     | (1) |
| 5CD3 3B     | DEC                         | SP        | (2) |
| 5CD4 3B     | DEC                         | SP        |     |
| 5CD5 E1     | POP                         | HL        |     |
| 5CD6 011000 | LD                          | BC, #0010 | (3) |
| 5CD9 09     | ADD                         | HL, BC    |     |
| 5CDA 1150F9 | LD                          | DE, #F950 | (4) |
| 5CDD 019606 | LD                          | BC, #0696 |     |
| 5CE0 EDB0   | LDIR                        |           | (5) |
| 5CE2 C9     | RET                         |           |     |
| 5CE3        | .... перебрасываемый массив |           |     |

Процедура имеет длину 19 байт и расположена в начале нулевой строки сразу же за REM, адрес в случае магнитофонного варианта равен 23760 (#5CD0). Но с одинаковым

успехом она будет работать и в любом другом месте. Кодовый массив, подлежащий переброске, располагается сразу же за процедурой. Для определения этого адреса используется такой прием. Сначала выполняется подпрограмма из ПЗУ (1). По адресу #007C в ПЗУ находится только одна команда RET. Но при выполнении инструкции CALL на стек записывается адрес команды, следующей за CALL, то есть в данном случае #5CD3. Для того, чтобы осуществить привязку к адресам, это значение возвращается (2) в регистр HL. Далее это значение увеличивается на величину смещения, заданного в регистре BC (3). Теперь в HL получилось: #5CD3 + #0010 = #5CE3. Затем задаются остальные параметры для выполнения команды LDIR (4). В DE - адрес места назначения перебрасываемого блока: 63324, а в BC - его длина: 1684 байта. (Конкретные значения подставлены те, которые используются в программе "МОЗАИКА".) Теперь выполняется команда LDIR (5) и происходит возврат в вызывающую программу.

В нулевой строке "МОЗАИКА" расположены следующие блоки кодов:

1. Процедура, осуществляющая переброску рабочих кодов в адрес 63824. Длина - 19 байт.

2. Рабочие коды (перебрасываемые в адрес 63824). Длина - 1684 байта.

Суммарная длина кодов, располагаемых в нулевой строке, равна: 19+1684=1703 байта.

При создании нулевой строки с большим объемом памяти после REM, Вы можете воспользоваться программой REM FILL, ранее опубликованной в "ZX-РЕВЮ" N 9-10 за 1992 г. на стр. 194-196. Сегодня же предложим новую версию этой программы. Она называется "REM 2". Кодовый блок этой версии имеет длину всего 62 байта. Это оказалось возможным осуществить благодаря использованию подпрограммы из ПЗУ, расположенной по адресу 5717, которая позволяет зарезервировать заданный объем памяти, раздвинуть текст Бейсик-строк и перерасчитать все системные переменные. Она используется процедурами редактора Спектрума при добавлении новых строк в программу. (О ней, в частности, упоминалось в "ZX-РЕВЮ" N 5-6 в статье Пашорина В.И. на стр. 112.) Программа "REM 2" состоит только из Бейсик-файла (кодовый блок формируется после старта Бейсик-программы):

```
1 REM
10 LET n=23296: LET s=0
20 FOR x=n TO n+61
30 READ y: POKE x,y: LET s=s+y
40 NEXT x
50 IF S<>6108 THEN PRINT FLASH 1; "ERROR IN LINE DATA": STOP
60 INPUT "No of extra bytes: "; n
70 RANDOMIZE n
80 POKE 23312,PEEK 23670: POKE 23313,PEEK 23671
100 DATA 042,083,092,229,054,000,035,054,000,035,094,035,086,213,025,001,008,000,197,205,
    085,022,035,229,209,019,054
110 DATA 000,193,197,011,237,176,209,225,025,235,225,035,035,115,035,114
120 DATA 000,033,001,000,205,110,025,229,033,016,039,205,110,025,209,205,229,025,201
200 CLEAR : RANDOMIZE USR 23296
```

Подробно принцип работы этой программы был изложен в указанной выше статье в "РЕВЮ" N 9-10. Поэтому, чтобы не повторяться скажем только, что после старта программа запрашивает число дополнительных байтов, которые будут вставлены между последним символом текста начальной строки и символом <ENTER>, завершающим эту строку. После ввода этого параметра происходит старт кодового блока. После сообщения 0: OK, область за REM в 1-й строке будет расширена на число добавочных байт и обнулена, номер этой строки станет 0, а все остальные Бейсик-строки будут уничтожены. Теперь в полученную REM область можно загрузить коды.

Первое значение DATA в строке 110 определяет код заполнения дополнительной области. Если в строке 120 первое значение DATA заменить на 201, то уничтожения остальных Бейсик-строк не будет. Экспериментируя с указанными значениями, удалите строку 50, которая проверяет контрольную сумму значений DATA, защищая от ошибок при наборе программы.

Теперь возвращаемся к "МОЗАИКЕ". На вопрос о числе добавочных байтов ответьте:

1703. После того, как нулевая строка с областью заданной длины после REM будет сформирована, можете загружать в нее коды и объединить ее при помощи MERGE с Бейсик-файлом "МОЗАИКИ". Если Вы используете БЕТА-ДИСК, то на всякий случай загляните в статью о TR-DOS в этом выпуске "РЕВЮ". Там автор упомянул об особенностях загрузки кодов в нулевую строку в отличие от магнитофонного варианта.

Теперь осталось изменить строки 2 и 3 Бейсик-файла "МОЗАИКА":

```
2 BORDER 7: PAPER 7: INK 0: CLEAR 49999: POKE 23739,111: RANDOMIZE USR (PEEK 23635+256*PEEK  
23636+5)
```

```
3 RANDOMIZE 4: GO SUB 10: LOAD "mozaik $"CODE 16354
```

В заключение же хочется пожелать всем юным пользователям "Спектрума" приятной работы с программой "МОЗАИКА".



# ПРИМЕНЕНИЕ АССЕМБЛЕРА ДЛЯ СОЗДАНИЯ БЫСТРОРАБОТАЮЩИХ ПРОГРАММ

## ПРЕДИСЛОВИЕ

Известные книги, справочники и руководства по машинному языку процессора Z-80 используемого в компьютере "ZX-Spectrum" в основном ограничиваются описанием действий всех команд процессора и порядком применения этих команд на простейших примерах программ в машинных кодах. В лучшем случае приводятся программы для сложения чисел или им подобные.

Практика показывает, что конкретному пользователю, желавшему начать освоение машинного кода нужен немного другой подход. Нужна книга, которая поможет ему немедленно, сразу взяться за дело и почувствовать мгновенную отдачу от своих усилий. Одним словом, нужна простейшая практика и живая заинтересованность.

Как нам кажется, в основу этой книги положена весьма плодотворная идея - опереться на знание пользователем БЕЙСИКа и постепенно развивать эти знания в направлении машинного кода, шаг за шагом показывая как алгоритмы БЕЙСИКа переносятся на язык АССЕМБЛЕРА. Эта концепция очень хорошо укладывается в тот стиль работы, который "ИНФОРКОМ" ведет уже третий год на страницах "ZX-РЕВЮ" и в прочих своих книгах.

Нам также показалась очень привлекательной идея книги дать широкое представление конкретному программисту о возможности использования процедур, "зашитых" в ПЗУ для своей повседневной работы. Это может заинтересовать и тех, кто достаточно хорошо знает "АССЕМБЛЕР, но не имеет доступной информации об использовании процедур ПЗУ в своих программах.

Мы горячо благодарим нашего постоянного соавтора из г. Балашова Саратовской обл. В. Пашорина за прекрасный перевод этой нужной книги и, главное, за достойный выбор самой книги для перевода.

К сожалению, мы не могли бы сказать, что оригинал написан простым доступным языком, понятным каждому неспециалисту. Но мы очень хорошо потрудились над ее адаптацией. Оставив по сути без изменения структуру книги и ее фактическое содержание, мы полностью переработали всю методику подачи материала и надеемся, что сделали ее доступной для неподготовленного читателя, впрочем о том, как это удалось, судить будете Вы сами.

Поскольку сама книга фактически не является учебником по машинному кодированию, а служит практическим пособием для тех, кто пошел по этому пути, мы должны напомнить Вам о целесообразности приобретения нашего издания "Программирование в машинных кодах. Первые шаги, практикум. Справочник". По этой книге уже обучились тысячи пользователей и она пока продолжает оставаться лучшим из всего, что было написано на данную тему.

Для тех наших читателей, которые сейчас решают вполне естественный вопрос о целесообразности приобретения всего комплекта "ZX-РЕВЮ-93", мы даем содержание книги, которая войдет в "РЕВЮ-93" полностью:

1. Вывод на экран.
2. Команды PLOT, DRAW, CIRCLE.
3. Счет очков в программах.
4. Случайные числа.
5. Опрос клавиатуры.
6. Перемещение объектов.
7. Музыкальные и звуковые эффекты.
8. Команды ATTR, SCREEN\$, POINT.

9. Работа с принтером.

10. Разбор конкретного примера перевода игровой программы из БЕЙСИКа в машинный код.

## 1. ВЫВОД НА ЭКРАН

Создание программ в машинных кодах, выполняющих вывод информации на экран - утомительная задача, требующая значительных затрат рабочего времени. Тем не менее, это одна из наиболее важных задач. Даже очень хорошие программы проиграют многое, если информация, выводимая на экран при их выполнении, не вполне понятна или, хуже того, допускает неоднозначную трактовку. Перед созданием любой программы необходимо подумать о том, чтобы наглядность выводимой на экран информации была обеспечена на должном уровне.

На компьютере "ZX\_SPECTRUM" можно создать 21 графический элемент, форма которых полностью определяется пользователем. Это так называемые символы графики пользователя (User Definable Graphics - UDG). Необходимо как можно полнее использовать эти символы в своих программах при выводе информации на экран. Например, при создании игровой программы типа "космических завоевателей" для получения отдельных фрагментов изображения экрана можно использовать те же символы, что используются для создания самих "завоевателей" или других объектов.

Для вывода информации на экран целесообразно использовать контрастные цвета. Не рекомендуется использовать вместе красный и розовый, желтый и белый и т.п. Благоразумно в программах использовать только черный и белый цвета, так как не все пользователи имеют цветные мониторы или цветные телевизоры.

Так как выводить информацию на экран необходимо при работе практически любой программы, то мы начнем нашу книгу с того, что рассмотрим в этой главе способы перевода команды БЕЙСИКа PRINT в машинный код.

Обычно БЕЙСИК-программа начинается с того, что определяются основные цветовые атрибуты, например:

```
10 PAPER 3: INK 1: BORDER 5
```

Затем эти атрибуты устанавливаются в системных переменных компьютера и становятся действующими. Делают это командой CLS.

```
20 CLS
```

Давайте попробуем сделать то же самое в машинном коде. Самой простой будет операция установления цвета бордюра. Она займет всего лишь 6 байтов в оперативной памяти.

Сначала в регистр A процессора заносится число 5 (оно означает зеленый цвет бордюра), а затем вызывается процедура ПЗУ, которая выполнит изменение этого цвета. Данная процедура называется BORDER и находится она в ПЗУ по адресу 229BH (8859 DEC).

Процедура не только установит необходимый цвет окантовки экрана, но и передаст значение номера цвета в системную переменную BORDER (23624 DEC - 5C48H). Для хранения номера цвета бордюра в этой системной переменной отведены 3 бита (с третьего по пятый) и, таким образом, всего могут быть установлены до 8 различных цветов. Оставшиеся биты в этой системной переменной используются для хранения цветовых атрибутов нижней части экрана (системного окна). Системное окно занимает как правило (не всегда) две нижние строки экрана, в которых Вы выполняете INPUT или редактирование программы. Биты 0...3 определяют номер цвета INK в системном окне, бит 6 отвечает за параметр BRIGHT, а бит 7 - за параметр FLASH.

Итак, программа в машинных кодах, эквивалентная команде БЕЙСИКа BORDER 5, будет выглядеть так, как показано на листинге 1.1 (Предполагается, что машинный код начинается с адреса, заданного директивой ACCEМБЛЕРа ORG).

Листинг 1.1

| АДРЕС | МАШ. КОД | АССЕМБЛЕР | КОММЕНТАРИЙ                                                      |
|-------|----------|-----------|------------------------------------------------------------------|
|       |          | ORG 23760 |                                                                  |
| 23760 | 3E 05    | LD A, 5   | ; Установили в аккумуляторе число<br>; "5" - код зеленого цвета. |

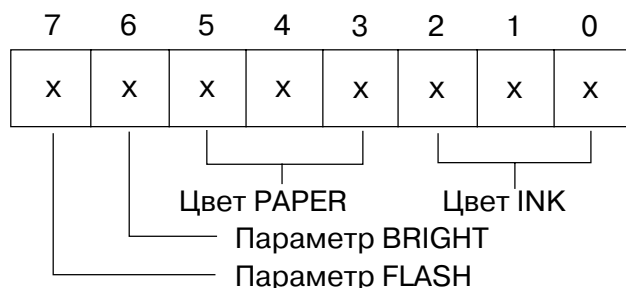
```

23762 CD 9B 22    CALL 8859    ; Вызов процедуры ПЗУ "BORDER".
23765 C9          RET          ; Выход.

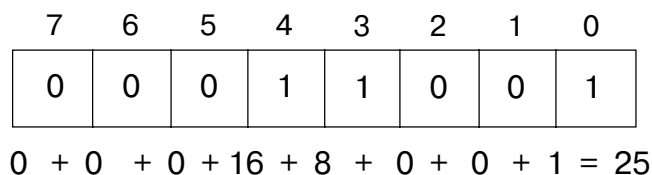
```

Выполнить установку цветов INK и PAPER немного сложнее и соответствующая программа в машинных кодах у нас займет уже не 6, а 13 байтов (она приведена на листинге 1.2).

За установку цветов PAPER и INK в основной части экрана отвечает системная переменная ATTR-P (23693 DEC = 5C8D H). Поэтому установка этих цветов из машинного кода сводится к переключению битов в данной системной переменной, а раскладка этих битов такова:



Если мы хотим из машинного кода дать команду, аналогичную PAPER 3; INK 1, то нам надо включить биты в системной переменной ATTR\_P следующим образом:



Т.е. фактически нам надо было бы в ячейку памяти с адресом 23693 заслать число 25, например так:

```
LD (23693), 25
```

Но такой команды, которая позволила бы заслать произвольное число в произвольную ячейку памяти компьютера у процессора Z-80 нет. Приходится немного комбинировать.

Сейчас у нас есть возможность познакомиться с интересным способом адресации в процессоре Z-80, который называется индексной адресацией. Дело в том, что в процессоре есть шестнадцатиразрядная регистровая пара IX, в которой может храниться какое-либо целое число от 0 до 65535. Работа компьютера "ZX-Spectrum" организована таким образом, что после включения его в сеть в этой регистровой паре устанавливается число 23610 (Это адрес системной переменной ERR\_NR). И этим активно пользуются на практике. Есть неофициальная договоренность, что для работы с системными переменными используется эта регистровая пара. При этом не надо указывать адрес нужной Вам системной переменной, а достаточно указать "смещение", т.е. величину, на которую отстоит нужный Вам адрес от заранее установленного в регистре IX. Тогда вместо LD (23693),25 можно записать команду:

```
LD (IX+83), 25
```

Эта запись выглядит понятнее уже хотя бы потому, что все, кто программируют на "Спектруме" в машинных кодах, сразу мгновенно понимают: раз используется IX, значит здесь загружают какую-то системную переменную. А какую именно, они быстро определяют, ОТКРЫВ таблицу системных переменных.

Вернемся к нашему примеру, см. листинг 1.2. Итак, в первой строчке мы установили необходимые цвета PAPER и INK.

## Листинг 1.2 .

| АДРЕС | МАШ. КОД    | АССЕМБЛЕР                   | КОММЕНТАРИЙ                        |
|-------|-------------|-----------------------------|------------------------------------|
| 23760 | FD 36 53 19 | ORG 23760<br>LD (IY+83), 25 | ; Установили в системной перемен-  |
| 23764 | 3E 02       | LD A, 02                    | ; ной BORDER значения INK и PAPER. |
| 23766 | CD 01 16    | CALL 5633                   | ; Подготовка к открытию канала 2.  |
| 23769 | CD 6B 0D    | CALL 3435                   | ; Открываем канал 2.               |
| 23772 | C9          | RET                         | ; Вызов процедуры CLS.             |
|       |             |                             | ; Выход.                           |

## Листинг 1.3

| АДРЕС | МАШ. КОД | АССЕМБЛЕР            | КОММЕНТАРИЙ                       |
|-------|----------|----------------------|-----------------------------------|
| 23760 | 3E 02    | ORG 23760<br>LD A, 2 | ; Подготовка к открыванию канала. |
| 23762 | CD 01 16 | CALL 5633            | ; Открываем канал экрана.         |
| 23765 | 3E 41    | LD A, 65             | ; Код символа "А".                |
| 23767 | D7       | RST 16               | ; Вызов процедуры печати символа. |
| 23768 | C9       | RET                  | ; Выход.                          |

## Листинг 1.4

| АДРЕС | МАШ. КОД | АССЕМБЛЕР            | КОММЕНТАРИЙ                       |
|-------|----------|----------------------|-----------------------------------|
| 23760 | 3E 02    | ORG 23760<br>LD A, 2 | ; Подготовка к открыванию канала. |
| 23762 | CD 01 16 | CALL 5633            | ; Открываем канал экрана.         |
| 23765 | 3E 48    | LD A, 72             | ; Код символа "Н".                |
| 23767 | D7       | RST 16               | ; Вызов процедуры печати символа. |
| 23768 | 3E 45    | LD A, 69             | ; Код символа "Е".                |
| 23770 | D7       | RST 16               | ; Вызов процедура печати символа. |
| 23768 | C9       | RET                  | ; Выход.                          |

Теперь мы должны дать команду CLS. Это тоже сделает за нас процедура, записанная в машинном коде ПЗУ. Данная процедура так и называется - CLS и находится по адресу 0D6BH = 3435 DEC.

Вместе с тем, Вы по-видимому обратили внимание на то, что перед вызовом этой процедуры вызывается еще одна процедура ПЗУ по адресу 5633 (1601H). Это весьма важная процедура, она называется CHAN\_OPEN, и мы будем еще не раз использовать ее при изучении материалов данной главы. Вызовом этой процедуры открывается канал вывода данных на экран - это канал под номером 2 и потому перед вызовом процедуры в регистр A загружается число 2. Когда Вы работали в БЕЙСИКе, то могли не думать о назначении каналов. Если же Вы выдадите информацию из машинного кода, то никто за Вас этого не сделает, а сделать это необходимо, иначе компьютер не поймет, куда надо подавать информацию.

Теперь Вы, уважаемые читатели, должны уметь устанавливать в машинном коде атрибуты экрана и выполнять очистку экрана. Давайте посмотрим как же нам напечатать на экране что-нибудь содержательное. Проще всего напечатать на экране какой-либо символ. В листинге 1.3 показан пример программы, которая напечатает символ "А" (символ имеет номер 65 DEC) в левом верхнем углу экрана с координатами позиции печати (0,0). Именно такие координаты текущей позиции устанавливаются после того, как отработает процедура CLS.

Если Вы хотите распечатать на экране какой-либо символ, то его код предварительно должен быть заслан в регистр A процессора, после чего должна быть выдана команда RST 16, что Вы и видите на листинге 1.3. По команде RST 16 процессор запустит процедуру ПЗУ PRINT\_A\_2, находящуюся по адресу 15F2H = 5618 DEC. Символ, код которого находится в аккумуляторе будет напечатан и осуществится переход к следующей позиции печати.

Если с этим Вам все понятно, то Вы без труда поймете работу программы 1.4, которая напечатает целое слово - "HE", поочередно засылая символы в аккумулятор и выдавая их командой RST 16.

Хорошо, что слово "HE" имеет только два символа, а как быть, если нужно напечатать

длинное предложение? Понятно ведь, что кодировать печать длинной фразы по символам дело очень утомительное, к тому же необходим большой расход оперативной памяти - он в два раза больше, чем длина Вашего сообщения, поскольку на каждый символ еще нужна команда RST 16, занимающая байт.

Для решения этой проблемы существуют как минимум три пути.

Главная идея состоит в том, чтобы записать все символы выводимого сообщения в некоторый банк данных и считывать их оттуда по одному до тех пор, пока весь банк не будет исчерпан. Этот прием показан в Листинге 1.5 (версия 1).

В этом листинге Вы знакомитесь с двумя новыми регистровыми парами - BC и DE. Как правило, пара BC используется программистами для организации в ней разного рода счетчиков, а пара DE - для указания на какой-либо адрес в памяти компьютера. Регистры B,C,D и E не обязательно должны использоваться парами, как это было с регистром Y. Они могут использоваться и как одиночные восьмибитные регистры для хранения целых чисел от 0 до 255.

Обратите внимание на пару операций

```
LD A,B
OR C
```

Это очень быстрый, экономичный и к тому же общепринятый способ проверки BC на ноль. Поскольку в регистре A у нас загружено содержимое регистра B, то результат операции OR C будет нулевым в том и только в той случае, если и B и C равны нулю.

Листинг 1.5 (Версия 1).

| АДРЕС | МАШ. КОД             | АССЕМБЛЕР  | КОММЕНТАРИЙ                       |
|-------|----------------------|------------|-----------------------------------|
|       |                      | ORG 23760  |                                   |
| 23760 | 3E 02                | LD A,2     | ;Подготовка к открыванию канала.  |
| 23762 | CD 01 16             | CALL 5633  | ;открываем канал экрана.          |
| 23765 | 11 E4 5C             | LD DE,DATA | ;В регистровую пару DE загрузили  |
|       |                      |            | ;адрес, в котором хранится наше   |
|       |                      |            | ;текстовое сообщение.             |
| 23768 | 01 07 00             | LD BC,7    | ;В регистровой паре BC организо-  |
|       |                      |            | ;вали счетчик на 7 символов.      |
|       |                      | LOOP       |                                   |
| 23771 | 78                   | LD A,B     | ;Проверка не обнулился ли счет-   |
| 23772 | B1                   | OR C       | ;чик символов в BC?               |
| 23773 | C8                   | RET Z      | ;Выход, если он обнулялся.        |
| 23774 | 1A                   | LD A,(DE)  | ;Взяли в аккумулятор текущий      |
|       |                      |            | ;символ из списка DATA.           |
| 23775 | D7                   | RST 16     | ;Напечатали его на экране.        |
| 23776 | 13                   | INC DE     | ;Перешли к очередному символу.    |
| 23777 | 08                   | DEC BC     | ;Уменьшили счетчик символов на 1. |
| 23778 | 18 F7                | JR LOOP    | ;Перешли на метку LOOP для оче-   |
|       |                      |            | ;редной проверки не закончился    |
|       |                      |            | ;ли список печатаемых символов.   |
|       |                      | DATA       |                                   |
| 23779 | 72 69 76 80 32 77 69 |            | ;Начиная с адреса 23779 мы храним |
|       |                      |            | ;сообщение "HELP ME".             |

Листинг 1.5 (Версия 2).

| АДРЕС | МАШ. КОД             | АССЕМБЛЕР  | КОММЕНТАРИЙ                      |
|-------|----------------------|------------|----------------------------------|
|       |                      | ORG 23760  |                                  |
| 23760 | 3E 02                | LD A,2     | ;Подготовка к открыванию канала. |
| 23762 | CD 01 16             | CALL 5633  | ;Открываем канал экрана.         |
| 23765 | 11 DF 5C             | LD DE,DATA | ;Адрес сообщения.                |
| 23768 | 01 07 00             | LD BC,7    | ;Длина сообщения - 7 символов.   |
| 23771 | CD 3C 20             | CALL 8252  | ;Вызов процедуры ПЗУ PR_STRING.  |
| 23774 | C9                   | RET        | ;Выход.                          |
|       |                      | DATA       |                                  |
| 23775 | 72 69 76 80 32 77 69 |            | ;Сообщение "HELP ME".            |

Обратите также внимание на то, что мы записали наше сообщение, начиная с адреса 23779. Но это совсем не обязательно. Можете хранить свои тексты где хотите, в любых адресах, удобных Вам. Важно только, чтобы в строке LD DE,DATA в регистровую пару DE был заслан истинный адрес начала Вашего сообщения ОТ 0 ДО 65535.

Второй вариант этой программы выглядит немного более простым. Если Вы заранее установили в регистровой паре DE адрес, с которого начинается Ваше сообщение, а в BC - длину этого сообщения, то больше можно вообще ничего не делать, а вызвать процедуру ПЗУ под названием PR\_STRING, которая сама напечатает за Вас это длинное сообщение. Процедура PR\_STRING находится по адресу 8252 DEC = 203CH. Пример см. в Листинге 1.5 (версия 2).

Третий способ - самый мощный. Его применяют в тех случаях, когда у Вас заготовлено не одно сообщение, а много, т.е. Вы имеете целую таблицу сообщений, но заранее не знаете, какое когда придется печатать. В этом случае Вам хотелось бы выдавать на экран нужное сообщение указав только его номер. Именно так и поступают в абсолютном большинстве программ, написанных в машинных кодах. Именно так организовано и само ПЗУ, которое, как Вы знаете, хранит в себе много разных сообщений типа "Start tape and press key..." и выдает вам всякий раз именно то, которое надо.

Чтобы программа знала, где кончается одно сообщение и начинается другое, применяют специальный прием - к коду последнего символа сообщения прибавляют число 127, т.е. фактически в последнем символе каждого сообщения принудительно включают старший (7-ой) бит. Таким образом, символ, код которого  $> 127$  является маркером конца сообщения. Теперь нетрудно создать программу для печати 1-го сообщения. Она просмотрит тексты, начиная с адреса, установленного в DE, отсчитает N-1 маркер и начнет печатать то, что найдет после него. Символы будут печататься по одному до тех пор, пока не будет найден N-ый маркер. Для печати последнего придется предварительно отнять от его значения число 127.

В листинге 1.5 приведен пример такой работы, но для простоты там принято, что у Вас есть не таблица, а только одно сообщение, заканчивающееся инвертированным символом (к коду символа прибавлено число 127).

Кстати, обратите внимание на то, что если числа, большие чем 27, являются маркерами конца сообщения, то очевидно нельзя допускать, чтобы в Ваших текстах были символы, код которых больше 27, т.е. символы блочной графики коды от 128 до 143) и символы графики пользователя (коды от 144 до 164) воспроизводить таким методом нельзя.

Похожая на эту, но более сложная процедура вывода на экран значений клавишных слов и сообщений об ошибках используется в ПЗУ компьютера. Стартовый адрес этой процедуры - 3082 -DEC = 0C0AH.

Вы можете и сами выдавать из машинного кода системные сообщения об ошибках, полный перечень которых обычно приводят в руководстве пользователя компьютера.

Так, они могут быть легко выведены на экран с помощью команды RST 8. Байт, следующий за этой командой, определяет какой вид сообщения необходимо вывести, например, если необходимо вывести на экран сообщение "K invalid colour", которое в перечне сообщений об ошибках является двадцать первым по счету, то необходимо записать:

```
RST 8
```

```
DEFB 19 (номер сообщения минус 2)
```

Причем нет необходимости давать команду RET после этого, так как сразу же после вывода на экран сообщения об ошибке осуществляется возврат в BASIC-систему.

Теперь мы можем приступить к рассмотрению способов вывода на экран символов или даже целых строк в заданную позицию на экране. В BASIC-программах это выполняется с помощью оператора PRINT AT, после которого записывается номер строки, номер столбца и, наконец, само сообщение, которое необходимо вывести на экран.

Для этого используются так называемые управляющие коды. Если Вы посмотрите любую таблицу символов ASCII, то увидите, что печатным символам соответствуют коды, начиная с 32 DEC (это символ "пробел". А коды с 0 до 31 являются непечатными (т.е. не имеют закрепленной за ними литеры) и используются в качестве управляющих. Так,

например, в "Спектруме" коды с 0 по 5 и с 24 по 31 вообще не используются, а с 6 по 23 - служат в качестве управляющих. (В компьютерах других систем это может быть и не так).

За координаты позиции печати отвечает код 22, он является аналогом БЕЙСИКовского оператора AT. К нашему счастью, команда RST 16 распознает код 22 как PRINT AT и использует следующие 2 байта как "х" и "у" координаты, задающие PRINT-позицию для вывода сообщения на экран. Программа 1.6 демонстрирует использование последовательности байтов

```
DEFB 22
DEFB 3
DEFB 5
```

как аналог оператора

```
PRINT AT 3,5.
```

Совершенно так же можно применять коды 23 и 13 как аналоги операторов TAB и ENTER, соответственно.

Листинг 1.5 (Версия 3).

| АДРЕС | МАШ. КОД    | АССЕМБЛЕР  | КОММЕНТАРИЙ                                                                |
|-------|-------------|------------|----------------------------------------------------------------------------|
|       |             | ORG 23760  |                                                                            |
| 23760 | 3E 02       | LD A,2     | ; Подготовка к открыванию канала.                                          |
| 23762 | CD 01 16    | CALL 5633  | ; Открываем канал экрана.                                                  |
| 23765 | 11 E5 5C    | LD DE,DATA | ; Адрес начала сообщения.                                                  |
|       |             | LOOP       |                                                                            |
| 23768 | 1A          | LD A,(DE)  | ; Взяли текущий символ.                                                    |
| 23769 | CB 7F       | BIT 7,A    | ; Проверяем его старший бит.                                               |
| 23771 | 20 04       | JR NZ,END  | ; Если он не нулевой, то это последний символ и мы переходим на метку END. |
|       |             |            | ; Иначе печатаем его на экране.                                            |
| 23773 | D7          | RST 16     | ; Перешли к очередному символу.                                            |
| 23774 | 13          | INC DE     | ; Возврат на метку LOOP.                                                   |
| 23775 | 18 F7       | JR LOOP    |                                                                            |
|       |             | END        |                                                                            |
| 23777 | CB BF       | RES 7,A    | ; Выключили старший бит последнего символа.                                |
| 23779 | D7          | RST 16     | ; Напечатали последний символ.                                             |
| 23780 | C9          | RET        | ; Выход DATA                                                               |
| 23781 | 72 69 76 80 | 32 77 197  | ; Сообщение "HELP ME".                                                     |

Листинг 1.6

| АДРЕС | МАШ. КОД    | АССЕМБЛЕР  | КОММЕНТАРИЙ                                   |
|-------|-------------|------------|-----------------------------------------------|
|       |             | ORG 23760  |                                               |
| 23760 | 3E 02       | LD A,2     | ; Подготовка к открыванию канала.             |
| 23762 | CD 01 16    | CALL 5633  | ; Открываем канал экрана.                     |
| 23765 | 11 DF 5C    | LD DE,DATA | ; Адрес начала сообщения.                     |
| 23768 | 01 0C 00    | LD BC,12   | ; Длина сообщения с учетом управляющих кодов. |
| 23771 | CD 3C 20    | CALL 8252  | ; Вызов процедуры ПЗУ PR_STRING               |
| 23774 | C9 RET      |            | ; Выход DATA                                  |
| 23775 | 22          | DEFB 22    | ; Код управления позицией печати              |
| 23776 | 03          | DEFB 3     | ; Координата "Y=3"                            |
| 23777 | 05          | DEFB 5     | ; Координата "X=5"                            |
| 23778 | 84 72 65 78 | 4B         | ; Текст сообщения "THANK YOU"                 |
| 23783 | 32 89 79 65 |            |                                               |

Листинг 1.7

| АДРЕС | МАШ. КОД    | АССЕМБЛЕР  | КОММЕНТАРИЙ                                   |
|-------|-------------|------------|-----------------------------------------------|
|       |             | ORG 23760  |                                               |
| 23760 | 3E 02       | LD A,2     | ; Подготовка к открыванию канала.             |
| 23762 | CD 01 16    | CALL 5633  | ; Открываем канал экрана.                     |
| 23765 | 11 E2 5C    | LD DE,DATA | ; Адрес начала сообщения.                     |
| 23768 | 01 24 00    | LD BC,36   | ; Длина сообщения с учетом управляющих кодов. |
| 23771 | CD 3C 20    | CALL 6252  | ; Вызов процедуры ПЗУ PR_STRING               |
| 23774 | CD 4D 0D    | CALL 3405  | ; Копирование ATTR-P в ATTR-T.                |
| 23777 | C9          | RET        | ; Выход DATA                                  |
| 23778 | 22 04 07 17 | 02         | ; Сообщение CAN YOU READ THIS c               |

|       |    |    |    |    |    |  |                                  |
|-------|----|----|----|----|----|--|----------------------------------|
| 33783 | 16 | 04 | 18 | 01 | 19 |  | ; инплантированными в нем кодами |
| 23788 | 01 | 21 | 01 | 20 | 01 |  | ; управления координатой позиции |
| 23793 | 67 | 65 | 78 | 32 | 89 |  | ; печати и кодами управления     |
| 23798 | 79 | 85 | 32 | 82 | 69 |  | ; цветовыми атрибутами.          |
| 23803 | 65 | 68 | 32 | 84 | 72 |  |                                  |
| 23808 | 73 | 83 | 21 | 00 | 20 |  |                                  |
| 23813 | 00 |    |    |    |    |  |                                  |

С помощью управляющих кодов можно не только задавать координаты позиции печати, но и цвета и прочие атрибуты печатаемых символов. Для этого есть свои управляющие коды.

|         |          |
|---------|----------|
| INK     | - код 16 |
| PAPER   | - код 17 |
| FLASH   | - код 18 |
| BRIGHT  | - 19     |
| INVERSE | - 20     |
| OVER    | - код 21 |

Программа 1.7 показывает как используются управляющие коды, соответствующие PAPER, INK, BRIGHT и т.д.

После использования временных цветов для вывода на экран сообщения необходимо вернуться к постоянным цветам для обеспечения дальнейшего вывода. Для режимов OVER и INVERSE это требование легко выполняется, так как команда RST 16 легко распознает их управляющие коды. Для изменения PAPER, INK, BRIGHT и FLASH, необходимо вызвать из ПЗУ процедуру (адрес 3405), которая копирует содержимое системной переменной ATTR-P (23693) в системную переменную ATTR-T(23695).

(Продолжение следует)



## ЗАЩИТА ПРОГРАММ

Мы продолжаем публикацию материалов, посвященных защите программного обеспечения от просмотра и копирования. Непосредственная тема сегодняшней статьи связана с подробным разбором стандартных процедур ПЗУ, осуществляющих загрузку данных с магнитофона.

(ПРОДОЛЖЕНИЕ)

Начало см. в "ZX-РЕВЮ-92":

N 1,2 - с. 9-16

N 3,4 - с. 53-60

N 5,6 - с. 97-104

N 7,8 - с. 141-146

N 9,10 - с. 185-192

N 11,12 - с. 230-237.

### **1.1 Процедуры в машинных кодах, реализующие загрузку программ с магнитофона.**

Мы с Вами рассмотрели систему кодирования информации на магнитной ленте. Теперь приступим к рассмотрению процедур ПЗУ, реализующих ввод программ "Спектрума" с магнитофона.

Для того, чтобы точно оценить временные характеристики операций, реализующих необходимые задержки в цикле загрузки, необходимо знать время исполнения отдельных инструкций процессора Z-80. Как Вам уже вероятно известно, скорость работы процессора на несколько порядков выше максимальной скорости записи кодированных логических единиц информации на магнитной ленте. Поэтому программа должна выполнять задержку после каждого записываемого на магнитофон бита данных.

Время задержки удобно считать не в секундах, а в циклах тактового генератора, который синхронизирует работу процессора Z-80. У каждой команды процессора есть своя продолжительность, которая может быть измерена в этих циклах. Впрочем, не для всех команд это величина постоянная. Так, например, для команд, связанных с каким-либо условием, (например JR NZ и т.п.) число циклов обычно зависит от того, выполняется или нет данное условие и для каждого случая различно, поэтому нам необходимо учитывать и этот факт и рассматривать соответствующие приемы задержки, используемые в данной процедуре обработки информации, поступающие с магнитной ленты.

Информацию о количестве циклов по каждой команде процессора Вы можете получить из справочника "ИНФОРКОМа" по микропроцессору Z-80, см. книгу "Программирование в машинных кодах. Первые шаги. Практикум. Справочник".

Процедура, выполняющая загрузку информации с магнитофона, начинается в ПЗУ с адреса 0556H = 1366 DEC. Ее действие заключается в многократной проверке шестого бита порта 254 и в определении временного интервала между двумя следующими друг за другом фронтами.

Таким образом, при чтении блока данных с ленты важнейшим является точное определение интервала между фронтами сигнала. Эту работу выполняет специализированная процедура, размещенная в ПЗУ компьютера под адресом 05E3H = 1507. Эту точку входа в ПЗУ можно использовать, когда надо установить, сколько времени прошло от момента вызова процедуры до обнаружения двух, следующих один за другим, смен уровней сигналов в гнезде EAR (и насколько оба они соответствуют допустимым пределам требуемых временных интервалов).

Второй важной точкой входа в ПЗУ является специализированная процедура,

позволяющая определить, сколько времени прошло с момента вызова процедуры до получения первого фронта. Она размещается по адресу 05E7H = 1511. Вызов этой процедуры выполняется после помещения в регистр В временной константы, ограничивающей количество проверок порта 254.

Если оказывается, что время ожидания смены фронтов сигнала превышено или будет установлен факт нажатия клавиши BREAK, происходит сброс флага С флагового регистра F. Для распознавания каждой из этих двух ситуаций служит флаг Z. Если он обнулен, значит во время работы данной процедуры была нажата клавиша BREAK.

Мы начнем рассмотрение процедуры загрузки с изучения этих подпрограмм. Названия процедур стали общепринятыми после выхода в свет книги Я.Логана и Ф.О'Хары "Полный дисассемблер ПЗУ Спектрума". Те же названия использованы и в наиболее полном описании работы "Спектрума" с магнитофоном Анджее Каллофа ("Спектрум и магнитофон") и мы сохраняем те же наименования процедур.

|   | МЕТКА     | АССЕМБЛЕР       | ЦИКЛЫ   |
|---|-----------|-----------------|---------|
| 1 | LD-EDGE-2 | CALL LD-EDGE 1  | ; 17    |
| 2 |           | RET NC          | ; 11, 5 |
| 3 | LD-EDGE-1 | LD A, #16       | ; 7     |
| 4 | LD-DELAY  | DEC A           | ; 1     |
| 5 |           | JR NZ, LD-DELAY | ; 12, 7 |
| 6 |           | AND A           | ; 4     |

Как Вы уже заметили, LD-EDGE-2 обеспечивает двукратный вызов процедуры LD-EDGE-1. Перед тем, как приступить к проверке порта, выжидается 358 тактов работы Z-80. процессор слишком быстродействующий по сравнению со временем, в течение которого на основе активного сигнала от ULA, стабилизируется состояние шестого бита порта 254.

В последней колонке приведено время затрачиваемое на выполнение каждой операции, измеренное в тактах микропроцессора. Так, в случае выполнения операции возврата из подпрограммы, в зависимости от состояния флага переноса C - RET NC - затрачиваются 5 или 11 тактов микропроцессора, в связи с тем, выполняется условие, или нет. По аналогичной причине мы приводим два значения и для процедуры условного перехода JR NZ, поскольку время, затрачиваемое на ее выполнение, может быть различным в зависимости от выполнения или невыполнения условия.

После начала работы процедуры LD-EDGE-1, в аккумулятор загружается шестнадцатеричное значение 16H. Благодаря этому мы организуем цикл задержки, основанный на уменьшении аккумулятора на единицу и сравнении этого значения с нулем. Если ноль не достигнут, мы продолжаем уменьшать аккумулятор на единицу, обеспечивая задержку. Так происходит ровно  $7 + 21 \times 16 + 4 + 7 + 4 = 358$  тактов микропроцессора.

Командой AND A сбрасывается флаг C, который, как мы знаем, выполняет функции маркера.

После обнуления маркера проверяется, достигла ли временная константа в регистре В значения нуля. Обратите внимание, что в противоположность процедурам записи, здесь значение регистра В увеличено на единицу.

|   |           |       |         |
|---|-----------|-------|---------|
| 7 | LD-SAMPLE | INC B | ; 4     |
| 8 |           | RET Z | ; 11, 7 |

Следующим этапом идет контроль нажатия клавиши BREAK. Для этого мы считываем числовое значение из порта 254 и, после ротации данного значения (причем интересующий нас бит смещается во флаг переноса C), мы осуществляем продолжение работы, либо выходим из данной процедуры. Все зависит от того, была ли нажата клавиша BREAK.

|    |  |             |         |
|----|--|-------------|---------|
| 9  |  | LD A, #7F   | ; 7     |
| 10 |  | IN A, (#FE) | ; 11    |
| 11 |  | RRA         | ; 4     |
| 12 |  | RET NC      | ; 11, 7 |

Теперь мы проверяем, изменился ли шестой бит 254 порта, в соответствии со значением, сохраняемым в регистре С (именно этот бит сигнализирует о поступлении сигнала с магнитофона). Если бит не изменился, о чем свидетельствует нулевой флаг Z, то он проверяется снова.

```

13          XOR C           ; 4
14          AND #20         ; 7
15          JR Z, LD-SAMPLE ; 32, 7

```

Фронт сигнала извлечен, инвертируются все биты в регистре С и одновременно изменяется информация о состоянии порта, а также цвет изменяется на дополнительный цвет бордюра (красный-голубой или синий-желтый). Эти операции выполняются с помощью специальной команды инверсии. Для этого мы заносим в аккумулятор содержимое регистра С, после чего над аккумулятором выполняется команда CPL, которая инвертирует каждый его бит на противоположный. В результате получается как бы дополнение содержимого аккумулятора до 255 (в абсолютной двоичной арифметике).

После этого инвертированное значение мы возвращаем в регистр С.

```

16          LD A, C         ; 4
17          CPL             ; 4
18          LD C, A         ; 4

```

После смены цвета бордюра на экране включается флаг С регистра F, что сигнализирует об успешном результате проверки. Соответствующие промежутки времени подсчитывает процедура, вызываемая на основе приращения значения в регистре В. Каждый вызов процедуры LD-EDGE-1 длится 465 тактов микропроцессора плюс по 58 тактов на каждый дополнительный тест.

```

19          AND #07         ; 7
20          OR #08          ; 7
21          OUT (#FE), A    ; 11
22          SCF             ; 4
23          RET             ; 10

```

Рассмотрим теперь основную загрузочную процедуру LD-BYTES. Она занимает в памяти объем начиная от 0556H до 05E2H. Перед началом работы процедуры в регистр DE необходимо занести длину читаемого блока, в регистр IX адрес байта, начиная с которого должна быть загружена информация. В аккумулятор заносится тип блока.

LD-BYTES может не только считывать файлы в память, но также и сравнивать их. На этом принципе основано действие БЕЙСИК-команды VERIFY. Необходимый режим работы определяется состоянием флага переноса С перед вызовом процедуры загрузки. Его установка в единицу означает считывание в память всего файла, а обнуление - сравнение. Поскольку микропроцессор может работать не более чем с шестнадцатиразрядными числами, длина читаемого блока не может превышать 65535 (FFFFH).

```

1  LD-BYTES  INC D           ; 4
2           EX AF, A'F'      ; 4
3           DEC D           ; 4
4           DI              ; 4
5           LD A, #0F        ; 7
6           OUT (#FE), A    ; 11
7           LD HL, #053F     ; 10
8           PUSH HL

```

Регистр D увеличивается с целью обнуления флага Z перед сохранением его в альтернативном наборе регистров микропроцессора. После этого значение регистра D

восстанавливается и запрещаются маскированные прерывания.

Занесение в регистр А значения 0FH знаменует собой установку белого цвета бордюра путем выдачи этого значения в порт 254. В регистровую пару HL заносится адрес процедуры обработки возврата SA-LD-RET, который сохраняется на стеке.

|    |             |      |
|----|-------------|------|
| 9  | IN A, (#FE) | ; 11 |
| 10 | RRA         | ; 4  |
| 11 | AND #20     | ; 7  |
| 12 | OR #02      | ; 7  |
| 13 | LD C, A     | ; 4  |
| 14 | CP A        | ; 4  |

Первый тест порта 254 представляет собой занесение в аккумулятор значения состояния данного порта с последующей его ротацией и выполнением операции логического "ИЛИ". Из всего байта оставляется только шестой бит, который характеризует состояние гнезда EAR. После выполнения этого цвет бордюра становится красным. После помещения в регистр С значения 22H или 02H включается флаг Z флагового регистра.

|    |          |                 |         |
|----|----------|-----------------|---------|
| 15 | LD-BREAK | RET NZ          | ; 11, 7 |
| 16 | LD-START | CALL LD-EDGE-1  | ; 17    |
| 17 |          | JR NC, LD-BREAK | ; 12, 7 |

Следующий отрезок программы представляет собой цикл, в котором программа будет задержана до нажатия клавиши BREAK или до смены состояния шестого бита порта 254. Во втором случае цвет бордюра будет изменен на голубой. Выполнение данной процедуры осуществляется многократным повторением вызова подпрограммы LD-BREAK.

|    |         |                |         |
|----|---------|----------------|---------|
| 18 |         | LD HL, #0415   | ; 10    |
| 19 | LD-WAIT | DJNZ LD-WAIT   | ; 10, 6 |
| 20 |         | DEC HL         | ; 6     |
| 21 |         | LD A, H        | ; 4     |
| 22 |         | OR L           | ; 4     |
| 23 |         | JR NZ, LD-WAIT | ; 12, 7 |

Перед началом непосредственного чтения информации с магнитной ленты выжидается около одной секунды. Это достигается за счет создания специального цикла задержки. Длительность ожидания зависит от значения, которое занесено в регистр HL. В нашем случае оно составляет #0415.

|    |  |                 |         |
|----|--|-----------------|---------|
| 24 |  | CALL LD-EDGE-2  | ; 17    |
| 25 |  | JR NC, LD-BREAK | ; 12, 7 |

Первым делом проверяется, не передается ли еще какой-нибудь сигнал за данный промежуток времени. За время, составляющее приблизительно 14000 тактов микропроцессора должны появиться два фронта сигналов. Обратим внимание на большой излишек времени перед началом загрузки данных. В пилотирующем сигнале фронты появляются в интервале порядка 2186 тактов.

|    |           |                 |         |
|----|-----------|-----------------|---------|
| 26 | LD-LEADER | LD B, #9C       | ; 7     |
| 27 |           | CALL LD-EDGE-2  | ; 17    |
| 28 |           | JR NC, LD-BREAK | ; 12, 7 |
| 29 |           | LD A, #C6       | ; 7     |
| 30 |           | CP B            | ; 4     |
| 31 |           | JR NC, LD-START | ; 12, 7 |

В регистр В заносится временная константа 9CH, позволяющая процедуре LD-EDGE-2 искать два фронта за время, не превышающее 7000 тактов микропроцессора. С другой стороны, после их отыскания проверяется прошло ли с момента вызова вышеуказанной

процедуры до появления заднего фронта не менее, чем 3366 тактов работы микропроцессора. Таким способом отличаются фронты пилотирующего сигнала от значительно более близких пар фронтов информационных сигналов.

```
32          INC H          ; 4
33          JR NZ, LD-LEADER ; 12, 7
```

В цикле мы начинаем искать фронты пилотирующего сигнала. После извлечения 256 пар, мы переходим к ожиданию импульса синхронизации, который сигнализирует нам о начале собственно самого файла данных.

```
34 LD-SYNC LD B, #C9      ; 7
35          CALL LD-EDGE-1 ; 17
36          JR NC, LD-BREAK ; 12, 7
```

Проверяется время появления каждого нового фронта сигнала. Он должен быть обнаружен в течение 3655 тактов работы микропроцессора. Необходимая точность обеспечивается занесением в регистр В временной константы C9H и вызовом процедуры LD-EDGE-1.

```
37          LD A, B        ; 4
38          CP #D4         ; 7
39          JR NC, LD-SYNC  ; 12, 7
```

Однако, если это произошло до истечения 1100 тактов работы процессора, значит, уже появился первый фронт синхроимпульса. Для определения его появления мы заносим в аккумулятор содержимое регистра В и сравниваем полученное значение с числовой константой D4H. В зависимости от состояния флага переноса, программа либо продолжает свою работу, либо же возвращается на начало процедуры проверки появления синхроимпульса по метке LD-SYNC. Значит, сигнал пришел слишком рано и это не тот сигнал, который нужен.

```
40          CALL LD-EDGE-1 ; 17
41          RET NC         ; 11, 5
```

После появления первого фронта синхроимпульса мы ожидаем появление второго. Это достигается за счет вызова процедуры LD-EDGE-1. Если после вызова данной процедуры, она возвращает значение флага переноса выключенным, то мы осуществляем возврат в вызвавшую программу по команде RET NC.

```
42          LD A, C        ; 4
43          XOR #03        ; 7
44          LD C, A        ; 4
```

После появления обоих фронтов синхроимпульса необходимо осуществить соответствующую индикацию на экране. Это делается подготовкой в регистре С сигнализации о появлении следующего фронта синим и желтым цветами. Для этого мы загружаем в аккумулятор содержимое регистра С и осуществляем операцию исключающие "ИЛИ" над содержимым аккумулятора. После этого полученное значение вновь возвращается в регистр С.

```
45          LD H, #00      ; 7
46          LD B, #80      ; 7
47          JR LD-MARKER   ; 12
```

В регистре Н создается байт четности, а в регистр В помещается временная константа для считывания первого бита первого байта, описывающего тип читаемого

файла.

Как Вы помните, значение первого байта каждого файла характеризует тип файла и должно равняться нулю для заголовка и 255 для блока данных.

```
48 LD-LOOP      EX AF,A'F'      ; 4
49              JR NZ,LD-FLAG    ; 12, 7
```

К этому фрагменту программа возвращается после чтения целого байта, поскольку предыдущая часть программы закончилась безусловным переходом на метку LD-MARKER. В этом участке программы флаг Z употреблен для различения байта, определяющего тип файла. Здесь также учитывается флаг C, определяющий, происходит ли процесс чтения или верификации.

```
50              JR NC,LD-VERIFY   ; 12, 7
51              LD (IX+0),L        ; 19
52              JR LD-NEXT        ; 12
```

В ходе выполнения данных операций считанный байт помещается под адресом, находящимся в IX. Для этого мы загружаем в ячейку памяти, адрес которой находится в (IX+0), содержимое регистра L. После этого осуществляется безусловный переход на метку LD-NEXT.

```
53 LD-FLAG      RL C              ; 4
54              XOR L              ; 4
55              RET NZ             ; 11, 5
```

При входе в данную процедуру в регистре A находится тип файла, который мы хотим считать, а в регистре L - первый считанный байт. Первая команда служит для временного хранения значений флага C в регистре C. Если считанный байт не совпадает с содержимым регистра A, то происходит возврат из процедуры со сброшенными флагами C и Z, что сигнализирует об ошибке.

```
56              LD A,C            ; 4
57              RRA                ; 4
58              LD C,A            ; 4
59              INC DE             ; 4
60              JR LD-DEC         ; 12
```

В случае совпадения типов восстанавливается состояние флага C и регистра C. Для этого мы загружаем содержимое регистра C в аккумулятор, осуществляем ротацию C через флаг переноса и возвращаем полученное значение из аккумулятора в регистр C. Увеличение регистра DE на единицу служит для компенсации его состояния при переходе к инструкции DEC DE при выполнении безусловного перехода на метку LD-DEC.

```
64 LD-NEXT      INC IX            ; 10
65 LD-DEC       DEC DE            ; 6
66              EX AF,A'F'        ; 4
```

Данные команды устанавливают регистры IX и DE в состояние, необходимое для правильной работы программ. После этого осуществляется сохранение флага C флагового регистра F в безопасном месте с помощью использования команды перехода к набору альтернативных регистров микропроцессора Z-80.

```
67              LD B,#B2          ; 7
68 LD-MARKER    LD L,01           ; 7
```

После установки временной константы B2H в регистре B, обнуляется регистр L и его

младший бит устанавливается в единицу. После следующих сдвигов влево он будет постепенно перемещаться по направлению к флагу С и в конце концов окажется во флаге переноса, сигнализируя считывание всего байта.

```
69      LD-8-BITS  CALL LD-EDGE-2      ; 17
70      RET NC      ; 11, 5
```

Данная процедура определяет время между двумя очередными фронтами сигналов. Для этого вызывается процедура LD-EDGE-2, которая работает в соответствии со значением временной константы занесенной в регистр В.

```
71      LD A, #CB      ; 7
72      CP B            ; 4
73      RL L            ; 4
```

Если пара фронтов найдена за время, меньшее чем около 2400 тактов относительно вызова LD-EDGE-2, то они представляют собой ноль, а если за большее время, то единицу. Значение, получающееся после анализа информации, несет в себе флаг С.

Для правильного выполнения заданной процедуры мы загружаем в аккумулятор константу CBH, относительно которой ведется проверка содержимого регистра В.

```
74      LD B, #B0      ; 7
75      JP NC, LD-8-BITS ; 10
```

После загрузки временной константы для следующего бита исследуется регистр В, который устанавливается только после считывания и размещения в регистре L всех восьми битов данного байта.

```
76      LD A, H        ; 4
77      XOR L          ; 4
78      LD H, A        ; 4
```

Этот участок программы осуществляет модификацию байта четности, о котором мы уже писали ранее.

```
79      LD A, D        ; 4
80      OR E           ; 4
81      JR NZ, LD-LOOP ; 12, 7
```

Здесь мы проверяем все ли байты были считаны и, если должны быть считаны еще какие-либо байты, то мы возвращаемся к метке LD-LOOP. Эта проверка осуществляется с помощью сравнения содержимого регистра DE с нулем и осуществлением операции условного перехода в зависимости от состояния флага Z флагового регистра.

```
82      LD A, H        ; 4
83      CP #01         ; 7
84      RET            ; 10
```

После считывания последнего байта (четности), в регистре H должен быть ноль. Если это так, то флаг С включается, сигнализируя этим правильность окончания загрузки файла.

Так работают процедуры загрузки файлов с ленты в память компьютера. Надеемся, что этот материал окажется полезным для Вас. Тем не менее, его невозможно рассматривать в отрыве от информации по вопросам записи файлов на магнитную ленту в формате "Спектрума". Поэтому в следующей главе мы рассмотрим более подробно этот вопрос.

## Глава 2. Стандартные процедуры записи информации на магнитную ленту.

Теперь мы переходим к рассмотрению процедур, реализующих запись информации на магнитную ленту и помещенных в ПЗУ компьютера.

Начнем с процедуры SA-BYTES, служащей для записи последовательности битов на кассету. Предполагается, что в момент ее запуска в регистре DE находится длина записываемого блока, в регистре IX - адрес первого байта, а в аккумуляторе - число, определяющее тип загружаемого блока. Напомним, что для заголовка этот байт равен нулю, в то время как для блока данных он равен 255. Первая процедура записи информации на магнитную ленту начинается с адреса 04C2H. Это является ее отправной точкой.

```
1      SA-BYTES    LD HL,#C53F          ; 10
2                      PUSH HL           ; 11
3                      LD HL,#1F80       ; 10
4                      BIT 7,A            ; 8
5                      JR Z,SA-FLAG      ; 12, 7
6                      LD HL,#0C98       ; 10
```

Первым делом загружаем в регистр HL адрес процедуры возврата к БЕЙСИКу - SA-LD-RET, после чего данное значение сохраняется на стеке. Далее мы загружаем в регистр HL временные константы 1F80H и 0C98H, которые определяют длительность пилотирующего сигнала. Эта длительность составляет пять секунд для заголовка, и две секунды для блока данных. Определить, какой тип файла мы сейчас записываем на ленту, компьютеру помогает седьмой бит аккумулятора. Напомним, что в аккумуляторе содержится число, определяющее тип блока.

```
7      SA-FLAG     EX AF,A'F'          ; 4
8                      INC DE            ; 6
9                      DEC IX            ; 10
10                     DI                ; 4
11                     LD A,#02           ; 7
12                     LD B,A             ; 4
```

После обработки блока содержимое регистра DE увеличивается на единицу. На столько же уменьшается и содержимое регистра IX. Все это делается после вызова альтернативного набора регистров микропроцессора и дает возможность определить байт как первый в записываемом блоке.

Выключение маскируемого прерывания необходимо, чтобы программа не была прервана для опроса клавиатуры (в обычном режиме опрос клавиатуры происходит пятьдесят раз в секунду, так как именно с такой регулярностью поступают импульсы на вход INT микропроцессора). Значение 2 в аккумуляторе установит предварительное напряжение в гнезде MIC на низкий уровень и цвет рамки на красный.

```
13     SA-LEADER    DJNZ SA-LEADER      ; 13, 8
14                     OUT (#FE),A       ; 11
15                     XOR #0F           ; 7
16                     LD B,#A4          ; 7
17                     DEC L              ; 4
18                     JR NZ,SA-LEADER    ; 12, 7
```

Следующий блок программы генерирует пилотирующий сигнал. Цикл SA-LEADER гарантирует задержку между последующими сменами напряжения в гнезде MIC. Затем, после установки напряжения в этом гнезде, четыре младших бита аккумулятора меняются на противоположные путем выполнения операции исключающее "ИЛИ" над содержимым аккумулятора. Это подготавливает изменение напряжения на разъеме MIC на противоположное и смену цвета бордюра с красного на голубой или наоборот. Затем



уменьшается младший байт счетчика импульсов L.

Многие пользователи "Спектрума" наверняка встречали компьютерные программы без характерных полос на бордюре, присущих стандартным процедурам загрузки. Это объясняется тем, что цвет бордюра в ходе загрузки файла не изменяется. Как Вы помните, за цвет бордюра ответственны три младшие бита компьютерного порта 254. Поэтому, если Вы будете проводить операцию исключающее "ИЛИ" только над битом, ответственным за состояние сигнала на разъеме MIC и не будете затрагивать биты цвета бордюра. Вам также удастся добиться аналогичного эффекта.

```
19          DEC B          ; 4
20          DEC H          ; 4
21          JP P, SA-LEADER ; 10
```

После обнуления младшего байта счетчика импульсов L уменьшается и старший байт H и модифицируется в соответствующей ситуации временная постоянная в регистре B с целью учета дополнительно выполненных инструкций (в данном случае это тринадцать тактов микропроцессора). Как Вы уже могли заметить, после некоторых команд стоит несколько значений, характеризующих длительность выполнения команды в циклах микропроцессора Z-80. Это объясняется тем, что команда может быть выполнена по-разному в зависимости от того, выполняется условие или нет. И, следовательно, длительность ее выполнения для каждого из этих случаев различна. Это объясняет тот факт, что в ходе работы такого рода процедур необходимо учитывать время, затрачиваемое на дополнительно выполняемые инструкции.

```
22          LD B, #2F      ; 7
23  SA-SYNC1  DJNZ SA-SYNC1 ; 10, 8
24          OUT (#FE), A   ; 11
```

После того, как был выслан пилотирующий сигнал, на порт MIC идет посылка переднего фронта синхроимпульса. Задержка в цикле SA-SYNC1 составляет 667 тактов, после чего содержимое аккумулятора передается в порт 254.

```
25          LD A, #0D      ; 7
26          LD B, #37      ; 7
27  SA-SYNC2  DJNZ SA-SYNC2 ; 10, 8
28          OUT (#FE), A   ; 11
```

После 735 тактов идет высылка заднего фронта синхроимпульса, устанавливающего цвет бордюра голубым, напряжение на разъеме MIC - высоким. Изменение цвета и управляющего напряжения осуществляется занесением в аккумулятор числовой константы 0DH. Задержка осуществляется загрузкой в регистр B временной константы 37H. После выполнения задержки в цикле SA-SYNC2, значение аккумулятора пересылается в компьютерный порт 254.

```
29          LD BC, #3B0E   ; 10
30          EX AF, A'F'     ; 4
31          LD L, A         ; 4
32          JP SA-START     ; 10
```

Загрузка константы 3B0EH в регистр BC представляет собой загрузку двух различных чисел в отдельные регистры данной регистровой пары. 3BH - это временная постоянная, загружаемая в регистр B, а 0EH - придает начальное значение регистру C, в котором будет сохраняться состояние последнего напряжения в гнезде MIC и цвет бордюра (начальное состояние низкое напряжение и желтый цвет бордюра). Воспроизведенный тип блока помещается в регистр L и посылается на ленту как первый байт данного блока. После этого осуществляется безусловный переход на метку SA-START.

|    |         |                 |         |
|----|---------|-----------------|---------|
| 33 | SA-LOOP | LD A, D         | ; 4     |
| 34 |         | OR E            | ; 4     |
| 35 |         | JR Z, SA-PARITY | ; 10, 7 |

Процедура SA-LOOP - это начало главного цикла, записывающего следующий байт. Если счетчик байтов, который организуется в регистре DE, достиг нуля, то остается записать только байт четности. Проверка на ноль осуществляется типичным способом, характерным для микропроцессора Z-80. В аккумулятор загружаем содержимое регистра D и, используя содержимое регистра E, выполняем операцию логическое "ИЛИ". Результат операции может быть равен нулю только в том случае, если оба байта равны нулю, поэтому нам только остается проверить содержимое флага Z флагового регистра процессора для точного установления результата выполнения данной операции.

|    |           |              |      |
|----|-----------|--------------|------|
| 36 |           | LD L, (IX+0) | ; 19 |
| 37 | SA-LOOP-P | LD A, H      | ; 4  |
| 33 |           | XOR L        | ; 4  |
| 39 | SA-START  | LD H, A      | ; 4  |

Данный отрезок программы служит для получения следующего байта для высылки его на магнитофон. Он также используется для запуска созданного в регистре H байта четности. Регистр H иницируется байтом, определяющим тип записываемого блока. (Методика построения байта четности в ходе операций записи и загрузки была нами рассмотрена в главе 1 данного тома).

|    |  |              |      |
|----|--|--------------|------|
| 40 |  | LD A, 1      | ; 7  |
| 41 |  | SCF          | ; 4  |
| 42 |  | JP SA-8-BITS | ; 10 |

Загружаемое в аккумулятор значение единицы установит напряжение в гнезде MIC на высокий уровень и цвет бордюра станет синим. Принудительное включение флага C флагового регистра F будет выполнять роль указателя, позволяющего утверждать, что выслано уже восемь битов данного байта.

В завершение процедуры мы осуществляем безусловный переход на метку SA-8-BITS.

|    |           |              |      |
|----|-----------|--------------|------|
| 43 | SA-PARITY | LD L, H      | ; 4  |
| 44 |           | JR SA-LOOP-P | ; 12 |

Процедура SA-PARITY готовит высылку байта четности, как последнего байта, записываемого на магнитофонную ленту. Для этого в регистр L, который обычно используется для хранения выслаемого байта, записываем байт четности, который находится в регистре H. После этого осуществляется переход к процедуре SA-LOOP-P, которая, собственно, и осуществляет запись.

|    |         |          |     |
|----|---------|----------|-----|
| 45 | SA-BIT2 | LD A, C  | ; 4 |
| 46 |         | BIT 7, B | ; 8 |

К процедуре SA-BIT2 мы обращаемся при втором проходе цикла, записывающего одиночный бит (это осуществляется перед высылкой заднего фронта сигнала). Проверка седьмого бита в регистре B имеет цель только обнуление флага C, сигнализируя таким образом второй проход.

|    |         |               |         |
|----|---------|---------------|---------|
| 47 | SA-BIT1 | DJNZ SA-BIT1  | ; 10, 8 |
| 48 |         | JR NC, SA-OUT | ; 12, 7 |

Следующая процедура формирует главный цикл задержки между последовательными фронтами сигналов. Флаг C несет в себе действительное значение выслаемого бита.

|    |        |             |         |
|----|--------|-------------|---------|
| 49 |        | LD B, #42   | ; 7     |
| 50 | SA-SET | DJNZ SA-SET | ; 10, 8 |

Перед высылкой логической единицы нам необходимо задержать работу на В55 дополнительных тактов. Для этого в регистр В мы заносим временную константу 42H, которая организует нам необходимый цикл задержки.

|    |        |                |         |
|----|--------|----------------|---------|
| 51 | SA-OUT | OUT (#FE), A   | ; 11    |
| 52 |        | LD B, #3E      | ; 7     |
| 53 |        | JR NZ, SA-BIT2 | ; 12, 7 |

Данная процедура организует высылку заднего фронта сигнала. Это происходит после высылки переднего фронта путем занесения в регистр В константы 3EH и вызова подпрограммы SA-BIT2.

|    |           |                |      |
|----|-----------|----------------|------|
| 54 |           | DEC B          | ; 4  |
| 55 |           | XOR A          | ; 4  |
| 56 |           | INC A          | ; 4  |
| 57 | SA-8-BITS | RL L           | ; 8  |
| 58 |           | JP NZ, SA-BIT1 | ; 10 |

После модификации временной константы в регистре В путем уменьшения ее на единицу, обнуляется флаг Z и в аккумулятор вносится значение единицы (бордюр становится синим, а напряжение в MIC - низкого уровня). Затем, после увеличения аккумулятора на единицу, во флаг C перемещается следующий бит из регистра L. Если регистр L отличен от нуля, то он знаменует собой начало цикла SA-BIT1, причем первый вход в этот цикл выполняется с установленным флагом C, в то время как все остальные - с обнуленным. Благодаря этому только после восьми проходов регистр L достигает значения нуля.

|    |  |           |      |
|----|--|-----------|------|
| 59 |  | DEC DE    | ; 6  |
| 60 |  | INC IX    | ; 10 |
| 61 |  | LD B, #31 | ; 7  |

Данный отрезок программы осуществляет модификацию счетчика путем уменьшения на единицу содержимого регистра DE и увеличения на единицу содержимого регистра IX. Следующим этапом идет подготовка временной константы путем засылки в регистр В значения 31H.

|    |  |             |         |
|----|--|-------------|---------|
| 62 |  | LD A, #7E   | ; 7     |
| 63 |  | IN A, (#FE) | ; 11    |
| 64 |  | RRA         | ; 4     |
| 65 |  | RET NC      | ; 11, 5 |

Здесь выполняется контроль нажатия клавиши BREAK и, если она нажата, то программа передает управление на адрес 053FH процедуре обработки возврата в БЕЙСИК SA/LD-RET.

Проверка нажатия клавиши начинается с чтения в аккумулятор значения из порта FEN. После получения значения осуществляется сдвиг аккумулятора вправо с использованием флага переноса, в который попадает нулевой бит аккумулятора. Этот бит и характеризует тот факт, была ли нажата клавиша BREAK. Завершает этот участок программы процедура условного возврата, которая, анализируя состояние флага переноса либо продолжает выполнение программы по нижеследующему маршруту, либо осуществляет возврат к процедуре выхода в БЕЙСИК.

|    |  |                |      |
|----|--|----------------|------|
| 66 |  | LD A, D        | ; 4  |
| 67 |  | INC A          | ; 4  |
| 68 |  | JP NZ, SA-LOOP | ; 10 |

Контроль счетчика байтов осуществляет предварительную проверку на достижение нуля в регистре D. Для этого мы передаем в аккумулятор значение регистра D, увеличиваем аккумулятор на единицу и проверяем содержимое регистра A на наличие нуля. Переход к метке SA-LOOP выполняется также в случае достижения регистром DE значения нуля, так как необходимо еще записать байт четности.

```
69          LD B, #3B          ; 7
70  SA-DELAY DJNZ SA-DELAY     ; 10, 8
71          RET                ; 10
```

После достижения в регистре DE значения #FFFF и короткой паузы управление передается к процедуре возврата к БЕЙСИКУ SA/LD-RET. Как Вы знаете, в компьютере при вычитании значений из регистра, содержащего ноль, у нас получается значение FFFFH, FFFEH, FFFDH и так далее. Пауза организуется с помощью занесения в регистр временной константы 3BH и создания цикла на основе этого значения. Последний байт процедуры SA-BITS помещается в ячейке с адресом 053EH.

Мы с Вами рассмотрели стандартные процедуры считывания и записи данных на магнитную ленту. Осталось рассмотреть процедуру обработки возврата в БЕЙСИК.

Эта процедура имеет название SA/LD-RET и используется для выхода в БЕЙСИК из подпрограмм загрузки или записи в случае успешного или неуспешного их завершения, или же нажатие клавиши BREAK. Она необходима для того, чтобы гарантировать возврат к БЕЙСИКУ из процедуры записи/считывания, как в случае их правильного окончаний, так и в случае появления ошибок или прерывания оператора, то есть клавишей BREAK. Время выполнения инструкции для этой процедуры не существенно.

```
1  SA/LD-RET  PUSH AF
2             LD A, (#5C48)
3             AND #38
4             RRCA
5             RRCA
6             RRCA
7             OUT (#FE), A
```

Помещение флагового регистра на стек имеет целью сохранение флага C: обнуление его означает прерывание клавишей BREAK или ошибку чтения. Затем в аккумулятор загружается значение системной переменной BORDER и на основе ее третьего, четвертого и пятого бита воспроизводится цвет бордюра экрана, который был перед запуском соответствующей процедуры, обслуживающей магнитофон. Это осуществляется путем высылки содержимого аккумулятора в компьютерный порт 254.

```
8             LD A, #7E
9             IN A, (#FE)
10            RRA
11            EI
12            JR C, SA/LD-END
```

В этом отрезке программы еще раз проверяется, была ли нажата клавиша BREAK для того, чтобы можно было выдать соответствующее информационное сообщение. После этого мы включаем маскируемое прерывание и осуществляем условный переход в зависимости от состояния флага переноса C, который изменяется при нашей проверке нажатия клавиши BREAK. В случае, если клавиша BREAK была действительно нажата, мы осуществляем переход к процедуре SA/LD-END.

```
13  REPORT-D  RST #08
14          DEFB #0D
```

Эти команды вызывают возврат к БЕЙСИКУ с выдачей сообщения:

D-BREAK-CONT REPEATS.

Данное сообщение, как Вы помните, выдается, если программа прерывается оператором с помощью клавиши BREAK. Это только один из типов сообщений, которые могут выдаваться в подобном случае. Оно говорит, что клавиша BREAK была нажата во время действия периферийной операции. Действие CONTINUE после этого оператора обычное.

```
15    SA/LD-END    POP AF
16                      RET
```

Первым делом мы извлекаем из стека содержимое флагового регистра. Это необходимо для воспроизведения флага переноса и правильного возврата к процедуре запуска. Следует отметить, что последний байт процедуры SA/LD-RET занимает ячейку за адресом 0555H.

Мы с Вами закончили рассмотрение процедур чтения и записи на магнитную ленту в формате "Спектрума". Данный материал мы старались построить так, чтобы он был легко доступен для начинающих и чтобы в то же время и профессионал мог почерпнуть в нем что-то новое.

В то же время, это был вынужденный шаг, поскольку цель, которую мы ставили перед собой - освоить защиту программ от копирования была бы не достигнута, если бы Вы не смогли разобраться в принципах действия стандартных загрузочных процедур, зашитых в ПЗУ "Спектрума". И, тем не менее, знание одной лишь работы стандартных подпрограмм Вам будет явно недостаточно для написания мощных защит от копирования. В следующей главе мы рассмотрим конкретную программу защиты от копирования информации на магнитной ленте. Это лишь один из многих способов защиты от копирования и мы надеемся, что читатель отнесется к изучению нашего материала творчески, то есть постарается не только вникнуть в суть публикации, но и разработает свои собственные более мощные программы.

### Глава 3. Методы защиты от копирования для ПЭВМ "Спектрум".

#### **1. Методика создания оригинальной процедуры записи информации на магнитную ленту.**

Информация, приведенная в предыдущих главах, поможет Вам поближе познакомиться со структурой хранения информации на магнитной ленте в формате "Спектрума". Однако, этого еще не достаточно для написания мощных программ защиты от копирования и тем более для создания оригинальных защищающих процедур, поскольку необходимо предварительно ознакомиться с основными существующими приемами такой защиты.

Как Вы помните, время, которое затрачивает процессор на обработку одной машинной операции несоизмеримо мало по сравнению с минимально допустимым временем, за которое можно закодировать один бит информации на магнитной ленте. Поэтому для того, чтобы правильно записать информацию на магнитный носитель, процессор вынужден делать временные задержки, оформляемые в программе в виде циклов с предварительно заданными временными константами, хранящимися в регистре В.

Таким образом, одно из направлений в создании собственной защиты - это изменение временных констант в сторону незначительного уменьшения или увеличения их значений, что позволяет сделать непригодными стандартные процедуры чтения данных, записанных в новом формате.

Для того, чтобы обеспечить надежную защиту, нет необходимости в полном изменении всех временных констант - на практике достаточно изменения одной из них. Мы можем изменить временную константу задержки, характерную для стандартного пилотирующего сигнала, стандартного синхроимпульса, а также характерную для

стандартной записи логического нуля или единицы. Все это приведет к тому, что созданная и выгруженная нами программа не сможет правильно считываться стандартными процедурами, встроенными в ПЗУ компьютера.

Для того, чтобы облегчить Вашу работу по созданию собственных защит, мы хотим предложить Вам достаточно удобный способ создания программ подобного рода. Если Вы внимательно читали предыдущие главы, то наверняка обратили внимание на то, что процедуры записи и чтения состоят из большого числа подпрограмм, которые периодически вызываются в ходе работы основной процедуры. Для создания своей процедуры записи и чтения Вам фактически необходимо лишь создать головную программу, которая внесет необходимые изменения во временные константы. Остальные подпрограммы можно использовать из того стандартного набора, который уже "зашит" в ПЗУ.

Возможно, что этот путь устроит не всех и кто-то захочет написать собственные процедуры записи и чтения программ в полном объеме самостоятельно. Что ж, мы не ставим своей целью навязывать Вам свои приемы программирования, а лишь хотим ознакомить Вас с основными методами защиты.

Итак, мы предлагаем Вам для рассмотрения достаточно простой способ защиты, а именно: изменение временной константы задержки для пилотирующего сигнала. Создание собственного, отличного от стандартного, пилотирующего сигнала, позволило бы Вам защитить свою программу от несанкционированного копирования уже в начальный момент загрузки. Это объясняется тем, что все программы, записанные на магнитную ленту в формате Спектрума начинаются именно с пилотирующего сигнала.

Для создания защищенной от копирования программы, Вам необходимо исключить возможность загрузки данной программы в копировщик. Поскольку копировщик использует стандартные процедуры записи и чтения информации с магнитной ленты, то Вам необходимо блокировать его работу путем создания файла, записанного в одном из созданных Вами форматов, отличном от стандартного формата "Спектрума". Чтобы защитить программу от копирования, на практике нет необходимости в полном изменении формата записи всех блоков данных, хранимых на магнитной ленте. Для этого достаточно записать в измененном формате всего лишь один небольшой блок данных, который хранит в себе информацию, без которой не может начаться выполнение программ.

Поскольку данный файл записан в измененном формате, то его невозможно будет скопировать с помощью копировщика, а так как он несет в себе жизненно важные сведения, необходимые для правильной работы программы, то без него программа не сможет нормально функционировать, даже если попытаться отбросить его в процессе копирования.

Рассмотрим принцип формирования пилоттона в формате "Спектрума".

|           |                                                                                                                          |
|-----------|--------------------------------------------------------------------------------------------------------------------------|
| SA-BYTES  | LD HL, #053F<br>PUSH HL<br>LD HL, #1F80<br>BIT 7, A<br>JR Z, SA-FLAG<br>LD HL, #0C98                                     |
| SA-FLAG   | EX AF, A'F'<br>INC DE<br>DEC IX<br>DI<br>LD A, #02<br>LD B, A                                                            |
| SA-LEADER | DJNZ SA-LEADER<br>OUT (#FE), A<br>XOR #0F<br>LD B, #A4<br>DEC L<br>JR NZ, SA-LEADER<br>DEC B<br>DEC H<br>JP P, SA-LEADER |

Перед входом в данную процедуру предполагается, что регистры хранят информацию о записываемом файле. В регистре DE находится длина записываемого блока, в регистре IX - адрес первого байта, а в аккумуляторе - число, которое определяет тип записываемого блока, причем ноль выставляется для заголовка, а 255(0FFH) - непосредственно для блока данных.

После начала работы процедуры SA-BITS идет подготовка к формированию пилотирующего сигнала. В регистр HL могут быть занесены два числа, определяющие длительность пилотирующего сигнала. 1F80H заносится в регистр HL в случае записи заголовка - и это соответствует приблизительно пяти секундам звучания пилотирующего сигнала, а значение 0C98H заносится в данный регистр в случае записи непосредственно блока данных, что соответствует приблизительно двум секундам звучания пилотирующего сигнала. Какой тип значения заносить в регистр HL определяется путем проверки содержимого аккумулятора, точнее, его седьмого бита.

Формирование пилотирующего сигнала осуществляется процедурой SA-LEADER. Временная задержка осуществляется с помощью занесения в регистр В значения A4H. Таким образом, путем изменения содержимого аккумулятора, мы достигаем периодического перепада напряжения на выходном разъеме, связывавшем Спектрум с магнитофоном и служащим для записи информации. Если в этой процедуре значение, заносимое в регистр В, то Вам удастся сформировать свой собственный пилотирующий сигнал, отличный от стандартного, который позволит защитить Вашу программу от копирования.

После формирования нестандартного пилотирующего сигнала остается продолжить запись Вашей программы на магнитную ленту. Одним из способов осуществления этого является переход к стандартной подпрограмме SA-SYNC1, которая, если Вы правильно оформили свою процедуру записи, то стандартная подпрограмма сама закончит работу по сохранению информации на магнитном носителе и возвратится в операционную среду "Спектрума".

Теперь Вы видите, что нет ничего сложного в том, чтобы составить свою собственную программу по записи информации на магнитофон в формате "Спектрума". Для этого достаточно лишь немного модернизировать одну из стандартных процедур записи, в нашем случае SA-LEADER. Остальные процедуры записи можно использовать стандартные и, лишь когда Вы более подробно познакомитесь со структурой встроенных процедур, Вы сможете начать писать собственные программы дальнейшей обработки информации и возврата в операционную среду. Мы предлагаем Вам начинать с простейшего и постепенно совершенствовать свои приемы программирования.

## **2. Методика создания оригинальной процедуры чтения информации с магнитной ленты.**

Мы с Вами рассмотрели методику создания оригинальных процедур записи информации на магнитофон, используемую при защите информации от копирования. При ее рассмотрении Вам был предложен метод, позволяющий ускорить подобного рода разработки за счет использования встроенных в "Спектрум" процедур. Однако, такой метод не всегда приемлем, и поэтому, приступая к рассмотрению процедуры чтения информации с магнитной ленты, мы остановим свой выбор на создании полноценной процедуры выполняющей все функции чтения информации с магнитофона.

Процедуры чтения информации с магнитной ленты, являются более сложными в сравнении с аналогичными процедурами записи. Это объясняется тем, что основа действия процедур заключается в многократной проверке шестого бита порта 254 и в определение интервала между двумя следующими друг за другом фронтами. Таким образом, при чтении блока с ленты необходимым является точное определение интервалов между фронтами сигналов. Работу по определению этих интервалов выполняет вспомогательная процедура LD-EDGE-2, размещенная по адресу 05E3H в ПЗУ "Спектрума". Эта точка входа в ПЗУ используется, когда необходимо установить, сколько времени прошло от момента вызова

процедуры до обнаружения двух, следующих одна за другим смен уровней сигналов в гнезде EAR и на сколько обе эти смены уровней сигнала соответствуют по времени допустимым пределам требуемых временных интервалов.

Второй важной точкой входа в ПЗУ является стандартная процедура LD-EDGE-1, расположенная по адресу 05E7H. Она позволяет определить, сколько времени прошло с момента вызова процедуры до извлечения первого фронта сигнала. Вызывается она после помещения в регистр В временной константы, ограничивающей количество проверок порта FEN. Превышение допустимого времени ожидания смены сигнала или определение нажатия клавиши BREAK сопровождается обнулением флага переноса. Для распознавания каждой из этих ситуаций служит флаг нуля. Если флаг Z обнулен, то в ходе работы программы была нажата клавиша BREAK, а если он включен, то произошло превышение временного интервала.

В рассматриваемой нами методике изменения стандартного пилотирующего сигнала Вам необходимо как можно более точно рассчитать необходимое время задержки для четкого поиска фронтов модернизированного пилоттона. Как мы и обещали, ниже приводится специализированная программа для загрузки с ленты файла, начинающегося с модернизированного пилоттона.

```

                                ORG 50000
                                LD DE,256
                                LD IX,23296
                                LD A,255
1      LD-BYTES      INC D          ;4
2                      EX AF,A'F'    ;4
3                      DEC D          ;4
4                      DI             ;4
5                      LD A,#0F       ;7
6                      OUT (#FE),A    ;11
7                      LD HL,#053F    ;10
6                      PUSH HL
9                      IN A,(#FE)      ;11
10                     RRA             ;4
11                     AND #20        ;7
12                     OR #02         ;7
13                     LD C,A         ;4
14                     CP A           ;4
15      LD-BREAK     RET NZ          ;11,7
16      LD-START     CALL LD-EDGE-1  ;17
17                     JR NC,LD-BREAK ;12,7
18                     LD HL,#0415    ;10
19      LD-WAIT      DJNZ LD-WAIT    ;10,6
20                     DEC HL         ;6
21                     LD A,H         ;4
22                     OR L           ;4
23                     JR NZ,LD-WAIT  ;12,7
24                     CALL LD-EDGE-2 ;17
25                     JR NC,LD-BREAK ;12,7
26      LD-LEADER    LD B,#9C        ;7
27                     CALL LD-EDGE-2 ;17
28                     JR NC,LD-BREAK ;12,7
29                     LD A,#C6       ;7
30                     CP B           ;4
31                     JR NC,LD-START ;12,7
32                     INC H          ;4
33                     JR NZ,LD-LEADER ;12,7
34      LD-SYNC      LD B,#C9        ;7
35                     CALL LD-EDGE-1 ;17
36                     JR NC,LD-BREAK ;12,7
37                     LD A,B         ;4
38                     CP #D4         ;7
39                     JR NC,LD-SYNC  ;12,7
40                     CALL LD-EDGE-1 ;17
41                     RET NC         ;11,5

```



|    |           |                 |         |
|----|-----------|-----------------|---------|
| 42 |           | LD A,C          | ; 4     |
| 43 |           | XOR #03         | ; 7     |
| 44 |           | LD C,A          | ; 4     |
| 45 |           | LD H,#00        | ; 7     |
| 46 |           | LD B,#B0        | ; 7     |
| 47 |           | JR LD-MARKER    | ; 12    |
| 48 | LD-LOOP   | EX AF,A'F'      | ; 4     |
| 49 |           | JR NZ,LD-FLAG   | ; 12, 7 |
| 50 |           | JR NC,LD-VERIFY | ; 12, 7 |
| 51 |           | LD (IX+C),L     | ; 19    |
| 52 |           | JR LD-NEXT      | ; 12    |
| 53 | LD-FLAG   | RL,C            | ; 4     |
| 54 |           | XOR L           | ; 4     |
| 55 |           | RET NZ          | ; 11, 5 |
| 56 |           | LD A,C          | ; 4     |
| 57 |           | RRA             | ; 4     |
| 58 |           | LD C,A          | ; 4     |
| 59 |           | INC DE          | ; 4     |
| 60 |           | JR LD-DEC       | ; 12    |
| 64 | LD-NEXT   | INC IX          | ; 10    |
| 65 | LD-DEC    | DEC DE          | ; 6     |
| 66 |           | EX AF,A'F'      | ; 4     |
| 67 |           | LD B,#B2        | ; 7     |
| 68 | LD-MARKER | LD L,01         | ; 7     |
| 69 | LD-8-BITS | CALL LD-EDGE-2  | ; 17    |
| 70 |           | RET NC          | ; 11, 5 |
| 71 |           | LD A,#CB        | ; 7     |
| 72 |           | CP B            | ; 4     |
| 73 |           | RL L            | ; 4     |
| 74 |           | LD B,#B0        | ; 7     |
| 75 |           | JP NC,LD-8-BITS | ; 10    |
| 76 |           | LD A,H          | ; 4     |
| 77 |           | XOR L           | ; 4     |
| 78 |           | LD H,A          | ; 4     |
| 79 |           | LD A,D          | ; 4     |
| 80 |           | OR E            | ; 4     |
| 81 |           | JR NZ,LD-LOOP   | ; 12, 7 |
| 82 |           | LD A,H          | ; 4     |
| 63 |           | CP #01          | ; 7     |
| 84 |           | RET             | ; 10    |
|    |           |                 |         |
| 1  | LD-EDGE-2 | CALL LD-EDGE 1  | ; 17    |
| 2  |           | RET NC          | ; 11, 5 |
| 3  | LD-EDGE-1 | LD A,#16        | ; 7     |
| 4  | LD-DELAY  | DEC A           | ; 4     |
| 5  |           | JR NZ,LD-DELAY  | ; 12, 7 |
| 6  |           | AND A           | ; 4     |
| 7  | LD-SAMPLE | INC B           | ; 4     |
| 8  |           | RET Z           | ; 11, 7 |
| 9  |           | LD A,#7F        | ; 7     |
| 10 |           | IN A,(#FE)      | ; 11    |
| 11 |           | RRA             | ; 4     |
| 12 |           | RET NC          | ; 11, 7 |
| 13 |           | XOR C           | ; 4     |
| 14 |           | AND #20         | ; 7     |
| 15 |           | JR Z,LD-SAMPLE  | ; 12, 7 |
| 16 |           | LD A,C          | ; 4     |
| 17 |           | CPL             | ; 4     |
| 18 |           | LD C,A          | ; 4     |
| 19 |           | AND #07         | ; 7     |
| 20 |           | OR #08          | ; 7     |
| 21 |           | OUT (#FE),A     | ; 11    |
| 22 |           | SCF             | ; 4     |
| 23 |           | RET             | ; 10    |

Приведенная выше программа полностью соответствует стандартной процедуре загрузки информации и может осуществлять чтение файлов в формате "Спектрума". Если Вы внимательно присмотритесь к данной программе в машинных кодах, то легко обнаружите, что за загрузку пилоттона ответственна лишь одна процедура LD-LEADER (в нашей программе ей соответствуют строки 26-31). Здесь временная константа 9CH позволяет процедуре LD-EDGE-2 искать два фронта сигнала за время, не превышающее 7000 тактов работы процессора. С другой стороны, после их нахождения проверяется, прошло ли с момента вызова процедуры LD-EDGE-2 до появления заднего фронта не менее, чем 3366 тактов работы процессора. Этим способом отличаются фронты пилотирующего сигнала от значительно более близких пар фронтов информационных сигналов.

Загружаемая в регистр В временная константа 9CH перед вызовом процедуры LD-EDGE-2 формирует необходимую задержку для правильной работы этой стандартной процедуры. После того, как мы возвращаемся из процедуры LD-EDGE-2, мы первым делом проверяем, была ли нажата клавиша BREAK и лишь в следующий момент сравниваем содержимое регистра В с предварительно загруженной в аккумулятор константой С6Н. Команда сравнения, применяемая в этом случае, позволяет нам сравнить содержимое аккумулятора с числом, содержащимся в регистре В. В результате операции числовые значения не изменяются. Результат сказывается только на флагах регистра F. Если содержимое аккумулятора больше сравниваемого числа или равно ему, то флаг переноса будет выключен. Если же оно меньше, то флаг переноса будет равен единице.

Как Вы помните, процедура LD-EDGE-2 позволяет установить, сколько времени прошло от момента ее вызова до обнаружения двух, следующих друг за другом смен уровней сигналов в магнитофонном гнезде. Для этого в регистр В заносится константа, определяющая количество проверок магнитофонного гнезда EAR. Если изменение фронта не обнаружено, то мы выходим из процедуры LD-EDGE-2 по обнулению регистра В. Если же мы обнаружили смену фронтов в тот момент, когда регистр В еще не достиг своего нулевого значения, мы выходим из вышеуказанной процедуры и сравниваем оставшееся в регистре В значение с константой, загружаемой в аккумулятор. Это необходимо для того, чтобы проверить, прошло ли с момента вызова LD-EDGE-2 до появления заднего фронта не менее, чем 3366 тактов работы процессора.

Для того, чтобы создать собственную процедуру чтения файлов с магнитофона, Вам необходимо рассчитать, какие значения заносить в регистр В перед вызовом LD-EDGE-2 из подпрограммы LD-LEADER и с каким значением аккумулятора необходимо сравнивать остаток регистра В после отыскания фронта сигнала.

Все вышесказанное касается лишь того случая, когда Вы будете использовать метод модернизации пилотирующего сигнала, чтобы добиться его отличия от стандарта. Для каждого варианта такого сигнала Вам необходимо создавать собственную оригинальную процедуру загрузки, то есть подсчитывать время задержки.

Мы с Вами рассмотрели методику защиты от копирования с использованием нестандартного пилотирующего сигнала. Однако самой важной целью этой работы было показать Вам, что Вы в состоянии сами написать как процедуру записи программы на магнитную ленту, так и процедуру чтения. Это еще ближе знакомит Вас с работой компьютера, помогает преодолеть естественную неуверенность перед машинным кодом и, как мы надеемся, поможет Вам реализовать по новому свои способности в программировании.

# СЕКРЕТЫ TRDOS

Алексеев А.Г.

Итак, Вы обзавелись БЕТА-ДИСК интерфейсом, а также чистыми дискетами. Если есть у кого переписывать готовые программы на дискетах - Вы на некоторое время будете избавлены от прочих забот, наслаждаясь тем, как лихо проходит загрузка игр. Вы даже поначалу не будете обращать внимание на то, что где-то подпорчена заставка (нет двух нижних строк), а где-то на экран загружается какая-то цветная дрянь. Правда, после этого игра стартует нормально, но выглядит это как-то кустарно. Кроме того, наверняка есть у Вас игры на магнитной ленте, которые Вы относите к самим "игручим", а их дисковых версий как-то не оказалось у Ваших друзей. В общем, постоянные "житейские" заботы будут каждый день подталкивать Вас на более глубокое изучение TR-DOS. Поэтому я поделюсь тем опытом, который накопился у меня с момента подключения БЕТА-диска.

Вам предлагается обобщенный материал, собранный из разрозненных источников, а также уникальные данные, полученные в результате экспериментальной работы.

Начальные сведения по дисковой операционной системе TR-DOS достаточно подробно изложены в многочисленных описаниях, распространяемых в достаточном количестве на компьютерных рынках страны. Поэтому в этой статье я буду подразумевать, что владельцы БЕТА-ДИСК интерфейсов уже знакомы с этими материалами.

Первое, что усваивает каждый пользователь TR-DOS, это то, что вместо, например:

```
LOAD "" CODE
```

в Бейсик-строке теперь надо ставить:

```
RANDOMIZE USR 15619 : REM : LOAD "name" CODE
```

а вход в систему TR-DOS из Бейсика производится командой:

```
RANDOMIZE USR 15616
```

Прежде всего, давайте выясним что это за адреса? 15616 и 15619 это же ведь адреса ПЗУ, того самого, которое было до подключения TR-DOS. Не вступаем ли мы в конфликт с машинным кодом основного ПЗУ Спектрума? Оказывается нет. С адреса 15616 в ПЗУ Спектрума расположен символьный набор, начиная с пробела, то есть, в нормальном режиме работы "Спектрума" никогда не происходит выполнение машинного кода в этих адресах. В случае БЕТА-ДИСК интерфейса, при попадании на этот адрес срабатывает селектор адреса, который подключает теневое ПЗУ и дальнейшая работа происходит в нем. В теновом ПЗУ с системой TR-DOS по этим адресам располагаются команды, передающие управление в соответствующие точки ПЗУ TR-DOS.

Рассмотрим теперь более детально, как же устроена TR-DOS.

## КАРТА ПАМЯТИ ОЗУ.

Те сведения, которые изложены в "фирменных" руководствах по TR-DOS, можно охарактеризовать как TR-DOS снаружи. Ну, например, в "фирменном" описании сказано, что для работы TR-DOS выделяется дополнительно 112 байт ОЗУ. И еще говорится о том, что в процессе работы TR-DOS динамически выделяется 256 байт, причем происходит это незаметно для пользователя. И все про это. Попробуем раскрыть поподробнее эти фразы, взглянув на TR-DOS изнутри.

Для начала посмотрим, где эти 112 байт ОЗУ выделяются. До инициализации БЕТА-ДИСК интерфейса карта памяти ОЗУ имела следующий вид (только то, что представляет интерес):

```

-----
Экранная память      16384

-----
Буфер принтера       23296

-----
Системные            23552
переменные
-----
Бейсик-программа     23755 <-- PROG

----- <--RAMTOP
Свободно
                        65535 <-- PRAMT
-----

```

При подаче команды RANDOMIZE USR 15616 происходит инициализация БЕТА-ДИСК интерфейса. При этом перестраивается карта памяти. Дополнительные 112 байт ОЗУ выделяется перед областью Бейсик-системы. Системная переменная PROG становится равной  $23755+112=23867$ . Бейсик-программа со всеми Бейсик-переменными и маркером конца передвигается на 112 байт ближе к RAMTOP. А область, отведенная под Бейсик-систему становится, таким образом, меньше на 112 байт:

```

-----
Экранная память      16384

-----
Буфер принтера       23296

-----
Системные            23552
переменные
-----
Системные
переменные TR-DOS
-----
Бейсик-программа     23867 <-- PROG

----- <--RAMTOP
Свободно
                        65535 <-- PRAMT
-----

```

Кроме того, непосредственно в тот момент, когда происходит чтение или запись сектора данных, Бейсик-программа отодвигается еще дальше к RAMTOP. Эту переброску осуществляет TR-DOS и происходит она для пользователя незаметно (однако это может доставить массу проблем при адаптации магнитофонных программ под диск). Бейсик-программа со всеми своими переменными отодвигается еще на 257 байт. Это место освобождается непосредственно для операции чтения-записи сектора данных на диск: 256 байт информации плюс один байт - контрольная сумма сектора. Карта памяти в этот момент имеет следующий вид:

```

-----
Экранная память      16384

-----

Буфер принтера 23296

-----

Системные            23552
переменные
- - - - -
Системные
переменные TR-DOS
- - - - -
257-байтное
пространство
-----
Бейсик-программа  24124 <-- PROG

----- <--RAMTOP
Свободно
                65535 <-- PRAMT
-----

```

По завершении операции чтения-записи Бейсик-программа возвращается на свое прежнее место - в адрес 23867.

Такое изменение адресов не то что происходит незаметно, но, как уже говорилось, может создать массу проблем для адаптации магнитофонных версий программ под диск. Большинство пользователей Спектрумов начинали с магнитофонных вариантов и БЕТА-дискос оснащали свои уже готовые компьютеры. Поэтому, конечно, одной из первых потребностей, с которой Вы сталкиваетесь, (и с которой столкнулся я), является адаптация своих собственных программ под БЕТА-диск. Переделка фирменных игровых программ - это тема для отдельного разговора, однако на рынке существует огромное множество дисковых версий игр. Дешевле пойти и купить то, что надо. Но свои собственные программы Вам придется переделывать самим. Проблемы могут быть вызваны, например, нехваткой памяти в связи с выделением дополнительных 112 байт ОЗУ. Если значение RAMTOP небольшое, порядка 25000, то под Бейсик-программу остается совсем мало места. И программа, которая нормально работает в магнитофонном варианте, может отказаться работать в дисковом. Значение RAMTOP изменить бывает проблематично, поэтому приходится прибегать к другим приемам по экономии памяти. Многие из этих приемов достаточно подробно разбирались на страницах ZX-РЕВЮ. Это, в частности, замена цифровых значений на VAL "... " и другие приемы. Может оказаться так, что в принципе Бейсик-программа работает, но если в программе встречаются команды TR-DOS с префиксом:

```
RANDOMIZE USR 15619: REM :
```

то вместо выполнения их, на экране появляется сообщение об ошибке: "Out of memory". Это значит, что свободного места не хватает при попытке выделить 257-байтное пространство для операции чтения-записи сектора.

Если способы экономии памяти исчерпаны, то возможно придется серьезно подумать над изменением структуры программы в целях увеличения значения RAMTOP. Кроме этого, следует упомянуть еще об одной проблеме. Во многих программах применяется нулевая строка для размещения в ней блоков машинных кодов за оператором REM. Об этом подробно рассказывалось в ZX-РЕВЮ N 9-10 за 1992 год, см. стр.193. Там говорилось, что если строка с REM это начальная строка программы, то свободное место за REM начинается с адреса PROG+5. В случае TR-DOS, это будет 23867+5=23872. Однако, если Вы попытаете загрузить с диска блок кодов по этому адресу при помощи команды TR-DOS:

LOAD "name" CODE 23872

то у Вас ничего не получится. Дело вот в чем. В тот момент когда происходит загрузка (выполнение процедуры чтения сектора данных), Бейсик-програма и оператор REM вместе с ней уже имеют другой адрес - на 257 больший, чем тот, который был. Поэтому загрузку кодов с диска в нулевую строку надо выполнять следующим образом:

LOAD "name" CODE (23872+257)

Тогда блок кодов попадет точно в адрес 23872.

Вышесказанное относится к загрузке блока кодов и адреса между PROG и RAMTOP. То же относится и к записи блока кодов из этого интервала адресов на диск.

## СИСТЕМНЫЕ ПЕРЕМЕННЫЕ TR-DOS.

Продолжая взгляд на TR-DOS изнутри, рассмотрим некоторые из системных переменных, составляющих дополнительные 112 байт ОЗУ. Еще раз упомяну: предполагается, что читатель знаком с описанием TR-DOS в начальном объеме. Сначала кратко.

23752 (#5CC8) - 1 байт - режим работы диска "A"  
23753 (#5CC9) - 1 байт - режим работы диска "B"  
23754 (#5CCA) - 1 байт - режим работы диска "C"  
23755 (#5CCB) - 1 байт - режим работы диска "D"  
23773 (#5CDD) - 8 байт - имя файла в кодах ASCII  
23781 (#5CE5) - 1 байт - тип файла  
23782 (#5CE6) - 2 байта - параметр START  
23784 (#5CE8) - 2 байта - длина файла  
23786 (#5CEA) - 1 байт - объем файла в секторах  
23787 (#5CEB) - 1 байт - номер первого сектора файла  
23788 (#5CEC) - 1 байт - номер первого трека файла  
23798 (#5CF6) - 1 байт - дисковод для временной операции  
23800 (#5CF8) - 1 байт - дисковод при операции с 2 файлами  
23801 (#5CF9) - 1 байт - признак операции READ/VERIFY  
23802 (#5CFA) - 1 байт - время перемещения головки дисковода "A"  
23803 (#5CFB) - 1 байт - время перемещения головки дисковода "B"  
23804 (#5CFC) - 1 байт - время перемещения головки дисковода "C"  
23805 (#5CFD) - 1 байт - время перемещения головки дисковода "D"  
23823 (#5D0F) - 1 байт - код ошибки TR-DOS  
23830 (#5D16) - 1 байт - копия системного регистра  
23833 (#5D19) - 1 байт - номер дисковода  
23838 (#5D1E) - 1 байт - номер файла, если он найден.

Теперь подробнее

С адреса 23752 по 23755 расположены 4 числа, по одному на каждый из возможных четырех подключенных дисководов A, B, C и D. 7-й бит этого числа определяет количество дорожек дисковода: 0 - 40-дорожечный дисковод, 1 - 80-дорожечный дисковод. 1-й бит этого числа равен 1, если дисковод двусторонний. 0-й бит этого числа равен 0, если выполняется команда по использованию 80-дорожечного дисковода, как 40-дорожечного.

Начиная с адреса 23773, 8 байт занимает имя файла в кодах ASCII. При инициализации интерфейса в эти ячейки заносится имя "boot". Так осуществляется подготовка к считыванию с диска программы-загрузчика.

Следующий байт после имени файла, ячейка 23781, определяет его тип. Это один из

четырёх кодов ASCII - "B", "C", "D" или "#".

Следующие два байта, с адреса 23782, это параметр "START". Если тип файла - "C", то это начальный адрес размещения кодов в памяти Спектрума. Если же тип файла "B", то двухбайтное число в этих ячейках является системной информацией, связанной с автостартом Бейсик-программы.

Два байта с адреса 23784 обозначают длину файла в байтах. Один байт по адресу 23786 определяет длину файла в секторах (по 256 байт). Разумеется, должно иметь место выражение:

```
PEEK 23786 = PEEK 23785 + (PEEK 23784<>0)
```

Очевидно, что если младший байт длины файла имеет хоть какое-то значение, отличное от нуля, то должен быть выделен сектор для размещения файла.

Когда происходит поиск файла на диске по заданному имени и типу, считывается содержимое нулевой дорожки. В том случае, если такой файл найден, его номер заносится в ячейку 23838. А в ячейки 23787 и 23788 заносятся соответственно начальные сектор и дорожка файла согласно его расположению на диске. Таким образом, теперь имеется вся необходимая информация для загрузки файла в память.

Если Вы выполняете, например, команду:

```
A> LOAD "B:name"
```

то номер дисководов, указанного Вами в качестве временного (в данном случае это "B"), заносится в ячейку 23798. Для дисковода "A" это 0, "B" - 1, "C" - 2, "D" - 3.

Ячейка 23801 содержит признак операции LOAD/VERIFY. Значение 0 соответствует чтению, а 255 - сравнению файла. Вы можете проделать интересный эксперимент: сделайте POKE 23801,255, а затем подайте команду TR-DOS, например, LOAD "name" CODE. Однако вместо загрузки будет выполнена верификация.

Интересна информация, находящаяся в ячейках 23802...23805. Здесь (по одному байту на каждый дисковод) расположен код, определяющий время перемещения головки дисководов A...D. Число, стоящее там после инициализации - 8. Оно соответствует наибольшей скорости перемещения головки. Если чтение сектора прошло неудачно, то это число увеличивается на единицу, то есть становится 9 - это соответствует меньшей скорости перемещения головки. Она отводится на нулевую дорожку, после этого повторяется чтение. Если опять неудача, то опять уменьшение скорости (PEEK 23802=10) и повторяется чтение и так до PEEK 23802=11. Это наименьшая скорость перемещения головки. Вы можете искусственно увеличить ее после неудачного чтения, подав команду (для дисковода "A", например): POKE 23802,8. (Кстати тот же эффект даст POKE 23802,12.)

Вышесказанное про скорость перемещения головки относится к TR-DOS версии 5.01. В более поздних версиях, например, 5.04s, установлена только быстрая скорость перемещения головки и она не зависит от содержимого ячеек 23802...23805. Хотя при инициализации дисковода туда по-прежнему заносится 8. Это число является признаком того, что инициализация произведена (до инициализации каждого из дисководов A...D в этих ячейках содержится значение 255).

В ячейке 23814 задается число байт, по которым в каталоге диска ищется файл. Начальное значение этого параметра - 9 (8 байт - имя файла плюс 1 байт - его тип).

Ячейка 23823 содержит код ошибки TR-DOS. На коде ошибки необходимо остановиться подробнее. Наберите и выполните следующую Бейсик-программу:

```
10 RANDOMIZE USR 15619 : REM : LOAD "abcde" CODE
20 PRINT PEEK 23823
```

В том случае, если файл "abcde" отсутствует на диске, то сообщения об ошибке не последует: программа остановится с сообщением 0 OK. 20:1. О наличии ошибки свидетельствует только единица на экране. По таблице кодов ошибок (смотри в описании TR-DOS) это соответствует ситуации "нет файла". Код ошибки из Бейсик-программы можно подучить еще и другим способом. Ранее в "ZX-РЕВЮ" мы уже говорили о том, что вместо

RANDOMIZE USR ... вполне можно использовать LET S=USR ... Все это относится точно также и к адресу 15619. Измените нашу программу:

```
10 LET err = USR 15619 : REM : LOAD "abcde" CODE
20 PRINT err
```

Результат работы этой программы будет точно такой же, как и первой. Можете использовать значение err в Бейсик-программе по своему усмотрению, например:

```
20 IF err<>0 THEN PRINT "ERROR"
```

Продолжаем разговор о системных переменных TR-DOS.

Ячейка 23830 это копия системного регистра. Если посмотреть на схему контроллера дисководов, то это тот регистр, который собран на микросхеме 555TM9. Интересная особенность, но у меня пока что нет никаких данных о том, как практически это можно использовать.

Ячейка 23833 определяет номер накопителя A...D (A-0, B-1, C-2, D-3). Можете проделать небольшой эксперимент. Войдите в TR-DOS. Вы увидите приглашение: A>. Теперь выполните: \*b". Приглашение сменится на: B>. Теперь перейдите из TR-DOS в Бейсик, выполнив RETURN, затем сделайте POKE 23833,2 и войдите опять в TR-DOS. Вместо приглашения: B> Вы увидите: C>, так как искусственно поменяли дисковод, изменив содержимое системной ячейки 23833.

Перечислены основные, но далеко не все системные переменные TR-DOS. Если кто-нибудь из читателей обладает более полной информацией, напишите об этом.

Теперь рассмотрим подробнее вопрос о том, как информация хранится на дискете.

## ФОРМАТ ЗАПИСИ НА ДИСК.

Данные на дискету записываются в виде секторов, имеющих объем 256 байт. Вдвое меньший, по сравнению с IBM PC объем одного сектора позволяет несколько сэкономить потери из-за недоиспользования места, так как длина файла не кратна объему одного сектора. При большом числе коротких файлов это дает определенный выигрыш.

Предположим, что имеется двусторонний 80-дорожечный дисковод (во всяком случае, таких, наверное, большинство). Если Вы отформатируете дискету, то при завершении форматирования увидите, что на дискете имеется 2544 свободных сектора. Но 80 дорожек умножить на 16 секторов получится 2560 секторов. Где остальные? Дело в том, что нулевая дорожка дискеты 16 секторов - является системной. А файлы могут располагаться начиная с 1-й дорожки по 159 (так как запись двусторонняя).

Информация на нулевую дорожку записывается в строго определенном формате. Эти 16 секторов поделены на две части. В первой части - сектора с 0 по 7 - хранятся заголовки файлов. При этом на один заголовок отводится 16 байт информации. Всего на восьми секторах может быть размещено 128 заголовков файлов. То есть:

8 секторов \* 256 байт = 2048 байт

то же самое, что:

128 файлов \* 16 байт = 2048 байт.

Из максимального числа файлов, кстати, происходит название: "БЕТА-ДИСК 128".

Для тех, кто программирует в машинных кодах может оказаться полезным еще и такой взгляд. Порядковый номер заголовка имеет следующее свойство: старший полубайт этого числа (старшие 4 бита) соответствует номеру сектора, в котором записан заголовок файла, а младший полубайт (младшие 4 бита) соответствует номеру заголовка в секторе.

О том, как в память компьютера считать нулевую дорожку - позже. А сейчас о формате заголовка файла. Он в точности совпадает с 16 ячейками системных переменных с 23773 по 23788. Из 16 байт, отведенных на один заголовок, первые восемь занимает имя файла в ASCII. Следующий байт - тип файла (тоже в ASCII) - B, C, D или #. Следующие два байта - параметр START, далее 2 байта - длина файла, следующий байт - объем файла в секторах и



далее 1 байт - номер начального сектора файла и 1 байт - номер начального трека (дорожки) файла на дискете.

Кроме сказанного, надо добавить следующее. При удалении файла первый символ имени (первый байт заголовка) заменяется на код 01. В таком виде файл считается удаленным, однако, если Вы не делали команду MOVE, то удаленный файл всегда можно восстановить, изменив первый символ его имени в заголовке либо при помощи программ типа "DISK-DOCTOR", либо при помощи всевозможных программ-оболочек типа "MOA-SERVICE". Однако, если удаляется последний файл каталога, то первый символ имени заменяется не на 1, а на 0. По значению 0 определяется конец каталога. Заголовки вновь записываемых файлов будут располагаться в каталоге, начиная с этого места.

Кроме секторов 0...7, в которых располагаются заголовки файлов, есть еще один сектор, восьмой, в котором содержится общая информация по диску. Эта информация расположена не с начала сектора, а ближе к его концу. Если начать нумерацию байтов сектора с 0, то вот содержание байтов, несущих информацию по диску.

Байты 225 и 226 хранят начало свободного места на диске, доступного для записи файлов - сектор и трек соответственно. Сразу же после форматирования дискеты эти байты имеют значения 0 и 1 соответственно (как говорилось выше, запись файлов на дискету начинается с дорожки номер 1).

Байт 227 определяет тип разметки диска. Значения этого байта могут быть следующими.

22 (#16) - 80 дорожек, 2 стороны;

23 (#17) - 40 дорожек, 2 стороны;

24 (#18) - 80 дорожек, 1 сторона;

25 (#19) - 40 дорожек, 1 сторона.

Байт 228 определяет общее количество файлов, записанных на диск. Это число включает в себя и удаленные файлы, так как физически они все равно присутствуют на диске. То есть это все файлы от начала каталога до начала свободного места.

Байты 229 и 230 (младший и старший) определяют количество свободных секторов на диске. Сразу же после форматирования дискеты командой TR-DOS: FORMAT "name", они равны соответственно 240 и 9. То есть  $240+9*256=2544$ . Если для форматирования воспользоваться программой "DCU" Н. Родионова, можно отформатировать диск на максимальное число дорожек, для большинства дискет практически до 83-84 дорожек. При этом байты 229 и 230 сразу же после форматирования могут иметь другие значения, например 2640 или 2672. На первый взгляд может показаться, что на такую дискету удастся записать больше файлов. Но это только на первый взгляд. Попробуйте отформатировать дискету таким образом, а потом записать на нее файл объемом, например, 1 сектор. После записи Вы увидите, что на дискете осталось 2543 свободных сектора ( $2544-1=2543$ ). То есть TR-DOS пересчитывает число свободных секторов при записи файла исходя не из того, что записано в байтах 229 и 230 системного сектора, а из того, что на диске может быть только 80 или 40 дорожек по 16 секторов.

Единственным преимуществом форматирования на максимальное число дорожек пока что видится возможность использования места за 80-й дорожкой для каких-то своих целей, например, там может располагаться пароль, защищающий дискету от копирования. Практически такой способ защиты очень распространен, но это - тема для отдельного разговора.

Если дискета форматировалась при помощи программы "DCU", то байты 223 и 224 содержат максимальное число форматированных секторов, доступных для записи файлов. Сразу же после форматирования дискеты значения этих байтов совпадают со значением байтов 229 и 230. Но значения последних изменяются по мере записи файлов на дискету, а значения байтов 223 и 224 остаются неизменными и используются при работе программы "DCU".

Если дискета форматировалась командой TR-DOS: FORMAT "name", то содержимое байтов 223 и 224 равно 0. Ненулевое значение этих байтов является признаком того, что

дискета форматирована при помощи программы "DCU".

Байт 231 всегда хранит код 16 (#10). Он определяет число секторов на одной дорожке и никогда не меняет своего значения. Это число обозначает принадлежность диска к операционной системе TR-DOS.

Байт 244 содержит число удаленных файлов - то есть тех, первый символ имени которых в каталоге диска имеет значение 01.

Байты с 245 по 252 хранят имя диска в кодах ASCII (всего - 8 байт).

Сектора с 9 по 15 не несут никакой информации, связанной с обычным функционированием файловой системы диска.

Вот это все, что касается нулевой дорожки. К сказанному можно добавить, что искажение системной информации диска и каталога диска широко используется для защиты от копирования, и разговор об этом будет уже в этой статье, но чуть позже. А сейчас мы подошли, пожалуй, к самой главной теме.

## ВЫПОЛНЕНИЕ КОМАНД TR-DOS ИЗ МАШИННОГО КОДА.

Трудно переоценить значение этих сведений для программиста, поэтому сразу к делу.

В фирменном руководстве по TR-DOS описан способ выполнения команд TR-DOS из машинного кода. Суть его в двух словах в следующем. Где-то в памяти Спектрума размещается фрагмент Бейсик-программы - некоторая фиктивная Бейсик-строка, содержащая команду TR-DOS. Формат записи этой Бейсик-строки такой же, каким он является в обычной Бейсик-программе. Затем в регистры процессора и системные переменные Бейсика заносится адрес, в котором находится новый фрагмент "фиктивной" Бейсик-программы. Затем происходит запуск интерпретатора Бейсика для выполнения этой фиктивной строки. Такой способ выполнения машинного кода при помощи Бейсика настолько громоздок и несуразен, что может иметь разве что чисто теоретическое значение и вряд ли кем-то может восприниматься серьезно. Но главное в том, что он выполняет только то, что может быть выполнено из обычного Бейсика и не более того.

Для выполнения команд из машинного кода в TR-DOS существуют специальные возможности. Основная из них - использование подпрограммы TR-DOS с точкой входа 15635 (#3D13). С этого адреса в конечном счете осуществляется переход к подпрограмме 10300 (#283C), которая в зависимости от кода, содержащегося в регистре С процессора передает управление соответствующей процедуре системы TR-DOS. Практически фрагмент выполнения команды TR-DOS из машинного кода может выглядеть следующим образом:

```
...  
LD C, ...  
CALL #3D13  
...
```

Кроме регистра С может потребоваться предварительная установка других регистров процессора.

Рассмотрим кратко основные процедуры, выполняемые в зависимости от содержимого регистра С.

С=0. Восстановление, сброс контроллера, при котором головка отводится на нулевую дорожку и ожидается появление сигнала INTRQ. Выход из режима ожидания может быть произведен при нажатии клавиши BREAK на клавиатуре.

С=1. Инициализация дисководов. Предварительно в регистре А должен быть задан номер накопителя 0...3. Здесь надо сказать, что если TR-DOS стартует при открытом кармане дисковода или без дискеты, то по адресам 23802...23805 заносятся числа 255. Это означает, что дисководы не инициализированы. В результате работы этой процедуры устанавливается время перемещения головки дисковода - заносится число 8 по одному из адресов 23802...23805 в зависимости от номера дисковода. Далее определяется количество дорожек дисковода и результат заносится по одному из адресов 23752...23755 в

соответствии с номером накопителя (первоначальное значение также 255). Если дисковод имеет 40 дорожек, то записывается 0, если 80 дорожек, то 128 (устанавливается 7-й бит). Далее задается номер дисковода для временной операции (берется из регистра А) и это значение заносится по адресу 23798. По адресу 23830 помещается копия системного регистра.

Если время перемещения головки (ячейки 23802...23805) уже установлено, то операции по инициализации дисковода не производятся.

С=2. Позиционирование. Предварительно в регистр А помещается номер дорожки, на которую должна быть установлена головка. При двустороннем дисководе первому физическому треку соответствуют номера 0 и 1, второму - 2 и 3 и т. д.

С=3. Эта процедура помещает содержимое регистра А по адресу 23807, задавая, таким способом номер сектора.

С=4. Процедура помещает содержимое регистра HL по адресу 23808 (2 байта), задавая таким способом адрес буфера.

С=5. Очень важная подпрограмма. Она позволяет читать блок секторов. Для этого необходимо подготовить начальные параметры в регистрах процессора. В регистр В помещается количество читаемых подряд секторов. В том случае, если В=0, то с диска считывается только область заголовка, считывание сектора в память не производится. В регистровую пару DE помещается начало считываемого блока секторов: в D - дорожка, а в E - сектор. В регистре HL должен быть задан адрес начала буфера в памяти Спектрума, куда будет производиться загрузка блока секторов.

С=6. Процедура записи на диск блока секторов. Все параметры аналогичны тем, которые были в предыдущей процедуре.

С=7. Вывод каталога диска (аналогично команде CAT TR-DOS). В регистре А должен быть задан канал для вывода каталога. Для вывода на дисплей, например, в регистре А должно быть 2 (кстати, при этом будет очищен экран). Для вывода на принтер - 3. Следует также сказать, что перед выполнением этой подпрограммы можно задать номер накопителя для временной операции, поместив его по адресу 23798. Эта процедура предварительно выполняет процедуру при С=22 - загрузку системного регистра (см. ниже).

С=8. Чтение заголовка файла. В регистре А должен быть номер интересующего файла (с 0 по 127). Из каталога диска 16 байт заголовка файла будут помещены в память, начиная с адреса 23773. Это может быть и удаленный файл, так как не проверяется, есть ли такой файл в действительности.

С=9. Запись в каталог заголовка файла. 16 байт из памяти, начиная с адреса 23773, переписываются в каталог диска на место заголовка файла, номер которого задается в регистре А аналогично предыдущей процедуре.

С=10. Процедура поиска файла. В каталоге диска ищется файл, который по имени и типу совпадает с тем, что задано в памяти, начиная с адреса 23773. Количество байт, на которые ведется поиск, задается по адресу 23814. Начальное значение - 9. В том случае, если файл найден, то его номер будет находиться в регистре С, а также в ячейке 23838. Если файл не найден, то в регистре С будет 128 (7-й бит установлен в 1), содержимое 23838 не изменяется, а в ячейке 23823 будет 255.

С=11. Запись кодового файла. При этом с адреса 23773 задается имя и тип файла, в регистре HL - адрес начала файла в памяти, в регистре DE - длина файла. Эта процедура

выполняет действия аналогичные команде TR-DOS:

SAVE "name" CODE Start, Length.

C=12. Запись Бейсик-программы. С адреса 23773 должны быть заданы имя и тип файла. В том случае, если тип файла отличается от "B", то файл записывается под именем "boot" <B>.

C=14. Загрузка или верификация файла. Имя и тип файла должны быть помещены по адресу 23773. Здесь есть несколько вариантов, в зависимости от значения регистра A. При A=0 адрес загрузки и длина файла берутся из каталога. При A=3 загрузка происходит с адреса, который задается в регистре HL, причем длина загрузки определяется значением регистра DE. При A=255 адрес загрузки берется из HL, а длина загружаемого файла - из каталога диска. Значение по адресу 23801 определяет команду: 0 - LOAD, 255 - VERIFY.

C=18. Удаление файла. Имя и тип файла должны быть заданы с адреса 23773. При этом удаляются все файлы с такими данными. Как уже говорилось выше, если файл не последний, первый символ имени файла заменяется кодом 01.

C=19. Эта процедура перебрасывает 16 байт информации с адреса, указанного в HL в адрес 23773.

C=20. Процедура, обратная предыдущей. 16 байт перебрасываются с адреса 23773 в адрес, указанный в HL.

C=22. Процедура загружает системный регистр интерфейса. Код - в аккумулятор (регистре A), при загрузке к содержимому аккумулятора добавляется #3C.

Для других значений C в различных источниках приводятся противоречивые или весьма смутные сведения, поэтому на них не будем останавливаться. Для любителей покопаться в ПЗУ TR-DOS привожу входные адреса процедур, соответствующие различным значениям регистра C.

|      |       |      |       |
|------|-------|------|-------|
| C=0  | #3D98 | C=13 | #01D3 |
| C=1  | #3DCB | C=14 | #290F |
| C=2  | #3E63 | C=15 | #01D3 |
| C=3  | #3F02 | C=16 | #01D3 |
| C=4  | #3F06 | C=17 | #01D3 |
| C=5  | #1E3D | C=18 | #2926 |
| C=6  | #1E4D | C=19 | #28E0 |
| C=7  | #28D8 | C=20 | #28E3 |
| C=8  | #165C | C=21 | #2739 |
| C=9  | #1664 | C=22 | #1FEB |
| C=10 | #1CF0 | C=23 | #1FF6 |
| C=11 | #28FB | C=24 | #0405 |
| C=12 | #28F2 |      |       |

Для выполнения команд TR-DOS из машинного кода существует еще один способ. Он позволяет обращаться сразу же непосредственно в теневое ПЗУ TR-DOS, в любую его точку. Вы спросите, как же это практически сделать, ведь обратившись, например, по одному из тех адресов, которые приведены выше, мы попадем всего лишь в основное ПЗУ Спектрума. Для этого в системе TR-DOS имеется специальная точка входа с адресом 15663 (#3D2F). Там находится всего лишь одна команда NOP, после чего сразу же идет RET. Дело в том, что при обращении по адресу 15663 селектор адреса БЭТА-ДИСК интерфейса подключает

тенивое ПЗУ, а выполнение команды RET приведет к переходу программы на адрес, записанный на стеке. Поэтому если перед обращением к 15663 на стек поместить нужный адрес, то можно осуществить переход на этот заданный адрес теневого ПЗУ командой: JP 15663.

В качестве практического примера использования машинного кода для выполнения команд TR-DOS приведу простую программу "DIR", которая позволяет выводить на экран параметры файлов (имя, тип, адрес загрузки, длина и т. п.), содержащиеся в каталоге диска. Кроме теоретического, эта программа представляет определенный практический интерес, так как выводит кроме обычной информации еще и данные о начале файлов на диске (команда LIST TR-DOS эту информацию не выводит).

### Программа "DIR".

Машиннокодовая часть программы выглядит следующим образом (адрес размещения - произвольный):

|          |      |           |     |
|----------|------|-----------|-----|
| 21 00 AB | LD   | HL, #AB00 | (1) |
| 11 00 00 | LD   | DE, #0000 | (2) |
| 06 08    | LD   | B, #08    | (3) |
| 0E 05    | LD   | C, #05    | (4) |
| CD 13 3D | CALL | #3D13     | (5) |
| C9       | RET  |           | (6) |

В регистре HL задается (1) адрес буфера, куда будет производиться чтение каталога диска (#AB00=43776). В регистрах D и E задаются начальные дорожка и сектор (2): чтение будет начинаться с 0 сектора 0 дорожки. Число считываемых подряд секторов задается (3) равным 8, столько занимает каталог диска. Затем задается (4) параметр, определяющий процедуру: C=5, то есть чтение блока секторов и далее (5) выполнение подпрограммы TR-DOS. После чего (6) - возврат в вызывающую программу.

После того, как содержимое нулевой дорожки считано в буфер, его можно рассмотреть более подробно. Этим уже займется Бейсик. Программа выглядит следующим образом:

```
2 BORDER 1: PAPER 7: INK 1: CLEAR 39999
3 RESTORE : FOR a=40000 TO 40013: READ b: POKE a, b: NEXT a: DATA 33,0,171,17,0,0,6,8,14,5,
  205,19,61,201
4 GO TO 100
5 RANDOMIZE USR 15619: REM : SAVE "DIR"
6 STOP
20 PRINT AT 0,0: PAPER 6: INK 2;"Name "; PAPER 2: INK 7; "Type" ; PAPER 3: INK 0; " Start
  Length " ; PAPER 4; "Beginn"; PAPER 5;" Len"
22 PRINT BRIGHT 1: INVERSE 1; " ASCII "; INVERSE 0; " B y t e s "; INVERSE 1;"Trk"; INVERSE
  0; " Sector"
24 RETURN
30 FOR a=a TO a+7: LET n$=n$+CHR$ PEEK a: NEXT a: REM name
31 LET t$ = CHR$ PEEK a: REM type
32 LET a=a+1: LET st=PEEK a+256*PEEK (a+1): REM start
33 LET a=a+2: LET len=PEEK a+256*PEEK (a+1): REM length
34 LET a=a+2: LET q=PEEK a: REM length SEC
35 LET a=a+1: LET sec=PEEK a: REM benin SEC
36 LET a=a+1: LET tr=PEEK a: REM begin TRK
37 LET a=a+1: RETURN
100 RANDOMIZE USR 40000
200 LET a=43776
300 GO SUB 20: LET j=2: BEEP .01, j+10
1000 LET n$="": GO SUB 30
1010 PRINT AT i,j: PAPER 6: INK 2; n$: PAPER 2: INK 7; t$;
1020 PRINT PAPER 3: INK 0; TAB 10; st; TAB 16; len; TAB 22; PAPER 4: INK 0; TAB
  (22+(tr<100)); tr; TAB 26; sec; TAB 28; PAPER 5: INK 0; TAB 29; q; TAB 32;
1030 LET j=j+1: BEEP .01, j+10: IF J>=22 THEN PAUSE 0: CLS : GO TO 300
1040 IF a<43776+256*8 THEN GO TO 1000
2000 PRINT #0; INVERSE 1; "DIR O.K.": BEEP .1,26: BEEP .1,20: PAUSE 0: CLS : GO TO 200
```

После набора программы сделайте RUN 5 - это запись программы на диск. Нет необходимости записывать программу на диск с автостартом. Любой загрузчик ("boot") запускает программу с начальной строки. При этом строка 2 формирует в памяти необходимый кодовый блок, после чего происходит чтение восьми секторов нулевой дорожки (строка 100). Считанная информация размещается с адреса 43776 (#AB00). GO SUB 20 в строке 300 - это распечатка "шапки". Переменная j - это номер строки экрана для вывода последующей информации.

Со строки 1000 расположена программа распечатки одного экрана информации. Подпрограмма GO SUB 30 производит расчет параметров для каждого заголовка файла. Первые 8 байт - имя программы (n\$, строка 30). Затем тип (t\$, строка 31). Следующие 2 байта определяют параметр START (st, строка 32). Затем 2 байта - длина файла (len, строка 33). Потом идет 1 байт - длина файла в секторах (q, строка 34). Следующая пара байт начало файла на диске - сектор и дорожка (соответственно sec и tr в строках 35 и 36).

Строки 1010 и 1020 - это распечатка полученных данных по одному заголовку - 16 байтам нулевой дорожки. Строка 1030 - завершающие операции по распечатке заголовка и проверка на достижение конца экрана. Если он достигнут, то очистка экрана и распечатка следующего (со строки 300).

Строка 1040 проверяет, не достигнут ли конец каталога (8 секторов по 256 байт). Если нет, то переход к расчету и печати информации о следующем файле (со строки 1000). Иначе - завершение работы переходом на строку 2000, где программа за циклируется, возобновляя вывод информации с первого заголовка файла.

Теперь предположим, что Вы запустили эту программу, вставив в дисковод какую-нибудь дискету. После того, как на экран будет выведена информация о всех существующих файлах (при этом удаленные файлы будут начинаться со знака "?"), если на дискете больше нет файлов, то вся последующая часть каталога будет состоять из нулей. Чтобы не "прокручивать" вхолостую весь каталог, можно добавить в программу такую строку:

```
1035 IF PEEK a=0 THEN GO TO 2000
```

Так как начало свободного места на диске обозначается нулевым первым кодом имени файла, то как только встретится такой код, вывод будет завершен переходом на 2000 строку.

Как еще можно усовершенствовать программу? Ну, например, можно в конце добавить вывод информации по диску, его имя, число файлов, наличие свободного места и т.д., в общем, все то, что можно "выудить" из восьмого системного сектора. Для этого надо чтобы с нулевой дорожки было считано не 8 секторов, а 9. Для этого в строке 3 после DATA число 8 должно быть заменено на 9. Теперь заодно с каталогом в память будет считан и системный сектор. Попробуйте сами сделать такое усовершенствование, если хотите, расположив эту часть программы, начиная со строки 2000. Вся необходимая информация по расшифровке системного сектора приведена выше.

Для того, чтобы просмотреть новый диск, можно нажать BREAK, затем сделать RUN. Используя блок кодов "ON ERROR GO TO" (см. ZX-РЕВЮ N 5-6 за 1992 год, стр.113), Вы можете еще более усовершенствовать программу: она будет перезапускаться с начальной строки при нажатии BREAK для того, чтобы можно было просмотреть новый диск.

## ПРИЕМЫ ЗАЩИТЫ ОТ КОПИРОВАНИЯ.

До сих пор мы вели разговор о том, как все должно работать, как говорится, "по букварю". Теперь подойдем более критически к некоторым вопросам. Например, длина файла. В 16 байтах заголовка длина файла фигурирует дважды. Действительно, длина файла это два байта: 11-й и 12-й. Кроме того, это 13-й байт - длина файла в секторах. Зачем он нужен? Ведь он же легко рассчитывается из известных 11 и 12 байтов. TR-DOS устроена так, что точная длина файла (11-12 байты) необходима для процедуры загрузки файла в память, а вот при операциях перезаписи файла TR-DOS оперирует именно с 13 байтом,

резервируя на диске столько места, сколько обозначено в 13 байте. Поэтому если искусственно, "вручную", например, при помощи программы типа "DISK-DOCTOR", изменить значение этого байта, задав нулевое значение, то файл будет загружаться и стартовать нормально (11 и 12 байты не изменены), но при перезаписи файла на другую дискету будет переписано ноль секторов, то есть фактически перезаписи не произойдет.

На этом принципе работают некоторые самые простые программы по защите файлов от копирования, например программа "CLOSE". Эта программа обнуляет 13-й байт заголовка у файлов, имеющих расширение "B", задавая, таким образом, нулевую длину файла в секторах. Кроме того, эта программа искажает системную информацию по диску, изменяя значения параметров системного сектора. Искажению подвергаются: начало и количество свободного места на диске, тип диска (вместо двустороннего задается односторонний), количество файлов. Кроме того, в имя первого файла вводятся непечатаемые управляющие коды. Однако при этом не нарушается нормальный старт программ с диска. Такие диски удастся скопировать только целиком при помощи специальных копировщиков типа "диск в диск", когда копируется без изменения содержимое всех 80 дорожек диска. Кстати, изложенные в этой статье сведения позволяют Вам самостоятельно разработать такой копировщик даже на Бейсике, с выполнением одного фрагмента в машинных кодах, аналогично программе "DIR". Пофайловое копирование всего диска при помощи команды TR-DOS: COPY B, а также копирование при помощи программ-оболочек типа "MOA-SERVICE" в этом случае не помогает.

Этот способ защиты широко распространен на рынке программного обеспечения, но пользуются им не разработчики оригинальных программ, а скорее "работники торговли". Их дискеты недостаточно хорошо скомпонованы, чтобы копировать их целиком. Обычно на дискете бывает несколько хороших программ, остальное - мусор для заполнения дискеты. Впрочем, это уже отклонение от темы. Так или иначе, Вы должны быть свободны в вопросах компоновки своих дискет, поэтому я поделюсь с Вами программой, созданной мной в противовес "CLOSE". Она называется "OPEN". На самом деле проблема достаточно примитивна. Испорченную длину файла в секторах легко можно восстановить, вычислив разность между началом этого файла (байты 14 и 15 заголовка) и началом следующего файла. Заодно программа "OPEN" также рассчитывает объем и начало свободного места на диске и восстанавливает всю остальную системную информацию по диску.

### Программа "OPEN".

```
2 BORDER 7: PAPER 7: INK 0: CLEAR 39999
3 RESTORE : FOR a=40000 TO 40012: READ b: POKE a,b: NEXT a:DATA 33,0,171,17,0,0,1,5,9,
    205,19,61,201
4 GO TO 100
5 RANDOMIZE USR 15619: REM : SAVE "OPEN"
6 STOP
100 CLS : PRINT #0; "INSERT DISK AND PRESS ANY KEY": PAUSE 0
110 INPUT INKEY$: PRINT #0;TAB 8; FLASH 1;" PLEASE WAIT "
200 RANDOMIZE USR 40000
210 LET a=43776: LET n=0: LET del=0: LET free=2544
220 GO SUB 1000
230 LET a=43776+128*16
240 GO SUB 2000
300 INPUT INKEY$: PRINT #0: "OWERWRITE TRACK <0> (Y/N) ? ": PAUSE 0: INPUT INKEY$
310 IF INKEY$="y" OR INKEY$="Y" THEN POKE 40007,6: RANDOMIZE USR 40000
320 STOP
1000 REM Katalog
1010 IF PEEK a=0 THEN RETURN
1020 LET n=n+1
1030 IF PEEK a=1 THEN LET del=del+1: GO TO 1050
1040 IF PEEK a<32 OR PEEK a>127 THEN POKE a,63
1050 LET beg1=PEEK (a+14)+16*PEEK (a+15)
1060 LET beg2=PEEK (a+14+16)+16*PEEK (a+15+16)
1070 LET lens=beg2-beg1
1080 IF PEEK (a+16)=0 THEN LET lens=PEEK (a+12)+(PEEK (a+11)<>0)
1090 POKE (a+13), lens
```

```

1100 LET free=free-lens
1110 LET a=a+16
1120 GO TO 1000
2000 REM Disk-data
2010 LET space=beg1+lens
2020 LET spt=INT (space/16): LET sps=space-spt*16
2030 POKE (a+225),sps: POKE (a+226),spt
2040 POKE (a+227),22
2050 POKE (a+228),n
2060 LET fr2=INT (free/256): LET fr1=free-fr2*256
2070 POKE (a+229),fr1: POKE (a+230),fr2
2080 POKE (a+244), del
2090 RETURN

```

По своей структуре программа похожа на "DIR". После старта с начальной строки, программа формирует кодовый блок для чтения девяти секторов нулевой дорожки, затем останавливается (строка 100) для того, чтобы можно было вставить в дисковод засекреченный диск. После нажатия клавиши происходит чтение нулевой дорожки, затем начинаются расчеты. Задаются начальный адрес каталога (a), число файлов (n) и число удаленных файлов (del) принимают для начала нулевые значения, а число свободных секторов на диске (free) - максимальное значение для двусторонней дискеты 80 дорожек - 2544 сектора.

Подпрограмма со строки, 1000 приводит в порядок каталог диска. Строка 1010 проверяет, не достигнуто ли свободное место на диске и если да, то выход из подпрограммы. Если нет, то число файлов увеличивается на единицу (строка 1020), затем проверяется, не удаленный ли это файл (строка 1030). Если да, то счетчик удаленных файлов увеличивается на единицу строка 1040 исправляет значения тех кодов в названии файла, которые могли быть искусственно изменены для невозможности произвести распечатку каталога командами TR-DOS: CAT или LIST. Управляющие коды либо токены ключевых слов, подставленные в имя файла заменяются символом ASCII с кодом 63 - это знак вопроса.

Далее рассчитывается длина файла в секторах (lens). Для этого определяется начало текущего файла (beg1), затем начало следующего файла (beg2). Эти значения рассчитываются непосредственно в секторах от начала диска. Их разность (строка 1070) определяет длину текущего файла. Однако в том случае, если текущий файл является последним на диске (строка 1080), то значение beg2 даст нулевое значение. Поэтому длину файла в секторах в этом случае можно рассчитать непосредственно по его длине в байтах. После занесения рассчитанного значения в память (строка 1090), рассчитывается оставшееся свободное место на диске (строка 1100), затем - переход к следующему заголовку путем возврата на строку 1000.

После того, как выполнена подпрограмма восстановления каталога (в строке 220), значение адреса устанавливается на начало системного сектора (строка 230). Подпрограмма со строки 2000 восстанавливает системную информацию диска. Начало свободного места на диске в секторах (space) определяется как начало последнего файла плюс его длина (строка 2010). Далее это значение преобразуется (строка 2020) в номер дорожки (spt) и сектора (sps) и заносится в память (строка 2030).

Строка 2040 устанавливает тип диска: двусторонний, 80 дорожек. Далее в память заносится общее число файлов (строка 2050), рассчитываются старший и младший байты числа свободных секторов (строка 2060) и полученные значения заносятся в память (строка 2070). Заносится в память и число удаленных файлов (строка 2080). Далее - возврат из подпрограммы на строку 300. Здесь выводится предупредительная табличка и, в случае подтверждения (клавиша "Y"), происходит перезапись нулевой дорожки диска (строка 310). Здесь значение регистра C в ячейке 40007 вместо чтения (C=5) заменяется на запись (C=6). После этого программа останавливается. Можете запускать дискету обычным порядком и копировать файлы с нее на другие дискеты.



# FORUM

## Русификация резидентной процедурой

Тема русификации это, пожалуй, вечная тема для "Спектрума". Пока что не существует способ (и будет ли когда-нибудь существовать), который бы был достаточно универсален и хорош для всех. Многие программисты ломают голову над решением этой проблемы. В этот раз интересное письмо пришло к нам от Александра Еременко из Свердловска. Он прислал программу в машинных кодах, позволяющую переключать символьный набор одновременным нажатием двух клавиш (SYMBOL SHIFT)+(ENTER). При первом нажатии включается загруженный в ОЗУ символьный набор, а при повторном нажатии - опять символьный набор из ПЗУ. Самое главное, что такое переключение можно сделать в любой момент как пишет Александр, неважно, где Вы работаете, в режиме редактирования Бейсик-строк или в программе, все равно можно производить переключение шрифта.

Нам понравилась идея уважаемого корреспондента и мы даем ее всем читателям, снабдив собственными комментариями и постаравшись немного ее развить.

Вот эта программа:

|      |        |      |            |
|------|--------|------|------------|
| CD7C | 3ECB   | LD   | A, #CB     |
| CD7E | 2120CD | LD   | HL, #CD20  |
| CD81 | 36CD   | LD   | (HL), #CD  |
| CD83 | 2B     | DEC  | HL         |
| CD84 | BC     | CP   | H          |
| CD85 | 20FA   | JR   | NZ, #CD81  |
| CD87 | 3ECC   | LD   | A, #CC     |
| CD89 | ED47   | LD   | I, A       |
| CD8B | F3     | DI   |            |
| CD8C | ED5E   | IM   | 2          |
| CD8E | FB     | EI   |            |
| CD8F | C9     | RET  |            |
| CD90 | C5     | PUSH | BC         |
| CD91 | D5     | PUSH | DE         |
| CD92 | E5     | PUSH | HL         |
| CD93 | F5     | PUSH | AF         |
| CD94 | 3E7F   | LD   | A, #7F     |
| CD96 | DBFE   | IN   | A, (#FE)   |
| CD98 | FEFD   | CP   | #FD        |
| CD9A | 202A   | JR   | NZ, #CDC6  |
| CD9C | 3EBF   | LD   | A, #BF     |
| CD9E | DBFE   | IN   | A, (#FE)   |
| CDA0 | FEFE   | CP   | #FE        |
| CDA2 | 2022   | JR   | NZ, #CDC6  |
| CDA4 | 3A10CD | LD   | A, (#CD10) |
| CDA7 | 3C     | INC  | A          |
| CDA8 | 3210CD | LD   | (#CD10), A |
| CDAB | CB47   | BIT  | 0, A       |
| CDAD | 2805   | JR   | Z, #CDB4   |
| CDAF | 01003C | LD   | BC, #3C00  |
| CDB2 | 1803   | JR   | #CDB7      |
| CDB4 | 01D0CC | LD   | BC, #CCD0  |
| CDB7 | 21365C | LD   | HL, #5C36  |
| CDBA | 71     | LD   | (HL), C    |
| CDBB | 23     | INC  | HL         |
| CDBC | 70     | LD   | (HL), B    |
| CDBD | 219701 | LD   | HL, #0197  |
| CDC0 | 116400 | LD   | DE, #0064  |
| CDC3 | CDB503 | CALL | #03B5      |
| CDC6 | F1     | POP  | AF         |
| CDC7 | E1     | POP  | HL         |

|      |        |                  |       |
|------|--------|------------------|-------|
| CDC8 | D1     | POP              | DE    |
| CDC9 | C1     | POP              | BC    |
| CDCA | C33800 | JP               | #0038 |
| CDCD | C390CD | JP               | #CD90 |
| CDD0 | .....  | символьный набор |       |

Как видим, принцип действия этой программы основан на использовании прерываний второго рода IM 2. Подробнее о прерываниях Вы можете прочитать в нашей книге, посвященной программированию в машинных кодах. Там мы рассказывали, что на использовании прерывания первого рода основан основной режим работы "Спектрума". В этом режиме (если он разрешен командой EI) всегда в ответ на прерывание, поступающее по шине INT (а это происходит 50 раз в секунду), выполняется обращение по адресу 0038H. Процедура, расположенная с этого адреса, обеспечивает сканирование клавиатуры в поисках нажатой клавиши.

Теперь о прерывании второго рода. В компьютерах иных систем, собранных на базе процессора Z-80, для программиста существует возможность самому задать адрес, по которому будет происходить обработка этого прерывания. Он определяется следующим образом. Старший байт поступает из регистра I процессора (он так и называется "вектор прерываний"), а младший байт должно выдавать периферийное устройство, от которого получено прерывание. В паре ячеек, адрес которых определен таким образом, хранится адрес процедуры, обслуживающей прерывание. Программ, обслуживающих такое прерывание может быть 128, так как старший байт определен регистром I, а младший байт - это 256 ячеек, в которых можно записать 128 двухбайтных адресов. В "Спектруме" режим прерываний второго рода не используется для взаимодействия с периферийными устройствами, поэтому появляется возможность использовать этот режим в своих целях.

Теперь конкретно о предложенной программе. Вначале выполняется инициализирующая процедура. Она запускается с адреса загрузки, то есть CD7CH (52604). При этом происходит следующее. Область, начиная с CC00H, и кончая адресом CD20H заполняется кодом CD. Затем в регистре I процессора задается старший байт адреса ячейки памяти, в которой хранится адрес обслуживающей процедуры CC. Далее происходит включение режима прерываний второго рода и выполняется возврат в вызывающую программу.

Что же происходит при получении сигнала на прерывание, (который, как мы помним, приходит 50 раз в секунду) Несмотря на то что неопределен младший байт ячейки с адресом обслуживающей программы, обратившись по любому адресу со старшим байтом CC (то есть начиная CC00H и кончая CCFFH), получим один и тот же адрес обслуживавшей программы CDCDH, так как вся эта область заполнена кодом CD.

По адресу CDCDH (см. листинг программы) видим безусловный переход на адрес CD90H. Это и есть начало программы, обслуживающей полученное прерывание. Перед началом выполнения каких либо действий, на стеке запоминается содержимое всех регистров процессора, которые могут быть изменены при работе обслуживающей процедуры. Затем выполняется сканирование двух полурядов клавиатуры (о том, как производится опрос клавиш, Вы также можете прочитать в нашей книге, посвященной программированию в машинных кодах). Вначале проверяется нажатие клавиши SYMBOL SHIFT и, если она не нажата, то сразу переход на завершение процедуры обработки прерывания. Если SYMBOL SHIFT нажата, то проверяется факт нажатия ENTER. Если нет, то также переход на завершение процедуры, а если и эта клавиша нажата, то происходит дальнейшая работа.

Проверяется содержимое ячейки, определяющей включение одного или другого символьного набора это ячейка CD10H. От значения младшего бита этой ячейки (четное число или нечетное) зависит выбор одного из двух символьных наборов, при этом происходит увеличение на единицу содержимого этой ячейки (то есть подготовка к включению в следующий раз противоположного символьного набора). Соответствующее значение системной переменной CHARS пересылается при помощи регистра BC, в область системных переменных, в ячейку 5C36H. Далее происходит вызов подпрограммы BEEPER из

ПЗУ (параметры: тон и длительность звукового сигнала задаются в регистрах HL и DE). Звуковой сигнал свидетельствует о произошедшем переключении символьного набора.

Далее выполняется финишная процедура: восстановление со стека прежних значений регистров процессора и безусловный, переход на адрес 0038H. Как это должно быть в случае обработки прерываний первого рода, с этого места работа компьютера совпадает с традиционной.

При реализации этого метода следует помнить, что символьный набор располагается сразу же за самой программой, с адреса CDD0H (52688). При этом системная переменная CHARS равна CCD0H. Но Вы можете расположить символьный набор там, где это Вам удобно, изменив значение CHARS в ячейках CDB5H и CDB6H. Кроме того, оба переключаемых символьных набора могут находиться в ОЗУ, и Вы можете поочередно переводить на любой из них. Для этого значение системной переменной chars для второго символьного набора должно быть задано в ячейках CDB0H и CDB1H. Следует также помнить, что для работы программы необходима область кодов с адреса CC00H (52224), то есть перед стартом инициализирующей процедуры (RANDOMIZE U5R 52604) надо резервировать это место, изменив RAMTOP командой CLEAR 52223.

Следует также учитывать тот факт, что раз уж мы работаем с прерываниями второго рода, призванными обслуживать периферийные устройства, то работа с последними, например с дисководом, после инициализации блока кодов будет невозможна. Для работы с периферией необходимо восстановить традиционный порядок работы. Об этом еще будет разговор ниже.

Вообще нам очень понравилась идея, заложенная в этой программе и тот изящный способ, при помощи которого эта идея реализуется. Для того, чтобы поглубже "прочувствовать" действие этой программы, наберите простую демонстрационную программку на Бейсике:

```
1 GO TO 100
2 CLEAR 52223: LOAD "rus" CODE 52604: LOAD "cbr" CODE 52688
3 RANDOMIZE USR 52604
4 RUN
100 PRINT "Press <S.S.>+<ENTER> for control": PAUSE 0
200 FOR a=32 TO 127: PRINT CHR$ a;: PAUSE 1: NEXT a: GO TO 200
```

После набора запустите её с первой строки командой RUN и внимательно понаблюдайте за реакцией компьютера. Хотя ничего неожиданного сейчас произойти не должно (компьютер не подпрыгнет, он по-прежнему останется на своем месте), но все-таки остановимся на происходящем подробнее. Сразу после старта на экране появится надпись: "нажмите <S.S.>+<ENTER> для управления" и программа остановится на PAUSE 0, ожидая нажатия клавиши. Попробуйте нажать указанную комбинацию клавиш. Пауза прервется (SYMBOL SHIFT здесь ни при чем, сработало нажатие ENTER) и начнется распечатка символьного набора из ПЗУ. Когда заполнится весь экран, появится привычный запрос: "scroll?". Попробуйте опять нажать SYMBOL SHIFT + ENTER. Вывод символов на экран будет продолжен.

Теперь остановите программу нажатием BREAK и запустите ее со второй строки (с которой должен происходить автостарт) RUN 2. При этом загрузите (заготовленные за ранее) исследуемый блок кодов "rus" и любой имеющийся у Вас символьный набор "chr" (лучше "утолщенный" или стилизованный, так будет нагляднее). После загрузки произойдет инициализация программы в кодах, после чего Бейсик программа запустится с начала, как это делали Вы командой RUN. Сравните реакцию компьютера. После появления надписи в верхней строке, нажмите SYMBOL SHIFT + ENTER. Вы услышите короткий звуковой сигнал, свидетельствующий о нажатии на клавиши, но больше ничего на экране не изменилось: пауза не прервалась, как раньше. То есть, указанная комбинация клавиш теперь не влияет никаким образом на работу Бейсик-программы. Теперь она воспринимается именно как SYMBOL SHIFT + ENTER, а не как просто ENTER. Прервите паузу нажатием на любую клавишу, и Вы увидите, что на экран выводится новый символьный набор, то есть

произошло переключение. Пока экран заполняется, можете попробовать несколько раз нажать `SYMBOL SHIFT + ENTER` и убедиться в том, что происходит переключение символьных наборов. Когда экран будет заполнен и появится запрос "scroll?", опять нажмите комбинацию клавиш. Кроме звукового сигнала, свидетельствующего о переключении символьных наборов, ничего не происходит. Для продолжения работы надо нажать, например, просто `ENTER`. Можно, таким образом, сделать вывод: приведенный блок кодов не оказывает никакого вмешательства в работу интерпретатора Бейсика, он только исправно выполняет то, что ему предписано - переключает символьные наборы.

Продолжаем наше исследование. Вызовите на редактирование 100 строку Бейсик-программы. Попробуем в кавычках напечатать текст, состоящий из символов разных символьных наборов. Для этого, находясь в середине текста, нажмите `SYMBOL SHIFT + ENTER`. Звуковой сигнал - переключение произошло, но на экране ничего не меняется до того момента, пока Вы не нажмете какую-нибудь клавишу. А происходит следующее. Весь текст редактируемой строки перерисовывается заново, причем символами нового символьного набора, отсюда второй вывод: нельзя при помощи этого блока кодов решить задачу совмещения символов разных символьных наборов внутри одного оператора `PRINT`, действительно, чудес ведь не бывает. Кодовый блок только переключает символьный набор, а текст, подлежащий выводу на экран, никак не изменяется, то есть каким было кодовое выражение строки текста, таким оно и осталось, таким и выводится на экран, только в новом символьном наборе. То же относится и к оператору `INPUT`. Нельзя при вводе текстовых выражений совместить русские и английские буквы.

И еще один вывод. Переключение символьного набора таким способом происходит только "вручную", то есть, если по алгоритму выполнения программы должно произойти переключение символьного набора, то оно должно быть выполнено, как обычно, изменением `CHARS` при помощи `POKE`. Так что же, новый изящный метод переключения шрифтов не вносит ничего нового по существу проблемы? Видимо, такое заключение преждевременно. Существуют ситуации, где изложенный способ сможет оказать действительно неоценимую пользу. Например, программа "МОЗАИКА", которая опубликована в этом выпуске `ZX-PEBIO` на с. 1. Там переключение из текстового в графический режим происходит включением регистра `CAPS LOCK`, из-за чего существует ограниченное число псевдографических символов, только по числу малых букв латинского символьного набора - 26. Если переключать символьные наборы новым способом, то во сколько раз можно расширить графические возможности, а следовательно и привлекательность этой программы! Ведь все до одного 96 символов нового символьного набора можно отдать графическим образам, да и захватить еще часть первого символьного набора, как это сделано в "МОЗАИКЕ".

Другой пример. Вы набираете программу, в которой заведомо будет только русский текст (какой-нибудь тест или обучающая программа). И естественно, хотите для придания программе большей привлекательности, использовать русские заглавные и строчные буквы, используя загружаемый только русский символьный набор. При этом, если Вы набираете текст программы, используя стандартный символьный набор ПЗУ, то легко можете допустить ошибки при наборе русского текста в операторах `PRINT`, так как набор ведется практически "вслепую". Если же Вы переключитесь на русский символьный набор, то сама Бейсик-программа, токены, имена переменных - все это станет "нечитаемым". Применение изложенного в этой статье приема позволит вам при наборе и отладке программы без труда переключаться с одного на другой символьные наборы в зависимости от того, что Вы в данный момент предпочитаете видеть.

Мы уверены, что если начать развивать эту тему, то появятся новые оригинальные программы с использованием этого метода. Определенные проблемы для начинающих может вызвать тот факт, что блок кодов нельзя переместить в другое место памяти, но при желании можно это сделать. Мы попробовали поэкспериментировать и у нас получился несколько измененный и немного усовершенствованный вариант этой программы. Вот то,

что у нас получилось.

|      |        |      |           |
|------|--------|------|-----------|
| FBF4 | 3EFB   | LD   | A, #FB    |
| FBF6 | ED47   | LD   | I, A      |
| FBF8 | ED5E   | IM   | 2         |
| FBFA | C9     | RET  |           |
| FBFB | 00     | NOP  |           |
| FBFC | ED56   | IM   | 1         |
| FBFE | C9     | RET  |           |
| FBFF | 01FC   | DEFB | #FC01     |
| FC01 | C5     | PUSH | BC        |
| FC02 | D5     | PUSH | DE        |
| FC03 | E5     | PUSH | HL        |
| FC04 | F5     | PUSH | AF        |
| FC05 | 3E7F   | LD   | A, #7F    |
| FC07 | DBFE   | IN   | A, (#FE)  |
| FC09 | FEFD   | CP   | #FD       |
| FC0B | 202D   | JR   | NZ, #FC3A |
| FC0D | 3EBF   | LD   | A, #BF    |
| FC0F | DBFE   | IN   | A, (#FE)  |
| FC11 | FEFE   | CP   | #FE       |
| FC13 | 2025   | JR   | NZ, #FC3A |
| FC15 | 2132FC | LD   | HL, #FC32 |
| FC18 | CB7E   | BIT  | 7, (HL)   |
| FC1A | 2807   | JR   | Z, #FC23  |
| FC1C | 11003C | LD   | DE, #3C00 |
| FC1F | CBBE   | RES  | 7, (HL)   |
| FC21 | 1805   | JR   | #FC28     |
| FC23 | 1158FB | LD   | DE, #FB58 |
| FC26 | CBFE   | SET  | 7, (HL)   |
| FC28 | 21365C | LD   | HL, #5C36 |
| FC2B | 73     | LD   | (HL), E   |
| FC2C | 23     | INC  | HL        |
| FC2D | 72     | LD   | (HL), D   |
| FC2E | 210002 | LD   | HL, #0200 |
| FC31 | 117F00 | LD   | DE, #007F |
| FC34 | CDB503 | CALL | #03B5     |
| FC37 | 00     | NOP  |           |
| FC38 | 00     | NOP  |           |
| FC39 | 00     | NOP  |           |
| FC3A | F1     | POP  | AF        |
| FC3B | E1     | POP  | HL        |
| FC3C | D1     | POP  | DE        |
| FC3D | C1     | POP  | BC        |
| FC3E | C33800 | JP   | #0036     |

Прежде чем начать пояснения, рассмотрим подробно карту памяти, тех областей, которые имеют значение для работы программы.

```

-----
                      РАМТОР  64499
-----
Программа в машинных      64500
кодах (новый вариант)     FBF4H
                      77 байт
-----
Свободное место; для ра-   64577
боты программы не исполь  FC41H
зуется                     23 байта
-----
Символьный набор          64600
(любой по вкусу)         FC58H
                      768 байт
-----
UDG-графика, стандартно   65368
расположенная здесь      FF58H
                      168 байт
-----
                      P_РАМТ

```

Эта область памяти должна быть зарезервирована при старте Бейсик-программы, путем переустановки RAMTOP командой CLEAR 64499. Файл, содержащий блок кодов, начинается с адреса 64500 и включает в себя, в зависимости от конкретных задач, либо только программу в кодах (до адреса 64577), либо программу и символьный набор (заодно с 23-мя неиспользуемыми байтами), либо еще и блок UDG-графики (то есть по адрес 65535 включительно). Область памяти с адреса 64577 (23 байта) в работе программы не участвует, Вы можете использовать ее по своему усмотрению.

Инициализация выполняется командой RANDOMIZE USR 64500. С этого адреса (FBF4H) начинается инициализируемая процедура. Кстати сказать, нет необходимости заполнять почти 300 байт кодом для задания адреса обслуживавшей процедуры. Дело в том, что при существующей архитектуре "Спектрума" младший байт всегда будет равен FF. То есть, если в регистре I процессора задать, например FB, то при прерывании второго рода переход произойдет всегда на программу, адрес которой хранится в FBFFH. В этой паре ячеек (FBFF - младший байт, в FC00 - старший байт) находится адрес процедуры, обслуживающей прерывание второго рода. Читаем этот адрес (см. листинг программы): FC01.

Начало процедуры, расположенной по этому адресу точно такое же, как и в неполном варианте. Так же опрашивается нажатие интересующих клавишей, а дальше начинается отличие. Системной ячейкой, определяющей, какой символьный набор должен включиться, выбрана ячейка FC32H. Точнее, определяющим является 7 бит числа, находящегося там, (получается либо 7F, либо FF). В этой ячейке содержится параметр, задающий время звукового сигнала для подпрограммы BEEPER, вызываемой из ПЗУ. Таким образом, переключение на символьный набор из ОЗУ сопровождается длинным звуком, а обратное переключение на символьный набор из ОЗУ - коротким.

После вызова подпрограммы BEEPER, зарезервировано 3 свободных ячейки памяти. Здесь, при необходимости, можно организовать вызов каких-то дополнительных сервисных процедур, если это потребуется в будущем (вызов какой-нибудь подпрограммы: CALL ... или переход - JP ... - все это занимает 3 байта) далее - финишные операции и переход на адрес 0033H.

Кроме указанных отличий, новый вариант программы предусматривает возможность отключения режима прерываний второго рода, это необходимо делать прежде, чем, например, обращаться к дисководу. Иначе произойдет сбой, зависание или сброс компьютера. "Де-инициализация" или восстановление традиционного режима работы выполняется командой RANDOMIZE USR 64508. С этого адреса (FBFCH) расположена восстанавливавшая процедура. Здесь включается режим прерываний первого рода, восстанавливая исходный режим работы "Спектрума".

В заключение хочется поблагодарить Александра Еременко за присланную программу. Надеемся, что она подтолкнет и других читателей к решению насущных проблем оригинальными, нетрадиционными методами.

### Расчет резонансных характеристик.

Кроме оригинальной программы для переключения символьных наборов, в своем письме Александр Еременко из Свердловска делится с читателями также программой на Бейсике, которую можно использовать для расчета резонансных характеристик пассивных фильтров в диапазоне звуковых частот. Расчет производится для двух типов фильтров по схемам, приведенным на рисунках 1 и 2. Мы приводим эту программу с некоторыми не принципиальными изменениями. Кроме того, мы русифицировали ее по методике, приведенной в "ZX-РЕВЮ"-92г. №1,2, стр. 31.

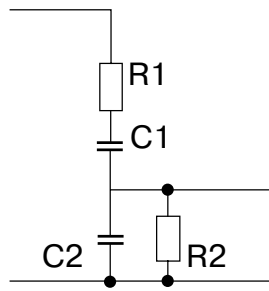


Рис. 1

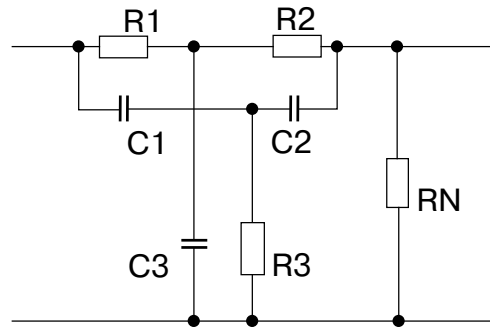


Рис. 2

Вот программа для расчета резонансной частоты фильтра по рис. 1 (автостарт программы со строки 9000, где загружается русский символьный набор):

```

1 BORDER 0: INK 7: PAPER 0: CLS
2 FOR F=0 TO 170 STEP 16: PLOT 0,F: DRAW 255,0: NEXT F
3 FOR F=0 TO 21 STEP 2: PRINT AT F+1,0; (20-F)*8: NEXT F
4 FOR F=54 TO 220 STEP 32: PLOT F,0: DRAW 0,170: NEXT F
5 PRINT AT 0,0;"          40  160 640 2560 10240"
10 INPUT "C1 (ФАРАД) : ";C1: INPUT "C2 (ФАРАД): ";C2: INPUT "R1(ОМ): ";R1: INPUT "R2 (ОМ): ";R2
11 LET F=1: LET X=16
12 LET F=F*1.0442738: LET X=X+1
20 LET C1R=1/(F*C1*2*PI)
30 LET C2R=1/(F*C2*2*PI)
40 LET R1C1=R1+C1R
50 LET R2C2=C2R*R2/(C2R+R2)
60 LET Y=R2C2*(170/(R1C1+R2C2))
70 IF X>255 THEN GO TO 10
80 PLOT X,Y
90 GO TO 12
9000 CLEAR 64599: LOAD "Chr" CODE 64600
9010 POKE 23606,88: POKE 23607,251: RUN
  
```

После старта программы на экран будет выведена сетка с оцифровкой. По горизонтали - частота, по вертикали - затухание фильтра. Введите по очереди номиналы тех элементов, которые будете использовать в фильтре. После ввода всех номиналов программа выдаст Вам амплитудно-частотную характеристику фильтра. Можете оценить ее и попробовать изменить вводимые номиналы для получения требуемой резонансной частоты, добротности и затухания фильтра. При любых изменениях номиналов результаты расчета тут же будут выведены на экран в виде изменившейся характеристики фильтра. Вы имеете возможность сравнивать характеристики между собой, так как все они на одном экране перед Вами. Благодаря этой программе подбор элементов для фильтра с требуемой амплитудно-частотной характеристикой можно выполнить легко и быстро. Результат работы программы представлен на рис. 3.

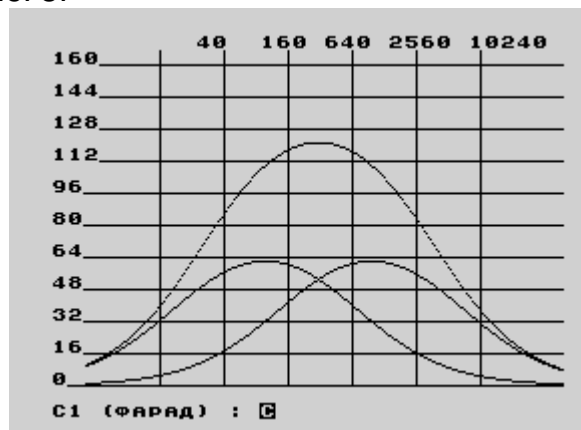


Рис. 3

Для фильтра, представленного на рисунке 2, надо изменить строку 10 программы для ввода большего числа входных параметров:

```

10 INPUT "C1 (ФАРАД): "; C1: INPUT "C2 (ФАРАД): "; C2: INPUT "C3 (ФАРАД): "; C3: INPUT "R1
(ОМ): "; R1: INPUT "R2 (ОМ): "; R2: INPUT "R3 (ОМ): "; R3: INPUT "R НАГРУЗКИ (ОМ): "; RN

```

Кроме того строки, начиная с 20 должны быть:

```

20 LET A=R2*RN
30 LET B=1/(F*2*PI*C3)
40 LET C=B*A/(B+A)
50 LET V1=C*(170/(R1+C))
60 LET V2=RN*(V1/(R2*RN))
70 LET D=1/(F*2*PI*C2)
80 LET E=D+RN
90 LET G=R3*E/(R3+E)
100 LET H=1/(F*2*PI*C1)
110 LET V3=G*(170/(H+G))
120 LET V4=RN*(V3/(D+RN))
130 IF V2>V4 THEN LET Y=V2: GO TO 150
140 LET Y=V4
150 IF X>255 THEN GO TO 10
160 PLOT X,Y
170 GO TO 12

```

Результат работы программы представлен на рис. 4.

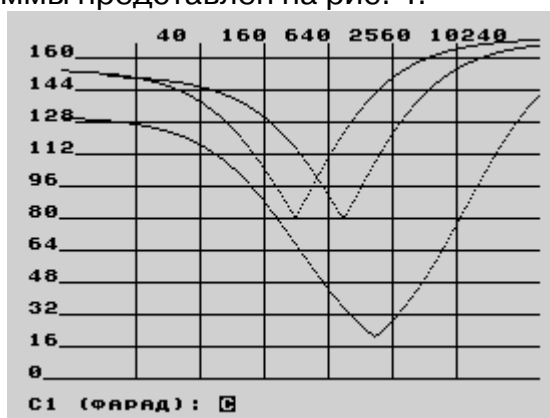


Рис. 4

Если Вас заинтересовали эти программы, то при практической реализации их, мы со своей стороны можем порекомендовать маленькое усовершенствование, которое, наверняка, сможет сделать каждый. Поскольку номиналы конденсаторов удобнее задавать в микрофарадах, а номиналы резисторов - в килоомах, то разумно будет этот пересчет поручить компьютеру. Организуйте для этого дополнительные строки. Кроме того, попробуйте объединить два варианта расчета в одну программу. Для этого можно посоветовать при загрузке готовой программы выводить экран-заставку, на котором будут изображены рисунки 1 и 2, а далее должен следовать запрос, для какой схемы будет производиться расчет.

### Нестандартная загрузка.

Комиссаров Павел из пос. Видово Мурманской обл. пишет о том, что он занимается взломом программ, в которых используется нестандартная загрузка или есть что-то необычное. У него уже набралась небольшая коллекция таких загрузчиков. Павел прислал распечатку одного из них. Он загружает блоки кодов с изменением цвета бордюрных полос. Павел пишет, что он взял этот загрузчик из программы "BARBARIAN-3" (BY BILL GILBERT). Он начинается с адреса 65000 и используется точно так же, как и процедура "LOAD BYTES" из ПЗУ, расположенная по адресу 1366 (0556H). Вот этот загрузчик:

```

FDE8 14      INC    D
FDE9 08      EX     AF, AF'
FDEA 15      DEC    D
FDEB F3      DI
FDEC 3E00    LD     A, #00
FDEE D3FE    OUT    (#FE), A
FDF0 213F05  LD     HL, #053F
FDF3 E5      PUSH   HL
FDF4 DBFE    IN     A, (#FE)

```



|      |        |      |           |
|------|--------|------|-----------|
| FDF6 | 1F     | RRA  |           |
| FDF7 | E620   | AND  | #20       |
| FDF9 | CD6F3C | CALL | #3C6F     |
| FDFC | BF     | CP   | A         |
| FDFD | 00     | NOP  |           |
| FDFE | CD79FE | CALL | #FE79     |
| FE01 | 30FA   | JR   | NC, #FDFD |
| FE03 | 216900 | LD   | HL, #0069 |
| FE06 | 10FE   | DJNZ | #FE06     |
| FE08 | 2B     | DEC  | HL        |
| FE09 | 7C     | LD   | A, H      |
| FE0A | B5     | OR   | L         |
| FE0B | 20F9   | JR   | NZ, #FE06 |
| FE0D | CD75FE | CALL | #FE75     |
| FE10 | 30EB   | JR   | NC, #FDFD |
| FE12 | 069C   | LD   | A, #9C    |
| FE14 | CD75FE | CALL | #FE75     |
| FE1T | 30E4   | JR   | NC, #FDFD |
| FE19 | 3EC6   | LD   | A, #C6    |
| FE1B | B0     | CP   | B         |
| FE1C | 30E0   | JR   | NC, #FDFE |
| FE1E | 24     | INC  | H         |
| FE1F | 20F1   | JR   | NZ, #FE12 |
| FE21 | 06C9   | LD   | B, #C9    |
| FEE3 | CD79FE | CALL | #FE79     |
| FE26 | 30D5   | JR   | NC, #FDFD |
| FE28 | 78     | LD   | A, B      |
| FE29 | FED4   | CP   | #D4       |
| FE2B | 30F4   | JR   | NC, #FE21 |
| FE2D | CD79FE | CALL | #FE79     |
| FE30 | D0     | RET  | NC        |
| FE31 | 79     | LD   | A, C      |
| FE32 | EE03   | XOR  | #03       |
| FE34 | 4F     | LD   | C, A      |
| FE35 | 2600   | LD   | H, #00    |
| FE37 | 06B0   | LD   | B, #B0    |
| FE39 | 181F   | JR   | #FE5A     |
| FE3B | 08     | EX   | AF, AF'   |
| FE3C | 2007   | JR   | NZ, #FE45 |
| FE3E | 300F   | JR   | NC, #FE4F |
| FE40 | DD7500 | LD   | (IX+0), L |
| FE43 | 160F   | JR   | #FE54     |
| FE45 | CB11   | RL   | C         |
| FE47 | AD     | XOR  | L         |
| FE48 | C0     | RET  | NZ        |
| FE49 | 79     | LD   | A, C      |
| FE4A | 1F     | RRA  |           |
| FE4B | 4F     | LD   | C, A      |
| FE4C | 13     | INC  | DE        |
| FE4D | 1807   | JR   | #FE56     |
| FE4F | DD7E00 | LD   | A, (IX+0) |
| FE52 | AD     | XOR  | L         |
| FE53 | C0     | RET  | NZ        |
| FE54 | DD23   | INC  | IX        |
| FE56 | 1B     | DEC  | DE        |
| FE57 | 08     | EX   | AF, AF'   |
| FE58 | 06B2   | LD   | B, #B2    |
| FE5A | 2E01   | LD   | L, #01    |
| FE5C | CB75FE | CALL | #FE75     |
| FE5F | D0     | RET  | NC        |
| FE60 | 3ECB   | LD   | A, #CB    |
| FE62 | B8     | CP   | B         |
| FE63 | CB15   | RL   | L         |
| FE65 | 0680   | LD   | B, #B0    |
| FE67 | D25CFE | JP   | NC, #FE5C |

|      |        |      |           |
|------|--------|------|-----------|
| FE6A | 7C     | LD   | A, H      |
| FE6B | AD     | XOR  | L         |
| FE6C | 67     | LD   | H, A      |
| FE6D | 7A     | LD   | A, D      |
| FE6E | B3     | OR   | E         |
| FE6F | 20CA   | JR   | NZ, #FE3B |
| FE71 | 7C     | LD   | A, H      |
| FE72 | FE01   | CP   | #01       |
| FE74 | C9     | RET  |           |
| FE75 | CD79FE | CALL | #FE79     |
| FE76 | D0     | RET  | NC        |
| FE79 | 3E16   | LD   | A, #16    |
| FE7B | 3D     | DEC  | A         |
| FE7C | 20FD   | JR   | NZ, #FE7B |
| FE7E | A7     | AND  | A         |
| FE7F | 04     | INC  | B         |
| FE80 | C8     | RET  | Z         |
| FE81 | 3E7F   | LD   | A, #7F    |
| FE83 | DBFE   | IN   | A, (#FE)  |
| FE85 | 1F     | RRA  |           |
| FE86 | 00     | NOP  |           |
| FE87 | A9     | XOR  | C         |
| FE88 | E6E0   | AND  | #20       |
| FE8A | 28F3   | JR   | Z, #FE7F  |
| FE8C | 79     | LD   | A, C      |
| FE8D | 2F     | CPL  |           |
| FE8E | 4F     | LD   | C, A      |
| FE6F | 82     | ADD  | A, D      |
| FE90 | A9     | XOR  | C         |
| FE91 | E607   | AND  | #07       |
| FE93 | F608   | OR   | #08       |
| FE95 | D3FE   | OUT  | (#FE), A  |
| FE97 | 37     | SCF  |           |
| FE98 | C9     | RET  |           |

Перед вызовом загрузчика надо в регистре IX задать адрес загрузки, а в регистре DE - длину загружаемого блока кодов. Процедура вызова загрузчика, например для загрузки экрана-заставки может иметь следующий вид:

|          |      |           |
|----------|------|-----------|
| DD210040 | LD   | IX, #4000 |
| 11001B   | LD   | DE, #1B00 |
| 3EFF     | LD   | A, #FF    |
| 37       | SCF  |           |
| CDE8FD   | CALL | #FDE8     |
| C9       | RET  |           |

При этом блок кодов, подлежащий загрузке, может быть записан на ленте обычным способом, надо лишь удалить заголовок, оставив только коды.

Цвет полос бордюра во время загрузки будет периодически меняться, временами полосы вообще будут исчезать, затем появляться вновь с новыми цветами. Клавиша BREAK при загрузке не действует.

По окончании загрузки значение флага переноса содержит информацию об ошибке при загрузке: 1 - не было, 0 - была. Здесь надо сказать, что и как в случае использования процедуры 1366 (0556H) ПЗУ, об ошибке свидетельствует только значение флага переноса, никакие дальнейшие действия, связанные с наличием ошибки не выполняются. Но если Вы хотите, чтобы при ошибке программа останавливалась с выдачей сообщения: "R Tape loading error", то надо поступить аналогично тому, как это делается в ПЗУ. Там для вызова процедуры 1366 используется процедура 2050. Применяя аналогичный прием, вызов приведенного загрузчика может выглядеть следующим образом:

|        |                   |       |
|--------|-------------------|-------|
| .....  | подготовка пара-  |       |
| .....  | метров (см. выше) |       |
| CDE8FD | CALL              | #FDE8 |
| D8     | RET               | C     |
| CF     | RST               | 8     |
| 1A     | DEFB              | #1A   |

На этом можно было бы и закончить, но вот что случилось спустя некоторое время, после того, как мы успешно оттестировали этот загрузчик. Неожиданно выяснилось, что загрузчик упорно не хочет работать на другом компьютере. Мы решили выяснить, почему это происходит. К счастью, программа "BARBARIAN-3" оказалась в нашей "фонотеке". Мы опробовали ее и выяснили, что на обоих компьютерах программа нормально загружается. В чем же дело? Стали исследовать коды загрузчика, который приводит Павел. Как и следовало ожидать, загрузчик похож на процедуру "LOAD BYTES", расположенную в ПЗУ по адресу 1366. Но стоп! Что это такое? По адресу FDF9H видим команду: CALL 3C6FH. Это вызов подпрограммы из ПЗУ. Но ведь мы знаем, что в ПЗУ на этом месте ничего нет. Или есть? Так вот, оказывается, где кроется разгадка. Дело в том, что на том компьютере, где этот загрузчик не работал, стоит стандартное ПЗУ, а на том, где работал нормально, стоит ПЗУ "ТУРБО-90", а котором на свободном месте, перед символьным набором расположены программы, расширяющие функциональные возможности компьютера, в частности, загрузка с удвоенной скоростью. Для этого процедура 0556H (1366) этого ПЗУ содержит в точности то, что видим в листинге загрузчика, который приводит Павел, по адресу FDF9H: выполнение подпрограммы 3C6FH. Это связано с загрузкой на удвоенной скорости. В обычном ПЗУ подпрограмма 5C6FH отсутствует, поэтому этот загрузчик и не может работать с другим ПЗУ, кроме "ТУРБО-90". Кстати, у Комиссарова Павла, наверняка, тоже ПЗУ "ТУРБО-90" (необходимый инструмент любого "взломщика").

Вроде бы разобрались, но опять непонятно, почему же программа "BARBARIAN-3" работает с любым ПЗУ: и с обычным, и с "ТУРБО-90". Полезли в программу. А выяснилась очень интересная вещь. В программе "BARBARIAN-3" отсутствует загрузчик кодов в том виде, в каком его приводит Павел. Загрузчик в кодах формируется в программе из подпрограммы 1366 ПЗУ. Мы немного поискали и нашли ту процедуру, которая выполняет эту работу. Она в процессе загрузки "BARBARIAN-3" располагается с адреса C3D5H (50133). Вот что там находится (фрагмент кодов приводим с некоторыми изменениями, в виде завершенной подпрограммы, готовой к употреблению):

|      |         |      |             |
|------|---------|------|-------------|
| C3D5 | 215605  | LD   | HL, #0556   |
| C3D8 | 11E8FD  | LD   | DE, #FDE8   |
| C3DB | 01AF00  | LD   | BC, #00AF   |
| C3DE | EDB0    | LDIR |             |
| C3E0 | 2119C4  | LD   | HL, #C419   |
| C3E3 | 118FFE  | LD   | DE, #FE8F   |
| C3E6 | 010A00  | LD   | BC, #000A   |
| C3E9 | EDB0    | LDIR |             |
| C3EB | 2175FE  | LD   | HL, #FE75   |
| C3EE | 220EFE  | LD   | (#FE0E), HL |
| C3F1 | 2215FE  | LD   | (#FE15), HL |
| C3F4 | 225DFE  | LD   | (#FE5D), HL |
| C3F7 | 21T9FE  | LD   | HL, #FE79   |
| C3FA | 22FFFFD | LD   | (#FDFF), HL |
| C3FD | 2224FE  | LD   | (#FE24), HL |
| C400 | 222EFE  | LD   | (#FE2E), HL |
| C403 | 2276FE  | LD   | (#FE76), HL |
| C406 | 215CFE  | LD   | HL, #FE5C   |
| C409 | 2268FE  | LD   | (#FE66), HL |
| C40C | AF      | XOR  | A           |
| C40D | 32EDFD  | LD   | (#FDED), A  |
| C410 | 32FDFD  | LD   | (#FDFD), A  |
| C413 | 3286FE  | LD   | (#FE86), A  |
| C416 | C9      | RET  |             |
| C417 | 00      | NOP  |             |
| C418 | 00      | NOP  |             |
| C519 | 82      | ADD  | A, D        |
| C41A | A9      | XOR  | C           |
| C41B | E607    | AND  | #07         |
| C41D | F608    | OR   | #08         |
| C41F | D3FE    | OUT  | (#FE), A    |
| C421 | 37      | SCF  |             |
| C422 | C9      | RET  |             |

Эта подпрограмма формирует загрузчик кодов, расположенный с адреса 65000, а далее Вы можете использовать его так, как рассказывалось выше.

Если сформировать загрузчик кодов, пользуясь приведенной подпрограммой C3D5H, при использовании ПЗУ "ТУРБО-90", получим тот загрузчик, который приводит Комиссаров Павел. А если сформировать загрузчик при использовании стандартного ПЗУ, то получим следующие отличия по-сравнению с вариантом для "ТУРБО-90":

|      |        |    |           |
|------|--------|----|-----------|
| FDF9 | F602   | OR | #02       |
| FDFB | 4F     | LD | C, A      |
| FE03 | 211504 | LD | HL, #0415 |

Кстати, если внести эти изменения в вариант Павла, то загрузчик будет работать с обоими ПЗУ. Почему же в "BARBARIAN-3" используется казалось бы более головомомный прием, когда загрузчик формируется при помощи другой подпрограммы (C3D5H) из процедуры 0556H ПЗУ? Ответ прост. Если у Вас стандартное ПЗУ, то Вы работаете как обычно. А если у Вас "ТУРБО-90", то сохранится возможность загружать всю игру в режиме с удвоенной скоростью (как это предусмотрено, введя "-" перед командой LOAD ""). Несмотря на то, что загрузчик кодов в общем-то нестандартный, в нем поддерживается турбо-режим. Это происходит как раз из-за вызова подпрограммы 3C6FH, который мы видим по адресу FDF9H. Те читатели, у которых есть "ТУРБО 90", могут убедиться в том, что загрузчик, приведенный Павлом, работает и на удвоенной скорости (правда при этом пропадает эффект переключения цвета бордюрных полос). Кроме того, процедура, которая создает загрузчик, гораздо короче самого загрузчика, поэтому такой путь еще и экономит память и время загрузки исходной программы.

Эта история показывает, насколько осторожным и предусмотрительным надо быть, проектируя автономные загрузчики для загрузки кодов. Иначе за Вашей замечательной программой, в которой Вы примените такой загрузчик, может закрепиться репутация "капризной" или "неработоспособной". Надеемся все же, что приведенный пример не отпугнет программистов, а вооружит сведениями, позволяющими выполнить эту работу более квалифицированно.

# ПРОФЕССИОНАЛЬНЫЙ ПОДХОД

## Отладчик машинного кода "TRACER"

Создание резидентных программ - один из самых интересных моментов в работе программиста. Резидентные программы работают как бы "поверх" основных и позволяют не выходя из главной программы выполнять еще какую-либо ценную задачу. Так, например, в разделе "ФОРУМ" Вам была представлена резидентная программа-русификатор, которая 50 раз в секунду проверяет нажатие комбинации клавиш SYM.SH. + ENTER и, если они не нажаты, то резидент себя не проявляет и остается "прозрачным" для пользователя.

В игровых программах резидентные процедуры как правило занимаются тем, что воспроизводят музыку одновременно с работой других процедур, а иногда им поручают обслуживание клавиатуры в поисках нажатия нужной клавиши. Нередко их используют и при организации мультипликации.

Создание резидентных программ для "Спектрума" основывается на использовании режима прерываний 2-го рода (IM2) и сегодня мы представляем Вашему вниманию еще одну полезную резидентную программу, которая поможет тем, кто программирует в машинных кодах, особенно там, где иные средства откажутся работать.

Кто не испытывал трудностей при отладке программ в машинных кодах, когда казалось бы вполне "хорошая" программа почему-то вдруг ни с того ни с сего "зависает" и никак не удастся определить, почему же это происходит. Проблему отчасти решает пошаговая отладка программы при помощи мощных мониторов, таких, как, например "MONS-3", "MONS-4". Но только отчасти, так как для того, чтобы вручную "прощелкать" достаточно большую программу, требуется уйма времени, ведь программы содержат массу различных циклов и придется повторять выполнение программы в одних и тех же адресах многократно. В общем задача эта - не из простых. Пошаговая отладка - это конечно, хорошо, но неужели только этим ограничиваются возможности "влезть внутрь" программы и посмотреть, что же там происходит. Сегодня мы предлагаем читателям еще один инструмент, позволяющий заглянуть внутрь программы во время ее работы, причем в режиме реального времени. Конечно, использование приведенного метода требует определенной квалификации, но если Вы хорошо разберетесь с ним, то получите в свой арсенал инструмент, значение которого трудно переоценить.

Итак, вниманию читателей предлагается отладчик машинного кода, использующий режим прерываний второго рода.

Программа-отладчик машинного кода называется "TRACER". Она позволяет выводить в правой части экрана заданную информацию в шестнадцатеричном коде. Что это может быть за информация? Это зависит от Ваших требований. Что Вы зададите. Например, Вы хотите просмотреть число, записанное в какой-нибудь ячейке памяти, как оно изменяется. Или как меняется содержимое какого-то регистра. Или что записано на вершине машинного стека. Принцип работы отладчика состоит в том, что 50 раз в секунду будет происходить прерывание Вашей исследуемой программы и 50 раз в секунду интересующие Вас сведения будут выведены на экран в столбец в правой части экрана. Процедура, обслуживающая прерывание, должна формировать требуемый параметр и выдавать его на печать. От того, что конкретно Вы хотите просмотреть, зависит начало этой процедуры. В качестве примера, рассмотрим вариант кодов, который позволяет просмотреть число, записанное на вершине машинного стека. Блок кодов, который это делает, приведен в распечатке (Листинг 1), а возможный пример работы отладчика - на рис. 1.

```

1 GO TO 100: REM
START-65271 STOP-65292
2 BORDER 7: PAPER 7: INK
LEAR 65115
3 LOAD "otl" CODE
4 RANDOMIZE USR 65271
100 LIST
10AF
15E8
1600
10AC
15F8
15E6
15FE
10A8
15EC
15DE
10B4
15FB
15EB
15E1
10AF
15F8
15E7
15FE
10AC
15FB
15EB
15E1
0 OK, 100:1

```

Рис. 1

Включение отладчика в работу происходит при выполнении инициализирующей подпрограммы START, расположенной с адреса FEF7H (RANDOMIZE USR 65271). При этом происходит следующее. Вначале задается адрес в дисплейном файле, куда будет производиться вывод результатов: в регистр HL записывается адрес верхней пиксельной линии того знакоместа экрана, куда будет производиться печать. Это значение заносится в системную ячейку FF13H. Затем обнуляется другая системная ячейка 5C81H (это неиспользуемый адрес в таблице системных переменных). Здесь будет организован счетчик строк для выводимой информации. После этого происходит включение режима прерываний второго рода. В регистр I процессора записывается число FEH. Поэтому при прерывании произойдет переход на программу, адрес которой записан в ячейке памяти FEFFH. Там записан адрес FE5CH. Это начало обрабатывающей процедуры. Затем происходит переключение на режим прерываний второго рода и возврат.

Теперь 50 раз в секунду будет выполняться обслуживающая процедура. По условию, которое мы задали, она должна выдавать значение двухбайтного числа, находящегося на вершине машинного стека. Считывание этого значения происходит следующим образом. Текущее значение регистра HL сохраняется в неиспользуемой ячейке 6CB0H в таблице системных переменных. Далее интересующее нас число с вершины стека заносится в регистр HL. При этом необходимо восстановить стек в том виде, в каком он был, что выполняется при помощи обратной записи числа из HL на стек. Далее на стеке сохраняются значения всех регистров процессора, которые задействованы и будут изменены в процедуре обработки прерывания.

Теперь надо интересующее нас число (в регистре HL) вывести на печать. Это выполняется следующим образом. Проверяется значение счетчика строк 5C81H. Если не достигнут нижний край экрана, то происходит переход на процедуру CONT, а если достигнут, то для последующего вывода задается опять верхняя строка экрана путем занесения в системный счетчик FF13H исходного адреса в дисплейном файле. Счетчик строк при этом обнуляется.

Процедура CONT начинается с того, что в регистр DE наносится текущее значение адреса для вывода в дисплейном файле. Затем старший байт интересующего числа записывается в регистр A и при помощи ротации вправо старший полубайт становится на место младшего. Выполняется его печать при помощи подпрограммы PRINT, которая выполняет печать четырех младших битов, содержащихся в регистре A (в шестнадцатеричном представлении четырем битам соответствует один символ). После печати значение DE увеличивается на единицу, что соответствует переходу к следующему знакоместу. Затем в регистр A опять записывается значение из регистра H и выполняется печать младшего полубайта. Значение DE опять увеличивается на 1. Далее то же повторяется и со значением регистра L, точно так же за два приема выполняется печать двух полубайтов.

Далее, путем выполнения ротации и суммирования, в счетчике FF13H получается следующий адрес в дисплейном файле, соответствующий выводу на следующей строке. Такой способ вычисления этого адреса связан с тем, что экранная память состоит из трех сегментов и надо получить правильный результат при переходе от одного сегмента к

другому. В нашей книге "Элементарная графика" мы подробно разбирали все вопросы, связанные с выводом на экран. Поэтому, для более ясного представления о том, как все это происходит, заказывайте и читайте нашу книгу.

## Листинг 1

|      |          |         |      |             |                          |
|------|----------|---------|------|-------------|--------------------------|
| FE5C | E2B05C   | INT S/R | LD   | (#5CB0), HL | ; Сохранение HL.         |
| FE5F | E1       |         | POP  | HL          | ; Получение требуемого   |
|      |          |         |      |             | ; параметра в HL.        |
| FE60 | E5       |         | PUSH | HL          | ; Восстановление стека.  |
| FE61 | F5       |         | PUSH | AF          | ; Сохранение             |
| FE62 | C5       |         | PUSH | BC          | ; значений               |
| FE63 | D5       |         | PUSH | DE          | ; регистров.             |
| FE64 | 3A815C   |         | LD   | A, (#5C81)  | ; Переход к              |
| FE67 | 3C       |         | INC  | A           | ; следующей              |
| FE68 | 32815C   |         | LD   | (#5C81), A  | ; строке.                |
| FE6B | FE16     |         | CP   | #16         | ; Проверка на достижение |
| FE6D | 200B     |         | JR   | NZ, #FE7A   | ; конца экрана.          |
| FE6F | 111C40   |         | LD   | DE, #401C   | ; Переход на             |
| FE72 | ED5313FF |         | LD   | (#FF13), DE | ; начало экрана.         |
| FE76 | AF       |         | XOR  | A           | ; Обнуление              |
| FE77 | 32815C   |         | LD   | (#5C81), A  | ; счетчика строк         |
| FE7A | ED5B13FF | CONT    | LD   | DE, (#FF13) | ; Текущий адрес          |
|      |          |         |      |             | ; в дисплейном файле.    |
| FE7F | 7C       |         | LD   | A, H        | ; Выделение              |
| FE7F | 1F       |         | RRA  |             | ; старшего               |
| FE80 | 1F       |         | RRA  |             | ; полубайта.             |
| FE81 | 1F       |         | RRA  |             | ;                        |
| FE82 | 1F       |         | RRA  |             | ;                        |
| FE83 | CDB9FE   |         | CALL | #FEB9       | ; Печать старшего        |
| FE86 | 13       |         | INC  | DE          | ; полубайта.             |
| FE87 | 7C       |         | LD   | A, H        | ;                        |
| FE88 | CDB9FE   |         | CALL | #FEB9       | ; Печать младшего        |
| FE8B | 13       |         | INC  | DE          | ; полубайта.             |
| FE8C | 7D       |         | LD   | A, L        | ; Выделение              |
| FE8D | 1F       |         | RRA  |             | ; старшего               |
| FE8E | 1F       |         | RRA  |             | ; полубайта.             |
| FE8F | 1F       |         | RRA  |             | ;                        |
| FE90 | 1F       |         | RRA  |             | ;                        |
| FE91 | CDB9FE   |         | CALL | #FEB9       | ; Печать старшего        |
| FE94 | 13       |         | INC  | DE          | ; полубайта.             |
| FE95 | 7D       |         | LD   | A, L        | ;                        |
| FE96 | CDB9FE   |         | CALL | #FEB9       | ; Печать мл. полубайта.  |
| FE99 | 2A13FF   |         | LD   | HL, (#FF13) | ; Расчет                 |
| FE9C | CB1C     |         | RR   | H           | ; адреса                 |
| FE9E | CB1C     |         | RR   | H           | ; в дисплейном           |
| FEA0 | CB1C     |         | RR   | H           | ; файле                  |
| FEA2 | 012000   |         | LD   | BC, #0020   | ; с учетом               |
| FEA5 | ED4A     |         | ADC  | HL, BC      | ; сегмента               |
| FEA7 | CB14     |         | RL   | H           | ; экрана.                |
| FEA9 | CB14     |         | RL   | H           | ;                        |
| FEAB | CB14     |         | RL   | H           | ;                        |
| FEAD | 2213FF   |         | LD   | (#FF13), HL | ;                        |
| FEB0 | D1       |         | POP  | DL          | ; Финишные операции      |
| FEB1 | C1       |         | POP  | BC          | ; по восстановлению      |
| FEB3 | F1       |         | POP  | AF          | ; значений               |
| FEB3 | 2AB05C   |         | LD   | HL, (#5CB0) | ; регистров.             |
| FEB6 | C33800   |         | JP   | #0038       | ; Переход на RST #38     |
| FEB9 | E60F     | PRINT   | AND  | #0F         | ; Преобразование         |
| FEBB | 87       |         | ADD  | A, A        | ; кода в символ          |
| FEBD | E5       |         | PUSH | ML          | ; согласно               |
| FEBD | 21D7FE   |         | LD   | HL, #FED7   | ; таблице                |

|      |        |       |      |             |                           |
|------|--------|-------|------|-------------|---------------------------|
| FEC0 | 0600   |       | LD   | B, #00      | ; с учетом                |
| FEC2 | 4F     |       | LD   | C, A        | ; шестнадцати-            |
| FEC3 | 09     |       | ADD  | HL, BC      | ; ричного                 |
| FEC4 | 46     |       | LD   | B, (HL)     | ; представления.          |
| FEC5 | 23     |       | INC  | HL          | ;                         |
| FEC6 | 4E     |       | LD   | C, (HL)     | ;                         |
| FEC7 | C5     |       | PUSH | BC          | ;                         |
| FEC8 | E1     |       | POP  | HL          | ;                         |
| FEC9 | 0608   |       | LD   | B, #08      | ;                         |
| FECB | 7E     | PRT   | LD   | A, (HL)     | ; Вывод                   |
| FECC | 12     |       | LD   | (DE), A     | ; символа                 |
| FECD | 23     |       | INC  | HL          | ; на                      |
| FECE | 14     |       | INC  | D           | ; экран.                  |
| FECF | 10FA   |       | DJNZ | #FECB       | ;                         |
| FED1 | 7A     |       | LD   | A, D        | ;                         |
| FED2 | D608   |       | SUB  | #08         | ;                         |
| FED4 | 57     |       | LD   | D, A        | ;                         |
| FED5 | E1     |       | POP  | HL          | ;                         |
| FEDS | C9     |       | RET  |             | ;                         |
| FED7 | 3D803D | TABLE | DEFB |             | ; Таблица                 |
| FEDA | 883D90 |       | DEFB |             | ; для                     |
| FEDD | 3D983D |       | DEFB |             | ; преобразования          |
| FEE0 | A03DA8 |       | DEFB |             | ; кода                    |
| FEE3 | 3DB03D |       | DEFB |             | ; в символ                |
| FEE6 | B83DC0 |       | DEFB |             | ; с учетом                |
| FBE9 | 3DC83E |       | DEFB |             | ; шестнадцати-            |
| FEEC | 083E10 |       | DEFB |             | ; ричного                 |
| FEED | 3E183E |       | DEFB |             | ; представления.          |
| FEF2 | 203E28 |       | DEFB |             | ;                         |
| FEF5 | 3E30   |       | DEFB |             | ;                         |
| FEF7 | 211C40 | START | LD   | HL, #401C   | ; Инициализирующая        |
| FEFA | 2213FF |       | LD   | (#FF13), HL | ; процедура               |
| FEFD | 1802   |       | JR   | #FF01       |                           |
| FEFF | 5CFE   | ADR   | DEFB |             | ; Адрес перехода по IM2.  |
| FF01 | AF     | PASS  | XOR  | A           | ; Продолжение             |
| FF02 | 32815C |       | LD   | (#5C81), A  | ; инициализирующей        |
| FF05 | 3EFE   |       | LD   | A, #FE      | ; процедуры.              |
| FF07 | ED47   |       | LD   | I, A        | ;                         |
| FF09 | ED5E   |       | IM   | 2           | ;                         |
| FF0B | C9     |       | RET  |             | ;                         |
| FF0C | ED56   | STOP  | IM   | 1           | ; Восстанавливающая       |
| FF0E | 3E3F   |       | LD   | A, #3F      | ; процедура.              |
| FF10 | ED47   |       | LD   | I, A        | ;                         |
| FF12 | C9     |       | RET  |             | ;                         |
| FF13 | 00     | SCRIP | NOP  |             | ; Текущее значение адреса |
| FF14 | 00     |       | NOP  |             | ; в дисплейном файле.     |

После выполнения печати искомой величины, происходят финишные операции. Это восстановление со стека значений регистров процессора. При этом значение регистра HL, как мы помним, находится в ячейке 5CD0H, оттуда его и считываем. Далее - переход на адрес 0038H, как и в случае прерывания первого рода.

Непосредственно печать выполняется при помощи подпрограмм печати PRINT и PRT. Вначале зануляются старшие байты регистра A (выполняется печать только одного полубайта). Для того, чтобы обеспечить печать в шестнадцатеричном виде, используется перевод кода в символ при помощи таблицы, расположенной с адреса FED7H. В зависимости от значения кода происходит установка в регистре HL адреса в символьном наборе, с которого находится изображение этого символа. Затем процедура PRT выполняет



переписывание восьми байт из символьного набора в дисплейный файл.

В результате работы программы-отладчика в правой части экрана Вы будете видеть столбец непрерывно меняющихся цифр. Это и есть результат работы программы - числа, которые находятся на вершине стека в процессе работы программы. Они меняются достаточно быстро, но ведь все происходит в режиме реального времени. Да и к тому же это просто демонстрационный пример, показывающий возможности программы.

На практике, для того, чтобы "вытаскивать" из исследуемой программы требуемые параметры, Вам придется несколько видоизменить начальную часть процедуры, обслуживающей прерывание. Хорошенько разобравшись с принципом действия этой программы, Вы сможете легко применять ее для своих вполне конкретных целей.

Для того, чтобы отключить отладчик, надо выполнить останавливающую подпрограмму, расположенную с адреса FF0CH (RANDOMIZE USR 65292). При этом происходит переключение на режим прерываний 1 рода - восстановление обычного режима работы "Спектрума".

И в заключении об одном ограничении на работу отладчика. Вывод на экран будет возможен только в том случае, если прерывания разрешены (EI). Если в исследуемой программе они запрещаются командой DI, то прерываний не будет и, следовательно, не будет вывода на экран.

# МАЛЕНЬКИЕ ХИТРОСТИ

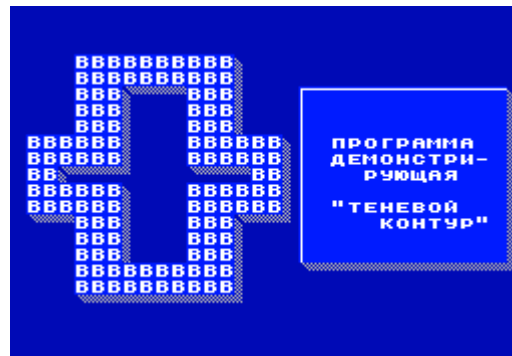
## ПРОГРАММА "ТЕНЕВОЙ КОНТУР"

На страницах "ZX-РЕВЮ" уже не раз приводились примеры того, как можно внести некоторые усовершенствования в программы на Бейсике, применяя для этого фрагменты в машинных кодах. Сегодня, в продолжение этой темы, мы приводим блок кодов, который позволит вносить определенные изменения в изображение, имеющееся на экране. Посмотрите на рисунок. Изображение, которое Вы видите, как бы объемно. Создается впечатление, что оно приподнято над плоскостью экрана. Вам, наверное, приходилось видеть такой прием в программах. Мы предлагаем Вашему вниманию оригинальный способ получения этого эффекта.

Суть метода заключается в том, что теневое оконтуривание выполняется автоматически программой в машинных кодах. Сначала "плоское" изображение выводится на экран обычным способом, будь то Бейсик или процедуры в машинных кодах.

Можно даже заготовить "плоскую" картинку при помощи графического редактора. В общем, представим, что имеется изображение, полученное любым способом. Затем, вызывая приведенную ниже программу в кодах, выполняется теневое оконтуривание этого изображения, причем весь экран может быть обработан за один прием, а можно вызывать кодовую программу несколько раз, выполняя оконтуривание в несколько приемов.

Критерием, определяющим то, что надо оконтуривать, а что не надо, является повышенная яркость (BRIGHT 1). Все, что на экране нарисовано с повышенной яркостью, будет оконтурено при вызове блока в машинных кодах. Как это эффективнее всего использовать в программах придумайте сами. Дело за Вашей фантазией. Мы же предлагаем программу в машинных кодах под названием "Теневой контур". Ее БЕЙСИК-загрузчик приведен в Листинге\_1, а машинный код с комментариями - в Листинге\_2.



Листинг\_1

```
10 FOR i = 1 TO 211
20 READ a
30 POKE 63999+i, a
40 NEXT i
50 DATA 243, 253, 229, 253, 33
60 DATA 255, 90, 6, 24, 197
70 DATA 6, 32, 197, 253, 126
80 DATA 0, 254, 64, 56, 15
90 DATA 253, 126, 1, 254, 64
100 DATA 220, 82, 250, 253, 126
110 DATA 32, 254, 64, 56, 93
120 DATA 253, 43, 193, 16, 228
130 DATA 193, 16, 222, 253
140 DATA 225, 251, 201, 122, 230
150 DATA 3, 7, 7, 7, 246
160 DATA 64, 103, 107, 201, 36
170 DATA 124, 230, 7, 192, 124
```

|     |      |      |      |      |      |     |
|-----|------|------|------|------|------|-----|
| 180 | DATA | 214, | 8,   | 103, | 125, | 198 |
| 190 | DATA | 32,  | 111, | 208, | 124, | 198 |
| 200 | DATA | 8,   | 103, | 238, | 88,  | 192 |
| 210 | DATA | 38,  | 0,   | 201, | 120, | 254 |
| 220 | DATA | 32,  | 200, | 253, | 229, | 209 |
| 230 | DATA | 19,  | 205, | 47,  | 250, | 221 |
| 240 | DATA | 33,  | 191, | 250, | 6,   | 8   |
| 250 | DATA | 62,  | 64,  | 253, | 190, | 33  |
| 260 | DATA | 56,  | 9,   | 6,   | 10,  | 253 |
| 270 | DATA | 190, | 32,  | 48,  | 2,   | 6   |
| 280 | DATA | 12,  | 221, | 126, | 0,   | 182 |
| 290 | DATA | 119, | 221, | 35,  | 205, | 58  |
| 300 | DATA | 250, | 16,  | 244, | 201, | 225 |
| 310 | DATA | 241, | 245, | 229, | 254, | 24  |
| 320 | DATA | 210, | 35,  | 250, | 253, | 229 |
| 330 | DATA | 209, | 235, | 1,   | 32,  | 0   |
| 340 | DATA | 9,   | 235, | 14,  | 2,   | 221 |
| 350 | DATA | 33,  | 203, | 250, | 205, | 47  |
| 360 | DATA | 250, | 6,   | 4,   | 221, | 126 |
| 370 | DATA | 0,   | 182, | 119, | 221, | 35  |
| 380 | DATA | 205, | 58,  | 250, | 16,  | 244 |
| 390 | DATA | 253, | 126, | 33,  | 254, | 64  |
| 400 | DATA | 210, | 35,  | 250, | 241, | 245 |
| 410 | DATA | 254, | 32,  | 210, | 35,  | 250 |
| 420 | DATA | 19,  | 13,  | 32,  | 220, | 195 |
| 430 | DATA | 35,  | 250  |      |      |     |
| 500 | DATA | 0,   | 0,   | 128, | 64,  | 160 |
| 510 | DATA | 80,  | 160, | 80,  | 160, | 80  |
| 520 | DATA | 160, | 80   |      |      |     |
| 600 | DATA | 42,  | 21,  | 10,  | 5,   | 128 |
| 610 | DATA | 64,  | 160, | 80   |      |     |

Для демонстрации действия блока машинных кодов надо набрать Бейсик-программу, приведенную ниже. Программа имеет автостарт со строки 2, где происходит загрузка необходимых кодов. Сам блок "теневой контур" надо оформить в виде файла "ten" CODE 64000,211. Кроме того, программа русифицирована по методике, приведенной в ZX-РЕВЮ-92 N1,2. стр.31. Русско-латинский символьный набор загружается в виде файла "chr" CODE 64600,768.

## ЛИСТИНГ\_2

|      |         |        |      |            |                          |
|------|---------|--------|------|------------|--------------------------|
| FA00 | F3      |        | DI   |            | ;                        |
| FA01 | FDE5    |        | PUSH | IY         | ;                        |
| FA03 | FD21FF5 |        | LD   | IY, #5AFF  | ;                        |
|      |         |        |      |            | Последний байт           |
|      |         |        |      |            | файла атрибутов.         |
| FA07 | 0616    |        | LD   | B, #16     | ;                        |
| FA09 | C5      | LOOP   | PUSH | BC         | Организация цикла        |
| FA0A | C620    |        | LD   | B, #20     | для 22 строк.            |
| FA0C | C5      | LOOP_1 | PUSH | BC         | Организация цикла        |
| FA0D | FD7E00  |        | LD   | A, (IY+0)  | для 32 колонок.          |
| FA10 | FE40    |        | CP   | #40        | Наличие BRIGHT 1         |
| FA12 | 380F    |        | JR   | C, #FA23   | в текущем знакоместе.    |
|      |         |        |      |            | Если нет, то переход     |
|      |         |        |      |            | на PASS.                 |
| FA14 | FD7E01  |        | LD   | A, (IY+1)  | Крайнее ли это знако-    |
| FA17 | FE40    |        | CP   | #40        | место с BRIGHT 1.        |
| FA19 | DC52FA  |        | CALL | C, #FA52   | Если да, то выполнение   |
|      |         |        |      |            | вертикального            |
|      |         |        |      |            | оконтуривания - VERT.    |
| FA1C | FD7E20  |        | LD   | A, (IY+32) | Если под текущим знако-  |
| FA1F | FE40    |        | CP   | #40        | местом не BRIGHT 1, то   |
| FA21 | 385D    |        | JR   | C, #FA80   | выполнение горизонталь-  |
|      |         |        |      |            | ного оконтуривания HORIZ |
| FA23 | FD2B    | PASS   | DEC  | IY         | Финишная процедура, если |

|      |          |        |      |           |                             |
|------|----------|--------|------|-----------|-----------------------------|
| FA25 | C1       |        | POP  | BC        | ; знакоместо не с BRIGHT 1  |
| FA26 | 10E4     |        | DJNZ | #FA0C     | ;                           |
| FA28 | C1       |        | POP  | BC        | ;                           |
| FA29 | 10DE     |        | DJNZ | #FA09     | ;                           |
| FA2B | FDE1     |        | POP  | IY        | ;                           |
| FA2D | FB       |        | EI   |           | ;                           |
| FA2E | C9       |        | RET  |           | ;                           |
|      |          |        |      |           |                             |
| FA2F | 7A       | ADRES  | LD   | A, D      | ; Эта подпрограмма выполня- |
| FA30 | E603     |        | AND  | #03       | ; ет расчет адреса в дисп-  |
| FA32 | 07       |        | RLCA |           | ; лейном файле по известно- |
| FA33 | 07       |        | RLCA |           | ; му адресу в файле атрибу- |
| FA34 | 07       |        | RLCA |           | ; тов.                      |
| FA35 | F640     |        | OR   | #40       | ;                           |
| FA37 | 67       |        | LD   | H, A      | ;                           |
| FA38 | 68       |        | LD   | L, E      | ;                           |
| FA39 | C9       |        | RET  |           | ;                           |
|      |          |        |      |           |                             |
| FA3A | 24       | OVER   | INC  | H         | ; Эта подпрограмма выполня- |
| FA3B | 7C       |        | LD   | A, H      | ; ет операции, связанные с  |
| FA3C | E607     |        | AND  | #07       | ; наложением оконтуривающе- |
| FA3E | C0       |        | RET  | NZ        | ; го изображения на тот ри- |
| FA3F | 7C       |        | LD   | A, H      | ; сунок, который имеется на |
| FA40 | D608     |        | SUB  | #08       | ; экране.                   |
| FA42 | 67       |        | LD   | H, A      | ;                           |
| FA43 | 7D       |        | LD   | A, L      | ;                           |
| FA44 | C620     |        | ADD  | A, #20    | ;                           |
| FA46 | 6F       |        | LD   | L, A      | ;                           |
| FA47 | D0       |        | RET  | NC        | ;                           |
| FA48 | 7C       |        | LD   | A, H      | ;                           |
| FA49 | 0606     |        | ADD  | A, #08    | ;                           |
| FA4B | 67       |        | LD   | H, A      | ;                           |
| FA4C | EE58     |        | XOR  | #58       | ;                           |
| FA4E | C0       |        | RET  | NZ        | ;                           |
| FA4F | 2600     |        | LD   | H, #00    | ;                           |
| FA51 | C9       |        | RET  |           | ;                           |
|      |          |        |      |           |                             |
| FA52 | 78       | VERT   | LD   | A, B      | ; Не последняя ли это коло- |
| FA53 | FE20     |        | CP   | #20       | ; нка, если да, то возврат, |
| FA55 | C8       |        | RET  | Z         | ; т.к. негде оконтуривать.  |
| FA56 | FBE5     |        | PUSH | IY        | ;                           |
| FA58 | D1       |        | POP  | DE        | ;                           |
| FA59 | 13       |        | INC  | DE        | ;                           |
| FA5A | CD2FFA   |        | CALL | #FA2F     | ;                           |
| FA5D | DD21BFFA |        | LD   | IX, #FABF | ; Базовый адрес таблицы     |
|      |          |        |      |           | ; для построения оконтури-  |
|      |          |        |      |           | ; вающего рисунка.          |
|      |          |        |      |           |                             |
| FA61 | 0608     |        | LD   | B, #08    | ;                           |
| FA63 | 3E40     |        | LD   | A, #40    | ;                           |
| FA65 | FDBE21   |        | CP   | (IY+33)   | ; Если знакоместо не угло-  |
| FA68 | 3809     |        | JR   | C, #FA73  | ; вое, то переход на LOOP_2 |
| FA6A | 060A     |        | LD   | A, #0A    | ; если угловое, то оконту-  |
| FA6C | FDBE20   |        | CP   | (IY+32)   | ; ривание снизу.            |
| FA6F | 3002     |        | JR   | NC, #FA73 | ;                           |
| FA71 | 060C     |        | LD   | B, #0C    | ;                           |
| FA73 | DD7E00   | LOOP_2 | LD   | A, (IX+0) | ; Проверка необходимости    |
| FA76 | B6       |        | OR   | (HL)      | ; выполнения наложения      |
| FA77 | 77       |        | LD   | (HL), A   | ; на имеющееся на экране    |
| FA78 | DD23     |        | INC  | IX        | ; изображение.              |
| FA7A | CD3AFA   |        | CALL | #FA3A     | ; Выполнение наложения.     |
| FA7D | 10F4     |        | DJNZ | #FA73     | ;                           |
| FA7F | C9       |        | RET  |           | ;                           |
|      |          |        |      |           |                             |
| FA80 | E1       | HORIZ  | POP  | HL        | ;                           |
| FA81 | F1       |        | POP  | AF        | ;                           |

|      |          |      |               |                            |
|------|----------|------|---------------|----------------------------|
| FA82 | F5       | PUSH | AF            | :                          |
| FA83 | E5       | PUSH | HL            | :                          |
| FA84 | FE18     | CP   | #18           | ; Не последняя ли строка.  |
| FA86 | D223FA   | JP   | NC, #FA23     | ; Если да, то негде окон-  |
|      |          |      |               | ; туривать и переход на    |
|      |          |      |               | ; PASS.                    |
| FA89 | FDE5     | PUSH | IY            | :                          |
| FA8B | D1       | POP  | DE            | :                          |
| FA8C | EB       | EX   | DE, HL        | :                          |
| FA8D | 012000   | LD   | BC, #0020     | :                          |
| FA90 | 09       | ADD  | HL, BC        | :                          |
| FA91 | EB       | EX   | DE, HL        | :                          |
| FA92 | 0E02     | LD   | C, #02        | :                          |
| FA94 | DD21CBFA | LD   | IX, #FACB     | ; Базовый адрес в таблице, |
|      |          |      |               | ; определяющей оконтурива- |
|      |          |      |               | ; ющий рисунок.            |
| FA98 | CD2FFA   | CALL | #FA2F         | :                          |
| FA9B | 0604     | LD   | B, #04        | :                          |
| FA9D | DD7E00   | LD   | A, (IX+0)     | ; Проверка необходимости   |
| FAA0 | B6       | OR   | (HL)          | ; выполнения наложения     |
| FAA1 | 77       | LD   | (HL), A       | ; на имеющееся на экране   |
| FAA2 | DD23     | INC  | IX            | ; Изображение.             |
| FAA4 | CD3AFA   | CALL | #FA3A         | ; Выполнение наложения.    |
| FAA7 | 10F4     | DJNZ | #FA9D         | :                          |
| FAA9 | FD7E21   | LD   | A, (IY+33)    | :                          |
| FAAC | FE40     | CP   | #40           | :                          |
| FAAE | D223FA   | JP   | NC, #FA23     | :                          |
| FAB1 | F1       | POP  | AF            | :                          |
| FAB2 | F5       | PUSH | AF            | :                          |
| FAB3 | FE20     | CP   | #20           | :                          |
| FAB5 | D223FA   | JP   | NC, #FA23     | :                          |
| FAB8 | 13       | INC  | DE            | :                          |
| FAB9 | 0D       | DEC  | C             | :                          |
| FABA | 20DC     | JR   | NZ, #FA98     | :                          |
| FABC | C323FA   | JP   | #FA23         | :                          |
| FABF | 000080   | DEFB | #00, #00, #80 | ; Базовая таблица, по ко-  |
| FAC2 | 40A050   | DEFB | #40, #A0, #50 | ; торой строится оконтури- |
| FAC5 | A050A0   | DEFB | #A0, #50, #A0 | ; вающий рисунок.          |
| FAC8 | 50A050   | DEFB | #50, #A0, #50 | :                          |
| FACB | 2A150A   | DEFB | #2A, #15, #0A | :                          |
| FACE | 058040   | DEFB | #05, #80, #40 | :                          |
| FAD1 | A050     | DEFB | #A0, #50      | :                          |

```

1 GO TO 100
2 CLEAR 63999: LOAD "ten" CODE 64000
3 LOAD "chr" CODE 64600
4 POKE 23606,88: POKE 23607,251: REM RUS
100 BORDER 1: PAPER 1: INK 7: BRIGHT 0: CLS
200 BRIGHT 1
300 LET A$="B"
1000 FOR Y=8 TO 12: FOR N=0 TO 15: PRINT AT Y, 1+N;A$;: NEXT N: NEXT Y
1010 FOR Y=3 TO 17: FOR N=0 TO 9: PRINT AT Y, 4+N;A$: NEXT N: NEXT Y
1020 FOR Y=5 TO 15: PRINT BRIGHT 0;AT Y,7;"    ":NEXT Y
1030 PRINT BRIGHT 0;AT 10,3;"    "
2000 FOR N=5 TO 15: PRINT AT N, 18;"    ": NEXT N
2010 PRINT AT 8, 20; "ПРОГРАММА"
2020 PRINT AT 9, 20; "ДЕМОНСТРИ-"
2030 PRINT AT 10, 22; "РЮЮЩАЯ"
2040 PRINT AT 12, 20; "ТЕНЕВОЙ"
2050 PRINT AT 13, 23; "КОНТУР"
2100 PAUSE 50: PLOT 145,48: DRAW 0,86: DRAW 102,0
3000 PAUSE 50: RANDOMIZE USR 64000
4000 BRIGHT 0

```

После старта программы со строки 2, загрузки кодовых блоков и включения русского символьного набора, происходит формирование изображения на экране, которое в дальнейшем будет оконтурено. Для этого перед подачей команд PRINT происходит включение режима повышенной яркости BRIGHT 1 (строка 200). Обратите внимание, что отверстие в левой фигуре, состоящей из букв "В", прорисовывается в режиме BRIGHT 0. В правой части экрана получен прямоугольник, подлежащий оконтуриванию. Это выполняется печатью нескольких строк пробелов в режиме BRIGHT 1 (строка 2000), а текст впечатывается уже потом. Для завершенности фигуры этот прямоугольник слева и сверху замкнут прорисовкой прямых линий (строка 2100), так как оконтуривание производится только справа и снизу.

Если оконтуривание должно быть выполнено поверх символов, напечатанных на экране и имеющих BRIGHT 0, то происходит наложение оконтуривающего фона на имеющееся изображение, это можно наблюдать, если добавить в программу строку:

```
150 FOR N=1 TO 32*22: PRINT "+"; : NEXT N
```

Итак, дело за Вашей фантазией, уважаемые читатели!

# Компьютерная новелла

Дорогие читатели!

С этого номера мы начинаем новую постоянную рубрику в "ZX-РЕВЮ" для тех, кто любит и ценит игровые программы. Эту рубрику мы назовем "Компьютерная новелла".

Изобретатель этой рубрики, он же ее главный редактор - наш постоянный корреспондент из Москвы Матвеев Юрий Александрович. Он экспериментирует в этом жанре уже не первый год и сегодня на Ваш суд предлагается первая работа. В ближайших выпусках Вас ожидают новые увлекательные приключения героев широкоизвестных компьютерных игр.

Должны сказать, что за прошедшие годы мы не раз пробовали экспериментировать с представлением игровых программ. Ключевой проблемой здесь является, на наш взгляд то, что подробная "раскрутка" игры и доведение ее до уровня простейших подсказок (hints) не только не способствует ее популяризации, но и наоборот, навсегда может оттолкнуть от нее возможного пользователя. Как Вы думаете, что скажут альпинисты, если подходы к основным вершинам мира кто-то услужливо забетонирует ступеньками?

Итак, основная цель хорошо составленного описания компьютерной игры должна быть, на наш взгляд в том, чтобы с одной стороны дать заинтересованному пользователю необходимые "хинты" для прохождения игры, но, с другой стороны, сделать это настолько мягко, чтобы они не бросались в глаза при первом прочтении и не портили бы будущую игру.

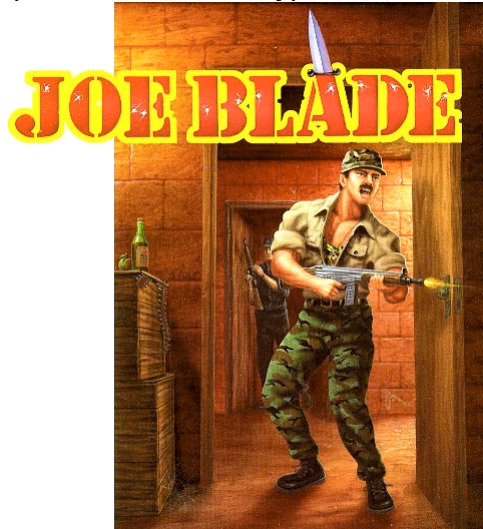
Так родилась идея компьютерной новеллы. Идея понравилась нам еще и потому, что такую новеллу могут читать отнюдь не только те, кто имеют данную программу и желают ее пройти. В принципе, она делается как независимое художественное произведение и может читаться даже теми, у кого нет и не было под рукой персонального компьютера. И если у них появится после этого желание приобрести компьютер, найти и запустить эту игру, вот тогда мы и будем считать, что наша главная цель - достигнута.

У Вас на глазах рождается новый жанр. Мы внимательно следим за новинками в зарубежных компьютерных журналах, но систематического исследования такого направления пока не встречали, т.е. дело это совершенно неисследованное. Вы можете и сами принять посильное участие в становлении и обкатке этого направления. Мы будем признательны за критические отзывы, советы, рекомендации. Может быть у Вас есть какие-то иные оригинальные идеи развития такого жанра. Мы с радостью примем Ваши пожелания, а наиболее интересные и дискуссионные обсудим на страницах "ZX-РЕВЮ". Письма можете направлять по нашему адресу на имя редактора раздела "К.Н." - Матвеева Ю.А.

\* \* \*

## КОНФЕРЕНЦИЯ

(по игре Колина Свинбурна "Джо Блэйд").



I

Джо Блэйд, удобно устроившись на заднем сидении автомобиля, прикурил сигарету и вопросительно посмотрел на мэра города.

- Едем, - сказал тот шоферу и, когда они выехали на шоссе, ведущее к городской тюрьме, наконец повернулся к Джо:

- Ты смелый парень и мне остается надеяться только на тебя. - Он сделал многозначительную паузу. - Ты, конечно, знаешь, кто такой Кракс Бладфингер: влиятельное лицо в городе, президент мощной военно-промышленной компании, почетный член городского магистрата.

Джо усмехнулся. Он никогда не признавал авторитетов, тем более тех, кто стоял у власти. Власть, по мнению Джо Блэйда, всегда ограничивала его свободу, мешала спокойно жить, а свободу он любил больше всего.

Мэр расценил усмешку собеседника как иронию и поспешил согласиться.

- Да, ты прав. Этот человек - оборотень, и не тот, за кого себя выдает. В полиции уже давно под него копают, но... Его организация словно спрут охватила важнейшие точки города. Везде есть его люди. По сути, мэр города - это он. Я - лишь прикрытие и ничего не могу сделать. Все идет так, как хочет он. О чем говорить, если даже городская тюрьма в его подчинении. Есть сведения, что Кракс превратил ее в склад боеприпасов. Также известно, что Кракс Бладфингер нелегально торгует не только оружием, но и наркотиками. У него четко отлаженная сеть сбыта и верная агентура.

Пока мэр рассказывал о Краксе, Джо Блэйд, приоткрыв окно, щелкнул в туман окурком. Мэр не открыл Америки, в народе уже давно ходили слухи о мафии Кракса, однако на официальном уровне это никогда не обсуждалось.

- Я так понимаю, - задумчиво отозвался Джо, - что на сегодняшний день вопрос встал ребром: или вы, или Кракс?

- Не совсем, - ответил мэр. - Все дело в том, что завтра здесь начинается международная конференция по правам человека. Это, можно сказать, крупнейшее событие в истории нашего города. Среди прочих к нам вчера приехали шесть лидеров мирового ранга. В какой-то степени они могут изменить положение. Их влияние в мире, их связи были бы способны очистить наш город от этого паука Кракса и его нечисти. Однако...

- Однако..? - повторил Джо Блэйд.

Вчера ночью Кракс похитил их и теперь шантажирует меня.

- Что? Всех шестерых?

- Да. Впрочем это, наверное, от жадности или от наглости. Для скандала достаточно



было бы и одного.

- Так чего же он хочет?

- Завтра на Конференции я, как мэр города, делаю доклад. Я собирался, когда готовил речь, дать понять о мощной мафиозной структуре, выросшей в нашем городе за последнее время. Я хотел косвенно указать на Кракса, как на главаря этой структуры. И потом...

- Интерпол..? - засмеялся Джо Блэйд.

- Вот именно. Но рукопись моей речи каким-то образом оказалась у Кракса. И он среагировал быстро. Возможно, ему было бы проще устроить мне автомобильную аварию, но я предусмотрительно усилил охрану и стал для него недоступен. Я даже подумать не мог, что он решится на захват заложников. Сегодня Кракс звонил мне и намеком дал понять, что если я делаю доклад в его пользу...

- То есть? - переспросил Джо.

- ...Кракс Бладфингер первое лицо в городе, отстаивающее права человека, борющееся с преступностью, безгранично верное Президенту и стране, гений, ну и так далее. Он хочет добиться благосклонности всех мировых лидеров, которые будут присутствовать на конференции. Он ждет аплодисментов, но он их не получит! - Мэр сжал кулаки.

- Конечно, он же показал теперь всем свое истинное лицо, - улыбнулся Джо Блэйд.

- Нет. Просто вчера, ближе к ночи, сразу после приезда делегатов, он предложил им в рамках ознакомления побывать в городской тюрьме, посмотреть условия жизни заключенных, уровень охраны... До сих пор они не вернулись. Кракс придумал предлог, чтобы задержать их там до начала конференции и только после моего доклада в его пользу они будут доставлены на место. Никто даже не заметит ничего подозрительного. Пятнадцать минут роли не играют. А в противном случае они вообще не появятся в зале и конференция будет сорвана. Обвинят потом, конечно, не его, а меня. Я обеспечиваю транспорт, питание, организацию, безопасность. Кракс выходит сухим из воды, я же - теряю пост мэра, общественность отвернется от меня, а уж потом в долгу не останется и Кракс. Понимаешь? - Мэр посмотрел в глаза Джо Блэйда, ища сочувствия.

Джо молчал.

- Я послал пять лучших агентов: Джаббу, Симона, Криса, Кевина и Колина для освобождения заложников, но они вернулись ни с чем. Кракс усилил охрану, и тебе, парень, будет нелегко. Теперь ты знаешь все. Я верю, ты поможешь нам, тем более, что твои крепкие кулаки и меткий глаз теперь найдут настоящую работу. - Мэр похлопал Джо по плечу. - Какое у тебя оружие?

Джо, не говоря ни слова, вытащил из-за пазухи блестящее короткоствольное ружье.

- Я видел эту штуковину у Кракса на заводе, - сказал мэр.

- Да, - подтвердил Джо, - это их последняя разработка - электронный распылитель.

Оставляет от человека легкое облачко дыма.

- Причем, опасное, - добавил мэр.

- Пока не рассеется, - кивнул Джо.

- Откуда это у тебя?

Джо повертел ружье в руках и спрятал обратно.

- Купил по случаю в одной лавке.

- Пользуешься услугами Кракса, - вздохнул мэр.

- Исключительно против его системы, - ответил Джо Блэйд и посмотрел в окно.

Утренний туман еще не растаял, да и день обещал быть серым и скучным. Но только не для него. Один, против целой банды головорезов, обученных и хорошо вооруженных. Схватка не на жизнь, а на смерть. И это его устраивало. Он считал возможным помочь мэру, да и всему городу, избавиться от хитрого и страшного Кракса Бладфингера.

- Самое главное это заложники, - сказал мэр. - С Краксом мы успеем разобраться. Для заложников ты - сотрудник безопасности. Чтобы снять их подозрения о каше, которая здесь заварилась, скажешь им чтонибудь о взрыве боеприпасов, который вот-вот должен произойти. Наши машины будут ждать недалеко от входа. Помни, что заложники ничего не

знают и чувствуют себя достаточно спокойно. Когда все шестеро будут за пределами базы, найди способ взорвать эту "богадельню", ну а сам сматывай удочки. У тебя будет несколько минут, чтобы унести ноги.

- Ровно двадцать минут, - сказал Джо Блэйд и пояснил: - бомбы Кракса последней разработки срабатывают через двадцать минут после запуска часового механизма.

- Сумеешь разобраться в коде?

Джо кивнул.

- У меня будет тридцать секунд до того, как система самоуничтожения сработает. Этого вполне достаточно, чтобы выставить нужную комбинацию.

Мэр с восхищением посмотрел на собеседника. Он понял, что этот парень знает толк в оружии.

- Запасные патроны есть? - на всякий случай поинтересовался он.

Джо Блэйд отрицательно покачал головой.

- Это только сапожник без сапог, а у Кракса всегда можно найти необходимое. Я надеюсь, что мне хватит двадцати патронов на первое время.

- Ну, смотри сам, - нахмурился мэр и, пристально вглядываясь в даль по ходу шоссе, тронул шофера за плечо. - Тормози, дальше опасно.

Следом за ним остановилась, взвизгнув тормозами, машина черного цвета. Блэйд нацупал ствол.

- Моя охрана, - успокоил его мэр.

Они находились почти за километр от серых кирпичных стен тюрьмы, расположенной чуть в стороне от шоссе на небольшом возвышении. Джо Блэйд осмотрелся. Ни души. Запретная зона. А по сути - одно из владений Кракса Бладфингера. Мэр и Блэйд вышли из машины.

- Советую пройти тем леском, - мэр вздохнул и добавил: - ну, успеха... Я верю в тебя.

Они пожали друг другу руки. Чуть пригнувшись, мэр направился ко второй машине. Хлопнула дверца, и машины почти одновременно рванули с места.

## II

Напевая про себя простенькую мелодию, Джо быстрым шагом пошел по направлению к тюрьме и через несколько минут оказался у высокого забора. Вход был чуть в стороне, справа, но Джо рассчитывал на внезапность. Недолго думая, Блэйд перелез через забор. Он оказался на дорожке, ведущей к одноэтажному кирпичному зданию. Стоявший у входа охранник не сразу сориентировался и не успел выстрелить: Джо Блэйд опередил его. "С почином", - подумал Джо и тут же увидел ключ, видимо выпавший из рук охранника. Он поднял его, подождал пока растворится в воздухе белое облачко и направился к двери. Самое трудное было впереди.

Он оказался в длинном, тускло освещенном коридоре. Здесь никого не было. Крепко сжимая в руках распылитель, Джо быстро шел вперед. Он обратил внимание на странные, в человеческий рост витражи, которые украшали стены. Рельефный металлический, устрашающего вида полу-человек, полу-вампир лязгал клыкастой челюстью. "Бред какой-то", - подумал Джо Блэйд. В конце коридора он заметил две человеческие фигуры. Блестящая каска и армейский мундир сразу бросились в глаза. Это был охранник. Второго он не успел разглядеть. Джо выстрелил в охранника. Когда облако рассеялось, он подбежал ближе. В неярком свете Джо увидел человека в строгом сером костюме с надвинутой на глаза шляпой. Заложив руки за голову, он сидел, прислонившись спиной к каким то ящикам. Похоже, заложник даже не обратил внимания на неожиданное исчезновение охранника и появление человека в куртке, кепке и джинсах. Свидание было коротким. Джо Блэйд представился сотрудником безопасности, кратко и не совсем внятно объяснил ситуацию так, как просил мэр.

- А мы ждем транспорт, - развел руками делегат конференции и робко улыбнулся. - Только я отстал и не знаю, где остальные.

- К выходу, там вас ждут! - почти выкрикнул Джо, уже открывая дверь в соседний коридор.

Сразу два охранника, чуть замешкавшись, бросились к нему. Дверь захлопнулась в ту же секунду и он понял, что обратной дороги нет. Джо уже не успевал уворачиваться от здоровенных детин в армейских мундирах. Дав задний ход, он прижался спиной к холодной стене и выстрелил прямо в упор. Дым рассеялся не сразу. Джо, впечатавшись в стену, ждал, пока воздух очистится от молекул распыленных тел. Однако, как он ни старался укрыться от едкого дыма, нескольких вдохов хватило для того, чтобы закружилась голова, а в ногах появилась слабость. В этот момент он пожалел о том, что не взял обыкновенный "Узи", хотя "Узи" все равно выдал бы его шумом.

На стене висел ключ от тюремной камеры. Джо снял его с крючка и пробежал до конца коридора. Здесь было сразу две двери: одна налево, чуть приоткрытая, другая направо - наглухо запертая тяжелая стальная дверь. Джо вставил ключ в замочную скважину стальной двери и, распахнув ее, шагнул в ярко освещенный коридор голубоватого цвета. Там расхаживал охранник с автоматом наперевес. В секунду его не стало. Коридор был свободен. Чувствуя легкое недомогание после прошлой газовой атаки, Джо открыл первую попавшуюся дверь. Это был склад. Он еще никогда не видел такого количества боеприпасов. Территория тюрьмы служила надежным хранилищем для оружия мафии Кракса. Огромные ящики, стоявшие друг на друге вдоль стен, оставляли лишь небольшой проход по центру. "Да ведь они никогда не будут стрелять здесь", - мелькнуло у Джо. Это было бы самоубийством. Столь простой и логичный довод прибавил ему сил. Он уверенно пошел вперед, внимательно осматривая склад. Неожиданно из-за горы ящиков на Джо бросился толстый улыбающийся кладовщик. В его руке сверкнул нож. Прежде, чем Джо успел выстрелить, лезвие задело висок. Кладовщик метил в лицо, но не попал. Он замахнулся еще раз. Распылитель хлопнул в то мгновение, когда, казалось, длинный кривой нож, как сабля разрубил тело пополам. Но Джо успел.

Пока растворялось облако, Джо переодевался в мундир охранника, найденный среди ящиков с боеприпасами. Тяжелая каска, сапоги, материал, пропитанный специальным составом, пожалуй, могли защитить и от опасных молекул распыленных тел, а уж о том, что в таком облачении Джо Блэйд ничем не отличался от всех остальных охранников, говорить не приходилось. Еще раз внимательно окинув взглядом склад, Джо побежал к выходу.

Двадцати патронов могло не хватить, и Джо это понимал. Он надеялся, что рано или поздно найдет патроны нужной системы. Однако среди этого военного великолепия ничего подходящего он не видел.

Тюрьма поражала своими размерами и Джо Блэйд подумал о плане помещений. Конечно, было бы не плохо его иметь, но, как всегда, приходилось рассчитывать только на себя. К тому же нужно было экономить время, патроны и силы. Пару раз Джо удалось проскочить незамеченным. Но даже когда он лоб в лоб столкнулся с охраной одной из камер, никто не обратил на него внимания. Его спасал мундир, найденный на складе.

Вскоре он нашел бомбу. Огромный шар черного цвета лежал прямо в проходе. Джо хотел оставить ее незаряженной, но, перепрыгивая через бомбу услышал щелчок, сигнал зуммера, и понял, что жить ему осталось ровно тридцать секунд. Высокочувствительный механизм срабатывал от малейшего движения. Код состоял из пяти букв и нужно было выставить определенную комбинацию. Шифр Джо знал. Один из продавцов оружия рассказал как-то о новой разработке военных - супербомбе, и сказал, что пока идут опытные образцы. Шифр прост - A B C D E. Именно в такой последовательности следовало выставить буквы. Сложность заключалась только в том, что передвигать буквы по табло можно было лишь по заданной схеме. Таймер безжалостно отсчитывал секунды. Когда до самоликвидации бомбы оставалось пять секунд, Джо установил последнюю букву на место и услышал длинный монотонный сигнал. На счетчике засветились цифры 20.00 и начался обратный отсчет времени. Теперь остановить таймер не смог бы и Господь Бог.

Почти одновременно с сигналом неожиданно выстрелило ружье Джо Блэйда. "Что за черт?" - подумал Джо. На один патрон, действительно, стало меньше и Джо понял, что сильный электромагнит в бомбе заставил сработать электронный боек его ружья. "Такими темпами у меня через пять минут будет пустой магазин". Но Джо решил, что лучше пожертвовать патронами, чем опоздать, и не стал особо переживать. Ему нужно было найти

еще пятерых заложников, да и одной заряженной бомбы на тюрьму явно не хватит.

В следующем помещении Джо понял, что от военной формы придется отказаться. Здесь охранники были одеты в ту же форму, но другого цвета. Он не рискнул нарушить порядок и в надежде на новую форму стянул с себя старый мундир. Пробежав через коридор и стараясь быть незамеченным, Джо нырнул в открытую дверь. Тюремщик с большим животом видимо собирался вкусно поест. На столе стояла бутылка воды, в широкой вазе лежали фрукты. Джо не был голоден, но знал, что при молекулярном отравления лучше всего помогают именно вода и фрукты. Тюремщик не успел даже вскочить из-за стола. "Ради этого стоило потратить патрон", - подумал Джо, доедая яблоко и запивая его водой. На часы он не смотрел. Лишняя суета - плохой союзник. Подкрепившись, он сразу почувствовал себя лучше.

Он плохо помнил, как в течение двух минут успел найти и зарядить еще две бомбы, а также освободить ничего не подозревающего делегата.

В длинных коридорах тюрьмы он редко стрелял, предпочитая уворачиваться от ударов и пробегать мимо охранников там, где это было возможно, зная, что ни один не решится выстрелить в набитой до отказа боеприпасами тюрьме, Джо просто прыгал как пантера через ошалевших бойцов и рвался вперед.

Он еще раз нашел новую, с иголки, форму и уже в ней, давая себе передышку, вышел на улицу во внутренний двор тюрьмы. Теперь стало ясно, что заложников развели по разным зданиям и коридорам. Каждый, видимо, думал, что отбился от основной группы. Охрана относилась к ним лояльно и поэтому бунта не возникало. Джо посмотрел на пасмурное небо и подумал о своем возвращении. Бежать из тюрьмы можно было только через выход, минуя коридоры и охрану. Дорогу назад он помнил. В одной из камер он оставил нетронутыми еще один мундир и каску, которые позволят уйти незаметно.

Только сейчас он рассмотрел на часы. До взрыва оставалось одиннадцать минут.

Убрав охранника возле небольшого кирпичного дома, Джо с радостью обнаружил внутри боекомплект для своего оружия. Десять патронов, тщательно упакованные в небольшом ящике, быстро переключались в магазин распылителя. На душе стало легче. В соседней комнате он нашел безмятежно спавшего мирового лидера. Джо объяснил ситуацию. Тот быстро согласился и направился бодрым шагом к выходу. Джо знал, что этот серый костюм никто не тронет и он спокойно найдет выход и всех остальных. Спустя несколько секунд Джо чуть было не поплатился жизнью, оказавшись один на один в тесной камере с каким-то заключенным. Вплотную прижав Джо к стене, он принялся душить его. Схватка была короткой, но решительной. Силы уже покидали Джо Блэйда, когда ему удалось, оттолкнувшись от стены, перепрыгнуть за спину противника и, быстро развернувшись, выстрелить тому в живот.

Сквозь облако испарений Джо увидел, что входная дверь захлопнулась и он оказался в каменном гробу из четырех стен. Пот лился градом, а Джо лихорадочно шарил по карманам. Наконец ключ был найден. Слава Богу, что он зашел сюда с запасным ключом, иначе пришлось бы ему последние десять минут жизни провести за тюремными стенами. Джо снова вышел во внутренний двор. Слева от него был целый ряд самых разных построек, но, чтобы проникнуть туда, нужно было пройти мимо центрального поста, а потом уж повернуть налево. Бдительный охранник получил свою порцию смертоносного луча, когда пытался задержать Джо у входа. Блэйд подошел к дверям. Их было две. Он открыл правую и на этот раз интуиция не подвела его: в одной из трех тесных комнат Джо обнаружил комплект боеприпасов для распылителя, немного еды и уставшего заложника. Полдела сделано. Ему осталось осмотреть ту часть построек, которые находились слева от центрального поста. Дверь еще не успела открыться, как в лицо Джо выдохнул перегаром стоявший напротив охранник.

- Стой! Кто такой?

Вопрос в лоб. Ответ был достоин вопроса. За Джо заговорило его ружье. Легкое облако быстро разносил ветер. "Интересно", - подумал Джо, - "они уже ищут меня или нет?" Судя по всему, информация о проникшем на территорию тюрьмы постороннем не поступала. Джо застал всех врасплох. Чуть больше десяти минут находится он здесь, а

кажется, что прошла целая вечность.

У входа в первое здание, слева от главной дороги, сразу два солдата угрожающе шагнули к Джо. Он выстрелил. Чуть подумав, решил не идти через дверь, а влез в окно.

Длинный зеленый коридор показался ему сплошным кошмаром. Он изрядно потратил патроны, прежде чем добрался до черной двери. Здесь он нашел сразу несколько ключей: они обеспечивали свободу перемещения и были нужны, как воздух. Три ключа Джо оставил в запасе для возвращения обратно. Вскоре он нашел еще двух заложников, отдыхавших по разным концам коридора. После прощания с ними Джо в темпе осмотрел несколько камер. В одной из них он нашел бомбу и на этот раз запустил ее без труда. Минуты летели с невероятной быстротой. Шесть кругов оставалось пройти секундной стрелке до взрыва, а он еще не нашел последнего, шестого делегата конференции. Эх, знать бы, где он!

А заложник был недалеко, в соседнем здании. Джо не стал ничего объяснять. Показав ему дорогу к выходу, он помчался по коридору в надежде найти патроны. патронов он не нашел, а за две минуты успел зарядить еще две бомбы. "Подарок Краксу", - подумал Джо, - "шесть бомб - шесть заложников. Меняемся!", - он улыбнулся своим мыслям, однако спустя несколько секунд понял, что радоваться рано. Возвращаясь к центральному посту, Джо наткнулся на вновь прибывшую смену. Не раздумывая, он нажал на курок, но ружье молчало. Патроны кончились, Джо повернул назад. К нему шли еще двое. В отчаянии он применил свой коронный прием: прыжок с выброшенной вперед ногой заставил чуть наклониться охранника и Джо буквально перелетел через него. Тот бросился сзади, но не успел - Блэйд бежал быстрее.

Три ключа были в кармане. Где-то в глубине самого первого здания его дожидался спасительный мундир. Нужно было только успеть. Теперь, видимо, информация о нем дошла и его, наверняка, попытаются либо убить, либо взять живым. Выйти из тюрьмы ему не дадут...

...За две минуты до взрыва Джо в форме охранника бежал по коридору к выходу. На него не обращали внимания. Все искали коренастого в штатском.

Последний ключ он вставил в дверь за полторы минуты до взрыва. Дорожка с которой он начинал свою миссию, теперь показалась ему слишком длинной. Но все! Он успевает!

Черный лимузин увозит его прочь от обреченной тюрьмы. В машине, кроме шофера, он один. Все остальные готовятся к конференции и лишь немногие знают, что через минуту тюрьма Кракса перестанет существовать.

Все остальное будет потом: награда, всеобщее признание, свобода, наконец. А в эту минуту Джо Блэйду хотелось одного: откинувшись на спинку сиденья, вдохнуть глубоко воздух и дожидаться красного зарева и раскатов грозы над тюрьмой. Джо все еще видел в зеркало заднего вида очертания тюремных стен и построек. Он молча ждал. Шофер тоже молчал. Только мягкое гудение двигателя да шорох шин нарушали тишину.

# СОВЕТЫ ЭКСПЕРТОВ

MYTH. History in the making



Автор: Neil Dodwell, 1989  
Фирма: THE SYSTEM THREE

Эксперт Матыцын Д.А.  
г. Москва

Эта игра является довольно развитым представителем жанра аркадных адвентюр и требует от играющего как быстрой реакции и сообразительности, так и логического мышления.

Фабула игры основана на том, что по неизвестным причинам нарушилась ткань времени и в нашу эпоху стали проникать мифические создания и герои древних легенд. Для исследования причины такого смещения Вам поручается провести историческую экспедицию и побывать в некоторых узловых точках истории человечества.

Игра многоуровневая, с пятью дозагрузочными этапами. Для перехода с одного уровня на другой, необходимо собрать пять энергетических шаров. Первые четыре из них обычно находятся довольно легко, последний же, как правило, хитроумно запрятан или защищается могучим стражем.

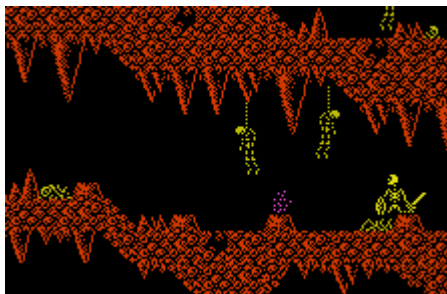
В верхней части экрана индицируется количество оставшихся попыток и очки (слева), жизненная энергия героя и количество собранных шаров (справа), а также задействованный в данное время предмет (в центре). Когда герой теряет более 80% энергии, его изображение начинает мерцать.

В начале каждого уровня дается обычно один предмет, используя который Вы добываете и другие. Смена предмета - нажатием клавиши SPACE. Кнопка "огонь" совместно с "вправо" или "влево" позволяет использовать текущий предмет.

I  
"The Road to Hell".  
("Дорога в ад")



Итак, Вы в далеком прошлом. Ваш маршрут начинается под старым кладбищем. Единственное исходное оружие - удары ногой или рукой. Они выполняются нажатием клавиш "вправо", "влево" или "вверх" при нажатой клавише "огонь". Ногой открываются сундуки и разбиваются сосуды, внутри которых могут быть найдены некоторые полезные атрибуты, восстанавливающие энергию героя, дающие временную неуязвимость, дополнительные заряды и т.п.



Основные опасности - скелеты, монстры, извергающие огонь, смертоносные вулканы. Первые три энергетических шара собрать легко - найдя шар, надо выстрелить в прыжке. Шар упадет и его можно взять. С остальными шарами дело обстоит сложнее...

Расстреляв не менее десяти ходячих скелетов, подберите оставшиеся от них черепа, спуститесь на самый нижний этаж и идите влево к огненному озеру. Там бросьте эти черепа в пламенные воды. Из пламени возникнет демон. Победив его, Вы сможете по возникшему островку добраться и до четвертого энергетического шара. На этом же островке надо взять мощное оружие - трезубец и идти к другому огненному озеру. Из середины озера поднимется островок, который позволит дойти до гигантского огнедышащего дракона, победить которого Вам поможет волшебный трезубец, а боевым трофеем станет ключ, которым открывается дверь пещеры, в которой хранится последний шар.



Последняя задача - переход на новый уровень. Портативный телепортер Вы найдете на нижнем этаже, неподалеку от огненных озер. А использовать его надо там, откуда Вы начинали игру (место отмечено воротами, на которых есть изображение Вашего телепортера).

## II

"Greece. Four Hundred BC".

(Греция. 400 лет до н. э.)

Пронизывая пласты времени, Вы попадаете в 400 год до н.э. И вот перед Вами древняя Греция. Свой первый шар Вы найдете без труда, а уничтожив мечом пару колонн, подберете спрятанные за ними атрибуты и волшебный мешок. Он пригодится впоследствии для транспортировки смертоносной головы Медузы Горгоны.



Если Вы сумеете успешно отразить атаки агрессивных привидений и минуете статую льва, то попадете в уютный дворик, в конце которого отдыхает прекрасная нимфа. Однако, при Вашем приближении она превратится в огнедышащего монстра, единственное спасение от которого - спрятаться за колонну. Вдоволь натешившись, нимфа улетит, а Вы станете обладателем еще двух энергетических шаров.



С опаской поглядывая вверх, откуда начал капать смертельно ядовитый дождь, подойдите к пустому портику. В него можно войти и передохнуть, ненадолго присев.

Ваша следующая задача - Медуза Горгона, охраняющая четвертый шар. Хорошо, что у нас есть бронзовый зеркальный щит - им можно прикрыться от ее испепеляющего взгляда. Когда возьмете шар, заодно отрубите Горгоне ее голову и осторожно положите ее в свою волшебную сумку.

Если на обратном пути хорошо обыскать пещеры, Вы непременно найдете огромного трехголового дракона. Быстро достаем голову медузы Горгоны и с ее помощью уничтожаем одну за одной все три головы страшной рептилии. Победив дракона и забрав пятый шар, Вы без труда вернетесь за телепортером в пещеру Горгоны, возвратитесь к месту старта и, немного отдохнув, продолжите свою экспедицию.

### III

"Scandinavia. Five Hundred AD"

(Скандинавия. 500 год н.э.)

Теперь мы попадаем к норманнам в Древнюю Скандинавию. Очутившись на корабле Драккаре в разгар свирепого шторма, сопровождающемся громом и молниями, Вы не сразу сообразите, что от Вас требуется.



Осмотрите, опробуйте свой огромный топор и можете приступать к битве с командой корабля. Если сражение пройдет нормально и Вы победите главного викинга - конунга, то взяв его оружие, Вы сможете приобрести первый энергетический шар, после чего окажетесь уже на суше.



Здесь Вас ждет толпа злобных гоблинов, вооруженных дубинами. Трофеями этой победы будут ножи, которые впоследствии Вам пригодятся. Изредка вашим взорам будут представлять прекрасные лесные феи - брумгильды, но они не менее опасны, чем гоблины. Идя налево и разбивая по пути ящики и сосуды, Вы найдете второй энергетический шар, после чего можно начать исследовать правый фланг.

Погасить жертвенный костер Вы сможете, вызвав дождь с помощью заклинания из старинной рукописи. На месте пепелища Вы найдете ключ от замка скандинавского бога Одина.





Двигаясь направо, Вы придете к третьему шару, но дальнейший путь преградит злобный дракон Нидхог, справиться с которым Вы сможете только метанием ранее собранных ножей с достаточно близкого расстояния.



Четвертый шар Вы найдете на подступах к замку Одина. Попастъ в замок Вы сможете, опустив подъемный мост через ров, воспользовавшись ключом из жертвенника. В замке Вас встречает сам бог Один. Есть только один предмет, которым можно его поразить. Интересно, успели Вы найти к этому моменту молнию? Если да, то победа и пятый шар будут Вашими.

Телепортер найдете на обратном пути, а пользоваться им Вы уже умеете.

#### IV

"Egypt Three Thousand BC".  
(Египет. 3000 лет до н. э.)



На этот раз Вас забросили в историю еще дальше. На календаре трехтысячный год до нашей эры. Древний Египет. Выбравшись из великой реки Нил, подойдите к пирамиде. Не правда ли, войти туда не удастся? Как будто неведомая сила отталкивает Вас от входа. К счастью, в карманах заваялся револьвер и пары выстрелов достаточно, чтобы разнести дверь в щепки. Теперь путь свободен! Мы попадаем в пирамиду и спускаемся вниз.

Главный мотив этого уровня лабиринт. Отсюда пути ведут в разные стороны. Слева - четыре ниши, в каждой из которых начинается своя часть лабиринта, а справа от развилки - пятая, которая откроется после прохождения первых четырех ветвей. На каждой ветви Вы найдете по одному энергетическому шару и по одному древнему предмету - артефакту.



Начнем свое исследование с самого левого хода. Сразу подберем первый шар. Уворачиваясь от острозаточенных маятников и вскрывая по пути сундуки и сосуды

доберемся до первого артефакта.

Вернувшись назад, пойдём во вторую ветвь и бежим направо. Здесь по шару надо выстрелить издалека. Ближко не дайдут подойти острые шипы, торчащие из пола. Впрочем, после того, как шар упадет, шипы задвинутся обратно. Забираем его и в левом тупике второй ветви берём второй артефакт.

В третьей ветви у Вас есть интересная возможность восстановить все жизни своего героя. Это делается с помощью магического креста. Надо использовать его, стоя под изображением аналогичного креста. Стоит время от времени приходить сюда на "подзарядку".

Как и в прочих ветвях, здесь Вы также найдете энергетический шар и артефакт.

Четвертая ветвь довольно проста, стоит опасаться только шипов в полу. Берем шар и артефакт, после чего возвращаемся на развилку.

Вы увидите, что изображение глаза в пятом проходе начало мерцать. Вставьте на это место и используйте такой же предмет, как и глаз - попадете в начало длинного туннеля. После тяжелой борьбы Вам удастся прорваться к золотому саркофагу. Здесь надо выложить все четыре артефакта и тогда Вы встретите основного противника - огромную золотую маску Рамзеса. Победить ее можно с помощью припасенной ранее головы фараона.



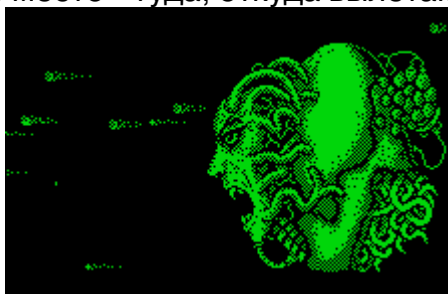
Дальше все просто: взять последний шар, найти телепортер и, собравшись с духом, переходить на следующий уровень.

Примечание: на четвертом уровне следует как можно чаще передвигаться прыжками, а не бегом. Здесь много мест с падающими стенами, с шипами в полу, с острыми секирами, которые при первом же прикосновении начинают с угрожающей быстротой раскачиваться, как маятники.

## V

### "Duel with Dameron" ("Дуэль с Дамероном")

Все дальше и дальше мы проникаем вглубь времени. Уже не видно и людей, вот и Земля куда-то пропала. Мы летим в бесконечном космосе. Неожиданно перед нами поднимается злой демон времени - ангел Хаоса Дамерон. Теперь понятно, кто виновен во всех неполадках. Ничего не остается, как сразиться с ним. Победить его непросто. Уворачиваясь от летящих в Вас огненных шаров и тщательно прицеливаясь, Вы поражаете Дамерона в единственно уязвимое место - туда, откуда вылетают его шары.



Но он не погиб. Потеряв значительный кусок своего тела, он перестраивается и продолжает извергать смертельные заряды. Снова и снова отбиваете Вы от него кусок за куском и, наконец, враг повержен.

Так заканчивается Ваша историческая миссия. Ткань времени восстановлена, цивилизация спасена!

# THE DARK WHEEL

Продолжение. Начало см. в ZX РЕВЮ-92, стр. 175-176, 218-220, 262-264.

## ГЛАВА 5

Выход из подпространства как всегда сопровождался тошнотой, потерей ориентации и головокружением. Легкая дрожь охватила тело. Прямо перед ними, на приличном расстоянии тускло светил красно-голубой диск планеты Ксезавр. Он не очень выделялся на фоне сверкающих звездных полей. Тусклое солнце этой системы находилось неподалеку. Умирающая звезда с трудом проливает красный свет на этот застывающий мир. И никакая индустриальная и научная мощь планеты не в силах остановить этот неумолимый процесс старения галактики.

Да, Ксезавр превратился в мир, в котором уют и тепло ценятся превыше всех благ на свете. Вот где они получают достойную цену за те шанаскильские меха, которыми загружен их корабль.

Как обычно, все шло по раз и навсегда установленному порядку. Подходит к концу еще один скучный торговый рейс. Элиссия сладко зевнула, а Алекс ввел замеры координаты в автопилот и приготовился убить время, необходимое для скучного подхода к планете.

Однообразный, установленный порядок, к которому Алекс уже давно привык: выход из подпространства, медленный перелет, швартовка у станции "Кориолис", - и нечего делать, и не на что смотреть.

Вдруг "Кобра" вздрогнула, эхом в боевой рубке отозвался скрежет разрываемого металла.

- У нас компания, - вскрикнул Алекс.

Элиссия очнулась от сна и захлопала удивленными глазами, стряхивая остатки дремоты. Она поняла ситуацию в мгновение ока, но осталась неподвижной в своем кресле. Алекс был у консоли и уже владел ситуацией.

В какой-то степени его поймали врасплох. И дело вовсе не в элементарной невнимательности, дело в том, что атакующие корабли оказались необычно близко к точке выхода из гипертуннеля. Более того, они занимали положение между объектом своей атаки и солнцем, так что какое-то время оставались невидимыми и имитировали при этом движение астероидов, беспорядочно кувыркаясь и дрейфуя в пространстве.

Алекс в общем-то их видел, но проигнорировал и теперь его "Кобра" приняла на себя первый залп.

Противники сгруппировались сзади и образовали боевой порядок. Непрерывно сканируя пространство, Алекс набирал максимальную скорость.

Вновь заскрежетали энергетические щиты, на которых заиграло лазерное пламя очередного залпа.

- Лучевые лазеры! Да, эти корабли неплохо вооружены. - подумал Алекс, - но и наша "Немезида" теперь им не уступит!

"Немезида" - такое драматическое имя древней богини мести дали Алекс и Элиссия своему кораблю.

Алекс включил экран заднего обзора, подогнал прицельный квадрат и дал две коротких вспышки из новехонького кормового лазера.

Корабли противников сломали строй и разошлись. Один из них закувыркался. Пока враг был на экране, Алекс нацелил ракету.

О приближении вражеской ракеты оповестил экран, вспыхнув сообщением об опасности, впрочем Алекс видел ее и так. Он немедленно привел в действие систему ЕСМ и после томительных секунд ожидания ракета исчезла в жаре и пламени.

Тряхнуло корпус и Алекс нырнул вниз. Краем глаза он успел заметить сброс энергии

защитных экранов в первый энергетический отсек.

Элиссия спокойно сидела в своем кресле. Алекс уверенно владел ситуацией. Прямо перед ними стремительно нарастал диск планеты. Казалось, что он то взлетал, то падал, вращаясь при этом с головокружительной скоростью - это Алекс маневрировал, занимая наилучшую боевую позицию.

Он действовал почти инстинктивно. Резко развернув свой корабль, он вдруг решительно бросил его навстречу ближайшему преследователю. Это оказался "Фер-де-Ленс", быстрый и изящный корабль, по-видимому сверху донизу забитый самым изощренным навигационным и боевым оборудованием, установленным его прежним владельцем. Впрочем, может быть и нет... Для содержания такого оборудования необходима куча денег, а этот корабль уже многое повидал на своем веку.

Сближаясь с пиратом, Алекс решил применить ракету. У них на борту было четыре ракеты, причем одна из них уже была нацелена. Он нажал на пусковую клавишу, "Кобра" вздрогнула, смертоносный снаряд пошел к своей цели. Она нашла ее и "Фер-де-Ленс" буквально исчез.

"Может ушел в гиперпереход?" - подумал Алекс, - "Вроде не похоже". На самом деле он проскочил мимо противника и теперь, включив задний экран, увидел огромное отливающее серебром облако, быстро расширяющееся навстречу звездам.

- Отличная стрельба! - вдохновенно откликнулась Элиссия.

Из облака пепла и металла проступили контуры другого корабля и Алекс вновь ушел в петлю. Новый лазерный заряд принял на себя кормовой экран. Теперь, когда охотник знал, что его жертва обладает системой подавления ракет, он решил "сесть на хвост" и добивать ее лазером.

Это тоже была "Кобра". Грузоприемники широко распахнуты, готовые подобрать контейнеры с драгоценный шанаскидским мехом разбитого торговца.

Но у Алекса на сей счет были иные планы. Маневр. Ксезавр опять перед ними. Огонь кормового лазера. Нырок вниз, разворот, выход в непростреливаемую зону и Алекс нацелил новую ракету.

- Не надо, побереги ее, - с сожалением вздохнула Элиссия.

- Я знаю, но нам хватит денег на замену, - сказал Алекс.

- Тогда не хватит на энергозаборники, - напомнила Элиссия и оба от души расхохотались. В такое время они думали о новых покупках!

До станции с ее системой обеспечения безопасности оставалось еще слишком далеко. Алекс сделал крутой вираж к солнцу и резко сбросил скорость. Преследователь идеально повторил первый маневр, но на несколько секунд запоздал с ориентацией и не успел затормозить. Он не успел еще понять, что произошло, а охотник и жертва уже поменялись местами.

- Давай, Алекс, давай! - кричала в восторге Элиссия, а Алекс вгонял один за другим лазерные заряды в тело вражеского корабля. "Кобра" на экране крутилась и изворачивалась, но Алекс не думал, он только реагировал и методично добивал противника. Температура носового лазера опасно повысилась. Пират выпустил ракету, но Алекс не стал отвлекаться на программирование ЕСМ и спокойно расстрелял ее лазером.

У Элиссии перехватило дыхание и она в восторге смотрела на юношу, уверенно державшего в руках их жизнь и судьбу.

Еще через мгновение все кончилось. Пират взорвался, энергии его защитных полей не удалось сдержать такой натиск. Алекс успел увидеть мелькание спасательной капсулы и на мгновение вспомнил ту вспышку огня, которая уничтожила его собственный корабль, он вспомнил гибель родной "Авалонии".

Его первой реакцией было броситься в погоню, но подумав, он успокоился и отказался от этой идеи. А вокруг сверкали и кувыркались контейнеры с разбитого корабля.

- Подумать только, какая жалость, что у нас до сих пор нет грузозаборника! - пробурчала Элиссия.

Зато мы получим премию за пиратов, - ухмыльнулся Алекс, - и притом за двоих.

- Алекс, ты был великолепен. Летать с тобой между звезд - большая честь.

Алекс спокойно вел свой корабль к планете и больше никто не сказал ни слова. Они даже не упомянули тот факт, что это было первое настоящее сражение, которое он провел без ее помощи.

## ГЛАВА 6

Они уже торговали в течение трех месяцев и теперь в "Немезиде" трудно было бы узнать ту "Кобру", которая когда-то была надгробным камнем Генри Белла. Покрашенная, расцвеченная опознавательными знаками, оснащенная новыми боевыми надстройками, она все более и более походила на боевой корабль.

Три месяца торговли и ни на час Алекс не забывал о главной цели своей жизни. Кто-то или что-то, замаскировавшись под мирного торговца, убил его отца и сделал это преднамеренно. Да, его отец вел сложную жизнь и, согласно неписанной галактической традиции, сделал все возможное, направив и сына по тому же пути. И Алекс Райдер оправдывает веру отца.

Вопросов перед Алексом возникало не меньше, чем боли и гнева. То же состояние переживала и Элиссия, хотя она, как и все женщины Теорга, редко проявляла свои эмоции. Тем не менее, Алекс чувствовал кипение горячих чувств под холодной внешней оболочкой.

У них была общая задача. Они должны были выжить, стать сильными и отомстить. Им предстояло долго вместе ждать и оба понимали, что лучшего и более молчаливого партнера для своей цели им не найти. В то же время, им обоим было нелегко...

Встреча с пиратами немного повредила кораблю. Здесь и там пострадала краска, разболтались некоторые панели облицовки боевых надстроек. Это означало необходимость посещения станции обслуживания. Впрочем, этой возможностью стоило воспользоваться, поскольку им, как победителям пиратов, этот сервис полагался бесплатно.

Это сражение для Алекса было первым самостоятельно выигранным боем, но это был не первый их бой. Если бы Элиссия не скрывалась, она вполне имела бы право подавать заявку на ранг "Опасный". А так как ей приходилось сохранять свое инкогнито, все победы заносились в реестр Алекса. Сегодняшний бой Алекс воспринял как значительный шаг вперед и впервые у него появилось чувство, что он по-праву носит тот ранг, который ему присвоили благодаря успехам Элиссии.

Он направил свой корабль внутрь тысячекилометровой зоны над поверхностью планеты и ее диск заполнил собой весь экран переднего обзора. Корабль развернулся на самой малой скорости и медленно вращающийся металлический куб засверкал перед ними - планетная станция с распахнутым зевом приемного отсека.

- Кстати о стыковочном компьютере.., - начал было Алекс, уравнивая скорости вращения корабля и станции.

- Пустая трата денег... - пресекла его попытку Элиссия. - Если ты не можешь стыковаться так, чтобы не ободрать краску с бортов, тебе вообще нечего делать в космосе.

Алекс был прекрасным пилотом, но вход в приемный отсек станций типа "Кориолис" был его самым слабым пунктом. Он, конечно, сделал это и на сей раз, но вздохнул спокойно только тогда, когда корабль, вошедший в просторный ангар, был подхвачен магнитным манипулятором и притянут к свободному стапелю. Тут же стыковочная штанга автокома скользнула вдоль корпуса и подключилась к разъемам корабля.

Пока Элиссия как обычно пряталась в спасательной капсуле, Алекс с интересом рассматривал кубическое внутреннее пространство станции: таможенные, полицейские, рекламные корабли, ремонтные модули, - все это медленно перемещалось в пространстве. У каждого свое дело, каждый знал свой бизнес.

По каналам автокона Алекс доложил о грузе на борту и получил подтверждение регистрации победы над пиратами и свидетельство начисления призовых тридцати кредитов. Это как раз то, что нужно для покупки новой ракеты.

Когда все формальности проверок и перепроверок были закончены, Элиссия наконец покинула свое убежище. Так уж получилось, что именно спасательная капсула стала для них предметом первой необходимости, без которого они не могли бы посетить ни один порт. Им удачно удалось купить поддержанную всего за четыре сотни и само собой

разумеется, что они не собирались использовать ее по назначению.

Теперь начались обычные торговые процедуры. Сначала продажа, затем обсуждение очередного рейса и, наконец, покупка нового товара.

Торговля во многом напоминает игру. При известном спросе и быстром обороте всегда можно получить небольшой, но надежный доход на продуктах, текстиле, простейшем промышленном оборудовании и незамысловатых безделушках для богатых. Но непрерывно возрастающая стоимость обслуживания корабля и потери от периодических стычек очень скоро съедают эту скромную, с таким трудом полученную прибыль, после чего все предприятие становится практически бессмысленным. Узнать цену на те или иные товары в другой звездной системе нет никакой возможности. Все правительства ревниво оберегают коммерческую информацию о своих рынках. За передачу информации о ценах по каналам факс-связи следует очень тяжелое наказание.

Естественно, цены нестабильны. Во всех системах, даже в самых бедных, орудуют сети спекулянтов. Еще вчера за тонну пусторесничника, купленного на Реорте за три креда давали на Цейнзиле по восемь, а сегодня с трудом покупают всего за два. И дело вовсе не в том, что спрос на пусторесничник сократился. Просто банда спекулянтов сыграла свою игру и подмяла рынок.

До сих пор Алексу и Элиссии везло. Они перевезли варгорнские шелка от Рексебе на Инеру и удвоили свою первую сотню кредитов. Потом они торговали золотистой чешуей геретеанских рептилий и с трудом вернули затраченное. Они поставили двадцать тонн семян подсолнечника амфибиоидам на Биерле, для которых это был изысканный деликатес и лишь прибыв к месту назначения с удивлением обнаружили, что всепланетная мутация изменила вкусовые железы обитателей... и поставила последних перед проблемой поиска новых деликатесов. Достаточно близко подходили для этой цели ароматические масла и лавандовая туалетная бумага, но все же это не то. Где-то все-таки должна быть возможность НАСТОЯЩЕГО дохода и рано или поздно он придет.

Переброска оборудования из высокоразвитых миров в среднеразвитые оказалась весьма прибыльным делом. Алекс и Элиссия быстро поняли также, что на развивающихся промышленных планетах всегда можно иметь хороший доход от торговли предметами роскоши, но самой прибыльной обещала стать операция с продажей шанаскильского меха на Ксезавре. Они купили его по тридцать кредитов за тонну и сейчас Алекс в нервном напряжении ожидал ответа на запрос биржевых сводок по мехам.

Он не удержался и в триумфальном жесте вскинул руки, когда понял, что они с Эмиссией утроили свой капитал. На этот раз им улыбнулось настоящее счастье.

Мех они продали без проблем и запросили сводку цен на бортовое оборудование и вооружение. Новая ракета стоила свои стандартные тридцать кредитов. Алекс заказал одну и автоматический робот приступил к исполнению заказа. Лучевые лазеры по тысяче за штуку являли собой почти непреодолимое искушение. Топливо и совершенно необходимое энергогрузозаборное устройство обходились в пятьсот двадцать пять, а энергетическая бомба стоила еще вдвое больше.

Конечно, энергозаборники обещали большую экономию. Можно будет в дальнейшем не тратиться на горючее, а пополнять баки, подпитываясь от звезд. Это было хорошее вложение, даже при том, что цены здесь были на сотню выше, чем обычно. Алекс заказал одно устройство. На доставку и монтаж уйдет примерно двадцать часов (один стандартный день). Заправив корабль, Алекс приступил к изучению списка Ксезаврианских деликатесов. У них еще оставалось триста двадцать кредитов - сумма, с которой чувствовать себя было неуютно. С другой стороны, корабль имел дополнительную энергетическую защиту, лазеры с четырех направлений, полный боекомплект ракет и грузозаборник. Пройдено более половины пути превращения торговой "Кобры" в боевой крейсер.

Элиссия тоже внимательно изучала список. Удивительно, но ксенозаврианцы, большие любители экзотики, мало что могли предложить в этом плане. Так, например, всего два вида наркотиков было на рынке: арктурианский лопнитрав и, как ни странно... табак. Алексу было над чем подумать.

- Думаю, что с табаком можно было бы попробовать...

- Не-а, - промолвила Элиссия. - Не выйдет, никотин смертелен в малых дозах для большинства рас.

- А если в гуманоидную систему?

- Все равно опасно.

Широко предлагались минералы, но цены были не те. Дюрасьон - руда, из которой после очистки и старения получают дюралиум для строительства кораблей, шел всего по восьми за тонну. А он исключительно хорошо продается на Лейве, но до Лейва отсюда столько световых лет! Не подходит.

Драгоценные камни? Предлагались каштанит и серебристый спектонал. Еще были красно-зеленые эмеронды. Но пираты учуют такую добычу даже за пару световых лет.

Чисто из любопытства Элиссия обратила внимание на продажу двухсот окаменевших костей диринофаксаурина по сорок кредов за штуку.

- Когда нибудь слышал о них?

Алекс ответил:

- Даже видел одну в музее у себя дома. Они живые и поют. Им более сорока миллионов лет и, тем не менее, они поют. Непонятно, чего они ждут. То ли им нужна какая-то насадка, то ли ждут изменения климата. Это тазовые кости, так что возможно, что они служат чем-то вроде инкубационной камеры, но толком никто ничего не знает.

- А они ценятся?

- Очень, хотя точную цену я не знаю.

- Проверь по списку запрещенных грузов.

Алекс проверил. Никаких известных ограничений на импорт этих окаменелостей он не нашел,

- Кажется, это интереснее, чем еда.

- Это уж точно.

- Так берем..?

- Пожалуй...

Но не успел Алекс набрать на клавиатуре заказ торговому центру, как вспыхнул экран "Получено сообщение..."

- Рейф. - радостно сказал Алекс. Элиссии тоже понравилась перспектива повидать Рейфа Зеттера и поговорить с ним еще раз.

Но вовсе не старый космический торговец появился на экране, когда Алекс вышел на связь.

Изображение на экране явно принадлежало человеческому существу, а не гуманоиду. Но его лицо не поддавалось никакому описанию. Есть масса способов изуродовать человека до неузнаваемости и превратить его в подобие ночного кошмара, например пролететь слишком близко от звезды, слишком часто подвергаться воздействию межзвездного вакуума, поработать в некоторых рудниках... Но сейчас, глядя на бугристые серые опухоли, покрывающие плоть человеческого лица, Алекс не мог себе даже представить что произошло с его собеседником.

Губы, как обрывки паутины, затрепетали на серой плоти, скелетообразная скрюченная рука, сквозь которую просвечивали красные кровеносные сосуды, коснулась тонкой пряди волос, причудливо спадавшей с деформированной головы.

- Ты Райдер?

По крайней мере, голос был нормальным и притом мужским.

- Назовите себя.

Игнорируя вопрос, незнакомец продолжал. - Чем сегодня торгуешь? Минералы? Специи?

- А вам зачем?

- Не знаю твоих планов, но что бы ты ни делал, у меня есть для тебя предложение получше.

- Не стал бы с тобой торговать даже ради спасения от сверхновой.

- А Рейф Зеттер бы стал. Чего это ты такой нервный?

- Ты знаешь Рейфа? – удивился Алекс, озадаченный упоминанием знакомого имени.

- И я и пол Вселенной. - Изуродованный человек наклонился вперед. Черты лица заняли весь экран. Это паразиты.

- Не понял?!

- Я говорю, это паразиты, - незнакомец прикоснулся к лицу. - Личинки пауков. Я работал на Дикстре, где и пришелся по вкусу этим червякам. Здесь их пара миллионов. Период инкубации примерно десять лет - это и будет моим концом. Хотел бы я быть на торжественном обеде у своего недруга, когда меня разорвет, но так трудно все точно рассчитать! В общем, парень, я не виню тебя за то, что ты мне не доверяешь, но не надо судить только по внешности, Алекс. Ты ведь Алекс, не так ли? Я имею в виду, скажи ради бога, туда ли я попал?

- Да, я Алекс Райдер.

- А я Патрик Мак Гревви. Я хочу сказать тебе только две вещи. Первое: убей змею. Приложи это привидение, которое преследует меня целых пять лет. Я не пилот. Кто я, не имеет значения, но таких, как я, больше, чем подсолнечных семечек, которые ты перевез за свою жизнь. Мы ждем возмездия и не можем получить его сами. Убей змею и ты поможешь всем нам.

Алекс не смог сдержать усмешки на своем лице, хоть и понимал, что менее всего сейчас пристало улыбаться. Он чувствовал, что им манипулируют. Им управляют, как запрограммированным роботом, заставляя ходить по замкнутому бессмысленному кругу. Что происходит вокруг него, черт побери?! Еще три месяца назад его счет побед фиксировался только на имитаторе. Краска на его лицензии еще не высохла и, тем не менее, какие то силы выбрали именно его орудием мести не опасному, а смертоносному противнику.

За ним наблюдает много людей и все они затаив дыхание чего-то ждут. Почему он?! Почему именно он?! И как сюда встраивается Элиссия?

- Хорошо, - спокойно сказал он. - Я понял. Но ты сказал: "Две вещи".

- Именно. Скажи, это Рейф посоветовал тебе повернуть операцию с шанаскильским мехом, как только ты сможешь ее осилить? Я прав?

Он был прав, это был один из последних ценных советов Рейфа, и Алекс его не забыл.

Мак Гревви продолжал:

- Дав тебе такой совет, Рейф тем самым послал тебя ко мне. Тебе нужно подковать свою клячу. Ты должен взять на борт что-то действительно стоящее. Лети в Саут-Сити, найдешь центр частной торговли в Комплексе Магеллана.

- Я уже подковал свою клячу.

- Это ты так считаешь. Тем не менее, лети ко мне. Не упusti хороший шанс. Саут-Сити. Комплекс Магеллана...

Алекс колебался недолго. Он глянул на Эллисию. Пожав плечами, она кивнула, и Алекс принял предложение.

(Продолжение следует).





МКП "ИНФОРКОМ" 121019, Москва, Г-19, а/я 16

Вниманию читателей!

С этого номера мы начинаем публикацию адресов пунктов, где Вы можете приобрести наши материалы:

г. Воронеж. Студия компьютерных игр SAN-SAN. Магазин-салон "Электроника". Тел. 14-00-73.

г. Днепропетровск, ул. Шевченко, 34. Фирма "ЭКОС".

г. Нижний Новгород, ул. Горького, 146. Маг. "Фотолюбитель". ИМА "Ф-ПЛЮС". Тел. 35-07-18.

Вниманию дистрибуторов! Адреса наших оптовых покупателей публикуются бесплатно.

## Спектрум в школе

В этот раз мы предлагаем учебную программу по математике. Но прежде, чем переходить непосредственно к программе, остановимся вот на каком вопросе. Выполните: PRINT 10000000. На экране, как и следовало ожидать, будет напечатано: 10000000. Теперь выполните: PRINT 100000000. Сейчас число на экране выглядит иначе: 1E+8. Вы видите особенность вывода на экран числовых значений большой величины. Для большинства случаев этот вариант подходит, а может быть и более удобен. Но представьте себе, что Вы создаете программу по бухгалтерскому учету или по математике и надо, к примеру, точно знать результат сложения двух десятизначных чисел с точностью до последнего знака.

Попробуйте выполнить простую Бейсик-программу:

```
10 LET A=1000000001
20 LET B=1000000000
30 PRINT A;"-";B;"=";A-B
```

На экране Вы увидите:

1E+8-1E+8=1

А желательно было бы видеть:

1000000001-1000000000=1

Осуществимо ли это?

В качестве примера, показывающего, как практически достигается такой результат, мы приводим учебную программу по математике, которая будет заниматься разложением чисел на сомножители. В том случае, если число ни на что не делится, то значит оно является простым. Программа называется "PRIME" (что в переводе означает "простое число"). Ее можно использовать как для разложения чисел на сомножители, так и только для поиска простых чисел, для чего в предлагаемой программе предусмотрен специальный режим. Максимальная величина числа, для которого можно получить результат с точностью до последнего знака, равен 2 в степени 32, что составляет 4294967296.

Программа русифицирована по методике, приведенной в "ZX-PEBIO"-92, N1,2; стр. 29. Итак, текст программы:

```
1 GO TO 100
2 CLEAR 64599; LOAD "chr" CODE 64600
4 RUN
5 SAVE "PRIME" LINE 2: SAVE "chr" CODE 64600,768: STOP
8 POKE 23606,88: POKE 23607,251: RETURN
9 POKE 23606,0: POKE 23607,60: RETURN
```

```

20 PRINT #0; AT 0,0; " ПРОДОЛЖЕНИЕ - ЛЮБАЯ КЛАВИША ВВОД НОВОГО ЧИСЛА - [SPACE]"
22 PAUSE 0: INPUT ;
29 RETURN
40 LET H=INT(N/1E5)
42 IF NOT H THEN PRINT N;: RETURN
44 PRINT H;(STR$(N-1E5*(H-1)))(2 TO );
49 RETURN
100 BORDER 1: PAPER 7: INK 0: CLS
110 POKE 23658,8
120 GO SUB 8
130 PRINT AT 2,7;"М А Т Е М А Т И К А"
140 PRINT AT 5,4;"РАЗЛОЖЕНИЕ НА СОМНОЖИТЕЛИ"
200 PRINT AT 12,4;"[S] - ПОИСК СОМНОЖИТЕЛЕЙ"
210 PRINT AT 14,4;"[P] - ПОИСК ПРОСТЫХ ЧИСЕЛ"
220 PAUSE 0: IF INKEY$<>"S" AND INKEY$<>"P" THEN GO TO 220
300 LET I$=INKEY$: CLS
500 INPUT AT 0,0; "ВВЕДИТЕ ЧИСЛО <=4294967296 ";I: LET M=I
1000 FOR J=1 TO 10
1010 LET N=M
1020 GO SUB 2000
1030 IF M=I THEN GO SUB 20: IF INKEY$=" " THEN GO TO 500
1040 LET M=M+1
1050 NEXT J: PRINT
1060 GO SUB 20: IF INKEY$=" " THEN GO TO 500
1100 GO TO 1000
2000 RESTORE 2800: IF I$="S" THEN GO SUB 40: INK 1
2010 LFT F=0: LET D=0
2020 FOR B=0 TO 52
2030 READ Z: LET D=D+Z
2100 LET Q=N/D: IF Q<D THEN GO TO 2200
2110 IF I$="P" AND Q=INT Q THEN LET M=M+1: LET N=M: GO TO 2000
2120 IF I$="S" AND Q=INT Q THEN PRINT "=" AND NOT F;D;"*";: LET F=1: LET N=Q: GO TO 2100
2130 NEXT B
2140 RESTORE 2900: LET B=4: NEXT B
2200 IF I$="P" THEN GO SUB 40
2210 IF NOT F THEN PRINT INK 2;" ПРОСТОЕ ЧИСЛО";
2220 IF F THEN GO SUB 40
2300 INK 0: PRINT : RETURN
2800 DATA 2,1,2,2,4
2900 DATA 2,4,2,4,6,2,6,4,2,4,6,6,2,6,4,2,6,4,6,8,4,2,4,2,4,8,6,4,6,2,4,6,2,6,6,4,2,4,6,2,
6,4,2,4,2,10,2,10

```

Автостарт программы происходит со строки 2, где загружается символьный набор, после чего выполняется переход на начальную строку программы. Фактическое начало программы - строка 100. В строке 120 происходит включение режима CAPS LOCK, это позволяет упростить опрос клавиатуры при помощи INKEY\$.

После подготовительных операций на экран выводится главное меню программы, в котором два пункта:

```

"ПОИСК СОМНОЖИТЕЛЕЙ"
"ПОИСК ПРОСТЫХ ЧИСЕЛ"

```

Нажав "S" или "P", мы переходим на строку 300, где продолжается работа программы. В переменной I\$ запоминается выбранный режим. В строке 500 предлагается ввести исследуемое число. Анализ его начинается в строке 1000. Введенное значение числа присваивается переменной N, которая является входной величиной для подпрограммы GO SUB 2000, которая выполняет поиск сомножителей и вывод результата на экран. После этого программа делает паузу. Теперь, если нажать "SPACE", можно вернуться к запросу нового числа. Если нажать любую другую клавишу, то анализ чисел будет продолжен со следующего числа (большего на 1). То есть, можно как анализировать числа по одному, вводя их по запросу, так и анализировать массив идущих подряд чисел, введя значение начального числа. Во втором случае вывод результатов будет производиться блоками по 10 чисел (определяется переменной J в строке 1000). После вывода каждого блока результатов можно нажатием "SPACE" опять вернуться на запрос нового числа, если это нужно.

Перед вызовом подпрограммы 2000, значение числа М присваивается временной переменной N. Это необходимо делать, так как в результате работы подпрограммы 2000, значение ее входной переменной N будет испорчено.

Переменная F в подпрограмме 2000 является флагом, переключающимся из 0 в 1 в том случае, если найден хотя бы один сомножитель для данного числа, то есть если число не простое. Переменная 0 - это возможные сомножители. Весь ряд сомножителей получается путем увеличения очередного значения D на величину смещения, полученного из строк DATA. Действительно, если попробовать получать значения D при помощи формулы в строке 2030, то получим числа возможных сомножителей: 2,3,5,7,9 ... и так далее. Это ничто иное, как ряд простых чисел.

Строка 2100 проверяет, все ли возможные сомножители проверены. Если все, то переход на строку 2200 для вывода результата. Строки 2110 или 2120 (в зависимости от заданного режима) проверяют, делится ли исследуемое число на сомножитель без остатка. В режиме анализа всех чисел, в том случае, если делится без остатка, то значит найден один из сомножителей, происходит вывод его на экран и установка флага F (строка 2120) и далее выполняется продолжение анализа остатка от деления для поиска остальных сомножителей. В режиме поиска только простых чисел, зафиксированное деление без остатка говорит о том, что это число не простое, поэтому дальнейший анализ не нужен и выполняется переход к проверке следующего числа (строка 2110).

Строка 2200 выполняет печать исследуемого числа и всех его сомножителей, если они есть или печать сообщения о том, что число простое. Собственно печать чисел выполняется подпрограммой GO SUB 40. Здесь, если число меньше, чем 100000, то происходит обычная его печать, а если больше, то печать выполняется в два приема. Сначала печатается то, что выходит за пределы 100000 (переменная N, строка 44), затем, для того, чтобы напечатать младшие разряды, происходит преобразование значения числа в строку. Поясним происходящее на примере. Пусть нам надо напечатать число 4000000027. Сначала берутся старшие разряды числа: N=40000, теперь вычисляется "остаток", увеличенный на 100000, и происходит печать его за старшими разрядами:

|          |            |                                                                        |
|----------|------------|------------------------------------------------------------------------|
| надо:    | 4000000027 |                                                                        |
| печать1: | 40000      | <- число                                                               |
| печать2: | 100027     | <- строка                                                              |
|          | ↑          | эта единица не выводится, так как полученный строка печатается (2 TO). |

В заключение отметим, что поскольку программа производит многократные вычисления при поиске сомножителей, и расчет выполняется в Бейсике, то получение результата для больших чисел может занять достаточно продолжительное время. Представьте себе, сколько сомножителей надо перебрать для того, чтобы убедиться в том, что десятизначное число является простым! Например, для числа 4294987291 - требуется 8 минут! Поэтому в качестве усовершенствования этой программы мы можем посоветовать читателям попробовать скомпилировать подпрограмму со строки 2000 для ускорения выполнения вычислений.

# Маленькие хитрости

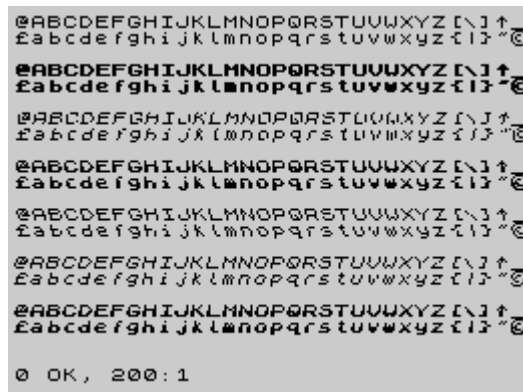
## Стилизованные шрифты

Вы, наверное, видели, как во многих программах применяются «стилизированные» шрифты. Утолщённые, наклонные и другие разновидности придают программам особую привлекательность. Если для этой цели применять загружаемый символьный набор, который занимает почти килобайт памяти, то требуется соответствующее время при загрузке программы, которая порой меньше, чем сам символьный набор. Поэтому для получения оригинальных шрифтов можно воспользоваться несложными процедурами в машинных кодах. Правда, выигрыш во времени загрузки получается только в том случае, если используется символьный набор ПЗУ, поскольку русский шрифт всё равно приходится загружать. Однако в обоих случаях появляется возможность разнообразить программу, используя один базовый символьный набор и оригинально модифицируя его. Например, в одном режиме работы программы вывод осуществляется утолщённым шрифтом, в другом - наклонным, в третьем - готическим и т.д. Для этого не надо загружать такое число символьных наборов, а все их можно получить из основного - базового.

В качестве примера, поясняющего вышесказанное, приводим простую Бейсик-программу, построенную на использовании машиннокодовых процедур, в основе которых лежат операции с битами. Преобразования выполняются с символьным набором ПЗУ, но Вы можете использовать изложенные принципы и для модифицированных загружаемых символьных наборов.

```
4 GO TO 100
8 POKE 23606,0: POKE 23607,117: RETURN
9 POKE 23606,0: POKE 23607,60: RETURN
40 FOR n=64 TO 127: PRINT CHR$ n;: NEXT n: PRINT '': RETURN
50 RESTORE 50: FOR n=30000 TO 30029: READ a: POKE n,a: NEXT n
55 DATA 33,0,61,17,0,118,1,0,3,237,176,33,0,118,17,0,3,126,79,203,63,177,119,35,27,122,179,
    32,244,201
59 GO TO 80
60 RESTORE 60: FOR n=30000 TO 30041: READ a: POKE n,a: NEXT n
65 DATA 33,0,61,17,0,118,1,0,3,237,176,33,0,118,14,96,6,2,126,203,63,203,63,119,35,16,247,6,
    4,126,203,63,119,35,16,249,35,35,13,32,231,201
69 GO TO 80
70 RESTORE 70: FOR n=30000 TO 30035: READ a: POKE n,a: NEXT n
75 DATA 33,0,61,17,0,118,1,0,3,237,176,33,0,118,6,96,197,35,35,35,35,6,4,126,79,203,63,177,
    119,35,16,247,193,16,237,201
80 RANDOMIZE USR 30000: GO TO 40
100 BORDER 7: PAPER 7: INK 0: CLEAR 29999: PRINT AT 1,0;
110 GO SUB 9: GO SUB 40: GO SUB 8
120 GO SUB 50
130 GO SUB 60
140 GO SUB 70
150 POKE 30027,0: PAUSE 20: GO SUB 80
160 POKE 30026,39: PAUSE 20: GO SUB 80
170 POKE 30027,177: PAUSE 20: GO SUB 80
200 GO SUB 9
```

После набора запустите программу командой RUN. Вы увидите на экране различные символьные наборы. (См. рисунок).



Подпрограммы со строк 50, 60, 70 - формируют в памяти с адреса 30000 (7503h) и выполняют процедуры в машинных кодах, которые модифицируют символьный набор. После этого происходит вывод на экран фрагмента полученного символьного набора при помощи подпрограммы со стоки 40.

Переключение символьных наборов происходит при помощи подпрограмм GO SUB 8 или GO SUB 9. Первая включает полученный символьный набор, а вторая - символьный набор ПЗУ.

Рассмотрим подробно, что же происходит при работе блоков в машинных кодах с символьным набором.

Листинг 1

Подпрограмма GO SUB 60 создаёт утолщённый символьный набор:

|      |          |                  |                                            |
|------|----------|------------------|--------------------------------------------|
| 7530 | 21 00 3D | LD HL, #3D00     | ; Переброска базового                      |
| 7533 | 11 00 76 | LD DE, #7600     | ; символьного набора                       |
| 7536 | 01 00 03 | LD BC, #0300     | ; в адрес 7600h (30208)                    |
| 7539 | ED B0    | LDIR             | ; для модификации.                         |
| 753B | 21 00 76 | LD HL, #7600     | ; Начиная сначала, каждый бит символьного  |
| 753E | 11 00 03 | LD DE, #0300     | ; набора, т.е. каждая линия символа        |
| 7541 | 7E       | LOOP1 LD A, (HL) | ; загружается в аккумулятор,               |
| 7542 | 4F       | LD C, A          | ; сохраняется в регистре C, затем          |
| 7543 | CB 3F    | SRL A            | ; сдвигается вправо на один пиксел         |
| 7545 | B1       | OR C             | ; и объединяется по «ИЛИ» с прежним        |
|      |          |                  | ; значением, т.е. включенные пикселы так   |
|      |          |                  | ; и останутся включенными, но включатся    |
|      |          |                  | ; ещё и те, которые получены путём сдвига. |
| 7546 | 77       | LD (HL), A       | ; Полученное значение записывается         |
|      |          |                  | ; обратно в символьный набор.              |
| 7547 | 23       | INC HL           | ; Переход к следующей ячейке, т.е. к       |
|      |          |                  | ; следующей линии символьного набора.      |
| 7548 | 1B       | DEC DE           |                                            |
| 7549 | 7A       | LD A, D          | ; Проверка на достижение конца символьного |
| 754A | B3       | OR E             | ; набора, т.е. нуля в счётчике DE          |
| 754B | 20 F4    | JR NZ, LOOP1     | ; Если нет, то повторение процедуры        |
| 754D | C9       | RET              | ; Если да, то возврат в Бейсик             |

Листинг 2

При помощи подпрограммы GO SUB 70 получается шрифт, напоминающий готический:

|      |          |                  |                                          |
|------|----------|------------------|------------------------------------------|
| 7530 | 21 00 3D | LD HL, #3D00     | ; Переброска                             |
| 7533 | 11 00 76 | LD DE, #7600     | ; символьного набора                     |
| 7536 | 01 00 03 | LD BC, #0300     | ; в адрес 7600h (30208)                  |
| 7539 | ED B0    | LDIR             | ; для модификации.                       |
| 753B | 21 00 76 | LD HL, #7600     | ; Начиная сначала, для всех              |
| 753E | 0E 60    | LD C, #60        | ; символов символьного набора            |
| 7540 | 06 02    | LOOP1 LD B, #02  | ; берутся две верхних линии (два байта). |
| 7542 | 7E       | LOOP2 LD A, (HL) | ; Байт записывается в аккумулятор,       |
| 7543 | CB 3F    | SRL A            | ; сдвигается вправо                      |
| 7545 | CB 3F    | SRL A            | ; на два пиксела                         |

|      |       |                  |                                            |
|------|-------|------------------|--------------------------------------------|
| 7547 | 77    | LD (HL), A       | ; и записывается назад в память.           |
| 7548 | 23    | INC HL           | ; Переход к адресу следующего байта.       |
| 7549 | 10 F7 | DJNZ LOOP2       | ; Повторение для второго байта.            |
| 754B | 06 04 | LD B, #04        | ; Берутся четыре следующих линии.          |
| 754D | 7E    | LOOP3 LD A, (HL) | ; Байт записывается в аккумулятор,         |
| 754E | CB 3F | SRL A            | ; двигается вправо на один пиксел          |
| 7550 | 77    | LD (HL), A       | ; и записывается назад в память.           |
| 7551 | 23    | INC HL           | ; Переход к адресу следующего байта.       |
| 7552 | 10 F9 | DJNZ LOOP3       | ; Повторение для всех четырёх линий.       |
| 7554 | 23    | INC HL           | ; Оставшиеся две линии из                  |
| 7555 | 23    | INC HL           | ; восьми не изменяются.                    |
| 7556 | 0D    | DEC C            | ; Счётчик символов уменьшается на 1.       |
| 7557 | 20 E7 | JR NZ, LOOP1     | ; Повторение цикла для следующего символа. |
| 7559 | C9    | RET              | ; Возврат в Бейсик.                        |

### Листинг 3

Оригинальный шрифт получается, если утолщённой сделать только нижнюю часть символа, оставив верхнюю без изменения:

|      |          |                  |                                          |
|------|----------|------------------|------------------------------------------|
| 7530 | 21 00 3D | LD HL, #3D00     | ; Переброска                             |
| 7533 | 11 00 76 | LD DE, #7600     | ; символьного набора                     |
| 7536 | 01 00 03 | LD BC, #0300     | ; базового в адрес 7600h (30208)         |
| 7539 | ED B0    | LDIR             | ; для модификации.                       |
| 753B | 21 00 76 | LD HL, #7600     | ; Начиная сначала                        |
| 753E | 06 60    | LD B, #60        | ; для всех символов символьного набора   |
| 7540 | C5       | LOOP1 PUSH BC    | ; запоминание счётчика цикла на стеке,   |
|      |          |                  | ; так как регистр участвует ещё          |
|      |          |                  | ; в одном, внутреннем цикле.             |
| 7541 | 23       | INC HL           | ; Четыре верхние                         |
| 7542 | 23       | INC HL           | ; линии символа                          |
| 7543 | 23       | INC HL           | ; оставляем                              |
| 7544 | 23       | INC HL           | ; без изменений.                         |
| 7545 | 06 04    | LD B, #04        | ; Цикл для четырёх оставшихся линий.     |
| 7547 | 7E       | LOOP2 LD A, (HL) | ; Байт заносится в аккумулятор,          |
| 7548 | 4F       | LD C, A          | ; сохраняется в регистре C,              |
| 7549 | CB 3F    | SRL A            | ; сдвигается вправо на один пиксел       |
| 754B | B1       | OR C             | ; и объединяется с прежним значением     |
|      |          |                  | ; (это выполняется аналогично            |
|      |          |                  | ; предыдущей программе).                 |
| 754C | 77       | LD (HL), A       | ; Полученное значение заносится в память |
| 754D | 23       | INC HL           | ; и переход к адресу следующего байта.   |
| 754E | 10 F7    | DJNZ LOOP2       | ; Повторение цикла для 4-х нижних линий. |
| 7550 | C1       | POP BC           | ; Повторение цикла для всех остальных    |
| 7551 | 10 ED    | DJNZ LOOP1       | ; символов символьного набора.           |
| 7553 | C9       | RET              | ; Возврат в Бейсик.                      |

Полученный при помощи программы 3 символьный набор, за счёт смещения нижней части вправо, получается как бы наклоненным влево. Некоторая модификация этой программы, а именно, замена команды OR C в ячейке 754Bh, командой NOP, позволяет получить только наклон символов, без расширения нижней части. Это выполняется в строке 150 Бейсик-программы. Наклон можно получить и в правую сторону. Для этого нужно использовать в ячейке 7549h вместо сдвига вправо (SRL) сдвиг влево (SLA), что выполняется в строке 160. Наклон вправо с расширенной нижней частью получается, если восстановить команду OR C в ячейке 754Bh, сохранив команду сдвига влево в ячейке 7549h. Это делается в строке 170.

Этим, конечно, не исчерпываются возможности по модификации символьного набора. Мы только показали, как практически можно достичь простейших эффектов. Уверены, что читатели разовьют эту тему и придумают новые оригинальные варианты такой модификации.

# ПРИМЕНЕНИЕ АССЕМБЛЕРА ДЛЯ СОЗДАНИЯ БЫСТРОРАБОТАЮЩИХ ПРОГРАММ

Продолжение. (Начало см. в "ZX-РЕВЮ-93" N 1,2)

Заканчивая рассмотрение вопросов выдачи информации на экран, мы должны еще обсудить возможность печати в нижней части экрана, т.е. в строках с номерами 22 и 23.

Для этого применяют те же способы, что были описаны выше для вывода на главную часть экрана. Отличие состоит лишь в том, что для вывода в нижнюю часть экрана необходимо открывать другой канал, прежде чем применять команду печати RST 16. Как это сделать, показано в демонстрационной программе 1.8.

Здесь для вывода в нижнюю часть экрана сначала открывается канал с номером "минус 3" (253). Чтобы информационное сообщение "O.O.K.", появляющееся после исполнения программы не переместило выведенную нами на экран строку, необходимо иметь эквивалент команды PAUSE 0 в машинных кодах.

Останов программы можно реализовать, если использовать команду ассемблера HALT, которая приостанавливает работу процессора до очередного прерывания (до очередного сканирования клавиатуры, выполняемого процедурой ПЗУ KEYSCAN). Для реализации команды, аналогичной PAUSE 0, необходимо после команды HALT выполнить проверку состояния 5-го бита системной переменной FLAGS, который включается при нажатии произвольной клавиши. Если этот бит сброшен, то выполняется возврат к команде HALT, в противном случае программа выполняется дальше. Обратите внимание на то, что здесь 22-я строка считается нулевой, а 23-я - первой.

При выводе данных на экран весьма полезной для Вас может оказаться одна процедура из ПЗУ, находящаяся там по адресу 3652. Она позволяет уладить заданное число строк с экрана, причем отсчет строк начинается с 23-й строки, т.е. с ее помощью Вы можете как стирать информацию в системном окне, так и на основном экране. Пользоваться этой процедурой придется, по-видимому очень часто.

Перед вызовом процедуры необходимо записать в регистр В число удаляемых строк. Атрибуты экрана остаются такими, какие записаны в системной переменной ATTR-P. Так, например, нижние две строки экрана могут быть очищены с помощью следующей короткой программы:

```
LD B, 2
CALL 3652
RET
```

Листинг 1.8

| АДРЕС | МАШ. КОД    | АССЕМБЛЕР     | КОММЕНТАРИЙ                                                           |
|-------|-------------|---------------|-----------------------------------------------------------------------|
|       |             | ORG 23760     |                                                                       |
| 23760 | 3E FD       | LD A, 253     | ; Подготовка к открыванию канала.                                     |
| 23762 | CD 01 16    | CALL 5633     | ; Открываем канал "системного<br>; окна компьютера"                   |
| 23765 | 11 EA 5C    | LD DE, DATA   | ; Адрес начала выводимых<br>; данных.                                 |
| 23768 | 01 1E 00    | LD BC, 30     | ; Длина данных с учетом<br>; управляющих кодов.                       |
| 23771 | CD 3C 20    | CALL 8252     | ; Вызов процедуры ПЗУ для<br>; печати сообщения.                      |
| WAIT  |             |               | ; Начало процедуры "задерж-<br>; ки", аналогичной PAUSE 0.            |
| 23774 | 76          | HALT          | ; Прекращение работы процессора<br>; до прихода системного прерывания |
| 23775 | FD CB 01 6E | BIT 5, (IY+1) | ; Проверка пятого бита сис-<br>; темной переменной FLAGS.             |

|       |                |               |                                                                                 |
|-------|----------------|---------------|---------------------------------------------------------------------------------|
| 23779 | 28 F9          | JR Z, WAIT    | ; Если он выключен - клавиша<br>; не нажималась, то возврат<br>; на метку WAIT. |
| 23781 | FD CB 01 AE    | RES 5, (IY+1) | ; Если он включен, то вы-<br>;ключаем его и продолжаем работу.                  |
| 23785 | C9             | RET           | ; Выход из процедуры.                                                           |
| DATA  | 22 00 10       |               | ; Аналог "AT 0, 10"                                                             |
|       | 73 78 80 85 84 |               | ; текстовое сообщение                                                           |
|       | 32 76 73 78 69 |               | ; "INPUT LINE 0", записан-                                                      |
|       | 32 48          |               | ; ное в кодах ASCII.                                                            |
|       | 22 01 10       |               | ; Аналог "AT 1, 10"                                                             |
|       | 73 78 80 85 84 |               | ; Текстовое сообщение                                                           |
|       | 32 76 73 78 69 |               | ; "INPUT LINE 1", записан-                                                      |
|       | 32 49          |               | ; ное в кодах ASCII.                                                            |

## 2. Команды PLOT, DRAW и CIRCLE

### PLOT

Команда PLOT x,y позволяет поставить на экране точку (включить пиксел), координата которой определяются значениями x и y, а цвет зависит от установленного значения INK. Координата x определяет номер столбца, а координата y - номер строки той точки экрана, где будет включен пиксел.

Пиксел, координаты которого равны 0,0, находится в нижнем левом углу экрана, а пиксел с координатами 175,255 - в верхнем правом углу экрана.

Для этой цели в ПЗУ компьютера имеется процедура PLOT, у которой есть две возможные точки входа.

Первая точка входа имеет адрес 8933 DEC (22E5 HEX). Перед вызовом этой процедуры по команде CALL 8933, необходимо в регистр B записать значение координаты y (0...175), а в регистр C - значение координаты x (0...255). Пример записи показан в программе 2.1.

#### Листинг 2.1

| АДРЕС | МАШ. КОД | АССЕМБЛЕР | КОММЕНТАРИЙ               |
|-------|----------|-----------|---------------------------|
|       |          | ORG 23760 |                           |
| 23760 | 08 7D    | LD B, 125 | ; Координата y=125.       |
| 23762 | 0E 4B    | LD C, 75  | ; Координата x=75.        |
| 23764 | CD E5 22 | CALL 8933 | ; Печать точки (125, 75). |
| 23767 | C9       | RET       | ; Возврат.                |

Использование десятиричных чисел вместо шестнадцатиричных мы применяем только для того, чтобы начинающие читатели быстрее адаптировались к сути работы в машинном коде. Они более наглядны, если применяется раздельная загрузка регистров B и C. Однако, если записывать координаты сразу же в регистровую пару BC, то лучше все же использовать шестнадцатиричные числа - при такой записи проще представить положение пиксела на экране.

Вторая точка входа имеет адрес 8924 DEC (22DC HEX). При этом значения координат x и y должны быть предварительно записаны не в регистровую пару BC, а на стек так называемого встроенного калькулятора, причем значение y должно находиться на вершине стека, а значение x - под ним.

Начинающий программистам надо сказать и несколько слов о стеке калькулятора.

Для работы с действительными числами (числами с десятичной точкой) целочисленных регистров процессора явно недостаточно. Более мощные машины применяют для этого математический сопроцессор. В "Спектруме" же реализован другой, более удобный и дешевый подход - в ПЗУ внедрена обширная программа-калькулятор, имеющая свою оригинальную систему команд. Когда Вы работаете в БЕЙСИКе, калькулятор работает автоматически и для Вас все происходит незаметно. При работе в машинном коде



приходится программировать его вручную.

Основным местом для хранения исходных данных и результатов расчета калькулятора является его стек, хотя кроме этого калькулятор имеет еще и несколько ячеек памяти. Подробно с использованием калькулятора Вы можете познакомиться в нашей книге "Программирование в машинных кодах. Первые шаги. Практикум. Справочник."

Остается, правда, пока открытым вопрос, а как же записать координаты своей точки на стек калькулятора? Очень просто, для этой цели тоже используется процедура ПЗУ компьютера? Она находится по адресу 11560 DEC (2D28 HEX). Вам нужно поместить желаемое число в регистр А процессора и вызвать эту процедуру. Число будет передано на стек, чтобы отправить туда два числа, операцию надо сделать дважды. Естественно, что то число, которое будет передано последним и будет размещено на вершине стека.

На первый взгляд кажется, что работа с первой точкой входа намного удобнее, т.к. не надо ничего перебрасывать через стек калькулятора. Но это только на первый взгляд. На самом же деле в программах очень редко приходится что-то печатать в заранее известных координатах. Почти всегда программа сама и рассчитывает эти координаты. Делается это с помощью встроенного калькулятора и потому взять результаты расчета со стека и использовать их напрямую оказывается весьма удобно.

#### Листинг 2.2

| АДРЕС | МАШ. КОД | АССЕМБЛЕР  | КОММЕНТАРИЙ                 |
|-------|----------|------------|-----------------------------|
|       |          | ORG 23760  |                             |
| 23760 | 3E 4B    | LD A, 75   | ; Координата x=75.          |
| 23762 | CD 28 2D | CALL 11560 | ; Поместили x на стек каль- |
|       |          |            | ; кулятора.                 |
| 23765 | 3E 7D    | LD A, 125  | ; Координата y=125          |
| 23767 | CD 28 2D | CALL 11560 | ; Поместили ее на стек.     |
| 23770 | CD DC 22 | CALL 8924  | ; Печать точки (125,75)     |
| 23773 | C9       | RET        | ; Возврат.                  |

#### Листинг 2.3

| АДРЕС | МАШ. КОД | АССЕМБЛЕР     | КОММЕНТАРИЙ                  |
|-------|----------|---------------|------------------------------|
|       |          | ORG 23760     |                              |
| 23760 | 3E 03    | LD A, 3       | ;                            |
| 23762 | FD 77 57 | LD (IY+87), A | ; Включение двух младших     |
|       |          |               | ; битов сист. перем. P_FLAG  |
| 23765 | 06 28    | LD B, 40      | ; Координата y=40.           |
| 23767 | 0E 19    | LD C, 25      | ; Координата x=25            |
| 23769 | CD E5 22 | CALL 8933     | ; Печать точки (40,25) в ре- |
|       |          |               | ; жиме OVER 1.               |
| 23772 | AF       | XOR A         | ; Очистка регистра A.        |
| 23773 | FD 77 57 | LD (IY+87), A | ; Восстановление P_FLAG.     |
| 23776 | C9       | RET           | ; Возврат.                   |

#### Листинг 2.4

| АДРЕС | МАШ. КОД    | АССЕМБЛЕР      | КОММЕНТАРИЙ                 |
|-------|-------------|----------------|-----------------------------|
|       |             | ORG 23760      |                             |
| 23760 | 2E 7F       | LD L, 64       | ; Номер символа "@"=64 DEC. |
| 23762 | 26 00       | LD H, 00       | ; Теперь в паре HL число 64 |
| 23764 | 29          | ADD HL, HL     | ; Умножили его на 2.        |
| 23765 | 29          | ADD HL, HL     | ; Умножили его на 4.        |
| 23766 | 29          | ADD HL, HL     | ; умножили его на 8.        |
| 23767 | ED 5B 36 5C | LD DE, (23606) | ; Загрузили в DE адрес, на  |
|       |             |                | ; который указывает систем- |
|       |             |                | ; ная переменная CHARS.     |
| 23771 | 19          | ADD HL, DE     | ; Прибавили к нему "смеще-  |
|       |             |                | ; ние" нашего символа от    |
|       |             |                | ; начала символьного набо-  |
|       |             |                | ; ра, ранее вычисленное в   |
|       |             |                | ; регистровой паре HL.      |
| 23772 | 06 08       | LD B, 8        | ; Счетчик строк в шабло-    |
|       |             |                | ; не символа, равный 8.     |

|       |             |                |                                 |
|-------|-------------|----------------|---------------------------------|
| 23774 | E5          | PUSH HL        | ; Запомнили адрес начала        |
| 23775 | 0E 08       | LD C, B        | ; шаблона на машинном стеке     |
| 23776 | 7E          | LD A, (HL)     | ; Счетчик столбцов в шабло-     |
| 23778 | C5          | PUSH BC        | ; не символа, равный 8.         |
| 23779 | ED 4B 0E 5D | LD BC, (COORD) | ; Принимаем в аккумулятор       |
| 23783 | 17          | RLA            | ; байт шаблона символа.         |
| 23784 | F5          | PUSH AF        | ; Запомнили содержимое BC,      |
| 23785 | 30 05       | JR NC, 23792   | ; чтобы освободить BC для       |
| 23787 | C5          | PUSH BC        | ; других дел.                   |
| 23788 | CD E5 22    | CALL 8933      | ; Ввод в BC координат пози-     |
| 23791 | C1          | POP BC         | ; ции печати.                   |
| 23792 | 0C          | INC C          | ; Ротация регистра A влево.     |
| 23793 | ED 43 0E 5D | LD (COORD), BC | ; Содержимое старшего бита      |
| 23797 | F1          | POP AF         | ; при этом переходит во         |
| 23796 | C1          | POP BC         | ; флаг CARRY регистра F.        |
| 23799 | 0D          | DEC C          | ; Запомнили регистры A и F      |
| 23800 | 20 E8       | JR NZ, 23778   | ; на машинном стеке.            |
| 23802 | C5          | PUSH BC        | ; Если флаг CARRY выключен,     |
| 23803 | ED 4B 0E 5D | LD BC, (COORD) | ; то точку печатать не надо.    |
| 23807 | 05          | DEC B          | ; Делаем обход печати.          |
| 23808 | 3E F8       | LD A, 248      | ; Перед печатью точки запо-     |
| 23810 | 81          | ADD A, C       | ; минаем координаты.            |
| 23811 | 4F          | LD C, A        | ; Вызов процедуры PLOT.         |
| 23812 | ED 43 0E 5D | LD (COORD), BC | ; Восстановили координаты.      |
| 23816 | C1          | POP BC         | ; Переход к очередному          |
| 23817 | E1          | POP HL         | ; столбцу символа.              |
| 23818 | 23          | INC HL         | ; Запомнили новую позицию       |
| 23819 | 10 D1       | DJNZ 23774     | ; печати.                       |
| 23821 | C9 RET      |                | ; Восстановление пары AF.       |
| COORD |             |                | ; Восстановление счетчиков в BC |
| 23822 |             | DEFB 100       | ; Уменьшение счетчика столбцов. |
| 23823 |             | DEFB 20        | ; Если он еще не обнулится      |

Программа 2.2 показывает использование процедуры PLOT со второй точкой входа 8924 DEC (22DC HEX). Процедура PLOT снимет два верхних числа со стека калькулятора и использует их как координаты для включения пиксела на экране. Числа на стеке при этом не сохраняются.

Использование калькулятора будет еще обсуждаться более подробно при рассмотрении программы 2.8. А пока рассмотрим программу 2.3, в которой показано, как в машинных кодах реализовать команду

PLOT OVER 1; x, y.

В этой программе перед вызовом процедуры PLOT необходимо включить нулевой и первый биты системной переменной P-FLAG (IY+87). А затем, после вызова процедуры PLOT, восстановить прежнее значение этих битов. Если этого не сделать, то и все последующие точки и символы будут выводиться на экран в режиме OVER 1.

Включение режима INVERSE аналогично включению режима OVER. Отличие состоит лишь в том, что теперь необходимо включить 2-ой и 3-ий биты системной переменной P-FLAG, т.е. надо записать в эту переменную число 12.

Используя точку входа 8933, мы можем написать программу для построения символа в любом месте экрана по точкам. При этом код ASCII символа должен быть в диапазоне 32...127, а шаблон символа (его конструкция) должен быть задан в участке символьного набора (см. программу 2.4).

Чтобы определить, где хранятся конструкции (шаблоны) символов, служит системная переменная CHARS (23606 DEC = 5C36 HEX). Тот адрес, который в ней хранится, указывает на 256 байтов ниже, чем начало символьного набора.

В программе 2.4 код выводимого на экран символа записывается в регистр L. Затем это значение умножается на 8 (т.к. в символьном наборе на конструкцию каждого символа использовано по 8 байтов) и результат суммируется с числом, которое хранится в системной переменной CHARS. Таким образом определяется начальный адрес восьми байтов, описывающих форму нужного вам символа.

Вы знаете, что первые 32 символа (с 0 по 31) являются непечатными, поэтому очень удобно, что CHARS указывает на 256 байтов ниже, чем начало набора, т.к. благодаря этому они-то как раз и оказываются пропущенными (32\*8=256).

Теперь регистровая пара HL используется как указатель адреса очередного байта, а регистровая пара BC - как счетчик элементов матрицы 8x8 битов, описывающей форму символа. Здесь в B хранится номер текущего ряда, а в C - номер столбца. Регистр A используется для манипуляций с текущим байтом.

## Листинг 2.5

| АДРЕС | МАШ. КОД    | АССЕМБЛЕР      | КОММЕНТАРИЙ                 |
|-------|-------------|----------------|-----------------------------|
|       |             | ORG 23760      |                             |
| 23760 | 2E 7F       | LD L, 87       | ; Номер символа "W"=87 DEC  |
| 23762 | .....       | .....          | ; Строки 23762... 23791     |
|       | .....       | .....          | ; этой программы соответ-   |
| 23791 | .....       | .....          | ; ствуют программе 2/4.     |
| 23792 | 04          | INC B          | ; Переход к очередной стро- |
|       |             |                | ; ке экрана.                |
| 23793 | ED 43 0E 5D | LD(COORD), 0BC | ; Запомнили позицию печати  |
| 23797 | F1          | POP AF         | ; Восстановление пары AF.   |
| 23798 | C1          | POP BC         | ; Восстановление счетчиков. |
| 23799 | 0D          | DEC C          | ; Уменьшение счетчика       |
|       |             |                | ; столбцов.                 |
| 23800 | 20 E8       | JR NZ, 23778   | ; Если он еще не обнулялся, |
|       |             |                | ; возвращаемся для печати   |
|       |             |                | ; соседней точки в ряду.    |
| 23802 | C5          | PUSH BC        | ; Переходя к очередному ря- |
|       |             |                | ; ду, запомнили счетчики.   |
| 33803 | ED 4B 0E 5D | LD BC, (COORD) | ; Возьмем координаты.       |
| 23807 | 05          | DEC B          | ; Уменьшим номер ряда.      |
| 23808 | 3E F8       | LD A, 248      | ; Уменьшение координаты     |
| 23810 | 80          | ADD A, B       | ; у на 8.                   |
| 23811 | 47          | LD B, A        | ;                           |
| 23812 | ED 43 0E 5D | LD (COORD), BC | ; Запомнили координату.     |
| 23816 | C1          | POP BC         | ; Восстановили счетчики ря- |
|       |             |                | ; дов и столбцов.           |
| 23817 | E1          | POP HL         | ; Восстановление указателя. |

|       |       |            |                                                                                             |
|-------|-------|------------|---------------------------------------------------------------------------------------------|
| 23818 | 23    | INC HL     | ; Очередной ряд.                                                                            |
| 23819 | 10 D1 | DJNZ 23774 | ; Уменьшаем на единицу счетчик рядов в В и если он еще не обнулится, то возвращаемся назад. |
| 23821 | C9    | RET        | ; Выход                                                                                     |
| COORD |       |            |                                                                                             |
| 23822 |       | DEFB 100   | ; Координата y=100                                                                          |
| 23823 |       | DEFB 20    | ; Координата x=20                                                                           |

Так, например, одной из манипуляций является ротация байта влево командой RLA, которая производит циклическое смещение всех битов байта влево и перенос седьмого бита во флаг CARRY. Если этот бит равен единице, то флаг CARRY включается, следовательно будет включен и соответствующий пиксел на экране, если же бит равен 0, то не включатся ни флаг, ни пиксел. Команда RLA выполняется по 8 раз для каждого из 8 байтов, описывающих шаблон символа.

После каждой команды RLA координаты очередного пиксела изменяются. Начальное значение этих координат соответствует верхнему левому углу матрицы 8\*8 битов, хранящей форму символа. Понятно, что при этом начальное значение координаты y должно быть в диапазоне 7...175, а значение координаты x может быть в диапазоне 0...255. Если же это значение окажется больше, чем 255, то это равносильно тому, что значение равно 0, так как если C=255 и выполняется команда INC C, то результат будет C = 0.

Эта программа может быть легко модифицирована для вывода на экран символа, развернутого на 90 градусов против часовой стрелки (см. программу 2. 5).

В этой программе начальные значения координат x,y соответствуют нижнему левому углу матрицы, хранящей форму символа и исследование байта идет не по горизонтали, а по вертикали.

С этого момента мы можем прийти к очень интересным решениям. Дело в том, что одной из первых проблем, встающих перед начинающим пользователем "Спектрума" является печать нестандартными шрифтами.

Обычно каждый спрашивает "а почему компьютер печатает только символами 8X8?" Почему нельзя печатать символами 5X8, 6X8 и даже 3X6, как например в программе "THE LAST WORD 2". Почему нельзя печатать символами 17X18? Может быть для тех, кто пишет программу на японском языке это было бы очень удобно.

Оказывается, если Вы печатаете из БЕЙСИКа или из машинного кода командой RST 16 или процедурой ПЗУ, которая опирается на RST 16, Вы действительно ограничены системным требованием размера 8x8. Так уж заложено в системных процедурах ПЗУ. Но если Вы выполняете графическую печать, то есть как бы не печатаете свои символы, а рисуете их по точкам, то можете обойти эти процедуры и создать свои, которые позволят вам делать все, что захотите.

## Листинг 2.6

```

5 OVER 0: INK 0: PAPER 6: BORDER 3: CLS
10 LET x=24: LET y=79: LET h=5: LET w =2: LET a$="STOP THE TAPE"
15 INK 2: RANDOMIZE USR 32393: PAUSE 100
20 LET w=8: LET y=150: LET x=0: LET a$="PLOT"
30 INK 1: RANDOMIZE USR 32393
35 LET a$ ="   ": LET y=140: INK 1: RANDOMIZE USR 32393
250 PRINT AT 8,0:"Высота 1-22?   ", "Ширина 1-32?   ", "x-коорд. 0-255?   ", "Y-коорд. 0-195?   ", "СООБЩЕНИЕ?", "   INK 0-9?",
255 LET z$=" "
257 LET v$=" "
260 PRINT OVER 1; AT 8,0; z$
262 INPUT h: PRINT AT 8,14; h;" ":OVER 1; AT 8,0; v$
265 PRINT OVER 1; AT 9,0; z$: INPUT w; PRINT AT 9,14; w: " "; OVER 1; AT 9,0; v$
270 PRINT OVER 1; AT 10,0;z$: INPUT x: PRINT AT 10,17;x;" "; OVER 1; AT 10,0;v$
272 PRINT OVER 1; AT 11,0;z$: INPUT y: PRINT AT 11,17; y:" "; OVER 1: AT 11,0;v$

```

```

273 PRINT OVER 1; AT 12,0;z$; AT 13,0; z$: INPUT a$: PRINT AT 13,0: a$; : FOR k=LEN a$ TO
    31: PRINT " ";: NEXT k: PRINT OVER 1; AT 12,0; v$; OVER 1: AT 13,0; v$
275 PRINT OVER 1; AT 14,0; z$: INPUT 1: PRINT AT 14,0; INK 1; OVER 1; v$
280 PAUSE 50: CLS: INK 1: RANDOMIZE USR 32393: INK 0
300 PRINT #0; AT 0,0 "Q - конец работы"; AT 1,0; "Прочие клавиши - повтор"
310 PAUSE 0: IF INKEY$="Q" OR INKEY$ = "q" THEN STOP
320 CLS: GO TO 250

```

Освоив способы вывода по точкам произвольного символа, мы можем перейти к нашей первой серьезной программе, позволяющей выводить по точкам на экран в произвольном месте целое сообщение, высота и ширина символов которого может быть выбрана нами.

Эта программа состоит из двух блоков. Первый блок - настроечный, он выполнен на БЕЙСИКе, т.к. от него не требуется выстродействие. второй блок - основной, он выполнен в машинных кодах. Связь между блоками нужна для передачи основных параметров: высота и ширина символов (h,w), начальные координаты сообщения (x,y) и само сообщение (a\$). И эта связь осуществляется благодаря тому, что эти параметры в БЕЙСИК-программе являются заданы программными переменными, а в подпрограмму в машинных кодах передается с помощью команд РОКЕ.

Может быть, Вы желаете использовать только программу машинных кодах, тогда Бейсик-программу можно опустить, передавая при этом необходимые параметры прямо в область памяти, зарезервированной для буфера принтера. Это выглядит более профессионально, но наличие БЕЙСИК-программы позволяет упростить работу и дает читателю возможность поэкспериментировать с различными значениями вышеупомянутых параметров.

БЕЙСИК-программа приведена в листинге 2.6.

Блок в машинных кодах приведен в листингах 2.7.1 - 2.7.5 и представляет собой расширенную версию программы 2.4.

Машиннокодový блок включает в себя следующие процедуры: FIND, SET, START, PLOT и SKIP. для простоты рассмотрим их по отдельности.

Процедура FIND (32335 - 32380) приведена в листинге 2.7.1. Она выполняет поиск значения БЕЙСИК-переменной по ее имени, которое должно быть предварительно задано. У нас имя искомой переменной задано в ячейке памяти 23728.

Это нужно сделать для того, чтобы передать из БЕЙСИКа в машинный код параметры x,y,h,w и само текстовое сообщение.

Сама по себе задача передачи параметров из БЕЙСИКа в машинный код - весьма интересная и важная. Для этого существуют много разных способов. Наиболее распространенный прием - через параметры функции пользователя FN ( ) мы рассмотрели и нашей книге "Элементарная графика" ("ИНФОРКОМ", М.:, 1992, с. 111).

Здесь С. Николс предлагает прием передачи параметров через БЕЙСИК-переменные. Вот их поиском и занимается процедура FIND. Если Вам покажется сложным то, как она это делает, значит Вы не вполне знакомы с форматом БЕЙСИК-переменных, в котором они хранятся в памяти компьютера. Тогда посмотрите "ZX-РЕВЮ-92" на с. 166.

Процедура SET (32381-32392) приведена в листинге 2.7.2. Она служит для того, чтобы передать параметры x,y,b,w, разысканные процедурой FIND в области БЕЙСИК-переменных в область буфера принтера (весьма удобное место для хранения данных, обычно ничем не занятое в компьютере. Эта область расположена всегда в адресах 23396-23551). Сюда же передается и количество символов в выводимой на экран сообщении, а также коды символов сообщения. Наша основная программа будет работать с параметрами, беря их из области буфера принтера.

Выполнение машинного кода в программе 2.7 начинается с адреса 32393. Это стартовый адрес процедуры START. Она приведена в листинге 2.7.3. Сначала (32393...32418) регистровая пара BC определяется в качестве указателя буфера принтера, после чего вызывается процедура SET для записи кода переменной в ячейку 23728. Затем (32419... 32442) определяется количество символов в переменной a\$ и переписываются

коды всех символов в буфер принтера. Процедура (32443... 32456) проверяет текущее значение координаты у. Если это значение будет больше 175, то выполняется операция у-176 и новые значения координат х и у записываются в ячейки 23728/9 и 23296/7.

Процедура PLOT (Листинг 2.7.4) представляет собой расширенную версию процедуры вывода символа на экран по точкам с добавлением циклов вывода на экран точки шириной w пикселей в h строках подряд.

Обратим внимание на то, что здесь неиспользуемые "Спектрумом" ячейки памяти 23728/9 служат для хранения текущих экранных координат позиции печати, а не для того, для чего они использовались в процедуре FIND.

Листинг 2.7.1

| АДРЕС | МАШ. КОД | АССЕМБЛЕР      | КОММЕНТАРИЙ                                                                                                                              |
|-------|----------|----------------|------------------------------------------------------------------------------------------------------------------------------------------|
| FIND  |          | ORG 32335      |                                                                                                                                          |
| 32335 | 2A 4B 5C | LD HL, (23627) | ; 23627 - адрес системной перемен. VARS,<br>; в которой хранится адрес, с которого<br>; начинается размещение переменных БЕЙСИКа.        |
| L2    |          |                |                                                                                                                                          |
| 32338 | 3A B0 5C | LD A, (23728)  | ; Код имени БЕЙСИК-переменной.                                                                                                           |
| 32341 | BE       | CP (HL)        | ; Сравнение и выход, если                                                                                                                |
| 32342 | C8       | RET Z          | ; переменная найдена, в HL<br>; остается ее адрес.                                                                                       |
| 32343 | CB 6E    | BIT 5, (HL)    | ; Если пятый бит в имени                                                                                                                 |
| 32345 | 20 08    | JR NZ, L1      | ; переменной равен нулю, то                                                                                                              |
| 32347 | 23       | INC HL         | ; это либо массив, либо                                                                                                                  |
| 32340 | 5E       | LD E, (HL)     | ; символьная строка. Нас                                                                                                                 |
| 32349 | 23       | INC HL         | ; они не интересуют, их на-                                                                                                              |
| 32350 | 56       | LD D, HL       | ; до пропустить. Следующие                                                                                                               |
| 32351 | 19       | ADD HL, DE     | ; два байта содержат их                                                                                                                  |
| 32352 | 23       | INC HL         | ; длину. На нее мы и перескакиваем.                                                                                                      |
| 32353 | 18 EF    | JR L2          | ; Возврат на повтор поиска.                                                                                                              |
| L1    |          |                |                                                                                                                                          |
| 32355 | CB 76    | BIT 6, (HL)    | ; Если шестой бит не равен                                                                                                               |
| 32357 | 20 0C    | JR NZ, L3      | ; нулю, то это либо одно символьная<br>; переменная, либо параметр цикла.<br>; Выясним это, перейдя на L3.                               |
| 32359 | 23       | INC HL         | ; Если же это многосим-                                                                                                                  |
| 32360 | 7E       | LD A, (HL)     | ; вольная переменная, то                                                                                                                 |
| 32361 | CB 7F    | BIT 7, (HL)    | ; пропускаем и ее и воз-                                                                                                                 |
| 32363 | 28 FA    | JR Z, -6       | ; вращаемся для дальнейших                                                                                                               |
| 32365 | 11 06 00 | LD DE, 6       | ; проверок на метку L2.                                                                                                                  |
| 32368 | 19       | ADD HL, DE     |                                                                                                                                          |
| 38369 | 18 DF    | JR L2          |                                                                                                                                          |
| L3    |          |                |                                                                                                                                          |
| 32371 | CB 7E    | BIT 7, (HL)    | ; Если седьмой бит равен                                                                                                                 |
| 32373 | 28 F6    | JR Z, -10      | ; нулю, это односимвольная<br>; переменная (хоть и<br>; не наша.) Вернемся на<br>; 32365, пропустим 6 байтов<br>; и повторим поиск с L2. |
| 32375 | 11 13 00 | LD DE, 19      | ; Седьмой бит равен единице                                                                                                              |
| 32378 | 19       | ADD HL, DE     | ; - это параметр цикла.<br>; В этом случае пропускаем<br>; 19 байтов и повторяем<br>; поиск с L2.                                        |
| 52379 | 18 D5    | JR L2          |                                                                                                                                          |

Листинг 2.7.2

| АДРЕС | МАШ. КОД | АССЕМБЛЕР     | КОММЕНТАРИЙ                                                      |
|-------|----------|---------------|------------------------------------------------------------------|
| SET   |          |               |                                                                  |
| 32381 | 32 B0 5C | LD (23728), A | ; в стандартном "Спектруме"<br>; ячейки 23728/9 не используются. |

|       |          |            |                             |
|-------|----------|------------|-----------------------------|
|       |          |            | ; Мы храним в них имя       |
|       |          |            | ; переменной (x, y, h...)   |
| 32384 | CD 4F 7E | CALL FIND  | ; Поиск значения переменной |
|       |          |            | ; по ее имени.              |
| 32387 | 23       | INC HL     | ; За именем переменной в    |
| 32388 | 23       | INC HL     | ; БЕЙСИKe хранится ее зна-  |
| 32389 | 23       | INC HL     | ; чение в пятибайтной фор-  |
|       |          |            | ; ме. Для целых чисел       |
|       |          |            | ; меньших 255 это форма:    |
|       |          |            | ; -----                     |
|       |          |            | ;  имя 00 Знак nn 00 00     |
|       |          |            | ; -----                     |
|       |          |            | ;                           |
|       |          |            | ; (HL)            (HL)+3    |
|       |          |            | ; Вот для чего нужны три    |
|       |          |            | ; команды INC HL подряд.    |
| 32390 | 7E       | LD A, (HL) | ; Переброска параметра в    |
| 32391 | 02       | LD (BC), A | ; буфер принтера.           |
| 32392 | C9       | RET        | ; Выход из процедуры.       |

Процедура SKIP (Листинг 2.7.5) завершает подпрограмму. Она ведет учет изменения экранной координаты позиции печати. В частности, она осуществляет переход к следующей строке, а когда достигнут низ экрана ( $y > 175$ ), она выполняет переход на первую строку за счет операции "y-175".

Выше мы отметили, что в процедуре ПЗУ PLOT имеется еще одна точка входа по адресу 8924, с помощью которой можно напечатать на экране точку, координаты которой заданы на вершине стека калькулятора. Мы говорили о том, что это может быть удобным при печати результатов расчета калькулятора.

Не углубляясь в подробности работы процедуры CALCULATOR, рассмотрим работу программы, которая использует адрес 8924 для входа в процедуру PLOT.

### Листинг 2.7.3

| АДРЕС | МАШ. КОД    | АСЕМБЛЕР       | КОММЕНТАРИЙ                 |
|-------|-------------|----------------|-----------------------------|
| START |             |                | ;Точка входа в программу.   |
| 32393 | 01 00 5B    | LD BC, 23296   | ;Начало буфера принтера.    |
| 32396 | 3E 7B       | LD A, 120      | ; 120 - код буквы "x".      |
| 32598 | CD 7D 7E    | CALL SET       | ; в 23296 - координаты "x". |
| 32401 | 03          | INC BC         | ;Очередной байт буфера.     |
| 32402 | 3E 79       | LD A, 121      | ; 121 - код буквы y.        |
| 32405 | CD 7D 7E    | CALL SET       | ; в 23297 - координаты "y". |
| 32407 | 03          | INC BC         | ;Очередной байт буфера.     |
| 32408 | 3E 68       | LD A, 104      | ; 104 - код буквы h.        |
| 32410 | CD 7D 7E    | CALL SET       | ; В 23298 - высота "h".     |
| 32413 | 03          | INC BC         | ;Очередной байт буфера.     |
| 32414 | 3E 77       | LD A, 119      | ; 119 - код буквы w.        |
| 32416 | CD 7D 7E    | CALL SET       | ; в 23299 - ширина "w".     |
| 32419 | 3E 41       | LD A, 65       | ; 65 - код буквы A.         |
| 52421 | 32 B0 5C    | LD (23728), A  | ; Подготовка и проведение   |
| 32424 | CD 4F 7E    | CALL FIND      | ; поиска переменной a\$.    |
| 32427 | 23          | INC HL         | ;                           |
| 32428 | 5E          | LD E, (HL)     | ; В DE помещается длина     |
| 32429 | 23          | INC HL         | ; нашего сообщения и пере-  |
| 32430 | 56          | LD D, (HL)     | ; дается в буфер принтера   |
| 32431 | ED 53 04 5B | LD(23300), DE  | ; По адресу 23300.          |
| 32435 | D5          | PUSH DE        | ; Переброска длины сообще-  |
| 32436 | C1          | POP BC         | ; ния из DE в BC.           |
| 32437 | 23          | INC HL         | ; Переброска самого текста  |
| 32438 | 11 05 5B    | LD DE, 23301   | ; сообщения в буфер принте- |
| 32441 | ED B0       | LDIR           | ; ра начиная с 23301.       |
| 32443 | 2A 00 5B    | LD HL, (23296) | ; В HL - координаты x, y.   |
|       |             |                | ; Причем y находится в H.   |

|       |             |                |                                    |
|-------|-------------|----------------|------------------------------------|
| 32446 | AF          | XOR A          | ; Сброс флага переноса (это        |
|       |             |                | ; необходимо перед операцией SBC). |
| 32447 | 7C          | LD A, H        | ; Координата y.                    |
| 32443 | DE B0       | SBC A, 176     | ; Проверка на >= 176.              |
| 32450 | 38 04       | JR C, +4       | ; Если нет, то переход на          |
|       |             |                | ; адрес 32456 (все 0.K.)           |
| 32452 | 67          | LD H, A        | ; В противном случае остав-        |
| 32453 | 22 00 5B    | LD (23296), HL | ; ляем "у минус 176" и за-         |
| 32456 | 22 B0 5C    | LD (23728), HL | ; поминаем копию в 23728/9.        |
|       |             |                | ; Итак, в 23728 - "х";             |
|       |             |                | ; в 23729 - "у-176".               |
| 32459 | 21 05 5B    | LD HL, 23301   | ; Указание на начало текста        |
| LP5   |             |                |                                    |
| 32462 | E5          | PUSH HL        | ; Запомнили начало текста.         |
| 32463 | 7E          | LD A, (HL)     | ; Взяли символ.                    |
| 32464 | 26 00       | LD H, 0        | ; Код символа передаем             |
| 32466 | 6F          | LD L, A        | ; в HL и                           |
| 32467 | 29          | ADD HL, HL     | ; умножаем его код на 8.           |
| 32468 | 29          | ADD HL, HL     |                                    |
| 32469 | 39          | ADD HL, HL     |                                    |
| 32470 | 11 00 3C    | LD DE, 15360   |                                    |
| 32473 | 19          | ADD HL, DE     | ; Нашли адрес, начиная с           |
|       |             |                | ; которого в ПЗУ хранится          |
|       |             |                | ; шаблон этого символа.            |
| 32474 | 06 08       | LD B, 8        | ; Счетчик байтов в шаблоне         |
| LP4   |             |                |                                    |
| 32476 | C5          | PUSH BC        | ; Запомнили его.                   |
| 32477 | ED 4B 01 5B | LD BC, (23297) | ; в "B" - высота символа;          |
|       |             |                | ; в "C" - его координата "у".      |
| LP3   |             |                |                                    |
| 32481 | 7E          | LD A, (HL)     | ; Байт шаблона символа.            |
| 32482 | E5          | PUSH HL        | ; Запомнили его.                   |
| 32483 | C5          | PUSH BC        | ; Запомнили "h" и "y".             |
| 32484 | 06 08       | LD B, 8        | ; Счетчик битов в байте            |
|       |             |                | ; шаблона.                         |
| LP2   |             |                |                                    |
| 32486 | C5          | PUSH BC        | ; Запомнили текущий байт.          |
| 32487 | 17          | RLA            | ; Ротация байта.                   |
| 32488 | F5          | PUSH AF        | ; Запомнили результат рота-        |
|       |             |                | ; ции вместе с флагами.            |
| 32489 | DA F9 7E    | JP C, PLOT     | ; Если флаг переноса вклю-         |
|       |             |                | ; чился, переходим на 32505        |
|       |             |                | ; для печати точки.                |
| 32492 | 2A 03 5B    | LD HL, (23299) | ; В регистр "L" помещается         |
|       |             |                | ; ширина "w".                      |
| 32495 | 3A B0 5C    | LD A, (23728)  | ; Координата "x".                  |
| 32498 | 85          | ADD A, L       | ; Нашли "x+w"                      |
| 32499 | 32 B0 5C    | LD (23728), A  | ; Запомнили в 23728.               |
| 32502 | C3 0F 7F    | JP SKIP        | ; Обход.                           |

Программа 2.8 демонстрирует построение по точкам синусоиды, описываемой формулой

$$88 + 80 * \sin(A/128 * \pi)$$

с применением процедуры CALCULATOR.

С помощью этой процедуры можно выполнить сложные арифметические операции, используя так называемые коды калькулятора. Обычно при вызове этой процедуры выполняется операция над двумя числами, являвшимися верхними на калькуляторном стеке, но возможны операции и над символьными величинами.

Например, если на вершине стека записаны два числа: 1234 и 2, а после включения процедуры CALCULATOR (RST 40) записан код 4 (DEFB 4), то эти два числа будут извлечены из калькуляторного стека, перемножены между собой, а их произведение будет записано обратно на вершину калькуляторного стека. Теперь на стеке будет записано только одно



число, равное 2468, если до этого там кроме сомножителей ничего не было записано.

Целые положительные числа могут быть занесены на стек следующими способами:

STACK VAL A - CALL 11560

STACK VAL BC - CALL 11563,

т.е. число предварительно записывается в регистр а или регистровую пару BC, а затем вызывается соответствующая процедура.

В программе 2.8 используется вызов STACK VAL A для размещения на стеке чисел в следующем порядке:

переменная А

88

80

переменная А

128

Вызов процедуры CALCULATOR осуществляется по команде RST40. Байты, записанные после этой команды, определяют вид выполняемой операции. В программе 2.8 сразу же после команды RST 40 следует байт, равный 5, что означает "Деление". При этом из стека извлекаются два верхних числа, выполняется деление одного числа на другое и результат записывается обратно в калькуляторный стек. Следующий байт - 163 является кодом операции "stack PI/2"- он переписывает на стек значение PI/2, хранимое в ПЗУ. Затем следует код 49 (duplicate top value) - он дублирует верхнее число на стеке, а код 15 "add top two values" производит суммирование двух верхних чисел и записывает результат обратно на стек. Теперь на вершине стека записано число PI. Следующий код 4 (multiply top two values) перемножает два верхних числа, т.е. число PI и A/128, и результат возвращается на стек. Код 31 (SIN of top value) определяет синус числа на вершине калькуляторного стека, а последующие коды 4 и 15 завершает расчет по записанной формуле. Последний код 56 "end calc" выполняет выход из процедуры, т.е. как бы выключает калькулятор и осуществляет возврат в программу в машинных кодах.

Листинг 2. 7.4

| АДРЕС         | МАШ. КОД    | АССЕМБЛЕР      | КОММЕНТАРИЙ                                                                |
|---------------|-------------|----------------|----------------------------------------------------------------------------|
| PL0T<br>32505 | ED 4B B0 5C | LD BC, (23298) | ; Процедура печати точки.<br>; В "B" - ширина "w"<br>; В "C" - высота "h". |
| LP1<br>32509  | C5          | PUSH BC        | ; Запомнили на стеке.                                                      |
| 32510         | ED 4B B0 5C | LD BC, (23298) | ; Повторили на стеке еще                                                   |
| 32514         | C5          | PUSH BC        | ; раз.                                                                     |
| 32515         | CD E5 22    | CALL 8933      | ; Вызов процедуры печати<br>; точки.                                       |
| 32518         | C1          | POP BC         | ; Переход к соседней                                                       |
| 32519         | 0C          | INC C          | ; точке.                                                                   |
| 32520         | ED 43 B0 5C | LD (23728), BC | ; Запомнили текущую коор-                                                  |
| 32524         | C1          | POP BC         | ; динату.                                                                  |
| 32525         | 10 EE       | DJNZ LP1       | ; Если не все точки напе-<br>; чатаны, возврат на 32509                    |

Листинг 2.7.5

| АДРЕС         | МАШ. КОД | АССЕМБЛЕР     | КОММЕНТАРИЙ                                                                                     |
|---------------|----------|---------------|-------------------------------------------------------------------------------------------------|
| SKIP<br>32527 | F1       | POP AF        | ; Восстановили текущий байт<br>; шаблона.                                                       |
| 32528         | C1       | POP BC        | ; Восстановили счетчик                                                                          |
| 32529         | 10 03    | DJNZ LP2      | ; Уменьшили его и, если он<br>; не обнулялся, возврат<br>; на LP2 для последующей<br>; ротации. |
| 32531         | 3A 00 5B | LD A, (23296) | ; Координата "x".                                                                               |
| 32534         | 21 B0 5C | LD HL, 23728  | ;                                                                                               |
| 32537         | 77       | LD (HL), A    | ; Принимаем "x" в качестве<br>; текущей экранной коорди-                                        |

|       |          |               |                             |
|-------|----------|---------------|-----------------------------|
| 32536 | 23       | INC HL        | ; наты.                     |
| 32539 | AF       | XOR A         | ; Указание на "у".          |
| 32540 | 7E       | LD A, (HL)    | ; Сброс флага переноса,     |
| 32541 | DE B0    | SBC A, 176    | ; Координата "у"            |
| 32543 | 38 03    | JR C, +3      | ; Проверка на "у>175".      |
| 32545 | 77       | LD (HL), A    | ; Если так, переход к 32546 |
| 32546 | 18 08    | JR +8         | ; "у"                       |
| 32548 | 7E       | LD A, (HL)    | ; Переход на 32556.         |
| 32349 | FE 00    | CP 0          | ; "у-176".                  |
| 32551 | 20 02    | JR NZ, +2     | ; Проверка на "ноль".       |
|       |          |               | ; Если не 0, переход на ад- |
|       |          |               | ; рес 32555.                |
| 32553 | 36 B0    | LD (HL), 176  |                             |
| 32555 | 35       | DEC (HL)      |                             |
| 32556 | C1       | POP BC        |                             |
| 32557 | E1       | POP HL        |                             |
| 32558 | 10 B1    | DJNZ LP3      |                             |
| 32560 | 23       | INC HL        | ; Переход к следующему бай- |
|       |          |               | ; ту шаблона.               |
| 32561 | C1       | POP BC        | ; Восстановили счетчик.     |
| 32562 | 10 A8    | DJNZ LP4      | ; Если он еще не обнулится, |
|       |          |               | ; переход на LP 4.          |
| 32564 | 3A 03 5B | LD A, (E3299) | ; Ширина символа "w".       |
| 33567 | 87       | ADD A, A      | ; Умножаем                  |
| 32566 | 67       | ADD A, A      | ; ее                        |
| 32569 | 67       | ADD A, A      | ; на 8.                     |
| 32570 | 6F       | LD L, A       | ; Запомнили в L.            |
| 32571 | 3A B0 5C | LD A, (25728) | ; Новая экранная коорд. "x" |
| 32574 | 85       | ADD A, L      |                             |
| 32575 | 32 00 5B | LD (23296), A | ; Запомнили ее.             |
| 32578 | 32 B0 5C | LD (23728), A | ; - "" -- "" -              |
| 32581 | 3A 01 5C | LD A, (23297) | ; Новая экранная коорд. "7" |
| 32584 | 3A B1 5C | LD (23729), A | ; Запомнили ее.             |
| 32587 | E1       | POP HL        | ; Восстановили указатель на |
|       |          |               | ; текущий символ.           |
| 32588 | 23       | INC HL        | ; Переход к новому символу. |
| 32569 | 3A 04 5B | LD A, (23300) | ; Длина сообщения.          |
| 32592 | 3D       | DEC A         | ; Умножаем ее на 1.         |
| 32593 | C8       | RET Z         | ; Проверка на конец. Если   |
|       |          |               | ; так, то выход.            |
| 32594 | 32 04 5B | LD (23300), a | ; Запомнили новую длину.    |
| 32597 | C3 CE 7E | JP LP5        | ; Возврат на печать очеред- |
|       |          |               | ; ного символа.             |

Существуют еще много других кодов калькулятора, которые могут быть использованы в Ваших программах для различных целей. С некоторыми из них мы еще познакомимся в следующих главах.

И еще одно важное замечание. Перед возвратом в BASIC-систему после применения процедуры CALCULATOR необходимо полностью очистить калькуляторый стек от всех записей. В программе 2.8 после последнего кода 56 на стеке все же остаются два числа: переменная A и результат вычисления по формуле  $88+80*\sin(A/128*\pi)$ . Процедура PLOT(CALL 6924) извлекает эти два значения, тем самым полностью очищая стек и использует эти два числа как координаты для пиксела.

Код программы для построения синусоиды приведен в листинге 2.8.

Листинг 2.8

| АДРЕС | МАШ. КОД | АССЕМБЛЕР  | КОММЕНТАРИЙ                 |
|-------|----------|------------|-----------------------------|
| 23760 | AF       | XOR A      | ; Обнуление аккумулятора.   |
| LOOP  |          |            |                             |
| 23761 | F5       | PUSH AF    | ; Запомнили его на стеке.   |
| 23762 | CD 28 2D | CALL 11560 | ; На кальк. стеке - текущее |
|       |          |            | ; значение аргумента.       |
| 23765 | 3E 5B    | LD A, 88   | ; На кальк. стеке -         |

|       |          |             |                                 |
|-------|----------|-------------|---------------------------------|
| 23767 | CD 28 2D | CALL 11560  | ; 88, A.                        |
| 23770 | 3E 50    | LD A, 80    | ; На кальк. стеке -             |
| 23772 | CD 28 2D | CALL 11560  | ; 80, 88, A.                    |
| 23775 | F1       | POP AF      | ; В акк-ре текущее значение     |
|       |          |             | ; аргумента.                    |
| 23776 | F5       | PUSH AF     | ; Запомнили его на стеке.       |
| 23777 | CD 28 2D | CALL 11560  | ; A, 80, 88, A.                 |
| 23780 | 3E 80    | LD A, 128   |                                 |
| 23782 | CD 28 2D | CALL 11560  | ; 128, A, 80, 88, A.            |
| 23765 | EF       | RST 40      | ; Включение калькулятора.       |
| 23766 | DEFB 05  | division    | ; A/128, 80, 88, A.             |
| 23787 | DEFB 163 | stack PI/2  | ; PI/2, A/128, 80, 88, A.       |
| 23788 | DEFB 49  | duplicate   | ; PI/2, PI/2, a/128, 80, 88, A. |
| 23789 | DEFB 15  | add         | ; PI, A/128, 80, 88, A.         |
| 23780 | DEFB 04  | multiply    | ; PI*A/128, 80, 88, A.          |
| 23781 | DEFB 31  | Sin         | ; SIN(A/128*PI), 80, 88, A.     |
| 23782 | DEFB 04  | multiply    | ; 80*SIN(A/128*PI), 88, A.      |
| 23783 | DEFB 15  | add         | ; 88+80*SIN(A/128*PI), A.       |
| 23784 | DEFB 56  | endcalc     | ; Выключение калькулятора.      |
| 23795 | CD DC 22 | CALL 8924   | ; Вызов процедуры PLOT.         |
| 23798 | F1       | POP AF      | ; Текущее значение аргумента.   |
| 23799 | 3C       | INC A       | ; Приращение аргумента.         |
| 23800 | FE 00    | CP 0        | ; Проверка на конец экрана.     |
| 23802 | 20 D5    | JR NZ, LOOP | ; Возврат для расчета и         |
|       |          |             | ; печати следующей точки.       |
| 23804 | C9       | RET         | ; Выход из процедуры.           |

### DRAW x,y.

Две точки входа в ПЗУ существуют и для реализации команды DRAW x,y. Для входа через первую точку (CALL 9402) необходимо, чтобы в регистре В было записано ABS(y), а в регистре С - ABS (x). Регистровая пара DE служит для хранения знака приращения координат x и y. Регистр D хранит знак приращения по x (SGN dx), а регистр С - знак приращения по y (1, если число положительное и 255, если число отрицательное). Программа 2.9 является модифицированной версией программы 2.8, которая демонстрирует реализацию команды DRAW. Обратите внимание, что перед вызовом процедуры DRAW, начальное значение регистровой пары H'L' (альтернативная) должно быть сохранено. Для этого содержимое этой пары записывается в машинный стек.

Начальное значение регистровой пары H'L' не должно быть потеряно в ходе выполнения программы, написанной пользователем, иначе при возврате в BASIC-систему произойдет сбой в работе компьютера. Начальное значение регистровой пары H'L' восстанавливается после команды DRAW. Команды OVER и INVERSE включаются таким же образом, как и PLOT OVER/INVERSE.

Для реализации команды DRAW x,y через вторую точку входа (CALL 9335) необходимо, чтобы значения x и y были записаны в калькуляторном стеке, причем значение y должно быть верхним в стеке. Чтобы использовать эту точку входа, необходимо прочитать главу 7, для ознакомления со способом записи 5-ти байтных чисел в калькуляторный стек.

### Листинг 2.9

| АДРЕС | МАШ. КОД | АССЕМБЛЕР | КОММЕНТАРИЙ                          |
|-------|----------|-----------|--------------------------------------|
| 23760 | 06 5F    | LD B, 95  | ; Координата "y".                    |
| 23762 | 0E 00    | LD C, 00  | ; Координата "x".                    |
| 23764 | CD E5 22 | CALL 8933 | ; Печать исходной точки.             |
| 23767 | D9       | EXX       | ; Включение регистров                |
|       |          |           | ; альтернативного набора.            |
| 23768 | E5       | PUSH HL   | ; Сохранение пара H'L'.              |
| 23769 | D9       | EXX       | ; Выключение альтернативного набора. |
| 23770 | 06 00    | LD B, 0   | ; Приращение по "y" = 0.             |
| 23772 | 0E FF    | LD C, 255 | ; Приращение по "x" = 255.           |
| 23774 | 16 01    | LD D, 1   | ; Знак приращ. по "x" = "+".         |
| 23776 | 1E 01    | LD E, 1   | ; Знак приращ. по "y" = "+".         |

|          |          |             |                                      |
|----------|----------|-------------|--------------------------------------|
| 23778    | CD BA 24 | CALL 9402   | ; Рисуем прямую линию.               |
| 23781    | D9       | EXX         | ; Включение регистров                |
|          |          |             | ; альтернативного набора.            |
| 23782    | E1       | POP HL      | ; Восстановление пары H'L'.          |
| 23783    | D9       | EXX         | ; Выключение альтернативного набора. |
| 23784    | AF       | XOR A       | ; Обнуление аккумулятора.            |
| LOOP     |          |             |                                      |
| 23765    | F5       | PUSH AF     | ; Запомнили его на стеке.            |
| 23766    | CD 28 2D | CALL 11560  | ; На кальк. стеке - текущее          |
|          |          |             | ; значение аргумента.                |
| 23789    | 3E 78    | LD A,120    | ; На кальк. стеке -                  |
| 23791    | CD 28 2D | CALL 11560  | ; 120, A.                            |
| 23794    | 3E 28    | LD A,40     | ; На кальк. стеке -                  |
| 23796    | CD 28 2D | CALL 11560  | ; 40, 120, A.                        |
| 23799    | F1       | POP AF      | ; в акк-ре текущее значение          |
|          |          |             | ; аргумента.                         |
| 23600    | F5       | PUSH AF     | ; Запомнили его на стеке.            |
| 23601    | CD 28 2D | CALL 11560  | ; A, 40, 120, A.                     |
| 23804    | 3E 10    | LD A,16     |                                      |
| 23806    | CD 28 2D | CALL 11560  | ; 16, A, 40, 120, A.                 |
| 23809    | EF       | RST 40      | ; Включение калькулятора.            |
| 23810    | DEFB 05  | division    | ; A/16, 40, 120, A.                  |
| 23811    | DEFB 163 | stack PI/2  | ; PI/2, A/16, 40, 120, A.            |
| 23813    | DEFB 49  | duplicate   | ; PI/2, PI/2, A/16, 40, 120, A.      |
| 23813    | DEFB 15  | add         | ; PI, A/16, 40, 120, A.              |
| 23814    | DEFB 04  | multiply    | ; PI*A/16, 40, 120, A.               |
| 23815    | DEFB 31  | Sin         | ; SIN(A/16*PI), 10, 120, A.          |
| 23816    | DEFB 04  | multiply    | ; 40*SIN(A/16*PI), 120, A.           |
| 25617    | DEFB 15  | add         | ; 120+40*SIN (A/16*PI), A.           |
| 23618    | DEFB 56  | endcalc     | ; Выключение калькулятора.           |
| 23819    | CD DC 22 | CALL 8924   | ; Вызов процедуры PLOT.              |
| 23822    | D9       | EXX         | ; Включение альтерн. набора          |
| 23823    | E5       | PUSH HL     | ; Сохраняя H'L'.                     |
| 23824    | D9       | EXX         | ; Выключение альтерн. наб.           |
| 23825    | 06 32    | LD B,50     | ; Приращение по "y".                 |
| 23827    | 0E 00    | LD C,0      | ; Приращение по "x".                 |
| 23829    | 16 FF    | LD D,255    | ; Знак приращения по "y"             |
|          |          |             | ; "минус".                           |
| 23831    | 1E 01    | LD E,01     | ; Знак приращения по "x" -           |
|          |          |             | ; "плюс".                            |
| 23833    | CD BA 24 | CALL 9402   | ; Рисование линии.                   |
| 23836    | D9       | EXX         | ; Включение альтерн. набора          |
| 23837    | E1       | POP HL      | ; Восстановление пары H'L'.          |
| 23838    | D9       | EXX         | ; Выключение альтернатив-            |
|          |          |             | ; ного набора.                       |
| 23839    | F1       | POP AF      | ; Текущее значение аргумента.        |
| 23840    | 3C       | INC A       | ; Приращение аргумента.              |
| 23841    | FE 00    | CP 0        | ; Проверка на конец экрана.          |
| 23843    | 20 D5    | JR NZ, LOOP | ; Возврат для расчета и              |
|          |          |             | ; печати следующей точки.            |
| 23845 C9 |          | RET         | ; Выход из процедуры.                |

### **DRAW x,y,a**

Точкой входа в ПЗУ для реализации этой команды служит адрес 9106. В этом случае необходимо, чтобы значения x, y и a были записаны в калькуляторный стек в таком порядке, чтобы значение "a" было верхним на стеке. Опять же, начальное значение регистровой пары H'L' (альтернативной) должно быть сохранено.

### **CIRCLE x,y,r**

Точкой входа в ПЗУ для реализации этой команды служит адрес 9005. При этом необходимо, чтобы значения x, y, и r были записаны в калькуляторном стеке. Начальное значение регистровой пары H'L' должно сохраниться в отдельном месте в течение работы

процедуры CIRCLE.

Программа 2.10 демонстрирует метод рисования концентрических окружностей. Обратите внимание на тот факт, что эта программа в машинных кодах выполняется также медленно, как и на языке BASIC. Это потому, что процедура CIRCLE в ПЗУ достаточно длинная. Если в Вашей программе возникнет необходимость построить окружность, то будет быстрее, если Вы будете строить ее по точкам, используя для постановки пиксела координаты, записанные как последовательность байтов с именем DATA.

Машинный код процедуры представлен в листинге 2.10. В качестве аналога выступает следующая БЕЙСИК-программа:

```
10 REM CIRCLE x,y,r
20 FOR b= 1 TO 2
30 FOR a= 1 TO 21 STEP 2
40 CIRCLE OVER 1; 128,88,a
50 NEXT a
60 NEXT b
```

Листинг 2.10

| АДРЕС       | МАШ. КОД    | АСЕМБЛЕР      | КОММЕНТАРИЙ                  |
|-------------|-------------|---------------|------------------------------|
| 23760<br>L2 | 06 02       | LD B, 2       | ; FOR b=1 TO 2               |
| 23762       | C5          | PUSH BC       |                              |
| 23763<br>L1 | 3E 01       | LD A, 1       | ; FOR a=1 TO ...             |
| 23765       | F5          | PUSH AF       |                              |
| 23766       | FD 36 57 03 | LD (IY+87), 3 | ; OVER 1                     |
| 23770       | 3E 80       | LD A, 128     |                              |
| 23772       | CD 28 2D    | CALL 11560    | ; На кальк. стеке - 128.     |
| 23775       | 3E 58       | LD A, 88      |                              |
| 23777       | CD 28 2D    | CALL 11560    | ; На кальк. стеке - 88, 128. |
| 23780       | F1          | POP AF        | ; текущее "a".               |
| 23781       | F5          | PUSH AF       |                              |
| 23782       | CD 28 2D    | CALL 11560    | ; На к. стеке - a, 88, 128.  |
| 23785       | D9          | EXX           |                              |
| 23786       | E5          | PUSH HL       | ; Сохранение H'L'.           |
| 23787       | D9          | EXX           |                              |
| 23788       | CD 2D 23    | CALL 9005     | ; CALL CIRCLE                |
| 23791       | D9          | EXX           |                              |
| 23792       | E1          | POP HL        | ; Восстановление H'L'.       |
| 23793       | D9          | EXX           |                              |
| 23794       | F1          | POP AF        | ; Текущее "a".               |
| 23795       | 3C          | INC A         |                              |
| 23796       | 3C          | INC A         | ; STEP 2                     |
| 23797       | FE 17       | CP 23         | ; a = 23?                    |
| 23798       | 20 DC       | JR NZ, L1     | ; Цикл не завершен.          |
| 23801       | C1          | POP BC        | ; цикл по "A" завершен.      |
| 23802       | 10 D6       | DJNZ L2       | ; Конец цикла по "b".        |
| 23804       | FD 36 57 00 | LD (IY+87), 0 | ; - OVER 0.                  |
| 23808       | C9          | RET           | ; Выход                      |

### 3. Счет

Во многих игровых программах необходимо постоянно вести счет, например, счет очков, потерянных "жизней", времени и т.д. Известны, по крайней мере, три способа организации счета в программах и вывода результатов счета на экран.

Первый настолько непригляден и занимает так много памяти, что мы рассмотрим его кратко, лишь в общих чертах. Способ этот позволяет организовать счет в программе от 0 до очень большого числа. Максимально возможное число определяется только количеством байтов, зарезервированных для представления этого числа. Например, для того, чтобы получить счетчик от 0 до 999 999, необходимо зарезервировать 6 байтов, т. е. для каждого

разряда числа нужен 1 байт. Поскольку показание счетчика выводится на экран как значение символьной переменной, то на экране в начальный момент времени будет 6 нулей: 000 000. Затем, с увеличением значения на 1, изменения произойдут только в разряде единиц. Если же последнее записанное в этом разряде число равно 9, то с добавлением 1 этот разряд обнулится, а единица добавится к числу в следующем разряде. Процедура проверки значения числа выполняется для каждого разряда и, если во всех шести разрядах записано число 9, то дальнейший счет прекращается. Для того, чтобы шаг счетчика был больше, чем 1, необходимо включить в процедуру счета специальные циклы.

Второй способ используется для создания счетчиков, показания которых не выходят за пределы диапазона 0...65535. Реализация этого способа показана в программе 3.1. В этой программе используется одна из процедур ПЗУ, позволяющая выводить на экран значение числа, записанного на вершине калькуляторного стека. Вызов этой процедуры осуществляется командой CALL 11747. При этом текущее значение счетчика может, например, храниться в неиспользуемой системной переменной 23726. Значение счетчика извлекается из этой системной переменной, увеличивается на 1, а затем вновь записывается в эту переменную и на вершину калькуляторного стека. После этого определяются параметры PRINT AT и вызывается процедура 11563, печатающая содержимое стека калькулятора.

### Листинг 3. 1

|       |               |                     |
|-------|---------------|---------------------|
| BEGIN | LD BC,0       | ; Инициализация     |
|       | LD (23728),BC | ; счетчика по       |
|       | CALL 11563    | ; адресу 23728.     |
|       |               | ; Переброска на     |
|       | LD A,2        | ; стек калькулятора |
|       | CALL 5633     | ; Открыли канал     |
|       | LD A,22       | ; печати на экран.  |
|       | RST 16        | ; Аналог ... AT 22  |
|       | LD A,16       |                     |
|       | RST 16        | ; Аналог AT 22,16   |
|       | CALL 11747    | ; Печать счетчика.  |
|       | LD BC,(23728) | ; Вызов счетчика.   |
|       | INC BC        | ; Приращение.       |
|       | LD A,B        | ; Проверка на       |
|       | OR C          | ; обнуление.        |
|       | RET Z         | ; Выход, если 0.    |
|       | BIT 5,(IY+1)  | ; Проверка не была  |
|       |               | ; ли нажата клавиша |
|       | JR Z, BEGIN   | ; Повтор, если не   |
|       |               | ; было нажатия.     |
|       | RET           | ; Выход, если кла-  |
|       |               | ; виша была нажата. |

Если в регистровой паре значение числа становится равным 65535, то осуществляется возврат в BASIC систему. В эту программу включена также процедура EXIT (строки 23795...23800), благодаря которой имеется возможность прервать счет и вернуться в BASIC систему, если нажата какая-нибудь клавиша. Это удобно, так как счет от 0 до 65535 выполняется несколько минут.

Третий способ также основан на использовании калькулятора. Выше, когда мы говорили о калькуляторном стеке, мы не останавливались на детальном рассмотрении способа записи и хранения чисел в стеке. Числа в стеке хранятся в 5-байтной форме, что позволяет калькулятору легко выполнять действия с ними. Более подробно об этом будет сказано в седьмой главе. Здесь же приводим программу, которая выполняет счет от 0 до бесконечно большого числа с шагом 250. При достижении показания счетчика 99 999 999, форма вывода на экран меняется на Е-форму (1Е+9). Однако счетчик, у которого диапазон представляемых чисел достигает 100 млн. вполне пригоден для большинства программ и на этот факт можно не обращать внимание.

В программе 3.2 есть много важных процедур, которые необходимо объяснить. Начальное значение счетчика, равное нулю, записывается в стек с помощью кода калькулятора 160 (stack\_zero). Это число занимает 5 верхних байтов стека, содержимое которых затем копируется в буфер принтера, используемого как "хранилище" для счетчика. Копирование выполняется командой LDIR. Значение, записанное в регистровой паре HL при этом считается равным STACKEND-5.

Значение счетчика выводится на экран с помощью процедуры 11747. Затем в стек записывается число 250 (шаг изменения показания счетчика) и число из "хранилища" в 5-байтной форме. Следующая команда вызова калькулятора RST 40 и код калькулятора 15(add) обеспечивают суммирование двух чисел, записанных на вершине стека, и осуществляется переход на начало цикла для передачи нового значения в "хранилище" и вывода этого значения на экран. В этой программе предусмотрена также процедура преждевременного выхода из цикла счета, если будет нажата любая клавиша. Перед возвратом в BASIC-систему после окончания счета стек обнуляется путем перемещения последнего записанного там значения.

Для создания счетчика, значение которого с каждым шагом уменьшается, необходимо в стек первоначально записать максимальное значение счетчика (запись в стек чисел больше, чем 65535, показана в главе 7), а вместо кода калькулятора для сложения - 15 (add) следует использовать код для вычитания 3 (subtract). Необходимо помнить, что при вычитании верхнее в стеке число вычитается из числа, записанного вторым. После каждой операции вычитания проверяется достигнут ли 0 (либо включен ли 7 бит результата) и если это так, то счет прекращается. Перед выводом очередного значения счетчика необходимо очищать экран, чтобы избежать ошибок, связанных с устареванием экранной информации. Например, если экран не очищать, то при изменении значения счетчика с 1000 на 999 на экране будет изображено не 999, а 9990.

## Листинг 3.2

```

L1  ORG 23760
    RST 40                ; включили кальк-р.
    DEFB 160              ; На стек - 0.
    DEFB 56               ; Выключили кальк-р.

    LD DE, 23296           ; начало буфера
                           ; принтера.
    LD BC, 5              ; Число перебрасы-
                           ; ваемых байтов.
    LDIR                  ; Переброска.
    LD A, 2               ; Канал экрана.
    CALL 5633             ; Открыли канал.
    LD A, 22              ; ... AT 22 ...
    RST 16
    LD A, 11              ; ... AT 22, 11
    RST 16
    CALL 11747            ; Печать показаний
                           ; счетчика.
    LD A, 250             ; Шаг счетчика.
    CALL 11560            ; Поместили его на
                           ; стек кальк-ра.
    LD HL, 23296          ; Начало буфера
                           ; принтера.
    LD DE, (23653)        ; 23553 - системная
                           ; переменная STKEND.
    LD BC, 5              ; Число перебрасы-
                           ; ваемых байтов.
    LDIR                  ; Переброска.
    LD (23653), DE        ; Новое значение
                           ; STKEND.
    RST 40                ; Включили кальк-р.
    DEFB 15               ; Код сложения (add)

```

|                |                     |
|----------------|---------------------|
| DEFB 56        | ; Выключение к-ра.  |
| BIT 5(IY+1)    | ; Проверка не была  |
|                | ; ли нажата какая - |
|                | ; либо клавиша.     |
| JR Z, L1       | ; Если нет, то      |
|                | ; повтор.           |
| LD HL, (23653) | ; (STKEND)          |
| DEC HL         |                     |
| DEC HL         |                     |
| DEC HL         |                     |
| DEC HL         |                     |
| DEC HL         |                     |
| LD (23553), HL | ; (STKEND)-5        |
| RET            | ; Выход             |

И завершим мы материал этого номера небольшой игрой, в которой объединим то, о чем Вы сегодня прочитали. Программа 3.3 показывает возможность использования счетчика для измерения скорости Вашей реакции.

Программа состоит из двух частей: первая часть написана на БЕЙСИКе и служит в качестве интерфейса между пользователем и компьютером. Она не требует скорости в работе. Машинный код служит для выдачи сообщений и запросов нестандартным шрифтом и для измерения скорости реакции. Печать нестандартным шрифтом выполняется с помощью блока машинного кода, приведенного в листинге 2.7, рассмотренного ранее. Он должен быть подгружен, начиная с адреса 32335. измерение скорости реакции выполнит процедура 3.3.2, приведенная ниже.

Измерение скорости вашей реакции производится следующим образом. Сначала процедура 2.7 нарисует на экране большую букву от А до Z, после чего будет включен режим CAPS LOCK и Вам надо нажать соответствующую клавишу. Время Вашей реакции замерит процедура 3.3.2.

В ней применен один хитрый прием. Дело в том, что рисование с помощью PLOT буквы на экране происходит не очень быстро, и за то время, пока идет этот процесс, Вы можете в принципе догадаться, что это за буква и подготовиться к нажатию соответствующей клавиши. Чтобы этого не было, буква рисуется желтым цветом INK=6 по желтому фону PAPER=6 и ее на экране не будет видно. Когда же дело дойдет до измерения Вашей реакции, процедура 3.3.2 включит на экране цвет пикселей INK=1 (черный). Произойдет это с огромной скоростью и буква появится мгновенно.

БЕЙСИК-программа стартует со строки 9800, в которой записана подгрузка модулей в машинных кодах. Поэтому набрав Бейсик выгрузите его на ленту со строкой автостарта 9800:

```
SAVE "counter" LINE 9800
```

### Листинг 3.3.1

```
12 PAPER 6: CLS
15 DATA 87,143,1,2, "COUNT"
20 DATA 95,63,1,2, "SLOW"
30 DATA 71,63,1,2, "AVERAGE"
40 DATA 95,63,1,2, "GOOD"
50 DATA 63,63,1,2, "VERY GOOD"
60 DATA 63,63,1,2, "EXCELLENT"
80 DATA 47,63,1,2, "NOT TRYING"
90 DATA 0,31,1,2, "ANOTHER GO? y/n"
95 DATA 55,125,5,2, "THANK YOU"
96 DATA 103,79,3,2, "for"
97 DATA 71,50,5,2, "PLAYING"
100 DATA 24,79,5,2, "STOP THE TAPE"
105 DATA 15,167,1,2, "REACTION TIMER"
110 DATA 13,164,2,2, "-----"
120 DATA 24,15,1,2, "PRESS ANY KEY"
125 RESTORE 100: FOR a=1 TO 4:PAUSE 25: INK 2: GO SUB 9000:NEXT a
180 PAUSE 0: BORDER 5: PAPER 6: INK 0: CLS
```



```

190 RESTORE 105: INK 0: GO SUB 9000
290 PRINT AT 20,5; "PRESS ANY KEY TO PLAY"
300 IF INKEY$<>" " THEN GO TO 300
302 IF INKEY$=" " THEN GO TO 302
310 CLS: RESTORE 105: INK 1: GO SUB 9000: INK 5: GO SUB 9000
330 RESTORE 15: INK 2: GO SUB 9000
350 INK 0: PLOT 102,114: DRAW 50,0: DRAW 0,-38: DRAW 50,0: DRAW 0,38
380 LET z=INT (RND*26+65)
390 LET x-111: LET y=111: LET h=4: LET w=4: LET a$=CHR$ z
400 INK 6: LET c=USR 33393: FOR a=1 TO RND*200+50: NEXT a
420 INK 0: LET c=USR 30000: LET a=PEEK 23728+256*PEEK 237729
450 IF a>=140 OR a=0 THEN RESTORE 20
460 IF a<140 THEN RESTORE 20
470 IF a<120 THEN RESTORE 30
480 IF a<100 THEN RESTORE 40
490 IF a<85 THEN RESTORE 50
500 IF a<70 THEN RESTORE 60
510 GO SUB 9000
7000 RESTORE 90:INK 1:GO SUB 9000
7010 IF INKEY$<>" " THEN GO TO 7010
7020 IF INKEY$=" " THEN GO TO 7020
7030 LET d$=INKEY$
7040 IF d$ <> "y" THEN GO TO 8000
7050 LET c=USR 30082: GO TO 350
8000 RESTORE 95: POKE 30083,20: LET c=USR 30082: POKE 30083,18
8030 FOR a=1 TO 3: GO SUB 9000: NEXT a: STOP
9000 READ x, y, h, w, a$: LET c=USR 32393: RETURN
9800 CLEAR 29999: LOAD ""CODE: GO TO 12

```

### Листинг 3.3.2

| АДРЕС | МАШ. КОД    | АСЕМБЛЕР       | КОММЕНТАРИЙ                  |
|-------|-------------|----------------|------------------------------|
| 30000 | 3E 7A       | LD A, 122      | ; Код буквы "z"              |
| 30002 | 01 81 5C    | LD BC, 23681   | ; Для хранения параметра "z" |
|       |             |                | ; выбирается неиспользуемая  |
|       |             |                | ; в стандартном "Спектруме"  |
|       |             |                | ; ячейка 23681.              |
| 30005 | CD 7D 7E    | CALL 32381     | ; Параметр "z" перебрасыва-  |
|       |             |                | ; ется в 23681 (см. проце-   |
|       |             |                | ; дуры 2.7.1, 2.7.2.)        |
| 30008 | FD CB 30 DE | SET 3, (IY+48) | ; Включение 3-го бита сис-   |
|       |             |                | ; темной переменной FLAGS2   |
|       |             |                | ; (23656) эквивалентно вклю- |
|       |             |                | ; чению режима CAPS LOCK.    |
| 30012 | 21 00 58    | LD HL, 22528   | ; Первый байт области цвето- |
|       |             |                | ; вых атрибутов экрана.      |
| L1    |             |                |                              |
| 30015 | 7E          | LD A, (HL)     | ; Проверка этого байта.      |
| 30016 | FE 36       | CP 54          | ; Переключение атрибутов     |
| 30018 | 20 02       | JR NZ, L2      | ; экранной области с ре-     |
| 30020 | 36 30       | LD (HL), 48    | ; жима PAPER = 6, INK = 6    |
| L2    |             |                | ; на режим PAPER = 6.        |
| 30022 | 23          | INC HL         | ; INK = 0.                   |
| 30023 | 7C          | LD A, H        | ; Ранее нарисованная бук-    |
| 30024 | FE 5B       | CP 91          | ; ва мгновенно "проявля-     |
| 30026 | 20 F3       | JR NZ, L1      | ; ется" на экране.           |
| 30028 | 01 00 00    | LD BC, 0       | ; Инициализация счетчика,    |
| L3    |             |                |                              |
| 30031 | C5          | PUSH BC        |                              |
| 30032 | CD 2B 2D    | CALL 11563     | ; Помещение его на стек      |
|       |             |                | ; калькулятора.              |
| 30035 | 3E 02       | LD A, 2        | ; Открываем канал печати     |
| 30037 | CD 01 16    | CALL 5633      | ; на экран.                  |
| 30040 | 3E 16       | LD A, 22       | ; Аналог ... AT ...          |
| 30042 | D7          | RST 16         |                              |

|       |             |                |                              |
|-------|-------------|----------------|------------------------------|
| 30043 | 3E 06       | LD A, 6        | ; Аналог ... AT 6...         |
| 30045 | D7          | RST 16         |                              |
| 30046 | 3E 0E       | LD A, 14       |                              |
| 30046 | D7          | RST 16         | ; Аналог ... AT 6, 14        |
| 30049 | CD E3 2D    | CALL 11747     | ; Печать показаний счетчика. |
| 30052 | C1          | POP BC         |                              |
| 30053 | 03          | INC BC         | ; Увеличение счетчика.       |
| 30054 | 78          | LD A, B        | ; Проверка счетчика на об-   |
| 30055 | B1          | OR C           | ; нуление.                   |
| 30056 | 28 0F       | JR Z, END      | ; Выход, если он успел обну- |
|       |             |                | ; литься (достиг 256).       |
| 30058 | FD CB 01 6E | BIT 5, (IY+1)  | ; Проверка не была ли нажата |
|       |             |                | ; какая-либо клавиша.        |
| 30062 | 28 DF       | JR Z, L3       | ; Если нет, то возврат на    |
|       |             |                | ; повтор.                    |
| 30064 | 3A 08 5C    | LD A, (23560)  | ; Код нажатой клавиши.       |
| 30067 | 21 81 5C    | LD HL, 23681   | ; Код буквы, которая была    |
|       |             |                | ; показана на экране.        |
| 30070 | BE          | CP (HL)        | ; Сравнили их между собой.   |
| 30071 | 20 06       | JR NZ, L3      | ; Если они не совпадают,     |
|       |             |                | ; значит нажата не та клави- |
|       |             |                | ; ша. Следует повторить по-  |
|       |             |                | ; пытку.                     |
| END   |             |                |                              |
| 30073 | ED 43 B0 5C | LD(23728), BC  | ; Результат испытания пере-  |
|       |             |                | ; дается из BC в адрес 23728 |
| 30077 | FD CB 30 9E | RES 3, (IY+48) | ; Выключается ранее включен- |
|       |             |                | ; ный режим CAPS LOCK.       |
| 30081 | C9          | RET            | ; Выход из программы.        |
| 13082 | 06 12       | LD B, 18       | ; Эта вспомогательная проце- |
| 30084 | CD 44 0E    | CALL 3652      | ; дура служит для очистки    |
| 30087 | C9          | RET            | ; к нижних строк экрана, где |
|       |             |                | ; k-число, установленное в   |
|       |             |                | ; адресе 30083 (исходно 18). |

# FORUM

Как эффективнее всего освоить машинные коды? Опыт показывает, что лучше всего освоение проходит, когда берешь какую-нибудь несложную готовую программу в машинных кодах и пытаешься понять, что и как там делается. При этом неизбежно возникает вопросы: "А почему так? Может можно все это сделать по-другому?" Начинаешь пробовать сделать по-другому и выясняется, что по-другому не выходит. Тогда становится понятно, почему автор организовал программу именно так, а не иначе. Но иногда встречаются ситуации, когда приходят действительно ценные идеи. Именно потому, что "ход мысли" начинающего еще не оформился в каких-то рамках, он не знает, что "можно", а что "нельзя". И он пытается выполнить задачу по-своему, а это зачастую приводит к находке новых, оригинальных решений.

Александр Балашов из г. Конаково Тверской обл. пишет нам о том, что он начал заниматься машинными кодами совсем недавно. Исследуя программы компрессии и декомпрессии, опубликованные в "ZX-РЕВЮ" N 1-2 за 1991 г., Александр обратил внимание, что под счетчик числа одинаковых байтов используется двухбайтный регистр ВС. Учитывая организацию экранной области "Спектрума", он предположил, что выгоднее, видимо, использовать один байт, так как 256 байтов перекрывают 8-ю штрихами третью часть экрана и старший байт двухбайтного регистра ВС используется лишь при практически пустом экране. Число, записанное в этом регистре занимает в скомпрессированном блоке 2 байта, один из которых почти всегда в нуле.

Этот факт побудил Александра разработать свою программу компрессии, в которой под счетчик числа одинаковых байтов он использовал только регистр В. Этот вариант позволяет получать длину скомпрессированного блока на 10-20% меньшую. Лишь при почти незаполненном экране, когда длина скомпрессированного блока меньше 300-400 байтов, длина его скомпрессированной картинки равна или уступает указанному варианту компрессора. Кроме того, его вариант не приводит к сбоям, которые могут иметь место при определенных ситуациях в варианте компрессора, упомянутом выше.

Александр приводит результаты сравнительных испытаний двух компрессоров для взятых наугад нескольких экранов игровых программ. Вот полученные длины скомпрессированных блоков (в байтах) соответственно для двух вариантов компрессоров:

|              |             |
|--------------|-------------|
| "Death Star" | 4619 и 5936 |
| "Cauldron"   | 3228 и 2821 |
| "Scooby Doo" | 5958 и 5265 |
| "XENO"       | 4985 и 4372 |
| "Saboteur"   | 4027 и 3474 |

Вариант компрессора, который приводит Александр, заинтересовал нас и мы провели тестирование присланного им блока кодов, которое подтвердило основные моменты, изложенные выше. Но выяснилось, что полученный скомпрессированный блок кодов нельзя загружать для декомпрессии под любой адрес, а только под адрес, в котором он был скомпрессирован. Это серьезное ограничение для пользователей, которое может свести к нулю преимущество нового варианта компрессора. Хотя проблема в общем-то решается довольно просто. Те, кто регулярно читают "РЕВЮ", наверняка знают, как осуществить привязку к конкретным адресам загрузки.

Мы сделали такое усовершенствование программы, присланной Александром. Изменения коснулись программы декомпрессии (она стала длиннее на 7 байтов), из-за чего пришлось на 7 байтов отодвинуть программу компрессии в сторону младших адресов. В таком усовершенствованном виде мы предлагаем программу читателям (см. Лист. 1)

Листинг 1.

|      |        |              |                                    |
|------|--------|--------------|------------------------------------|
| 7F8B | 210040 | LD HL, #4000 | ; В HL указатель адреса в экранной |
|      |        |              | ; области.                         |
| 7F8E | 01001B | LD BC, #1B00 | ; В BC счетчик байтов экрана.      |
| 7F91 | 110080 | LD DE, #8000 | ; В DE задан адрес, куда будет     |
|      |        |              | ; производиться компрессирование.  |

|      |      |      |              |                                                                                                                                                                                                  |
|------|------|------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7F94 | D5   |      | PUSH DE      | ; Он запоминается на стеке для того, чтобы потом можно было вычитать длину полученного сжатого блока.                                                                                            |
| 7F95 | 1B   |      | DEC DE       | ; Эта операция выполняется ввиду того, что при возврате из операции подсчета повторов и сравнений байтов в DE должен находиться адрес первого байта пустого места.                               |
| 7F96 | 13   | L0   | INC DE       | ; Проверка BC на ноль, то есть определение конца компрессирования.                                                                                                                               |
| 7F97 | 78   | L1   | LD A, B      | ; Если ноль, то переход на END1                                                                                                                                                                  |
| 7F98 | B1   |      | OR C         | ; в аккумулятор заносится байт из экранной области и переносится по адресу, который указан в DE. DE и HL увеличиваются, а BC - уменьшается на единицу.                                           |
| 7F99 | 2828 |      | JR Z, #7FC3  | ; Сравнение следующего байта с предыдущим.                                                                                                                                                       |
| 7F9A | 7E   |      | LD A, (HL)   | ; Если не равны, то возврат на L1.                                                                                                                                                               |
| 7F9B | EDA0 |      | LDI          | ; Проверка BC на ноль (конец). В данном случае нельзя использовать аккумулятор, т.к. в нем находится байт из экранной области, который сравнивается со следующим.                                |
| 7F9C |      |      |              | ; Загрузка первого повторяющегося байта в приемник.                                                                                                                                              |
| 7F9D |      |      |              | ; Приращение DE для того, чтобы организовать по этому адресу счетчик повторов. Один повтор уже есть, поэтому сразу заносится единица, а не ноль. Это оказывается удобнее при декомпрессировании. |
| 7F9E | BE   |      | CP (HL)      | ; Сохраняем в A экранный байт, вызываем альтернативный регистр для организации в нем счетчика повторов.                                                                                          |
| 7F9F | 20F6 |      | JR NZ, #7F97 | ; Как было сказано выше, в счетчик сразу заносится единица. Здесь будет организован счетчик повторов.                                                                                            |
| 7FA0 | 0D   |      | DEC C        | ; Проверка на переполнение. Максимальное число повторов - 255.                                                                                                                                   |
| 7FA1 | 0C   |      | INC C        | ; Запись в приемник числа повторов.                                                                                                                                                              |
| 7FA2 | 2004 |      | JR NZ, #7FA9 | ; Переход к следующему байту экрана.                                                                                                                                                             |
| 7FA3 | 05   |      | DEC B        | ; Уменьшение счетчика байтов экрана.                                                                                                                                                             |
| 7FA4 | 04   |      | INC B        | ; Проверка BC                                                                                                                                                                                    |
| 7FA5 | 281A |      | JR Z, #7FC3  | ; на ноль                                                                                                                                                                                        |
| 7FA6 | 12   | L2   | LD (DE), A   | ; без задерживания                                                                                                                                                                               |
| 7FA7 |      |      |              | ; аккумулятора.                                                                                                                                                                                  |
| 7FA8 |      |      |              | ; Обратный обмен регистров для сравнения текущего байта с предыдущим.                                                                                                                            |
| 7FA9 |      |      |              | ; Сравнение.                                                                                                                                                                                     |
| 7FAA | 13   |      | INC DE       | ; Если не равны, то возврат на L1                                                                                                                                                                |
| 7FAB | 08   |      | EX AF, AF'   | ; Если равны, то восстанавливаем счетчик повторов и переходим на L3 для наращивания счетчика.                                                                                                    |
| 7FAC | 3E00 |      | LD A, #00    | ; Эта операция нужна для того, чтобы при выходе из программы компрессии в DE содержался адрес последнего байта сжатого блока.                                                                    |
| 7FAD | 3C   | L3   | INC A        | ; Берем последний байт из DE, инвертируем его                                                                                                                                                    |
| 7FAE |      |      |              | ; и записываем в следующий адрес;                                                                                                                                                                |
| 7FAF | 28E5 |      | JR Z, #7F96  |                                                                                                                                                                                                  |
| 7FB0 |      |      |              |                                                                                                                                                                                                  |
| 7FB1 | 12   |      | LD (DE), A   |                                                                                                                                                                                                  |
| 7FB2 | 23   |      | INC HL       |                                                                                                                                                                                                  |
| 7FB3 | 0B   |      | DEC BC       |                                                                                                                                                                                                  |
| 7FB4 | 0D   |      | DEC C        |                                                                                                                                                                                                  |
| 7FB5 | 0C   |      | INC C        |                                                                                                                                                                                                  |
| 7FB6 | 2004 |      | JR NZ, #7FBC |                                                                                                                                                                                                  |
| 7FB7 | 05   |      | DEC B        |                                                                                                                                                                                                  |
| 7FB8 | 04   |      | INC B        |                                                                                                                                                                                                  |
| 7FB9 | 2608 |      | JR Z, #7FC4  |                                                                                                                                                                                                  |
| 7FBA | 08   | L4   | EX AF, AF'   |                                                                                                                                                                                                  |
| 7FBB |      |      |              |                                                                                                                                                                                                  |
| 7FBC |      |      |              |                                                                                                                                                                                                  |
| 7FBD | BE   |      | CP (HL)      |                                                                                                                                                                                                  |
| 7FBE | 20D6 |      | JR NZ, #7F96 |                                                                                                                                                                                                  |
| 7FC0 | 08   |      | EX AF, AF'   |                                                                                                                                                                                                  |
| 7FC1 | 18EB |      | JR #7FAE     |                                                                                                                                                                                                  |
| 7FC2 |      |      |              |                                                                                                                                                                                                  |
| 7FC3 | 1B   | END1 | DEC DE       |                                                                                                                                                                                                  |
| 7FC4 | 1A   | END2 | LD A, (DE)   |                                                                                                                                                                                                  |
| 7FC5 | 2F   |      | CPL          |                                                                                                                                                                                                  |
| 7FC6 | 13   |      | INC DE       |                                                                                                                                                                                                  |

|      |        |                |                                                                                                                                                                                                               |
|------|--------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7FC7 | 12     | LD (DE), A     | ; это сделано для упрощения декодирующей процедуры, т.к. в ней можно избежать лишней проверки счетчика на ноль.                                                                                               |
| 7FC8 | EB     | EX DE, HL      | ; Обмен DE и HL для сохранения DE.                                                                                                                                                                            |
| 7FC9 | D1     | POP DE         | ; Возвращаем со стека начальное значение DE.                                                                                                                                                                  |
| 7FCA | AF     | XOR A          | ; Сброс флага переноса, необходимо для выполнения следующей команды.                                                                                                                                          |
| 7FCB | ED52   | SBC HL, DE     | ; Отняв от конечной величины DE (которая теперь в HL) начальную, получим число байт скомпрессированной области, но без учета последнего контрольного байта, который является инверсной копией предпоследнего. |
| 7FCD | 22E57F | LD (#7FE5), HL | ; Подставляем полученное значение в программу декомпрессии.                                                                                                                                                   |
| 7FD0 | 012600 | LD BC, #0026   | ; Смещение, учитывающее длину декомпрессирующей процедуры.                                                                                                                                                    |
| 7FD3 | AF     | XOR A          | ; Сброс флагов.                                                                                                                                                                                               |
| 7FD4 | ED4A   | ACD HL, BC     | ; Получение в HL суммарной длины блока кодов с учетом декодирующей процедуры. Перезапись полученного значения в регистр BC и увеличение его на 1 для учета последнего (контрольного) байта.                   |
| 7FD6 | E5     | PUSH HL        |                                                                                                                                                                                                               |
| 7FD7 | C1     | POP BC         |                                                                                                                                                                                                               |
| 7FD8 | 03     | INC BC         |                                                                                                                                                                                                               |
| 7FD9 | C9     | RET            |                                                                                                                                                                                                               |

Последняя часть программы (с 7FD0H) позволяет получить выходной параметр в удобном для последующего использования виде. Для этого он был предусмотрительно помещен в регистр BC. Подав команду PRINT USR 32651 (7F8BH) результат работы компрессирующей процедуры (из регистра BC) будет выведен на экран: длина скомпрессированного блока (с учетом контрольного байта) плюс длина декомпрессора. А при подаче команды

```
LET L=USR 32651
```

- результат будет помещен в переменную L. На магнитную ленту должен быть выгружен блок кодов с адреса 32730 (7FDAH), длиной L.

|      |        |              |                                                                                                                          |
|------|--------|--------------|--------------------------------------------------------------------------------------------------------------------------|
| 7FDA | CD7C00 | CALL #007C   | ; Так выполняется привязка к адресу загрузки. В результате в HL получается адрес начала скомпрессированного блока кодов. |
| 7FDB | 3B     | DEC SP       | ; Оно образуется в результате увеличения адреса загрузки на величину, занимаемую программой декомпрессии.                |
| 7FDE | 3B     | DEC SP       | ; Сейчас здесь ноль, но при работе программы компрессии сюда будет подставлена длина скомпрессированного блока кодов.    |
| 7FDF | E1     | POP HL       | ; Адрес приемника - экрана.                                                                                              |
| 7FE0 | 012300 | LD BC, #0023 | ; Проверка BC                                                                                                            |
| 7FE3 | 09     | ADD HL, BC   | ; на ноль.                                                                                                               |
| 7FE4 | 010000 | LD BC, #0000 | ; Выход, если ноль.                                                                                                      |
| 7FE7 | 110040 | LD DE, #4000 | ; В аккумулятор заносится байт из скомпрессированной области.                                                            |
| 7FEA | 78     | LD A, B      | ; Перезапись байта в экран и переход к следующему байту экрана и приемника.                                              |
| 7FEB | B1     | OR C         | ; Сравнение его с предыдущим.                                                                                            |
| 7FEC | C8     | RET Z        | ; Если не равен, то переход на L0 и повтор операции перезаписи и сравнения.                                              |
| 7FED | 7E     | LD A, (HL)   |                                                                                                                          |
| 7FEE | EDA0   | LDI          |                                                                                                                          |
| 7FF0 | BE     | CP (HL)      |                                                                                                                          |
| 7FF1 | 20F7   | JR NZ, #7FEA |                                                                                                                          |

|      |      |               |                                         |
|------|------|---------------|-----------------------------------------|
| 7FF3 | 23   | INC HL        | ; Переход к следующему байту, в котором |
| 7FF4 | C5   | PUSH BC       | ; находится кол-во повторов.            |
| 7FF5 | 46   | LD B, (HL)    | ; Счетчик запоминается на стеке.        |
| 7FF6 | 12   | L1 LD (DE), A | ; Организуем в B счетчик повторов,      |
| 7FF7 | 13   | INC DE        | ; число которых берется из HL.          |
| 7FF8 | 10FC | DJNZ #7FF6    | ; Запись байта в экран.                 |
|      |      |               | ; Переход к следующему байту экрана.    |
|      |      |               | ; Уменьшение счетчика, если не 0 -      |
|      |      |               | ; то продолжение перезаписи байта       |
|      |      |               | ; из аккумулятора в экран.              |
| 7FFA | C1   | POP BC        | ; Снимаем BC со стека без проверки      |
|      |      |               | ; на ноль, т.к. переход в цикл          |
|      |      |               | ; загрузки одинаковых байтов может      |
|      |      |               | ; быть только из положительных зна-     |
|      |      |               | ; чений BC (нулевой байт инвертиро-     |
|      |      |               | ; ван по отношению к последнему).       |
| 7FFB | 23   | INC HL        | ; Переход к следующему байту ском-      |
| 7FFC | 0B   | DEC BC        | ; прессированной области.               |
|      |      |               | ; Теперь BC указывает на байт в HL,     |
|      |      |               | ; в котором находится число повто-      |
|      |      |               | ; ров и тоже не может быть в нуле.      |
| 7FFD | 0B   | DEC BC        | ; Уменьшение счетчика.                  |
| 7FFE | 18EA | JR #7FEA      | ; Переход на проверку BC и пере-        |
|      |      |               | ; запись байтов.                        |
| 8000 | .... |               | ; Сжатый блок кодов.                    |

Для того, чтобы можно было компрессировать экраны, записанные без заголовка, Александр рекомендует дополнить программу компрессора процедурой загрузки блока кодов без заголовка, которая использует процедуру 1386 (0556H) ПЗУ:

```

7F7E DD210040 LD IX, #4000
7F6E 11001B LD DE, #1B00
7F85 3EFF LD A, #FF
7F87 37 SCF
7F88 CD5605 CALL #0556

```

Эффективность той или иной программы во многом зависит от того, насколько продумана Бейсиковая ее часть. Мы приводим вариант программы "Компрессор", которая в основе содержит программу, присланную Александром. Она русифицирована способом, которым мы постоянно пользуемся (см. "ZX-РЕВЮ"-92 N 1,2, стр. 30).

```

1 GO TO 100
2 CLEAR 30000: LOAD "chr" CODE 64600
3 FOR N=32636 TO 32767: READ M: POKE N,M: NEXT N
4 RUN
5 SAVE "COMPR" LINE 2: STOP
6 POKE 23606,88: POKE 23607,251: RETURN
9 POKE 23606,0: POKE 23607,60: RETURN
100 BORDER 7: PAPER 7: INK 0: CLS
110 GO SUB 8
200 PRINT AT 7,1;"<1> - ЗАГРУЗКА ЭКРАННОГО ФАЙЛА" ' ТАБ 7; "С ЗАГОЛОВКОМ"
210 PRINT AT 10,1;"<2> - ЗАГРУЗКА БЛОКА КОДОВ" ' ТАБ 7; "БЕЗ ЗАГОЛОВКА" ТАБ 7; "С КОНТРОЛЕМ
    СЧИТЫВАНИЯ"
220 PRINT AT 14,1;"<3> - ЗАГРУЗКА БЛОКА КОДОВ" ' ТАБ 7; "БЕЗ ЗАГОЛОВКА" ТАБ 7; "БЕЗ КОНТРОЛЯ
    СЧИТЫВАНИЯ"
300 PAUSE 0: IF INKEY$<>"1" AND INKEY$<>"2" AND INKEY$<>"3" THEN GO TO 300
310 LET I$=INKEY$
400 CLS : PRINT #0; TAB 11; FLASH 1;" ЗАГРУЗКА ": GO SUB 9
410 IF I$="1" THEN LOAD ""CODE 16384,6912: LET L=USR 32651
420 IF I$="2" THEN POKE 32649,2: POKE 32650,8: LET L=USR 32638
430 IF I$="3" THEN POKE 32649,66: POKE 32650, 5: LET L=USR 32636
500 GO SUB 8: INPUT ; : PRINT #0; PAPER 0; INK 7; BRIGHT 1, FLASH 1;" ДЛИНА="; L.
510 BEEP .1,20: PAUSE 400

```

```

600 PRINT AT 21,0;"ИМЯ ФАЙЛА ДЛЯ ЗАПИСИ: ": GO SUB 9: INPUT F$
610 SAVE F$ CODE 32730,L
700 RUN
900 DATA 221,33,0,64,17,0,27,62,255,55,205,86,5
910 DATA 33,0,64,1,0,27,17,0,128,213,27,19,120,177,40,40,126
920 DATA 237,160,190,32,248,13,12,32,4,5,4,40,26,18,19,8,62,0
930 DATA 60,40,229,16,35,11,13,12,32,4,5,4,40,8,8,190,32,214,8
940 DATA 24,235,27,26,47,19,18,235,209,175,237,32,34,229,127,1,38,0,175,237,74,229,193,3,201
950 DATA 205,124,0,59,59,225,1,35,0,9,1,0,0,17,0,64,120,177,200
960 DATA 126,237,160,190,32,247,36,197,70,18,19,16,252,193,35,11,24,234

```

Со строки 200 выводится меню программы. Возможны три варианта загрузки. Первый - обычная загрузка экрана, записанного с заголовком. Второй и третий - загрузка экрана, записанного без заголовка. Эти два режима могут оказаться довольно удобными, так как во многих играх встречаются блоки кодов - экраны, записанные без заголовка. Режимы загрузки экрана без заголовка отличаются между собой тем, что в одном случае для загрузки вызывается подпрограмма 2050 из ПЗУ, которая в случае ошибки выводит сообщение: "Tape loading error", а в другом случае используется подпрограмма 1366 ПЗУ, при этом работа БЕЙСИКа не прекращается независимо от правильности считывания.

Сразу же после окончания загрузки внизу экрана Вы увидите результат работы компрессирующей программы: длину скомпрессированного блока кодов. После нажатия любой клавиши программа предложит Вам ввести имя для записи полученного результата вместе с Программой декомпрессии в виде единого файла. После записи можете загружать скомпрессированный блок в произвольный адрес ADR. Вывод на экран - RANDOMIZE USR ADR.

### Другой вариант компрессии.

Продолжая разговор о компрессорах экрана, приведем еще одну программу.

Очень часто при разработке каких-либо (например обучающих) программ, приходится решать вопрос размещения в памяти компьютера большого количества графической информации, рисунков, схем и т. д. При этом во многих случаях графика занимает не весь экран, а скажем, только средний сегмент экрана. При этом верхний и нижний сегменты используются для других целей. Поэтому за счет компрессии не полного экрана, а лишь одного его сегмента, можно значительно сократить длину скомпрессированного блока кодов. Кроме того, при декомпрессии можно вызывать сегменты из памяти независимо один от другого, не нарушая соседние, правда в этом случае придется компрессирование проводить за два этапа, отдельно для дисплейной части и атрибутов, так как между вами появляется разрыв.

Декомпрессирующий блок является единственным для всех скомпрессированных экранов. При таком подходе в памяти можно получать десятки (а если небольшие, то может быть даже сотни) экранов. Перед запуском декомпрессирующей программы надо номер нужного экрана занести в системную ячейку декомпрессора при помощи POKE, а затем вызывать декодирующую процедуру.

В основе приводимой программы содержится известный многим пользователям "Спектрума" блок кодов "compress" CODE 28000,650. Но Бейсиковая часть сделана практически заново, что позволило пользоваться программой с большим удобством и удовольствием. Кроме того, мы приводим два варианта программы: для магнитофона и для дисковода.

Этот вариант компрессора является более изощренным, чем приведенные до сих пор программы. (Достаточно обратить внимание на длину его кодового блока). Однако эта длина сама по себе ни о чем не говорит. Ведь важна длина конечного продукта - скомпрессированного экрана плюс программы декомпрессии. Этот вариант компрессора позволяет получить более высокую степень компрессии, чем те, которые мы приводили до сих пор. Мы провели сравнительные испытания этого компрессора для тех же экранов.

|              |           |
|--------------|-----------|
| "Death Star" | 3185 байт |
| "Cauldron"   | 2128 байт |

|              |           |
|--------------|-----------|
| "Scooby Doo" | 3946 байт |
| "XENO"       | 3439 байт |
| "Saboteur"   | 2545 байт |

Кроме более высокой степени компрессии, этот вариант компрессора позволяет получить следующие режимы, недоступные другим программам:

Возможно компрессирование как всего экрана, так и произвольно взятых одного любого или двух соседних сегментов экрана. Программа анализирует расположение графической информации на экране и выбирает, в зависимости от этого, наиболее выгодный способ компрессирования. При этом во время декомпрессии Вы можете увидеть, что некоторые экраны разворачиваются сверху вниз, аналогично тому, как это происходит при загрузке. А другие экраны - разворачиваются слева направо, (что несомненно усиливает зрительный эффект). Правда работает эта программа медленнее, чем другие варианты компрессоров. Компрессия полного экрана занимает порядка 1-3 секунд (однако это время не имеет значения для конечного продукта).

Так как кодовый блок программы (мы его полностью приводим ниже) расположен с адреса 28000, при использовании программы с дисководом Вам может не хватить свободного места. Для этого в программе используются приемы по сохранению памяти. Мы уже приводили такие материалы в "РЕВЮ". Это, в частности, замена числа 0 на NOT PI, числа 1 - на SGN PI, использование вместо числа выражения: VAL "число". Без таких преобразований оставлены только строки программы, реализующие работу меню, чтобы не замедлять время выполнения этих операций. Все эти приемы могут оказаться излишними для магнитофонного варианта, но они обязательны для дискового. Текст БЕЙСИК-программы вначале приведен для магнитофонного варианта. Затем будут даны дополнительные строки, которые позволят Вам получить дисковый вариант программы.

Вот Бейсиковая часть программы. Она традиционно русифицирована по известной Вам схеме.

```

1 GO TO VAL "100"
2 BORDER VAL "7": PAPER VAL "7": INK NOT PI: CLEAR VAL "27995": LOAD "chr"CODE 64600
3 LOAD "compress"CODE 28000,650
4 GO TO NOT PI
5 SAVE "COMPRESS"LINE 2: SAVE "compress"CODE 28000, 650: STOP
8 POKE VAL "23606", VAL "86": POKE VAL "23607", VAL "251":RETURN
9 POKE VAL "23606",NOT PI: POKE VAL "23607",CODE "<": RETURN
30 BORDER 1: PAPER 0: INK 7: CLS : GO SUB 8: GO SUB 40
31 FOR M=1 TO NN: READ M$: PRINT INK 6;AT (Y0+(M-1)*DY),X0;M$: NEXT M: PRINT INK 4; AT Y1,
    3; "SPACE, DOWN, UP, ENTER, BREAK"
32 IF MM<1 THEN LET MM=NN
33 IF MM>NN THEN LET MM=1
34 PRINT AT (Y0+(MM-1)*DY),X0-DX: PAPER 7; INK 2; BRIGHT 1; OVER 1;S$: BEEP .03,2*MM+10
35 PAUSE 0: LET I=(INKEY$=" " OR 1 INKEY$= "6" OR CODE INKEY$=10)+(INKEY$="7" OR CODE
    INKEY$=11)*2+(INKEY$= "0" OR CODE INKEY$=12 OR CODE INKEY$=13)*3: GO TO (35+I)
37 PRINT INK 6;AT (Y0+(MM-1)*DY),X0-DX; OVER 1;S$: LET MM=MM-2*I+3: GO TO 32
38 FOR M=1 TO NN: PRINT PAPER 8; INK 8; BRIGHT 6; FLASH (M=MM); OVER (M=MM);AT (Y0+(M-
    1)*DY),X0-DX:S$: NEXT M: BEEP .1,36: PAUSE 10: RETURN
40 PRINT AT 2,9;"ЧИСЛО ЭКРАНОВ: ";SCR
42 PRINT AT 3,3;"КОМПРЕССИЯ СЕГМЕНТА : ";BEG;: IF END<>BEG THEN PRINT AT 3,21;"0B";AT
    3,26:"-";END
44 PRINT AT 4,6: "СВОБОДНОЙ ПАМЯТИ : ";65536-LAST
49 RETURN
50 BORDER VAL "7": PAPER VAL "7": INK NOT PI: CLS
52 PRINT AT VAL "21",NOT PI;"ИМЯ ФАЙЛА:": GO SUB VAL "9": INPUT F$: CLS
59 RETURN
100 LET BEG=SGN PI: LET END=VAL"3": LET SCR=NOT PI: LET FIRST=VAL "28350": LET LAST=VAL
    "28650": LET S=NOT PI
190 LET MM=SGN PI
200 DATA "ЗАГРУЗКА ЭКРАНА","РЕЖИМ КОМПРЕССИИ","ПРОСМОТР ЭКРАНА","УДАЛЕНИЕ ЭКРАНА","ЗАПИСЬ
    ФАЙЛА"
201 RESTORE VAL "200": LET NN=VAL "5": LET X0=VAL "8": LET V0=VAL "9": LET DY=SGN PI: LET
    DX=VAL "3": LET S$="
    ": LET Y1=VAL "19": GO SUB VAL "30"

```



```

210 IF SCR=NOT PI AND MM>=VAL "3" THEN PRINT #NOT PI; FLASH SGN PI;"          HET ЭКРАНОВ!
    ": BEEP SGN PI, NOT PI: PAUSE VAL "100": GO TO VAL "190"
220 GO TO VAL "1E3"*MM
300 DATA " BECЬ ЭКРАН","СЕГМЕНТЫ 1-2","СЕГМЕНТЫ 2-3"," СЕГМЕНТ 1"," СЕГМЕНТ 2"," СЕГМЕНТ
    3"
301 RESTORE VAL "300": LET NN=VAL "6": LET X0=VAL "10": LET Y0=VAL "8": LET DY=SGN PI: LET
    DX=VAL "3": LET S$="          ": LET Y1=VAL "19": GO SUB VAL "30": RETURN
1000 GO SUB VAL "50"
1010 LOAD F$ CODE 16384,6912
1030 POKE VAL "28005",INT (LAST/VAL "256"): POKE VAL "28004",LAST-VAL "256"*PEEK VAL "28005"
1040 POKE VAL "28006",BEG-SGN PI: POKE VAL "28007",END-(BEG-SGN PI)
1050 RANDOMIZE USR VAL "28010"
1060 LET LEN=PEEK VAL "28008"+VAL "256"*PEEK VAL "26009": LET LAST=LAST+LEN+SGN PI: LET
    SCR=SCR+SGN PI: LET S=SCR-SGN PI
1070 GO SUB VAL "8": PRINT #NOT PI; PAPER NOT PI; INK VAL "7"; BRIGHT SGN PI; FLASH SGN PI;"
    ДЛИНА="; LEN;" "
1080 PAUSE VAL "400"
1090 GO TO VAL "200"
2000 LET MM SGN PI: GO SUB VAL "300"
2010 RESTORE VAL "2000": FOR N=SGN PI TO MM: READ BEG: READ END:NEXT N
2020 GO TO VAL "190"
2100 DATA SGN PI,VAL "3",SGN PI, VAL "2",VAL "2", VAL "3",SGN PI,SGN PI,VAL "2",VAL "2",VAL
    "3",VAL "3"
3000 LET S=S*(S<SCR)+(S<=SCR)
3010 INPUT ("HOMEP ЭКРАНА (1-";SCR;"): ";S;" : "); LINE F$: IF F$<>"" THEN LET S=VAL F$
3020 IF S<SGN PI OR S>SCR THEN GO TO VAL "3010"
3030 BORDER VAL "7": PAPER VAL "7": INK NOT PI: CLS
3040 POKE VAL "28352",S: RANDOMIZE USR VAL "28350": PAUSE NOT PI
3050 GO TO VAL "200"
4000 IF SCR SGN PI THEN LET LAST=VAL "28650": GO TO VAL "4100"
4010 LET A=NOT PI: LET C=A: LET LEN=VAL "28649": FOR N=SGN PI TO SCR: LET LEN=LEN+A+C*VAL
    "256"+SGN PI: LET A=PEEK LEN: LET C=PEEK (LEN+SGN PI): NEXT N: LET LAST=LEN
4100 LET SCR=SCR-SGN PI: LET S=SCR-SGN PI
4110 GO TO VAL "200"
5000 GO SUB VAL "50"
5010 SAVE F$CODE 28350, LAST-FIRST
5030 VERIFY F$ CODE
5050 GO SUB VAL "8" : PRINT #NOT PI; INVERSE SGN PI; "    ЗАПИСЬ И ВЕРИФИКАЦИЯ - O.K.    "
5060 BEEP VAL ".1",VAL "26": BEEP VAL ".1",VAL "20": PAUSE VAL "100"
5070 GO TO VAL "190"

```

Программы, подобные этой, Вы уже встречали на страницах "РЕВЮ", поэтому не будем подробно останавливаться на каждой ее части. Перечислим только основные фрагменты:

Строка 2 - автостарт, загрузка кодовых блоков.

Строка 5 - самозапись программы на магнитную ленту.

Строки 8 и 9 - переключатели шрифтов, соответственно русского и латинского.

Строка 30 - подпрограмма работы меню. Она была подробно рассмотрена в "ZX-РЕВЮ" N 9-10 за 1992 г., стр. 197.

Строка 40 - вывод основных режимов и состояния компрессора. На экран выводится следующая информация: число скомпрессированных экранов, режим компрессии - какие сегменты заданы для компрессирования, количество оставшейся свободной памяти компьютера.

Строка 50 - ввод имени файла для загрузки и записи. Если при выполнении загрузки в ответ на запрос имени нажать ENTER, ничего не вводя, то загрузится первый встретившийся кодовый файл.

Строка 100 - инициализация.

Строка 200 - главное меню программы, задающее режимы работы компрессора.

Строка 300 - второе меню, задающее сегменты для компрессирования.

Строка 1000 - загрузка очередного экрана.

Строка 2000 - задание сегментов для режима компрессии (вызов второго меню).

Строка 3000 - просмотр скомпрессированных экранов. Здесь следует отметить организацию ввода номера просматриваемого экрана. При запросе предлагается вариант по соглашению. Если предложенное число Вас не устраивает, введите свое значение, если устраивает - просто нажмите ENTER. Предложение варианта по соглашению (строка 3000) организовано так, что после загрузки очередного экрана к просмотру предлагается этот последний экран. После его просмотра по соглашению в следующий раз будет предложен первый экран, после его просмотра - второй и так далее до последнего, затем опять первый и так далее. Так можно просмотреть все экраны, ничего не вводя, а можно в любой момент вмешаться в этот порядок и ввести свое значение.

Строка 4000 - удаление экрана. Надо ли здесь организовать запрос для подтверждения Вашего решения? Организуйте его сами, если хотите. Удаляется всегда только последний экран. Это может создавать некоторое неудобство. Хорошо, если бы можно было удалять любой из скомпрессированных экранов, но это потребует вмешательство в блок кодов. В общем-то ничего принципиально сложного здесь нет. Надо только рассчитать место удаляемого экрана, определить его длину и организовать при помощи машиннокодовой команды LDIR переброску следующих экранов на место удаляемого. Все, что необходимо для такого расчета, находится в строках с 4000. Может быть кто-нибудь из читателей и сделает такое усовершенствование.

Строка 5000 - запись всех скомпрессированных экранов вместе с программой декомпрессии в виде единого файла. Если при задании имени файла нажать ENTER, ничего не вводя, то программа вернется в режим основного меню (см. строку 5000).

При работе с БЕТА-диском, должны быть изменены строки, связанные с загрузкой и выгрузкой. Мы приводим ниже вариант строк для адаптации программы под дисковод. Вы можете набрать и хранить их отдельно, а при адаптации подгрузить к основной БЕЙСИК-программе при помощи MERGE.

```
2 BORDER VAL "7": PAPER VAL "7": INK BIN : CLEAR VAL "27999": RANDOMIZE USR VAL "15619": REM
  : LOAD "chr"CODE 64600
3 RANDOMIZE USR VAL "15619": REM : LOAD "compress"CODE 28000,650
5 GO SUB VAL "90": STOP
6 RANDOMIZE USR VAL "15616"
7 STOP
90 RANDOMIZE USR VAL "15619": REM : ERASE "COMPRESS"
92 RANDOMIZE USR VAL "15619": REM : SAVE "COMPRESS" LINE 2
99 RETURN
1000 GO SUB VAL "50": IF F$="" THEN GO TO VAL "200"
1010 LET ERR USR VAL "15619": REM : LOAD F$ CODE 16364,6912
1020 IF ERR<>NOT PI THEN GO TO VAL "1500"
1500 GO SUB VAL "8": PRINT AT VAL "10",VAL "9": FLASH SGN PI;" ОШИБКА ";ERR:" "" FLASH NOT
  PI"" ВЫВОДИТЬ КАТАЛОГ ДИСКА?(Y/N)"
1510 BEEP SGN PI, NOT PI: PAUSE NOT PI: IF INKEY$<>"y" AND INKEY$<>"Y" THEN GO TO VAL "200"
1520 GO SUB VAL "9": RANDOMIZE USR VAL "15619": REM : CAT
1530 PAUSE VAL "400"
1540 GO TO VAL "200"
5010 LET ERR=USR VAL "15619": REM : SAVE F$ CODE 28350, LAST-FIRST
5020 IF ERR<>NOT PI THEN GO TO VAL "1500"
5030 LET ERR=USR VAL "15619": REM : VERIFY F$ CODE
5040 IF ERR<>NOT PI THEN GO TO VAL "1500"
```

Строка 5 - как и раньше, самозапись программы (только Бейсиковой части) на диск. Она выполняется при помощи подпрограммы со строки 90, так как в одной пятой БЕЙСИК-строке не может выполняться более одного оператора TRDOS.

Строка 6 - переход в TR-DOS для выполнения каких-либо действий. После возврата оттуда по команде RETURN, попадаем на STOP в строке 7.

Строки с 1000 - загрузка экрана с диска. Здесь в случае ошибки при загрузке (например отсутствие файла на диске), происходит переход на строку 1500. Выведенный на экран код ошибки позволит Вам сориентироваться, почему произошла ошибка (коды ошибок см. в руководстве по TR DOS).

Строка 1500 предлагает вывести каталог диска, для того, чтобы Вы могли еще раз уточнить, есть ли нужный файл на этом диске или нет. На строку 1500 программа может перейти и из строк с 5000.

Строка 5000 - режим записи файла на диск, а также выполнение его верификации после записи.

Для тех читателей, которым не знаком кодовый блок программы "COMPRESS", мы приводим его полностью. (Тем, у кого он есть, можно его не набирать, а взять из имеющейся программы в готовом виде.) Для набора его Вы можете воспользоваться рекомендациями и программой для шестнадцатиричного ввода см. "ZX-РЕВЮ"-91, N 3, с. 59.

|       |                            |       |                            |
|-------|----------------------------|-------|----------------------------|
| 6D60: | 30 01 00 00 EA 6F 00 01:58 | 6EA8: | 00 00 00 00 00 00 00 00:16 |
| 6D68: | 8D 03 FD 21 60 6D FD 36:83 | 6EB0: | 00 00 00 00 00 00 00 00:1E |
| 6D70: | 03 01 CD 85 6D E5 FD 36:B8 | 6EB8: | 00 00 00 00 00 00 18 04:42 |
| 6D78: | 03 00 CD 85 6D D1 ED 52:B7 | 6EC0: | 01 FF FF 0C CD 7C 00 38:BD |
| 6D80: | D8 FD 36 03 01 FD 6E 04:6B | 6EC8: | 3B E3 33 33 A7 11 07 00:79 |
| 6D68: | KD 66 05 11 04 00 19 1E:A9 | 6ED0: | ED 52 E5 FD E1 11 2A 01:7C |
| 6D90: | 00 FD 7E 06 87 87 87 C6:D9 | 6ED8: | 19 FD 7E 00 3D 28 06 5E:A3 |
| 6D98: | 40 57 FD 77 08 1A 2F FD:5E | 6EE0: | 23 56 19 18 F7 11 00 40:40 |
| 6DA0: | 77 00 FD 36 01 01 FD 36:EC | 6EE8: | 23 23 FD 7E 01 3C 20 04:78 |
| 6DA8: | 02 00 FD 7E 03 18 74 FD:1E | 6EF0: | 7E FD 77 01 23 FD 7E 02:F1 |
| 6DB0: | 4E 00 1A FD 77 00 B9 28:DA | 6EF8: | 3C 20 04 7E FD 77 02 23:DD |
| 6DB8: | 0E FD CB 02 46 28 08 FD:70 | 6F00: | 7E FD 77 03 FD 7E 01 FE:DE |
| 6DC0: | CB 02 86 FD 36 01 01 A7:5C | 6F08: | 03 38 04 FD 36 01 00 FD:E7 |
| 6DC8: | 20 16 FD CB 02 46 20 10:AB | 6F10: | 7E 02 FE 04 38 04 FD 36:70 |
| 6DD0: | FD 7E 01 3D 23 04 23 71:B6 | 6F18: | 02 01 FD 86 01 FE 04 38:48 |
| 6DD8: | 18 F9 3C 23 0E 00 18 13:EE | 6F20: | 04 FD 36 02 01 FD 7E 02:46 |
| 6DE0: | B9 20 30 FD CB 02 46 20:86 | 6F28: | A7 20 04 FD 36 02 01 23:BB |
| 6DE8: | 15 FD 7E 01 3C FD 77 01:97 | 6F30: | 1E 00 FD 7E 01 87 87 87:CE |
| 6DF0: | FE 04 D8 36 00 23 77 23:2A | 6F38: | C6 40 57 4F 06 01 FD CB:22 |
| 6DF8: | 71 FD 36 02 01 C9 2B 34:34 | 6F40: | 03 46 20 02 06 20 C5 FD:02 |
| 6E00: | 08 23 FD 36 01 01 08 C0:96 | 6F48: | 46 02 C5 06 08 C5 06 08:A5 |
| 6E08: | 7E 2F FD 77 00 FD 36 02:CC | 6F50: | C5 06 01 FD CB 03 46 28:C4 |
| 6E10: | 00 C9 71 23 FD 35 01 20:2E | 6F58: | 02 06 20 D5 C5 18 43 C1:A5 |
| 6E18: | F9 77 FD 36 01 01 FD 36:5E | 6F60: | 13 10 F9 D1 C1 14 10 E8:89 |
| 6E20: | 02 00 C9 77 06 01 FD CB:9F | 6F68: | 7A D6 08 57 7B C6 20 5F:46 |
| 6E28: | 03 46 20 02 06 20 C5 FD:E9 | 6F70: | C1 10 DA 7A C6 08 57 C1:EA |
| 6E30: | 46 07 C5 06 08 C5 06 08:91 | 6F78: | 10 D0 51 1C C1 10 C7 1E:EA |
| 6E38: | C5 06 01 FD CB 03 46 28:AB | 6F80: | 00 3E 58 FD 86 01 57 FD:5D |
| 6E40: | 02 06 20 D5 CD AF 6D 13:A7 | 6F88: | 46 02 4B FD CB 03 DE C5:F8 |
| 6E48: | 10 FA D1 C1 14 10 E9 7A:D9 | 6F90: | 18 10 C1 0B 33 78 B1 20:4F |
| 6E50: | D6 08 57 7B C6 20 5F C1:74 | 6F98: | F6 FD 36 01 FF FD 36 02:65 |
| 6E58: | 10 DB 7A C6 00 57 C1 10:21 | 6FA0: | FF C9 FD CB 03 56 28 0D:2D |
| 6E60: | D1 FD 56 08 1C C1 10 C6:AD | 6FA8: | D9 05 79 D9 28 03 12 18:9C |
| 6E68: | 11 00 58 0E 01 FD 46 07:98 | 6FB0: | 18 FD CB 03 96 7E 23 A7:E0 |
| 6E70: | FD 7E 06 82 57 C5 CD AF:79 | 6FB8: | 20 0E 7E D9 47 D9 23 7E:6D |
| 6E78: | 6D C1 0B 13 78 B1 20 F5:70 | 6FC0: | D9 4F D9 23 FD CB 03 D6:F4 |
| 6E60: | FD 5E 04 FD 56 05 A7 ED:39 | 6FC8: | 12 FD CB 03 5E 20 C3 18:6D |
| 6E83: | 52 EB 73 23 72 FD 7E 06:BC | 6FD0: | 8E 08 10 11 12 12 14 00:2E |
| 6E90: | 23 77 23 FD 7E 07 77 EB:9F | 6FD8: | 03 00 01 02 01 00 03 00:51 |
| 6E98: | 22 68 6D C9 00 00 00 00:C6 | 6FE0: | 0F 19 0E 00 04 00 08 18:A9 |
| 6EA0: | 00 00 00 00 00 00 00 00:0E | 6FE8: | 22 22 00 00 00 00 00 00:9B |

## Профессиональный подход

### Методы Билла Гильберта.

По-видимому, среди программ, находящихся у пользователей, имеющих Синклер-совместимые компьютеры, ничто так не распространено, как программы, взломанные Биллом Гильбертом.

В этой статье Вашему вниманию будет предложена информация о специфике некоторых приемов оформления программ, применяемых этим хаккером. Несмотря на внешнюю схожесть эффектов, получающихся в ходе работы программ, Гильберт не всегда действует по одному шаблону. У него их достаточно много, хотя иногда они очень похожи и, разобравшись с одним из них, Вам не составит труда понять принципы действия остальных.

Как Вы уже вероятно знаете, Билл Гильберт всегда оставляет текстовые сообщения на взломанных им программах. Несмотря на то, что в последние годы он явно поскромнел, и большинство его сообщений можно перевести с английского, как "Дисковая версия Билла Гильберта" (по-английски: Disked by Bill Gilbert), в отличие от ранее использовавшегося автографа "Взломано Биллом Гильбертом" (по-английски: Cracked by Bill Gilbert), его методы остались теми же и основная цель, которую он преследовал вставкой подобных сообщений не изменилась. При изучении его "творчества" необходимо осознавать неэтичность действий Гильберта и относиться к этому так же, как и к тому, что некто на наших глазах роется в чужих карманах (хоть и делает это высокопрофессионально).

В то же время, опыт хаккера может быть использован при оформлении программ, совершенствовании их дизайна, что делает работу более простой, удобной и увлекательной.

При использовании приводимого нами материала необходимо учитывать, что большинство рассматриваемых программ написаны в машинных кодах. Это еще раз говорит о том, что высокого уровня профессионализма можно добиться только изучив программирование на языке Ассемблера. Несмотря на то, что большинство рассматриваемых процедур снабжены подробным комментарием, мы настоятельно рекомендуем Вам перед прочтением данного материала поближе познакомиться с программированием в машинных кодах. Одной из лучших книг, которые нам приходилось встречать по данной тематике, является методическая разработка "ИНФОРКОМа" "Практикум по программированию в машинных кодах".

Мы рекомендуем Вам для начала просто ознакомиться с текстом статьи, получить представление об основных приемах и методах, а потом уже детально изучать проблему по разделам. Мы старались подготовить информацию таким образом, чтобы она не требовала специального запоминания, а усваивалась постепенно по ходу внимательного прочтения.

Необходимо также учитывать, что данный материал тесно связан с той информацией, которая давалась в предыдущих статьях, особенно это касается методик взлома рассмотренных систем защиты. Впрочем, если Вы и не читали предыдущих статей, то не отчаивайтесь - нижеследующий текст отличается определенной автономией от предыдущего материала и не требует особой подготовки, интересен как для профессионалов, так и для начинающих. Желаем Вам удачи при изучении!

### Современные разработки Билла Гильберта.

Билл Гильберт работает над взломом программ уже много лет. За это время ему удалось накопить достаточно большой опыт в этих вопросах. Постоянно совершенствуя свою технику программирования он достиг того уровня, который мы видим в последних взломанных им программах.

Наиболее любопытными среди них являются Iron Man и Sim City. В этих программах использован оригинальный вывод текстовых сообщений о взломе, который свидетельствует о высоком профессиональном уровне программирования хаккера. В обеих программах используется совмещение программы на БЕЙСИКе с программой в машинных кодах, обе

эти программы схожи также и по структуре загрузки и по методике взлома, примененной хаккером. Это позволяет рассматривать обе программы как две оригинальные ветви одного типа взлома. В них можно найти очень много общего, в том числе: общие процедуры на Ассемблере, схожесть расположения процедур в адресном пространстве и многое другое. Но, в тоже время, существуют довольно разительные отличия, которые определяют необходимость индивидуального рассмотрения каждой программы. В первую очередь это касается методики вывода текстовых сообщений на экран и, во-вторых, методики создания оригинальных моделей шрифта.

Рассмотрим более подробно эти методы на обеих программах.

### Iron Man.

```
10 PAPER NOT PI
20 INK NOT PI
30 POKE VAL "23624",NOT PI
40 CLEAR VAL "24499"
50 OUT VAL "254",NOT PI
60 RANDOMIZE USR VAL "23789"
70 LOAD "IRNMAN1" CODE
80 LOAD "IRONMAN2" CODE
90 RANDOMIZE USR VAL "25E3"
100 REM NO POKE
110 RANDOMIZE USR VAL "24036"
```

Как видим, данная программа практически не имеет стройной системы защиты; такой вид она имела сразу же после того, как ее обработал Билл Гильберт. В строке 100 хаккер сам предупреждает нас, что в программе отсутствуют защитные POKES. Структура программы на БЕЙСИКе достаточно проста. Строки 10-20 делают цвет INK и PAPER черным, строка 50 делает черным и цвет бордюра. Как Вы могли заметить, в программе трижды используется запуск процедур в машинных кодах, но нас будет интересовать только самый первый запуск в строке 60. Именно благодаря ему создается оригинальное текстовое сообщение характеризующееся необычным сочетанием цветов, и использованием необычного шрифта. Рассмотрим подпрограмму в машинных кодах более подробно.

```
CALL N5D07
CALL N5D23
CALL N5D45
CALL N5D64
CALL N5D77
CALL N5DA1
LD HL,3D00
DEC H
LD (5C36),HL
RET
```

```
N5D07    LD HL,3D00
        LD DE,9C40
        PUSH DE
        LD BC,0300
N5D11    LD A,(HL)
        RRA
        OR (HL)
        LD (DE),A
        INC HL
        INC DE
        DEC BC
        LD A,B
        OR C
        JR NZ,N5D11
        POP DE
        DEC D
        LD (5C36),DE
        RET
```

|       |              |
|-------|--------------|
| N5D23 | LD HL, 5DD6  |
|       | LD B, 1E     |
| N5D28 | LD A, (HL)   |
|       | CP 0D        |
|       | JR Z, 5D30   |
|       | INC HL       |
|       | DJNZ N5D28   |
|       | LD A, 1E     |
|       | SUB B        |
|       | LD (5D40), A |
|       | LD B, A      |
|       | LD A, 20     |
|       | SUB B        |
|       | SRL A        |
|       | LD (5D41), A |
|       | RET          |
| N5D40 | XOR A        |
|       | CALL 1601    |
|       | LD A, 16     |
|       | RST 10       |
|       | XOR A        |
|       | RST 10       |
|       | LD A, (5D41) |
|       | RST 10       |
|       | LD A, 10     |
|       | RST 10       |
|       | XOR A        |
|       | RST 10       |
|       | LD A, (5D40) |
|       | LD B, A      |
|       | LD HL, 5DD6  |
| N5D5E | LD A, (HL)   |
|       | RST 10       |
|       | INC HL       |
|       | DJNZ N5D5E   |
|       | RET          |
| N5D64 | LD HL, 5CD0  |
|       | LD A, 16     |
|       | RST 10       |
|       | LD A, 01     |
|       | RST 10       |
|       | INC A        |
|       | RST 10       |
| N5D6F | LD A, (HL)   |
|       | CP 0D        |
|       | RET Z        |
|       | RST 10       |
|       | INC HL       |
|       | JR N5D6F     |
| N5D77 | XOR A        |
|       | LD (5D42), A |
|       | LD B, 07     |
| N5D7D | PUSH BC      |
|       | LD A, (5D42) |
|       | LD HL, 5AC0  |
|       | LD DE, 5AC1  |
|       | LD BC, 001F  |
|       | LD (HL), A   |
|       | LDIR         |
|       | CALL N5D9B   |
|       | LD A, (5D42) |
|       | INC A        |

|       |               |
|-------|---------------|
|       | LD (5D42), A  |
|       | POP BC        |
|       | DJNZ N5D7D    |
|       | RET           |
| N5D9B | LD B, 05      |
| N5D9D | HALT          |
|       | DJNZ N5D9D    |
|       | RET           |
| N5DA1 | LD IX, 0000   |
|       | LD (5D43), IX |
|       | LD HL, 5AF0   |
|       | LD BC, 0011   |
|       | LD D, 05      |
| N5DB1 | PUSH HL       |
|       | PUSH BC       |
|       | LD BC, (5D43) |
|       | PUSH BC       |
|       | INC BC        |
|       | LD (5D43), BC |
|       | POP BC        |
|       | PUSH HL       |
|       | SBC HL, BC    |
|       | LD (HL), D    |
|       | POP HL        |
|       | ADD HL, BC    |
|       | LD (HL), D    |
|       | POP BC        |
|       | POP HL        |
|       | HALT          |
|       | HALT          |
|       | HALT          |
|       | DEC BC        |
|       | LD A, B       |
|       | OR C          |
|       | JR NZ, N5DB1  |
|       | RET           |

Если Вы внимательно присмотритесь к программе в машинных кодах, то обнаружите, что текст ее составлен весьма профессионально. Программа на языке Ассемблера обычно характеризуется очень большим объемом. Чтобы облегчить работу с ней, некоторые логически законченные действия оформляются в виде отдельных процедур - подпрограмм, которые следуют одна за другой. Управление этими процедурами осуществляется из головной программы на языке Ассемблера, которая последовательно вызывает каждую процедуру, используя оператор CALL.

Внимательно присмотревшись к листингу программы, Вы обнаружите, что программа Билла Гильберта не исключение. В этом листинге очень четко просматривается головная программа управления и процедуры в машинных кодах.

Рассмотрим, что выполняют процедуры этой программы. Для этого проследим, какой визуальный эффект получается при работе программы в целом. После запуска программы экран становится черным, после этого на нижних строчках появляется надпись "IRON MAN" и постепенно высвечивается сообщение "DISKED BY BILL GILBERT 1991". После того, как последняя надпись высветилась полностью, все буквы начинают переливаться разными цветами. Особый колорит оформления достигается за счет использования в программе оригинального шрифта.

Как Вы видите, программа состоит из шести подпрограмм, каждая из которых выполняет определенную функцию для достижения визуального эффекта, описанного выше. Таким образом, одна из них формирует утолщенный шрифт, следующая затемняет экран, следующая за ней процедура ответственна за распечатку первого текстового сообщения, второе текстовое сообщение, как Вы помните, выводится на экран достаточно оригинальным способом - путем постепенного высвечивания информации от середины к

краям экрана, следующая процедура обеспечивает переливающуюся различными цветами окраску букв.

Однако, наибольший интерес для читателя, на наш взгляд, представляют не все процедуры, используемые Биллом Гильбертом, а лишь некоторые из них. В частности, формирование оригинального шрифта. Поэтому не будем здесь подробно рассматривать каждую процедуру программы в машинных кодах, а остановимся на рассмотрении лишь самых интересных из них, Оставленные за пределами нашего изучения процедуры достаточно просты и не нуждаются в подробном объяснении и комментарии.

Рассмотрим процедуру формирования утолщенного шрифта. Одним из недостатков знакогенератора Спектрума являются тонкие буквы - поэтому многие пользователи жалуются на нечеткость надписей, выводимых на экран. Самым практичным методом создания утолщенного шрифта можно считать моделирование нового знака, как наложение нормального знака и знака, сдвинутого на одну точку влево. Формирование нового шрифта обычно начинается перемещением встроенного знакогенератора "Спектрума" в свободную область оперативной памяти. Следующим этапом является изменение знакогенератора, находящегося в ОЗУ, в соответствии с нашими требованиями и завершающим этапом является переключение системной переменной, указывающей на место расположения нового знакогенератора, образовавшегося в ходе ваших переделок. Рассмотрим подпрограмму на Ассемблере, выполняющую данные функции.

```
N5D07      LD HL, 3D00
           LD DE, 9C40
           PUSH DE
           LD BC, 0300
N5D11      LD A, (HL)
           RRA
           OR (HL)
           LD (DE), A
           INC HL
           INC DE
           DEC BC
           LD A, B
           OR C
           JR NZ, N5D11
           POP DE
           DEC D
           LD (5C36), DE
           RET
```

В начале в регистр HL мы загружаем начало встроенного знакогенератора Спектрума - 3D00. После этого в регистр DE загружается адресе нового месторасположения шрифта, а в регистр BC мы заносим общую длину области знакогенератора. Далее начинается работа в цикле. Мы загружаем в аккумулятор то, что находится в ячейке памяти, на которую указывает регистр HL, то есть первый байт области знакогенератора. Следующий этап - это ротация аккумулятора вправо, причем бит 0 перемещается во флаг переноса, а флаг переноса - в бит 7. Более подробно эта операция показана на структурной схеме, приведенной ниже.

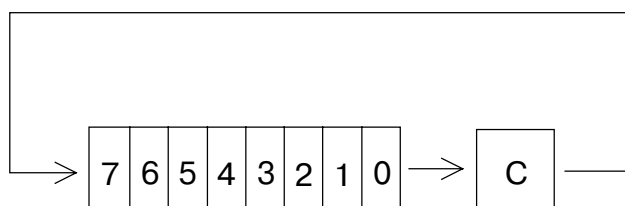


Схема 1.0 : C - флаг переноса.

Таким образом, мы получаем байт знакогенератора, который является смещенным вправо на один бит относительно первоначального байта. Теперь нам необходимо выполнить команду "ИЛИ" (OR) над образовавшимися байтами. Команда "ИЛИ" (OR) выполняется побитным сравнением двух двоичных чисел. Результат ее работы равен единице, если данный бит включен хотя бы в одном из операндов: в первом, или во втором,



или в обоих вместе. Таким образом, в результате может быть ноль только если в обоих операндах ноль, в противном случае получается единица.

Пример 1.0:

```
Первый операнд 1010 1010 (AA)
Второй операнд 1100 0000 (C0)
-----
Результат OR 1110 1010 (EA)
```

Продолжим рассмотрение программы, формирующей утолщенный шрифт. После того, как мы произвели операцию логическое "ИЛИ" над двумя байтами, мы загружаем образовавшийся байт в ячейку памяти, адрес которой находится в DE. Таким образом начинается формирование нового знакогенератора в свободной области оперативной памяти, поскольку именно в регистре DE содержится начало заданной для этих целей области ОЗУ.

Следующим этапом мы увеличиваем содержимое регистров HL и DE на единицу, т.е. переходим к следующему байту знакогенератора и записываем его измененный вариант в новое место оперативной памяти. В то же время, нам необходим контроль, чтобы модифицировать только байты знакогенератора и не больше. Для этой цели мы используем регистр BC, значение которого уменьшается на единицу после модернизаций очередного байта знакогенератора.

Следующим этапом программы является проверка достижения нуля в регистре BC. Если ноль не достигнут, то мы повторяем все операции сначала, только со следующим байтом шаблона символа. Однако, если в регистре BC остался ноль, это свидетельствует о том, что мы обработали всю область знакогенератора.

После этой проверки мы восстанавливаем со стека значение регистра DE, причем обратите внимание на то, что на стек засылалось значение регистра, которое соответствует месторасположению новой области знакогенератора в ОЗУ. Теперь нам достаточно уменьшить значение регистра D на единицу, и в регистре DE у нас окажется адрес, указывающий на 256 байтов ниже той области памяти, где расположен альтернативный знакогенератор, формируемый нами в ходе выполняемых операций. Теперь нам осталось занести это значение в область системных переменных, чтобы указать на месторасположение нового набора шаблонов символов. Последним шагом является возврат в головную программу в машинных кодах.

Как видим, ничего сложного в подобном преобразовании нет. При желании Вы можете использовать эту процедуру в своей работе. Для этого мы написали похожую процедуру в виде отдельной программы на БЕЙСИКе, запустив которую, Вы сможете подробно разобраться с работой генератора утолщенного шрифта.

```
10 INPUT "START ADRESS = ";ADRES
20 LET SUMMA=0
30 FOR N=ADRES TO ADRES + 68
40 READ BYTE:POKE N,BYTE
50 LET SUMMA = SUMMA + BYTE
60 NEXT N
65 IF SUMMA <> 6932 THEN STOP
100 DEF FN G(A$) = USR ADRES
110 PRINT AT 15,7;
115 LET ZM=FN G(IS IT GOOD ?)
120 PRINT AT 17,7;"AND NOW?"
200 DATA 42, 11, 92, 35, 35, 35, 35, 126, 92, 35, 126, 67, 35, 70, 235, 94
210 DATA 123, 254, 31, 216, 254, 127, 208, 22, 0, 197, 229, 33, 160, 0
220 DATA 237, 75, 123, 92, 9, 235, 41, 41, 41, 237, 75, 54, 92, 9, 6, 8, 126
230 DATA 203, 63, 128, 18, 19, 35, 16, 247, 62, 2, 205, 1, 22, 62, 164, 215
240 DATA 225, 35, 193, 16, 203, 201
```

Данная программа является перемещаемой: ее можно поместить по любому адресу.

Вызов : LET ZM = FN G ("ТЕКСТ")

Текст может содержать знаки с кодами от 32 до 127 включительно. Если присутствуют

другие символы, вывод на экран автоматически останавливается. Программа строит утолщенные знаки, как GRAPHICS "U", поэтому в своей программе нельзя переопределить этот символ, так как после выполнения команды вывода текста GRAPHICS "U" стирается.

На этом закончим рассмотрение того, как Билл Гильберт оформил программу "Iron Man" и теперь займемся программой "Sim City".

### Sim City

```
10 PAPER NOT PI
20 INK NOT PI
30 POKE VAL "23624",NOT PI
40 CLEAR VAL "24499"
50 OUT VAL "254",NOT PI
60 RANDOMIZE USR VAL "23791"
70 LOAD "SIMCITY1" CODE VAL "4E4"
80 CLS
90 RANDOMIZE USR VAL "4E4"
100 POKE VAL "23739",CODE "o"
110 LOAD "SIMCITY2" CODE
120 CLS
130 LOAD "SIMCITY3" CODE
140 RANDOMIZE USR VAL "24500"
150 REM NO POKE
160 RANDOMIZE USR VAL "20480"
```

Как видим, БЕЙСИК-загрузчик данной программы очень схож с аналогичным загрузчиком программы "IRON MAN". В нем практически отсутствуют какие-либо хитроумные операции направленные на защиту программы от просмотра. Билл Гильберт сам предупреждает нас о том что, программа практически не защищена.

Фактически эффект, возникающий в ходе работы программы в машинных кодах очень схож с аналогичным эффектом в программе "Iron Man". Однако, профессиональный программист сразу же обратит внимание на то, что вся информация выводится своеобразным шрифтом. Причем, если в предыдущей программе для вывода сообщения использовался обычный утолщенный шрифт, в данном случае мы имеем дело с неким подобием готического шрифта. Начинаящий программист может предположить, что этот шрифт загружается вместе с исходным текстом программы, однако дополнительные 768 байтов может себе позволить далеко не каждый загрузчик. В нашем примере Биллу Гильберту с помощью использования специальных команд процессора удалось сформировать готический шрифт из обычного шрифта "Спектрума". Кроме того, подпрограмма в машинных кодах, работающая в программе "Sim City", обладает целым рядом новшеств, делающих ее очень увлекательным.

Рассмотрим листинг процедуры в машинных кодах более подробно.

```
CALL N5D2D
CALL N5D93
CALL N5D74
CALL N5D4A
CALL N5D02
CALL N5D69
RET
;
N5D02 LD B,07
      XOR A
      LD (5D73),A
N5D08 LD HL,5AC0
      PUSH BC
      LD A,(5D73)
      INC A
      LD (5D73),A
      LD B,40
      PUSH BC
      LD A,(5D73)
      LD (HL),A
```

|       |               |
|-------|---------------|
|       | INC HL        |
|       | POP BC        |
|       | CALL N5D25    |
|       | DJNZ N5D08    |
|       | RET           |
| N5D25 | PUSH BC       |
|       | LD B, 05      |
| N5D2B | HALT          |
|       | DJNZ N5D28    |
|       | POP BC        |
|       | RET           |
| N5D2D | LD HL, 5DFA   |
|       | LD B, 1E      |
| N5D32 | LD A, (HL)    |
|       | CP 0D         |
|       | JR Z, N5D3A   |
|       | INC HL        |
|       | DJNZ N5D32    |
| N5D3A | LD A, 1E      |
|       | SUB B         |
|       | LD (5D72), A  |
|       | LD B, A       |
|       | LD A, 20      |
|       | SUB B         |
|       | SRL A         |
|       | LD (5D71), A  |
|       | RET           |
|       | ;             |
| N5D4A | XOR A         |
|       | CALL 1601     |
|       | LD A, 16      |
|       | RST 10        |
|       | XOR A         |
|       | RST 10        |
|       | LD A, (5D71)  |
|       | RST 10        |
|       | LD A, 10      |
|       | RST 10        |
|       | XOR A         |
|       | RST 10        |
|       | LD A, (5D72)  |
|       | LD B, A       |
| N5D63 | LD HL, 5DFA   |
|       | LD A, (HL)    |
|       | RST 10        |
|       | INC HL        |
|       | DJNZ N5D63    |
|       | RET           |
|       | ;             |
| N5D69 | LD HL, 3D00   |
|       | DEC H         |
|       | LD (5C36), HL |
|       | RET           |
|       | INC C         |
|       | EX AF, AF     |
|       | RLCA          |
|       | ;             |
| N5D74 | XOR A         |
|       | CALL 1601     |
|       | LD HL, 5CD0   |
|       | LD A, 16      |
|       | RST 10        |
|       | LD A, 01      |
|       | RST 10        |
|       | LD A, 02      |

|       |                |
|-------|----------------|
|       | RST 10         |
|       | LD A, 10       |
|       | RST 10         |
|       | XOR A          |
|       | RST 10         |
|       | LD B, 1C       |
| N5D8B | PUSH BC        |
|       | LD A, (HL)     |
|       | RST 10         |
|       | INC HL         |
|       | POP BC         |
|       | DJNZ 5D8B      |
|       | RET            |
|       | ;              |
| N5D93 | DI             |
|       | LD A, R        |
|       | LD E, A        |
|       | EX DE, HL      |
|       | SUB L          |
|       | LD L, A        |
|       | OR 40          |
|       | LD H, A        |
|       | SUB H          |
|       | PUSH HL        |
|       | PUSH HL        |
|       | POP DE         |
|       | INC A          |
|       | LD E, A        |
|       | RRA            |
|       | CPL            |
|       | LD C, A        |
|       | SCF            |
|       | RLA            |
|       | DAA            |
|       | RLA            |
|       | RES 4, (IY+01) |
|       | RLA            |
|       | LD A, (000E)   |
|       | INC A          |
|       | INC A          |
|       | LD B, A        |
|       | LD (HL), L     |
|       | LDIR           |
|       | POP HL         |
|       | LD A, H        |
|       | SUB 03         |
|       | LD H, A        |
|       | SUB H          |
|       | LD L, A        |
|       | XOR C8         |
|       | LD D, A        |
|       | LD E, L        |
|       | LD C, E        |
|       | LD B, L        |
|       | SET 0, B       |
|       | XOR C8         |
|       | SET 1A         |
|       | OR B           |
|       | LD B, A        |
|       | PUSH DE        |
|       | LDIR           |
|       | POP HL         |
|       | PUSH HL        |
|       | LD BC, (1A22)  |
|       | LD B, C        |

```

N5DD7      PUSH BC
           SUB A
           LD B, A
           SET 2, B
           INC HL
           INC HL
           INC HL
           INC HL
N5DE0      LD A, (HL)
           LD C, A
           SLA A
           OR C
           LD (HL), A
           INC HL
           DJNZ N5DE0
           POP BC
           DJNZ N5DD7
           POP DE
           EX DE, HL
           DEC H
           LD (5C36), HL
           EI
           RET

```

Данная программа на Ассемблере по своей структуре очень похожа на рассмотренную ранее в программе "Iron Man". Однако, это только внешнее сходство. Большинство процедур в машинных кодах выполнены по новому принципу. Однако, все они достаточно примитивны и мы надеемся, что читатель сам сможет разобраться с ними. Мы же остановимся лишь на процедуре создания готического шрифта. Она начинается с метки N5D93 (см. листинг выше).

При создании этой процедуры Билл Гильберт, очевидно, не задавался целью сделать ее доступной для понимания широкого круга читателей, скорее наоборот. Он запутал ее как сумел. Нам пришлось провести работу по расшифровке и полученной информацией мы готовы поделиться с читателем.

В этой подпрограмме в машинных кодах Билл Гильберт совместил выполнение двух функций: затемнение экрана (экран становится черным) и создание оригинального готического шрифта. Нас мало интересует процедура затемнения экрана, поэтому все свое внимание мы сконцентрируем на подпрограмме создания готического шрифта. В то же время, мы не можем рассматривать эти процедуры разрозненно, поскольку они тесно связаны между собой, и находятся в единой подпрограмме, которая вызывается второй по счету из головной программы в машинных кодах, с использованием оператора CALL.

Четкого разграничения между этими процедурами не существует, потому что после выполнения одной Биллу Гильберту понадобилось изменить содержимое всех регистров микропроцессора с целью правильного выполнения второй процедуры. В то же время, вместо того, чтобы напрямую занести содержимое чисел в регистр микропроцессора, Билл Гильберт начинает "заметать следы", используя ряд ухищрений с употреблением лишних в данной ситуации команд микропроцессора. Вместе с тем, к концу махинаций в регистрах оказываются именно те значения, которые необходимы для правильной работы процедуры, создавшей готический шрифт. Это Вы можете легко проверить, используя дизассемблер со встроенной процедурой отладки.

Фактически подпрограмма создания готического шрифта берет свое начало с метки N5DD7. Именно относительно этой метки создается цикл, который преобразует весь встроенный шрифт "Спектрума" в оригинальный готический шрифт. Все это делается тем же способом, как и в программе "IRON MAN". К началу работы процедуры в регистре BC находится значение 0300H, что соответствует длине области встроенного в компьютер шрифта. Регистр HL несет в себе значение 3D00H, что соответствует началу этой области. В регистре DE содержится значение C800H (десятиричный эквивалент - 51200). Это адрес в оперативной памяти, куда осуществляется перенос нового шрифта.

Если рассматривать любой символ шрифта как знакоместо 8X8, то мы можем обнаружить, что в предыдущей программе создания утолщенного шрифта образующийся символ создавался за счет смещения всего исходного символа вправо с сохранением исходного символа на своем месте. В этой программе мы смещаем не полное знакоместо, а лишь последние четыре байта, к тому же смещение этих байтов осуществляется не вправо, а влево, что наглядно демонстрируют приведенные ниже диаграммы.

|  |   |   |   |   |   |  |  |
|--|---|---|---|---|---|--|--|
|  |   |   |   |   |   |  |  |
|  |   | X | X | X |   |  |  |
|  | X |   |   |   | X |  |  |
|  | X |   |   |   | X |  |  |
|  | X |   |   |   | X |  |  |
|  | X |   |   |   | X |  |  |
|  |   | X | X | X |   |  |  |
|  |   |   |   |   |   |  |  |

Диаг. 1. Стандартный символ "О" из области знакогенератора "Спектрума".

|  |  |   |   |   |   |   |  |
|--|--|---|---|---|---|---|--|
|  |  |   |   |   |   |   |  |
|  |  |   | X | X | X |   |  |
|  |  | X |   |   |   | X |  |
|  |  | X |   |   |   | X |  |
|  |  | X |   |   |   | X |  |
|  |  | X |   |   |   | X |  |
|  |  |   | X | X | X |   |  |
|  |  |   |   |   |   |   |  |

Диаг. 2. Изображение смещенной буквы "О", образующееся после сдвига вправо всех байтов шаблона 8X8. Подобный принцип сдвига реализован в программе "Iron Man".

|  |   |   |   |   |   |   |  |
|--|---|---|---|---|---|---|--|
|  |   |   |   |   |   |   |  |
|  |   | X | X | X | X |   |  |
|  | X | X |   |   | X | X |  |
|  | X | X |   |   | X | X |  |
|  | X | X |   |   | X | X |  |
|  | X | X |   |   | X | X |  |
|  |   | X | X | X | X |   |  |
|  |   |   |   |   |   |   |  |

Диаг. 3. Утолщенная буква "О", образующаяся в результате операции "ИЛИ" над стандартным символом "О" и смещенным символом "О".

|   |   |   |   |   |   |  |  |
|---|---|---|---|---|---|--|--|
|   |   |   |   |   |   |  |  |
|   |   | X | X | X |   |  |  |
|   | X |   |   |   | X |  |  |
|   | X |   |   |   | X |  |  |
| X |   |   |   |   | X |  |  |
| X |   |   |   |   | X |  |  |
|   | X | X | X |   |   |  |  |
|   |   |   |   |   |   |  |  |

Диаг. 4. Символ "О", в котором нижние четыре байта смещены влево операцией SLA.

|   |   |   |   |   |   |  |  |
|---|---|---|---|---|---|--|--|
|   |   |   |   |   |   |  |  |
|   |   | X | X | X |   |  |  |
|   | X |   |   |   | X |  |  |
|   | X |   |   |   | X |  |  |
| X | X |   |   | X | X |  |  |
| X | X |   |   | X | X |  |  |
|   | X | X | X | X |   |  |  |
|   |   |   |   |   |   |  |  |

Диаг. 5. Символ "O", образующийся в результате операции "ИЛИ" над стандартным символом "O" и символом, изображенным на диаг. 4. Принцип используется Биллом Гильбертом в программе "Sim City".

Теперь рассмотрим, каким образом это реализуются в процедуре, написанной в машинных кодах. Перед входом в процедуру мы сохраняем на стеке значение регистра DE, которое, как Вы помните, указывает на место расположения нового шрифта в оперативной памяти компьютера. После этого мы сохраняем на стеке значение регистра BC, которое указывает на номер обрабатываемого символа из таблицы стандартного шрифта. Далее следует очистка аккумулятора, после чего мы очищаем содержимое регистра B, засылая в него содержимое аккумулятора, равное 0. Затем "зажигается" второй бит регистра B. Таким образом Билл Гильберт записывает в регистр в число 4. Это достаточно простой прием, который направлен на то, чтобы запутать начинающего программиста. В данном случае вместо одной команды:

```
LD B, 4
```

Гильберт использует целых три, причем результат работы в обоих случаях один и тот же.

На следующем этапе компьютер начинает анализировать стандартный шрифт. Первые четыре байта остаются неизменными, остальные четыре байта нам необходимо сдвинуть на один бит влево и осуществить операцию логического "или" с их исходными аналогами. Делается это следующим образом. Для начала в аккумулятор заносится значение, содержащееся в ячейке памяти, на которую указывает регистр HL. После этого значение регистра A копируется в регистр C. Далее над содержимым регистра A производится команда арифметического сдвига влево: по этой команде все биты сдвигаются. В бит 0 (младший) засылается 0. Бит 7 сдвигается во флаг переноса C. Операция эквивалентна умножению байта на 2. Если при этом образуется число большее, чем 255, включается флаг переноса C. Более подробно эта операция показана на структурной схеме.

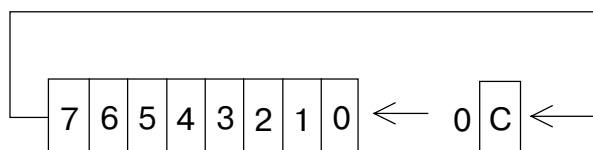


Схема 2.0 : C - флаг переноса.

После выполнения логического сдвига, мы выполняем функцию "ИЛИ" над содержимым регистров A и C. После этого новое значение сохраняем в памяти в области нового шрифта. Далее содержимое регистра HL увеличивается на единицу, чтобы он указывал на значение следующего байта из области стандартного шрифта.

Поскольку в регистре B содержалось 4, а следующей командой идет оператор DJNZ, который задает цикл, мы начинаем повторять все предыдущие операции, пока не очистится регистр B.

Следующим этапом нашей работы является восстановление со стека содержимого регистра DE, который, как Вы помните, указывал на место расположения сформированного нами шрифта. Для того чтобы поместить в соответствующую системную переменную указание на месторасположения данного шрифта, нам необходимо, чтобы в регистре HL содержалось значение на 256 байтов меньше, чем действительный адрес

месторасположения шрифта в ОЗУ. Это достигается уменьшением на единицу старшего регистра пары HL, то есть регистра H. После необходимых изменений мы заносим содержимое этого регистра в системную переменную, указывающую на месторасположение действующего в данный момент шрифта "Спектрума".

В заключение мы восстанавливаем прерывания и возвращаемся в вызвавшую нас процедуру с помощью команды RET.



# SINCLAIR LOGO

"ИНФОРКОМ" всегда уделял большое внимание пропаганде языков программирования для пользователей "ZX-Spectrum". Только на страницах "ZX-РЕВЮ" уже были опубликованы "MEGA BASIC" и три версии языка "BETA BASIC". Не прекращаются наши публикации, посвященные АСЦЕМБЛЕРу. Еще ранее, в 1989 - 1990 г.г. в виде брошюр мы выпустили методички, посвященные языкам "LASER BASIC", ПАСКАЛЬ (HP4T) и "ZX-FORTH".

Сегодня мы начинаем публикацию языка программирования "ЛОГО". Об этом языке написано уже немало. Имеется достаточный выбор отечественной литературы и мы не стали бы повторяться, если бы не одно маленькое обстоятельство. Дело в том, что доступные книги по "ЛОГО" никак не привязаны к компьютеру "Спектрум". Мы же предлагаем Вашему вниманию книгу, которая так и называется "LOGO on the SINCLAIR SPECTRUM". Таким образом, все, что Вы почерпнете из этой серии статей, вполне может быть использовано Вами в работе с вашим компьютером. Дело остается за малым - найти и запустить программу и освоить новый язык. Не знаем, как обстоит дело с этой программой на наших рынках сейчас, но в конце 80-х она была распространена очень широко и достать ее не было никаких проблем. Автор книги - известный английский программист и разработчик обучающего программного обеспечения Грэхем Филд. Авторизованный перевод с английского языка и редакция - наши.

## 1. Что такое "ЛОГО".

ЛОГО - это язык программирования, начало разработки которого было положено в Массачусетском Технологическом институте в 1967 году.

В его основе лежит концепция подготовки языка, удобного как для подготовки детей к основам компьютерной грамотности, так и для решения математических задач и задач, относящихся к области искусственного интеллекта. Использование этого языка программирования для этих целей стало возможным в результате значительного объема исследований, проведенных как в МТИ, так и в других научных и учебных центрах США и Европы, в частности, в Эдинбургском университете.

Уже первый опыт использования этого языка показал его высокую эффективность в обучении детей, а также особый интерес к нему не только со стороны учащихся, но и со стороны родителей и педагогов. Вместе с тем, ЛОГО не следует рассматривать только с этой стороны. Это не просто игрушка и в опытных руках он может быть серьезным инструментом. У ЛОГО есть ряд значительных преимуществ по сравнению, например с БЕЙСИКом и он может быть вполне использован в качестве универсального средства решения повседневных задач на компьютерах бытового класса. Итак, несмотря на то, что ЛОГО прост в освоении детьми и лицами, не имеющими никакой компьютерной подготовки, он является мощным инструментальным средством, обеспечивающим удобный доступ к ресурсам компьютера.

## "Черепашка".

Для того, чтобы проще приобщить детей к ЛОГО, было изобретено специальное механическое устройство - "Черепашка". Оно представляет собой маленькую механическую тележку с электродвигателем, подключенную к компьютеру с помощью проводов. Тележка может кататься по полу, по столу и т.п., но как правило ее запускают по большому листу бумаги. На тележке установлен держатель, в котором закрепляется перо, карандаш, фломастер. Перо может подниматься и опускаться по командам от компьютера. В случае перемещения "черепашки" по бумаге с опущенным пером, вычерчивается линия, соответствующая траектории движения. Для управления "черепашкой" служит программа, написанная на ЛОГО.

В принципе, "Спектрум" тоже может управлять подобной "черепашкой", как и многими другими электромеханическими устройствами, но мы от нее отвлечемся и будем думать о ней, имея дело с курсором на экране телевизора, с помощью которого будем строить

различные простейшие геометрические фигуры, за которыми так и закрепилось в литературе название "черепашья графика".

Вся прелесть (и мощь) ЛОГО состоит в том, что если Вы один раз "объяснили" компьютеру как надо исполнять ту или иную задачу, Вы можете больше никогда ему этого не "объяснять". Для этого надо Вашему объяснению присвоить какое-либо имя. После этого данная задача становится для компьютера известной ему процедурой и впоследствии, когда Вам опять потребуется исполнить такую же задачу, достаточно только назвать имя нужной процедуры. Получается так, что по мере того, как в своем понимании программирования развивается Вы, также параллельно вместе с Вами развивается и Ваш компьютер. Единственным ограничением при этом является размер памяти Вашей машины, но у Вас есть возможность сохранять готовые процедуры на магнитной ленте и загружать в память только те, которые нужны для решения текущей проблемы.

ЛОГО был разработан специалистами, основным направлением исследований которых является искусственный интеллект, главная проблема которого состоит в изучении приемов мышления человека и введении аналогичных приемов в компьютер. Это исследование привело в свое время к созданию языка программирования ЛИСП, основная задача которого состоит в обработке различных списков данных. К сожалению, оказалось, что этот язык труден для усвоения. Может быть он и не столь труден, как это кажется начинающим, но тем не менее он по крайней мере неудобен хотя бы тем, что содержит в своей основе многие специфические термины, которые относились к конкретным особенностям той ЭВМ, на которой он создавался и которая уже давным давно не существует. ЛОГО родился, как изящная альтернатива, сочетающая методический подход ЛИСПа с доступностью терминологии и естественной наглядностью.

К сожалению, по отношению к ЛОГО сложилось обывательское негативное представление о том, что "черепашья графика" - это основной аспект языка. Есть даже некоторые несведущие люди, убежденные в том, что это единственная область его применения. В последние годы даже появились некоторые программы, в той числе и высококласные, которые занимаются "черепашьей графикой" и потому содержат слово "ЛОГО" в своих названиях. "СИНКЛЕР ЛОГО" к ним не относится. Эта версия языка для компьютера "Спектрум" содержит все возможности оригинала и даже добавляет некоторые (например обращение к подпрограммам в машинных кодах), которые связаны со спецификой компьютера и не содержатся в эталонном языке, но могут быть очень полезны авторам неординарных программ.

Как и другие языки программирования, ЛОГО имеет немало специфических терминов. Многие не любят новой терминологии и боятся ее во всех областях, кроме той области, в которой сами являются специалистами. Но, тем не менее, некоторые термины все-таки необходимы, если мы хотим серьезно обсуждать возможности нового языка и смотреть, как они соотносятся с другими языками. Вы ничуть не хуже будете водить машину, если не будете знать, что те четыре штуковины, которые торчат из двигателя, называются свечами зажигания. Но это знание немного улучшит ваше положение, когда в один прекрасный день Ваш автомобиль не заведется. Оно поможет понять что произошло, когда кто-то более опытный объяснит Вам ситуацию и пригодится при покупке запчастей.

Эта книга не перенасыщена терминологией и не шокирует начинающего читателя. Все необходимые технические термины будут разбираться по мере их появления на простых и понятных примерах.

## **ЛОГО и БЕЙСИК**

Для тех, кто уже знаком с БЕЙСИКом, мы дадим краткое сравнение этих языков. Если же БЕЙСИК Вам еще неизвестен, то этот раздел Вы можете просто пропустить.

Эти языки во многом различаются. Наверное наиболее очевидным отличием, сразу бросающимся в глаза, является то, что в ЛОГО нет номеров строк в программе. Они просто не нужны. Если Вам нужно внести изменения в часть программы, написанной на ЛОГО, Вы должны воспользоваться программой-редактором, которая является частью ЛОГО-системы. Этот редактор отличается от того строчного редактора, с который Вы имеете дело

на "Спектруме". С его помощью можно обрабатывать не одну программную строку, а сразу целую процедуру или даже несколько. И этот редактор намного более удобен, чем тот, с которым Вы работаете в стандартном БЕЙСИКе.

Процедуры ЛОГО обычно короче и даже намного короче, чем программы БЕЙСИКа. Благодаря этому их легче писать, проще отлаживать и легче понимать, чем БЕЙСИК-программу. Редактор может одновременно обрабатывать большее количество строк, чем помещается на одном экране, но в ЛОГО не считается хорошей практикой делать процедуры столь длинными.

Еще одно необычное отличие состоит в том, что если Вы привыкли к тому, что в памяти компьютера может содержаться одновременно только одна БЕЙСИК-программа, то в ЛОГО одновременно в памяти могут быть несколько совершенно не связанных друг с другом процедур.

Еще одно, незаметное на первый взгляд отличие - то, что каждый оператор БЕЙСИКа имеет собственную структуру и всегда должен быть записан только в соответствии с ней. ЛОГО же, с другой стороны, имеет только одну допустимую структуру для своих операторов и две небольших вариации, которые не обязательны, а существуют только для большего удобства программиста.

Если Вы привыкли к тому, что операторы и функции БЕЙСИКа вызываются одним нажатием клавиши, то может быть будете несколько разочарованы, поскольку в ЛОГО этого нет. Это не позволяет сделать сама структура языка. Вам придется набирать все слова полностью по буквам. Впрочем, многие из стандартных инструкций ЛОГО имеют сокращения и к тому же в ЛОГО есть возможность самому задать сокращения для прочих слов, если Вам это хочется.

Есть еще много других отличий, о которых имеет смысл говорить после освоения обоих языков, но мы на них сейчас не будем останавливаться. Сказанного выше уже достаточно для того, чтобы Вы поняли, что ЛОГО - это совершенно особый язык программирования, не имеющий к БЕЙСИКу никакого отношения.

### **Диалекты ЛОГО.**

Если Вы сравните те конструкции, которые увидите в этой книге, с другими книгами по ЛОГО, то найдете очень значительные отличия. Так происходит потому, что ЛОГО, по видимому, имеет большее количество диалектов, чем любые другие языки. Это, конечно, неприятно, но на самом деле ЛОГО это ведь не столько язык программирования, сколько необычный и весьма продуктивный образ мышления. Во всяком случае, основные структуры, как и возможности того или иного диалекта, будут как правило похожими. Разобравшись, Вы сможете конвертировать другие программы, написанные на ЛОГО для других компьютеров в форму, приемлемую для "Спектрума". Слова могут отличаться, но стояние за ними идеи будут теми же.

### **Как пользоваться этой книгой.**

При написании программ на ЛОГО общепринято пользоваться прописными (заглавными) буквами и делать отступ в конструкциях языка. В этой книге будет использован тот же подход, но Вы можете им и не пользоваться, поскольку Синклер-ЛОГО - достаточно гибкий диалект. В нем можно без проблем использовать и строчные буквы, а отступ в строках - это просто удобная форма представления и чтения процедур, которая для Вас не обязательна.

Так что, если Вы наберете что-то вроде:

```
TO SAYHELLO  
PRINT [ДОБРО ПОЖАЛОВАТЬ В СИНКЛЕР-ЛОГО]  
END
```

то на экране компьютера может получиться например так:

```
?to sayhello  
>print [ДОБРО ПОЖАЛОВАТЬ В СИ!  
>НКЛЕР-ЛОГО)  
>end
```

Символы ? и > в начале строк и символ ! в конце - это специфические символы языка и

в свое время мы с ними разберемся.

При работе с книгой желательно попробовать приведенные в ней примеры на своем компьютере, чтобы почувствовать, как ЛОГО реагирует на те или иные команды и на ошибки. Тогда Вы научитесь использовать возможности языка наиболее эффективно. Все примеры в книге, за исключением самых примитивных, преднамеренно сделаны так, что их можно развивать и дорабатывать. Идеи о том, что еще можно доработать во многих случаях даны. В книге нет контрольных вопросов, да они и не нужны. Если в ЛОГО написанная Вами процедура работает и делает то, что должна делать, она написана верно.

## 2. Первые шаги.

В отличие от БЕЙСИКа, который уже встроен в ПЗУ Вашего "Спектрума", ЛОГО следует загружать с магнитной ленты. Это делается обычным способом:

```
LOAD ""
```

После загрузки на экране появится сообщение:

```
WELCOME TO SINCLAIR LOGO
```

Ниже Вы увидите вопросительный знак, служащий запросом к Вам и сигналом о том, ЛОГО ждет ваших команд. Рядом виден мигающий квадрат - это курсор. Кроме этого, Вы можете увидеть в правом нижнем углу символ "L", который означает, что от Вас ожидается ввод буквы или цифры.

Мы начнем наши эксперименты с "черепашьей графики". Наберите SHOWTURTLE и нажмите ENTER. Набор можно сделать и строчными буквами. Вы увидите, что на экране появится "черепашка" в виде стрелки. Одновременно экран разделится, оставив две нижние строки для ввода Ваших команд. Основная часть экрана будет использована для изображения результатов движений "черепашки".

Попробуйте дать команду

```
FORWARD 50
```

Вы увидите, что стрелка переместилась в том направлении, в котором она указывает. Здесь надо обратить внимание на то, что слова, из которых состоят команды языка ЛОГО должны разделяться пробелами. Так что, если Вы попытаете дать команду FORWARD50, то увидите, что ЛОГО Вас не поймет. Попробуйте это, когда появится сообщение об ошибке. Вы увидите изображение стрелки в правом нижнем углу экрана. Это означает, что Вам нужно нажать какую-либо клавишу, чтобы продолжить работу.

Теперь пойдем дальше:

```
BACK 50
```

Стрелка на экране вернется назад. Число 50 задает расстояние, на которое должна перенестись "черепашка". Слова FORWARD и BACK - это процедуры, а число 50 - параметр. Трудно сказать, в каких единицах измеряется расстояние перемещения черепашки, поскольку это зависит от размеров экрана Вашего телевизора или монитора, но во всяком случае максимальное количество шагов по вертикали -175, а по горизонтали - 255. Мы можем считать, что расстояние измеряется в пикселах.

Попробуйте подвигаться вперед-назад, привыкая к чувству расстояния и, когда Вам надоест двигаться по одной прямой, мы начнем осваивать новые направления. Сначала наберите CLEARSCREEN. Если Вы немного знаете английский, то уже очевидно догадались, что эта процедура очищает экран и при этом выставляет "черепашку" в исходную позицию.

Чтобы повернуть "черепашку", можно воспользоваться процедурами RIGHT (направо) и LEFT (налево). Эти процедуры должны иметь при себе параметр - угол, на который следует повернуться. Угол измеряется в градусах, полная окружность - 360 градусов, поэтому команда RIGHT 180 развернет "черепашку" в противоположном направлении, а команда LEFT 90 - повернет ее на прямой угол.

Попробуйте теперь:

```
FORWARD 50
```

```
RIGHT 120
```

```
FORWARD 50
```

```
RIGHT 120
```

```
FORWARD 50
```

Вы увидите, что "черепашка" пройдет по замкнутому контуру и встанет в той точке, с которой начинала. Справедливости ради надо сказать, что ее положение не строго идентично исходному, поскольку смотрит она не туда. Но если Вы добавите еще одну команду RIGHT 120, то она встанет точно в исходное положение.

Может быть, Вы сообразили, что это простой и изящный метод рисования треугольников? Да, наверное, это так. А как быть с пятиугольниками или, скажем, с десятиугольниками? Не кажется ли Вам, что рисовать фигуру с десятью сторонами и десятью углами таким приемом было бы несколько утомительно? К счастью, ЛОГО имеет простое средство избавиться от этой неприятности. Опять исполните CLEARSCREEN и попробуйте такую команду:

```
REPEAT 3(FORWARD 50 RIGHT 120)
```

Здесь используются квадратные скобки, просьба не путать с круглыми. В ЛОГО разные виды скобок используются для разных целей.

Процедура REPEAT означает, что то, что стоит в квадратных скобках, должно быть повторено. Параметр 3 указывает на количество повторений.

Команды REPEAT могут быть довольно длинными. Для того, чтобы их было удобнее читать, строку разбивают в тех местах, где удобно и продолжение строки печатают с отступом.

Если при написании команды она дойдет у Вас до правого края экрана и начнется на новой строке не пугайтесь, этот разрыв не страшен. ЛОГО не считает строку законченной до тех пор, пока не нажата клавиша ENTER. В качестве указателя того, что команда продолжается на новой строке компьютер поставит в конце экранной строки восклицательный знак "!", служащий здесь как бы символом переноса. Поэтому, если в этой книге Вам встретится например такая команда:

```
REPEAT 8 (FORWARD 20 RIGHT 90  
          FORWARD 10 RIGHT 90  
          FORWARD 20 LEFT 15]
```

Смело набирайте ее на экране в одну строку, давая ENTER только тогда, когда будет введено все до конца.

Но мощь ЛОГО, конечно же заключена не в способности многократно повторять одни и те же действия, а в способности создавать новые команды. Поскольку Вы теперь знаете, как изображается треугольник, Вы можете теперь создать команду для его изображения.

Сначала напечатайте:

```
TEXTSCREEN
```

По этой команде "черепашка" будет удалена с экрана и все 22 строки экрана будут доступны для написания текста. А теперь делайте так:

```
TO TRIANGLE
```

(появится приглашение ">")

```
REPEAT 3 [FORWARD 50 RIGHT 120]
```

На сей раз команда не будет выполнена непосредственно, поскольку оператор TO... свидетельствует о том, что Вы не рисуете треугольник в данный момент, а занимаетесь тем, что объясняете компьютеру, как это делается. Вместо этого появляется новое приглашение ">", оно означает, что Вы не закончили описание новой команды. Чтобы завершить дело, нужно набрать еще один оператор:

```
END
```

Компьютер ответит Вам на это следующим сообщением:

```
TRIANGLE defined
```

Это означает, что процедура для изображения треугольников под именем TRIANGLE определена и теперь она известна компьютеру.

После этого на экране появится приглашение ">". Оно означает, что ЛОГО находится в командном режиме и ждет от Вас ввода команды. Перейдите к "черепашке" командой SHOWTURTLE и дайте команду TRIANGLE. Если все было сделано правильно, треугольник будет нарисован. Отныне команда TRIANGLE не менее знакома компьютеру, чем команды FORWARD и BACK.

Более того, теперь Вы можете встраивать команду TRIANGLE в другие, еще более мощные команды. Попробуйте, например:

```
REPEAT 12 [TRIANGLE RIGHT 30]
```

- и посмотрите, что будет нарисовано на экране. Попробуйте также самостоятельно оформить эту команду в процедуру и дайте ей имя по вкусу.

Развивая мысль, мы с Вами попробуем нарисовать человечка. Он (или она) будет состоять из квадратной головы, туловища, которое будет изображено одной прямой линией, пары прямых рук и пары прямых ног (см. рис. 1).

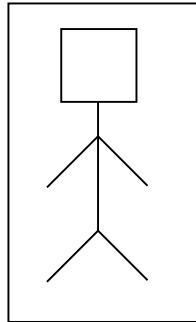


Рис. 1.

Программа выглядит очень просто:

```
TO MAN
  HEAD      (голова)
  BODY      (тело)
  LEGS      (ноги)
  ARMS      (руки)
END
```

Можете набрать эту процедуру. Пусть пока Вас не смущает, что ни мы, ни компьютер не знаем пока определения процедур HEAD, BODY, LEGS и ARMS. Мы их еще напишем. Если при их написании Вам захочется посмотреть, как они работают, просто перейдите в командный режим (курсор "?") и дайте имя команды.

Итак, "голову" мы изобразим, как прямоугольник и процедура очень похожа на ту, которая изображала треугольники. Но нам надо еще после всего поместить "черепашку" в центр нижней стороны и развернуть ее вниз, чтобы она была готова перейти к рисованию "тела". Для этого потребуются дополнительно три команды. В целом процедура HEAD будет выглядеть так:

```
TO HEAD
  REPEAT 4 [FORWARD 20 RIGHT 90]
  RIGHT 90
  FORWARD 10
  RIGHT 90
END
```

"Тело" еще проще:

```
TO BODY
  FORWARD 40
END
```

Для изображения "ног" развернем слегка "черепашку" влево, переместим вперед, вернем назад, повернем вправо на вдвое больший угол, чем в начале, опять вперед и назад и снова развернем влево, возвратив в первоначальное положение.

```
TO LEGS
  LEFT 30
  FORWARD 30
  BACK 30
  RIGHT 60
  FORWARD 30
  BACK 30
  LEFT 30
END
```

Если Вы посмотрите на рисунок, то увидите, что "руки" ничем принципиально от "ног" не отличаются, поэтому для изображения "рук" достаточно отогнуть "черепашку" назад и пририсовать к "телу" пару "ног" на уровне "плеч".

```
TO ARMS
  BACK 35
```

LEGS  
END

Обратите внимание на саму методику работы. Вы видели, как мы упростили задачу рисования человечка до четырех задач рисования его отдельных частей. Это очень удобный метод программирования. Он состоит в расчленении одной большой задачи на группу более простых, с каждой из которых можно заниматься порознь. Их, в свою очередь, тоже можно дробить до тех пор, пока вся задача, как бы сложна она ни была, не будет сведена к элементарнейшим операциям.

**ИМЕННО ДЛЯ ТАКОГО СТИЛЯ ПРОГРАММИРОВАНИЯ ЯЗЫК ЛОГО И ПРЕДНАЗНАЧЕН.**

Если Вы в восторге от того, что нарисовали, можете вывести рисунок на принтер по команде COPYSCREEN.

### **Редактирование.**

Если до сих пор Вы не сделали ни одной ошибки, считайте, что Вам повезло, но надо все-таки знать, каким образом можно внести изменения и исправления в свои процедуры. Для этого служит редактор (editor). Он вызывается по команде EDIT, после которой надо указать имя процедуры, которую Вы хотите переработать, например:

EDIT "LEGS

На экране появится текст процедуры LEGS. Курсор перемещается обычными "синклеровскими" курсорными клавишами 5,6,7,8 в сочетании с CAPS SHIFT.

Нажатие клавиши ENTER в той позиции, где стоит курсор, создает новую строку. Вы можете добавить новый текст, например PRINT "HELLO".

Стирание выполняется как обычно с помощью DELETE (CAPS SHIFT + 0).

При редактировании Вы можете воспользоваться и расширенным режимом (курсор "E"). После одновременного нажатия CAPS SHIFT и SYMB SHIFT в правом нижнем углу загорится символ "E".

В E-режиме курсорные клавиши действуют немного не так. Совместно с CAPS SHIFT клавиши 5 и 8 "перегоняют" курсор в начало или конец текущей строки, а клавиши 6 и 7 - в нижнюю или в верхнюю строку экрана. Можно перемещать курсор и на еще большие расстояния. Так, в этом режиме при нажатии клавиши "B" Вы попадете в начало текста (BEGIN), а при нажатии клавиши "E" - в конец текста, соответственно (END). Это очень удобно при редактировании процедур, длина которых больше, чем размер экрана.

Еще одно огромное удобство в этом режиме предоставляют клавиши "Y" и "R". С их помощью можно переносить строки программы с места на место. Попробуйте выставить курсор на строке RIGHT 60, затем войдите в E-режим и нажмите "Y". Строка RIGHT 60 исчезнет с экрана, но она не пропала окончательно, а осталась в памяти компьютера. Опустите курсор на две строки вниз и в E-режиме нажмите клавишу "R" - строка RIGHT 60 появится на новом месте.

Не бойтесь ничего испортить, поработайте с редактором. Любой редактор имеет свои особенности и к ним надо привыкнуть. Время, затраченное на эксперименты, многократно окупится в будущем.

Выйти из редактора можно двумя способами.

Если Вы не хотите, чтобы все сделанные Вами изменения отразились на программе, Вы можете нажать CAPS SHIFT и SPACE одновременно.

С другой стороны, если Вам надо, чтобы ваши изменения были зафиксированы, выход выполняется нажатием клавиши "C" в E-режиме.

У Вас есть возможность использования клавиш редактора и тогда, когда Вы не находитесь в самом редакторе. Так, например, если при наборе текста Вы заметите, что сделали ошибку, можете "перегнать" курсор и внести исправление до того, как была нажата клавиша ENTER и строка ушла в память компьютера.

Если Вы войдете в редактор, задав имя процедуры, которая до сих пор еще не была определена, то компьютер подумает, что Вы хотите создать новую процедуру. В этом случае перед вами будет пустой экран и слово "ТО, напечатанное в его начале. Таким образом, у Вас есть еще один метод создания процедур, отличающийся от того, который мы рассмотрели ранее. В принципе, этот путь более удобный, т.к. к вашим услугам все

имеющиеся ресурсы редактора, в том числе и такие, как вставка новых строк. Те, кто попробовал программировать самостоятельно, знают как это важно.

### **Сохранение программ на кассете.**

Мы дошли то той точки, где Вам может быть захочется сохранить только что написанные процедуры для дальнейшего использования, выключить компьютер и немного отдохнуть. Если просто выключить "Спектрум", то все, что хранилось в его памяти, будет навсегда утрачено. Даже если Вам и нет нужды сохранять результаты своих экспериментов, Вы все же попробуйте это сделать, чтобы убедиться, что все идет нормально.

Как и большинство языков программирования, ЛОГО имеет дело с "файлами" (так мы будем называть блоки информации, записанные на магнитофонной ленте (или на диске и т.п.). Каждый файл имеет свое собственное имя. В ЛОГО имена файлов (в отличие от прочих ключевых слов) не могут иметь более 7 символов. Это ограничение в данном случае не связано с ЛОГО, оно относится к самому "Спектруму". Так, например, если мы хотим сохранить процедуры, которые использовались при рисовании человечка в файле под именем PERSON, наберите:

```
SAVE "PERSON [MAN HEAD BODY LEGS!  
ARMS] <ENTER>
```

Включите магнитофон на запись и нажмите любую клавишу. Выгрузка идет, как обычно. По окончании сохранения файла на экране появится приглашение "?".

Если теперь Вам надо будет загрузить этот пакет процедур обратно в память компьютера, Вы можете делать следующим образом: LOAD "PERSON <ENTER>

### **Обслуживание памяти.**

Система ЛОГО занимает часть памяти Вашего компьютера. Еще часть памяти расходует сам ЛОГО, отслеживая ход исполняемой работы. Нужна также память для хранения промежуточных результатов вычислений. Так, например при расчете  $2+5-3$  компьютер сначала долже найти  $2+5$  и запомнить где-то этот промежуточный результат. Оставшаяся часть памяти может быть использована для хранения Вашей информации и созданных Вами процедур. Мы назовем эту часть памяти "рабочей областью".

Рано или поздно программист так или иначе столкнется с проблемой нехватки памяти в рабочей области. В этот момент ему было бы чрезвычайно важно узнать, какие же процедуры хранятся у него сейчас в памяти. Так, например, возможным выходом из создавшегося положения могло бы быть удаление части процедур, если они для работы не нужны.

Для этой цели ЛОГО имеет несколько встроенных команд. Вы можете попробовать их в работе, но для того, чтобы увидеть, как они действуют, надо, чтобы в памяти сейчас у Вас было несколько Ваших процедур. Так что если их нет, то либо наберите их, либо загрузите с ленты (если при прочтении прошлого параграфа Вы их отгрузили).

Первая команда, которую мы рассмотрим - это команда POTS. Название может быть выглядит не очень осмысленным, но это даже хорошо, так ее проще запомнить. Расшифровывается она, как Print Out TitleS (распечатать заголовки). По этой команде Вы можете получить, например вот такой результат:

```
TO TRIANGLE  
TO MAN  
TO HEAD  
TO ARMS  
TO BODY  
TO LEGS
```

Конечно это не все процедуры, которые знакомы для ЛОГО. Мы, например, знаем, что ЛОГО понимает такие процедуры, как FORWARD, BACK, RIGHT, LEFT и др. Но они по команде POTS не распечатываются, поскольку являются встроенными в систему. Это обязательные элементы языка, их еще называют "примитивами".

Другая команда с не менее нелепым названием - POPS означает Print Out ProcedureS (распечатать процедуры) выполняет печать всех Ваших процедур одну за другой. Так Вы можете их просмотреть.



Просмотреть одну какую-либо процедуру Вы можете в редакторе, вызвав ее по имени, но можно это сделать также и командой PO(Print Out). Например:

```
PO "LEGS
```

Может быть, Вы хотите, чтобы печать Ваших процедур производилась не на экран, а на принтер? Тогда Вы можете дать команду PRINTON. Начиная с этого момента принтер станет устройством вывода информации и результат работы POTS, POPS и PO будет распечатываться на принтере. Отключить принтер и вновь перевести весь вывод на экран можно командой PRINTOFF.

Итак, Вы научились просматривать рабочую область. Освободить ее от ставших ненужными процедур можно с помощью команды ERASE. Например:

```
ERASE "HEAD
```

Попробуйте это сделать, а потом дайте команду POTS для того, чтобы убедиться, что все прошло нормально. Нелишне напомнить, что перед удалением из памяти временно ненужных процедур стоит подумать о том, чтобы сохранить их на ленте.

И, наконец, рассмотрим команду для полного освобождения рабочей области. Это команда ERPS (ERase ProcedureS). После нее можно начинать работу сначала.

Все команды, рассмотренные в этом разделе, имеющие дело с конкретной процедурой, например EDIT, PO, ERASE должны иметь открывающие кавычки перед именем этой процедуры. Но они могут иметь после своего имени в квадратных скобках еще и список входящих в них процедур, так, как это было при рассмотрении команды SAVE. В этом случае поданная команда распространяется не только на головную процедуру, но и на все входящие, например:

```
EDIT "ARMS [LEGS]
```

даст Вам возможность одновременно редактировать не только процедуру ARMS, но и процедуру LEGS. Это может быть весьма удобно например в тех случаях, когда Вы хотите перенести строку из одной процедуры в другую.

Команда PO [HEAD BODY] распечатает обе процедуры, причем первой пойдет HEAD.

Команда ERASE [MAN TRIANGLE] удалит обе процедуры из рабочей области.

### **Печать текста.**

Если Вы хотите распечатать на экране какое-либо сообщение, Вы можете воспользоваться командой ЛОГО - PRINT. Обратите внимание на то, что если включен режим PRINTON, то с тем же успехом печать будет выполнена на принтере.

Если надо распечатать всего одно слово, то оно может быть написано после кавычек, например:

```
PRINT "HELLO
```

В этом случае есть разница, какими буквами прописными или строчными Вы набрали свое слово. Напечатано оно будет точно так, как было набрано.

Если Вам надо напечатать целое предложение (иди список слов), то весь список должен быть заключен в квадратные скобки:

```
PRINT [WELCOME TO LOGO]
```

Если Вы хотите напечатать пустую строку, воспользуйтесь командой PRINT " или PRINT [].

Мы продемонстрируем возможность этой команды на следующем примере, который назовем TZAR ("Царь").

```
TO TZAR
  PRINT []
  PRINT [Жил был царь]
  PRINT [у царя был двор]
  PRINT [На дворе был кол]
  PRINT [На колу мочало]
  PRINT [Не начать ли сказку!
        сначала?]
```

```
TZAR
END
```

Запустите процедуру командой TZAR. Остановить ее можно будет только нажав BREAK (CAPS SHIFT + SPACE).

### 3. Переменные.

В ЛОГО, как и в большинстве других языков, можно пользоваться словами для выражения каких-либо данных, например чисел. Так, например, предположим, перед Вами стоит задача рассчитать величину налога на добавочную стоимость (НДС). Предположим, что стоимость изделия - 30 руб. в этом случае Вы можете присвоить переменной PRICE значение 30. Это делается командой MAKE:

```
MAKE "PRICE 30
```

Кавычки перед словом PRICE обозначают, что это имя переменной, а не имя процедуры.

Теперь попробуйте дать команду

```
PRINT "PRICE
```

Компьютер напечатает слово PRICE. Да это и естественно, ведь кавычки перед ним говорят о том, что это просто слово, а не переменная. В то же время, есть возможность напечатать не PRICE, а то, что этим словом обозначено. Для этого может служить двоеточие.

Например:

```
PRINT :PRICE
```

Теперь компьютер напечатает число 30. Итак, в команде PRINT кавычки " обозначают само слово, а двоеточие (:) обозначает значение переменной, обозначенной этим словом.

Теперь мы можем перевести на язык ЛОГО следующее утверждение: "НДС равен 20-ти процентам от стоимости изделия". Оно будет выглядеть так:

```
MAKE "NDS :PRICE/100*20
```

Итак, команда MAKE требует наличия двух аргументов. Первый - это имя переменной (слово), а второй - это то значение, которое эта переменная теперь будет иметь (например число). Не забывайте о кавычках перед словом, которое идет после MAKE. Есть случаи, когда эти кавычки не нужны, но это особые случаи.

Обратите внимание и на то, что перед словом PRICE в нашем примере стоит двоеточие. Это потому, что не слово PRICE нас интересует, а то, что за ним скрывается, его числовое значение, равное 30, которое мы хотим разделить на 100 и умножить на 20 (НДС в России в 1993 г. равен 20-ти процентам). Как видите символы умножения (\*) и деления (/) совпадают с БЕЙСИКом. После того, как переменная NDS создана командой MAKE, мы можем ее напечатать:

```
PRINT :NDS
```

- и опять двоеточие, поскольку нам нужно не слово, а его значение.

В компьютерной терминологии слова PRICE и NDS называются переменными, поскольку они представляют заранее неизвестные значения, которые к тому же по ходу расчетов могут и изменяться. Посмотрите, например, как оперируют с переменной, выражающей изменяющуюся длину:

```
TO TRISPI  
  SHOWTURTLE  
  MAKE "LENGTH 10  
  REPEAT 21 [FORWARD :LENGTH RIGHT 120 MAKE "LENGTH :LENGTH +5 ]  
END
```

Внимательно посмотрите на последнюю команду в квадратных скобках. Этой командой переменной LENGTH присваивается новое значение путем прибавления к ранее имевшемуся значению числа 5. Запустите эту процедуру и посмотрите, как она работает.

При работе с переменными ЛОГО различает прописные и строчные буквы. Так,

```
MAKE "NDS 7  
MAKE "nds 4
```

создают две совершенно различные переменные.

```
PRINT :NDS даст 7  
PRINT :nds даст 4
```

Это может вызвать проблемы, если Вы печатаете все в нижнем регистре и забываете ставить двоеточия в кавычки там, где это нужно. Так, наберите:

```
print nds
```

и компьютер сам трансформирует запись в

```
PRINT NDS
```

поскольку он будет полагать, что и print и nds - имена процедур.

Поставить двоеточие перед NDS будет недостаточно, придется в редакторе обратно переправить NDS на nds.

### Арифметика.

Кроме арифметических знаков (\*),(+) и (/), с которыми мы уже встречались, существуют также и арифметические процедуры. Так, например, процедура SUM отыскивает сумму чисел, следующих за ней. Обычно это бывают два числа, но в некоторых случаях их бывает и больше. Тогда надо ставить круглые скобки так, как показано ниже.

Примеры:

```
PRINT SUM 2 3
5
PRINT (SUM 1 2 3 4 5)
15
```

Процедура PRODUCT выполняет умножение. Примеры:

```
PRINT PRODUCT 7 3
21
PRINT (PRODUCT 2 3 5 7)
210
```

Внимание: таким путем для встроенных функций могут вводиться более, чем по два аргумента. Это может вызвать проблему, если их использовать внутри квадратных скобок, например в команде REPEAT. Таких случаев следует избегать.

Процедура DIV выполняет деление. Она может использоваться только с двумя аргументами, например:

```
PRINT DIV 10 5 2
```

Внимание: в ЛОГО, как и во всех других языках деление на ноль невозможно.

Проверьте и посмотрите, что будет:

```
PRINT DIV 5 0
```

Это обычная ошибка начинающих программистов, особенно часто возникающая, когда делятся две переменные:

```
PRINT DIV :TOTAL :NUMBER
```

Программисты забывают учесть, что при определенных ситуациях переменная NUMBER может принимать нулевое значение.

Для вычитания нет своей процедуры. Поэтому это действие можно выполнить либо с помощью знака (-) либо с помощью суммирования SUM с отрицательным аргументом.

```
PRINT 12 - 4
```

дает 8

Знак "минус" выполняет двойную роль. Между двумя числами он обозначает вычитание, но если он стоит перед целым числом, то обозначает, что это число отрицательное.

Во всех случаях ЛОГО пытается разобраться с тем, что Вы написали и во многие случаях делает это весьма успешно. Так,

```
PRINT 12-4
```

или

```
PRINT 12 - 4
```

будет принято успешно.

Вы можете вычесть и отрицательное число, например:

```
PRINT 12 - - 4
```

или

```
PRINT 12--4
```

Тем не менее, все-таки существуют случаи, когда ЛОГО может быть введен в заблуждение неадекватным использованием знака "минус", поэтому на всякий случай выработайте в себе привычку всегда ставить пробелы перед знаком "минус" у отрицательного числа и не оставлять пробела между знаком "минус" и самим числом. Так, последний пример наиболее грамотно записать так:

```
PRINT 12 - -4
```

В одном выражении Вы можете объединять все четыре арифметических действия.

Например:

```
PRINT 3*4 + 2
```

дает 14

И дело вовсе не в том, что вычисления выполняются слева направо. Сравните:

```
PRINT 2 + 3*4
```

тоже дает 14

Порядок арифметических действий, принятый в ЛОГО выглядит так:

- действия в скобках;
- деление;
- умножение;
- вычитание;
- сложение.

Таким образом, Вы можете пользоваться скобками ( ) для того, чтобы изменять естественный порядок действий. Например:

```
PRINT (2 + 3) * 4
```

даст 20

При расчете выражения, стоящего в скобках, ЛОГО придерживается естественного порядка действий. Если Вы не уверены при написании выражения в порядке его расчета, смело расставляйте скобки, чтобы порядок был именно таким, какой Вам нужен. Еще два примера:

```
MAKE "CENTIGRADE (:FAHRENHEIT - 32) * 5 / 9
```

выполняет сначала вычитание, а

```
MAKE "FAHRENHEIT :CENTIGRADE *9/5 + 32
```

выполняет деление, умножение и последним сложение.

С точки зрения арифметики все равно выполнять сначала умножение или деление, но с точки зрения ЛОГО деление лучше выполнять раньше, чтобы избежать появления слишком больших чисел, с которыми будет трудно работать.

Одна из полезнейших возможностей в использовании переменных состоит в том, что благодаря им можно создавать процедуры со вводными параметрами. Например, Вы хотите иметь процедуру SQUARE для изображения на экране квадратов различной величины. Так, SQUARE 50 должна рисовать квадрат со стороной 50 единиц, а SQUARE 100 - со стороной 100 единиц. В этом случае мы можем ввести параметр SIZE, от которого будет зависеть размер квадрата и, меняя это параметр при вызове процедуры, изменять результат ее работы. Тогда определение процедуры SQUARE будет выглядеть так:

```
TO SQUARE :SIZE
```

```
  REPEAT 4[FORWARD :SIZE RIGHT 90]
```

```
END
```

В заключение в порядке эксперимента напишите процедуру POLYGON, которая должна иметь два входных параметра :SIZE и :NUMBER. Параметр SIZE задает размер стороны правильного многоугольника, а параметр NUMBER - число его сторон. Можете это сделать, модернизировав приведенную выше процедуру SQUARE. Правда, придется помозговать насчет угла поворота "черепашки" после изображения каждой стороны. Вам может помочь тот факт, известный из геометрии, что сумма углов выпуклого многоугольника равна  $180 \cdot (n-2)$  градусов, где  $n$  - число его сторон.

А пока мы прощаемся до следующего выпуска.

(Продолжение следует).

# AFRICAN SEEDS

Уважаемые читатели! Мы получаем массу писем с просьбой больше давать программ на БЕЙСИКЕ для самостоятельного набора начинающими. Многие полагают, причем совершенно справедливо, что это является прекрасной школой обучения программированию. Мы с этим утверждением полностью солидарны. И дело даже не в том, что при наборе программы начинающий программист осваивает клавиатуру, использует новые для себя операторы и конструкции языка, узнает новые приемы. Основная прелесть состоит в том, что как бы ни старался читатель набрать программу без ошибок, в 99 случаях из 100 у него это не выйдет. Практически все при любом опыте наделают ошибок. А вот поиск этих ошибок и отладка программы и являются по-настоящему ценной школой начинающего программиста.

Интерпретатор БЕЙСИКа устроен так, что от большинства ошибок он Вас застрахует. Если при наборе строки Вы нарушите синтаксис языка, он Вас немедленно предупредит и строка не будет введена в программу клавишей ENTER до тех пор, пока Вы эту ошибку не исправите. Тем не менее, существуют еще сотни возможных видов ошибок, которые он не увидит. Например, он ничего не сможет поделаться, если Вы пропустите какую-либо строку. Об этом Вы узнаете только запустив программу, причем ошибка проявится совсем не в том месте, в котором Вы пропустили строку. Она проявится там, где компьютеру надо будет воспользоваться данными, содержащийся в пропущенной строке.

Поэтому прежде, чем Вы приступите к набору программы, нам хотелось бы дать Вам некоторые приемы, которые помогут сделать мучительный процесс отладки чуть менее мучительным. Тем не менее, настройтесь все-таки на тяжелую и кропотливую работу.

Во-первых, когда программу набирают со страниц журнала или книги, большой процент ошибок возникает в связи с плохой различимостью символов. Так, например, часто путают буквы "i", "l", "I" и цифру "1". Путают букву "O" и цифру "0". Можно спутать букву "B" и цифру "8". Имея дело с этими символами относитесь к ним предельно осторожно.

Теперь о ходе отладки. Вы, конечно понимаете, что предусмотреть все возможные виды ошибок, которые Вы наделаете, заранее никто не сможет, хотя кое-какая статистика у нас имеется. Можно сказать, что до 90 процентов самых больших неприятностей связано с тремя видами ошибок:

Variable not found (переменная не найдена).

Out of screen (печать вне пределов экрана).

End of DATA (нехватка данных).

Что касается Variable not found, - это наиболее часто встречающаяся неприятность. К ней, как правило, и приводит путаница в символах, о которой мы писали выше. Хотя это, конечно, не единственный способ напороться на данную ошибку.

Когда компьютер сообщает об этой ошибке, он указывает и номер строки, в которой она произошла. Внимательно сверьте эту строку с исходным текстом. Если все в порядке, попробуйте распечатать переменные "подозрительного" выражения прямой командой. Та, которая ошибочна, распечатана не будет и Вы опять получите сообщение "Variable not found". Теперь надо найти программную строку, в которой эта переменная была в первый раз определена. Может быть эту строку Вы и пропустили или неправильно набрали переменную.

Иногда эта ошибка появляется не тогда, когда Вы что то пропустили, а наоборот набрали что-то лишнее, что было воспринято, как имя переменной.

Ошибка Out of screen возникает при печати (PRINT, PLOT, DRAW, CIRCLE), если координаты печати выходят за пределы экрана. Бороться с этой ошибкой весьма трудно, поскольку ошибка возникает в тех случаях, когда параметры координат позиции печати выражены не числом, а переменными, распечатайте их прямой командой PRINT.

Посмотрите, какой параметр "врет". Вернитесь по программе назад, найдите строку, в которой этот параметр вычисляется. Пусть это будет например строка 5550. Поставьте после нее отладочную строку:

```
5551 PRINT <имя перем.>: PAUSE 0
```

Так, постепенно "раскручивая" эту переменную, Вы и найдете в какой строке вычисления с ней портят Вашу программу.

Ошибка "Out of DATA" возникает в случае несоответствия количества данных, прочитанных программой по оператору READ и имеющихся в строках DATA. Борьба с ней довольно проста и требует только внимательности и точности.

А теперь, закончив с этим затянувшимся вступлением, мы предлагаем нашим читателям старинную африканскую игру "AFRICAN SEEDS". Игра относится к направлению, называемому "Манкала". В этом семействе имеются сотни разных игр. Выбранная нами игра отличается относительной простотой правил и несложной алгоритмизацией. Несмотря на свою простоту, компьютерная версия игры отличается тем, что победить ее непросто. Мы не будем здесь рассказывать правил игры, Вы их прочитаете непосредственно после запуска программы. Укажем только, что играть можно не только с компьютером. Воспользовавшись камешками или ракушками Вы можете играть с друзьями на пляже или с помощью сосновых шишек в лесу. В связи с приближением отпускного сезона, Вам это может пригодиться.

Программа переведена на русский язык. Если Ваш "Спектрум" имеет русский регистр, то набор русского текста не вызовет у Вас трудностей. Но, учитывая тот факт, что большинство компьютеров у наших пользователей нерусифицированные, мы использовали в программе загружаемый символьный набор "НС" в кодах КОИ-7 в стандарте ASCII (он приведен ниже). Это "утолщенный" русско-латинский символьный набор, предложенный для русификации программы "MF-09" (см. статью Алексеева А.Г. в "ZX-РЕВЮ" N 1-2 за 1992 г. стр. 29). Для тех читателей, которые подписались на "РЕВЮ" только в этом году, мы повторяем этот способ русификации. Мы будем всякий раз ссылаться на него при публикации в дальнейшем БЕЙСИК-программ для самостоятельного набора.

Для формирования символьного набора используется программа для шестнадцатиричного ввода (см. "ZX-РЕВЮ"-93, N 1-2, стр. 5-7).

### СИМВОЛЬНЫЙ НАБОР "НС" КОИ-7

|                                  |                                  |
|----------------------------------|----------------------------------|
| FC58:00 00 00 00 00 00 00 00 :54 | FD10:00 7E 66 0C 18 30 30 00 :75 |
| FC60:00 18 18 18 18 00 18 00 :D4 | FD18:00 3C 66 3C 66 66 3C 00 :FB |
| FC68:00 6C 6C 6C 00 00 00 00 :A8 | FD20:00 3C 66 66 3E 06 3C 00 :A5 |
| FC70:00 6C FE 6C 6C FE 6C 00 :18 | FD28:00 00 18 18 00 18 18 00 :85 |
| FC78:00 18 7E 58 7E 1A 7E 18 :90 | FD30:00 00 18 18 00 18 18 30 :BD |
| FC80:00 62 64 08 10 26 46 00 :C6 | FD38:00 00 0C 18 30 18 0C 00 :AD |
| FC88:00 30 58 30 7A CC 76 00 :F8 | FD40:00 00 00 7C 00 7C 00 00 :35 |
| FC90:00 18 30 00 00 00 00 00 :D4 | FD48:00 00 30 18 0C 18 30 00 :E1 |
| FC98:00 0C 18 18 18 18 0C 00 :0C | FD50:00 3C 66 0C 18 00 18 00 :2B |
| FCA0:00 30 18 18 18 18 30 00 :5C | FD58:00 7C CE D6 DE C0 7C 00 :8F |
| FCA8:00 00 6C 38 FE 38 6C 00 :EA | FD60:00 3C 66 66 7E 66 66 00 :AF |
| FCB0:00 00 18 18 7E 18 18 00 :8A | FD68:00 7C 66 7C 66 66 7C 00 :0B |
| FCB8:00 00 00 00 00 18 18 30 :14 | FD70:00 3C 66 60 60 66 3C 00 :71 |
| FCC0:00 00 00 00 7C 00 00 00 :38 | FD78:00 7C 66 66 66 66 7C 00 :05 |
| FCC8:00 00 00 00 00 18 18 00 :F4 | FD80:00 7E 60 7C 60 60 7E 00 :15 |
| FCD0:00 00 06 0C 18 30 60 00 :86 | FD88:00 7E 60 7C 60 60 60 00 :FF |
| FCD8:00 3C 66 6E 76 66 3C 00 :FC | FD90:00 3C 66 60 6E 66 3C 00 :9F |
| FCE0:00 18 38 18 18 18 3C 00 :B0 | FD98:00 66 66 7E 66 66 66 00 :11 |
| FCE8:00 3C 66 06 3C 60 7E 00 :A6 | FDA0:00 3C 18 18 18 18 3C 00 :75 |
| FCF0:00 3C 66 0C 06 66 3C 00 :42 | FDA8:00 0E 06 06 66 66 3C 00 :C7 |
| FCF8:00 0C 1C 2C 4C 7E 0C 00 :1E | FDB0:00 66 6C 78 78 6C 66 00 :41 |
| FD00:00 7E 60 7C 06 66 3C 00 :FF | FDB8:00 60 60 60 60 60 7E 00 :13 |
| FD08:00 3C 60 7C 66 66 3C 00 :25 | FDC0:00 C6 EE D6 D6 C6 C6 00 :A9 |

|         |    |    |    |    |    |    |     |         |         |    |    |    |    |    |    |     |     |
|---------|----|----|----|----|----|----|-----|---------|---------|----|----|----|----|----|----|-----|-----|
| FDC8:00 | 66 | 66 | 76 | 6E | 66 | 66 | 00  | :41     | FE90:00 | 00 | 3E | 30 | 30 | 30 | 30 | 00  | :8C |
| FDD0:00 | 3C | 66 | 66 | 66 | 66 | 3C | 00  | :DD     | FE98:00 | 00 | 66 | 3C | 18 | 3C | 66 | 00  | :F2 |
| FDD8:00 | 7C | 66 | 66 | 7C | 60 | 60 | 00  | :59     | FEA0:00 | 00 | 66 | 66 | 6E | 76 | 66 | 00  | :B4 |
| FDE0:00 | 3C | 66 | 66 | 6E | 6E | 3E | 00  | :FF     | FEA8:00 | 18 | 66 | 66 | 6E | 76 | 66 | 00  | :D4 |
| FDE8:00 | 7C | 66 | 66 | 7C | 6C | 66 | 00  | :7B     | FEB0:00 | 00 | 66 | 6C | 78 | 6C | 66 | 00  | :CA |
| FDF0:00 | 3C | 60 | 3C | 06 | 66 | 3C | 00  | :60     | FEB8:00 | 00 | 1E | 36 | 36 | 36 | 66 | 00  | :DC |
| FDF8:00 | 7E | 18 | 18 | 18 | 18 | 00 | :EB | FEC0:00 | 00      | C6 | EE | D6 | C6 | C6 | 00 | :D4 |     |
| FE00:00 | 66 | 66 | 66 | 66 | 66 | 3C | 00  | :38     | FEC8:00 | 00 | 66 | 66 | 7E | 66 | 66 | 00  | :DC |
| FE08:00 | 66 | 66 | 66 | 66 | 3C | 18 | 00  | :F2     | FED0:00 | 00 | 3C | 66 | 66 | 66 | 3C | 00  | :78 |
| FE10:00 | C6 | C6 | D6 | D6 | FE | 44 | 00  | :88     | FED8:00 | 00 | 7E | 66 | 66 | 66 | 66 | 00  | :EC |
| FE18:00 | 66 | 3C | 18 | 18 | 3C | 66 | 00  | :8A     | FEE0:00 | 00 | 3E | 66 | 3E | 36 | 66 | 00  | :5C |
| FE20:00 | 66 | 3C | 18 | 18 | 18 | 00 | :20 | FEE8:00 | 00      | 7C | 66 | 66 | 7C | 60 | 00 | :0A |     |
| FE28:00 | 7C | 0C | 18 | 30 | 60 | 7C | 00  | :D2     | FEF0:00 | 00 | 3C | 66 | 60 | 66 | 3C | 00  | :92 |
| FE30:00 | 1E | 18 | 18 | 18 | 1E | 00 | :CA | FEF8:00 | 00      | 7E | 18 | 18 | 18 | 18 | 00 | :D4 |     |
| FE38:00 | C0 | 60 | 30 | 18 | 0C | 06 | 00  | :B0     | FF00:00 | 00 | 66 | 66 | 3E | 06 | 3C | 00  | :4B |
| FE40:00 | 78 | 18 | 18 | 18 | 18 | 00 | :8E | FF08:00 | 00      | D6 | D6 | 7C | D6 | D6 | 00 | :DB |     |
| FE48:00 | 18 | 3C | 5A | 18 | 18 | 00 | :3C | FF10:00 | 00      | 7C | 66 | 7C | 66 | 7C | 00 | :4F |     |
| FE50:00 | 00 | 00 | 00 | 00 | 00 | FF | :4D | FF18:00 | 00      | 60 | 60 | 7C | 66 | 7C | 00 | :35 |     |
| FE58:00 | 00 | DC | F6 | F6 | F6 | DC | :F0 | FF20:00 | 00      | C6 | C6 | F6 | DE | F6 | 00 | :75 |     |
| FE60:00 | 00 | 3C | 66 | 7E | 66 | 66 | :4A | FF28:00 | 00      | 3C | 66 | 0C | 66 | 3C | 00 | :77 |     |
| FE68:00 | 00 | 7C | 60 | 7C | 66 | 7C | :A0 | FF30:00 | 00      | C6 | D6 | D6 | D6 | FE | 00 | :75 |     |
| FE70:00 | 00 | 6C | 6C | 6C | 6C | 7E | :A2 | FF38:00 | 00      | 78 | 0C | 3C | 0C | 78 | 00 | :7B |     |
| FE78:00 | 00 | 3C | 6C | 6C | 6C | FE | :BA | FF40:00 | 00      | C6 | D6 | D6 | D6 | FF | 03 | :89 |     |
| FE80:00 | 00 | 7E | 60 | 7C | 60 | 7E | :B6 | FF48:00 | 00      | 66 | 66 | 3E | 06 | 06 | 00 | :5D |     |
| FE88:00 | 00 | 7C | D6 | D6 | 7C | 10 | :3A | FF50:00 | 00      | 70 | 30 | 3C | 36 | 3C | 00 | :9D |     |

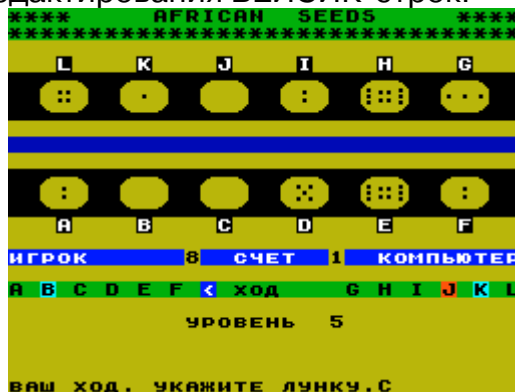
После того, как работа по вводу этих кодов будет завершена, запишите полученный символьный набор на магнитную ленту: SAVE "chr" CODE 61600,768. Он пригодится Вам и для других программ.

Перед тем, как набирать текст БЕЙСИК-программы, сделайте следующее:

```
CLEAR 64599: LOAD "chr"CODE
POKE 23606,88: POKE 23607,251
```

Теперь включен загруженный символьный набор. Попробуйте набирать БЕЙСИК программу. Токены ключевых слов "Спектрума", состоящие из заглавных латинских букв, будут печататься как обычно. Текст, набираемый при помощи буквенных клавиш при отключенном режиме CAPS LOCK, будет набираться русскими буквами. Так же он будет выглядеть и на экране. При включении регистра CAPS LOCK, текст будет набираться латинскими буквами. При этом, правда, придется обходиться только заглавными русскими и латинскими буквами, но зато это позволит легко совмещать русский и английский текст внутри одного оператора PRINT. Имена переменных для удобства чтения набираются заглавными латинскими буквами.

Кроме формирования самого символьного набора, в программу вводятся дополнительные строки. Это строка 2, загружающая символьный набор. С нее должен происходить автостарт программы. Еще это переключатели шрифта, строки 8 и 9 (могут быть любые другие номера, такие, какие удобно вписываются в программу). Используя команду GO SUB 8, можно включить загруженный символьный набор, а командой GO SUB 9 - выключить его. Эти команды можно как включать в БЕЙСИК-строки, так и подавать напрямую, находясь в режиме редактирования БЕЙСИК-строк.



Следует также предупредить тех пользователей, которые имеют БЕТА-ДИСК

интерфейс. В программах, в которых используется блок UDG-графики, стандартно расположенный после рестарта компьютера с адреса 65368, полезно бывает перед стартом программы включать еще и конструкцию, задающую системную переменную UDG:

```
POKE 23675,88:POKE 23676,255
```

Это, казалось бы лишнее, так как после рестарта компьютера в указанные ячейки и так находятся указанные числа. Дело в том, что значения этих ячеек могли быть изменены в процессе работы дискового загрузчика "boot". Так некорректно ведет себя, например, в общем-то очень хороший и широко распространенный загрузчик "MOA-SERVICE". Эта рекомендация убережет Вас от возможных "накладок" и в других случаях.

А теперь, когда есть определенность с вопросом русификации, можно переходить непосредственно к тексту программы "AFRICAN SEEDS".

```
1 GO TO 1000
2 CLEAR 64599:LOAD "CHR" CODE 64600
3 POKE 23675,88:POKE 23676,255:RESTORE 2100:FOR P=0 TO 95:READ G:POKE USR "A"+P,G:NEXT P
4 RUN
5 SAVE "AFRICA"LINE 2:SAVE "CHR" CODE 64600,768:STOP
8 POKE 23606,88:POKE 23607,251:RETURN:REM RUS
9 POKE 23606,0:POKE 23607,60:RETURN:REM LAT
100 PRINT AT 17, CODE A$(H);BRIGHT 1:PAPER 2;SCREEN$(17, CODE A$(H)):LET S=H(H):LET
    H(H)=0:LET D=D+1
110 IF S THEN LET S=S-1:LET H=H+1-12*(H=12):LET H(H)=H(H)+1:GO TO 110
200 PRINT AT 17, CODE A$(H);BRIGHT 1;INK 0:PAPER 2;SCREEN$(17, CODE A$(H)):IF H(H)<>1 THEN
    FOR B=10 TO 10+H(H):BEEP .002,B+D:NEXT B:RETURN
210 PRINT AT 17, CODE A$(H);BRIGHT 1:PAPER 5;SCREEN$(17, CODE A$(H)):LET END=1:LET OP=13-
    H:FOR B=20 TO 20+H(OP):BEEP .02,B:NEXT B:IF (NOT ZX AND OP<7) OR (ZX AND OP>6) OR NOT
    H(OP) THEN RETURN
220 LET WIN=H(OP):LET H(OP)=0:LET S(2-ZX)=S(2-ZX)+WIN*SC:PRINT AT 17, CODE A$(OP);FLASH
    1;BRIGHT 1:PAPER 5;SCREEN$(17, CODE A$(OP)):FOR B=30 TO 30+WIN:BEEP .025,B:NEXT
    B:RETURN
300 PRINT AT 19,25;" ";AT 19,11;BRIGHT 1:PAPER (D-8*INT (D/8));INK 9;" ХОД ";D;"
    ";:FOR P=1 TO 12:RESTORE 2300+B(P):READ N$
330 PRINT AT CODE X$(P), CODE Y$(P);N$:LET H(P)=B(P):BEEP .005,D+B(P):NEXT P
340 PRINT AT 15,11;S(2);:PRINT AT 15,20;S(1):PRINT AT 19,11;L$:RETURN
400 BRIGHT 0:PRINT AT 17,0:PAPER 4;"А В С D E F ХОД G H I J K L":RETURN
500 PRINT ("ЧИСЛО ПРОИГРАННЫХ ВАМИ ОЧКОВ: "+STR$ (S(1)-S(2)) AND S(1)>S(2))+("ЧИСЛО
    ВЫИГРАННЫХ ВАМИ ОЧКОВ: "+STR$ (S(2)-S(1)) AND S(1)<S(2))
510 GO SUB 400
520 PRINT AT 0,0:INPUT "ХОТИТЕ СЫГРАТЬ ЕЩЕ РАЗ? (Y) ";LINE U$:IF CODE U$=121-32*(U$="Y")
    THEN GO TO 1040
530 STOP
580 PRINT AT 18,0;"ЛУНКИ ПРОТИВНИКА ПУСТЫ. ВСЕ ОС-ТАВШИЕСЯ ФИШКИ ПЕРЕШЛИ К НЕМУ."
600 PRINT AT 20,0;"ИГРА ";("ПРОДОЛЖАЕТСЯ." AND S(1)=S(2));("ЗАКОНЧЕНА." AND S(1)<>S(2)):GO
    TO 500
700 IF S(1)>24 OR S(2)>24 THEN GO TO 600
710 IF S(1)+S(2)>39 AND NOT WIN THEN LET C=C+1:IF C>10 THEN GO TO 600
720 RETURN
1000 GO SUB 8:POKE 23658,8:LET D=0:DIM S(2):DIM B(12):DIM H(12):DIM A$(12):DIM C$(12):DIM
    X$(12):DIM Y$(12)
1020 GO SUB 2000:PRINT AT 19,1;INK 0;"ВЫ БУДЕТЕ СМОТРЕТЬ ИНСТРУКЦИЮ?";AT 21,4;"(ЕСЛИ ДА, ТО
    ВВЕДИТЕ Y)":INPUT LINE U$:IF CODE U$+32*(U$="Y")=121 THEN
GO SUB 2400
1030 RESTORE 2160:FOR N=1 TO 12:READ A:LET A$(N)=CHR$ A:NEXT N
1040 GO SUB 2020
1060 GO SUB 2080:INPUT LINE U$:IF U$<"0" OR U$>"9" THEN GO TO 1060
1070 LET L=VAL U$+20*(U$="0"):PRINT AT 15,0;" ";:FOR P=15 TO 21 STEP
    2:PRINT AT P,17;" ";:NEXT P:LET L$="УРОВЕНЬ "+U$+" "
1080 RESTORE 3000:FOR N=1 TO 12:READ X,Y:PRINT AT X,Y;" ";AT X+2-4*(N>6),Y+1;INK 7:PAPER
    0;BRIGHT 1;CHR$ (64+N):LET X$(N)=CHR$ X:LET Y$(N)=CHR$ Y:NEXT N
1090 PRINT AT 15,0;BRIGHT 1:PAPER 1;INK 7;"ИГРОК СЧЕТ КОМПЬЮТЕР"
1100 DIM M(12):LET D=0:LET T=0:LET SC=0:LET C=0:LET S(1)=0:LET S(2)=0:FOR P=1 TO 12:LET
    B(P)=4:LET H(P)=4:NEXT P
1110 RANDOMIZE :LET ZX=INT (RND *2)
```





```

2080 PRINT AT 15,0;"ВВЕДИТЕ УРОВЕНЬ: 1-3=НОВИЧОК" TAB17;"4-6=ЛЕГКИЙ" TAB17;"7-
      9=ТЯЖЕЛЫЙ" TAB19;"0=ЭКСПЕРТ"
2090 RETURN
2100 DATA 0,0,0,24,24,0,0,0
2101 DATA 0,24,24,0,0,24,24,0
2102 DATA 24,24,0,24,24,0,24,24
2103 DATA 0,102,102,0,0,102,102,0
2104 DATA 195,195,0,24,24,0,195,195
2105 DATA 102,102,0,102,102,0,102,102
2106 DATA 102,102,0,219,219,0,102,102
2107 DATA 219,219,0,102,102,0,219,219
2108 DATA 255,255,255,255,31,7,3,1
2109 DATA 255,255,255,255,248,224,192,128
2110 DATA 128,192,224,248,255,255,255,255
2111 DATA 1,3,7,31,255,255,255,255
2160 DATA 0,2,4,6,8,10,21,23,25,27,29,31
2300 DATA ""
2301 DATA " A "
2302 DATA " B "
2303 DATA "AAA"
2304 DATA " D "
2305 DATA " E "
2306 DATA "BBB"
2307 DATA "BCB"
2308 DATA "BDB"
2309 DATA "CCC"
2310 DATA "CDC"
2311 DATA "CEC"
2312 DATA "DDD"
2313 DATA "DED"
2314 DATA "EDE"
2315 DATA "FCF"
2316 DATA "FDF"
2317 DATA "GCG"
2318 DATA "FFF"
2319 DATA "HCH"
2320 DATA "HDH"
2321 DATA "GGG"
2322 DATA "HFN"
2323 DATA "HGH"
2324 DATA "HHH"
2325 DATA " 25"
2326 DATA " 26"
2327 DATA " 27"
2328 DATA " 28"
2329 DATA " 29"
2330 DATA " 30"
2400 BORDER 4:PAPER 4:INK 0:CLS:PRINT "***** AFRICAN SEEDS
      *****"
2410 PRINT ""
      ИГРА ОСНОВАНА НА АФРИКАНСКИХ
      ИГРАХ ТИПА ""МАНКАЛА"", ХОТЯ ЕЕ
      ПРАВИЛА ПРОЩЕ, ЧЕМ У БОЛЬШИНСТВА
      ЕЕ ПРЕДШЕСТВЕННИЦ."
2420 PRINT ""
      ИМЕЕТСЯ 12 ЛУНОВ С ЧЕТЫРЬМЯ
      ФИШКАМИ КАЖДАЯ. ОНИ ПОРОВНУ РАЗ-
      ДЕЛЕНА МЕЖДУ ИГРОКАМИ. ВАША ЦЕЛЬ
      - НАБРАТЬ ОЧКИ, СОБИРАЯ КАК МО-
      ЖНО БОЛЬШЕ ФИШЕК. 25 ОЧКОВ ОБЕ-
      СПЕЧИВАЕТ ПОБЕДУ."
2430 INPUT "ДЛЯ ПРОДОЛЖЕНИЯ НАЖМИТЕ ENTER ";LINE U$
2440 BORDER 4:PAPER 4:INK 0:CLS:PRINT "***** AFRICAN SEEDS
      *****"

```

```

2450 PRINT '''
        ИГРОКИ ПО ОЧЕРЕДИ БЕРУТ ФИШКИ
        ИЗ ЛЮБОЙ СВОЕЙ ЛУНКИ И РАССЫПАЮТ
        ИХ ПО ОДНОЙ В СОСЕДНИЕ ЛУНКИ
        ПРОТИВ ЧАСОВОЙ СТРЕЛКИ, НАЧИНАЯ
        С ЛУНКИ, НАХОДЯЩЕЙСЯ ПО-СОСЕД-
        СТВУ С НАЧАЛЬНОЙ."
2460 PRINT '''
        ЕСЛИ ПОСЛЕДНЯЯ ФИШКА ОКАЖЕТСЯ
        В ЗАНЯТОЙ ЛУНКЕ, ТО ИГРОК БЕРЕТ
        ФИШКИ ИЗ НЕЕ И ПРОДОЛЖАЕТ ХОД В
        ТОМ ЖЕ ПОРЯДКЕ. ТАК ПРОДОЛЖАЕТ-
        СЯ, ПОКА ХОД НЕ ЗАВЕРШИТСЯ В ПУ-
        СТОЙ ЛУНКЕ."
2470 PRINT '''
        ЕСЛИ ХОД ЗАКАНЧИВАЕТСЯ В ПУС-
        ТОЙ ЛУНКЕ ПРОТИВОПОЛОЖНОГО РЯДА,
        ИЛИ НАПРОТИВ ПУСТОЙ ЛУНКИ, ТО
        ХОД ПЕРЕХОДИТ К ПРОТИВНИКУ."
2480 INPUT "ДЛЯ ПРОДОЛЖЕНИЯ НАЖМИТЕ ENTER ";LINE U$
2490 BORDER 4:PAPER 4:INK 0:CLS:PRINT "***** AFRICAN SEEDS
        *****"
2500 PRINT '''
        ЗАХВАТ ФИШЕК ВОЗМОЖЕН ТОЛЬКО
        ЕСЛИ ПОСЛЕДНЯЯ ФИШКА В ХОДЕ ПО-
        ПАДАЕТ В ПУСТУЮ ЛУНКУ СВОЕГО РЯ-
        ДА, НАПРОТИВ ЗАНЯТОЙ ЛУНКИ ПРО-
        ТИВНИКА. ТОГДА ФИШКИ ПРОТИВНИКА
        ИЗЫМАЮТСЯ, А СЧЕТ УВЕЛИЧИВАЕТСЯ."
2510 PRINT '
        ИГРА ПРОДОЛЖАЕТСЯ, ПОКА ОДИН
        ИЗ ИГРОКОВ НЕ НАБЕРЕТ БОЛЕЕ 24
        ОЧКОВ ИЛИ ОБА ПАРТНЕРА НЕ СОГЛА-
        СЯТСЯ НА НИЧЬЮ."
2520 PRINT '
        ВНИМАНИЕ! ЕСЛИ ВЫ ОСТАВИТЕ
        СВОЕГО ПРОТИВНИКА БЕЗ ФИШЕК И
        БЕЗ ХОДА, ТО ВСЕ ФИШКИ, ИМЕЮЩИЕ-
        СЯ НА ДОСКЕ, ПЕРЕХОДЯТ К НЕМУ."
2530 INPUT "ДЛЯ ПРОДОЛЖЕНИЯ НАЖМИТЕ ENTER ";LINE U$
2540 BORDER 4:PAPER 4:INK 0:CLS:PRINT "***** AFRICAN SEEDS
        *****"
2550 PRINT '
        ПРАВО ПЕРВОГО ХОДА ОПРЕДЕЛЯЕТ
        КОМПЬЮТЕР СЛУЧАЙНЫМ ОБРАЗОМ."'''
        ВЫ ИМЕЕТЕ ЛУНКИ ОТ А ДО F. КОМ-
        ПЬЮТЕР ИМЕЕТ ЛУНКИ ОТ G ДО L."
2560 PRINT '
        КОГДА БУДЕТ ВАШ ХОД, ВВЕДИТЕ
        СИМВОЛ ТОЙ ЛУНКИ, ИЗ КОТОРОЙ ВЫ
        ХОТИТЕ СДЕЛАТЬ ХОД. КОМПЬЮТЕР
        ЖДЕТ НАЖАТИЯ ENTER ДЛЯ НАЧАЛА
        СВОЕГО ХОДА."
2570 PRINT '
        КОГДА ИГРА БУДЕТ ЗАВЕРШЕНА,
        КОМПЬЮТЕР ПОДСЧИТАЕТ И ОБЪЯВИТ
        РЕЗУЛЬТАТ. МОЖНО ПРЕРВАТЬ ИГРУ
        В ЛЮБОЕ ВРЕМЯ, ВВЕДЯ: "" STOP ""."
2580 INPUT
        ВВЕДИТЕ R ДЛЯ ПОВТОРА ИНСТРУКЦИИ
        ИЛИ ENTER, ЧТОБЫ НАЧАТЬ ИГРУ. ";LINE 3$:IF CODE 3$=114-32*(3$="R") THEN GO TO 2400
2590 RETURN
3000 DATA 11,2,11,7,11,12,11,17,11,22,11,27,5,27,5,22,5,17,5,12,5,7,5,2

```

Набирая программу, Вы периодически можете записывать результат на магнитную ленту, выполняя: RUN 5. При наборе программы обратите внимание, пожалуйста, на те места, где набор происходит с использованием графического регистра. Это строки 2000-2005, в которых используется символ с кодом 143 (регистр [G], затем CS+8). Это также символы с кодами 131, 140, 143 блочной графики и символы "I"-"L" UDG-графики в строках 2030, 2050. Символы UDG-графики "A"-"H" используются при наборе строк 2300-2324.

Автостарт программы происходит со строки 2, где прежде всего резервируется место для расположения загружаемого символьного набора, путем переустановки значения RAMTOP оператором CLEAR. После загрузки символьного набора происходит задание банка символов UDG-графики. На этом подготовку можно считать законченной. Далее строка 4 передает управление на начало программы. При отладке, после остановки программы, ее можно запустить командой RUN с начальной строки.

Так как мы используем русско-латинский символьный набор, то перед тем, как будет произведен ввод параметров при помощи оператора INPUT, корректно будет принудительно включать режим CAPS LOCK (устанавливать курсор [C]), так как вводиться должны параметры в виде английских букв. Это происходит, например, в начале выполнения программы, в строке 1000 подачей команды POKE 23658,6, а также в других местах.

Подпрограмма GO SUB 2000 в строке 1020 - это вывод на экран титульной заставки. После этого происходит запрос для вывода инструкции. Результат запроса (переменная U\$) анализируется, причем результат не зависит от того, нажата ли клавиша "Y" в режиме CAPS LOCK или нет. В том случае, если нажата клавиша "Y", происходит вызов подпрограммы GO SUB 2400, которая выводит на экран инструкцию с правилами игры.

Подпрограмма GO SUB 2020 в строке 1040 обеспечивает прорисовку игрового поля - игровые лунки играющего и компьютера.

В строке 1060 происходит ввод уровня сложности игры. Кстати, при тестировании программы выяснилось, что эта величина нигде дальше в программе не участвует. Непонятно, для чего автор ввел этот параметр. Может быть, рассчитывая в дальнейшем на усовершенствование игры, или на то, что "пытливые умы" сами сделают такое усовершенствование. В общем, есть простор для творческого поиска. А можно пойти по другому пути: просто исключить все, что касается ввода уровня. Но нам кажется, что вариант с заданием уровня (пусть фиктивным) смотрится динамичнее.

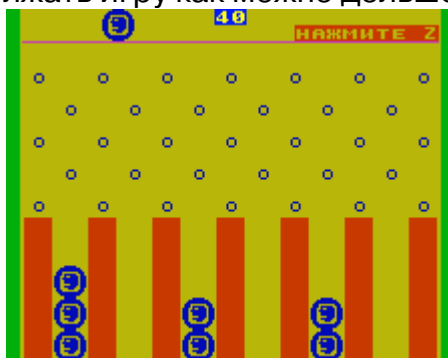
Далее идет подготовка игрового поля и предустановка исходных данных. В строке 1110 происходит определение того, кто делает первый ход. Если право первого хода выпало компьютеру, тогда после нажатия ENTER программа переходит на строку 1400. Кстати, свой первый ход компьютер делает случайным образом выбирая одну из своих лунок. Это происходит путем задания переменной PREF в строке 1150. Но это касается только первого хода. В дальнейшем, перед тем, как сделать ход, компьютер рассчитывает наиболее удачный вариант. Это осуществляется в строках, начиная с 1200. В этой строке, если право очередного хода имеет человек, то происходит переход на строку 1500. Если ход принадлежит компьютеру, то ожидается нажатие ENTER, после чего ситуация анализируется (со строки 1300) а затем, в строке 1360, выбирается наиболее предпочтительный вариант PREF. Далее, со строки 1400 происходят действия, непосредственно связанные с выполнением кода.

Если очередной ход делает человек, то происходит переход на строку 1500. На эту строку мы попадаем сразу после того, как будет закончен ход компьютера (в строках с 1400) или в начале игры, в случае первого хода, (см. строки 1110-1200). В конце этой части программы происходит передача хода компьютеру (строка 1580) и повторение цикла со строки 1200.

И, в заключение, предлагаем еще одну игру для тех, кто любит программы, развивающие внимательность, наблюдательность и быстроту реакции.

# CASH-FLOW

Игра имитирует игровой автомат. Монета падает вниз, ударяясь о неподвижные штыри. Таким образом, путь монеты в значительной мере случаен. В нижней части игрового поля монета попадает в один из шести монетоприемников, в каждом из которых уже может находиться до четырех монет. Ваша задача - довести их количество до пяти. Тогда монеты будут выданы Вам в качестве приза (разумеется, при наличии у вашего "Спектрума" соответствующей приставки для выдачи "наличных"). Выигранные монеты можно использовать для продолжения игры. Результат игры в основном зависит от момента, когда Вы бросаете монету. В то же время, у Вас есть возможность подтолкнуть падающую монету в любом направлении, но только один раз и только в верхней части доски со штырями. Монетоприемник N 4 "проглатывает" все монеты, поэтому его следует избегать всеми силами. Всего на игру Вам дается 6 монет по 10 пенсов каждая. Постарайтесь как можно раньше получить выигрыш и продолжать игру как можно дольше.



```
1 GO TO 10
2 CLEAR 64599:LOAD "CHR" CODE 64600
3 POKE 23606,88:POKE 23607,251
4 GO SUB 3000:RUN
5 SAVE "CASH-FLOW"LINE 2:SAVE "CHR" CODE 64600,768:STOP
10 POKE 23658,8:BORDER 5:PAPER 6:INK 1
15 LET OW=0
20 LET A$="AB":LET B$="CD"
25 GO SUB 2500
30 CLS
35 FOR N=0 TO 21:DIM I$(26):PRINT PAPER 6;AT N,3;I$:DIM J$(2):PRINT PAPER 5;AT N,0;J$;AT
  N,30;J$:NEXT N
40 FOR N=0 TO 192 STEP 32
45 CIRCLE 32+N,77,2:BEEP .05,10:CIRCLE 32+N,109,2:BEEP .05,20:CIRCLE 32+N,141,2:BEEP .05,30
50 NEXT N
55 FOR N=0 TO 160 STEP 32
60 CIRCLE 48+N,93,2:BEEP .05,40:CIRCLE 48+N,125,2:BEEP .05,20
65 NEXT N
70 DEF FN T( )=INT ((65536*PEEK 23674+256*PEEK 23673+PEEK 23672)/50)
72 LET T1=FN T( )
75 FOR N=0 TO 21
80 PRINT INK 4;AT N,2;"F";AT N,29;"E"
85 IF N>12 THEN PRINT INK 2;AT N,3;"EF EF EF EF EF EF EF"
90 NEXT N
95 PLOT INK 3;24,159:DRAW INK 3;207,0
100 LET W=0:LET L=0
120 GO TO 1000
130 PRINT AT X,Y;" ";AT X+1,Y;" "
135 IF INKEY$="Q" AND L>40 THEN LET NW=W-L:GO TO 2000
140 LET K=2*COS (PI*(INT (RND *2)))
150 LET X=X+2
151 IF I=0 AND M<4 AND Y<=25 AND INKEY$="P" THEN BEEP .05,32:LET Y=Y+2:LET I=1:GO TO 160
152 IF I=0 OR M<4 AND Y>=5 AND INKEY$="O" THEN BEEP .05,32:LET Y=Y-2:LET I=1:GO TO 160
154 IF K>0 AND Y=27 THEN LET Y=Y-K:GO TO 160
155 IF K>0 AND Y<26 THEN LET Y=Y+K
```

```

157 IF K<0 AND Y=3 THEN LET Y=Y-K:GO TO 160
158 IF K<0 AND Y>4 THEN LET Y=Y+K
160 PRINT INK 1;AT X,Y;A$;AT X+1,Y;B$
165 BEEP .05,20
166 FOR N=1 TO 100:NEXT N
170 LET M=M+1
180 IF M=6 THEN GO TO 500
190 GO TO 130
500 IF Y=5 THEN LET G=A:LET A=A+1:GO TO 530
505 IF Y=9 THEN LET G=B:LET B=B+1:GO TO 530
510 IF Y=13 THEN LET G=C:LET C=C+1:GO TO 530
515 IF Y=17 THEN LET G=0:GO TO 530
520 IF Y=21 THEN LET G=E:LET E=E+1:GO TO 530
525 IF Y=25 THEN LET G=F:LET F=F+1
530 IF G=4 THEN GO TO 600
540 FOR N=0 TO 3-G:PRINT INK 1;AT 12+2*N,Y;" ";AT 13+2*N,Y;" ";AT 14+2*N,Y;A$;AT
    15+2*N,Y;B$
550 PAUSE 20
560 NEXT N
565 IF Y=17 THEN PRINT AT 20,17;" ";AT 21,17;" ":BEEP 1,0
568 IF CR<=0 THEN GO TO 2000
570 PRINT INK 2;FLASH 1;AT 1,20;"HAXMMTE A"
575 IF INKEY$="A" THEN PRINT INK 2;FLASH 1;AT 1,28;"Z":GO TO 1100
576 LET T=FN T():IF T>T1+240 THEN GO TO 1990
578 IF INKEY$="Q" AND L>40 THEN LET NW=W-L:GO TO 2000
580 GO TO 575
600 BEEP 1,40
604 FOR N=0 TO 4
606 LET W=W+10:LET NW=W-L:LET CR=NW+60
608 PRINT AT 12+2*N,Y;" ";AT 13+2*N,Y;" ":BEEP .1,30:PAUSE 20
610 PRINT AT 0,15;" "
612 PRINT BRIGHT 1;FLASH 1;AT 0,15;CR
615 NEXT N
618 IF CR<=0 THEN GO TO 2000
620 IF Y=5 THEN LET A=0
621 IF Y=9 THEN LET B=0
622 IF Y=13 THEN LET C=0
624 IF Y=21 THEN LET E=0
625 IF Y=25 THEN LET F=0
630 PRINT INK 2;FLASH 1;AT 1,20;"HAXMMTE A"
635 IF INKEY$="A" THEN PRINT INK 2;FLASH 1;AT 1,28;"Z":GO TO 1100
638 IF INKEY$="Q" AND L>40 THEN GO TO 2000
640 GO TO 635
1000 LET A=INT (RND *3):FOR N=1 TO A:IF A<>0 THEN PRINT INK 1;AT 23-2*N,5;B$;AT 22-
    2*N,5;A$:BEEP .05,0:NEXT N
1010 LET B=INT (RND *3):FOR N=1 TO B:IF B<>0 THEN PRINT INK 1;AT 23-2*N,9;B$;AT 22-
    2*N,9;A$:BEEP .05,10:NEXT N
1020 LET C=INT (RND *5):FOR N=1 TO C:IF C<>0 THEN PRINT INK 1;AT 23-2*N,13;B$;AT 22-
    2*N,13;A$:BEEP .05,20:NEXT N
1040 LET E=INT (RND *5):FOR N=1 TO E:IF E<>0 THEN PRINT INK 1;AT 23-2*N,21;B$;AT 22-
    2*N,21;A$:BEEP .05,30:NEXT N
1050 LET F=INT (RND *3):FOR N=1 TO F:IF F<>0 THEN PRINT INK 1;AT 23-2*N,25;B$;AT 22-
    2*N,25;A$:BEEP .05,30:NEXT N
1055 LET CR=60:PRINT BRIGHT 1;FLASH 1;AT 0,15;CR
1060 PRINT INK 2;FLASH 1;AT 1,20;"HAXMMTE A"
1065 IF INKEY$="A" THEN PRINT INK 2;FLASH 1;AT 1,28;"Z":GO TO 1100
1068 IF INKEY$="Q" AND L>40 THEN GO TO 2000
1070 GO TO 1065
1100 LET L=L+10:LET NW=W-L:LET CR=NW+60:LET I=0
1101 PRINT AT 0,15;" "
1102 PRINT BRIGHT 1;FLASH 1;AT 0,15;CR
1104 PRINT INK 1;AT 0,3;A$;AT 1,3;B$:PAUSE 6
1105 FOR N=3 TO 27
1110 PRINT AT 0,N;" ";AT 1,N;" "
1115 IF N=27 AND CR<=0 THEN GO TO 2000

```

```

1120 IF N=27 THEN GO TO 1060
1130 PRINT INK 1;AT 0,N+1;A$;AT 1,N+1;B$
1140 PAUSE 6
1150 IF INKEY$="Z" AND INT (N/2)<>INT ((N-1)/2) THEN LET Y=N+1:PRINT AT 1,20;"      ":GO
      TO 1200
1160 NEXT N
1200 LET X=0:LET M=0
1210 PRINT AT X,Y;"  ";AT X+1,Y;"  "
1212 PRINT AT 0,15;"  "
1215 PRINT BRIGHT 1;FLASH 1;AT 0,15;CR
1220 LET X=X+2:LET M=M+1
1230 PRINT INK 1;AT X,Y;A$;AT X+1,Y;B$
1240 BEEP .05,20:FOR N=1 TO 50:NEXT N
1250 IF INT ((Y+1)/4)=INT ((Y+3)/4) THEN GO TO 1300
1260 IF M=2 THEN GO TO 1300
1270 GO TO 1210
1300 PRINT AT X,Y;"  ";AT X+1,Y;"  "
1310 PLOT INK 3;24,159:DRAW INK 3;207,0
1320 GO TO 135
1990 PRINT AT 0,15;"  ";AT 1,20;"  "
1992 PRINT INK 1;FLASH 1;AT 1,8;" ИГРА ОКОНЧЕНА "
1994 FOR N=0 TO 400:NEXT N
2000 PRINT AT 0,15;"  ";AT 1,20;"  "
2002 LET OW=OW+NW
2003 IF OW>=0 AND INT ((OW-10)/100)=INT (OW/100) THEN PRINT INK 7;PAPER 1;BRIGHT 1;AT
      1,4;"ИТОГОВЫЙ ВЫИГРЫШ #";OW/100;"0 "
2004 IF OW>=0 AND INT ((OW-10)/100)<>INT (OW/100) THEN PRINT INK 7;PAPER 1;BRIGHT 1;AT
      1,4;"ИТОГОВЫЙ ВЫИГРЫШ #";OW/100;"0"
2006 IF OW<0 AND INT ((OW-10)/100)=INT (OW/100) THEN PRINT INK 7;PAPER 1;BRIGHT 1;AT
      1,4;"ИТОГОВЫЙ ПРОИГРЫШ #";-OW/100;"0 "
2007 IF OW<0 AND INT ((OW-10)/100)<>INT (OW/100) THEN PRINT INK 7;PAPER 1;BRIGHT 1;AT
      1,4;"ИТОГОВЫЙ ПРОИГРЫШ #";-OW/100;"0"
2009 FOR N=0 TO 8
2010 DIM I$(26)
2020 PRINT PAPER 2;BRIGHT 1;AT 13+N,3;I$
2030 NEXT N
2040 PRINT INK 7;PAPER 2;AT 14,4;" ВЫ ИЗРАСХОДОВАЛИ - ";BRIGHT 1;L
2050 PRINT INK 7;PAPER 2;AT 16,7;" И ПОЛУЧИЛИ - ";BRIGHT 1;W
2060 IF NW>=0 THEN PRINT INK 7;PAPER 2;AT 18,5;"ВАША ПРИБЫЛЬ - ";INK 7;PAPER 0;BRIGHT 1;AT
      18,22;NW
2070 IF NW<0 THEN PRINT INK 7;PAPER 2;AT 18,7;"ВАШИ ПОТЕРИ - ";INK 7;PAPER 0;FLASH 1;BRIGHT
      1;-NW
2080 PRINT INK 7;PAPER 1;BRIGHT 1;FLASH 1;AT 20,4;"СЫГРАЕТЕ ЕЩЕ РАЗ? ДА-<Y>"
2082 IF T>T1+240 AND INKEY$="Y" THEN LET T1=FN T():LET OW=0:GO TO 2100
2085 IF INKEY$="Y" THEN GO TO 2100
2090 GO TO 2040
2100 PRINT AT 1,3;"      "
2105 FOR N=0 TO 8
2110 DIM I$(26)
2120 PRINT PAPER 6;AT 13+N,3;I$
2130 NEXT N
2140 GO TO 75
2500 DIM I$(704):PRINT AT 0,0;I$
2520 PLOT INK 6;26,144:DRAW 0,-32,11*PI/10:BEEP .05,10
2523 PAUSE 30
2524 PLOT 32,112:DRAW 12,32:BEEP .05,10:DRAW 12,-32:BEEP .05,10:PLOT 35,120:DRAW 18,0:BEEP
      .05,10
2526 PLOT INK 6;80,136:DRAW -8,-8,3*PI/2:BEEP .05,10:PLOT INK 6;64,120:DRAW 8,8,3*PI/2:BEEP
      .05,10
2527 PAUSE 30
2528 PLOT 88,112:DRAW 0,32:BEEP .05,10:PLOT 104,112:DRAW 0,32:BEEP .05,10:PLOT 88,128:DRAW
      16,0:BEEP .05,10
2529 PAUSE 30
2530 PLOT 116,128:DRAW 8,0:BEEP .05,10
2531 PAUSE 30

```

```

2532 PLOT 136,112:DRAW 0,32:BEEP .05,10:DRAW 16,0:BEEP .05,10:PLOT 136,128:DRAW 10,0:BEEP
    .05,10
2533 PAUSE 30
2534 PLOT 176,112:DRAW -16,0:BEEP .05,10:DRAW 0,32:BEEP .05,10
2535 PAUSE 30
2536 PLOT INK 6;194,144:DRAW 0,-32,5*PI/6:BEEP .05,10:DRAW 0,32,5*PI/6:BEEP .05,10
2537 PAUSE 30
2538 PLOT 214,144:DRAW 8,-32:BEEP .05,10:DRAW 8,16:BEEP .05,10:DRAW 8,-16:BEEP .05,10:DRAW
    8,32:BEEP .05,10
2540 FOR N=13 TO 21
2545 PRINT INK 2;AT N,3;"EF EF EF EF EF EF EF"
2550 PRINT INK 4;AT N,2;"F";AT N,29;"E"
2555 NEXT N
2558 LET Q=5
2560 FOR N=0 TO 3:PRINT INK 1;AT 12+2*N,Q;" ";AT 13+2*N,Q;" ";AT 14+2*N,Q;A$;AT
    15+2*N,Q;B$:BEEP .05,Q:NEXT N
2561 FOR N=0 TO 2:PRINT INK 1;AT 12+2*N,Q;" ";AT 13+2*N,Q;" ";AT 14+2*N,Q;A$;AT
    15+2*N,Q;B$:BEEP .05,Q:NEXT N
2562 FOR N=0 TO 1:PRINT INK 1;AT 12+2*N,Q;" ";AT 13+2*N,Q;" ";AT 14+2*N,Q;A$;AT
    15+2*N,Q;B$:BEEP .05,Q:NEXT N
2563 PRINT INK 1;AT 14,Q;A$;AT 15,Q;B$:BEEP .05,Q
2565 LET Q=Q+4:IF Q<29 THEN GO TO 2560
2570 IF Q=29 THEN PRINT FLASH 1;BRIGHT 1;INK 1;AT 10,1;"ВАМ -НУЖНА ИНСТРУКЦИЯ? ДА - <Y>"
2575 PAUSE 0:IF INKEY$="Y" THEN GO TO 2600
2580 RETURN
2600 CLS
2610 PRINT INVERSE 1;AT 1,10;" CASH-FLOW "
2615 PRINT AT 3,0;"
    ВАША ЦЕЛЬ - СОБРАТЬ 5 МОНЕТ В
    ОДНОМ МОНЕТОПРИЕМНИКЕ. В ЭТОМ
    СЛУЧАЕ ВСЕ ОНИ ДОСТАНУТСЯ ВАМ."
2620 PRINT AT 6,0;"
    У ВАС ЕСТЬ 6 МОНЕТ ПО 10 ПЕН-
    СОВ ПЛЮС ВСЕ, ВЫИГРАННЫЕ ВАМИ."
2622 PAUSE 100:PRINT INK 6;PAPER 2;FLASH 1;AT 9,4;" НАЖМИТЕ ЛЮБУЮ КЛАВИШУ "
2624 PAUSE 500:PRINT AT 9,4;"
2628 PRINT INK 6;PAPER 2;AT 9,9;" УПРАВЛЕНИЕ "
2630 PRINT AT 11,0;"А - ВСТАВИТЬ МОНЕТУ В АВТОМАТ";AT 13,0;"Z - ЗАПУСТИТЬ МОНЕТУ"
2635 PRINT AT 15,0;"О И Р - ПОДТОЛКНУТЬ МОНЕТУ ВЛЕВО ИЛИ ВПРАВО (ТОЛЬКО 1 РАЗ)"
2640 PRINT AT 18,0;"
    Q - ПРЕКРАЩЕНИЕ ИГРЫ (ТОЛЬКО ПО-
    СЛЕ ТОГО, КАК ВЫ ИЗРАСХОДУЕТЕ
    ПЕРВЫЕ 50 ПЕНСОВ"
2642 PAUSE 100
2644 DIM I$(160):PRINT PAPER 6;AT 3,0;I$
2650 PRINT INK 2;FLASH 1;AT 5,3;" НАЧИНАТЬ ИГРУ? ДА - <Y> "
2660 IF INKEY$="Y" THEN RETURN
2670 GO TO 2660
3000 RESTORE :LET H=0
3005 POKE 23675,88:POKE 23676,255
3010 LET U=PEEK 23675+256*PEEK 23676
3020 READ J:IF J=.5 THEN RETURN
3030 POKE U+N,J:LET N=N+1:GO TO 3020
3040 DATA 7,31,63,112,119,228,239,231,224,248,252,14,230,247,247,247,231,224,231,97,112,
    63,31,7,247,247,231,198,14,252,248,224
3050 DATA 127,127,127,127,127,127,127,127,254,254,254,254,254,254,254,254,.5

```

Программа русифицирована по методике, описанной выше. Автостарт программы - со строки 2, где происходит загрузка символьного набора. В строке 3 происходит включение символьного набора, а в строке 4 - формирование банка символов UDG-графики при помощи подпрограммы 3000. Формируются только символы "А" -"F", так как только они применяются в программе. После этого происходит старт программы с начальной строки. Так Вы будете запускать программу при остановке во время отладки: командой RUN.



Фактическое начало выполнения программы - строка 10. Здесь происходит принудительное включение регистра CAPS LOCK для того, чтобы упростить процедуру опроса нажатых клавиш при помощи INKEY\$. Например, чтобы проверить нажатие клавиши "А", обычно применяется конструкция:

```
IF INKEY$="a" OR INKEY$="A" ...
```

Включив CAPS LOCK, можно исключить первую половину этого выражения, так как невозможно будет нажатие "а".

В строке 15 обнуляется счетчик суммарного выигрыша на протяжении нескольких игр, проведенных подряд. Далее задается изображение верхней и нижней половины монеты при помощи символов UDG-графики. В строке 25 выполняются действия, связанные с прорисовкой титульной страницы игры при помощи подпрограммы 2500. В строке 2500 фактически выполняется очистка экрана, но, в отличие от CLS, это происходит плавно, для более приятного зрительного эффекта. Строки 2520 - 2538 воспроизводят название игры, причем это делается довольно оригинальным способом.

Строки 2540 - 2555 прорисовывают монетоприемники, а строки 2560 -2565 заполняют их монетами.

В строке 2570 начинаются действия, связанные с выводом на экран инструкции игры. Для этого надо нажать "Y". Если это зафиксировано, то подпрограмма переходит на строки 2600 - 2670 для вывода краткой инструкции по игре. Если нет, то подпрограмма завершается и выполняется возврат на продолжение выполнения программы со строки 30.

Строки 30 - 95 обеспечивают прорисовку игрового поля. 35 -голубые полосы слева и справа экрана. 40 - 65 - штыри для изменения направления падения монеты. 75 - 90 - монетоприемники. 95 -направляющая для движения монеты.

В строке 100 обнуляются счетчики выигрыша и проигрыша, затем происходит переход на строку 1000. Здесь монетоприемники заполняются монетами случайным образом. Ну не совсем случайным, все-таки имеются некоторые ограничения. В крайних монетоприемниках задается не более двух монет, а в третьем и пятом их количество может достигать четырех. Количество монет в монетоприемниках 1-6 определяется соответственно переменными А-F. Так как в 4-м монетоприемнике не может быть монет по условиям игры, то переменная D не задается.

Строка 1055 задает Ваш "начальный капитал". Далее выводится предписывающая надпись и автомат ждет, когда Вы опустите монету в автомат. Если Вы это сделаете (нажатием "А"), то программа переходит на строку 1100.

В строках с 1100 происходит движение монеты по направляющей, проходящей внутри автомата. Теперь Вы должны в нужный момент сбросить ее вниз, нажав "Z". Если Вы не сделаете это вовремя, пока она не докатится до конца направляющей, то Вы ее просто потеряете. При нажатии "Z" программа переходит на строку 1200. Это начало падения монеты. Дистанция по вертикали, которую монета прошла от верхней направляющей, обозначена переменной М. При достижении М=2, строки с 1300 обеспечивают "закрывание щели", через которую монета провалилась в автомат. Далее программа переходит на строку 135.

В строках 151 и 152 происходит проверка нажатия клавиш "О" или "Р", при помощи которых можно подтолкнуть монету. Обратите внимание на условия, при которых возможно подталкивание монеты. Это, прежде всего, флажок, запоминающий факт подталкивания - I. Монету можно подтолкнуть только один раз. Кроме того, это дистанция, на которой монета находится от верхней направляющей. Если монета упала слишком глубоко (М больше или равно 4), то подтолкнуть монету уже не удастся. Факт подталкивания монеты сопровождается коротким звуковым сигналом. Строка 160 обеспечивает прорисовку монеты на новом месте. Строка 166 определяет скорость падения монеты.

При падении монеты, когда она достигнет уровня монетоприемников (что определяется по величине дистанции М), программа переходит на строку 500. В этих строках вычисляется, сколько же монет стало в том монетоприемнике, куда упала монета. Если там к этому времени уже находилось 4 монеты, то программа переводит на строку 600. Если монета упала в четвертый монетоприемник, то она пропадает (строка 565). Если это

была последняя монета, то переход на финал игры, строку 2000 (определяется в строке 568). Строка 570 обеспечивает продолжение игры.

Если в монетоприемнике оказалось 5 монет, то программа продолжается со строки 600. Здесь Вам достаются все выигранные 5 монет, а монетоприемник освобождается. В строках 620-625 обнуляются счетчики монет в соответствующем монетоприемнике. Со строки 630 все действия повторяются.

Финальная часть программы - это строки с 2000. Здесь производится расчет результата игры - разницы между израсходованными и выигранными монетами, расчет суммарного результата по итогам нескольких игр и вывод всех этих данных на экран. В строке 2060 Вам предлагается продолжить игру. В том случае, если Вы согласитесь, нажав "Y", игра будет продолжена.

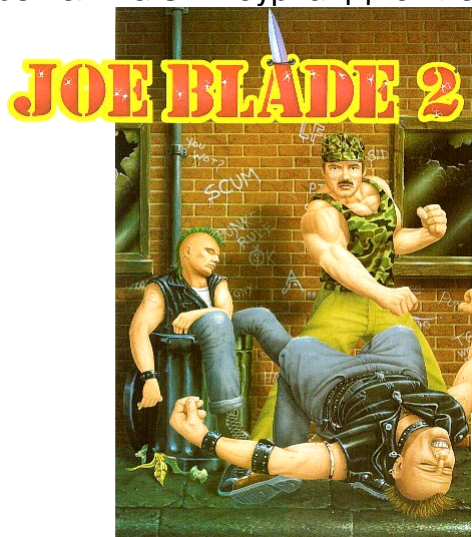
Начиная набивать программу, наберите строки 1-3. Потом сделайте RUN 2 и загрузите заготовленный русско-латинский символьный набор. После его включения продолжайте набивать программу. Имена переменных набирайте в режиме CAPS LOCK. Обратите внимание, что символы "А" - "О" в строке 20 набираются при помощи графического регистра. То же относится и к символам "Е" - "F" в строках 80, 85 и 2545, 2550. Периодически выполняйте RUN 5 для записи результатов работы на магнитофон.

# Компьютерная новелла

Матвеев Ю. А.

## ОТВЕТНЫЙ УДАР КРАКСА

(по игре Колина Свинбурна "Джо Блэйд-2").



Было около полудня, когда в квартире Джо Блэйда резко зазвонил телефон. Разгоняя остатки сна, Джо успел пару раз перевернуться с боку на бок прежде, чем дотянулся затекшей рукой до холодной трубки телефона.

- Говорите, - прохрипел он сонно.

- Все спишь? - после небольшой паузы спросил грубый незнакомый голос.

- Нет... По телефону разговариваю. - Джо почесал щеку и подумал о том, что побриться сегодня не помешало бы. - А кто это?

На другом конце провода послышался смех. Джо уже хотел швырнуть трубку, как вдруг услышал фразу, которая окончательно привела его в чувство:

- Смотри, а то смерть проспишь, щенок...

Незнакомец отключился. Джо положил трубку, потянулся и сел на кровати.

Так с ним никто еще не шутил. Джо понял, что дело серьезное и, прежде чем переходить к активным действиям, решил принять душ и хорошенько все обдумать.

Еще вчера вечером звонили из мэрии и просили подъехать на прием завтра к часу дня. Джо хотел, чтобы его соединили с мэром напрямую, но секретарь отказал. Там шло совещание.

С тех пор, как Джо взорвал тюрьму Кракса, мэр практически исчез из поля зрения. Он видел его лишь один раз на церемонии награждения. Мэр приколот на грудь Джо Блэйда медаль Почета, пожал руку и, как-то вяло поблагодарив, стал поспешно зачитывать фамилии сотрудников безопасности, получивших благодарность.

Церемония проходила в узком кругу доверенных лиц и прессой не освещалась. Отчасти поведение мэра объяснялось тем, что Краксу удалось тогда выкрутиться и отделаться легким испугом. Он не потерял места в городской Думе, а также остался президентом военно-промышленной компании. Его влияние в городе, старые связи с ушедшими в отставку после той истории, по-прежнему позволяли сохранять в своих руках основные рычаги власти.

Но Кракс стал осторожнее, он на время ушел в тень, стал реже мелькать на телеэкранах и страницах газет. Было очевидно, что рано или поздно мэру придется туго, ибо Кракс никогда не забывал обид. И вчера подготовка к ответному удару, видимо была закончена. Иначе зачем мэру опять потребовался Джо Блэйд? Чем он мог помочь?

Приняв душ и перекусив на скорую руку, Джо подошел к телефону и набрал номер приемной мэрии. Пока в трубке шли длинные гудки, Джо закурил сигарету.

- Вас слушают.  
- Здравствуйте. Говорит Джо Блэйд. Соедините меня с мэром.  
- Это невозможно. Идет совещание.  
- Послушайте, я сегодня... В трубке что-то щелкнуло и Джо услышал голос мэра:  
- Это ты? Боюсь, что мы опоздали.  
- В чем дело? - Джо выдохнул густое облако дыма.  
- В городе беспорядки. Толпа каких-то молодчиков шатается по улицам, крушит витрины магазинов, терроризируют население... Полиция бездействует.  
- Почему? - спросил Джо, догадываясь, что главный виновник происходящего уже обо всем позаботился.  
- Силы моих людей небезграничны, а остальные не в моем подчинении...  
Джо заметил, что мэр перешел на шепот и старается не называть имен.  
- Он рядом?  
- В зале заседаний. Там все слышно. Я вышел позвонить, а тут ты... - Мэр тяжело вздохнул.  
- Что же вы теперь решаете?  
- Они предлагают ввести чрезвычайное положение, чтобы навести порядок.  
- Я так и думал. Но вы, конечно, не соглашаетесь?  
- Голосов поровну и пока ничья. А ничья в мою пользу. Теперь они требуют собрать расширенное заседание Думы и я не ручаюсь за исход битвы. Им будет достаточно перевеса в один голос и тогда он станет командантом города на время действия чрезвычайного положения.  
- Сколько еще придет горожан на расширенное заседание?  
- Шестнадцать человек.  
- И какое у них настроение?  
- Ни за что не ручаюсь. Я доверяю им, конечно, но где гарантия, что Кракс не подкупил и их.

Джо Блэйд посмотрел на часы.

- Когда начнется заседание?  
- Через час... - обреченно ответил мэр.

Джо Блэйд задумался. Что можно сделать в этой ситуации? Выловить заседателей где-нибудь на подходе к мэрии и объяснить обстановку? Они сочтут это за прямое давление и, чего доброго, согласятся с Краксом. Разогнать всю толпу хулиганов? Одному это вряд ли по силам. Для того, чтобы хоть что-то сделать, нужно знать намерения Кракса Бладфингера в отношении этих шестнадцати горожан. Они, конечно, "людишки" для него, но их голоса сейчас стоят очень дорого.

Мысли Джо прервал тихий голос мэра:

- Я видел краем глаза на столе Кракса пакет уже готовых документов и распоряжений. Среди прочих, там есть приказ о твоём аресте...

Эти слова мэра Джо оценил как руководство к действию,

- Понятно, - сухо ответил он. - До свидания.

- Что собираешься делать? - спросил мэр, но ответа не дождался: Джо бросил трубку.

Еще с минуту он собирался с мыслями, потом подошел к окну. Джо жил в стареньком трехэтажном домике на втором этаже. Окна его квартиры выходили на узкую серую улочку. Обычно многолюдная, улица сегодня была абсолютно пустынной. Только в самой ее конце у витрины галантерейного магазина наблюдалось какое-то движение.

Джо быстро оделся и заглянул в шкаф, где на верхней полке среди прочих вещей поблескивал хромом распылитель. Решив, что с оружием он привлечет к себе лишнее внимание, Джо не стал брать его с собой. Спустя минуту он уже стоял под окнами своего дома.

С другого конца улицы доносились крики и хохот. Потом Джо услышал звон разбитого стекла и чью-то брань.

Человек пять или шесть шли прямо к нему, разбивая витрины магазинов и кидая в окна домов камни. Им, видимо, было очень весело и свободно на этих пустынных улицах, где

они чувствовали себя хозяевами. Все как один были пьяны.

Джо двинулся им навстречу. Его, казалось, не замечали.

Какой-то панк, высоко подпрыгнув, попытался разбить уличный фонарь, но не достал, а пролетев вперед пару метров чуть не столкнулся с Джо.

Окинув его стеклянным взглядом, и видимо приняв за своего, панк пошел дальше.

"Это хорошо", - подумал Джо и буквально нокаутировал ударом ноги в челюсть подошедшего к нему наркомана.

Никто на действия Джо не прореагировал. Вся группа прошла мимо, пребывая в каком-то экстазе разрушения.

"Без наркотиков здесь не обошлось", - подумал Блэйд, глядя им вслед.

Развернувшись, он быстро пошел по пустынной улице в сторону центра города, где заходило здание мэрии. Джо надеялся перехватить заседателей на подходе.

По пути он встретил еще одного панка, видимо отбившегося от той шумной группы. Ударом ноги в прыжке, Джо надолго успокоил наркомана.

Неожиданно он заметил у стены дома на тротуаре небольшой круглый предмет с циферблатом на лицевой стороне, очень похожий на будильник. Прибор вел обратный отсчет времени, все это было до боли знакомо. Джо покопался в собственной памяти. Ну конечно! Это был военный таймер-универсал, имевший свою передающую антенну и способный задействовать любой электронный прибор на расстоянии. Может применяться в самых различных целях.

Как оказалась эта штука на улице? Какой механизм она должна привести в действие через девять минут, если верить ее показаниям?

Джо не нашел ответа на эти вопросы. У него возникло желание вдребезги разбить таймер, но, подумав, он нажал кнопку на задней стенке прибора, установив тем самым счетчик времени на первоначальное значение. Что бы ни должно было произойти, Джо оттянул время еще на десять минут. Этот отрезок времени был задан программой таймера.

Без сомнения, Кракс затеял опасную игру и теперь во что бы то ни стало нужно было помешать ему достичь своей цели.

Джо подошел к двери подъезда и толкнул ее. Он рассчитывал выйти с черного хода на другую улицу, чтобы не обходить кругом целый квартал. Однако, дверь была заперта. Жители города в панике закрылись в своих домах в надежде спастись от буйных панков. Никто не знал, что это может помешать Джо осуществить его замысел.

Впрочем, у Джо Блэйда всегда имелось в запасе штук двадцать всякого рода отмычек. Одной из них он и воспользовался.

Джо вышел на соседнюю улицу. Здесь так же бесчинствовали панки, алкоголики и наркоманы. Нескольких, самых буйных, Джо завалил своим коронным приемом. Во всеобщей неразберихе никто ни на что не обращал внимания.

Пройдя вперед несколько метров, Джо заметил какого-то гражданина почтенного возраста в сером плаще и черной шляпе. Он явно куда-то торопился, однако ему все время приходилось прижиматься к краю тротуара, чтобы пропустить пробегающих мимо молодчиков. Джо подошел, к нему сзади и тихо, чтобы не испугать человека, окликнул:

- Здравствуйте, я от мэра.

Человек в плаще вздрогнул и обернулся. Это был один из членов городской Думы. Джо знал его в лицо. Такие люди часто появляются на экране телевизора и спутать их с кем-нибудь невозможно.

- Ваш мэр, - нахмутив брови ответил старичок. - Просто полный идиот... Я не хочу с вами разговаривать.

- Постойте, - Джо схватил его за руку. - Вы же всегда были за мэра, что случилось?

- А сейчас против, и дай Бог Краксу сил и здоровья!

Джо еле сдержал себя, чтобы не съездить уважаемому члену городской Думы по физиономии, но он заставил себя быть вежливым и лишь сурово посмотрел на собеседника.

- Да! Я буду голосовать за Кракса Бладфингера! - завизжал старик. - Только он может навести порядок в этом городе. Все! Разговор окончен!

Джо нехотя отпустил руку. В чем дело? Откуда такая ненависть? Неожиданно в голову

пришла сумасшедшая мысль. Перед глазами у Джо проплыл таймер, найденный на улице и картина стала резко проясняться.

- Минуточку. - Джо быстро заломил руку заседателя за спину и, пока тот сопротивлялся и пытался, в темпе проверил его карманы. И он не ошибся. В левом кармане плаща лежал небольшой, со спичечный коробок, предмет. В одно мгновение он оказался в руке у Джо.

Его подозрения оправдались. Кракс Бладфингер занялся электронным гипнозом, чтобы заполнить голоса членов Думы.

Отмахиваясь от озверевшего старичка, как от назойливой мухи, Джо нажал кнопку "ВЫКЛ.", но опять столкнулся с любимой идеей Кракса - секретным кодом. Кракс защищал себя как мог - он предвидел, что кто-нибудь может обнаружить гипнотизатор и тогда, при ошибке ввода кода, гибель нашедшего эту штуковину неизбежна.

Шестьдесят секунд между жизнью и смертью. Джо торопился, нервничал и это чуть его не подвело. Но он сумел взять себя в руки и справился с этой задачей. Приборчик замолк. Джо обернулся, старичок, улыбаясь, стоял сзади и хитро подмигивал.

- Я вас узнал, Джо Блэйд. Мэр должен сегодня победить!

Потом он приложил палец к губам и, наклонив на глаза шляпу, отправился своей дорогой на голосование.

Джо от радости лихо подпрыгнул и одним ударом свалил в сточную канаву сразу двух панков.

Теперь все прояснилось. Джо не сомневался, что люди Кракса Бладфингера каким-то образом сумели подбросить всем шестнадцати заседателям электронные гипнотизаторы

Кракс хотел полной победы и, вполне вероятно, что даже те, кто был настроен к нему лояльно, не избежали этой участи.

Гипнотизатор сильная штука, и, чтобы не убить человека резким вторжением в его мозг, на улицы города, по которым пойдут депутаты, подбросили таймеры, которые поддерживали десятиминутный щадящий режим работы гипнотизатора. По прошествии этого срока процесс становился необратимым и человек уже не смог бы избавиться от наваждения.

Джо мчался по улице, сбивая на своем пути всех хулиганов и наркоманов. Он засекал время и видел, что оставалось шесть минут до окончательного перехода заседателей Думы на сторону Кракса. Шесть минут - если только он не найдет таймер. Как назло двери всех переходов между улицами были заперты. Джо стал подозревать и в этом Кракса. Отмычки кончались со скоростью света и Джо понял, что такое дело нельзя пускать на самотек. На одной из улиц с огромным отвращением он перевернул мусорный бак в надежде найти хотя бы кусок проволоки. К сожалению, ничего подобного он не обнаружил.

Он пересек еще несколько улиц и увидел таймер. Прибор сразу бросился в глаза. Джо подбежал и, уже готовый сбросить его на прежнее время, заметил заседателя городской Думы. Тот неспеша проходил мимо.

Сколько всего таймеров подброшено на улицы города? Джо не находил ответа на этот вопрос. Может быть по одному на каждого заседателя, а может быть всего два, и поэтому он не стал рисковать. Оставив на время таймер, Джо подкрался к заседателю и, не говоря ни слова, принялся его обыскивать. Прибор он нашел быстро и ситуация повторилась. Не обращая внимания на возмущенные крики горожанина, Джо занимался кодом. Надо сказать, что здесь код оказался сложнее и Джо изрядно попотел, прежде чем решил эту задачку.

Очнувшись от чар электронного гипноза, заседатель любезно раскланялся с Блэйдом и заверил его в своей преданности мэру.

Теперь настало время сбросить таймер и отсрочить развязку еще на десять минут. Но Джо не расслаблялся. Все улицы, прилегающие к зданию мэрии, необходимо было тщательно контролировать, чтобы не пропустить ни одного заседателя, а свои отмычки Джо почти израсходовал. Опрокинув пять мусорных баков, Джо пополнил использованные запасы десятью ключами.

Время шло, но по мере того, как Джо встречал все новых и новых членов городской

Думы, его действия приобретали почти автоматический характер: внезапное нападение, гипнотизатор, установка кода, вежливое прощание...

Позже, вспоминая эти сумасшедшие минуты, он улыбался своей мысли о том, что со стороны его поведение могло показаться странным. Это он был загипнотизирован, ибо действовал как робот...

Джо Блэйд вздохнул свободно только тогда, когда шестнадцатый человек ушел на голосование, глядя на мир не через пленку, накиннутую Краксом Бладфингерон, а своими глазами.

\* \* \*

Из протокола N29 экстренного заседания Думы в расширенном составе:

"... считать нецелесообразным введение чрезвычайного положения. С учетом принятого решения по этому вопросу, передать все полномочия на время, необходимое для наведения порядка, мэру города..."

# THE DARK WHEEL

Станция "Кориолис" представляет собой огромный мир, воздвигнутый на шести панелях и заполняющий образованное таким образом обширное замкнутое пространство. На противоположных панелях расположены два крупнейших центра - Саут Сити и Норт Сити. Огни, сверкающие в ночи над головой жителя Саут Сити - это ничто иное, как ярко освещенные дома и улицы противоположного города.

Алекс отметился в регистрационном пункте своего причала и взял воздушное такси. Юркий автоматический корабль аккуратно скользнул между громадами прибывающих и отходящих кораблей. В восторженном настроении Алекс наблюдал, как из серой дымки над ним проступали массивы городских строений Саут Сити. Слева от него очень хорошо просматривались улицы и парки другого конгломерата, известного под названием Коммандер Сити. Этот город располагался как раз напротив входа на станцию и традиционно там проживали высокопоставленные чины администрации станции, а также послы и представители других планетных систем. Условия их проживания были особо комфортными. Этот город имел специально сконструированный ландшафт с озерами, реками и парками. Здесь были даже склоны для катания на горных лыжах, покрытые настоящим снегом.

"Немезида" под ним превратилась сначала в бесформенную темную колючку, а потом и совсем исчезла из виду, растворившись на фоне посадочной платформы, а над ним, словно гигантские сталактиты, нависли городские массивы Саут Сити.

Кораблик развернулся и после мгновенной потери ориентации все опять стало на свое место. Дома оказались внизу, а "Немезида" стала едва заметной точкой в темном небе. Такси мягко спустилось на уровень городских улиц между двумя монолитными сооружениями. Вокруг сверкали разноцветные огни и казалось, что тонкий слой атмосферы, покрывавшей город, сам туманно мерцал вместе с этими огнями.

Улицы ночного города были запружены толпами народа и Алекс быстро понял, что на этой станции Саут Сити был своеобразным злачным местом. Здесь, по-видимому, процветала торговля всевозможным запрещенным товаром - рабами, роботами, наркотиками, сенсостимуляторами и замороженными органами. Инопланетники медленно и осторожно продвигались по улицам: многие из них были одеты почти в полный космический костюм, что само по себе указывало на небезопасность окружения. Здесь и там громоздились рекламные объявления, обещавшие все мыслимые и не мыслимые, в большинстве своем запрещенные удовольствия. Высоко в небе, среди рекламных кораблей парили несколько полицейских катеров. Улицы города были заполнены толпами, суетой и грязью.

Темный куб Комплекса Магеллана расположился среди этого хаоса, как огромный монстр. Никаких окон снаружи не просматривалось. Здесь и там по стенам скользили наружные лифты. При виде их медленно перемещавшихся зеленых огоньков казалось, что это сооружение - живое.

Алекс прибыл сюда без оружия и уже начал об этом жалеть. Практически все, кого он видел, были вооружены, несмотря на то, что ношение оружия в пределах орбитальных станций было запрещено. Он аккуратно пробирался сквозь толпу рептилиоидов, амфибиоидов, вооруженных инсектоидов, гротескных робо-танков, похожих то на гигантских моллюсков, то на червей.

Он вошел в здание и здесь ему остро ударил в нос отвратительный запах, объединивший в себе выделения тысяч различных живых форм. Бывают, правда, случаи, когда существа, питающиеся жидким метаном, потеют эфирами с запахом сирени, но увы так бывает далеко не всегда.

Частный торговый центр представлял собой огромный зал, вокруг которого расположились конторы и склады. В этом зале, запруженном толпой, продавались как правило такие товары, которые выставить на открытом рынке было бы слишком рискованно



или вследствие своей изощренной специфики они все равно не нашли бы там покупателя. Во всяком случае, торговые корабли, загружавшие свой товар здесь, немедленно регистрировались официальными службами и более жесткий таможенный мониторинг экспортной службы перед отлетом им был обеспечен. Более того, многим из них был уготован гораздо более плотный контакт со спецслужбами в порту назначения, чем им хотелось бы.

Алекс внимательно пригляделся к стенам в поисках вывески складов Мак Греви. Наконец, он отыскал ее и направился ко входу, когда путь ему преградили два высоких и довольно решительно выглядевших инсектоида. Их тела были покрыты светло серым панцирем, а фасеточные глаза немигая следили за Алексом. Они о чем-то совещались, пощелкивая и потрескивая в своеобразной манере общения. Алекс шагнул в сторону, сердце бешено забилося и кровь хлынула к голове - "... Фасеточные глаза, шарнирные конечности, антенны на голове, двухрядные челюсти... - Таргоиды! - эта мысль мгновенно родилась в его голове. - Таргоиды, здесь на космической станции?!".

Таргоиды были смертельно опасны. У пилотов их кораблей удалялись железы, вырабатывающие вещества, которые вызывают чувство страха. Это совершенно бесстрашные и безжалостные противники. Абсолютно негуманоидные, они были самыми злобными и непримиримыми врагами людей в известном космосе. Награда за уничтожение таргоида была огромной, но еще выше объявленная премия тому, кто доставит в исследовательский центр образцы форм детенышей таргоидов - тарглетов.

Но что могут делать таргоиды здесь, на космической станции? Оба чудовища продолжали беседу, холодно рассматривая Алекса. Алекс обратил внимание на придатки пластины, закрывающей грудную клетку, за которой таргоиды хранят свои ручные лазерные пистолеты.

- Назад, - зашептал голос ему на ухо и Алекс обернулся. Мак Греви стоял рядом. Только сейчас Алекс узнал о его невысоком росте - он едва доставал ему по грудь.

- Таргоиды... - прошептал Алекс.

- Ерунда, сказал Мак Греви и потянул Алекса в сторону. - Это оресрианцы и единственная их опасность - в том, что они очень похожи на таргоидов и их часто путают точно так же, как спутал их и ты. А таргоиды, кстати, их злейшие враги. В следующий раз обращай внимание на пластину, закрывающую грудную клетку и на форму четвертого сустава задней ноги, прежде чем делать поспешные выводы.

Алекс с облегчением последовал за Мак Греви, прочь от продолжавших шептаться странных существ.

Склад Мак Греви был небольшим, до предела загроможденным и очень вонючим. Алекс прошел в тускло освещенный коридор и почувствовал явный дискомфорт, когда Мак Греви закрыл входную дверь. В нескольких больших и прозрачных клетках шевелились и урчали странные создания, обеспокоенные неожиданным вторжением.

Это то, что ты хотел мне предложить? - низким голосом промолвил Алекс. Мак Греви кашлянул. Он подошел к одной из клеток и включил свет, более ярко высветивший то, что находилось внутри.

Алекс застыл. Создание выглядело явно знакомым, но память отказывалась служить. У животного был плотный панцирь, аккуратно разделенный на ячейки. Из образованного панцирем костяного домика через регулярно расположенные отверстия высывались конечности. Втянув конечности, существо оказалось полностью под защитой своей оболочки.

- Кто это?

- Мимурты,- ответил Мак Греви. - Если тебе они кажутся знакомыми, так это потому, что они выглядят точно так же, как и животные Древней Земли. Кажется, тех называли черепахами или что-то в этом роде. У этих две головы, четыре ноги и еще две каких-то внешних органеллы, которые неизвестно для чего нужны. Их называют по имени той планеты, откуда они родом - Мимурт. Твоя задача - перевезти их на Сираг. У сирагианцев особое отношение к мимуртам.

- Они их едят? - догадался Алекс.

- Они им молятся, - движением губ поправил его догадку Мак Грев.

- Молятся?

Мак Грев кивнул. - Для жителей Сирага мимурты - это нечто вроде живого воплощения их богов. Слышал когда нибудь о реинкарнации? Ну так вот, есть у них такой особый бог Аватар, который возрождается в облике мимурта. Так что мимурты - это живые формы их бога. Мимурты очень похожи на древние представления легенд и мифов Сирага об их боге. Конечно же, мимурты родом с другой планеты и никакого отношения ни к Сирагу, ни к их богам не имеют, но любая сирагская семья отдаст маленькое состояние, чтобы иметь себе мимурта в домашнем храме.

Алекс почувствовал интерес, а успокоенные создания опять начали шевелиться. Из отверстия в корпусе появились и задвигались розоватые конечности. Животные поползли по слякоти, заполнявшей клетку.

- А "маленькое состояние", это сколько?

- Каждый из них потянет на сотню кредитов, может чуть больше. Здесь их двадцать восемь штук. Двадцать восемь сотен, этого тебе хватит на любые защитные экраны и лазеры.

- А почему ты не хочешь доставить их сам?

- С моим-то криминальным списком? Нет уж, благодарю покорно, но я не желаю выходить в космос. У меня теперь другой бизнес. Обычно у меня уходит год, чтобы собрать такую партию, как сейчас, и обычно к этому времени Рейф Зеттер присылает ко мне торговца, нуждающегося вроде тебя в быстром заработке для какого-то дела.

Алекс поймал себя на том, что спокойно и без прежнего отвращения смотрел прямо в изуродованное лицо. Он больше не обращал внимания на пульсацию чужеродной жизни под тонкой кожей человека. Он чувствовал, что должен верить этому знакомому Рейфа, ему даже хотелось верить и все-таки он не мог.

- Сделай мне такое предложение, чтобы я не смог отказаться, - продолжил Мак Грев. И Алекс с огорчением вернулся к обыденной прозе жизни.

- Три сотни, - сказал он.

Мак Грев прокашлялся и покачал головой.

- Нет, парень, ты не понял. Идея вовсе не в том, чтобы обобрать тебя. Наоборот, именно ты должен иметь с этого дела основной доход. Откуда ты его возьмешь, если будешь давать мне за голову в три раза больше, чем получишь сам?

- Я имел в виду... три сотни за всех.

На секунду Мак Грев застыл, уставясь на молодого человека

- Это что, шутка?

- Нет, не шутка. У меня всего лишь триста кредитов. я не тот человек, который вам нужен, мистер Мак Грев.

- Да ты же только что продал груз шанаскильского меха?

- Да, и купил оружие и оборудование. К тому же, закупал я эти меха с потерей, в общем, я вам говорю, что я не торговец, а боец. - Алекс еще раз взглянул на мимуртов и произнес - Я беру восемь штук. Договорились?

- Я продаю все или ничего и мне нужно за них пятнадцать сотен. Рейф Зеттер говорил, что...

- Значит, он был неправ. Поищи себе другого сопляка.

Алекс развернулся, собираясь уходить и со злорадным удовольствием услышал нотки паники в голосе Мак Грева.

- Я берег их для Рейфа. Где я найду еще кого-то, кто будет торговать мимуртами?

- За три сотни я возьму десять штук. Чем больше ты будешь думать, тем меньше я дам.

Торговля начинала нравиться Алексу.

- Но я должен отправить на Сираг именно всю партию.

- Интересно, где этот Сираг, - подумал Алекс. Это название не вызывало в его голове никаких ассоциаций.

- Тогда тебе придется доверить мне так же, как ты доверял Рейфу. Я даю триста

кредов авансом и потом еще треть того, что получу за них на Сираге. Расплатусь, когда вернусь.

Мак Греви молча разглядывал юношу. - Третью вряд ли покроет мои расходы. Пятьдесят процентов.

- Сорок процентов, - сказал Алекс. - Это последнее предложение.

Мимурты опять сосредоточенно завозились в клетках. Мак Греви огорченно пожал плечами и кивнул. Он тут же вызвал по видеоканалам двух свидетелей и в их присутствии контракт был подписан.

Дело обещало быть выгодным для Алекса, если религиозные фанатики этого Сирага раскупят всех мимуртов. Но где же этот Сираг, черт побери?.. Да, теперь "Немезида" будет вооружена лучшими лазерами, снабжена дополнительными энергетическими отсеками и энергетической бомбой. Вот когда начнется настоящая охота!

Алекс вернулся на корабль с докладом о результатах торговли.

## ГЛАВА 7.

Теперь они были на нуле. И, надо сказать, сами играючи залезли в такую петлю. Оставалось только найти Сираг и двигать к нему. Алекс проверил официальный планетный регистр, но Сираг там не значился. Так вот почему название казалось ему незнакомым! Вообще-то невключение в официальный регистр еще ничего не значило. Туда включались только обитаемые планеты. Существовали миллионы планет, интересные только для шахтеров, охотников и добытчиков руд, которые упоминались только в галактическом справочнике. Но ведь Сираг совсем иное дело, это планета с разумной жизнью!

Это означало только одно. Значит, Сираг был независимым миром и не признал Федерацию. Стало быть, это было очень опасное и, может быть, даже смертельно опасное место для посещения. Скорее всего, это рай для пиратов, преступников и прочих бандитов всех мастей. По-видимому, это система, в которой сперва стреляют, а потом разговаривают.

- Мы, кажется, рехнулись... - сказала Элиссия.

Алекс не возражал. - Слушай, а может быть Сираг - это и есть Ракксла? Может быть, его имел в виду мой отец перед смертью?

- Не может быть. Сираг - это Сираг, а Ракксла, если она вообще существует, находится где-то в другой галактике. Ты ведь знаешь легенды. Судя по всему, Сираг - это чертова дыра. Верни этому парню его мимуртов и давай лучше попробуем толкнуть эти окаменелые кости.

Но Алекс отказался. Он чувствовал, что вокруг него что-то происходит. То, как его "направляли" и как им манипулировали, развило в нем непонятную тягу к этому предприятию. В конце концов, его ждали впереди немалые деньги и наконец-то появлялась возможность закончить вооружение "Немезиды" и начать настоящую охоту, начать жизнь мстителя.

- В общем, пан или пропал? Так? И, как говорит Рейф, если нас ждет неудача, то нас не будет, чтобы ее переживать.

- Мы, наверное, сошли с ума... - опять повторила Элиссия.

- По крайней мере, будет нам наука, как общаться с незнакомыми людьми.

\* \* \*

Перед ними парил Сираг, пастельно желтая планета с темными пятнами то ли гор, то ли пустынь, рисунок которых напомнил Алексу о костях. Десять световых лет от Ксезавра "Немезида" покрыла с двумя дозаправками и сейчас в ее баках оставалось горючего на прыжок в два световых года, а ближайшая система находилась вдвое дальше.

Впрочем, теперь это не имело значения. С новыми топливозаборниками им достаточно пройти над солнечной короной и горючего будет достаточно.

Солнце Сирага было большой желтой звездой, достаточно старой, но еще вполне активной. Когда Элиссия, сидевшая у навигационной консоли, развернулась к звезде, два

мощных протуберанца вспыхнули на ее поверхности и устремились в космос. Эти воронкообразные вихри очень красиво смотрелись через поляризованные фильтры "Немезиды".

- Давай немного подзаправимся, - сказала Элиссия и дала полное ускорение, но лететь им пришлось не более минуты.

- Святая Звездная Мать! - Алекс бросил взгляд на экран сканера, и почувствовал, как желудок выворачивается наизнанку. Яркие метки на экране были столь велики, что они могли принадлежать либо крейсерам класса "Боа", либо "Анаконде". Четыре ярких точки, расположенные в боевом строю и рой сопровождающих истребителей не оставляли сомнения о намерениях их экипажей.

- "Боа", - сказала Элиссия. Ведут себя, как боевые крейсера. Хорошо хоть они медленные. Держись...

Алекс вцепился в кресло. Хорошо, что он был пристегнут. Вселенная вздрогнула, а внутренние органы сделали сальто. Элиссия исполнила лихую петлю и строй истребителей, а это были "Мамбы", распался и начал расходиться веером. Это означало погоню. Но Элиссия не останавливаясь сделала еще один вираж и обманула преследователей, перейдя в лобовую атаку.

Спокойно и аккуратно она поднырнула под брюхо ведущего корабля. Казалось, что это было сделано с удовольствием. Противник открыл огонь с нижней полуплоскости и "Кобра" развернулась вокруг оси, приведя к бою бортовые лазеры. Они как раз пролетали под брюхом огромного крейсера.

- Опознавательные знаки незнакомы, - сказал Алекс. Это были черно-зеленые флаги с изображением ярком солнечной вспышки и неземными иероглифами по бокам.

- Зато их намерения очень знакомы, - вздохнула Элиссия. К этому времени две преследующие "Мамбы" завершили маневр охвата и приближались сзади. Вспышки огня лазеров прорезали черное пространство вокруг яркого солнечного диска. Более крупные корабли к этому времени уже тоже развернулись и угрожающе приближались. И Алекс и элиссия прекрасно понимали, что теперь им не дойти до звезды на дозаправку.

Элиссия развернула корабль. Во время разворота она нацелила и выпустила первую ракету. Ближайший истребитель мгновенно исчез в облаке сверкающей пыли. Несколько зарядов, принятых лобовым экраном, заставили их корабль дрогнуть, но два коротких нажатия ловкими пальцами на кнопку бортового лазера остановили второй истребитель и он закувыркался. Элиссия приблизилась для последнего удара...

Убит!..

Из темноты вышел "Боа". Он медленно вращался и лучи лазера играли у носовой надстройки гигантского корабля. Элиссия нацелила еще одну ракету. По ее лицу текли крупные капли пота, пальцы побелели от напряжения. Алекс, чувствуя свою бесполезность, сидел, напряженно вцепившись в кресло, всем телом повторяя маневры Элиссии.

"Боа" уничтожил ракету, едва она прошла десятую часть дистанции. Но "Немезида" вновь скользнула под брюхо огромного корабля и, развернувшись боком, и уравнивая скорости, начала разносить чувствительные участки гиганта огнем из бортового лазера.

Но наконец то, что рано или поздно должно было случиться, произошло. Страшный удар в кормовую часть потряс их корабль. "Немезида" задрожала и начала быстрое беспорядочное вращение. Алекс выругался, почувствовав, как в тело впились пристяжные ремни. Неожиданный удар чуть не оторвал ему голову. С трудом он выпрямился, пытаясь разобраться в ситуации. Сзади приближались две "Мамбы", а неподалеку парила гигантская "Анаконда", готовая их поглотить.

- Мы еще посмотрим кто кого, - громко сказал Алекс и обернулся к Элиссии, недоумевая почему она не маневрирует.

Она обмякла в своем кресле. С головы и носа текла кровь. Она, во-видимому, была слабо пристегнута, когда удар потряс "Кобру" и разбила голову о приборную консоль.

Рывком Алекс покинул кресло второго пилота, освободил Элиссию и буквально бросил ее на пол. Сейчас было не до вежливости. Он круто изогнул траекторию корабля и всадил заряд в распахнутый грузовой отсек "Анаконды". Увернувшись от огня лазера, он тут

же увернулся и от ракеты, сбивать которую уже не было времени. На мгновение диск планеты проскочил перед ним и Алекс развил полную скорость, пытаясь уйти от смертельной опасности.

Во время этой отчаянной попытки оторваться от врагов неприятная мысль поразила его. "А где гарантия, что на этой планете станция будет его защищать, если он окажется в ее зоне?" Такой гарантии у него не было. Вполне возможно, что и станция будет против него. Надо только дать им знать, какой груз он везет. Если они будут знать, что у него на борту груз их обожаемых богов, они вышлют корабли и прогонят пиратов.

Справа неизвестно откуда вынырнула "Мамба". Он развернул "Немезиду" и встретил ее кормовым лазером, сбросил скорость, опять развернул корабль и поразил противника огнем с правого борта. Корабль не взорвался, но закувыркался и вышел из строя.

Ах, если бы он мог бросить груз, выкинуть в космос контейнеры с системой жизнеобеспечения мимуртов и, может быть, все прекратилось бы. Они с Элиссией потеряют на этом три сотни кредитов, ту и что? Они ведь в конце концов еще не "элита".

Новые залпы потрясли корабль. Это опять "Мамба". Алекс нацелил ракету, но запускать ее не стал, открыв огонь бортового лазера.

В это время Элиссия начала приходить в сознание. Она поднялась на ноги и сквозь залитые кровью глаза наблюдала за битвой. Сираг был все ближе и ближе. Маленькая точка серебристого света мигнула им навстречу, но не наполнила Алекса радостью.

- Слушай, в этих контейнерах должно быть не мимурты, - тихо произнесла Элиссия.

- Позже поговорим, - ответил Алекс, маневрируя и выходя из под огня более крупных кораблей противника.

Элиссия покинула мостик. Спасая их жизни, она направилась в грузовой отсек...

И вдруг внезапно атака прекратилась. Алекс чуть не подпрыгнул от изумления. Еще мгновение назад его кормовые экраны едва сдерживали огонь противника. Его бортовой лазер перегрелся от непрерывного огня и вдруг... ничего, тяжелые массы пиратских кораблей отошли назад. Две легких "Мамбы", вцепившиеся в их хвост, выпустили по последнему отчаянному заряду вдогонку и исчезли, промелькнув в солнечных лучах и растаяв в черноте пространства.

Алекс замедлил ход и проверил степень повреждений. К счастью, ничего серьезного. Они лишились двух ракет, низок энергетический уровень, зато груз не поврежден. А раз пираты прекратили преследование, значит Сираг будет защищать своих визитеров.

На мостик вернулась Элиссия, в руках у которой была просвечивающая камера.

- Слушай, эти животные выглядят как черепахи, ведут себя как черепахи и от них воняет как от черепах, но я сделала пару просвечивающих снимков, чтобы посмотреть нет ли там еще чего-нибудь.

- Отличная идея. Давай посмотрим.

- Сейчас, еще две--три минуты.

Она поставила камеру, присела в кресло второго пилота и взглянула на Алекса.

- Ты в порядке?

Алекс кивнул.

- А ты?

- Чувствую себя чертовски разбитой. Мы в безопасной зоне?

- Вроде да.

Перед ними неспешно вращалась станция "Кориолис", ярко освещенная солнцем. Она отбрасывала пятна серозеленой тени на поверхность своей планеты. Неподалеку парили несколько пришвартованных к буям кораблей. Они выглядели достаточно безмятежно. На станции вспыхнули огни. Вскоре все вокруг приветливо засверкало.

Алекс грациозно пролетел сквозь орбитальный город парящих судов и приблизился к точке входа.

Но входа не было!..

- Что за черт?

Он сидел неподвижно. Корабль вращался, уравнивая скорость своей ротации со станцией, но вместо входа перед ними серел металл. Увеличив разрешение приборов.

Алекс все-таки сумел различить признаки входного отверстия, но оно было плотно закрыто.

- Может, они опасаются чужестранцев? - пожала плечами Элиссия.

- Но нам очень нужно топливо. Ну нельзя же нас опасаться до такой степени.

Наконец, раздался треск в устройстве громкоговорящей связи:

- "Назовитесь! Назовитесь! Говорит орбитальная станция планеты Сираг."

- "Немезида", торговый корабль класса "Кобра", - сообщил Алекс. - Мы несем груз мимуртов. Просим открыть шлюз.

На некоторое время последовала тишина, хотя потрескивание в канале свидетельствовало о том, что связь не прервана, и затем:

- Внимание "Немезида"! Торговля мимуртами на станции запрещена!

- Что?

- Избавьтесь от груза перед входом на станцию. Бросьте груз. Вы получите компенсацию.

Алекс взглянул на Элиссию.

- Что же нам теперь делать?

- Мне кажется, что все это выглядит как-то непрофессионально. Что-то здесь есть фальшивое.

Она взяла в руки камеру и извлекла оттуда обработанную пленку. Внезапно она застыла, уставившись в два готовых отпечатка, как будто не в силах оценить то, что увидела и наконец произнесла:

- Бог мой... - с этими словами она передала отпечатки Алексу.

В это время начало постепенно открываться приемное отверстие станции. Два ярких прожектора проступили из темноты отверстия, как два горящих глаза.

Алекс вгляделся в отпечатки и на секунду растерялся при виде необычного, гротескного зрелища. Просветив тела живых мимуртов, камера выхватила какие-то паукообразные формы, живущие внутри этих безобидных черепашек. Зрелище было очень неприятным. Суставчатые конечности протянулись в каждую из конечностей мимуртов, заполняя все внутренне пространство. На темном теле просматривались фасеточные глаза. Два длинных отростка протянулись от него внутрь голов мимурта.

- Что это? - прошептал Алекс.

- Это беда. Это детеныши таргоидов.

Сердце Алекса учащенно забилося. Так это тарглеты?! Значит, он транспортировал тарглетов - личиночную стадию самой опасной жизненной формы в известных галактиках. Он подставлен? Нет, "подставка" - это слишком мягкое слово для того, что с ним сделали на Ксезавре. Чего же теперь удивляться, что пираты накинулись на него столь отчаянно?

- За тарглетов дают неплохое вознаграждение. Военные охотно принимают их для исследовательских целей.

- Пираты выращивают их и используют в качестве пилотов на своих истребителях. Мы привезли на Сираг будущих пилотов пиратских кораблей. Понятно, что они на нас накинулись. Им нельзя оставлять свидетелей.

Алекс уставился на станцию. На какое-то время слова Элиссии пролетели мимо и никак не отложились в его голове. Он задумался о пиратах, атаку которых они отбили, он думал, что все позади и раз они находятся рядом со станцией, то бояться больше нечего, кроме обвинения в торговле запрещенным товаром. Да, он думал, что они в безопасности.

Два желтых глаза выскользнули из черной утробы станции, а за ними потянулось и само темное тело корабля. За ним вспыхнул свет и темная тень незнакомого судна упала на "Немезиду". Змеиная тень, тень "Кобры".

Этот корабль он узнал бы где угодно. Прошли месяцы после встречи с ним, но не было ни одной ночи, когда бы мысль о нем не возвращалась к нему в ночных кошмарах. Корабль, погубивший "Авалонию" медленно приближался и ни малейшего сомнения Алекс не испытывал. Не было сомнений и у Элиссии.

Она затаила дыхание и подалась вперед всем телом. - Отдай мне его. Пусти меня к приборам.

- Сядь. - холодно ответил Алекс и Элиссия зло на него посмотрела.

- У меня с ними большие счета...

- Пилот этого корабля убил моего отца.

- Они убили всю мою семью! Мы бежали с Теорга и обратились к ним за помощью, мы просили немного припасов. Они захватили меня и сестру в качестве рабов, а корабль моего отца расстреляли. Мне удалось бежать, а сестра погибла. Алекс, отдай мне этого ублюдка!

- Сейчас уже поздно...

Вспышка полыхнула со стороны "Кобры". "Немезиду" потрянуло. Алекс нацелил ракету и всадил мощный заряд из носового лазера в противника. Желтым цветком рассеялась энергия на защитных экранах "Кобры".

Она ускорила свой ход. Алекс тоже прибавил скорость, но завис над станцией и противником.

"Мы не можем сражаться с ней. У нас пока нет ни достаточного вооружения, ни средств защиты. Пока еще нет! Что же делать?" - проносились мысли в его голове.

На заднем экране Алекс видел контур мрачного убийцы, выплывающего из-за орбитальной станции. Вспыхнул сигнал приближавшейся ракеты и Алекс задействовал систему ЕСМ на ее подавление. Сделав это, он развернул корабль. Оба корабля прошли встречными курсами, разрывая друг друга на части. Два величественных металлических галеона, поливающих друг друга огнем. Снова разворот и очередное сближение.

Дважды они схватывались в такой дуэли. "Немезида" стонала под тяжестью лазерных ударов. Энергетические отсеки начали сдавать и вновь сомнения захватили Алекса. Эта "Кобра" знает, кто перед ней и не даст ему уйти. Он тоже хочет ее смерти. Но ведь он еще не готов! Нет, еще нет!

Так, несмотря на упреки Элиссии, Алекс развернул корабль и направил его к звезде. "Кобра" начала преследование. Оба корабля маневрировали, сплетая в петли траектории полета. То тормозясь, то ускоряясь Алекс не упускал ни одной возможности для стрельбы кормовым лазером и это немного сдерживало напор пирата. Алекс сбил еще три выпущенных противником ракеты. Хотелось думать, что весь запас ракет пирата исчерпан, но Алекс не позволял себе расслабиться. Собственная ракета оставалась нацеленной и готовой к пуску. Алекс медлял, сознавая, что ее ждет быстрая и бессмысленная кончина.

Звезда придвинулась. Она сияла в своем огромном великолепии. Температура в кабине "Немезиды" начала постепенно повышаться. Сверхестественными чудовищами, порожденными расплавленным океаном, всплывали вихрящиеся рукава раскаленной плазмы необъятных размеров. К одному из них направил свой корабль Алекс, подготовив топливозаборное устройство.

"Кобра" сделала еще выстрел и вновь заскрежетали силовые экраны. Сражающиеся корабли начали погружение в раскаленную преисподнюю...

- Смотри, оно работает, - сказал Алекс.

Стрелка датчика горючего поползла вверх, по мере того, как топливозаборное устройство всасывало потоки сырой плазмы и конвертировало ее в форму, необходимую для работы гиперпространственных двигателей. "Немезида" скользила по краю огненного океана. Рукава выбросов солнечной короны достигали миллионов миль в длину и тысяч миль в поперечнике. Они свивались в огромные воронки, похожие на водовороты. В центре такой воронки образовывается относительно спокойная, более безопасная и менее раскаленная область.

Туда и направил корабль Алекс. Рубка наполнилась сверхестественным сиянием, солнце светило непереносимым блеском и температура корабля начала катастрофически повышаться. Пламя заиграло на обводах корпуса, застонали силовые экраны.

- Только не долго, - сказала Элиссия. В конце концов и она начала понимать, что они еще не готовы к последней битве с врагом. Сейчас им надо уйти отсюда и побыстрее. До ближайшей звезды шесть световых лет, а у них топлива, судя по показаниям датчика - всего на четыре, но стрелка продолжала подниматься.

Выбрав спокойный участок в океане огня, корабль остановился. Где-то в ярком потоке слепящей плазмы, окружившей их, продолжал свои поиски смертельный враг. По-видимому, здесь они были в безопасности - никакие электронные средства обнаружения не

могли работать в потоке интенсивной радиации, излучаемой солнечной короной. Враг остался без глаз и без ушей.

- Пять световых лет, заряд нормальный. Приготовься к старту. Курс введен...

- Готов, - ответил Алекс. Он старался не думать о возможных последствиях такого дальнего неконтролируемого гиперпрыжка. В первый момент ему пришла в голову мысль попробовать перемещаться малыми скачками, но он решил этого не делать - гиперпривод мог не выдержать многократных запусков.

Алекс развернул корабль и придал ему небольшое вращение, внимательно просматривая окружающее пространство в поисках возможной опасности.

- Пять точка пять световых лет. Еще одна минута. Всего шестьдесят секунд...

- Всего тридцать секунд...

Корабль вибрировал всем корпусом. По лицу Алекса сбегали крупные капли пота.

- Еще двадцать секунд, Алекс и мы полетим.

Едва различимое пятнышко мигнуло на экране сканера, выдавая присутствие пиратской "Кобры" где-то рядом. Она находилась по другую сторону плазменного вихря. Занавес сплошного огня разделил их. "Немезида" и убийца стояли друг против друга, отделенные стеной солнечного пламени.

- Мы готовы, - доложила Элиссия. - Вперед, Алекс! Пошел!

Алекс Райдер отрицательно тряхнул головой. - Нет, пока еще нет...

- Алекс!

Он тронул корабль вперед, в направлении огня. Мерцающее, еле различимое изображение на экране сканера тоже переместилось. Началось сближение.

Дикий крик вырвался из груди Алекса, когда он дал максимальное ускорение кораблю, вложив всю возможную мощь в двигатели. Корабль рванулся вперед навстречу стене огня и плазмы. Исчезли все изображения на экранах, единственное, что видел Алекс перед собой - это лицо отца и яркую вспышку пламени, в котором погибла "Авалония".

Горечь, гнев и ненависть - только эти чувства он ощущал в это мгновение. Он знал, что пока у него есть ракета, уже нацеленная на вражескую "Кобру", у него еще остается один последний отчаянный шанс.

Корабли сближались. Теперь их разделяла только тонкая стена плазменного водоворота. Плазма играла на корпусе "Немезиды" и силовые поля уже не выдерживали напора. Дальше двигаться было нельзя. Глубже - нельзя... опасность!.. И Алекс запустил ракету.

Крошечный снаряд нырнул в солнечное пламя, отыскивая свой путь к цели. Алекс не видел ракету на экране, не видела ее и пиратская "Кобра", пока не стало слишком поздно.

В последний момент "Кобра" включила ЕСМ. За вспышкой света последовала детонационная волна и еще через мгновение ракета превратилась в стремительно вращающийся огненный клубок, вобравший в себя инерцию ракеты, силу взрыва и потоки раскаленной солнечной плазмы. Развивая скорость и мощь, этот клубок рванулся к вражескому кораблю и не было никакой силы защитных экранов, способной остановить этот клубок солнечной энергии. Через мгновение "Кобра" купалась в потоках плазмы, захваченная вихрем протуберанца. Алекс не отрывал глаз от экранов и вдруг... все кончилось. "Кобра" была уничтожена. Она погибла, сгинула навсегда.

Медленно развернувшись, "Немезида" легла на обратный курс. Никто на мостике не проронил ни слова, но в ярких лучах старевшего светила блеснули слезы на двух счастливых лицах.

## ГЛАВА 8.

Галоизображение Рейфа Зеттера дрожало в боевой рубке корабля и гордо светилось. За ним на переднем экране просматривался гостеприимный диск планеты Лейа. Остатки мимуртов с их драгоценными паразитами к этому времени уже были перегружены на два "Эспа" военного флота. Сумма окончательного вознаграждения, правда, еще не была согласована, но получалось не меньше, чем по сотне кредитов за экземпляр.

- Я знал, что ты сможешь это сделать, - сказал Рейф, счастливо пережевывая свою



жвачку и лихо сплевывая ее в сторону. - Я был уверен настолько, что направил тебя на Сираг немного раньше, чем ты окончательно подготовился.

- Но мы могли погибнуть. От этой системы мурашки бегут по коже.

- Настоящий боец, даже если он "Элита", знает, когда надо отступить и как надо отступать. Я горжусь тобой. Ты отступил ... и победил.

В это время пришло сообщение с "Кориолис-6" из штаб квартиры галактической полиции.

"От имени Галактического Содружества Миров поздравляем Алекса Райдера и приносим благодарность за проявленное умение и мужество при уничтожении пиратских кораблей. Согласно представленному рапорту, подтвержденному пленкой бортовой регистрационной V-системы, присваиваем боевой ранг "Смертельный". Ваш правовой статус "В розыске" снимается, новый боевой рейтинг "Смертельный" будет введен в каналы Галактической сети в течение стандартного дня. Желаем быть мудрым и сильным в бою."

Вот так и случилось, что не достигнув и двадцати лет, Алекс оказался всего на шаг от боевого класса, о которой большинство людей даже и не мечтают. Он вошел в класс "Смертельных". Он уничтожил "Кобру", которая неизвестно за что убила его отца. Почему это было сделано, Алекс конечно не знал, а спросить у пилота-убийцы не подумал. Он предполагал, что это было просто заказное убийство, за которое бандит получил вознаграждение.

Вместо этого, он спросил Зеттера: - Ты знал, что этот корабль на Сираге?

- Предполагал. Алекс, вот почему мы послали тебя с грузом тарглетов. Ты знаешь, никто, абсолютно никто не в состоянии устоять перед искушением захватить такую добычу. Я знал, что ты соберешь всех пиратов в радиусе светового года, но я также знал и то, что ты с ними справишься.

Более того, я был даже уверен, что такая наживка привлечет и эту "Кобру".

Ты бился отлично. Я вижу в тебе те же инстинкты бойца, которые были и в Джейсоне. И он был прав. Ты именно тот человек, который сможет пойти по его стопам.

- Куда пойти по его стопам?

Рейф кашлянул и покачал головой. - Знаешь, это сложный вопрос. Твой отец искал следы мифической Раккслы. Он хотел узнать, существует ли она на самом деле? Если легенды не врут, то там должна быть установка, созданная инопланетным разумом, с помощью которой открывается проход в другие Вселенные, ко всем их сокровищам.

Джейсон Райдер был уверен, что Ракксла реально существует. Вот почему он прошел специальную подготовку и был включен в состав лиги Темного Колеса, лиги легендарных первопроходцев. На несколько лет он исчез и я не имел никаких сведений ни от него, ни о нем до самого последнего времени перед его гибелью. Тогда он рассказал мне, что нашел реальные доказательства существования Раккслы. Он вернулся из глубин космоса, чтобы сформировать команду. - Рейф горько улыбнулся. - Но перед самым отлетом он взял несколько дней для отдыха в спокойном месте вместе с сыном... где его и встретил наемный убийца.

- Но почему? Почему его убили из-за Раккслы?

- Потому, что на Ракксле уже давно люди. Правда, это только догадка, но судя по тому, что они сделали с Джейсоном, она недалеко от истины. Мы давно подозревали, что корпус "Элиты" создал там базу и широко использует переход в иные Вселенные. Это могущественные и решительные люди. У них достаточно средств, чтобы нанять негодяя и уничтожить того, кто несет угрозу их превосходству.

Рейф приблизился чуть вперед, его глаза сияли.

- Я контролировал твои шаги, Алекс, а также и Элисии. Темное Колесо нуждается в вас. В обоих вас. Но поверь мне, то, через что вы уже прошли - ничто по сравнению с тем, что Вас ждет в будущем. Алекс, вы должны стать "Элитой". А это означает многие месяцы, а может быть и годы тренировок и сражений, но после этого Вселенная распахнется перед Вами так, как Вы не можете себе даже представить.

Алекс стоял и глядел на старика в молчании и задумчивости. Наполовину скрытая в тени, в уголке стояла Элиссия, напуганная тем, что только что услышала.

- Ну как, твое горе утихло? - спросил Рейф и Алекс кивнул. Старый торговец улыбнулся.

- Скажи, что ты чувствуешь став богатым?

- Опустошенность. - и Рейф рассмеялся.

- Да, ты подойдешь для Темного Колеса, это точно!



"ИНФОРКОМ" 121019, Москва, Г-19, а/я 16

Вниманию читателей!

Мы продолжаем публиковать адреса пунктов, в которых можно приобрести наши материалы:

г. Воронеж. Студия компьютерных игр SAN-SAN. Магазин-салон "Электроника". Тел. 14-00-73.  
г. Днепропетровск, ул. Шевченко, 34. Фирма "ЭКОС".  
610014, г. Киров, ул. Производственная, д. 27, "Дом науки и техники". Магазин-салон "МАРС".  
г. Москва, Новый Арбат, д.2, 19-ое отделение связи, 1-ый этаж операционного зала. Ежедневно, кроме выходных, с 10 до 17 часов. Перерыв с 14.00 до 15.00.  
г. Нижний Новгород, ул. Горького, 146. Маг. "ФОТОЛЮБИТЕЛЬ". ИМА "Ф-ПЛЮС". Тел. 35-07-18.

Вниманию дистрибуторов! Адреса наших оптовых покупателей публикуются бесплатно.

## СПЕКТРУМ В ШКОЛЕ

Дорогие друзья! Вам никогда в детстве не приходилось играть в игру "Холодно-Горячо"? Это когда один играющий прячет в комнате какой-то предмет, а другой пытается его отыскать по сообщениям типа "Холодно!" - "Холодно" - "Теплее" - "Горячо" - "АФРИКА!!!".

Вполне серьезная игра для научных работников всех специальностей, имеющих дело со сложными алгебраическими уравнениями и с системами таких уравнений. Вы пока учитесь в школе, и Вам может быть рано решать проблемы численных методов математической физики, но сыграть в такую игру с компьютером Вы вполне можете.

Давайте задумаемся, ведь в любом алгебраическом уравнении "спрятан" его корень (если он есть). Если вместо неизвестного  $X$  в уравнение подставить этот корень, то уравнение обратится в ноль. А если, подставить не корень, а другое число? Тогда оно в ноль не обратится, а будет равно какому-то числу " $d$ " и чем больше эта "невязка"  $d$ , тем "холоднее" Ваш ответ. Вот так играя мы можем решать достаточно сложные уравнения.

Возьмем простенькое уравнение:

$$5x^3 - 3x - 70.625 = 0$$

И попробуем "угадать" его решение. Подставим вместо  $x$  единицу.

$$d = 5 \cdot 1^3 - 3 - 70.625 = -68.125$$

Пока "холодно", подставим  $x=2$ .

$$d = 40 - 6 - 70.625 = -36.625$$

Это уже "теплее", проверим теперь  $x=3$ .

$$d = 5 \cdot 27 - 15 - 70.625 = 49.375$$

Надо же - опять "холодно", но обратите внимание на то, что наша "невязка"  $d$  изменила знак. Раньше она была отрицательной, а теперь стала положительной. Это говорит о том, что мы проскочили мимо спрятанного корня и ушли не в ту сторону. Надо вернуться назад и искать более тщательно, пойдем от числа  $x=2$  более аккуратно. Вместо шагов по единице будем шагать по 0.1.

Проверим  $x=2.1$ ,  $x=2.2$  и так далее, пока не дойдем до  $x=2.5$ . Вот это и будет "АФРИКА" - точное решение, при котором "невязка" равна нулю.

К сожалению, точное решение возможно только для таких простых уравнений, как наше. А что делать, если уравнение посложнее? Например:

$$2x^{0.8} - \ln(3x) - 1.7 = 0$$

Оно не имеет точного решения, да ведь оно нам и не нужно! Давайте зададим величину "b", которая является предельно допустимой "невязкой", и будем бегать, подставляя X, взад-вперед, уменьшая шаг в десять раз после каждого изменения знака "невязки" до тех пор, пока она не окажется меньше ранее заданного числа "b".

Как видите, все очень просто. Конечно, с карандашом на бумаге двадцать раз вычислять одно и то же выражение занятие не из приятных, ну а для чего нам компьютер? Он сам и подставит значение "x" и подсчитает невязку "d" и сравнит ее с предельной величиной "b" и, если предел не достигнут, изменит x, снова все посчитает, проверит знак "невязки" и т.д. Этот прием очень просто алгоритмизируется.

```
10 INPUT x: REM Начальное значение x Вы должны задать сами.
20 INPUT b: REM Предельное значение "невязки". Чем оно меньше, тем точнее Ваш расчет, но тем
    больше времени он потребует.
30 INPUT a: REM Это начальное значение шага по "x". Далее компьютер будет рассчитывать его
    сам.
40 GO SUB 500: REM Расчет "невязки".
50 IF ABS d<b THEN GO TO 800
60 LET m=d
70 LET x=x+a
80 GO SUB 500
90 IF ABS d<b THEN GO TO 800
100 IF SGN d <> SGN m THEN GO TO 600
110 IF ABS d > ABS m THEN GO TO 700
120 GO TO 60
500 LET d =.....
    (Здесь надо записать Ваше уравнение)
510 PRINT "x= ";x, "d= ";d
520 RETURN
600 REM Если мы "проскочили" мимо корня, надо вернуться на шаг назад и уменьшить величину
    шага.
610 LET x=x-a
620 LET a=a/10
630 GO TO 70
700 REM Если "невязка" после шага стала больше, чем была, значит мы пошли не в ту сторону.
    Надо вернуться на шаг назад и изменить знак шага (идти в другую сторону).
710 LET x=x-a
720 LET a=-a
730 GO TO 70

800 STOP
```

Этим методом ученые пользуются уже не одну сотню лет. Он называется методом простой итерации. На его основе разработано еще очень много других полезных методов, позволяющих решать очень большие системы самых сложных уравнений. Если Вы подумаете, то безусловно найдете пути, как улучшить этот метод. Правда, к сожалению, скорее всего может оказаться, что Ваше улучшение уже было кем-то открыто несколько десятков лет назад и уже носит чье-то имя. На всякий случай, если что-то изобретете, то загляните в какой-нибудь математический справочник в раздел, посвященный численным методам.

Желаем успеха!

Ваш "ИНФОРКОМ"

# SINCLAIR LOGO

(Продолжение. Начало см. на стр. 69 - 74)

## Команды "Черепашки".

"Черепашка" в ЛОГО имеет небольшую систему собственных команд. Рассмотрим эти команды.

### **FENCE.**

Обычно, когда "черепашка" доходит до границы экрана, она может продолжать свою работу, выходя за него. При этом если она покинула экран справа, то появляется на экране слева. Если покинула экран сверху, то появляется, соответственно, снизу. Действует эффект автоматического возврата. Это во многих случаях бывает удобным, но не всегда. Если Вы хотите, чтобы Ваша "черепашка" останавливалась дойдя до границ экрана и не возвращалась, то можете дать команду FENCE.

### **WINDOW.**

Команда FENCE отключит возврат "черепашки" с противоположной стороны, но не позволит ей покинуть пределы экрана. Может быть, Вам нужен вариант, когда она может покидать экран в любом направлении без возврата. В этом случае используется команда WINDOW. Представьте себе, что Ваша "черепашка" ходит по огромному столу, а Вы смотрите сверху на этот стол через небольшое "окно" своего экрана размером 255\*176. Покинув экран, "черепашка" продолжает движение в любом направлении. Она может удалиться от центра экрана на расстояние до 32767 шагов.

### **WRAP.**

Этой командой отбивают действие включенных команд FENCE и WINDOW и возвращают "черепашку" к нормальной работе "с возвратом".

### **PENUP.**

Обычно в своих перемещениях "черепашка" оставляет вычерчиваемый на бумаге след. Это можно устранить, дав команду PENUP. Перо поднимается и "черепашка" может переместиться в новую точку, не оставляя линию на бумаге.

### **PENDOWN.**

Команда на опускание пера. "Черепашка" опять готова к рисованию.

Если Вы вспомните, что "черепашка" - это как бы маленький робот-манипулятор, установленный на колесной тележке и подключенный к компьютеру, то Вам все станет ясно. С помощью электромагнита по командам от компьютера ее пишущий узел (карандаш, фломастер, шариковый стержень и т.п.) может подниматься или опускаться.

### **PENERASE.**

"Черепашка" на нашем экране - воображаемая и потому она может делать немного больше разных вещей, чем настоящий робот. Так, например, с помощью команды PENERASE можно дать задание на стирание всех линий, которые встретятся на ее пути.

PENREVERSE - еще более хитрая команда. Для "черепашки" это указание стирать все нарисованные линии на своем пути там, где они есть, а там, где их нет - наоборот рисовать.

SETPC - эта команда расшифровывается как SET PEN COLOR - установка цвета пера. "Спектрум" работает с восемью цветами:

- 0 - черный
- 1 - синий
- 2 - красный
- 3 - пурпурный
- 4 - зеленый
- 5 - голубой

6 - желтый

7 - белый

Вы можете задать любой из восьми цветов в качестве цвета пера. Например, команда SETPC 6 установит в качестве текущего цвета желтый. Все новые линии будут изображаться желтым цветом. Более того, все то, что нарисовано в пределах одного знакоместа рядом с нарисованной линией, тоже будет окрашено в желтый цвет.

Для работы с цветом, ЛОГО имеет еще одну удобную процедуру. Это процедура PENCOLOR. С ее помощью можно получить число, которое соответствует номеру включенного в данный момент цвета. Таким образом, манипулируя с цветами, Вы можете запомнить текущий цвет в какой-либо переменной и потом, когда надо, к нему вернуться:

```
MAKE "MYCOLOR PENCOLOR
```

```
.....
```

```
.....
```

```
.....
```

```
SETPC :MYCOLOR
```

Очень хорошей практикой при работе с цветом является при входе в любую процедуру, в которой изменяются цвета, запоминать текущие цвета в переменных, потом делать то, что требуется, а перед выходом из процедуры восстанавливать ранее запомненные цвета.

SETBG - эта команда расшифровывается как SET Background. Она устанавливает цвет фона экрана. Таким образом, она так же эквивалентна команде PAPER БЕЙСИКА, как команда SETPC эквивалентна команде INK.

Процедура BACKGROUND аналогична процедуре PENCOLOR и применяется в тех же целях. С ее помощью можно определить текущий включенный в данный момент цвет фона и запомнить его в какой-либо переменной.

```
MAKE "OLDCOLOR BACKGROUND
```

```
.....
```

```
.....
```

```
SETBG :OLDCOLOR
```

```
SETBORDER
```

Эта команда служит для переключения цвета рамки экрана и эквивалентна команде БЕЙСИКА BORDER.

В заключение этого раздела мы рассмотрим несколько примеров процедур. Вы можете развить эти примеры, дооснастив их манипуляциями с цветом.

### Спирали.

Без этого примера не обходится ни одна книга по ЛОГО. Не знаем, что уж в нем так понравилось авторам книг, наверное дело в том, что здесь используется рекурсия.

Начнем так же, как мы начинали при рисовании квадрата:

```
FORWARD :SIZE RIGHT 90
```

Но теперь, чтобы получить эффект спирали, нам надо увеличить размер стороны (SIZE) "квадрата" после одного витка и повторить все тоже самое несколько раз.

```
TO SQUARESPI :SIZE
```

```
FORWARD :SIZE
```

```
RIGHT 90
```

```
MAKE "SIZE :SIZE+5
```

```
SQUARESPI :SIZE
```

```
END
```

Рекурсия здесь состоит в том, что в описании процедуры SQUARESPI есть вызов самой этой же процедуры. При этом вызове компьютер обращается к ее описанию и в нем "натыкается" опять же на ее вызов и так далее. В данном случае эта спираль могла бы рисоваться бесконечно, поскольку никакого разумного выхода из рекуррентной последовательности нет. Вам же придется нажать BREAK для того, чтобы прервать работу процедуры.

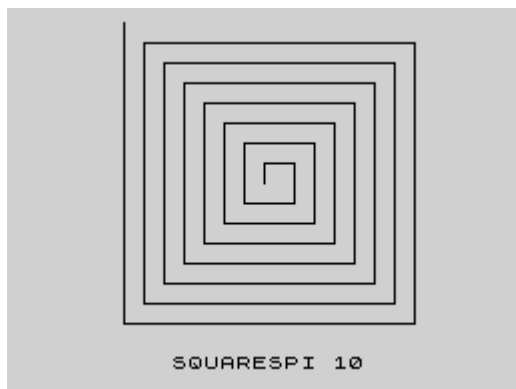


Рис. 1

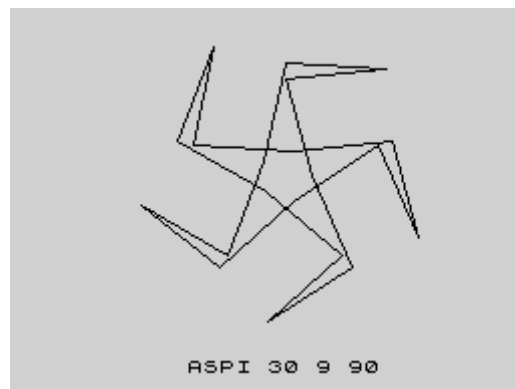


Рис. 2

Спирали не обязательно должны быть прямоугольными. Вы можете изменить приведенную выше процедуру, задав поворот после каждого шага не на 90 градусов, а на столько, на сколько Вам нужно. Может быть, Вам нужно управлять и величиной приращения, на которую увеличивается размер спирали после каждого витка. Это могло бы быть полезным в тех случаях, когда "черепашка" слишком быстро выходит за пределы экрана. Так в итоге мы получим процедуру для рисования спиралей с тремя параметрами:

**SIZE** - размер начальной стороны;

**ANGLE** - угол поворота;

**INC** - приращение стороны на каждом новом витке.

```
TO SPIRAL :SIZE: ANGLE: INC
  FORWARD :SIZE
  RIGHT :ANGLE
  MAKE "SIZE :SIZE + :INC
  SPIRAL :SIZE :ANGLE :INC
END
```

Проверьте эту процедуру на конкретных примерах. Наиболее интересные эффекты получаются, когда угол **ANGLE** почти (но не точно) равен углу правильного многоугольника. Испытайте, например, следующие примеры:

```
SPIRAL 10, 121, 5
SPIRAL 10, 119, 5
SPIRAL 10, 135, 5
```

Вы можете менять не только линейный размер витков **:SIZE**, но и угол поворота спирали **:ANGLE**.

```
TO ASPI :SIZE: ANGLE: INC
  FORWARD :SIZE
  RIGHT :ANGLE
  MAKE "ANGLE :ANGLE + :INC
  SPIRAL :SIZE :ANGLE :INC
END
```

Посмотрите, что получится для следующих вариантов:

```
ASPI 10 0 5
ASPI 10 40 30
ASPI 10 2 20
```

Один из примеров приведен на рис. 2.

Не менее интересным будет изображение части спирали, а затем повторение ее вновь из другой точки, в которой в данный момент времени находится "черепашка". В этом случае вместо изменения значения **SIZE** (которое должно быть постоянным) мы делаем шаг **FORWARD** на переменную величину, которая рассчитывается как произведение **SIZE** на некоторое число **COUNT**, изменяющееся от единицы до максимального значения **MAX**.

```
TO RESPI :SIZE :ANGLE :MAX
  MAKE "COUNT 1
  REPEAT :MAX [FORWARD :SIZE* :COUNT
                RIGHT :ANGLE
                MAKE "COUNT :COUNT + 1]
  RESPI :SIZE :ANGLE :MAX
END
```

Один из возможных примеров приведен на рис. 3:

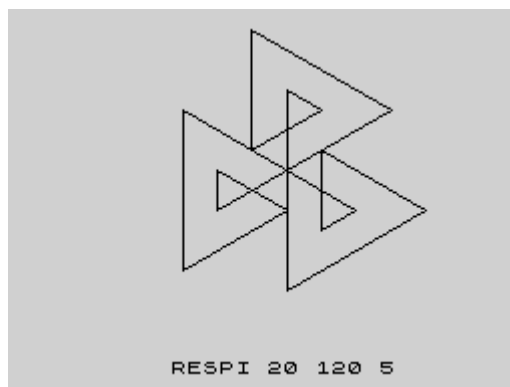


Рис. 3

#### ПРИМЕЧАНИЕ "ИНФОРКОМА".

Подобная технология имеет очень широкие возможности применения в создании игровых программ. Аналогичными приемами, основанными на рекурсии, в последнее время начали пользоваться для созданий графических изображений квазирегулярной структуры. К ним в первую очередь относятся естественные природные объекты: лес, трава, облака, снежинки, ледники, контур береговой линии материка или острова, дым от горящего танка и многое-многое другое. Смысл состоит в том, чтобы не хранить в компьютере картинку с изображением, например, облака, а всякий раз, когда надо его нарисовать, вызывать для этого процедуру-генератор облака (дыма, леса и т.п.).

Конечно, такие процедуры пишут не на ЛОГО, а на АССЕМБЛЕРЕ, но для "обкатки" алгоритма, прежде чем переносить его на "АССЕМБЛЕР", ЛОГО очень удобен, ведь надо подобрать и исследовать все параметры.

Технология создания подобных объектов квазирегулярной структуры хоть и не сложна, но почему-то малоисследована - то ли у многих до нее не дошли руки, то ли мало кто об этом знает, то ли информацию об этой технологии закрывают как ценное "ноу-хау"

Вы имеете уникальную возможность провести подобные исследования, используя ЛОГО в качестве удобного подручного инструмента. В исследованиях стоит стремиться к тому, чтобы Ваш объект (например дым) выглядел как можно более натурально. Он может быть динамичным, т.е. постоянно меняться и переливаться на экране. Возможно задействование команд управления цветом. Стоит также стремиться к тому, чтобы алгоритм, положенный в основу генератора, был легко переносим на АССЕМБЛЕР.

Мы с радостью напечатаем результаты наиболее интересных исследований. Так что дерзайте, Вас могут ждать потрясающие открытия.

### Объекты ЛОГО.

Если Вы еще не запутались со всеми двоеточиями и кавычками, с которыми пришлось иметь дело в предыдущих разделах, то это очень хорошо. Дело не в том, что трудно разобраться, где что нужно ставить, просто это такой вопрос, который лучше не объяснять, а демонстрировать на примере. Если же сейчас что-то покажется не очень понятным, не пугайтесь, у Вас всегда будет возможность вернуться к этому разделу, когда Вы будете сами писать свои процедуры.

Итак, ЛОГО имеет дело с четырьмя различными видами объектов:

- Слова
- Числа
- Процедуры
- Списки

О списках мы поговорим в следующей главе.

Что касается чисел, так официально их тоже квалифицируют как "слова", но поскольку они подчиняются несколько иным правилам, мы их выделили в отдельный объект. Числа состоят из десятичных цифр от 0 до 9. Возможно использование знака "минус" в начале



числа, а также использование десятичной точки. Числа, превышающие 10 миллионов в ЛОГО печатаются в экспоненциальной форме. Так, например, 2.3E+9 означает:

```
2 300 000 000
```

Запись E+9 обозначает, что десятичную точку следует перенести на девять позиций вправо от того места, в котором она показана в числе 2.3.

Точно так же для очень малых чисел величина экспоненты может быть отрицательной. Запись 1.4E-6 обозначает перенос десятичной точки влево на шесть позиций. Получим:

```
0.0000014
```

При такой форме записи можно работать почти с любыми числами. Если Ваше число окажется больше, чем 1E+38 (а это крайне маловероятно), то тогда будет выдано сообщение об ошибке

```
Number too big Число слишком велико.
```

Если же Ваше число окажется меньше, чем 1E-38, то оно воспринимается компьютером как ноль.

Обычно ЛОГО считает, что всякое слово, которое встречается в программе, является командой, которую следует исполнить, или именем процедуры, которую следует разыскать в памяти, прежде чем исполнять или выдавать сообщение об ошибке, если эта процедура не разыскана. Поэтому, если Вы хотите, чтобы Ваше слово воспринималось только как слово, а не как команда и не как имя процедуры, Вам необходимо перед ним поставить кавычки, например:

```
"FRED
```

Будьте внимательны и следите за тем, чтобы между кавычками и словом не образовалось случайного пробела. Дело в том, что ЛОГО признает существование "пустого слова", в котором нет ни одного символа. В программе такое "пустое слово" изображается кавычками, за которыми следует пробел. Этот странный объект на самом деле не столь бесполезен, как многим может показаться. Когда мы манипулируем со словами, обрабатывая по очереди букву за буквой, "пустое слово" является отличным индикатором того места, где следует остановиться.

Слово может быть именем какого-либо другого объекта. Мы с Вами уже рассматривали примеры, когда слово являлось именем процедуры или именем числовой переменной. Как Вы помните, последнее достигалось с помощью команды MAKE.

```
MAKE "NDS 15*40/100
```

По этой команде будет рассчитано значение NDS, равное 6. Мы можем сказать так, что "6" - это содержание того, что именуется словом NDS. Для того, чтобы по имени найти его содержание служит команда THING. Например:

```
PRINT THING "NDS
```

В результате получите число 6.

Поскольку поиск содержания переменной по ее имени очень и очень часто встречающаяся задача, решено было упростить форму записи и вместо THING ставить двоеточие:

```
PRINT :NDS
```

Впрочем, команда THING и двоеточие - это не вполне эквивалентные вещи и не всегда могут быть использованы одно вместо другого. Об этом мы еще поговорим чуть ниже.

Команда MAKE всегда имеет следующую конструкцию:

```
MAKE <имя> <содержание>
```

Поскольку первым параметром всегда идет имя, оно обычно начинается с кавычек. Второй параметр может быть словом, числом или списком.

```
MAKE "PRICE 27
```

```
PRINT PRICE
```

```
27
```

Число 27 является значением переменной с именем PRICE.

Другой пример:

```
MAKE "MARY "CONTRARY
```

```
PRINT "MARY
```

```
MARY
```

Поскольку в команде PRINT стояли кавычки, то мы точно указали, что мы хотим напечатать именно слово MARY, что и было сделано. А вот если сделать так:

```
PRINT :MARY
```

- то это указание на то, чтобы распечатать содержимое MARY и в итоге получим:

```
CONTRARY
```

А вот еще более сложный пример:

```
MAKE "JEAN :MARY
```

```
PRINT :JEAN
```

```
CONTRARY
```

Команда MAKE присвоила имени JEAN содержимое имени MARY. Оно, как мы помним равно CONTRARY, что мы и получили по команде PRINT, дав указание распечатать содержимое JEAN.

На первый раз может показаться нелепым использовать одно слово вместо другого, но при создании программ это может быть полезным. Так, например, Вы хотите, чтобы Ваша программа сначала спросила как зовут того, кто с ней работает, а потом в диалогах с ним всегда пользовалась этим именем. Тогда Вы, не зная заранее имени пользователя, всюду в диалогах используете вместо него например имя ANSWER. А когда узнаете его имя, присвойте переменной ANSWER необходимое содержание.

```
TO GREET
```

```
PRINT [WHAT IS YOUR NAME]
```

```
MAKE "ANSWER READLIST
```

```
PRINT "HELLO
```

```
PRINT :ANSWER
```

```
END
```

Когда Вы работаете в БЕЙСИКе, то для ввода имени пользователя Вы использовали бы команду INPUT. Здесь же мы воспользовались для этой цели вызовом процедуры READLIST. О том, что это за процедура мы поговорим, когда придет время. Пока же Вам достаточно знать, что она возьмет слово, набранное пользователем с клавиатуры (или список слов) и присвоит его в качестве параметра в команде MAKE.

Теперь Вы можете видеть, что если, скажем JOHN является именем человека, работающего на компьютере, то ANSWER является именем переменной, содержанием которой является слово JOHN.

Рассмотрим еще более сложный пример:

```
MAKE "GIRL "MARY
```

```
PRINT "GIRL
```

```
GIRL
```

Пока все идет тривиально просто.

```
PRINT :GIRL
```

```
MARY
```

Это мы тоже уже проходили.

```
PRINT THING :GIRL
```

```
CONTRARY
```

А вот это уже что-то новенькое. Фактически мы дали команду напечатать содержание имени, которое является содержанием переменной по имени GIRL.

В принципе конструкция

```
THING :GIRL
```

эквивалентна конструкции

```
:: GIRL ,
```

которая не разрешена, или конструкции

```
THING THING GIRL
```

Последнюю конструкцию Вы можете проверить сами.

Фактически мы имеем два уровня вложения имени:

GIRL ---> MARY ---> CONTRARY

Вот перед Вами как раз тот случай, когда команда THING и знак "двоеточие" не вполне взаимозаменяемы, о чем мы и упомянули выше.

Зачем нам может быть нужен более, чем один уровень вложения имени? - Для большей гибкости языка, для создания более универсальных процедур.

Рассмотрим пример. У Вас на складе лежат разные товары: кассеты, дискеты, джинсы и пр. Каждый из этих товаров обладает разной ценой (PRICE). Предположим, что Вы хотите написать процедуру, которая быстро сообщит вам данные по запрашиваемому товару.

Чтобы не делать по каждому виду товара свою процедуру, Вы сделаете подстановку имени:

```
TO NAMEPRINT :OBJECT
PRINT :OBJECT
PRINT THING :OBJECT
END
```

Предположим, что джинсы стоят 15 долларов, кассеты - 2, а дискеты - 1.

```
MAKE "JEANS 15
MAKE "CASSETTES 2
MAKE "DISKETTES 1
```

```
.....
.....
```

Когда же эта информация нам будет нужна, мы всегда ее получим с помощью процедуры NAMEPRINT, задавая при вызове интересующий нас товар (OBJECT).

```
.....
NAMEPRINT "JEANS
.....
```

По такой команде получим сообщение:

```
JEANS
15
```

Если воспользоваться еще и оператором SENTENCE (с ним мы будем разбираться позже), то можно программировать целые отчеты от компьютера, например:

```
TO NAMEPRINT :OBJECT
PRINT (SENTENCE "THE_PRICE_OF
:OBJECT "IS THING :OBJECT)
END
```

Если теперь дать команду NAMEPRINT "JEANS, то получите сообщение:

```
THE PRICE OF JEANS IS 15
(Цена джинсов - 15)
```

Еще более интересно использовать такую технику в команде MAKE, что дает возможность косвенного присвоения и переприсвоения переменных. Пример:

```
MAKE "QUANTITY "PRICE
MAKE :QUANTITY 27
PRINT :PRICE
27
```

Самое интересное в том, что мы нигде не присваивали числового значения переменной PRICE и, тем не менее, когда оно нам понадобилось, смогли его получить по команде PRINT. Все дело во второй строчке, где содержимому переменной QUANTITY присваивается значение 27. Поскольку содержимое имени QUANTITY равно имени PRICE, то и получилось так, что значение 27 присвоилось переменной, имя которой PRICE.

В заключение, говоря о правилах использования кавычек, мы должны сказать еще пару слов.

Как Вы знаете, в ЛОГО все слова, которые не являются числами и не начинаются с кавычек или двоеточия, интерпретируются, как имена процедур. Таким образом, встретив например слово JUMP, ЛОГО либо перейдет к поиску и исполнению процедуры JUMP, либо (если она не задана), выдаст сообщение об ошибке. Но ведь бывают случаи, когда Вам надо сделать с процедурой что-либо иное, например вызвать ее для редактирования. В этом случае перед именем процедуры должны стоять кавычки:

```
ED "JUMP
```

То же самое относится и к системным операторам PO, ER и SAVE.

```
MAKE "P LIST FIRST THING :A SENTENCE "PR WORD CHAR 34 :A
```

Рис. 4а

```
MAKE "P LIST FIRST THING :A SENTENCE "PR WORD CHAR 34 :
  2      2      1      1      2      2      1
```

Рис. 4б

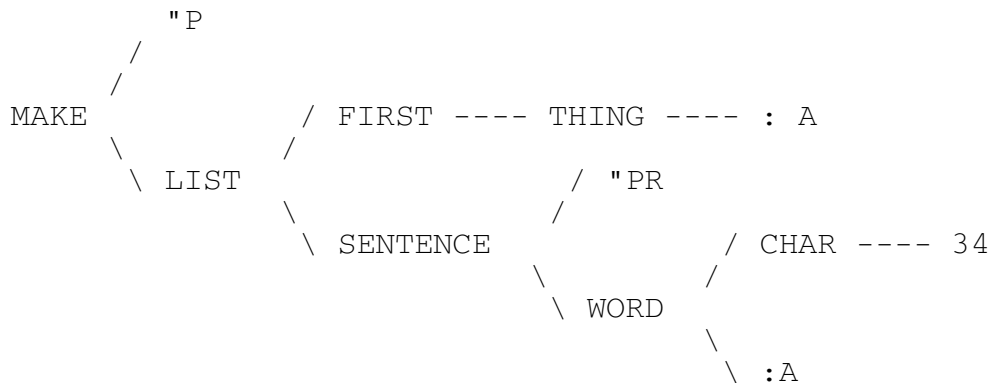


Рис. 4в.

### Виды процедур.

В ЛОГО имеет смысл различать два вида процедур.

Первый вид - команды, которые дают компьютеру указание выполнить какие-то действия, например:

```
FORWARD 50
PRINT "HELLO
PENUP
SAVE "MYPROC
```

и т.п.

Второй вид - операторы. Их задача - рассчитать результат и выдать его для исполнения командой. Например:

```
SUM
DIV
```

и т.п.

В ЛОГО все команды принимают параметры (если они нужны для работы) справа от себя, а всякий оператор выдает результат своей работы влево от себя. Именно поэтому, когда мы рассматривали арифметические операции, мы всегда слева вставляли команду PRINT.

Единственным исключением этого общего принципа "Вход справа - выход налево" являются знаки арифметических операций +, -, /, >, <, =. Эти знаки ставятся между своими операндами - это исключение является как бы данью традиции.

Те процедуры, которые встроены в ЛОГО без Вашего участия и являются системными, называют ПРИМИТИВАМИ.

Иногда строки в ЛОГО могут выглядеть очень сложными. Посмотрите, например, на строку, приведенную на рис. 4а.

Тем не менее, существует простой способ разобраться с такой конструкцией. Прежде всего, найдите в строке имена процедур. Это не числа и они не начинаются с кавычек или двоеточия. Разберитесь сколько параметров каждая из них ожидает после себя и поставьте это число под именем процедуры (рис. 4.б.).

Большинство слов в этой строке Вам пока незнакомо, но это даже и хорошо для того, чтобы научиться анализировать непонятные строки. Пока скажем только, что все процедуры

являются встроенными примитивами.

Теперь на листе бумаги напишите первое слово строки. Это обязательно команда. К нему проведите столько стрелок, сколько параметров требует эта команда, см. рис 4в. В конце первой стрелки запишите следующее слово. Если это не имя процедуры, то следующее слово запишите в конце следующей стрелки. Если же это процедура, то к ней тоже проведите столько стрелок, сколько параметров она требует.

Продолжайте таким образом, пока не дойдете до конца ЛОГО-строки. Результат для данной строки показан на рис. 4в. Здесь ясно видно как и от кого каждая процедура ожидает входные данные и куда направляет результат своей работы.

Если же этот метод не сработал и либо в строке есть слова, которые некуда подключить, либо есть стрелки, которые ничем не кончаются, то значит в написании строки имеется ошибка.

## ГЛАВА 4. СЛОВА И СПИСКИ

Что такое список, знают конечно все и объяснять этого не надо. Даже совсем не рассеянные люди, готовясь к поездке, составляют список того, что надо захватить в дорогу.

Список как объект языка ЛОГО ничем не отличается от тех списков, к которым Вы привыкли в обыденной жизни. Правда, ЛОГО любит, чтобы списки для него заключали в квадратные скобки, а отдельные элементы списков отделяли друг от друга с помощью пробелов. Порядок элементов в списке играет важную роль. Так, два нижеследующих списка являются различными:

```
[JANUARY FEBRUARY MARCH]
```

```
[JANUARY MARCH FEBRUARY]
```

Элементами списков являются либо слова, либо тоже списки. Со списками, состоящими из списков мы поговорим в следующей главе, а пока остановимся на списках, состоящих из слов.

Обычно нет необходимости слова в списках начинать с кавычек. ЛОГО полагает, что два нижеследующих списка идентичны:

```
[JANUARY FEBRUARY MARCH]
```

```
["JANUARY "FEBRUARY "MARCH]
```

Для чисел мы вводили имена переменных, точно так же можно дать имя списку. Это тоже делается командой MAKE.

```
MAKE "SHOPPING [BREAD BUTTER MILK STAMPS]
```

Списку предполагаемых покупок (хлеб, масло, молоко, почтовые марки) мы присвоили имя SHOPPING.

Преимущество работы с именованным списком состоит в том, что мы можем ссылаться на него по имени. Например, команда

```
PRINT :SHOPPING
```

```
BREAD BUTTER MILK STAMPS
```

Обратите внимание на то, что при печати списка ЛОГО опускает квадратные скобки.

Мы можем присвоить списку другое имя или, наоборот, списку с заданным именем присвоить иное содержание, например:

```
MAKE "BOUGHT :SHOPPING
```

```
PRINT :BOUGHT
```

```
BREAD BUTTER MILK STAMPS
```

Так у нас появился новый список BOUGHT.

### Выбор элементов списка.

В списках было бы очень мало пользы, если бы с ними можно было работать только как с одной цельной вещью (хранить, переименовывать и распечатывать). Нам нужны приемы, с помощью которых мы могли бы иметь доступ к отдельным элементам списка. ЛОГО предоставляет несколько таких приемов. Самый простой и наиболее широко применяющийся - оператор FIRST. Он выдает первый элемент списка.

Например:

```
PRINT FIRST [JANUARY FEBRUARY MARCH]
```

```
JANUARY
```

Оператор FIRST может работать и с именованными списками тоже:

```
PRINT FIRST :SHOPPING
```

```
BREAD
```

Разобравшись с первым элементом списка, мы хотели бы разобраться и с остальными. Для этого наиболее стандартным является оператор BUTFIRST. Он создает второй список, состоящий из всех элементов исходного списка, кроме первого. Т.е. как бы отбрасывается первый элемент, а оставшиеся элементы становятся новым списком.

```
PRINT BUTFIRST [JANUARY FEBRUARY MARCH]
```

```
FEBRUARY MARCH
```

совершенно аналогично для именованных списков:

```
PRINT BUTFIRST :SHOPPING
```

```
BUTTER MILK STAMPS
```

Кроме естественного различия между FIRST и BUTFIRST есть еще одно очень важное различие:

FIRST - дает слово;

BUTFIRST - дает список.

Специально для удобства пользователей со списками можно работать не только с начала, но и с конца. Оператор LAST дает последний элемент списка:

```
PRINT LAST :SHOPPING
```

```
STAMPS
```

Есть и оператор BUTLAST, который, как Вы уже очевидно догадались, создает новый список, равный исходному без последнего элемента.

```
PRINT BUTLAST [MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY]
```

```
MONDAY TUESDAY WEDNESDAY THURSDAY
```

Можно выбирать не только первый или последний элемент из списка. Для того, чтобы получить любой произвольный элемент, служит оператор ITEM, который требует после себя указания двух параметров - номера нужного элемента списка и самого списка.

```
PRINT ITEM 2 [APRIL MAY JUNE]
```

```
MAY
```

Список может быть задан своим именем:

```
PRINT ITEM 3 :SHOPPING
```

```
MILK
```

Часто бывает важным знать, сколько же элементов содержится в Вашем списке. Для этого служит оператор COUNT.

```
PRINT COUNT [JANUARY FEBRUARY MARCH]
```

```
3
```

```
PRINT COUNT :SHOPPING
```

```
4
```

Многие стандартные процедуры ЛОГО воспринимают списки в качестве входных параметров. С некоторыми мы уже сталкивались. Такова, например, команда REPEAT, после которой должны быть два параметра. Первый параметр - число повторов. Второй параметр - список повторяющихся команд. Это один из тех случаев, когда кавычки могут быть внутри списка, причем опустить их нельзя:

```
REPEAT 4[MAKE "NUMBER :NUMBER+1]
```

Хоть команда REPEAT и принимает список как список, тем не менее команда MAKE, стоящая в списке, требует, чтобы ее первым параметром было имя. Если Вы захотите испытать этот пример, то предварительно не забудьте задать первоначальное значение переменной NUMBER.

Некоторые из встроенных примитивов могут работать как со словами, так и со списками. Примером может служить команда PRINT, которая распечатает и слово и список. Фактически именно для таких случаев в ЛОГО и ввели правило, по которому списки должны быть заключены в квадратные скобки.

Вы можете создавать и свои процедуры для работы со списками. Так, например, ниже приведена процедура для печати элементов списка в столбик по одному (обычно список распечатывается в строку).

Процедура работает в цикле REPEAT, переходя от одного элемента к другому и выхватывая его с помощью оператора ITEM.

```
TO LBL :ALIST
```

```

MAKE "COUNTER 1
REPEAT COUNT :ALIST
  [PRINT ITEM :COUNTER :ALIST
   MAKE "COUNTER :COUNTER+1]
END

```

Тогда команда:

```

LBL [LBL STANDS FOR LINE BY LINE]
даст результат
LBL
STANDS
FOR
LINE
BY
LINE

```

Попробуйте сами в порядке эксперимента написать процедуру, которая сможет печатать список в обратном порядке.

Используя уже изученные инструменты и приемы, мы можем применять списки для того, чтобы хранить и обрабатывать информацию. Например, Вам надо сохранить в базе данных свои ежемесячные расходы в течение года, а потом найти их среднюю величину и сумму. Тогда Вы составите список, в котором первым элементом будет расход за январь, вторым - за февраль и т.д. Нижеприведенная процедура пройдет по списку таким же образом, как это делала процедура LBL. Только она не будет распечатывать найденные элементы, а будет накапливать их текущую сумму в переменной TOTAL, которая конечно в начале процедуры должна быть занулена. Средняя величина расходов определится делением суммы на двенадцать.

```

TO AVERAGE :ALIST
MAKE "TOTAL 0
MAKE "COUNTER 1
REPEAT 12 [MAKE "TOTAL, :TOTAL + ITEM: COUNTER :ALIST
  MAKE "COUNTER :COUNTER + 1]
MAKE "AVERAGE :TOTAL/12
PRINT "TOTAL
PRINT :TOTAL
PRINT "AVERAGE
PRINT :AVERAGE
END

```

Испытайте эту процедуру с любым удобным Вам списком из двенадцати чисел.

### Ввод слов и списков.

Обычно ввод списков с клавиатуры выполняют с помощью оператора READLIST. Результат его работы должен быть тут же передан какой-либо переменной или процедуре. Обычно это делают с помощью команды MAKE. Например:

```
MAKE "ANSWER READLIST
```

По этой команде компьютер переходит в состояние ожидания до тех пор, пока Вы не наберете то, что Вам нужно на клавиатуре. Обозначается это состояние вопросительным знаком на экране. Список ввода состоит из тех слов, разделенных пробелами, которые Вы наберете до нажатия клавиши ENTER, после чего ввод будет закончен.

Попробуйте и посмотрите, что Вы набрали:

```
MAKE "ANSWER READLIST
PRINT :ANSWER
```

LOGO не имеет соответствующего оператора для ввода только одного слова, но в нем и нет необходимости. Обычно, когда в этом возникает необходимость, пользуются конструкцией:

```
FIRST READLIST
```

которая вводит первое слово входного списка. Все остальные слова игнорируются.

Может быть, Вы хотите получить отдельно первое слово списка и всю остальную часть списка. В этом случае неприменима такая конструкция:

```
MAKE "START FIRST READLIST
MAKE "REST BUTFIRST READLIST
```

Просто встретив READLIST второй раз, компьютер снова войдет в состояние ожидания и примет окончание списка, НО ЭТО БУДЕТ УЖЕ НЕ ТОТ СПИСОК.

Если Вы хотите, чтобы у Вас все получилось нормально, то единственный путь - принять весь список и только потом заниматься его разделением.

```
MAKE "ANSWER READLIST
MAKE "START FIRST :ANSWER
MAKE "REST BUTFIRST :ANSWER
```

### **Пустой список.**

Список, в котором не содержится ни один элемент, является необычным, но очень полезным объектом. Обозначается он - [] и называется пустым списком. Он нужен, например, при создании списков в качестве исходного элемента. Он же используется для сигнализации о конце работы, когда какой-то заполненный список сканируется и элемент за элементом извлекаются из него и отправляются на обработку процедурами.

### **Создание и слияние списков.**

Для этих целей широко используется процедура SENTENCE. Мы уже встречались с ней в команде PRINT. Она принимает в качестве входных параметров два или более слова или списка и объединяет их в один список. Например:

```
MAKE "PROVERB SENTENCE "MANY "HANDS
PRINT :PROVERB
MANY HANDS
```

Мы слили два слова: MANY и HANDS в один список MANY HANDS.

Но SENTENCE может сливать и списки:

```
MAKE "PROVERB SENTENCE [MANY HANDS] [MAKE LIGHT WORK]
PRINT :PROVERB
MANY HANDS MAKE LIGHT WORK
```

Или:

```
MAKE "NEWSAYING SENTENCE :PROVERB [OR SPOIL BROTH]
PRINT :NEWSAYING
MANY HANDS MAKE LIGHT WORK OR SPOIL BROTH
```

Оператор SENTENCE может иметь более двух входных параметров. В этом случае и SENTENCE и параметры должны быть заключены в круглые скобки.

```
PRINT (SENTENCE [A ROLLING][STONE GATHERS][NO MOSS])
```

или:

```
MAKE "OLDSAYING (SENTENCE [THE SAYING] :PROVERB [IS TRUE]) PRINT :OLDSAYING
THE SAYING MANY HANDS MAKE LIGHT WORKS IS TRUE
```

Команда PRINT всегда требует при себе только одного входного параметра - слова или списка, поэтому Вы не можете напечатать:

```
PRINT "MANY "HANDS
```

Вам на помощь приходит список:

```
PRINT [MANY HANDS]
```

Но если текст еще более сложен и требует слияния списков и слов в одно целое, то мы пользуемся оператором SENTENCE:

```
PRINT SENTENCE [THE SHOPPING LIST IS] :SHOPPING
```

В результате Вы получите:

```
THE SHOPPING LIST IS BREAD BUTTER MILK STAKES
```

До сих пор мы использовали SENTENCE для слияния списков. Теперь посмотрим, как можно создавать новые списки. Пусть, например, перед нами стоит задача создать список ежемесячные расходов, которым мы уже пользовались в одном из вышеприведенных примеров. Процедура должна печатать по порядку названия месяцев, принимать с клавиатуры величину расхода в данном месяце и вводить ее в список расходов.

Мы начнем с того, что создадим список месяцев - YEAR, а список расходов EXPEND сделаем пока пустым списком. Затем повторим двенадцать раз процедуру ввода данных с клавиатуры - INMONTH.

```
TO BUILDLIST
MAKE "YEAR [JANUARY FEBRUARY
MARCH APRIL MAY
```



```

JUNE JULY AUGUST
SEPTEMBER OCTOBER
NOVEMBER DECEMBER]
MAKE "EXPEND []
MAKE "COUNTER 1
REPEAT 12[INMONTH]
END

```

Процедура INMONTH должна печатать название очередного месяца и принимать с клавиатуры данные по расходам за этот месяц. Для этого используется техника FIRST READLIST, рассмотренная выше.

```

TO INMONTH
MAKE "MONTH ITEM :COUNTER :YEAR
PRINT :MONTH
MAKE "EXPEND SENTENCE :EXPEND
FIRST READLIST
MAKE COUNTER :COUNTER + 1
END

```

Есть еще один изощренный метод исполнения того, что делает оператор SENTENCE. Обычно изощрения в программировании вовсе не являются необходимыми улучшениями, но в данном случае у нас появляется одно преимущество. В этой технологии используется идея, описанная в конце предыдущей главы. Давайте рассмотрим слова в списке YEAR как имена переменных, содержанием которых является ежемесячный расход. Тогда, например, "JANUARY - это имя месячного расхода за январь. Тогда процедура INMONTH может выглядеть так:

```

TO INMONTH
MAKE "MONTH ITEM :COUNTER :YEAR
PRINT :MONTH
MAKE :MONTH FIRST READLIST
MAKE "COUNTER :COUNTER+1
END

```

Вторая команда MAKE использует содержание MONTH в качестве своего первого параметра, в результате JANUARY станет именем тех данных, которые будут считаны с клавиатуры по FIRST READLIST.

Такой подход дает нам то преимущество, что каждый элемент данных может быть легко индивидуально изменен. При написании программ всегда следует думать о том, что должен сделать пользователь, если случайно допустит ошибку. Если мы будем работать таким приемом, то нам не нужен список расходов EXPEND.

Как упражнение напишите процедуру CHANGE, которая примет с клавиатуры название месяца и расход в этом месяце и изменит эти значения в списке.

Если Вам захочется сохранить результаты этих экспериментов на кассете в виде сборника процедур для ведения личной бухгалтерии, ЛОГО обеспечивает для этого только одну возможность. Вы можете воспользоваться командой SAVEALL, после которой надо указать имя файла:

```
SAVEALL "MONEY
```

По этой команде на ленту будет выгружено все, что находится в рабочей области, т.е. и все процедуры и все переменные. В этом тоже есть удобство, поскольку когда Вам опять понадобится файл MONEY, то загрузив его Вы не только загрузите все данные, но и все процедуры, необходимые для работы с ним.

### Работа со словами.

Многие процедуры, предназначенные для обслуживания списков и слов внутри списков, могут на самом деле применяться только для работы со словами и даже с символами, из которых эти слова состоят. Так, например, команда FIRST, если ее применить не к списку, а к отдельному слову, выдаст первый символ этого слова.

```
PRINT FIRST "ABC
A
```

Точно так же:

```
PRINT COUNT "Friday
```

даст в результате:

6

Из рассмотренных в этой главе процедур только две неприменимы для работы с отдельными словами. Это процедуры ITEM и SENTENCE.

Аналог процедуре ITEM мы сделаем несколько позже, воспользовавшись для этого FIRST, LAST, BUTFIRST, BUTLAST и рекурсией.

А вот в качестве аналога процедуры SENTENCE для создания слов есть специальная процедура WORD. Она и работает как SENTENCE, принимая два или более слов, но сливает их не в список, а в одно слово, т.е. не оставляет между составляющими словами пробелов.

```
PRINT WORD "BLACK "POOL  
BLACKPOOL
```

Так же, как и SENTENCE, процедура WORD требует наличия круглых кавычек, если входных данных более, чем два.

```
PRINT (WORD "BIRM "ING "HAM)  
BIRMINGHAM
```

В качестве примера мы рассмотрели несложную процедуру, которая принимает с клавиатуры от пользователя слово и печатает его задом наперед. Для приема слова мы опять используем технику FIRST READLIST. Затем, начиная с конца, это слово сканируется и символ за символом собираются в новое слово с помощью оператора WORD. Исходно новое слово задается пустым словом.

```
TO REVERSE  
  PRINT (GIVE ME A WORD)  
  MAKE "INWORD FIRST READLIST  
  MAKE "NEWWORD "  
  REPEAT COUNT :INWORD  
    [MAKE "NEWWORD  
      WORD :NEWWORD LAST :INWORD  
      MAKE - INWORD BUTLAST : INWORD]  
  PRINT :NEWWORD  
END
```

Те, кто программируют на БЕЙСИКе, должны быть знакомы с процедурами, отрезающими левую или правую часть слова. В ЛОГО нет соответствующих встроенных примитивов, вместо этого мы напишем свою процедуру, которая принимает в качестве входных параметров слово и число символов, которые должны быть "отрезаны" от слова, считая слева, то есть от начала слова. Эта процедура создаст два новых слова, назовем их LEFT и RIGHT.

Работа этой процедуры похожа на работу ранее рассмотренной процедуры REVERSE. В качестве исходного значения правой (RIGHT) части задается все исходное слово, а в качестве исходного значения левой (LEFT) его части задается пустое слово. Затем необходимое количество символов переносится из RIGHT в LEFT.

```
TO SPLIT :AWORD :NUM  
  MAKE "RIGHT :AWORD  
  MAKE "LEFT "  
  REPEAT :NUM  
    [MAKE "LEFT WORD :LEFT FIRST  
      :RIGHT  
      MAKE "RIGHT BUTFIRST :RIGHT]  
END
```

для эксперимента попробуйте:

```
SPLIT "DEFINED 4  
PRINT :RIGHT  
PRINT :LEFT
```

В результате должно получиться

```
NED DEFINI
```

А сейчас мы прощаемся с Вами до следующего выпуска, в котором рассмотрим работу с циклами и еще немного углубим свои взгляды на "черепашью графику".

(Продолжение следует)

## Маленькие хитрости

Короткое, но очень интересное письмо прислал наш читатель Радзевич А. А. из г. Нальчик. Оно очень заинтриговало нас. Вот его письмо практически дословно:

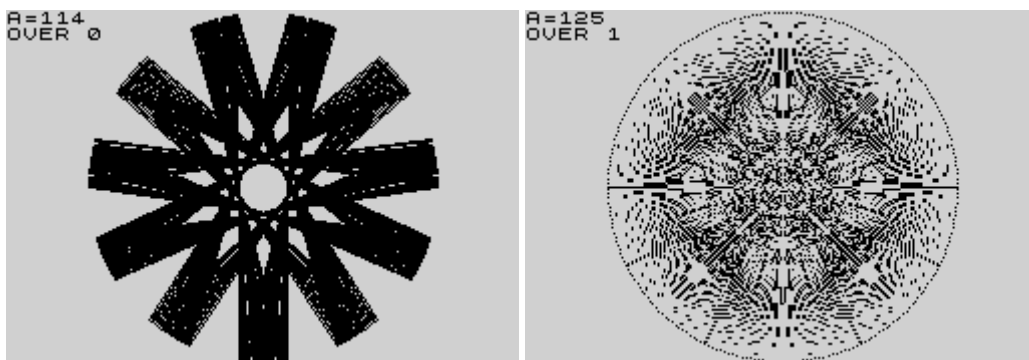
"Мучая свой SINCLAIR я раскопал нечто такое, что Вас может заинтересовать. Запустите следующую программу и Вы все увидите сами. "

```
10 INPUT a
20 CLS
30 PLOT 41,87
40 DRAW 174,0,(2*a+1)*PI
50 GO TO 10
```

Наиболее любопытные эффекты возникают, если ввести числа 91, 100, 125, а также и любые другие. Можно еще включить режим OVER 1. Попробуйте, Вы не зря потратите время!"

Набрав эту программу, мы убедились, что это действительно "нечто такое". Не скроем, мы с интересом "играли" с этой программой, задавая всевозможные числа и наблюдая за тем, что происходит при этом на экране компьютера.

Программа предельно проста, но эффекты, которые она создает, совсем неочевидны. Через некоторое время, "наигравшись", мы задумались над тем, как же все-таки она работает. Причудливый орнамент образуется на экране, и весь он создается всего лишь одним оператором DRAW. Что лежит в основе происходящего? Работа встроенного калькулятора? Округления чисел? Любители головоломок! Если Вы заинтересовались происходящим, попытайтесь объяснить, как же все-таки работает эта программа?



# ПРИМЕНЕНИЕ АССЕМБЛЕРА ДЛЯ СОЗДАНИЯ БЫСТРОРАБОТАЮЩИХ ПРОГРАММ

Перевод с английского Пашорина В. И.

Продолжение. Начало см. с. 9-12, 48-61.

## 4. СЛУЧАЙНЫЕ ЧИСЛА

Случайные числа являются неотъемлемой частью большинства игровых программ. Получить случайные числа с помощью программы в машинных кодах довольно легко и, в зависимости от диапазона выборки, определяющего границы интервала, в котором необходимо получить случайные числа, различают два способа.

Первый способ, вполне удовлетворительный для большинства игровых программ, основан на использовании системных переменных SEED и FRAMES. Применяться этот способ может при следующих диапазонах выборки:

0...1,  
0...3,  
0...7,  
0...15,  
0...31,  
0...63,  
0...127,  
6...255.

Для этого в регистровую пару HL записывается значение системной переменной SEED (5C76HEX = 23670 DEC), а в регистровую пару DE - значение младшего байта системной переменной FRAMES (5C78HEX = 23672 DEC). Затем эти два числа суммируются и результат заносится в системную переменную SEED (это нужно для выборки очередного случайного числа). В регистр A поочередно записываются значения H и L и путем логической операции AND маскируются незначимые биты для задания диапазона выборки случайного числа.

В программе 4.1, к примеру, для задания диапазона выборки 0...31 т.е. для получения таких же случайных чисел, как и по BASIC-команде INT(RND\*32), в регистре A маскируются биты с пятого по седьмой.

Как уже отмечалось, такой способ может быть использован для получения случайных чисел в строго определенных диапазонах. Однако, представленную программу можно усовершенствовать и расширить диапазон фиксированных интервалов для случайных чисел. Например, для реализации INT(RND\*23) необходимо добавить:

```
LD    A, H
AND   15
LD    H, A
LD    A, L
AND   7
ADD   A, H
```

Обратите внимание на то, что в программе 4.1 регистр B используется как счетчик шагов цикла (10 шагов) и значение этого же регистра используется в PRINT-процедуре: PRINT AT B, 10 для вывода случайных чисел на экран.

| АДРЕС | МАШ. КОД    | АССЕМБЛЕР      | КОММЕНТАРИЙ                   |
|-------|-------------|----------------|-------------------------------|
|       |             | ORG 23760      |                               |
| 23760 | 06 0A       | LD B, 10       | ; Счетчик на 10 шагов.        |
| 23762 | C5          | PUSH BC        | ; Запомнили состояние счет-   |
|       |             |                | ; чика.                       |
| 23763 | 2A 76 5C    | LD HL, (23670) | ; Приняли в HL состояние      |
|       |             |                | ; системной переменной SEED.  |
| 23766 | ED 5B 78 5C | LD DE, (23672) | ; Приняли в DE состояние      |
|       |             |                | ; системной перемен. FRAMES.  |
| 23770 | 19          | ADD HL, DE     | ; Нашли их сумму              |
| 23771 | 22 76 5C    | LD (23670), HL | ; и установили результат в    |
|       |             |                | ; SEED.                       |
| 23774 | 7D          | LD A, L        |                               |
| 23775 | E6 1F       | AND 31         | ; Гашение ненужных битов.     |
| 23777 | F5          | PUSH AF        | ; Запомнили результат.        |
| 23776 | 3E 02       | LD A, 2        | ;                             |
| 23780 | CD 01 16    | CALL 5633      | ;  Открытие канала печати на  |
| 23783 | 3E 16       | LD A, 22       | ;  экран и выдача управля-    |
| 23785 | D7          | RST 16         | ;  ющих кодов для установки   |
| 23786 | 78          | LD A, B        | ;  позиции печати             |
| 23787 | D7          | RST 16         | ;                             |
| 23788 | 3E 0A       | LD A, 10       | ;   AT B, 10                  |
| 23790 | D7          | RST 16         | ;                             |
| 23791 | F1          | POP AF         | ; Восстановление числа со     |
|       |             |                | ; стека.                      |
| 23792 | CD 28 2D    | CALL 11560     | ; Процедура STACK-A (помеща-  |
|       |             |                | ; ет на стек калькулятора     |
|       |             |                | ; число, записанное в акку-   |
|       |             |                | ; муляторе процессора.        |
| 23795 | CD E3 2D    | CALL 11747     | ; Процедура PRINT_FP. Печата- |
|       |             |                | ; ет действительное число,    |
|       |             |                | ; находящееся на вершине      |
|       |             |                | ; стека калькулятора.         |
| 23798 | C1          | POP BC         | ; Восстановление счетчика.    |
| 23799 | 10 D9       | DJNZ L1        | ; Уменьшение счетчика и воз-  |
|       |             |                | ; врат на новый шаг.          |
| 23801 | C9          | RET            | ; Выход из процедуры.         |

Для получения случайных чисел из произвольного интервала применяется второй способ. Этот способ основан на использовании RND-процедуры из системного ПЗУ компьютера. К сожалению, эту процедуру нельзя вызвать простой командой CALL RND, т.к. она не завершается командой RET. Процедура эта занимает участок памяти с адреса 9725 по адрес 9765 (25FDHEX...2625HEX) и в своей работе использует встроенный калькулятор, а по завершении работы меняет значение переменной SEED. Для этого в начале работы это значение извлекается из SEED, преобразуется, а затем дважды записывается на вершину калькуляторного стека. Первое значение на вершине стека передается в системную переменную SEED для генерации очередного случайного числа, а оставшееся на вершине стека значение преобразуется для получения соответствующего ему числа в диапазоне 0...1 (но не = 1). Чтобы использовать эту процедуру, необходимо ее скопировать из ПЗУ и затем вызывать, как подпрограмму.

Подпрограмма 4.2 показывает применение этого способа для получения случайных чисел

```
INT (RND*12345)
```

и вывода 44 из них в два столбца на экране путем использования CHR\$6.

RND-процедура использует несколько новых для нас кодов калькулятора, которые могут быть полезны и для других процедур:

Код 161 (A1H) - stack\_one - помещает на вершину стека калькулятора единицу.

Код 52 (34H) - stack\_data - интерпретирует следующие за ним пять байтов как действительное число с плавающей точкой, записанное в интегральной форме и помещает

его на вершину стека калькулятора.

Код 50 (32H) -  $n\_mod\_m$  - вместо делимого ( $n$ ) и делителя ( $m$ ) эта команда помещает на стек остаток от деления и целое частное.

Код 02 - delete - команда служит для удаления числа с вершины стека калькулятора.

Последние два кода редко используются в игровых программах. Вызов процедуры FP\_TO\_BC, находящейся по адресу 11682 DEC (2DA2HEX) - CALL 11682, позволяет извлечь число с вершины стека, округлить его до ближайшего целого числа и затем записать в регистровую пару BC. В отличие от этой процедуры, процедура FF\_TO\_A, находящаяся по адресу 11733DEC (2BD5HEX), передает число с вершины стека в регистр A.

В данной программе скопированная RND-процедура заканчивается по адресу 23810, а начиная с адреса 23811 и по адрес 23820 находится процедура, осуществляющая дальнейшее преобразование случайного числа, записанного на стеке. Для этого на стек записывается число 12345, определяющее диапазон выборки, а затем производится перемножение случайного числа из диапазона 0...1, находящегося на стеке, с этим числом. Затем кодом калькулятора 02 (delete) удаляем со стека дробную часть произведения и на вершине стека остается целое число  $INT(RND*12345)$ . Процедура PRINT VALUE выводит на экран 44 случайных числа в 2 столбца.

#### Листинг 4. 2

| АДРЕС | МАШ. КОД          | АССЕМБЛЕР      | КОММЕНТАРИЙ                  |
|-------|-------------------|----------------|------------------------------|
|       |                   | ORG 23760      |                              |
| 23760 | 3E 02             | LD A, 2        | ; Открываем канал для печат- |
| 23762 | CD 01 16          | CALL 5633      | ; ти на экран.               |
| 23765 | CD 6B 0D          | CALL 3435      | ; Очистка экрана.            |
| 23768 | 06 2C             | LD B, 44       | ; Инициализация счетчика.    |
| 23770 | C5                | PUSH BC        | ; Запомнили показания счет-  |
|       |                   |                | ; чика.                      |
| 23771 | ED 4B 76 5C       | LD BC, (23670) | ; Приняли в BC текущее зна-  |
|       |                   |                | ; чение системной перемен-   |
|       |                   |                | ; ной SEED.                  |
| 23775 | CD 2B 2D          | CALL 11563     | ; Переброска содержимого BC  |
|       |                   |                | ; на стек калькулятора.      |
| 23778 | EF                | RST 40         | ; Включение калькулятора.    |
| 23779 | A1                | stack_one      | ; На стеке (SEED), 1         |
| 23780 | 0F                | add            | ; (SEED)+1                   |
| 23781 | 34 37 16          | stack_number   | ; На стек помещаем число 75. |
| 23784 | 04                | multiply       | ; ((SEED)+1)*75              |
| 23785 | 34 80 41 00 00 80 | stack_number   | ; Число 65537                |
| 23791 | 32                | n_mod_m        | ; Деление на 65537 с остат-  |
|       |                   |                | ; ком.                       |
| 23792 | 02                | delete         | ; Частное удалили, оставили  |
|       |                   |                | ; только остаток.            |
| 23793 | A1                | stack_one      | ; "остаток", 1               |
| 23794 | 03                | subtract       | ; "остаток"-1                |
| 23795 | 31                | duplicate      | ; На стеке сделали копию.    |
| 23796 | 36                | endcalc        | ; Выключение калькулятора.   |
| 23797 | C0 A2 2D          | CALL 11682     | ; Перенос результата со сте- |
|       |                   |                | ; ка калькулятора в регист-  |
|       |                   |                | ; ровую пару BC.             |
| 23800 | ED 43 76 5C       | LD (23670), BC | ; Установка нового значения  |
|       |                   |                | ; системной переменной SEED. |
| 23804 | 7E                | LD A, (HL)     | ; Экспонента числа, находя-  |
|       |                   |                | ; шегося на вершине стека.   |
| 23805 | A7                | AND A          | ; Выставление флагов.        |
| 23806 | 28 03             | JR Z, L2       | ; Обход, если экспонента = 0 |
| 23808 | D6 10             | SUB 16         | ; Вычитание числа 16 из экс- |
|       |                   |                | ; поненты эквивалентно деле- |
|       |                   |                | ; нию на 65536.              |
| 23810 | 77                | LD (HL), A     | ; Изменили экспоненту числа, |
|       |                   |                | ; на вершине стека.          |
|       |                   |                | ; Теперь на вершине стека    |

|       |          |              |                               |
|-------|----------|--------------|-------------------------------|
|       |          |              | ; случайное число в диапазо-  |
|       |          |              | ; не от 0 до 1.               |
| L2    |          |              |                               |
| 23811 | 01 39 30 | LD BC, 12345 |                               |
| 23814 | CD 2B 2D | CALL 11563   | ; Число 12345 DEC переносим   |
|       |          |              | ; на стек.                    |
| 23817 | EF       | RST 40       | ; Включение калькулятора.     |
| 23818 | 04       | multiply     | ; Умножение. На стеке слу-    |
|       |          |              | ; чайное число от 0 до 12345  |
| 23819 | 27       | int          | ; Выделение целой части.      |
| 23820 | 38       | endcalc      | ; Выключение калькулятора.    |
| 23821 | 3E 02    | LD A, 02     |                               |
| 23823 | CD 01 16 | CALL 5633    | ; Открыли канал печати на     |
|       |          |              | ; экран.                      |
| 23826 | CD E3 2D | CALL 11747   | ; Процедура PRINT_FP. Печата- |
|       |          |              | ; ет действительное число,    |
|       |          |              | ; находящееся на вершине      |
|       |          |              | ; стека калькулятора.         |
| 25829 | 3E 06    | LD A, 06     |                               |
| 23831 | D7       | RST 16       | ; Печать кода CHR\$ 6.        |
| 23832 | C1       | POP BC       | ; Восстановление счетчика.    |
| 23833 | 10 BF    | DJNZ L1      | ; Возврат на повторный шаг.   |
| 23835 | C9       | RET          | ; Выход из процедуры.         |

Демонстрационная программа 4.3 показывает, как применяется процедура RND. Здесь эта процедура используется для задания случайным образом цвета и высоты домов в городе и может быть использована в программах типа "Бомбардировщик". Все необходимые символы UDG здесь уже определены и Вы можете легко усовершенствовать эту программу для своих целей.

Начальное изображение на экране формируется с помощью БЕЙСИКа, а последующее, через несколько секунд, с помощью программы в машинных кодах, что позволяет сравнить скорость выполнения функции RND. Чтобы не менять начальное значение для генератора случайных чисел, используйте для вызова программы в машинных кодах не RANDOMIZE USR "адрес", а LET "переменная" = USR "адрес". Можно использовать команду PRINT USR "адрес", если Вы хотите получить число, записанное в регистровой паре BC после выхода из процедуры в машинных кодах. Для установки цветов PAPER и INK в процедуре используется

```
LD (IY+83), 47
вместо
LD A, 47 и
LD (IY+83), A.
```

как это было в предыдущих примерах, что позволяет уменьшить требуемый для процедуры объем памяти.

Перед вызовом генератора случайных чисел в регистр A записывается значение коэффициента n в формуле  $RND \cdot n$ , которое затем передается в ячейку по адресу 23681 (в стандартной версии это неиспользуемая ячейка в области системных переменных). Это число округляется до ближайшего целого числа и записывается опять в регистр A, а оттуда, с помощью процедуры CALL 11733 переписывается на вершину стека.

Необходимая для организации вывода на экран информация находится в буфере принтера в следующей последовательности:

```
AT(22), x,y, INK(16),a$, b$.
```

## 5. ОБСЛУЖИВАНИЕ КЛАВИАТУРЫ

Существуют два способа создания программ в машинных кодах для опроса состояния клавиатуры. Оба способа вполне приемлемы, но конкретный выбор одного из них зависит от типа программы.

### 5.1. Использование системной переменной LAST\_KEY.

В области памяти ОЗУ компьютера "ZX-Spectrum" имеется процедура KEYSKAN которая вызывается каждую 1/50 секунды для опроса состояния клавиатуры и изменения соответствующих системных переменных. В случае, если будет нажата произвольная клавиша, то установится 5-й бит системной переменной FLAGS(23611 DEC = 5C3B HEX), сигнализирующий о факте нажатия клавиши, а код нажатой клавиши запишется в системную переменную LAST\_KEY. Если же не будет нажата ни одна клавиша, то при каждом вызове процедуры KEYSKAN будет увеличиваться на 1 только значение системной переменной FRAMES.

Поскольку команда процессора HALT приостанавливает выполнение программы до очередного вызова процедуры KEYSKAN, то можно реализовать в машинных кодах программу, аналогичную по действию BASIC-команде PAUSE 0 (см. листинг 5.1). Эта программа зациклена на непрерывной проверке 5-го бита системной переменной FLAGS и, как только будет обнаружено, что этот бит включен (т.е. после очередного нажатия клавиши), выполнение программы будет продолжено дальше. Необходимо при этом сразу же сбросить 5-й бит переменной FLAGS для того, чтобы при очередном вызове процедуры опять сработала задержка до нажатия произвольной клавиши.

Листинг 4.3

| АДРЕС | МАШ. КОД    | АССЕМБЛЕР      | КОММЕНТАРИЙ                                                            |
|-------|-------------|----------------|------------------------------------------------------------------------|
|       |             | ORG 23760      |                                                                        |
| 23760 | 3E 00       | LD A, 0        | ; Черный цвет бордюра.                                                 |
| 23762 | CD 9B 22    | CALL 8859      | ; Окраска бордюра.                                                     |
| 23765 | FD 36 53 2F | LD (IY+83), 47 | ; Установка цвета:<br>; PAPER = 5; INK = 7                             |
| 23769 | 3E 02       | LD A, 2        | ; Канал экрана.                                                        |
| 23771 | CD 01 16    | CALL 5633      | ; Открыли канал.                                                       |
| 23774 | CD 6B 0D    | CALL 3435      | ; Очистка экрана CLS.                                                  |
| 23777 | FD 21 00 5B | LD IY, 23296   | ; Регистровая пара IY становится<br>; указателем на буфер принтера.    |
| 23781 | FD 36 00 16 | LD (IY+0), 22  | ; В буфер принтера помещаем<br>; код 22 – код AT.                      |
| 23785 | FD 36 03 10 | LD (IY+03), 16 | ; Туда же (со смещением +3)<br>; помещаем код 16 – код INK.            |
| 23789 | FD 21 3A 5C | LD IY, 23610   | ; Восстановление IY в ис-<br>; ходное состояние.                       |
| 23793 | 3E 1F       | LD A, 31       | ; Счетчик столбцов экрана.                                             |
| L1    |             |                |                                                                        |
| 23795 | 32 02 5B    | LD (23298), A  | ; Текущую координату X по-<br>; местили на ее место в буфере принтера. |
| 23798 | 3E 0A       | LD A, 10       | ; Множитель 10 для RND.                                                |
| 23800 | 32 81 5C    | LD (23681), A  | ; Сохраняем его в 23681.                                               |
| 23803 | CD 61 5D    | CALL RND       | ; Вызов копии процедуры RED.<br>; Копия организована в 23905           |
| 23806 | 6F          | LD L, A        | ; Случайное число.                                                     |
| 23807 | 3E 13       | LD A, 19       |                                                                        |
| 23809 | 95          | SUB L          | ; 19-RND*10 – это Y-координата<br>; позиции печати.                    |
| 23810 | 32 01 5B    | LD (23297), A  | ; Заложили ее на свое место<br>; в буфере принтера.                    |
| 23813 | 3E 02       | LD A, 2        |                                                                        |
| 23815 | 32 81 5C    | LD (23681), A  | ; Множитель при RND равен 2.                                           |
| 23818 | CD 61 5B    | CALL RND       | ; INT (RND*2)                                                          |
| 23821 | C6 94       | ADD A, 148     | ; INT (RND*2) + 148 – выбор<br>; символа UDG – 148 или 149.            |
| 23823 | 32 05 5B    | LD (23301), A  | ; Код печатаемого символа<br>; переносим в буфер принтера              |
| 23826 | 3E 04       | LD A, 4        |                                                                        |
| 23826 | 32 81 5C    | LD (23681), A  | ; Множитель при RND равен 4.                                           |
| 23831 | CD 61 5D    | CALL RND       | ; INT (RND*4)                                                          |
| 23834 | 32 04 5B    | LD (23300), A  | ; Код цвета INK 0...3                                                  |



|       |                   |               |                                                                     |
|-------|-------------------|---------------|---------------------------------------------------------------------|
| 23837 | 3E 02             | LD A,02       | ; Канал 2.                                                          |
| 23839 | CD 01 16          | CALL 5633     | ; Открыли канал экрана.                                             |
| 23842 | 3E 16             | LD A,22       |                                                                     |
| 23644 | D7                | RST 16        | ; Печать кода 22 - AT.                                              |
| 23845 | 3A 01 5B          | LD A, (23297) | ; Координата Y.                                                     |
| 23848 | 3D                | DEC A         | ; Y-1                                                               |
| 23849 | D7                | RST 16        | ; ...AT (Y-1)....                                                   |
| 23850 | 3A 02 5B          | LD A, (23298) | ; Координата X.                                                     |
| 23853 | D7                | RST 16        | ; ...AT (Y-1),X                                                     |
| 23854 | 3E 93             | LD A,147      | ; Символ UDG (CHR\$ 147).                                           |
| 23856 | D7                | RST 16        | ; PRINT AT (Y-1),X ;CHR\$ 147 L2                                    |
| 23857 | 11 00 5B          | LD DE,23296   | ; Начало буфера принтера.                                           |
| 23860 | 01 06 00          | LD BC,06      | ; Шесть кодов из буфера.                                            |
| 23863 | CD 3C 20          | CALL 8252     | ; Печать серии символов.                                            |
| 23866 | 3A 01 5B          | LD A, (23297) | ; Координата Y.                                                     |
| 23869 | 3C                | INC A         |                                                                     |
| 23870 | 32 01 5B          | LD (23297),A  | ; Y=Y+1                                                             |
| 23873 | FE 15             | CP 21         | ; Проверка на конец игрово-<br>го поля по вертикали.                |
| 23875 | 20 EC             | JR NZ,L2      | ; Возврат на L2, если конец<br>не достигнут.                        |
| 23877 | 3E 99             | LD A,153      | ; Код символа UDG.                                                  |
| 23879 | 32 05 5B          | LD (23301),A  |                                                                     |
| 23882 | 3E 07             | LD A,7        | ; Белый цвет INK.                                                   |
| 23884 | 32 04 5B          | LD (23300),A  |                                                                     |
| 23887 | 11 00 5B          | LD DE,23296   | ; Начало буфера принтера.                                           |
| 23890 | 01 06 00          | LD BC,06      | ; Шесть кодов из буфера.                                            |
| 23893 | CD 3C 20          | CALL 8252     | ; Печать управляющих кодов.                                         |
| 23896 | 3A 02 5B          | LD A, (23298) | ; Координата X.                                                     |
| 23899 | 3D                | DEC A         | ; X=X-1                                                             |
| 23900 | FE FF             | CP FF         | ; X сравнили с -1.                                                  |
| 23902 | 20 93             | JR NZ,L1      | ; Если еще не все столбцы<br>обработаны, то возврат на<br>метку L1. |
| 23904 | C9                | RET           | ; Возврат.                                                          |
|       |                   | ORG 23905     |                                                                     |
| RND   |                   |               |                                                                     |
| 23905 | ED 4B 76 5C       | LD BC,(23670) | ; Значение SEED,                                                    |
| 23909 | CD 2B 2D          | CALL 11563    | ; Переносим его на стек.                                            |
| 23912 | EF                | RST 40        | ; Включение калькулятора.                                           |
| 23913 | A1                | stack_one     |                                                                     |
| 23914 | 0F                | add           |                                                                     |
| 23915 | 34 37 16          | stack_number  | Копия системной процедуры                                           |
| 23918 | 04                | multiply      | RND. Комментарий см. в 4.1                                          |
| 23919 | 34 80 41 00 00 80 | stack_number  |                                                                     |
| 23925 | 32                | n_mod_m       |                                                                     |
| 23926 | 02                | delete        |                                                                     |
| 23927 | A1                | stack_one     |                                                                     |
| 23928 | 03                | subtract      |                                                                     |
| 23929 | 31                | duplicate     |                                                                     |
| 23930 | 38                | endcalc       |                                                                     |
| 23931 | CD A2 2D          | CALL 11682    |                                                                     |
| 23934 | ED 43 76 5C       | LD (23670),BC |                                                                     |
| 23938 | 7E                | LD A,(HL)     |                                                                     |
| 23939 | A7                | AND A         |                                                                     |
| 23940 | 28 03             | JR Z,L3       |                                                                     |
| 23942 | D6 10             | SUB 16        |                                                                     |
| 23944 | 77                | LD (HL),A     |                                                                     |
| L3    |                   |               |                                                                     |
| 23945 | 3A 81 5C          | LD A,(23681)  | ; Множитель при RND.                                                |
| 23948 | CD 28 2D          | CALL 11560    | ; Поместили его на стек.                                            |
| 23951 | EF                | RST 40        | ; Включение калькулятора.                                           |
| 23952 | 04                | multiply      | ; Умножение.                                                        |
| 23953 | 27                | int           | ; Выделение целой части.                                            |

|       |          |            |                              |
|-------|----------|------------|------------------------------|
| 23954 | 38       | endcalc    | ; Выключение калькулятора.   |
| 23955 | CD D5 2D | CALL 11733 | ; Перенос результата в акк-р |
| 23958 | C9       | RET        | ; Выход из процедуры RND.    |

## Листинг 5.1

| АДРЕС     | МАШ. КОД    | АССЕМБЛЕР     | КОММЕНТАРИЙ                                                           |
|-----------|-------------|---------------|-----------------------------------------------------------------------|
| ORG 23760 |             |               |                                                                       |
| L1        |             |               |                                                                       |
| 23760     | FD CB 01 6E | BIT 5, (IY+1) | ; Проверка 5-го бита системной<br>; переменной FLAGS.                 |
| 23764     | 28 FA       | JR Z, L1      | ; Зацикливание на метку L1,<br>; если никакая клавиша не нажата.      |
| 23766     | FD CB 01 AE | RES 5, (IY+1) | ; Принудительное выключение<br>; 5-го бита, если клавиша была нажата. |
| 23770     | C9          | RET           | ; Выход из процедуры.                                                 |

## 5.2 Опрос клавиатуры, как внешнего порта: IN A, (C)

Аналогично команде БЕЙСИКа IN, при этом способе проверяется состояние одной из половин ряда клавиатуры. Спецификация проверяемого ряда находится в регистровой паре BC, а код нажатой клавиши передается в регистр A. Так как каждая половина ряда клавиатуры включает по 5 клавиш, то для идентификации нажатой клавиши используют с 0 по 4 биты аккумулятора. Нулевой бит соответствует внешней клавише ряда, а четвертый бит соответствует внутренней клавише полуряда клавиатуры.

Все биты регистра A изначально установлены и, если не нажата ни одна из клавиш, то значение регистра A должно быть равно 255. Если же одна из клавиш будет нажата, то сбрасывается бит, соответствующий этой клавише. Например, если будет нажата клавиша "0", то сбрасывается нулевой бит регистра A и значение этого регистра теперь будет равно 254. При этом способе можно определить одновременное нажатие более чем одной клавиши путем анализа битов аккумулятора.

Этот способ оказывается весьма полезным для создания игр, в которых участвуют два и более игроков, либо для организации перемещения объектов по экрану одновременно в двух направлениях. Одним словом, этим приемом пользуются в тех случаях, когда надо обрабатывать нажатие пользователем второй клавиши при неотпущенной первой. В программе 5.2 показано, как осуществить задержку в выполнении программ до тех пор, пока не будет нажата клавиша "0".

Есть еще один случай, когда опрос клавиатуры как внешнего порта оказывается единственным рабочим приемом - случай, когда прерывания работы процессора отключены командой DI.

Даже во время выполнения Вашей собственной программы, написанной в машинных кодах, 50 раз в секунду будет вызываться процедура KEYSCAN и, в результате, увеличивается время выполнения Вашей программы. Можно отменить эти прерывания, чтобы сократить время выполнения программы, используя команду DI. Однако, это будет означать, что все прерывания запрещены и, следовательно, нельзя будет использовать процедуру LAST\_KEY для считывания состояния клавиатуры, т.е. команда IN остается единственным доступным способом.

В программе 5.3 показан порядок работы при запрете прерываний.

Всегда помните, что перед выходом из процедуры, где Вы используете команду DI, необходимо записать команду EI, разрешающую прерывания. Попробуйте выполнить программу 5.3 без этой команды и увидите, что из этого получится. Сброса программы не произошло, но компьютер "завис" и для того, чтобы клавиатура вновь заработала, необходимо выключить и опять включить компьютер.

Программа 5.4 является аналогом БЕЙСИК-команды PAUSE n, т.е. позволяет остановить выполнение программы на заданный промежуток времени, либо продолжить выполнение сразу же после нажатия произвольной клавиши. Эта программа основана на том, что сканирование клавиатуры происходит строго через каждые 0.02 секунды, а команда HALT приостанавливает выполнение программы до очередного сканирования клавиатуры.

## Листинг 5.2

| АДРЕС       | МАШ. КОД | АССЕМБЛЕР    | КОММЕНТАРИЙ                                                                                                     |
|-------------|----------|--------------|-----------------------------------------------------------------------------------------------------------------|
| L1<br>23760 | 01 FE EF | LD BC, 61438 | ; Установка в BC адреса<br>; внешнего порта, соответ-<br>; ствующего верхнему правому<br>; полуряду клавиатуры. |
| 23763       | ED 78    | IN A, (C)    | ; Ввод данных из порта в<br>; аккумулятор.                                                                      |
| 23705       | CB 47    | BIT 0, A     | ; Проверка нулевого бита,<br>; который соответствует<br>; внешней клавише полуряда.                             |
| 23767       | C8       | SET Z        | ; Если бит выключен, то кла-<br>; виша нажата, следует воз-<br>; врат из процедуры.                             |
| 23768       | 18 F6    | JR L1        | ; В противном случае состоя-<br>; ние ожидания следует про-<br>; должить и вернуться на L1.                     |

## Листинг 5.3

| АДРЕС          | МАШ. КОД       | АССЕМБЛЕР             | КОММЕНТАРИЙ                                                                                                      |
|----------------|----------------|-----------------------|------------------------------------------------------------------------------------------------------------------|
| 23760<br>L1    | F3             | ORG 23760<br>DI       | ; Запрет прерываний.                                                                                             |
| 23761          | 01 FE EF       | LD BC, 61438          | ; Установка в BC адреса<br>; внешнего порта, соответ-<br>; ствующего верхнему правому<br>; полуряду клавиатуры.  |
| 23764          | ED 78          | IN A, (C)             | ; Ввод данных из порта в<br>; аккумулятор.                                                                       |
| 23766<br>23760 | CB 47<br>20 F7 | BIT 0, A<br>JR NZ, L1 | ; Проверка нулевого бита.<br>; Если бит включен, то кла-<br>; виша не нажата. Следует -<br>; повторять проверку. |
| 23770          | FB             | EI                    | ; Разрешение прерываний пе-<br>; ред выходом.                                                                    |
| 23771          | C9             | RET                   | ; Выход из процедуры.                                                                                            |

## Листинг 5.4

| АДРЕС                            | МАШ. КОД                | АССЕМБЛЕР                              | КОММЕНТАРИЙ                                                                                                                              |
|----------------------------------|-------------------------|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| 23760<br>L1                      | 01 F4 01                | ORG 23760<br>LD BC, 500                | ; Размер паузы                                                                                                                           |
| 23763                            | 76                      | HALT                                   | ; Остановка процессора до<br>; прохождения прерывания.                                                                                   |
| 23761                            | FD CB 01 6E             | BIT 5, (IY+1)                          | ; Проверка 9-го бита систем-<br>; ной переменной FLAGS.                                                                                  |
| 23768                            | 20 05                   | JR NZ, L2                              | ; Если 5-ый бит FLAGS вклю-<br>; чился, значит была нажата<br>; какая-то клавиша и пауза<br>; прерывается, следует пере-<br>; ход на L2. |
| 23770<br>23771<br>23772<br>23773 | 0B<br>78<br>B1<br>20 F7 | DEC BC<br>LD A, B<br>OR C<br>JR NZ, L1 | ; Уменьшение счетчика.<br>; Проверка счетчика на 0.<br>; Если счетчик не обнулен,<br>; то пауза продолжается.                            |
| L2<br>23775                      | FD CB 01 AE             | RES 5, (IY+1)                          | ; Принудительное выключение<br>; пятого бита FLAGS.                                                                                      |
| 23779                            | C9                      | RET                                    | ; Выход из процедуры.                                                                                                                    |

## 6. МУЗЫКАЛЬНЫЕ И ЗВУКОВЫЕ ЭФФЕКТЫ

В этой главе мы рассмотрим некоторые способы улучшения работы Ваших программ путем включения в них специальных звуковых эффектов или музыки. Но прежде мы должны разобрать способ представления чисел с плавающей запятой в компьютере "ZX-Spectrum", а также способы записи этих чисел в упакованной форме.

### Интегральная форма записи действительных чисел.

Любые числа: целые, дробные, положительные, отрицательные, могут быть представлены в 5-байтной форме. Первый байт в такой форме записи числа определяет экспоненту, а оставшиеся четыре - мантиссу числа. Перевод обычной записи числа в 5-байтную форму осуществляется в четыре этапа:

1. Переводим число в его эквивалент в двоичной системе счисления.

Например:

$11,375 = 1011,011.$

Напомним, что в двоичной системе счисления

$0,5 = 0,1$

$0,25 = 0,01$

$0,125 = 0,001$  и т.д.

2. Определяем экспоненту числа. Для этого перемещаем десятичную точку влево или вправо, пока не дойдем до крайней единицы справа или слева. При этом считаем число шагов перемещений. Экспонента будет равна 128 плюс число шагов. Если движение производилось влево, то оно считается положительным, а если вправо - то отрицательным. В нашем примере получим после четырех шагов вправо:

$.1011011$

и, следовательно, экспонента равна  $128+4 = 132$ .

3. Меняем состояние бита справа от десятичной точки в зависимости от того, положительным или отрицательным было наше исходное число. Если число было положительным, то этот бит надо выключить, а если отрицательным, то его надо оставить включенным. Таким образом, этот бит будет сигнализировать о знаке числа.

В нашем примере число было положительным и поэтому после смены состояния бита оно будет иметь вид:

$.00110110$

4. Определяем мантиссу числа. Для этого, не обращая внимание на десятичную точку, добавим справа определенное количество нулей, чтобы получить 32-х разрядное число. Теперь распределим 32 разряда числа между четырьмя байтами и определим значение каждого байта:

00110100 (54)    00000000 (0)

00000000 (0)    00000000 (0)

Таким образом, 5-байтное представление числа 11,375 будет иметь следующий вид:

132   54   0   0   0

Если Вы хотите проверить все вышесказанное, то Вам необходимо ввести короткую программу, позволяющую считывать 5 последовательных байтов (например, через PEEK) из участка памяти, где записаны переменные BASIC-программы. Значения переменных в этой области хранятся именно в 5-ти байтной форме.

Теперь вернемся к главной теме этой главы. Операнды BASIC-команды BEEP, определяющие длительность и тональность звука, записываются в калькуляторный стек. Калькуляторный стек оперирует и хранит числа, записанные именно в 5-байтной форме. Поэтому, чтобы написать программу в машинных кодах для получения звука или музыки, необходимо значение этих операндов занести в калькуляторный стек именно в 5-байтной форме.

Процедура BEEP, находящаяся по адресу 1016, извлекает эти два числа с

калькуляторного стека и воспроизводит звуковой сигнал в соответствии со значениями этих чисел. Например, для получения эффекта, аналогичного команде БЕЙСИКа ВЕЕР .1,11, необходимо числа 0.1 и 11 вначале перевести в их 5-байтную форму:

```
0.1 = 125 76 204 204 204
11 = 132 48 0 0 0
```

Теперь необходимо передать последовательно эти 10 байтов на калькуляторный стек. Как это сделать, показано в программе 6.1. Здесь используется команда передачи блока LDIR. При этом в регистровую пару HL записывается адрес начала этих 10 байтов в памяти, а в регистровую пару DE - адрес вершины стека, который постоянно хранится в системной переменной STKEND (23653). В регистровую пару BC записывается число перемещаемых байтов. Процедура ВЕЕР извлекает два числа в 5-байтной форме с вершины стека и адрес вершины стека восстанавливается, т.е. становится таким же, как и до записи туда чисел.

#### Листинг 6.1

```

23760      ED 5B 65 5C      ORG 23760
                        LD DE, (23653)      ; В DE установили то, что
                                           ; хранится в STKEND
23764      21 E4 5C      LD HL, DATA      ; В HL - адрес наших 10 байтов
23767      01 0A 00      LD BC, 10         ; В BC - количество пересылаемых байтов
23770      ED B0          LDIR              ; Переброска данных на машинный
                                           ; стек калькулятора
23772      ED 53 65 5C      LD (23653), DE   ; Переустановили новое значение STKEND
23776      CD F8 03      CALL 1016          ; Вызов процедуры ВЕЕР
23779      C9            RET                ; Выход

23780      DATA          DEFM 125, 76, 204, 204, 204      ; Число 0.1
23785                                DEFM 132, 48, 0, 0, 0      ; Число 11

```

Другой способ передачи чисел на стек калькулятора основан на использовании команды LD A, n и процедуры ПЗУ STACK\_A, находящейся по адресу 11560, если передаются целые числа в диапазоне 0...255.

Если надо передать целые числа из диапазона 0...65535, то можно воспользоваться переброской через регистровую пару BC и вызов процедуры ПЗУ STACK\_BC, находящейся по адресу 11563. В программе 6.2 показано, как можно передать в калькуляторный стек число 11, используя регистр A.

#### Листинг 6.2

```

23760      ED 5B 65 5C      ORG 23760
                        LD DE, (23653)      ; В DE установили то, что
                                           ; хранится в STKEND
23764      21 E9 5C      LD HL, DATA      ; В HL - адрес наших 10 байтов
23767      01 05 00      LD BC, 5         ; В BC - количество пересылаемых байтов
23770      ED B0          LDIR              ; Переброска данных на машинный
                                           ; стек калькулятора
23772      ED 53 65 5C      LD (23653), DE   ; Переустановили новое значение STKEND
23776      3E 0B          LD A, 11          ; В аккумуляторе число 11
23778      CD 28 2D      CALL 11560        ; Число 11 помещено на вершину
                                           ; стека калькулятора
23781      CD F8 03      CALL 1016          ; Вызов процедуры ВЕЕР
23784      C9            RET                ; Выход

23785      DATA          DEFM 125, 76, 204, 204, 204      ; Число 0.1

```

### Упакованная форма записи действительных чисел.

Теперь познакомимся с записью чисел с плавающей запятой в упакованной форме и с использованием кода калькулятора 52DEC для работы с числами, находящимися на калькуляторном стеке в этой форме.

Перевод 5-байтного числа в упакованную форму включает в себя следующие этапы:

1. Начиная с пятого байта, убираем все числа, равные нулю до тех пор, пока не выйдем на ненулевое число или не достигнем второго байта 5-байтной записи числа.
2. Определяем число удаленных байтов  $k$  и рассчитываем число  $g = 3 - k$ .
3. Вычитаем 80 из числа, записанного в первом байте и, если результат больше или равен 64, то переходим к п.6.
4. Меняем значение экспоненты (число в первом байте) на число: экспонента минус 80 плюс  $g \cdot 64$ .
5. Переходим к п.7.
6. Экспонента числа занимает 2 байта. Значение первого байта равно  $g \cdot 64$ , а второго 80.
7. Записываем оставшиеся байты после модифицированной экспоненты.

Для выполнения обратного перехода необходимо:

1. Разделить значение первого байта на 64.
2. Если будет остаток от деления, то экспонента будет равна  $80 + \text{остаток}$ .
3. Если остатка от деления на 64 не будет, то экспонента равна 80 плюс значение второго байта.
4. Частное от деления на 64 используется для определения количества последующих байтов, это количество равно частному от деления плюс 1.

Например, для числа 11 упакованная форма имеет вид:

52 48

Делим 52 на 64 и получаем: частное = 0, а остаток = 52. Значит, после экспоненты, равной  $80 + 52 = 132$ , следует только  $0 + 1 = 1$  байт. В упакованной форме записи он равен 48. Следовательно, 5-байтная запись числа 11 будет иметь вид:

132 48 0 0 0

Программа 6.3 показывает, как может использоваться упакованная форма для реализации команды ВЕЕР .1,11 в машинных кодах.

Листинг 6.3

| АДРЕС | МАШ. КОД             | АССЕМБЛЕР | КОММЕНТАРИЙ                |
|-------|----------------------|-----------|----------------------------|
|       |                      | ORG 23760 |                            |
| 23760 |                      | RST 40    | ; Включение калькулятора.  |
| 23761 | EF 34 ED 4C CC CC CC | stk_data  | ; На стеке 0.1             |
| 23767 | 34 34 30             | stk_data  | ; На стеке 0.1, 11         |
| 23770 | 38                   | endcalc   | ; Выключение калькулятора. |
| 23771 | CD FB 03             | CALL 1016 | ; Вызов процедуры ВЕЕР.    |
| 23784 | C9                   | RET       | ; Выход.                   |

Для получения различных звуковых эффектов можно использовать процедуру ВЕЕР с точкой входа 949. При этом значения длительности и тональности звука подаются не на стек калькулятора, а соответственно в регистровые пары DE и HL.

Программа 6.4 показывает, как можно программно менять тональность звука, используя процедуру ВЕЕР в цикле. При этом будет меняться и длительность, потому что длительность звучания является функцией значений, записанных в регистровых парах HL и DE.

Листинг 6.4

| АДРЕС | МАШ. КОД | АССЕМБЛЕР | КОММЕНТАРИЙ                         |
|-------|----------|-----------|-------------------------------------|
|       |          | ORG 23760 |                                     |
| 23760 | 06 FF    | LD B, 255 | ; Инициализация счётчика.           |
| 23762 | 21 01 00 | LD HL, 1  | ; Инициализация высоты тона.        |
| 23765 | 11 01 00 | LD DE, 1  | ; Инициализация длительности звука. |
| 23768 | E5       | PUSH HL   | ; Надо предохранить HL и BC         |
| 23769 | C5       | PUSH BC   | ; от порчи процедурой ВЕЕР.         |
| 23770 | CD B5 03 | CALL 949  | ; Вызов процедуры ВЕЕР.             |
| 23773 | C1       | POP HL    |                                     |
| 23774 | E1       | POP BC    |                                     |
| 23775 | 23       | INC HL    | ; Повышение высоты тона.            |
| 23776 | 10 F3    | DJNZ LOOP | ; Повтор цикла.                     |
| 23778 | C9       | RET       | ; Выход из процедуры.               |

Существует ещё один метод воспроизведения звука в компьютерах Spectrum. Для этого нужно знать частоту смены высокого и низкого уровня для каждой ноты и отправлять сигнал прямо на динамик, минуя процедуры ПЗУ. Этот метод можно использовать в игровых программах для создания "белого шума", эффекта взрыва и т.д. При этом используется команда OUT (254),A. Причем этой командой можно не только управлять работой динамика, но и менять цвет рамки экрана (цвет BORDER).

1. Если только нулевой бит регистра A включен, то BORDER - синий.
2. Если только первый бит регистра A включен, то BORDER - красный.
3. Если только второй бит регистра A включен, то BORDER - зеленый.
4. Если бит 4 регистра A включен, то на динамик компьютера идет высокий сигнал, а если этот бит сброшен - то низкий.

Понятно, что если периодически менять состояние 4-го бита регистра A и использовать команду OUT (254),A, то частота звука из динамика будет пропорциональна частоте смены состояния 4-го бита. Для создания "белого шума" период чередования состояния 4-го бита должен быть случайным, но при использовании стандартной процедуры генерации случайных чисел, программа будет выполняться слишком медленно. Поэтому лучше использовать в качестве случайных чисел содержимое в каких-то ячейках памяти и, тем самым, переключать уровень сигнала на динамике (см. программу 6.5). В этой программе в одном цикле трижды используется команда

OUT (254),A,

а также ряд команд NOP, что необходимо для создания временной задержки.

Длительность звука ограничена. Для этого используется команда

CP 60

Цвет BORDER также меняется случайным образом, чтобы этот цвет оставался постоянным, необходимо переписать биты 3-5 с системной переменной BORDER в биты 0-2 регистра A.

## Листинг 6.5

|       |          |               |                                   |
|-------|----------|---------------|-----------------------------------|
|       |          | ORG 23760     |                                   |
| 23760 | 21 00 00 | LD HL, 00     | ; Мы начинаем брать данные для    |
|       |          |               | ; "белого шума" прямо из ПЗУ.     |
| 23763 | 7E       | L1 LD A, (HL) | ; Очередной байт из ПЗУ.          |
| 23764 | 00       | NOP           | ; Пауза.                          |
| 23765 | D3 FE    | OUT (254),A   | ; Выдача сигнала на динамик.      |
| 23757 | 00       | NOP           |                                   |
| 23758 | 00       | NOP           |                                   |
| 23769 | D3 FE    | OUT (254),A   |                                   |
| 23771 | 00       | NOP           |                                   |
| 23772 | 00       | NOP           |                                   |
| 23773 | D3 FE    | OUT (254),A   |                                   |
| 23775 | 00       | NOP           |                                   |
| 23776 | 23       | INC HL        | ; Переход к следующей ячейке ПЗУ. |
| 23777 | 7C       | LD A, H       |                                   |
| 23778 | FE 3C    | CP 60         | ; Проверка на конец работы.       |
| 23780 | 20 ED    | JR NZ, L1     | ; Возврат к началу цикла.         |
| 23782 | C9       | RET           | ; Выход из процедуры.             |

Если Вы хотите поэкспериментировать с созданием музыки, используя этот метод, то вам необходимо помнить, что каждую 1/50 секунды ваша программа будет прерываться процедурой сканирования клавиатуры. Поэтому, чтобы сохранить частоту звука, необходимо в начале программы запретить прерывания командой DI, а в конце программы вновь разрешить их командой EI. Кстати, именно поэтому невозможно прервать выполнение команды BEEP клавишей BREAK.

Помните также, что в игровых программах звуковые эффекты не должны быть слишком длинными, т.к. на это время может приостанавливаться движение персонажей.

(Продолжение следует)

# FORUM

Нам пишет Галунов В. А.:

Расскажите, пожалуйста в "ZX-РЕВЮ-93" о спрайтах: что это такое (формулировка). Как их хранить в памяти, как ими пользоваться и как создавать. Расскажите, пожалуйста о мультипликации. Если Вы намечаете осветить эти вопросы в своих книгах о графике "Спектрума", то большое спасибо...

ИФК: Спасибо Вам, уважаемый читатель за хороший вопрос. Действительно, все о спрайтах войдет в третий том "Графики Спектрума", который называется "Динамическая графика" и работа над которым сейчас близится к концу.

КОРР: ... Все Ваши книги я с радостью покупаю и буду покупать.

ИФК: После этих слов хочется работать, работать и работать.

КОРР: Да, кстати, побольше рассказывайте об адаптации программ под TR-DOS и о многом другом, связанном с этой системой, ведь все большее и большее число пользователей приобретает дисководы.

ИФК: К сожалению все большее число пользователей уходит не только в TR-DOS, но и еще дальше - на работу с IBM-совместимыми машинами. Поскольку мы всегда думаем не только о завтрашнем дне, но и о послезавтрашнем, то и сами постепенно дрейфуем в том же направлении. Т.е. все меньше людей остаются на "Синклеровском" фронте и нам бывает трудно объять необъятное.

Тем не менее, нет никаких проблем для того, чтобы печатать на наших страницах те материалы, которыми, готовы поделиться наши читатели, и для TR-DOS мы всегда дадим зеленую улицу, если такие энтузиасты найдутся.

КОРР: И еще один вопрос: как снять со стека калькулятора число, чтобы работать с ним в машинном коде?

ИФК: Поделимся простым приемом. Если Вы вызовете калькулятор из маш. кода командой RST 28H, а потом тут же выйдете из него командой DEFB 38H (end\_calc), то в результате этого автоматически в регистровой паре HL установится значение, равное STKEND -5. Таким образом, Ваша регистровая пара HL укажет на начало пятибайтного числа, хранящегося на вершине стека. Далее байт за байтом Вы "прощелкаете", что там есть и перегоните его туда, куда Вам надо и так, как Вам надо. Надеемся, что с форматом записи таких чисел Вы знакомы, а если нет, то надеемся, что заказанная вами литература дошла успешно и Вы этот вопрос разберете.

Это, так сказать, теория, а на практике Вам вряд ли когда захочется обрабатывать вне калькулятора действительные числа с плавающей точкой. Скорее всего, Вам и нужно-то взять оттуда целое число в диапазоне от -65535 до +65535. А раз это так, то можно сделать еще проще, воспользовавшись готовой процедурой ПЗУ, которая называется FP\_TO\_BC и находится, по адресу 2DA2 (11682 DEC). Вызовом этой процедуры Вы округлите то число, которое есть на вершине стека калькулятора до целого и перегоните его в регистровую пару BC, после чего можете делать с ним все, что хотите.

Как Вы понимаете, не всякое действительное число может быть "втиснуто" в такие ограниченные рамки. Может быть, оно туда и не "влезло" и процедура не сработала. Чтобы узнать об этом, надо после вызова процедуры FP\_TO\_BC проверить состояние некоторых флагов регистра F.

Если результат слишком велик, т.е. больше 65535, то процедура включает флаг переноса CARRY.

О том, что число отрицательное сигнализирует выключенный флаг нуля - флаг ZERO.

\* \* \*

Нам пишет Д. Зайцев из Пензы:

Я провел небольшое исследование программы ELITE. Байты 16 и 17 - это внутренние часы игры. Только по определенным значениям этих байтов Вам предлагают выполнить



МИССИИ....

ИФК: Напомним читателям, что речь идет о байтах отгружаемого блока при сохранении состояния. Байты 16,17 не были идентифицированы ранее (см. ZX-P-92, N 5,6, стр. 123,124).

КОРР:... Например, если заслать в 16 байт число 31, то после гиперперехода Вам предложат выполнить первую миссию. Если же записать в этот байт число 128, то Вам встретятся корабли-невидимки, при этом байт 93 должен быть обнулен (он свидетельствует о наличии на борту CLOAKING DEVICE). Это надо сделать, потому что если там находится число 67, то ни первую, ни вторую миссию Вам уже не предложат. Числа, которые должны быть записаны в 16 и 17 байты для перехода к третьей миссии мне неизвестны, т.к. я имею рейтинг DANGEROUS (в байтах с 13 по 15 у меня числа 175,2,9).

Теперь о неизвестных байтах 71-74. По моему мнению, это данные о планете, на которой Вы находитесь. К примеру, 73 байт - это техническое развитие планеты.

Неизвестный 92-ой байт - это байт, отвечающий за третью миссию. Если Вы запишите в него число 1 или 2, то Вам предложат ее выполнить. Если Вы установите в этом байте число 3, то миссию Вам уже предложили и Вы можете лететь взрывать станцию, захваченную Таргонами.

\* \* \*

Нам пишет Пушкарев Александр Николаевич из г. Петропавловска-Камчатского:

Я по поводу ELITE. Копаясь в JOYSTICK CLUB - версии я нашел один интересный POKE 39517,0. Он отключает расход энергии. Самое интересное, что теперь можно без ограничений использовать CLOAKING DEVICE, которое из малополезного устройства, зря расходующего энергию, превратилось в очень полезную вещь.

Еще я решил исследовать один вопрос, а куда же летят все корабли?...

ИФК: Мы страшно горды тем, что вкус к исследовательской работе стал характерной чертой истинного "Синклериста".

КОРР: ... Все корабли летят со скоростью 0,5 максимальной и направляются они к звезде, где бесследно исчезают. Обычно я пристраивался в хвост и, уравнивая скорость и курс, плелся за ними следом. Интересно, что когда я выключал CLOAKING DEVICE, корабли резко набирали скорость и взмывали вверх.

ИФК: Еще бы! Кому же приятно, когда у него на хвосте сидит корреспондент "ZX-РЕВЮ"!

КОРР: Исследовал я и вопрос, почему не у всех получается заправка топливом у звезды. Дело это непростое и требует аккуратности. Надо подлететь к звезде перпендикулярно, снизить скорость до половины и ждать, когда датчик температуры поднимется тоже до половины. Только тогда появится сообщение о заправке.

Решил я попробовать и "грязный бизнес" - торговлю наркотиками. И вот, что оказалось. Примерно через 10 - 15 перелетов при входе на станцию корабль оказывался захваченным пиратами. И никаких компромиссов, это происходит всякий раз. Пираты подстраивают ловушки дельцам, занимающимся грязным бизнесом непрерывно.

МФК: Блеск! если бы Вы знали, сколько килограммов писем с жалобами на неэтичное поведение пиратов мы получаем. Оказывается, "кто-то кое-где у нас порой" ведет себя в космосе не по-христиански и бывает заслуженно за это наказан. Особенно нас обрадовало в этом сообщении то, что такую гипотезу мы выдвигали в одном из ранних выпусков "РЕВЮ" и теперь она подтвердилась.

КОРР: Хочу также отметить одну особенность JOYSTICK-CLUB-версии: здесь очень часто встречаются "Кобры" и ни разу не видел ни одного корабля "ADDER". Наверное, их заменили на "KRAIT".

Хочу также поделиться с читателями несколькими разысканными мною POKES.

1. DIZZY 3 - 63001,0 (DEC A).
2. TOI (Fase 1) - 50297,0
3. TOI (Fase 3) - 49693,0

4. TOI (Fase 4) - 49485,0
5. AT\_WAR 2 - 43831,255 (LD A,n)
6. COP OUT - 32817,255
7. NINJA COMMANDO -29076,255

8. C-TRASER - загрузить в #61A8, #88BA, #BDC0 - число не более 9. Программа выполняет проверку!

9. INDY (Last Crusade)

Блок длиной 20992 (#5200) загрузить начиная с #6000. После этого:

#0ED0,0 (не убывает оружие)

#AD44,#3C (долгая жизнь)

#ADE4,#3C

10. DRACONUS - 64215,0

11. SHOCK WAY - загрузить стандартно, а затем:

#B427,0; #B426,0.

Во многих современных играх найти POKES для "бессмертия" через аккумулятор не удастся. В программах, например DRACONUS, SHOCK WAY и др. не используется операция DEC A. Вместо аккумулятора работает регистровая пара HL. Выглядит это примерно так:

```
LD A,n
LD (адрес),A
LD HL,(адрес)
DEC HL
```

Для обессмерчивания игр я использую монитор WE88H++, который загружается в экран, благодаря чему можно грузить файлы любой длины, кроме тех, которые сами тоже грузятся в экран. Обессмертив игру, я выгружаю ее или здесь же и запускаю.

ИФК: Спасибо, Александр Николаевич! С нетерпением ждем новых открытий.

\* \* \*

Нам пишет Алексей Березин из г. Колпашево, Томской области:

Я подписался на Ваш журнал в 1991 и в 1992 г.г. и теперь с удовольствием отмечаю, что не зря потратил деньги. Особенно мне нравится раздел FORUM...

ИФК: Спасибо, Андрей! Ты уже понял, что после принятых мер он будет еще более оживленным и надеемся, что теперь он тебе понравится еще больше.

KOPP: ... Кроме того, сам я немного хаккер и люблю вскрывать разные игры. Мне даже удалось переделать игру LODERUNNER. Переделал я не только уровни, но и персонажей.

Весьма заинтересовала меня Ваша классификация версий программы "ELITE", данная в номерах 3,4 за 1992 г. Сам я давно пользуюсь версией, помеченной Джеком О'Лантерном (Jack O'Lantern presents), а недавно я нашёл и версию Joystick Club'a. Теперь имею возможность сравнить их.

Во-первых, сразу после загрузки версия J.C. выдает надпись Press Space, Commander, а версия J.O'L. - надпись "Load New Commander Yes or No?"

Если нажать "Y" и выгрузить ситуацию, а потом войти в игру, то вам дают несколько миллионов кредитов, статус Fugitive, рейтинг ELITE, а корабль забросят в 47-ю галактику, да к тому же сразу предложат и первую миссию (спасти беженцев). После этого Вы попадете в 8-ую галактику, причем если совершить оттуда межгалактический перелет, то Вы попадете в первую галактику, но ни одной знакомой звезды там не увидите, нет там ни LAVE, ни TIONISLA, нет к сожалению и планеты Raxxla.

Во-вторых, в версии J.O'L отсутствует корабль KRAIT, но доказано, что существует STRANGER, похожий на астероид...

ИФК: В свое время мы о нем неоднократно упоминали.

KOPP: Я неоднократно встречал этот "объект" в 1-ой галактике между планетами ONRIRA и LARALE и около TIONISLA, Если начать стрелять в него, он выпускает 2-3 "Сайдуиндера" или "Эддера".

В-третьих, корабли "Сайдуиндер", "Эддер" и, возможно, "Эсп-МК-2" здесь всегда пираты. Впрочем, насчет "Эспа" я не совсем уверен, хотя, кажется, ни разу не встречал

мирно настроенного "Эспа".

Захваты пиратами в версии J.O'L. очень редки. Со мной такое произошло лишь один раз...

ИФК: Это не факт. Возможно, что Вы просто вели целомудренный образ жизни и Вас не за что было наказывать?

KOPP: ... Вокруг станции всегда пустынно и не летают даже полицейские "Вайперы". Таргоны, если встречаются, летают всегда поодиночке. В версии J.O'L. работают все POKES, напечатанные Вами. Также хорошо проходит заправка топливом от звезды.

В заключение хочу предложить собственную разработку описания к программе RICK DANGEROUS, некоторые POKES и несколько "жучков".

Некоторые POKES.

1. COSA NOSTRA - 39717,0
2. REX-1 - 39402,0; 40057,0
3. REX-2 - 39176,0; 40303,0
4. RICK DAN. - 58356,0; 64075,0; 64166,0
5. STAR RUNNER - 49560,183
6. DIZZY-3 - 30273,182
7. DIZZY-4 - 63001,183
8. DYNAMITE DAN - 29002,182
9. FARENHEIT-3000 - 30818,0

Некоторые пароли.

|              |                  |
|--------------|------------------|
| FREDDY-2     | 897653           |
| HYPUSUS-2    | DROW S S APT     |
| PHANTIS-2    | 18757            |
| TRUENO-2     | 270653           |
| NAVY MOVES-2 | 63723            |
| GAME OVER-2  | 18024            |
| REX-2        | 8980818908609407 |

Некоторые секреты.

1. CHRONOS - В таблице лучших результатов набрать большими буквами слова: JING IT BABY -получите супер-лазер.

2. DOWN TO EARTH - при игре нажмите одновременно клавиши 2,3,4,5 и, удерживая их, нажмите 1. Загорится надпись "PLANET CLEARED" и Вы перейдете на следующий уровень.

ИФК: Большое спасибо, Алексей. Ну какой же "синклерист" откажется от свежего POKE. Без хорошего POKE компьютер сирота.

Правда, с описанием R.D. мы, пожалуй подождем, знаете ли, игра ведь в общем-то аркадная и требует не столько работы головой, сколько пальцами. Так что вроде бы и описание для нее не очень критично. Хотя все равно спасибо за труд, особо отметим, как красиво Вы его распечатали.

Если сможете "раскрутить" что-либо требующее специальных знаний, имитатор там какой-нибудь или, скажем адвентюру, не говоря уж о стратегических программах, нашим восторгам не будет предела. Ждем контакта.

\* \* \*

Нам пишет Чугайнов А.Г. из Владивостока.

Огромное спасибо всем, кто пишет "ИНФОРКОМу" в раздел "Форум". Мне очень помогли Ваши советы, а особенно POKES...

ИФХ: Боже! Как Вы правы! Ведь если кому и говорить спасибо, так это нашим читателям и корреспондентам, которые четыре года поддерживают наше общее дело и

добрыми словами и полезными советами и, что греха таить, материально. Сейчас ведь без крепкого рубля в кулаке ни строчки не напечатаешь.

КОРР: ... К сожалению, я не нашел пока ничего для своих любимых игр TRUENO\_2 и GAMEOVER\_2. Вы ведь знаете, некоторые фирменные игровые программы делают в нескольких частях, а если вторая часть защищена паролем, то не удастся войти в программу. Вот с такой проблемой я столкнулся...

ИФК: Здесь нам трудно удержаться от того, чтобы не поставить мегабайт восклицательных знаков. Смотрите предыдущее письмо!!!

КОРР: ...И еще. Недавно я достал 4 программы из серии DIZZY. Программа адвентюрные, а значит очень интересные...

ИФК: Не совсем так. Это не адвентюрные программы, а аркадно-адвентюрные. Но они действительно безумно интересные и, кстати, далеко не самые сложные, поскольку все действия в них довольно корректно логически обоснованы. Кстати, тех любителей этого жанра, которые планируют переходить на IBM-технику, должны предупредить, что там этот жанр представлен весьма и весьма слабо. Там развитие программного обеспечения идет по другим направлениям. У нас коллекция немалая, порядка 1000 программ, но мы испытываем грусть и ностальгию из-за нехватки аркадных адвентюр. Пока мы знаем их не более десяти.

КОРР:... Первую часть я наполовину прошел, только про некоторые предметы не знаю, для чего они, а за остальные пока еще не брался. Не могли бы Вы напечатать в "ZX-РЕВЮ", как пройти, эти программы.

ИФК: Дорогие читатели. Здесь есть масса проблем, в принципе, конечно, это можно сделать. Мы и сами их прошли, да и писем от тех, кто это сделал, накопилось немало. Несколько человек уже прислали свои проработки этих программ, еще больше получено предложений.

Дело в другом, Вы знаете клятву Гиппократы - "Не сделай больному хуже". Мы уже двадцать раз могли бы опубликовать технику прохода этих программ, но есть какое-то сомнение, которое мы никак не можем перешагнуть, ведь после этого будет утрачен весь интерес к игре, в общем, нет у нас пока такого законченного материала, который мы не побоялись бы опубликовать, а дать список "хинтов" было бы просто насильем над интересными программами, тем более что по рынкам страны и так ходит штук двадцать разных кустарных сборников "хинтов".

Давайте мы поступим так. Вы знаете, что мы начали эксперименты над новым жанром "Компьютерной новеллы". Если у нас это дело пойдет, то в выигрыше будут все, кто любит игры и тогда, может быть, мы сможем перешагнуть через тот груз ответственности, который сейчас вяжет нас по рукам и ногам.

\* \* \*

Нам пишет Денис Садошенко из г. Днепропетровска:

Здравствуй, коллеги! Очень обрадовался я заглянув на 211 страницу выпуска 9-10 "ZX-РЕВЮ"-92. Как Вы понимаете, там было описание игры BLINKY's, любезно проверенное Вашим редактором. Результатом я вполне удовлетворен, хотя все же некоторые нюансы имеются... Но я очень рад, ведь это мое первое сочинение и первая его публикация.

ИФК: Спасибо Денис! Будем работать еще лучше. Кстати, Вашу работу почти не пришлось редактировать, Вы пишете очень хорошо.

Позвольте высказать только одно пожелание. Когда Вы пишете новеллу, то перед Вами, очевидно, возникает образ человека, играющего в эту игру. Попробуйте заменить этот образ на человека, который в эту игру не играет и вообще ни в какие игры не играет, да и компьютера у него никогда не было. Попробуйте сменить акценты. На первом месте поставить не задачу дать необходимые подсказки, а просто заинтересовать его игрой, а подсказки по-возможности запрятать где иносказанием, где встроив их в какой-либо диалог, а где только намекнув на них. Посмотрите, что тогда получится, конечно, это только мнение и оно никак не должно ограничивать Ваше творчество.

Недавно у нас был очень интересный опыт именно такой проработки. Мы в несколько рук "раскрутили" игру "The Eye of Beholder" (это для IBM-совместимых машин) и передали ее для публикации нашим коллегам в журнал "МОНИТОР". Выйдет она в июне-июле. Если сможете достать этот журнал, почитайте, не пожалеете. Кстати, там почти в каждом номере есть наши статьи по игровому ПО.

KOPP: ... Скоро Ваша копилка пополнится еще одним описанием, схожем с BLINKY's - это сериал DIZZY с 1 по 6. Это та же компьютерная новелла, так же нравится моим друзьям и знакомым.

ИФК: Ждем с нетерпением. Может быть, наконец-то это именно то, на что мы никак не можем решиться?? Такой сериал нам сейчас был бы очень кстати, поскольку наш редактор раздела "Компьютерная новелла" сейчас очень занят работой над сериалом SPELLBOUND - KNIGHT Tyme - STORMBRINGER, а те, кто знают что такое адвентюрные игры, понимают, насколько круты эти программы.

KOPP: Я хочу предложить вашему вниманию список адвентюрных игр, которые я знаю. Понятно, конечно, что здесь присутствуют далеко не все. Может быть Вы его опубликуете? Пусть читатели "РЕВЮ" его дополнят.

ИФК: Мысль интересная. Таким способом можно определить наличие игр этого жанра, имеющих хождение по СНГ. Правда, в Вашем списке нет названий фирм и этот недостаток существенен, поскольку многие пополняют свои игротеки не по названиям игр, а по фирмам. Точно так же, как Вы собираете себе библиотеку по любимым авторам. Где смогли, мы дали названия фирм от себя.

#### ИГРЫ ЖАНРА ADVENTURE.

|                        |                 |
|------------------------|-----------------|
| AFTERSHOCK             | Interceptor     |
| ADVENTURE 1            | Adventure       |
| ADVENTURE "A"          |                 |
| ("PLANET OF DEATH")    | Artic           |
| THE BOGGIT             | CRL             |
| BORED OF THE RINGS     | Delta 4         |
| THE CHALLENGE          | Temptation      |
| DOOMDARK'S REVENGE     | Beyond          |
| DAVY JONES LOCKER      | J. Lockerby     |
| DRAKKHEM & THE CHAMBER |                 |
| EMERALD ISLE           | Level 9         |
| ESPIONAGE ISLAND       |                 |
| (ADVENTURE "D")        | Artic           |
| INCA CURSE             |                 |
| (ADVENTURE "B")        | Artic           |
| ICE PALACE             |                 |
| INHERITANCE            | Infogrames      |
| INSPECTOR FLUKEIT      |                 |
| GHOULIES               | IMS             |
| GNOME RANGER           |                 |
| GREMLINS               | Adventure       |
| HEROES OF KARN         | Intrceptor      |
| THE HELM               | Firebird        |
| HIJACK                 | Electric Dreams |
| HOBBIT                 | Melbourne House |
| HEAVY ON THE MAGIC     | Gargoyle Games  |
| HIMAN                  | Adventure       |
| HUNCHBACK-1,2,3        | Ocean           |
| HAMPSTEAD              | Melbourne House |
| HEARTLAND              | Odin            |

|                                             |                    |
|---------------------------------------------|--------------------|
| JEWELS OF BABYLON                           | Interceptor        |
| KARUSSIA                                    | Incentive          |
| KAYLETH                                     | Adventure          |
| KWAH!                                       |                    |
| KING'S QUEST                                | Sierra-on-Line     |
| KOBYACHI NARU                               | Mastertronic       |
| LOCK SHER 2                                 |                    |
| LITTLE COMPUTER PEOPLE                      | Activision         |
| LAMBERY MYSTERY                             |                    |
| LORD OF THE RINGS 1,2                       | Melbourne House    |
| MASTER OF MAGIC                             | Mastertronic       |
| MINDSHADOW                                  | Activision         |
| MYSTERY OF ARKHEM MANOR                     | London Chronicle   |
| NEVERENDING STORY 1,2,3                     | Ocean              |
| PUSZKA PANDORY                              |                    |
| ROBIN OF SHERLOCK                           | Delta 4/Silversoft |
| RONNIE GOES TO HOLLYWOOD                    | 5-th Day           |
| REBEL PLANET                                | US Gold            |
| RED HAWK                                    | Melbourne House    |
| RED MOON                                    | Level 9            |
| SHERLOCK                                    | Melbourne House    |
| SLANE KING                                  |                    |
| SHIP OF DOOM                                |                    |
| ADVENTURE "C"                               | Artic              |
| SHADOWS OF MORDOR (см. LORD OF THE RINGS 2) |                    |
| SILICON DREAMS                              | Level 9            |
| SNOWBALL                                    | Level 9            |
| SORDERON'S SHADOW                           | Beyond             |
| STORMBRINGER                                | Mastertronic       |
| SIDNEY AFFAIR                               | Infogrames         |
| SPELLBOUND                                  | Mastertronic       |
| SAM CRUISE                                  | Microsphere        |
| VERA CRUS AFFAIR 1,2                        | Infogrames         |
| WITCH'S CAULDRON                            | Micro-gen          |
| WULFAN                                      | Code Masters       |
| WORM IN PARADISE                            | Level 9            |

\* \* \*

Нам пишет Владимир Зареев.

Здравствуйтесь, пишет Вам Ваш читатель из г. Заволжье Нижегородской области. Я читаю Ваши описания и заметки по программе ELITE и хочу внести свой вклад для начинающих игроков. Дело в том, что во всех описаниях, которые я прочел, стыковка с базой описывается, как сложное мероприятие и таким способом трудно стыковаться даже опытному пилоту. Мой же способ позволил многим моим друзьям преуспеть в космическом бизнесе.

Однажды, вылетев из базы и решив покончить жизнь самоубийством, я нацелился пониже входного отверстия и на полной скорости ... влетел в базу. Меня это очень удивило, но попробовав еще несколько раз, я убедился, что это не случайность. Этот способ я проверял на двух версиях программы: с пометкой M128 и в версии Jack O'Lantern. При этом очень удобно пользоваться бестопливным перелетом с Кориолиса на Кориолис, описанным в ZX-РЕВЮ за 1991 год.

Прошу напечатать эту заметку, т. к. она поможет многим.

ИФК: Печатаем с удовольствием.

От себя хотим добавить, что бог сделал все нужное простым, а все сложное ненужным.

Да и вообще, как показывает жизнь, очень часто интересные находки бывают неожиданными. У нас тут тоже на днях был интересный эпизод. Заходим как-то в одной адвентюре в одну комнату, а там нападает какое-то чудовище, то ли воробей, то ли собака, но челюсти, кажется составляют большую часть туловища. Как назло, ружье забыли взять из сундука и как от этого монстра ни отбивались, ничто не помогает. В полном отчаянии хватаем со стола вазу и в него. Монстра, конечно, не убили, зато ваза раскололась и из нее выпал ... ключ, которого ох как не хватало. Впрочем, это отступление к делу отношения не имеет и служит скорее для поддержания духа экспериментаторства. Так что пойдем дальше.

KOPP: А теперь несколько слов об игре ATF, описанной С. А. Фокиным в "ZX-РЕВЮ-92" (Выпуск 12, с.258). Там сказано, что не надо слишком доверять автопилоту, т.к. самолет может не успевать отслеживать рельеф местности и задевает за вершины холмов.

Все дело в том, что надо убирать шасси. Это делается клавишей "V". Тогда самолет и скорость разовьет больше и за холмы задевать не будет. Кстати, режим автопилота у меня включается клавишей "T", а не "U". При посадке же самолет тормозит автоматически.

\* \* \*

Нам пишет Дворецков А. И. из г. Кургана.

Привет, Инфорком!

Я отправил вам денежный перевод около Нового года, в надежде получить "ZX-РЕВЮ" за 91 и 92 год. Отправил, и потерял все надежды получить ответ. Но каково же было мое удивление, когда я, наконец, получил его (ZX-РЕВЮ) через 4 месяца! Как это понимать?...

ИФК: К сожалению, сейчас такие задержки происходят все чаще и чаще. Дело в том, что типографии, способные оперативно нас обслужить как правило маломощные и потому наши тиражи невелики (2 - 3 тыс. экземпляров), а при работе с более мощными типографиями срок от подачи заявки и материала до выхода книжки растягивается на полгода (у них стоит очередь и госзаказы они пропускают вперед, отодвигая нас все дальше и дальше). Так и в Вашем случае, в январе несколько оптовых покупателей оставили нас без сборника ZX-РЕВЮ-91. Мы ожидали его очередную партию и спокойно отдали. А типография получила госзаказ на печать бандеролей для упаковки банковских купюр и два месяца печатала только их (интересно, сколько же купюр нас ожидает в ближайшие месяцы? Примерно так это и бывает, конечно же Вам от этого не легче и мы просим прощения как у Вас так и у других читателей за подобные задержки. Надеемся, что Вы нас простите и поймете. К сожалению. Вы читаете быстрее, чем мы пишем и печатаем.

KOPP: ... Но когда я стал читать, все мои опасения "как ветром сдуло". Да я просто не предполагал увидеть что-либо подобное. Это превзошло все мои ожидания! Я думаю, что все синклеристы со мной согласятся. Ну, а теперь немного критики.

1. Считаю ненужным давать описания различных программ: BETA BASIC, MEGA BASIC. Лучше выпустить их отдельной книгой. То же самое и "Секреты ПЗУ".

2. Статья Михайленко В.С. "Защита программ" - полезная вещь, но много пустых слов. Например, много было написано о структуре заголовка (я называю его "шапкой") и о том, как лишить программу автостарта. Все гораздо проще. Загрузите Бейсик-загрузчик в СОРУ-86М и нажмите клавишу R. Выгрузив на ленту Бейсик-загрузчик, Вы лишите его автостарта. Вообще всем хакерам я бы посоветовал брошюру Н. Родионова "Адаптация программ к системе TR-DOS. Издатель: фирма "Питер" 1992.

3. Дж. Хардман, Э. Хьюзон "40 лучших процедур" - я в восторге!

4. "Профессиональный подход" Стива Тернера - выше всяких похвал.

5. Слово экспертам - аналогично пункту 4 - так держать!

6. "Форум". Ну а этому разделу я отдаю наибольшее предпочтение.

На этом вся критика закончилась.

Могу предоставить вам экспертную проработку программы "LEGION OF DEATH" (Бесплатно, а в награду я хотел бы получить достоверную информацию: где мне купить

дисковод и контроллер, возможно вместе со Спектрумом по доступной цене. Помогите!)

ИФК: Спасибо огромное! Нас очень интересует эта проработка. Наверное, с этим согласятся и многие наши читатели. Поскольку мы сами аппаратными вопросами не занимаемся и аппаратурой не торгуем, давайте попросим всех, кто знает, как это сделать, написать Вам. Наверное, и в самом Кургане Вы найдете надежных консультантов. Кроме того, очень хорошо поставлено это дело и в регионах, соседних с Вашим - в Челябинской, Свердловской, Оренбургской областях. Даем Ваш адрес:

640004, г. Курган ул. Омская, д.4, кв.3, Дворецкову А. И.

KOPP:... Ну вот и все! с уважением, Денис Дворецков 16 лет.

ИФК: До свидания. Денис. Ждем с нетерпением "Легионы смерти", а пока поспешите заказать самый большой почтовый ящик, какой только бывает. Стандартный может не выдержать, но вы, надеемся, будете этому рады.

\* \* \*

Нам пишет Камразе Тимофей Михайлович, г. Москва:

Здравствуйте уважаемые составители рубрики "Форум".

Я увлекаюсь составлением карт к играм и хочу поблагодарить Вас за вашу отличную работу: "Форум" во многом облегчил мне жизнь в составлении плана к "HEAVY ON THE MAGIC". Если вам интересно, то могу предложить карты "TIR-NA-NOG", "JACK THE NIPPER 2", "ALIENS"...

ИФК: Огромное спасибо! У Вас очень интересное хобби. Мы с радостью примем карты, хотя здесь есть проблема. Дело в том, что подготовка к печати в текстовом редакторе художественно выполненной карты крайне трудоемкий процесс, да и вообще не всегда возможна, а у нас нет постоянных кадров, которым можно было бы это дело поручить. Просить Вас, чтобы Вы сами сделали карты в виде, пригодном для размножения, разумеется было бы бестактно. Вашу карту к игре ALIENS мы с благодарностью принимаем и даем для наших читателей, хотя она, конечно многое потеряла при наборе в текстовом редакторе (см. на следующей странице).

KOPP:... Сейчас я работаю над программой "DUN-DARACH". В связи с этим я был бы вам очень признателен за освещение этой и других программ фирмы "GARGOYLE GAMES".

ИФК: Шесть лет назад, в мае 87-го работая с журналом "Sinclair User" в библиотеке ГПНТБ, что на Кузнецком Мосту, мы видели огромную статью, посвященную этой программе, с картой в разворот журнала. Ах, если бы тогда знать, что она может пригодиться... Карта представляла из себя план средневекового ирландского города, весьма большого и запутанного. Текст статьи был насыщен терминологией из области древних ирландских, кельтских и шотландских легенд и, надо сказать, был довольно труден для чтения на понимание при нашем тогда еще не очень свободном знании английского языка. Ориентировочно этот журнал относился к 84 году и уже тогда был зачитан почти до дыр. К сожалению, сейчас мало шансов разыскать его там.

KOPP:... Теперь к вопросу об "ELITE". Этой зимой на рынке появилась программа "ELITE-2" (распространитель "S-STUDIO" хорошо известен "тушинским" синклеристам). В чем разница между ELITE и ELITE-2 установить не удалось. Отличие лишь в том, что невозможен гиперпрыжок сквозь станцию. "ELITE-2" очень напоминает "версию 3" (см. ZX-РЕВЮ N3,4 1992).

ИФК: В чем разница? Пожалуй, на этот вопрос лучше всего ответят наши читатели.

KOPP:... И еще одно. Есть идея: хотелось бы переделать ELITE под сеть для игры вдвоем. Возможность летать с другом, который может тебя прикрыть или вести бой с живым противником, сделает игру неподражаемой.

К сожалению, у меня нет достаточных знаний. Поэтому я дарю свою идею тому, кто сможет ее воплотить в жизнь.

ИФК: К сожалению, здесь вряд ли что получится. Скажутся и недостаточное быстродействие компьютера и малый объем памяти и отсутствие аппаратного и программного обеспечения сетевого варианта. Проще, по-видимому, написать новую



программу, которая и будет заниматься исключительно космическими боями, а вопросы торговли и т.п. обойдет стороной.

Кстати, это как раз тот случай когда целесообразно использовать IBM-совместимую машину. Последние два года боевые имитаторы для этих машин выпускаются с возможностью работы через модем с живым партнером (противником), а в зарубежных журналах появились объявления типа "Ищу партнера для игры в .... и прилагается список игр".

\* \* \*

Нам пишет Неведомский Александр Александрович, г. Томск:

Здравствуйте! Давно хотел Вам написать, да все не получалось... Прошлым летом я прошел программу ALIENS по фильму ("Чужие"), но до сих пор ни у кого не видел и нигде не слышал о том, чтобы кто-нибудь составил для нее карту. Без карты эту игру можно пройти только либо будучи бессмертным, либо если помнить весь лабиринт. Посылаю вам составленную мной карту. А это долгий и кропотливый труд....

ИФК: Большое спасибо, Александр Александрович, это действительно труд, заслуживающий большого уважения и надеемся, что наши читатели его оценят. Конечно же, Ваш план и план Тимофея Михайловича совпадают, ведь так и должно быть, поэтому мы здесь приводим один план, принадлежащий двум авторам. От себя же добавим, что присланный Вами оригинал выглядит гораздо лучше, чем то, что мы из него сделали (с учетом высказанного выше соображения о сложности наборе плана в текстовом редакторе).

КОРР:... Здесь же привожу полезные советы для игроков, составленные по собственному опыту, хотя некоторые из них встречаются в других описаниях игры.

1. Для того, чтобы пройти игру, необходимо привести в комнату "матки" (а их в отличие от фильма - несколько) не менее трех человек, иначе не хватит оружия (боеприпасов). Да и то только в том случае, если Вы будете метко стрелять.

2. Выбирайте для этой цели тех героев, управление которыми наиболее удобно (чтобы можно было следить за его уровнем энергии, номером комнаты и, конечно, экраном). Одного героя обязательно оставьте в комнате N 1. Там его никто не убьет, а Вы вспомните мой совет добрым словом в конце игры.

3. Не рекомендую стрелять в "чужих" до комнаты 78, в которую надо довести максимальное число героев.

4. Кокон имеет смысл "гасить" только в комнате 248.

5. Всегда целюсь в голову. У всех "чужих" голова находится на уровне дверного замка.

6. Нет смысла "запечатывать" двери. Вы не пройдете, а "чужие" все равно пройдут.

7. Не убейте по неосторожности ребенка. Это девочка. Ее зовут Нют и ее надо спасти. За ее обнаружение получите дополнительные 1000 очков.

8. Не пытайтесь открыть двери с пробитым замком. Не хватит здоровья.

9: Не устраивайте боев в помещении генераторной.

10. Не убивайте "чужого" перед дверью. Вы только опалите его шкуру, а он будет ожидать Вас за дверью до тех пор, пока кто-то не убьет его с другой стороны (лужа перед дверью).

11. Давайте отдохнуть своим героям, иначе они потеряют подвижность.

12. Если кого-то из героев захватили и посадили в кокон, его еще можно попытаться спасти. Некоторое время он остается живым. Спасти его можно уничтожив в комнате всех инсекторептилий и разбив все коконы.

13. Боезапас пополняется только в оружейной.

14. Не распыляйте свой отряд по разным помещениям. Так труднее сохранить людей живыми.

ИФК: От имени наших читателей благодарим за полезные советы. От себя хочется добавить, что наилучшим советом было бы если и не посмотреть кинофильм (ведь не у всех есть видеомэгнитофоны), то хотя бы прочитать книгу А. Д. Фостер "Чужой. Чужие. Чужой-3." В одном томе собраны все три научно-фантастических романа.

КОРР: Я обращаюсь к Вам с просьбой. Есть хорошая программа TOTAL ECLIPSE 2 (1991, Incentive). Я ее почти прошел, но не знаю, как пройти два места:

- место, где дверь быстро исчезает и появляется (причем над ней черный квадрат);
- место, в котором требуется слишком много ключей (порядка 4), а их нет (в наличии только 2).

Это очень интересная игра, в ней много сюрпризов и ни один не повторяется. Перечислю их: прозрачные стены, порог, забирающий воду, двойные стены, потайной ход в камне, тайная плита. Может быть есть что-то еще. Кстати, кто знает, для чего нужен орел, который поворачивает голову при выстреле в него.

ИФК: Спасибо за мнение. Наверное те, кто думают над пополнением своей игротеки к нему прислушаются. А может быть кто-нибудь и поможет с решением описанных проблем?

КОРР: А недавно я прошел программу DEFENDER OF CROWN. Это как бы разминочная программа для тех, кто любит стратегические игры. У нее великолепная музыка, да и исполнена она великолепно.

МФК: И опять же спасибо за мнение. Любые отзывы о программах важны нашим читателям.

\* \* \*

Нам пишет Петухов Владимир из г. Златоуст Челябинской обл:

Здравствуй, уважаемый "Инфорком"!

Очень рад, что Вы возобновили подписку по почте. Журнал очень нужный. Трое моих коллег после ознакомления с ним будут его выписывать. Мою подписку за 1991 год зачитали до дыр. Придется новую выписывать.

Дай вам Бог здоровья и благополучия каждому и процветания журналу и фирме на долгие годы!

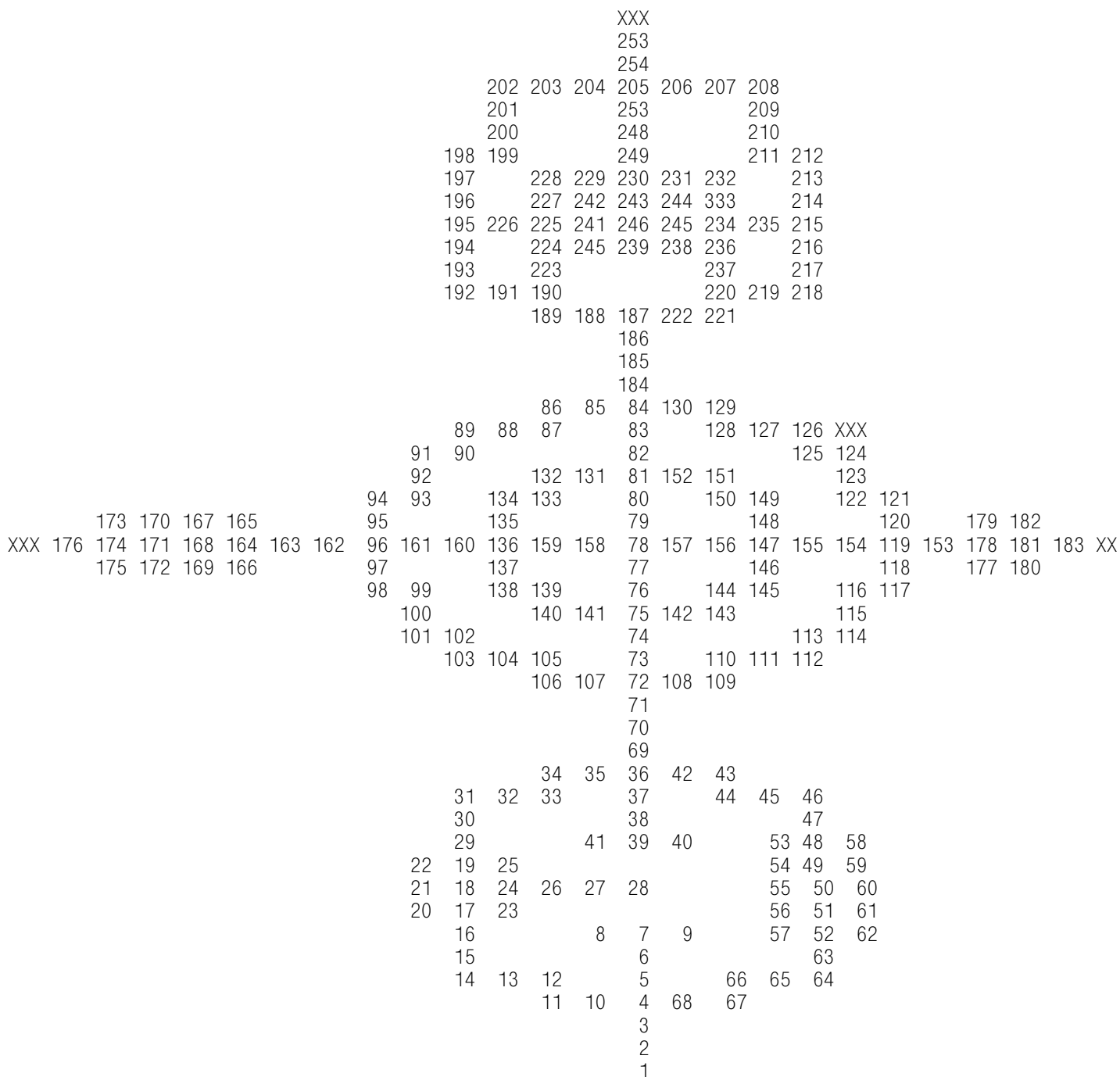
В качестве темы в журнале хотелось бы видеть почаще материалы, посвященные дисководам, дисковой операционной системе TR-DOS, управлению дисководом из машинных кодов. Хорошо бы увидеть материал в конкретных примерах, например как это сделано в книге "Методы программирования на Бейсике" И. Кергаль. Перевод А.И. Брядинского под редакцией Ю. И. Банковского. Издательство "Мир" Москва 1991 год глава 7.

У Вас, я уверен, это получится хорошо. В целом, занимаясь компьютером с 1990 года, я могу с уверенностью сказать, что хотя и медленно, но уверенно общество переходит на компьютеры с дисковыми, и если "деды" начинали с магнитофоном, то "новички" уже рвутся к компьютеру сразу с дисководом, невзирая на материальные затраты и отсутствие опыта и знаний. Вот этот момент, на мой взгляд, важно не пропустить "Инфоркому".

В заключение своего письма еще раз желаю вам всего доброго.

ИФК: Спасибо, Владимир, за теплые слова. Будем стараться по-возможности и при наличии материала развивать высказанные Вами пожелания, включаем все каналы в режим "On Line". Теперь все зависит от наших читателей.

**КАРТА ПРОГРАММЫ ALIENS**  
**(С) Камразе Тимофей Михайлович, 1993**  
**(С) Неведомский Александр Александрович, 1993**



1 - База      78 - Комната управления      181 - столовая      XXX - "смертельно"  
 28 - Оружейная      174 - Генераторная      248 - Комната "матки"

**НАИЛУЧШИЙ МАРШРУТ**

1-2-3-4-10-11-12 13-14-15-16-17-20- 21 - 22 - 19 - 29 - 30 - 31 32 - 33 - 34 - 35 - 36 - 69  
 - 70 - 71 - 72 - 107 - 106 - 105 104 - 103 - 102 - 101 - 100 - 99 98 - 97 - 96 - 161 - 160 - 136 159 -  
 158 - 78 - 79 - 80 - 81 - 82 83 - 81 - 184 - 185 - 186 - 187 188 - 189 - 190 - 227 - 224 245 - 239 -  
 246 - 241 - 242 227 228 - 229 - 230 - 249 - 248

**"Вечная жизнь":**

POKE 31914,0  
 POKE 30738,0  
 POKE 34484,195

Нам пишет Кохан Валерий Витович, Южно-Сахалинск.

Уважаемая редакция "ZX-РЕВЮ"! У меня огромное желание иметь Ваш журнал в личной библиотеке. Я имею "ZX-SPECTRUM" дома и моя работа напрямую связана с этой замечательной машиной. Сам я работаю заведующим лабораторией ЭВМ Сахалинского областного Дома техники и уже 3 года веду курсы (кружки) "Работа на ZX SPECTRUM и программирование на языке Бейсик". Желающих заниматься много, ведь у многих дома есть эти компьютеры. "SPECTRUM" уверенно завоевывает симпатии сахалинцев.

Я написал много программ по курсу информатики для компьютерных классов "ZX-Spectrum" (я работаю на классе Протвинского варианта - Калужская область, Жуковский р-н). Это программы-демонстраторы, тренажеры, экзаменаторы с выставлением оценки и подсчетом затраченного времени на опрос по следующим темам:

- опрос по любому предмету (укажи номер правильного ответа)
- программа-диктант (найди и исправь ошибки в тексте);
- переводы чисел в системах счисления (2, 10, 16);
- логические элементы И, НЕ, ИЛИ;
- составление арифметических выражений в Бейсике;
- работа простейшего процессора (решение арифметических выражений);
- текстовый режим экрана (рисование картинки оператором PRINT)
- графический режим экрана (рисование картинки по координатам);
- наложение символов и создание символов UDG;
- системные переменные;
- программы - демонстраторы применения тех или иных операторов Бейсика

(программа-будильник, секундомер, жеребьевка, бюджет семьи, база данных "фамилии, адреса, телефоны", тренажер и экзаменатор знаний арифметики и т. д.

Программы исключительно делового применения при ведении занятий по информатике или курсов программирования на стандартном Бейсике для классов ZX SPECTRUM. Написаны только на стандартном Бейсике, без модулей музыкальных и графических заставок, поэтому проблем с работой этих программ при пользовании дисководом и магнитофоном нет. При ведении занятий я только объясняю теоретическую часть материала, а всю демонстрационную, тренажерную и опросную работу выполняет сама ЭВМ. Я на это время становлюсь консультантом.

Сейчас я готовлю программы к предложению на рынок. Можно ли иметь сотрудничество в этом плане с вашим журналом? Буду Вам благодарен, если дадите любой ответ, я человек влюбленный в Спектр и считаю, что информатику в школах, ПТУ лучше всего давать на этом компьютере. Мне хотелось бы, чтобы мои программы помогали преподавателям, работающим с компьютерными классами ZX SPECTRUM.

МК: Уважаемый Валерий Витович! С радостью поможем Вам. Для нас, поверьте, это большая честь, ведь основная наша цель - именно образование.

КОРР:... литературу по ZX достать сложно, особенно вам, живущим на краю России.

ИФК: Да, мы об этом знаем и всегда это учитываем. Если у нас случаются проколы с нехваткой каких-либо материалов, мы стараемся в первую очередь обслужить жителей именно отдаленных регионов. Кстати, основная масса наших постоянных клиентов проживает именно там. Все-таки жителям центральных областей проще добывать ваши материалы, которые нещадно копируются и перепечатываются пиратским способом многими недобросовестными "фирмами".

ВСЕ, КТО ЗАИНТЕРЕСОВАН В ПРИМЕНЕНИИ "ZX-SPECTRUM" в учебном процессе, пишите по адресу:

693008, Россия, Сахалинская область, г. Южно-Сахалинск, пр-т Победы. 75, кв. 19.  
Кохан Валерию Витовичу.

Нам пишет Епифанов Андрей Валерьевич из Самары:

Привет, Инфорком!

Читаю я "ZX-РЕВЮ" и нравится мне там рубрика "Слово экспертам": в ней я много узнал о программах, которые у меня были, но я не знал, как и что в них делать.

И вот решил я поделиться с вами, а заодно и со всеми читателями "ZX-РЕВЮ" о похождениях маленького дракончика в программе "LITTLE PUFF", выпущенной фирмой CODE MASTER в 1989 г.

Это программа типичного аркадно-адвентюрного жанра. В этой игре Вы управляете маленьким заблудившимся дракончиком, которому надо вернуться домой. Для этого ему надо собрать четыре части паспорта (пропуска) и золотую монетку. Если Вы захотите составить карту, то лучше сразу бросьте, т.к. уровни (а их три - один под другим) смещены относительно друг друга довольно-таки жутко. В игре приятная музыка и графика, смещение плавное, а цвета не утомляют. В лабиринте есть ловушки, в которые лучше не попадать.

Как играть в LITTLE PUFF? В лабиринте разбросаны разные вещи, которые можно поделить на два типа, которые можно использовать и которые не нужны (так я, например, так и не нашел применения пистолету), а также всякие фрукты, за которые Вы получаете очки. Запас вещей, которые Вы можете нести одновременно, - ограничен, поэтому иногда Вам придется проходить в два приема.

Несколько советов:

1. Чтобы сжечь дерево, нужна бутылка бензина.
2. Чтобы проплыть (?) по воде, нужна помпа и спасательный круг.
3. Чтобы включить лифт (?) надо молотком рубить коробку и открыть дверь выпавшим ключом.
4. Если открыть бутылку штопором, то у Вас будет еще одна часть пропуска.
5. Чтобы уберечься от кокоса, надо взять третью часть пропуска в (?).
6. Если положить кокос на кнопку, то где-то можно отключить защиту и взять четвертую (последнюю) часть пропуска.
7. Для того, чтобы получить золотую (а не дырявую) монетку, надо распилить кокос.
8. Подойдя к мосту, встаньте на него и подождите, а потом проходите его прыжками.

ИФК: Мы благодарим Вас, уважаемый корреспондент за заботу о наших читателях. Наверное, многих заинтересует эта программа и они скажут Вам спасибо. К сожалению, в некоторых местах почерк Вашего письма был не вполне разборчив и там мы поставили в скобках вопросительный знак. Впрочем, вряд ли это кому-то помешает сыграть в хорошую игру.

КОРР:... Как видите, это не самая трудная и сложная программа, но Вам постоянно будут мешать разные обитатели, а жизнью у Вас всего одна.

Еще я хотел задать Вам один вопрос и высказать одну просьбу:

Вопрос: не знаете ли Вы или кто-нибудь из читателей, что и как делать в программе SWORD AND SORCERY? И какой пароль в игре CPT.TRUENO II?

Просьба: в №1 за 1991 год я прочитал ваше объявление о рассылке вами описания дизайнера программ HURG. Не могли бы Вы прислать его мне? Конверт прилагаю.

ИФК: Что касается геймдизайнера HURG, мы уже очень давно закончили его распространение, но для Вас экземпляр найдется, высылаем!

По доводу TRUENO 2 давайте вместе скажем спасибо Алексею Березину.

А вот что касается S&S, то мы с этой программой не знакомы и лучше предоставим слово читателям, давайте наберемся терпения и наверное кто-то, кто прошел эту программу, поделится своим опытом.

Нам пишет Алексеев Николай Алексеевич из г. Инта, республика Коми:

Здравствуйте, уважаемая редакция!

Я читал 3 и 4 номера РЕВЮ-92 и решил написать Вам об одной из версий "ELITE", вскрытой Родионовым.

Я не согласен с тем, что версия, где появляется надпись "M1 LOADING" похожа на "M128". Внешне "M1" и версия Родионова похожи как две капли воды, различие только в том, что изменен звук лазера. В обеих версиях при попытке атаковать станцию никаких надписей нет, просто не обращают внимания, только невозможно потом на станцию залететь. Полицейские из станции не вылетают. Ракетой станцию не уничтожить, даже если иметь Е.С.М. JAMMER, исключение составляет только база таргонов ISREMA. При попадании в Ваш корабль защита выбивается наполовину, так что если за Вами привязались штуки 3 ракеты, то можно и погибнуть. Конечно, с противоракетной системой проблем нет. Ваши ракеты действуют здесь неплохо, ими лучше уничтожить корабли ADDER, COBRA, PYTHON - это самые уязвимые корабли.

Пиратов в этой игре много, полиция не очень серьезная. Если пираты могут нападать скопом, то полиция всегда атакует по-одиночке. Все типы кораблей могут быть пиратами, за исключением полицейских. В игре нет корабля типа KRAIT. Пираты обычно появляются вне видимости. Встречаются также станции (РЕВЮ-92, N1,2). Их легко искать, надо только полететь на феодальную или анархическую планету с низким уровнем. Маневрируя, они выпускают корабли типа SIDEWINDER и ADDER от 1 до 3 шт. Очень сложно выполнить вторую миссию, когда пираты появляются до 10 сразу. С таргонами лучше всего сражаться с технологическим лазером, 15-ть попаданий выбивают их из строя.

В обеих версиях можно попасть в 47 галактику, у Вас будет много денег, отличное вооружение. Но после вылета попадете в 1 галактику, где нет планеты LAVE. Так что фактически это не 1 галактика, а 41. Где-то в 44 и 45 (4 и 5) точно не помню, есть планета LAVE! Только уровень у нее 10 - демократический. Так что многие планеты могут повторяться. Цены на товар здесь стабильны. Чем выше уровень, тем меньше цены, чем ниже уровень, тем выше цены. Разница изменений цен, к примеру, на низких уровнях, незначительна. Если Вы случайно убьете мирного торговца, у Вас изменится только статус. Вас пустят на станцию даже если Вы убийца. Единственное условие, не стрелять по тем, кто по Вам не стреляет, когда загорается знак станции S. На статус можно не обращать внимания, но тем, кто не хочет портить статус и быстро передвигается к планете, дам совет. На полной скорости врежьте в корабль, статус останется прежним, а энергию быстро восстановите.

Предлагаю небольшой жучок для тех, кому трудно выполнить вторую миссию. Когда корабль попадет в зону видимости (можно узнать по локатору), надо быстро переключиться на задний, а потом на передний экран, и секунд на 10 он будет виден.

И еще для тех, кто пользуется спасательной капсулой. Когда капсула отлетит от корабля, стреляйте в нее, дайте газу, подберите контейнер, и окажетесь на станции. Только делать нужно быстро.

В обеих версиях можно быстро погибнуть, если не предпринимать мер, особенно новичкам. Так что версии достаточно сложны.

Еще я делал эксперимент. В отгрузочном блоке поставил в байтах 61-66 нули, и ни одной планеты не было. Потом поставил во всех байтах 255. Все галактики были похожи как две капли воды. Сам я считаю, что галактик может быть до миллиона, хотя планеты могут повторяться. Я, как и все, ищу планету RAXXLA, но пока не нашел. Хотя у меня есть версия, что ее нет! Доказательство: Д. П. Шилин открыл текстовый блок. Вот коды 148 RA. 149 - LA, а кода где были бы XX нет! Если же его найдут, то тогда действительно можно найти планету с таким названием.

ИФК: Спасибо, Николай Алексеевич за исследование. Оно существенно пополняет нашу картину знаний об этой уникальной программе. Что же касается поисков планеты

RAXXLA, то с Вашей логикой нельзя не согласиться. Все верно, кода XX в программе нет. Жаль расставаться с прекрасной мечтой, но ведь можно предположить, что для такой избранной планеты авторы могли заложить иной способ кодировки названия.

\* \* \*

Нам пишет Трофимчук Владимир Викторович из Владивостока:

Здравствуйте!

Пишу Вам впервые, поэтому хочу поблагодарить за самое интересное, полезное и уникальное издание - "ZX-РЕВЮ". Я хоть и не подписчик, но имею все номера "РЕВЮ" и мне все в них нравятся. Особое предпочтение я отдаю статьям об адвентюрных играх, но жаль, что в этом году Вы отошли от этой темы.

ИФК: Вы правы, Владимир Викторович, нам и самим очень жаль, что мы отошли от темы адвентюрных игр, ведь это наш любимый жанр. Дело в том, что сейчас уже не остается времени на то, чтобы от души погонять игровые программы, мы очень загружены работой над "РЕВЮ", новыми книгами, обучающими программами, деловыми системами и пр. К сожалению, мы никак не можем выйти из ритма работы по 12-14 часов в день без выходных в течение уже трех лет.

Тем не менее, мы с радостью вернемся к этому жанру, если будут интересные письма от читателей, к тому же Вы уже знаете из этого номера, что сейчас ведется работа над сериалом SPELLBOUND - KNIGHT Tyme - STORMBRINGER.

Если же Вы имеете какой-то доступ к IBM-овской технике, то можем порекомендовать Вам нашу работу "Адвентюрные игры: эволюция жанра", которую мы опубликовали в журнале "МОНИТОР" (NN 8,9 за 1992 год).

Впрочем, и этим наши усилия не кончаются. Вот уже более полугода мы координируем и финансируем работу крупного авторского коллектива над огромной книгой, посвященной игровым программам. Это будет красочный том в подарочной исполнении весом килограмма в полтора. Раздел, посвященный адвентюрам занимает там не последнее место. Проект пока секретен и название книги мы не сообщаем, но в конце 1993 - начале 1994 г. мы очень широко оповестим об этом всю страну. Так что мы еще не совсем потеряны для любителей адвентюр (и не только адвентюр).

KOPP: ... немного о себе: 23 года, имею "Ленинградец" около года, занимаюсь в основном с адвентюрами.

Почему решил написать. Как-то в №6 (1991) в Вашем издании был помещен список адвентюр на двух кассетах и просьба, если кто-то пройдет эти игры от начала до конца - черкнуть Вам.

Недавно у меня появились эти сборники, но все по-порядку.

Приобретя Ваш трехтомник по программированию в машинных кодах, я стал немного разбираться в машинном коде фирменных программ, и попробовал для начала русифицировать MASTERFILE 09; конечно, я это делал не так, как описано в одном из номеров 1992 года, а с помощью MONS4, что гораздо удобнее. У меня получилось, и я пошел дальше - перекинул на адвентюрные игры. Так, мной были русифицированы KENTILLA; MESSAGE FROM ANDROMEDA; PLANET OF DEATH; HUNCHBACK 1,2,3.

ИФК: Уважаемый Владимир Викторович! Ваши успехи - лучшая реклама нашему трехтомнику! Кстати, теперь это уже не трехтомник, а однотомник, ведь мы выпустили новое дополненное издание в одном томе. Оно недавно вышло из печати и доступно всем желающим;

KOPP: ...Эти игры я прошел полностью, но про KENTILLA могу сказать, что ее "фирменный" вариант содержит грубые ошибки (программа на Basic), из-за чего возможно пройти только 50% игры. В своем варианте я эти ошибки исправил.

"HUNCHBACK" искалечен фирмой "M1 Loading", из-за чего стал доступен любой этап, но с теми предметами, которые вставила фирма, а не нашел играющий. Это я тоже исправил, да к тому же сделал удобный ввод для пользователя.

Так вот, ближе к делу, я перевел такую программу, как GREMLINS .... но не могу пройти ее до конца, чтобы проверить правильность перевода текста. Эту игру очень трудно пройти, хотя команды, которые она понимает, я отыскал, а вот в какие моменты их нужно давать и с какими предметами - не знаю. Программа на все отвечает: "Вы еще не можете, сделать это". Помогите, пожалуйста, может Вам написал кто-нибудь, кто прошел GREMLINS или другие игры (не ANDROMEDA, KENTILLA, ADVENTURE A, HUNCHBACK) от начала до конца,

ИФК: Нет, пока никто не сообщал о победе над программами этих сборников.

KOPP: А мой опыт можете использовать для помощи другим. Мой адрес:

690087 Владивосток, Ул. Котельникова д. 7 кв. 188

Трофимчук Владимиру Викторовичу.

ИФК: Спасибо за дружескую руку. Раз Вы не возражаете в том, чтобы оказать помощь нашим читателям, мы даем Ваш адрес. Если будет трудно ответить всем по почте, воспользуйтесь нашими возможностями. Наиболее часто встречающиеся проблемы в адвентюрах можете освещать в "ZX-РЕВЮ". Мы будем рады опубликовать какую-либо статью под Вашим именем.

\* \* \*

Нам пишет Чертков Вадим Юрьевич из Кургана:

Здравствуй, Инфорком!

У нас в Кургане "Спектрумы" изготавливает кооператив "Дисплей" в течении нескольких лет. В основе их положена доработанная схема Балтики. Так что они почти полностью совместимы с оригиналом, за исключением порта #FF, он всегда выдает 255. Поэтому не идут программы: SHORT CIRCUIT 1,2; RENEGADE. С остальными никаких проблем нет. Идут и SAMANTHA FOX и TURBO COMP и ARCANOID и т.д. т.е. все, что не работало у ваших читателей у меня работает.

ИФК: Спасибо, Вадим Юрьевич за информацию, которая может пригодиться нашим читателям. Мы хотели бы попросить Вас показать последнюю страницу этого выпуска кому-нибудь из сотрудников "Дисплея", им это может быть полезным.

KOPP:... На моем "Спектруме" стоит прошивка SINCLAIR 1990. Вы о ней вскользь упоминали. У нас в Кургане очень у многих есть эта прошивка, однако на этой прошивке наблюдается некоторая несовместимость программ. У меня имеется около 1 тыс. игр, из них не так как надо работают несколько штук:

EXOLON - нет звука

CYBERNOID - нет звука

RACER - нет звука

STORM LORD - нет звука

Но это все ерунда по сравнению с тем, что можно сделать с этой прошивкой. Можно пожертвовать несколькими играми ради этого.

Одна из прелестей: при нажатии кнопки на линии NMI компьютер переходит в "монитор". Единственное ограничение для этого: надо, чтобы stack не был в экранной области. Иначе компьютер зависнет.

ИФК: Еще раз спасибо за объективную информацию.

KOPP: В 1991 году была объявлена атака на игру TRANTOR. Вот что я могу сказать по этому поводу:

Игру пройти очень сложно, поэтому я привожу POKE:

POKE 49505,0 - LIVES (ENERGY)

POKE 49620,0 - TIME OFF

Необходимо пройти по всем этажам, на каждом этаже по 1 компьютеру. Если подойти к компьютеру, он выдает букву и т.д. Надо посетить все 8 компьютеров. На пути могут встретиться также два больших компьютера, но о них чуть позже. Так вот, собрав все восемь



букв, из них составляем слово близкое по смыслу к вычислительной технике: SPECTRUM, CASSETTE, SINCLAIR, MEGAGAME, SOFTWARE, JOYSTICK, KEYBOARD, KEMPSTON, COMPUTER, GRAPHICS, HARDWARE, TERMINAL, PRINTERS, WARGAMES, WARRIORS.

Затем идете к NIK-SECURITY-TERMINAL. И вводите составленное вами слово. Если Вы собрали не все буквы, то компьютер выдаст сообщение: PASSWORD NOT COMPLETED. Для входа же в компьютер с собой надо иметь магнитную карточку и отвертку.

Если Вы собрали все буквы, то он напечатает PASSWORD ANALYSIS. Если слово составлено верно и на терминале все замигает и т.д., он выдаст код для компьютера, управляющего транспортером, состоящий из трех букв. Теперь надо бежать к TRANSPORTER TERMINAL DAVE 29565 и ввести в него этот код (из трех букв). Если Вы ввели код правильно, то по окончании ввода услышите короткий звук. Если же код неверен, то звука не будет.

Затем встаете на транспортер и нажимаете букву "B". Если Вы что-то сделаете не так, то при нажатии на "B" космонавт погибнет.

У меня игра TRANTOR версии BILL GILEERT. Привожу пример загрузчика с бессмертием.

```
10 BORDER 0: PAPER 0: INK 0: CLEAR 24999
20 FOR Y = 18432 TO 18499
30 READ A: POKE Y, A: NEXT Y
40 RANDOMIZE USR 18432
50 DATA 243,221,33,168,97,17,248,117,62,255,55,205,2,8,221,33,160,215,17,63,14,62,255,
55,205,2,8,49,255,87,33,0,91,229,229,221,33,0,54,17,168,6,62,255,55,205,2,8,205,
168,97,175,50,97,193,50,212,93,33,0,64,209,1,168,6,195,195,51
```

Это загрузчик всех блоков, кроме заставки (т.е. при загрузке надо пропустить загрузку заставки). Ставим магнитофон в том месте, где только что закончилась заставка и делаем RUN.

В заключение привожу еще несколько POKE, найденных самостоятельно.

|                  |   |             |
|------------------|---|-------------|
| MONTY ON THE RUN | - | 34714,0     |
| ZOMBIES          | - | 29553,0     |
|                  |   | 29565,0     |
| AIRWOLF 2        | - | 53433,201   |
|                  |   | или 53474,0 |
| BLOODY EYES      | - | 43393,0     |
| LA MAGA          | - | 40064,0     |
| CHRONOS          | - | 56909,0     |
| MISS PACMAN      | - | 52867,0     |
| BOSCONIAN        | - | 33356,0     |
| LEGEND OF CAGE   | - | 37054,0     |
|                  |   | 37055,0     |
| F. PATROL 2      | - | 40562, 0    |
| BOMB JACK        | - | 48984,0     |
| AQUARIUS         | - | 31054,0     |
| GAMEOVER         | - | 39337,0     |
| энергия          | - | 39273,201   |
|                  |   | 33482,124   |
| бомбы            | - | 33483,0     |
| ZZOOM            | - | 24741,0     |
|                  |   | 24742,0     |
| VOYAGE           | - | 54492,0     |
| PANAMA JOE       | - | 38629,0     |
|                  |   | 38630,0     |
| COOCKIE          | - | 28697,0     |
| VICTORY ROAD     | - | 39385,255   |
|                  |   | 39402,255   |
| FLY SHARK        | - | 42464,255   |
| ROCKMAN          | - | 56357,27    |
|                  |   | 56358,220   |
|                  |   | уровни:     |

|                     |                   |            |
|---------------------|-------------------|------------|
|                     | 'E' - ONYX        | 'M' - SAGE |
|                     | 'I' - GURU        | 'Q' - CLAW |
| SABOTAGE            | - 32979,62        |            |
|                     | 32980,6           |            |
|                     | 32981,50          |            |
|                     | 32982,132         |            |
|                     | 32983,169         |            |
| STRYKER             | - 63196,0         |            |
|                     | 57263,201         |            |
|                     | 59645,201         |            |
| GONZZALEZZ          | - 31498,0         |            |
|                     | 31501,0           |            |
| GONZZALEZZ2         | - 35749,0         |            |
| RIVER RAID          | - 28170,0 (fuel)  |            |
| VIXEN II            | - 51811,195       |            |
|                     | 51812,90          |            |
|                     | 51613,202         |            |
| IKARI               | - 41178,255       |            |
| TUJAD               | - 27212,183       |            |
| DEATH STAR          | - 38351,0 36682,0 |            |
| BAGDAD              | - 59857,0         |            |
| ARABIAN NIGHT       | - 57832,0         |            |
| BARBARIAN 3         | - 37132,0         |            |
| PYJAMARAMA          | - 48669,0         |            |
| DARK SIDE           | - 35166,0         |            |
|                     | 35167,0           |            |
|                     | 35168,0           |            |
| BLADE ALLEY         | - 35129,0         |            |
|                     | 35130,0           |            |
| FROST BYTE          | - 28236,0 (time)  |            |
|                     | 30991,0 lives     |            |
| CYBERNOID           | - DEF. KEYS: KG88 |            |
|                     | или POKE 37402,0  |            |
| CYBERNOID 2         | - DEF. KEYS: ORGY |            |
|                     | или POKE 34769,0  |            |
| OLI & LISSA         | - 35874,0         |            |
|                     | энергия 36062,0   |            |
| VHEELIE             | - 30034,201       |            |
|                     | или 30037,0:      |            |
|                     | 30126,4           |            |
|                     | RAND USR 27934    |            |
| KRAZY KONG          | - 24601,0         |            |
| SAMANTHA FOX        | - 24130,6         |            |
| AUFWIDERSEHEN MONTY | - 41128,0         |            |
|                     | 41129,0           |            |
| PHANTOMAS           | - 44819,0         |            |
|                     | 46149,201         |            |
| TRANTOR             | - 49505,0         |            |
|                     | 49620,0           |            |
| XEVIOUS             | - 54501,0         |            |
|                     | 54050,0           |            |
| MONTYMOLE           | - 38003,0         |            |
| PSSST               | - 24983,0         |            |
|                     | 25372,0           |            |
| QARX                | - 42488,0         |            |
| TOI ACID GAME:      |                   |            |
| FASE 1              | - 50297,0         |            |
| FASE 2 CODE 517     | 49808,0           |            |
| FASE 3 CODE 124     | 53438,0           |            |
| FASE 4 CODE 500     | 49485,0           |            |
| FAST FOOD           | - 47844,0         |            |
| CARTOON             | - 42448,0         |            |
| BOUNTY BOB          | - 43852,0         |            |
| EQUINOX             | - 45914,0         |            |
| THUNDERCATS         | - 31402,0         |            |

|                 |                  |
|-----------------|------------------|
| YETI            | - 47894,0        |
| CROSSWIZE       | - 32955,255      |
| CABAL           | - 30764,0        |
| MONTY FREE      | - 38301,0 (live) |
|                 | 34116,0 (time)   |
| SAVAGE          | - 37457,0        |
|                 | 51573,0          |
|                 | 52546,0          |
| SAVAGE 2        | - 32625,62       |
|                 | 32626,0          |
|                 | 32627,0          |
| RUFF & READY    | - 33567,0        |
| HOLLYWOOD POKER | - 38565,175      |
|                 | 36621,0          |
| GAMEOVER 2      | - 38691,0 (live) |
| (bomb)          | 33455,195        |
| (power)         | 32532,195        |
| TYPE-ROPE       | - 28955,0        |
|                 | 28956,0          |
| KRAKOUT         | - 36060,0        |
| XECUTOR         | - 54017,0        |
| DYNATRON        | - 41629,0        |
|                 | 45189,0          |
| F-16            | - 31564,0        |
| FIRE FLY        | - 45453,183      |
| VIXEN           | - 51789,0        |
| SOLOMON'S KEY   | DEFINE KEYS:     |
|                 | EBORP            |
|                 | или 49344,0      |
| SAVAGE 3        | - 34398,255      |
| ALIEN 8         | - 51736,0        |
| ENDURO RACER    | - 43650,0        |
|                 | 43651,0          |
| FREDDY HARD 3   | - 58978,0        |
| TREASURE ISLAND | - 61281,0        |
|                 | 59672,0          |
|                 | 60126,0          |
|                 | 61142,0          |

ИФК: Вадим Юрьевич, Вы проделали огромную работу, "расколов" такое множество программ. Огромное спасибо от имени тех читателей, которые любят вносить изменения в программы для удобства личного пользования. От себя только напомним всем читателям, что нельзя ни в коем случае распространять программы с жестко внесенными в них изменениями.

Это бестактно по отношению к другим пользователям. Никто не скажет за это спасибо. Кому же хочется докуривать чужие окурки.

\* \* \*

Нам пишет Сайфутдинов Евгений Владимирович из поселка Приполярный республики Коми:

Эй! Кого интересовал код 71 FRAME в игре "IMPACT"? Он очень прост: " J.D." В догонку еще 31 пароль к игре "EARTH SHAKER", 1990 MICHAEL BATTY. Игра схожа с сериалом BOULDER DASH, но сложнее и интереснее - 32 уровня:

- 1 - не требуется пароль.
- 2 - THE BUBBLE BATH.
- 3 - THE GRAVITY CHAMBER
- 4 - HEAD IN THE CLOUDS
- 5 - HUSH TOED ICELAND

6 - DIAMONDS OF MINE  
7 - THE TWO OF HEARTS  
8 - CENTRAL INCINERATOR  
9 - INTERNAL REACTOR INC  
10 - NUCLEAR ORE CAVERNS  
11 - GRAVITON SHELLS  
12 - THE OIL WELLS  
13 - WET HELL SOIL  
14 - THE GRAVLOCK CELL  
15 - HONEYCOMBS  
16 - HAIR BRAISED SURGERY  
17 - BURY DRAINAGE HIRERS  
18 - SPACEBSE VASTERPIPES  
19 - THE SHAKE  
20 - RUSE NO DOUBT  
21 - OUTER BOUNDS  
22 - SPRING LODED  
23 - ALARM CLOCK WORKS  
24 - FOG RISE WATERFALL  
25 - GREAT WALLS OF FIRE  
26 - CHINA CANAL  
27 - THE COMBINATION LOCK  
28 - YOUR SAFE WITH ME  
29 - YE FIT WARM HOUSE  
30 - THE KEEP  
31 - FOR GOODNESS SAKE  
32 - NO DEVIL LIVED ON

G - ABORT LEVEL

H - QUIT SCREEN

F - MAP

Коды добыты с помощью программы DOCTOR v. 4.3 (Родионов)

\* \* \*

Нам пишет Левашов Назар Михайлович из г. Киева:

Добрый день!

Недавно я исследовал программу "SPOOKED" и обратил внимание на мастерски сделанный загрузчик. Первое, что бросается в глаза - загрузка "задом наперед" - отличный метод защиты. Второе (мало кто пользовался встроенным копировщиком этой программы, поэтому это почти никому неизвестно), если после загрузки программы нажать одну из цифр от 1 до 5, то программа копируется на ленту. 1 - выгрузка в обычном режиме, 5 - выгрузка в турбо-режиме, причем пилоттон и бейсик-загрузчик выгружаются на стандартной скорости. Но самое интересное состоит в следующем: если нажать клавишу 2, 3 или 4, то программа выгружается на скорости на 1/4, 1/2, 3/4 больше стандартной соответственно (кроме пилоттонов и бейсик-загрузчика). Такой потрясающий метод защиты я нигде не видел. Нет такого копировщика, который скопировал бы что-нибудь на такой скорости.

Я не разбираюсь в ассемблере, и знаю только три команды: RET, CALL и LD, дисассемблера у меня нет, но я точно установил, что программа в ассемблере начинается с адреса 23291, а запускается с 23319. По-моему, программа переносит себя в конец памяти, т. к. я, просматривая машинный код, установил, что все CALL адресованы либо в ПЗУ, либо в

конец памяти.

Думаю, что все читатели были бы рады разбору того, как эта программа работает, а если об этом напечатать в "ZX-РЕВЮ", то это было бы прекрасным дополнением к книге Михайленко "Защита программ".

P. S. Такой же загрузчик я видел в программе "ISS". Обе эти игры составила фирма "THE E TEAM".

ИФК: Уважаемый Назар Михайлович! Мы уверены, что многих заинтересует Ваша информация, но мы верим и в Ваши силы. Ведь группа команд CALL, LD и RET включает в себя чуть ли не 50% того, что пишут на АССЕМБЛЕРЕ, а если добавить сюда еще команды RST, которые ничем от CALL не отличаются, да еще и команду NOP, то почти весь АССЕМБЛЕР Вам известен. Наверное Вам осталось только разобраться с флагами регистра F, да узнать о наиболее часто вызываемых программах ПЗУ, а об этом мы очень часто пишем в "ZX-РЕВЮ".

С глубоким уважением, "ИНФОРКОМ".

\* \* \*

Нам пишет семейный экипаж Троеглазовых П. и Г. из с. Гайтер Хабаровского края. (См. также раздел "ФОРУМ" в РЕВЮ-92 на с. 24).

Здравствуйте!

Приветствуем атаку на программу "ACADEMY". У нас есть чем поделиться, но так как мы еще мало с ней работали, мы хотели бы увидеть пути поисков более опытных пилотов.

И все-таки, мы вновь и вновь возвращаемся к "ELITE". Эта программа неисчерпаема. Мы уже и не считаем, в скольких галактиках побывали, Вы правы - им нет счета! Гера до сих пор не верит, что это так: "Па, это невозможно, - шесть в шестой степени, да еще каждая цифра может меняться от нуля до 255(!) - должен же быть какой-то предел!"

Вряд ли... Как бы мы ни меняли цифры в байтах 61-66, мы каждый раз попадали в новую восьмерку галактик, и, конечно же, в этом сонме галактик найдется планета RAXXLA. А может быть и не одна. Да... Получается, что всей жизни не хватит, чтобы просмотреть все галактики. Невероятно !

Браво, ELITE!!!

В дальнейшем, говоря о галактиках, вряд ли нужно присваивать им какие-то номера, - где тут первая, где тысячная, кто скажет? Те галактики, что мы видим после загрузки, вероятно, стоило бы назвать просто "первоначальный вариант".

Проще, говоря о каких-либо особенностях в любой из галактик, достаточно дать ее галактические байты (с 61 по 66). То, что собственные номера присваивать бесполезно, легко доказать. Берем группу цифр 1,2,3,4,5,6 и засылаем их в 61-66 байты. Можно ли в этой восьмерке галактик определить, где первая, а где восьмая? Ведь существует цикл. В данном случае это:

|   |    |    |    |    |     |     |     |  |
|---|----|----|----|----|-----|-----|-----|--|
| 1 | 2  | 4  | 8  | 16 | 32  | 64  | 128 |  |
| 2 | 4  | 6  | 16 | 32 | 64  | 128 | 1   |  |
| 3 | 6  | 12 | 24 | 48 | 96  | 192 | 129 |  |
| 4 | 8  | 16 | 32 | 64 | 128 | 1   | 2   |  |
| 5 | 10 | 20 | 40 | 80 | 160 | 65  | 130 |  |
| 6 | 12 | 24 | 48 | 96 | 192 | 129 | 3   |  |

Значит, мы можем набрать любую группу цифр и начать отсчет галактик (1-8) от первой появившейся на экране монитора. А возьмем первоначальный вариант, - 74,90, 72,8,83,183 - почему это именно ПЕРВАЯ галактика? Ни логики, ни смысла.

Счет можно дать восьмеркам галактик. Берем, скажем, 1,0,0,0,0,0, делаем цикл и исключаем их из общего числа галактик и так до конца. Сколько же получится таких восьмерок, кто решит эту задачу? Задача не из легких, здесь нужно просчитывать каждый цикл, исключать его из общего числа вариантов, помня о невозможности превышения (при удвоении) цифры 255. Если кто-либо из программистов сможет решить ее, то пусть пришлет

вариант программы. Интересно посмотреть распечатку этих восьмерок и узнать точно, сколько их в программе.

Открыв путь к самому интересному, мы вдруг начинаем искать "достойную замену". С этим нельзя согласиться, "ACADEMY" - сама по себе, "ELITE" - сама по себе. Эта программа будет еще не один год удивлять и восхищать всех нас!

Мы с Герой исследовали уже сотни галактик и сейчас расскажем о некоторых из них.

В галактике с "галактическими шифрами" - 7,77,8,88,9,99 мы натолкнулись на неизвестный нам корабль, назовем его PSEUDO-PYTHON. Шел он мирно и мы имели возможность спокойно осмотреть его со всех сторон, срисовать при помощи кнопки ПАУЗА и расстрелять его, тем самым проверив его боевые качества, - вел он себя в бою, как PYTHON, защита та же.

Мотаясь по галактикам, мы имели возможность побывать на двойных, тройных, четверных и даже пятерных, семерных звездах. Далее я буду давать в строчку через запятую значения байтов с 61 по 66 тех галактик, в которых есть что-либо необычное, красивое, загадочное, интересное. В восьмерке галактик 1,2,3,4,5,6 - 128,1,129,2,130,3 мы увидели множество двойных, тройных и четверных звезд и даже целые их группы.

В цикле из восьми галактик 7,77,8,88,9,99 там, где мы повстречали PSEUDO-PYTHON, в одной из галактик (193,83,2,22,66,216) можно увидеть семерную звезду.

Очень красивые галактики в цикле 128,48,16,32,64,254 - 64, 24,8,16,32,127.

Общее расположение звезд в галактике 128,48,16,32,64,254 такое - диагональное, интересны расположения звезд и в других галактиках этого же цикла.

Тройные, четверные и другие звездные системы встречаются сплошь и рядом и, думаю, о них уже достаточно сказано.

Но вот новая неожиданность: ФЕОДАЛЬНАЯ ГАЛАКТИКА(!) (72,72,72,72,72,72). Уровень развития не превышает в ней десяти, космос забит пиратами.

Запускаем "галактические" байты - 5,10,15,20,25,30 и обнаруживаем, что в первой из восьми галактик этого цикла только четыре типа правительств - Корпорация, Диктатура, Конфедерация и Феодалы. Во второй (10,20,30,40,50,60) власть делят Демократы и Диктатура и вся галактика довольно высокоразвита (средний уровень развития 9-10). В третьей (20,40,60,80,100,120) уже властвуют только Корпоративные и Конфедеративные интересы, эта галактика тоже очень высокоразвита. В четвертой снова всего лишь 2 типа власти - Корпорация и Диктатура, да и сама эта галактика очень красива, звезды собраны в скопления, похожие на звездные дороги (40,80,120,160,200,240). В шестой галактике (160,65,225,130,35,195) нет ни Демократии, ни Анархии, ни мультиправительств. В 7-й (65, 130,195,5,70,135) есть все виды правительств, на всю галактику лишь 2-3 планеты Анархистов и общий уровень развития довольно низок - очень редко попадаете 10-й, еще реже 12-й.

Но вот новая тайна "ELITE"! Запускаем в "галактические" байты цифры - 1,8,1,8,1,8 и путешествуем по этой восьмерке: 1-я, 2-я... казалось бы ничего слишком необычного, но чудеса начинаются с третьей галактики. В этой галактике (4,32,4,32,4,32) правят лишь Анархисты и Коммунисты и здесь мы вдруг натываемся на планету без имени. Все есть, все причиндалы, а вот названия нет. Общий уровень развития низкий (4-5).

Летим дальше. Снова сплошь Феодальная галактика и, чтобы перелететь в следующую, нужно долго мотаться по галактике, отыскивая Феодалов с 10м уровнем развития. Приобретаем у них галактический привод и летим дальше. В пятой галактике - только Мультиправительства и та же беда с приобретением галактического привода. Следующая галактика - рай для ГКЧП! - (32,1,32,1,32,1).

Берем галактический привод и рвем когти из этого "рая" - попадаем... к Анархистам (64,2,64,2, 64,2). В следующую галактику путь закрыт, т.к. уровень развития здесь не превышает 8. Что же это за таинственная галактика, в которую таким вот путем (невозможность приобрести галактический привод) закрыли дорогу?

Приходится отгрузиться, вскрыть 102-байтный блок и выбрать одну из трех дорог - получить миссию "Сверхновая" (46-й байт = 1), приобрести галактический привод (58-й байт = 255) или удвоить шесть "галактических" байтов (128,4,128,4,128,4). В любом случае мы

попадаем в ... "ЧЕРНУЮ ГАЛАКТИКУ". На общей карте мы не видим ни одной звезды, но нажав клавишу "О" видим странную картину - в - галактике полно невидимых звезд и на крупной карте они обозначены только названиями, звезд не видно! Тип правительства - только Анархия, уровень развития - только 2 и 6. Да и названия довольно странные - они двойные:

ZAISRE        RAMATE        INMA  
ZAISRE,       RAMATE,       INMA

и в очень редких случаях мы видим название планеты, состоящее из одного слова - ZAISAR. Много одинаковых названий, например, мы обнаружили

две планеты ZAISRE,  
                         ZAISRE  
три планеты INMA и т. д.  
                         INMA

В описаниях к планетам говорится везде, что цивилизации на этих планетах безобидные (HARMLESS), и находятся в самом начале своего развития.

Мы могли бы рассказывать и рассказывать о новых и новых открытиях и тайнах "ELITE", но довольно. После публикации в номере 5-6 (1992), вероятно, не один десяток пилотов бросились в атаку на галактики и Вы получили, наверное, не одно письмо на эту тему. С интересом ждем публикации этих писем.

Несколько слов о байтах в отгружаемом блоке - фраза "Браво, Командор!" появляется при увеличении байта 15 на единицу.

41-й байт (грузоподъемность) по ходу игры изменяется сама вот таким образом - 20,35,69,138 и снова 20... и т.д. Засылая сюда другие цифры, мы обнаруживаем, что во время игры снова возвращаемся к этим же цифрам (к ближайшей из них), а поставив цифру 69 и подкрепив ее 69 т. груза, мы с удивлением обнаруживаем, что можем загрузить во время игры 138 т. груза. Поставив же в этот байт цифру 138, мы можем здорово "пролететь": после одного-двух перелетов мы не сможем взять больше 20-ти тонн груза и все попытки купить дополнительный грузовой отсек будут пресекаться сообщением "ALL PRESENTS..."

В последней распечатке байт 73 стоит у Вас под вопросом, но мы уже писали, что он означает технический уровень и вновь подтверждаем это. Здесь, как и в байте 18 нужно прибавлять единицу, т. е. 0=1,4=5,11=12 и т. д.

Байт 74 - тип правительства:

- 0 - Анархия
- 1 - Феодалы
- 2 - Мультиправительства
- 3 - Диктатура
- 4 - Коммунисты
- 5 - Конфедерация
- 6 - Демократия
- 7 - Корпорация

В зависимости от этих цифр, видимо, и количество противников у очередной планеты. Мы пробовали заслать корпоративной планете с уровнем 15 цифру 1 и еле-еле отбились от пиратов.

И еще небольшая справка-эксперимент, чтобы "отработать" рейтинг DANGEROUS от и до и получить DEADLY, нужно включить функцию "F" и провести 100 боев с Таргоидами.

И еще несколько слов о "Черных галактиках" (их много в программе). Имена планет могут быть и тройными    ZARGI  
                                 (ZARGI)  
                                 ZARGI

но это не значит, что планета тройная или двойная. Есть еще одна странность. На крупной карте ("О") мы читаем название планеты, затем нажимаем "R", набираем имя этой планеты и курсор прыгает в другой конец галактики(?). На планеты с двойным названием курсор указывает только через "D". Есть и другие чудачества...

ИФК: Большое спасибо Вам Павел и Герасим за огромный труд. Вы правы, по Вашим стопам пошли многие исследователи. Если Вы позволите, то Ваш труд можно было бы назвать "Исследование циклических свойств генератора случайных данных в программе ELITE". Начатое Вами исследование подхвачено и привело к новым открытиям, правда они уже относятся не совсем к игре, а к математике и к программированию. На эту тему мы недавно получили весьма серьезную работу от нашего корреспондента из г. Новосибирска Збитнева В. А. Ее Вы увидите в следующем выпуске "ZX-РЕВЮ" под рубрикой "Профессиональный подход".



# ВОЗВРАЩАЯСЬ К НАПЕЧАТАННОМУ

(С) Ефремов Александр Александрович, г. Челябинск, 1993г.

## Опыт работы с BETA-BASIC 1.8

Предлагаю вниманию читателей небольшие выводы из своего опыта работы на BETA BASIC 1.8 и более ранних версиях. Полагаю, что публикация этого материала может значительно облегчить программирование на BETA BASIC как начинающим, так и уже имеющим опыт работы с ним.

Известные трудности при программировании на BETA BASIC 1.0 и 1.8 вызывает процесс передачи параметров при обращении к процедурам. В отличие от BETA BASIC 3.0, где применяется формат обращения к процедуре типа:

```
PROC Name(a1,a2,...,aN),
```

где  $a_1, a_2, \dots, a_N$  - параметры,

более ранние версии этого языка принуждают использовать громоздкие конструкции с большим количеством операторов LET, что ухудшает читаемость программы и требует значительных затрат оперативной памяти.

Предлагаемый мной метод заключается в использовании для передачи фактических значений формальных параметров процедуры пары операторов READ и DATA. В программе это выглядит таким образом:

```
10 DEF PROC Name
20 READ a1,a2,...,aN
...
50 END PROC
100 DATA x1, x2,...,xN: RESTORE 100: PROC Name
```

где  $x_1, x_2, \dots, x_N$  - значения переменных  $a_1, a_2, \dots, a_N$ .

Кроме экономии памяти и наглядности, метод позволяет использовать одну строку DATA для нескольких вызовов процедуры с одинаковыми параметрами, переводя перед каждым таким обращением указатель оператора READ на строку с необходимыми значениями. Одновременно с этим появляется возможность гибко задавать число параметров:

```
10 DEF PROC Table
20 READ N
30 DIM A (N)
40 FOR I=1 TO N: READ A(I): PRINT A(I) : NEXT I
50 END PROC
100 DATA 4, 1.1, 2.67, 3.15, 4.16
      | |-----|
      | |         |
      | |элементы передаваемого
      | |массива
      | |
      | - количество элементов
      |   массива, передаваемо-
      |   го в процедуру.
110 RESTORE 100: PROC Table
```

Применение такого подхода в моих программах позволило сократить использованные под текст программы ресурсы памяти на 15-20% и значительно повысить наглядность и читаемость. Надеюсь, и другие пользователи BETA BASIC найдут применение методу, описанному выше и по достоинству оценят его преимущества.

## Сортировка русифицированных символьных строк

В статье "Полная русификация Мастерфайл-09" автор предлагает читателям обсудить проблему сортировки русских слов по алфавиту. (См. ZX-РЕВЮ-92, с. 29, 71).

В своих программах я эту "проблему" решаю очень просто. Например, с адреса 60000 я размещаю таблицу длиной 33 байта (ровно столько букв в русском алфавите) и в ней последовательно располагаю коды букв русского алфавита от А до Я. Предположим, что буква А у нас в нашем дополнительном наборе не изменялась, т.е. ей соответствует код 65, как и в стандартном наборе "Спектрума", а буква "Б" у нас закодирована по адресам буквы b (латинская строчная) т.е. ей соответствует код 98. Тогда по адресу 60000 мы запишем число 65, по адресу 60001 число 98 и так далее до ячейки 600032. В ней будет храниться код символа, вместо которого "защита" буква Я.

Больше там ничего не нужно. В программах сортировка у меня происходят на ассемблере, что многократно повышает скорость работы, но это можно сделать и из Бейсика, правда при этом скорость работы немного снизится.

А делается это так. Допустим, у нас есть символьная переменная A\$ = "БАБА ЯГА" (это какая-то фамилия или имя). Программа определяет код первой буквы - у нас это 98. Далее делается цикл:

```
100 FOR M = 60000 TO 60032: IF PEEK M = CODE A$(I) THEN LET M = M - 59999: GO TO ANOTHER
110 NEXT M
```

Т.е. если код первой буквы равен содержимому одной из ячеек нашей таблицы, то переменная М становится указателем, какая эта буква по порядку в русском алфавите.

У нас М становится равным двум. Дальше уже идет сам процесс перестановки (или сортировки) имен. Если же сортировка производится по нескольким начальным буквам, то мы просто организуем еще один цикл, например, с именем N и будем сравнивать PEEK M с CODE A\$(N).

Я не пишу все это более подробно, потому что идея эта сама по себе тривиальна и годится для самостоятельного разбора.

Так что, как видите, тут никакой проблемы и нет, все очень просто. Но еще лучше, как я уже сказал выше сделать всю эту сортировку на языке Ассемблера, особенно если сортируется большой массив данных.

## ОПЕЧАТКА В ПРОГРАММЕ "КРИББЕДЖ" (ZX-РЕВЮ-92, с. 187)

Дианов Р.Ю. из г.Братска Иркутской обл. просит уточнить строку 4515 из программы "Криббедж", опубликованной в N 7,8 "ZX-РЕВЮ-92". Там пропущена переменная:

```
4515 LET ? = 5: LET rlen = 2
```

Нам очень стыдно и неудобно, но почему-то с листингами такое бывает. После подготовки оригинал-макета в печать сколько не выверяй программу и не лови "блох", все равно какая-нибудь гадость просочится.

Правильно должно быть так:

```
4515 LET c = 5: LET rlen = 2
```

# ЧИТАТЕЛЬ - ЧИТАТЕЛЮ

## ПРОСМОТР ЗАЩИЩЕННЫХ ПРОГРАММ

Письмо на тему взлома защищенных Бейсик-программ прислал нам наш юный читатель из г. Чехова Бессонов Александр. Ему всего 14 лет, но несмотря на это, он предложил вполне профессиональную и достаточно квалифицированную программу. Вот, что он пишет.

"Еще полгода назад я написал программу "LOOK BASIC PROGRAMS" (далее LBP). Я использую ее уже долгое время и она позволяет взламывать все программы (я работаю с дисководом и не уважаю клавишу "MAGIC", предпочитаю переписывать игры с ленты на диск по файлам). Также я использую программу "ПАКЕТ", состоящую из LBP, MONS4, LOAD/MERGE (последнюю можно взять, например, в БАЙТЕКе N 1 за 1992 г.).

Я писал эту программу, основываясь на том, что главными способами защиты являются управляющие символы и 5-байтное представление чисел. Введя программу из листинга 1, Вы получите LBP с адреса 55000. Теперь можете выполнить NEW и загружать защищенную программу (через MERGE или "LOAD/MERGE"). Вызов LBP - RANDOMIZE USR 65000.

Формат, в котором LBP выводит Бейсик-программу:

```
10 [23775] (20) REM <<<<<<<< IPF BAV 1992 IP
```

Здесь 10 - номер строки;

[23755] - адрес начала строки (в инверсном виде);

(20) - длина строки (может быть ложной).

Далее идет обозначение управляющих символов (все в инверсном виде):

I - INK (CHR\$ 16,N)

P - PAPER (CHR\$ 17,N)

F - FLASH (CHR\$ 18,N)

B - BRIGHT (CHR\$ 19,N)

V - INVERSE (CHR\$ 20,N)

A - AT (CHR\$ 22, X, Y)

T - TAB (CHR\$ 23, N, 0)

< - BACK (CHR\$ 8).

После каждого числа LBP выводит в квадратных скобках и в инверсном виде настоящее число, представленное в пятибайтной форме. Например:

```
10 [23755] (20) REM <<<<<<<< IP BAV 1992 IP
```

```
20 [23779] (30) INK 6E4 [0]: PAPER 12,5 [3]: LOAD "CODE 12345 [30000]
```

```
30 [23803] (10) RANDOMIZE USR 0 [30000.12]
```

В строке 30 напечатано число 0, а по-настоящему здесь число 30000.12. Это сделано для того, чтобы "вручную" разгадать это число было трудно, а функция USR просто откинет дробную часть.

Примечания:

1. Если после того, как Вы напечатали RAND. USR 65000 Вам выдался не весь листинг, то выполните: CLS: RANDOMIZE USR 65000.

2. LBP не проверяет наличие программы в ОЗУ. Если ее нет, то просто экран заполнится разными символами и в ответ на запрос да в Бейсик."

Комментарий ИНФОРКОМА: это касается второго пункта примечаний. Наличие программы не проверяется не потому, что это недостаток программы LBP, а потому, что заранее невозможно предвидеть, какими способами защиты воспользовался автор защищенной программы. Вам предоставляется "листинг" в том виде, как он может быть интерпретирован Бейсиком, а Вы уже сами должны решать, программа это или случайный набор символов.

## Текст программы LBP представлен в Листинге 1.

## Листинг 1.

```

10 CLEAR 64999: LET C=0: FOR t=65000 TO 65343: READ a: LET c=c+a: POKE t,a: NEXT t
20 IF c<>46381 THEN PRINT "ERROR !": STOP
2000 DATA 62,2,205,1,22,205,3,255,42,83,92,237,91,75,92,237,83,253,254,62,13,205,255,254,62,
13,205,255,254,229,205,40,26,225,229,229,193,205,43,45,62,123,205,255,254,253,203,87,
214,205
2010 DATA 227,45,253,203,87,150,62,125,205,255,254,225,35,35,78,35,70,35,62,40,205,255,254,
229,205,43,45,205,227,45,225,62,41,205,255,254,126,254,32,48,124,253,203,87,214,254,1
6,204,148,254
2020 DATA 254,17,204,156,254,254,18,204,160,254,254,19,204,164,254,254,20,204,168,254,254,
21,204,172,254,254,22,204,176,254,254,23,204,181,254,254,8,204,186,254,253,203,87,150
,254,14,204,198,254,254
2030 DATA 13,32,16,35,229,237,91,253,254,167,237,82,124,181,225,194,251,253,201,35,24,170,
62,73,205,255,254,175,35,201,62,80,24,246,62,70,24,242,62,66,24,238,62,86,24,234,62,7
9,24,230
2040 DATA 62,65,35,24,225,62,84,35,24,220,62,60,43,24,215,205,255,254,35,195,62,254,35,62,
91,205,255,254,221,229,229,221,225,221,126,0,221,94,1,221,86,2,221,78,3,221,70,4,221,
225
2050 DATA 229,205,182,42,253,203,87,214,205,227,45,253,203,87,150,225,62,93,205,255,254,17,
4,0,25,175,201,0,0,229,215,225,201,17,11,255,175,205,10,12,201,128,13,126,76,79,79,75
,32,66
2060 DATA 65,83,73,67,32,80,82,79,71,82,65,77,83,126,13,65,89,32,66,69,83,83,79,78,79,86,32,
65,46,13,18,1,66,65,86,32,127,49,57,57,49,18,0,141

```

## Дисассемблер машинно-кодového блока программы LBP представлен в Листинге 2.

В заключение мы хотим от всей души поздравить начинающего автора с удачным дебютом в "ZX-РЕВЮ".

## Листинг 2.

|      |           |          |             |                                |
|------|-----------|----------|-------------|--------------------------------|
| FDE8 | 3E02      | LD       | A, #02      | ; Открыть канал                |
| FDEA | CD0116    | CALL     | #1601       | ; экрана.                      |
| FDED | CD03FF    | CALL     | #FF03       | ; Вывод сообщения об авторе.   |
| FDF0 | 2A535C    | LD       | HL, (#5C53) | ; Значение сист. перем. PROG.  |
| FDF3 | ED5B4B5C  | LD       | DE, (#5C1B) | ; Значение сист. перем. VARS.  |
| FDF7 | ED53FD FE | LD       | (#FEFD), DE | ; Сохранение его в ячейке.     |
| FDFB | 3E0D      | STROK LD | A, #0D      | ; В A - код "ENTER".           |
| FDFD | CDFFFE    | CALL     | #FEFF       | ; Печать этого символа.        |
| FE00 | 3E0D      | LD       | A, #0D      | ; В A - код "ENTER".           |
| FE02 | CDFFFE    | CALL     | #FEFF       | ; Печать этого символа.        |
| FE05 | E5        | PUSH     | HL          | ; Печать                       |
| FE06 | CD281A    | CALL     | #1A28       | ; номера                       |
| FE09 | E1        | POP      | HL          | ; строки.                      |
| FE0A | E5        | PUSH     | HL          | ; Сохр. адреса на стеке.       |
| FE0B | E5        | PUSH     | HL          | ; Перезапись                   |
| FE0C | C1        | POP      | BC          | ; из HL в BC                   |
| FE0D | CD2B2D    | CALL     | #2D2B       | ; Передача BC на стек калькул. |
| FE10 | 3E7B      | LD       | A, #7B      | ; в A - код "[".               |
| FE12 | CDFFFE    | CALL     | #FEFF       | ; Печать этого символа.        |
| FE15 | FDCB57D6  | SET      | 2, (IY+87)  | ; Включение инверсии.          |
| FE19 | CDE32D    | CALL     | #2DE3       | ; Печать значения числа, пред- |
|      |           |          |             | ; ставленного в 5-байт. виде.  |
| FE1C | FDCB5796  | RES      | 2, (IY+87)  | ; Выключение инверсии.         |
| FE20 | 3E7D      | LD       | A, #7B      | ; В A - код "]".               |
| FE22 | CDFFFE    | CALL     | #FEFF       | ; Печать этого символа.        |
| FE25 | E1        | POP      | HL          | ; В HL - адрес начала строки.  |
| FE26 | 23        | INC      | HL          | ; Теперь в HL - адрес пары     |
| FE27 | 23        | INC      | HL          | ; ячеек с длиной строки.       |
| FE28 | 4E        | LD       | C, (HL)     | ; Перезапись                   |
| FE29 | 23        | INC      | HL          | ; длины строки                 |
| FE2A | 46        | LD       | B, (HL)     | ; в регистровую                |
| FE2B | 23        | INC      | HL          | ; пару BC.                     |
| FE2C | 3E28      | LD       | A, #28      | ; В A - код "(".               |
| FE2E | CDFFFE    | CALL     | #FEFF       | ; Печать этого символа.        |

|      |          |        |                 |                                                                     |
|------|----------|--------|-----------------|---------------------------------------------------------------------|
| FE31 | E5       |        | PUSH HL         | ; Печать                                                            |
| FE32 | CD2B2D   |        | CALL #2D2B      | ; значения                                                          |
| FE35 | CDE32D   |        | CALL #2DE3      | ; длины                                                             |
| FE38 | E1       |        | POP HL          | ; строки.                                                           |
| FE39 | 3E29     |        | LD A, #29       | ; В А - код ")"                                                     |
| FE3B | CDFFFE   |        | CALL #FEFF      | ; Печать этого символа.                                             |
| FE3B | 7E INF   |        | LD A, (HL)      | ; В А - код из памяти.                                              |
| FE3F | FE20     |        | CP #20          | ; Проверка, что за код?                                             |
| FE41 | 307C     |        | JR NC, #FEBF    | ; Если печатный символ,<br>; то переход на PRINT.                   |
| FE43 | FDCB57D6 |        | SET 2, (IY+87)  | ; Если упр. код, то вкл.<br>; инверсии и проверка,<br>; что за код. |
| FE47 | FE10     |        | CP #10          | ; Проверка INK.                                                     |
| FE49 | CC94FE   |        | CALL Z, #FE94   | ; Если да, то выполн. INK.                                          |
| FE4C | FE11     |        | CP #11          | ; Проверка PAPER.                                                   |
| FE4E | CC9CFE   |        | CALL Z, #FE9C   | ; Если да, то выполн. PAPER.                                        |
| FE51 | FE12     |        | CP #12          | ; Проверка FLASH.                                                   |
| FE53 | CCA0FE   |        | CALL Z, #FEA0   | ; Если да, то выполн. FLASH.                                        |
| FE56 | FE13     |        | CP #13          | ; Проверка BRIGHT.                                                  |
| FE58 | CCA4FE   |        | CALL Z, #FEA4   | ; Если да, то выполн. BRIGHT.                                       |
| FE5B | FE14     |        | CP #14          | ; Проверка INVERSE.                                                 |
| FE5D | CCA8FE   |        | CALL Z, #FEA8   | ; Если да, то выполн. INV.                                          |
| FE60 | FE15     |        | CP #15          | ; Проверка OVER.                                                    |
| FE62 | CCACFE   |        | CALL Z, #FEAC   | ; Если да, то выполн. OVER.                                         |
| FE65 | FE16     |        | CP #16          | ; Проверка AT.                                                      |
| FE67 | CCB0FE   |        | CALL Z, #FEB0   | ; Если да, то выполн. AT.                                           |
| FE6A | FE17     |        | CP #17          | ; Проверка TAB.                                                     |
| FE6C | CCB5FE   |        | CALL Z, #FEB5   | ; Если да, то выполн. TAB.                                          |
| FE6F | FE08     |        | CP #08          | ; Проверка BACK.                                                    |
| FE71 | CCBAFE   |        | CALL Z, #FEBA   | ; Если да, то выполн. BACK.                                         |
| FE74 | FDCB5796 |        | RES 2, (IY+87)  | ; Отключение инверсии.                                              |
| FE78 | FE0E     |        | CP #0E          | ; Проверка "ЧИСЛО".                                                 |
| FE7A | CCC6FE   |        | CALL Z, #FEC6   | ; Если да, то выполн.                                               |
| FE7D | FE0D     |        | CP #0D          | ; Проверка "КОНЕЦ СТРОКИ".                                          |
| FE7F | 2010     |        | JR NZ, #FE91    | ; Если это код от 18 до 31,<br>; то переход на PROB.                |
| FE81 | 23       |        | INC HL          | ; Переход к следующему адресу.                                      |
| FE82 | E5       |        | PUSH HL         | ; Сохранение его на стеке.                                          |
| FE83 | ED5BFDFE |        | LD DE, (#FEFD)) | ; в DE - значение VARS.                                             |
| FE87 | A7       |        | AND A           | ; Сброс флага переноса.                                             |
| FE88 | ED52     |        | SBC HL, DE      | ; Проверка на достижение                                            |
| FE8A | 7C       |        | LD A, H         | ; области VARS, а следова-                                          |
| FE8B | B5       |        | OR L            | ; тельно, конца листинга.                                           |
| FE8C | E1       |        | POP HL          |                                                                     |
| FE8D | C2FBFD   |        | JP NZ, #FDFB    | ; Если не конец, то на STROK.                                       |
| FE90 | C9       |        | RET             | ; Если конец, то выход.                                             |
| FE91 | 23       | PROB   | INC HL          | ; Переход к следующему адресу                                       |
| FE92 | 18AA     |        | JR #FE3E        | ; и повторение INF.                                                 |
| FE94 | 3E49     | INK    | LD A, #49       | ; В А - код " "                                                     |
| FE96 | CDFFFE   | KOL    | CALL #FEFF      | ; Печать этого символа.                                             |
| FE99 | AF       |        | XOR A           | ; Обнуление аккумулятора.                                           |
| FE9A | 23       |        | INC HL          | ; Переход к следующему адресу                                       |
| FE9B | C9       |        | RET             | ; и возврат.                                                        |
| FE9C | 3E50     | PAPER  | LD A, #50       | ; В А - код "P".                                                    |
| FE9E | 18F6     |        | JR #FE96        | ; Переход на KOL.                                                   |
| FEA0 | 3E46     | FLASH  | LD A, #46       | ; В А - код "F".                                                    |
| FEA2 | 18F2     |        | JR #FE96        | ; Переход на KOL.                                                   |
| FEA4 | 3E42     | BRIGHT | LD A, #42       | ; В А - код "B".                                                    |
| FEA6 | 18EE     |        | JR #FE96        | ; Переход на KOL.                                                   |
| FEA8 | 3E56     | INV    | LD A, #56       | ; В А - код "V".                                                    |
| FEAA | 18EA     |        | JR #FE96        | ; переход на KOL.                                                   |
| FEAC | 3E4F     | OVER   | LD A, #4F       | ; В А - код "O".                                                    |
| FEAE | 18E6     |        | JR #FE96        | ; Переход на KOL.                                                   |
| FEBO | 3E41     | AT     | LD A, #41       | ; В А - код "A".                                                    |

|       |          |       |      |                         |                               |
|-------|----------|-------|------|-------------------------|-------------------------------|
| FEB2  | 23       |       | INC  | HL                      | ; Пропуск аргумента AT.       |
| FEB3  | 18E1     |       | JR   | #FE96                   | ; Переход на KOL.             |
| FEB5  | 3E54     | TAB   | LD   | A, #54                  | ; В А - код "Т".              |
| FEB7  | 23       |       | INC  | HL                      | ; Пропуск аргумента TAB.      |
| FEB8  | 18DC     |       | JR   | #FE96                   | ; Переход на KOL.             |
| FEBA  | 3E3C     | BACK  | LD   | A, #3C                  | ; В А - код "<".              |
| FEBC  | 2B       |       | DEC  | HL                      | ; Переход к предыдущему адр.  |
| FEBD  | 18D7     |       | JR   | #FE96                   | ; Переход на KOL.             |
| FEBF  | C2FFFE   | PRINT | CALL | #FEFF                   | ; Печать символа и            |
| FEC2  | 23       |       | INC  | HL                      | ; переход к следующему адр.   |
| FEC3  | C33EFE   |       | JP   | #FE3E                   | ; Возврат на INF.             |
| FEC6  | 23       | CHISL | INC  | HL                      | ; Переход к следующему адр.   |
| FEC7  | 3E5B     |       | LD   | A, #5B                  | ; В А - код "[".              |
| FEC9  | CDFFFE   |       | CALL | #FEFF                   | ; Печать этого символа.       |
| FECC  | DDE5     |       | PUSH | IX                      |                               |
| FECE  | E5       |       | PUSH | HL                      | ; Перезапись значения HL.     |
| FECF  | DDE1     |       | POP  | IX                      | ; в регистр. пару IX.         |
| FED1  | DD7E00   |       | LD   | A, (IX+0)               | ; Подготовка                  |
| FED4  | DD5E01   |       | LD   | E, (IX+1)               | ; данных                      |
| FED7  | DD5602   |       | LD   | D, (IX+2)               | ; для расчета                 |
| FEDA  | DD4E03   |       | LD   | C, (IX+3)               | ; значения числа              |
| FEDD  | DD4604   |       | LD   | B, (IX+4)               | ; в 5-байтном виде.           |
| FEE0  | DDE1     |       | POP  | IX                      |                               |
| FEE2  | E5       |       | PUSH | HL                      |                               |
| FEE3  | CDB62A   |       | CALL | #2AB6                   | ; Передача регистров на стек. |
| FEE6  | FECB5706 |       | SET  | 2, (IY+87)              | ; Включение инверсии.         |
| FE EA | CDE32D   |       | CALL | #2DE3                   | ; Печать значения числа.      |
| FEED  | FDCB5796 |       | RES  | 2, (IY+87)              | ; Выключение инверсии.        |
| FEF1  | E1       |       | POP  | HL                      |                               |
| FEF2  | 3E5D     |       | LD   | A, #5D                  | ; В А - код "]".              |
| FEF4  | CDFFFE   |       | CALL | #FEFF                   | ; Печать этого символа.       |
| FEF7  | 110400   |       | LD   | DE, #0004               | ; Переход на оставшиеся       |
| FEFA  | 19       |       | ADD  | HL, DE                  | ; 4 байта вперед.             |
| FEFB  | AF       |       | XOR  | A                       | ; Обнуление аккумулятора.     |
| FEFC  | C9       |       | RET  |                         |                               |
| FEFB  | 0000     | POKE  | DEFS | 2                       | ; Здесь хранится VARS.        |
| FEFF  | E5       | WRITE | PUSH | HL                      | ; Процедура                   |
| FF00  | D7       |       | RST  | #10                     | ; печати                      |
| FF01  | E1       |       | POP  | HL                      | ; символа.                    |
| FF02  | C9       |       | RET  |                         |                               |
| FF03  | 110BFF   | AWTOR | LD   | DE, #FF0B               | ; Вывод                       |
| FF06  | AF       |       | XOR  | A                       | ; сообщения                   |
| FF07  | CD0A0C   |       | CALL | #0C0A                   | ; об авторе.                  |
| FF0A  | C9       |       | RET  |                         |                               |
| FF0B  | 800D     | S00B  | DEFB | 128, 13                 |                               |
| FF0D  |          |       | DEFM | "`LOOK BASIC PROGRAMS`" |                               |
| FF22  | 0D       |       | DEFB | 13                      |                               |
| FF23  |          |       | DEFM | "BY BESSONOV A."        |                               |
| FF31  | 0D1201   |       | DEFB | 13, 18, 1               | ;                             |
| FF34  |          |       | DEFM | "BAV " C 1991           |                               |
| FF3D  | 12008D   |       | DEFB | 18, 0, 13+128;          |                               |

## КАКИЕ КОМПЬЮТЕРЫ МЫ ВЫБИРАЕМ

Уважаемая редакция!

В своем письме я затронул одну из важных проблем компьютеризации - выбор начинающим радиолюбителем модели персонального компьютера (ПК) для самостоятельной сборки. По самым скромным подсчетам за 1991 год у нас в стране было собрано более 90 тыс. Spectrum-совместимых ПК, что существенно превысило общее количество всех других, в том числе и промышленно выпущенных, и ввезенных из других стран компьютеров различных моделей. Учитывая достаточно широкий выбор наиболее распространенных версий. По отзывам всех знакомых мне радиолюбителей, подобные публикации всегда вызывают повышенный интерес, что важно в условиях конкуренции различных изданий для радиолюбителей.

Начиная с 1987 года у магазина "Пионер", затем на Соколе, в Покровско-Стрешнево и еще бог знает где в Москве, а впоследствии у станции метро "Тушинская" появилось около 10 основных моделей "Spectrum"-совместимых ПК. Эти модели и их различные версии стали теперь основным парком бытовых ПК в бывшем СССР благодаря их простоте, дешевизне и огромному количеству отличных программ.

Радиолюбителю, а порой и профессионалу сложно разобраться в преимуществах одних моделей перед другими, так как практически отсутствует информация о тонкостях и сложностях в настройке и эксплуатации, а также совместимости с программным обеспечением. Проблема также состоит в том, что в настоящее время сбыт некоторых моделей ПК оказался в кругу интересов их разработчиков. Как правило, вся информация о таких ПК - это реклама, и теневые стороны не освещаются.

Первой массовой моделью стал ПК "Москва". При сложности в изготовлении и настройке, а также при большом количестве доработок, которые необходимо вносить, этот ПК до сих пор остается наиболее полноценным повторением модели "ZX-SPECTRUM" с точки зрения машинных циклов и организации памяти. Следует также отметить, что в телевизионном кадре у "Москвы" 312 строк, что соответствует стандарту, а не 320, как у большинства других моделей. Только в этой модели предусмотрено торможение процессора при видео-выводе и обращении в адреса с 4000H по 8000H (16384-32767 DEC). Однако, сложности в настройке делают ПК "Москва" малопривлекательным для начинающих радиолюбителей.

Следующим этапом ПК был "Балтик" (название произошло от того, что плату и схему разработали в Вильнюсе). Основным достоинством этой модели является простота и высокая надежность в работе. На плате мало исправлений и компьютер прост в наладке. Но наличие микросхем K556PT4 и K155PE3, необходимых для организации машинных циклов и работы видеопроцессора, которые необходимо предварительно программировать, накладывает некоторые ограничения на доступность комплектующих. Жесткая организация машинных циклов и существенные отличия в организации работы памяти, а также повышенная вследствие этого до 4 Мгц тактовая частота процессора, делает эту модель менее совместимой программно. По этим причинам в настоящее время версия "Балтик" мало распространена.

Существенным шагом явилось появление модели "Москва 128" (разработанной скорее всего не в Москве). Это первая модель, где используется "прозрачное" ОЗУ, т.е. режим, в котором процессор при обращении к памяти не тормозится. Прототипом этой модели послужил ПК "SINCLAIR-128". Но использование работы памяти в критичных режимах, а также отсутствие музыкального сопроцессора сделало эту модель весьма убогим повторением.

В "Москве 128" был впервые предусмотрен интерфейс принтера LX-PRINT и оригинальный программируемый джойстик, также предусмотрено подключение двух KEMPSTON-джойстиков и полноценный TV-RGB выход. К несчастью, эта модель не получила распространения из-за малого количества программ, рассчитанных только под "SINCLAIR-

128" и сложностей в настройке, проявляющихся как "сбойность" в ОЗУ.

Самой массовой моделью "Spectrum"-совместимых машин без контроллера дисководов стал "Ленинград 1". Его основные достоинства: простота, дешевизна, небольшое количество исправлений на плате, повторяемость. Из-за отсутствия достойных конкурентов эта модель приобрела большую популярность. К сожалению, за простоту пришлось заплатить плохой совместимостью. Неправильная адресация портов, а как следствие - побочные эффекты и сложность подключения внешних устройств. На плате не предусмотрено системного разъема и подключить любую периферию очень сложно. Хотя торможение процессора в цикле M1 (при выборке кода инструкции из ОЗУ) и облегчает режим работы памяти, но сильно вредит совместимости с программным обеспечением. В настоящее время эта модель постепенно сходит с рынка, хотя и ее последователи не лишены многих допущенных в ней ошибок.

| Название ПК   | Год  | Размер  | К-во микр | Контроллеры |    |      | Сист. шина | Дополн. особенности     |
|---------------|------|---------|-----------|-------------|----|------|------------|-------------------------|
|               |      |         |           | BETA        | LX | KEMP |            |                         |
| "Москва"      | 1988 | 200*125 | 68        | -           | -  | -    | +          | MICRODOS<br><br>CPM 2.2 |
| "Балтика"     | 1988 | 210*120 | 48        | -           | -  | +    | -          |                         |
| "Москва 128"  | 1989 | 280*137 | 63        | -           | +  | ++   | +          |                         |
| "Ленинград-1" | 1989 | 208*125 | 44        | -           | -  | +    | -          |                         |
| "Пентагон"    | 1989 | 278*142 | 64        | +           | -  | +    | -          |                         |
| "Красногорск" | 1990 | 224*141 | 42        | -           | -  | +    | -          |                         |
| "Ленинград-2" | 1991 | 210*115 | 48        | -           | -  | +    | +          |                         |
| "ZX-PROFI"    | 1991 | 335*152 | 105       | +           | +  | -    | -          |                         |
| "Пентагон128" | 1991 | 312*142 | 84        | +           | +  | -    | -          |                         |
| "ATM-turbo"   | 1991 | 312*132 | 99        | +           | +  | -    | -          |                         |
| "Angstrom"    | 1992 | 275*115 | 13        | -           | -  | -    | -          |                         |

**ПРИМЕЧАНИЯ:**

1. Размер платы указан в мм.
2. Количество корпусов микросхем может отличаться в различных версиях модели ПК.
3. Указывается наличие на плате ПК контроллера дисководов, для TR-DOS, интерфейса принтера ZX-LPRINT, интерфейса KEMPSTON-JOYSTICK.
4. Указывается наличие на плате места под разъем системной шины. Системные шины разных моделей ПК не стандартны.
5. ПК "ZX-FROFI" состоит из двух плат. Указан размер наибольшей платы.

Одновременно с "Ленинградом 1" на рынке появился компьютер "Пентагон" или "Пентагон 48", прозванный так за пятиугольную разводку земляной шины по контуру печатной платы. Это была первая модель SPECTRUM-совместимого ПК, в котором на одной плате расположен компьютер и контроллер дисководов. "Пентагон" много позаимствовал от "Москвы 128", в частности, "прозрачное" ОЗУ, адресацию портов и недостатки, свойственные "Москве 128". Динамическая память в этой модели работает на предельных частотах, да и плата разведена не лучшим образом, из-за чего приходится усиливать шины питания. На плате отсутствует схема формирования TV RGB сигнала и ее приходится делать навесной. Как положительную сторону этой модели можно отметить устройство ввода с магнитофона (на КМОП К561ЛН2), а как отрицательную - отсутствие системного разъема. "Пентагон 48" популярен еще и сейчас, хотя постепенно вытесняется аналогичными машинами со 128-килобайтами ОЗУ.

Чуть позднее "Пентагона" на рынке появился ПК "Красногорск". Это оригинальная модель, в которой для формирования телевизионных сигналов используется ПЗУ К573РФ2(5), с защитой в ней таблицей. В "Красногорске" использован хоть и прозрачный, но облегченный за счет жестких машинных циклов режим ОЗУ, на плате разведен формирователь TV-RGB-сигналов. Эта версия ПК стала попыткой исправить недостатки "Балтики" и "Ленинграда 1", но из-за трудности приобретения ПЗУ с таблицей, сложности расширения, из-за отсутствия системного разъема и неправильной адресации портов, эта модель не стала такой же массовой, как "Ленинград 1". В настоящее время "Красногорск" очень мало распространен на рынке.



После появления в начале 1990 года "Красногорска", в силу ряда причин на рынке бытовых ПК больше года не появлялось новых моделей "Spectrum"-совместимых компьютеров. Появление ПК "Орион 128" на базе процессора КР580ИК80, ни с чем не совместимого программно, не оказало существенного влияния на развитие бытовых ПК. Однако, весной и летом 1991 года на рынке появилось сразу несколько новых моделей. "Ленинград 2" представляет собой улучшенный вариант "Ленинграда 1", по сравнению с ним в новой модели исправлена адресация порта KEMPSTON-джойстика, хотя остались ошибки с адресацией "бордюрного" порта FE (254). Видеопроцессор формирует 312 строк в кадре, выведена внешняя шина. "Ленинград 2" существенно лучше своего предшественника, но по всей видимости таким же популярным он не стал. Одной из причин этого является сложность расширения, как и у всех предыдущих ПК, отсутствие контроллеров, разработанных под конкретную модель.

Компьютер "ZX-PROFI" первый из разработанных у нас ПК, в котором помимо режима 48 и 128 предусмотрено и использование операционной системы СРМ. Компьютер состоит из двух плат и достаточно сложен в настройке, но имеет наиболее полный набор периферии в одном блоке. Авторы и распространители проводят правильную рыночную политику, направленную на широкое распространение этой модели. Но их ошибкой является то, что сложный и дорогой полупрофессиональный ПК не нужен основной массе потенциальных потребителей. Достоинством этой модели является режим "TURBO", а недостатком - неполная совместимость, как с "ZX-SPECTRUM", так и с СРМ ПК "ROBOTRON 1715".

По сравнению с "ZX-PROFI" "Пентагон 128" оказался более простым и массовым ПК, хотя по сути это лишь скрещивание "Москва 128" и "Пентагон 48" без существенного улучшения. Машина сложна в настройке, критична к ОЗУ, на плате нет места под музыкальный сопроцессор AY-8912, но в настоящее время этот компьютер, к сожалению, единственный в своем роде.

"АТМ-turbo" - чисто коммерческий ПК. Политика его авторов или изготовителей, очевидно направленная на защиту "авторских прав", скорее направлена на получение максимальной прибыли в короткие сроки. Это несомненно оригинальный ПК, но он является и самым дорогим и сложным в настоящее время. В нем предусмотрено несколько режимов графики (только в СРМ), но, к сожалению, те "навороты" - наличие АЦП, ЦАП и элементов для модема и АОНа, на одной плате, в настоящее время никак не поддерживаются программно. В компьютере нет достаточной гибкости и даже системного разъема. Похоже, проблемы совместимости с "ZX-Spectrum" разработчиков интересовали мало, и "АТМ" можно скорее рассматривать как СРМ-совместимый ПК.

На время написания данной статьи последней моделью является "ANGSTREM", в нем впервые применена микросхема 1515 ХМ 1. Это набор счетчиков, мультиплексоров и мелкой логики. Простота в изготовлении и дешевизна, несомненно сделают в ближайшее время эту модель популярной. Но существенным шагом в ней явилось лишь технологическое упрощение с использованием 1515 ХМ 1 вместо набора отдельных микросхем. Среди достоинств этого ПК - простота в настройке, а среди недостатков - малая гибкость, в частности, невозможность расширения ОЗУ и дефицитность 1515 ХМ 1.

А что же дальше? Недавно получена информация о скором появлении нового ПК с двумя процессорами Z-80. Со слов авторов в нем решены как почти все проблемы совместимости, так и возможность расширения. Плата в этой модели похожа по структуре на Motherboard IBM PC, т.е. с несколькими системными разъемами. Хотелось бы верить, что это тоже не реклама. А разработчикам новых ПК желательно учесть: доступность, небольшой набор элементов, надежность, открытость архитектуры (легкость расширения), а с точки зрения совместимости, правильную адресацию портов, наличие на плате так называемого "порта FF", правильное положение и длительности сигнала INT (запроса прерывания). Ну, а тем, кто не хочет ждать (да и дождется ли) появления "идеального" ПК можно выбрать модель по вкусу и карману.

Желаем Вам удачи!

## НОВЫЕ POKES

Наш корреспондент из г. Красноярска Татаренко А.А. предлагает вниманию читателей некоторые разысканные им POKES:

1. SPELL BOUND - 27871,0: 36133,0 - энергия  
или 35101,195: 35102,53: 35103,206
2. DIZZY 2 - 29289,201 - неуязвимость
3. DIZZY 3 - 42481,X - кол-во жизней  
63001,0 - беск. жизни
4. DIZZY 4 - 29623,0: 29624,195 - беск. жизни
5. DIZZY 5 - 51291,0 - беск. жизни
6. STAR BOWLS - 47806,X - кол-во жизней  
46278,0 - беск. жизни
7. SCUMBALL - 49098,X - кол-во жизней
8. COCORAMA - 43855,0: 48658,0 - беск. жизни
9. TOMCAT - 37174,0 - беск. жизни  
37171,201 - неуязвимость
10. TRACER - 50273,0 - время  
50613,0 - энергия  
50649,0 - беск. жизни
11. TWISTER - 42412,0: 42490,0 - энергия
12. DARK FUSION - 50407,6: 50408,0 - беск. жизни
13. XENON - 25148,201 - беск. жизни
14. CONQUEST (ERBE SOFT) - 62370,0 - останавливает противников

## РАСПАХНУТЫЕ ДВЕРИ ИЛИ ОТКРЫТЫЙ ЗАНАВЕС?

Глубокоуважаемая редакция!

Вот уже третий год я являюсь постоянным читателем Вашего журнала "ZX-РЕВЮ", неизменно следя за всеми публикациями, касающимися как "секретов" ZX-Spectrum, так и приемов программирования на языке Ассемблера Z-80. Сам я занимаюсь созданием микропроцессорных систем с начала 1988 года и не могу не оценить той услуги, которую Вы мне оказываете. Надо полагать, что эту услугу Вы оказываете всем читателям Вашего журнала!

За время работы со своим компьютером (я три года эксплуатировал "Новосибирский вариант", а сейчас уже год как построил "Москву-128" с Beta Disk Interface) мной создана довольно обширная библиотека процедур в машинных кодах, которую я широко использую при написании собственных программ.

Руководствуясь тем, что долги надо возвращать (а я, получая от Вас информацию, считаю себя Вашим должником), а также желая поделиться с другими читателями собственными разработками (негоже каждый раз изобретать заново велосипед), я отдаю на Ваш суд одну из своих разработок. Если сочтете необходимым, поступите с ней по Вашему усмотрению.

К написанию этой процедуры меня натолкнул эффект раздвигающихся дверей в игровой программе "ROBIN OF THE WOOD" фирмы "ODIN". Отсюда и название ее "DOORS". Не зная, как этот эффект достигается в фирменной программе (я ее не вскрывал, как-то еще не доходили руки), но встретив в "ZX-РЕВЮ-92" статью "40 лучших процедур", сразу все понял и спустя чуть больше двух часов подпрограмма была готова.

Предлагаемая процедура осуществляет очистку области пикселей и установку рабочих атрибутов экрана, но в отличие от стандартной процедуры ПЗУ CLS (#0D6B) этот процесс сопровождается сравнительно плавным перемещением пикселей влево и вправо от центра экрана (каждый раз на одно знакоместо слева и справа от центра) и заполнением свободных от пикселей столбиков новыми атрибутами, то есть довольно точно имитируется эффект раздвигающихся дверей. Процедура занимает 101 байт и еще один байт необходим для хранения данных - текущих атрибутов экрана (или вновь устанавливаемых атрибутов). Столь небольшой ее объем получен за счет использования всех регистровых пар как основного, так и альтернативного блоков, причем основной блок участвует в перемещении пикселей, а альтернативный - в установке атрибутов. Для обеспечения корректного выхода из процедуры требуется сохранение исходного адреса в регистровой паре H'L', что также осуществляется в ней, давая возможность ее использования и как машинно-кодовой вставки в BASIC-программы.

Для полной очистки всех строк пикселей экрана (#C0 - устанавливается в регистре С) требуется #0F пар перемещений знакомест (устанавливается в регистре В), тогда как число пар атрибутов, устанавливаемых в процессе работы, должно быть равно числу знакомест в полустроке, то есть #10. С этой целью в регистр А копируется константа #0F из регистра В, процесс перемещения пикселей завершается при обнулении регистра В, а процесс установки атрибутов - до появления отрицательного числа в аккумуляторе (контролируется флаг знака).

В регистровой паре HL устанавливается начальный адрес #4000 области пикселей, в регистровой паре H'L' - начальный адрес \$5800 области атрибутов экрана. В регистровых парах DE и D'E' - количество знакомест в строке (#0020), а в регистре С - константа #18(24) - число знакорядов в экранной области.

В приведенном ниже листинге предлагаемой процедуры:

1) В адресах C000 - C009 - установка новых атрибутов экрана. Если предполагается сохранять атрибуты неизменными, то эти команды можно исключить вместе с константой в адресе C065 (ATTR) и устанавливать их непосредственно в командах по меткам A1 и A2. В

этом случае размер процедуры сокращается на 10 байт;

2) В адресах C015 - C01E перемещение пикселей на одну пару знакомест во всех 192 (#C0) строках экранной области. При этом в адресах C016 - C020 - перемещение в левой половине экрана, начиная с левого края, а в адресах C023-C02E - перемещение в правой половине экрана, начиная с правого края;

3) В адресах C03F - C045 - перемещение атрибутов на одну пару знакомест во всех 24 (#18) знакорядах экрана. При этом освободившиеся знакоместа заполняются новыми атрибутами. В адресах C040-C04B - операции в левой половине экрана, начиная с левого края, а в адресах C04E-C054 - операции в правой половине экрана, начиная с правого края.

```
*****
(C) SpySoft  PROGRAM "DOORS"
    Written by M. N. Stinov
        KHABAROVSK-1993
*****
```

```
C000          ORG    #C000
C000 3A65C0    LD     A, (ATTR)
C003 3249C0    LD     (A1+1), A
C006 3258C0    LD     (A2+1), A
C009 112000 DOORS LD    DE, #0020
C00C 060F      LD     B, #0F
C00E 78        LD     A, B
C00F F5        L7     PUSH  AF
C010 0EC0      LD     C, #C0
C012 210040    LD     HL, #4000
C015 C5        L5     PUSH  BC
C016 E5        PUSH  HL
C017 23        L1     INC   HL
C018 7E        LD     A, (HL)
C019 2B        DEC    HL
C01A 77        LD     (HL), A
C01B 23        INC    HL
C01C 10F9      DJNZ   L1
C01E 72        LD     (HL), D
C01F E1        POP    HL
C020 19        ADD    HL, DE
C021 C1        POP    BC
C022 C5        PUSH   BC
C023 E5        PUSH   HL
C024 2B        DEC    HL
C025 2B        L2     DEC    HL
C026 7E        LD     A, (HL)
C027 23        INC    HL
C028 77        LD     (HL), A
C029 2B        DEC    HL
C02A 10F9      DJNZ   L2
C02C 72        LD     (HL), D
C02D E1        POP    HL
C02E C1        POP    BC
C02F 0D        DEC    C
C030 20E3      JR     NZ, L5
```

```
C032 F1        POP    AF
C033 3D        DEC    A
C034 D5        PUSH   DE
C035 06        EX     AF, AF'
C036 D9        EXX
C037 D1        POP    DE
C038 E5        PUSH   HL
C039 210058    LD     HL, #5800
C03C 01180F    LD     DC, #0F18
C03F C5        L6     PUSH  BC
C040 E5        PUSH  HL
C041 23        L3     INC   HL
C042 7E        LD     A, (HL)
C043 2B        DEC    HL
C044 77        LD     (HL), A
C045 23        INC    HL
C046 10F9      DJNZ   L3
C048 3500      A1     LD     (HL), #00
C04A E1        POP    HL
C04B 19        ADD    HL, DE
C04C C1        POP    BC
C04D C5        PUSH   BC
C04E E5        PUSH   HL
C04F 2B        DEC    HL
C050 2B        L4     DEC    HL
C051 7E        LD     A, (HL)
C052 23        INC    HL
C053 77        LD     (HL), A
C054 2B        DEC    HL
C055 10F9      DJNZ   L4
C057 3500      A2     LD     (HL), #00
C059 E1        POP    HL
C05A C1        POP    BC
C05B 0D        DEC    C
C05C 20E1      JR     NZ, L6
C05E E1        POP    HL
C05F D9        EXX
C060 08        EX     AF, AF'
C061 F20FC0    JP     P, L7
C064 C9        RET
C065 28        ATTR  DEFB  #28
```

# STRATEGIC GAMES

Сегодня мы предлагаем Вашему вниманию обзор двух программ стратегического жанра. Они не случайно объединены здесь в одной подборке, поскольку являют собой полные противоположности и могут служить поучительными примерами для тех, кто сами подумывают о написании подобных программ.

Первая программа очень тщательно проработана с содержательной и исторической точки зрения. Она заслуживает высших оценок, но... с точки зрения играбельности с ней не все в порядке. Лишь очень преданные стратегии пользователи, располагающие к тому же солидными запасами свободного времени смогут преодолеть эту прямо скажем незахватывающую игру. Ее никак нельзя рекомендовать начинающим стратегам, ибо после нее они надолго потеряют вкус к этому жанру.

Вторая программа демонстрирует редкий казус. Она плохо проработана с военно-исторической точки зрения, содержит немало "ляпов", задумана как скучное противостояние двух армий, но... именно благодаря этим "ляпам" неожиданно оказывается играбельной и даже интересной.

## ANNALS OF ROME

"PSS"

## ANNALS OF ROME

© MCMLXXXVI PSS

Обозреватель:"ИНФОРКОМ"

В античные времена люди полагали, что у каждого города есть свои боги, которые свысока наблюдают за его судьбой. А римляне, кроме того, были уверены, что самые сильные боги приглядывают за Римом.

В этой игре Вам предстоит попробовать свои силы в качестве такого бога. Вы начинаете игру в 273 г. до н.э., когда Рим имел неустойчивый контроль над территорией современной Италии и должны развиться в могущественную империю и довести ее до падения Константинополя в 1453 г., т.е. прожить немалый исторический период длиной в 1726 лет.

География игры охватывает мир, известный в те времена, с центром в Средиземноморье. Он разделен на 28 отдельных регионов, в каждом из которых есть свое население и своя система управления.

Игра строится по тактовому принципу. Через неравные временные интервалы Вам предлагается выдать управляющее воздействие. Длина каждого такого интервала зависит от того, как много исторических событий произошло в этот период, но в среднем на одно столетие приходится примерно 8 ходов. Вы управляете политической карьерой сенаторов, производите назначения на должности, распределяете обязанности по защите империи и неуклонно расширяете область, подвластную Вашему управлению.

Боевые сражения в этой игре не главное. Это скорее игра, рассчитанная не на военную, а на политическую стратегию. Крупные военачальники продвигаются все выше и выше после одержанных побед, потом стареют и сходят со сцены, а иногда погибают в боях, нередко их амбиции приводят к мятежам и переворотам. Непокойно и на границах империи.

По мотивам этой игры можно писать книги и романы. Самое интересное, что любые индивидуальные усилия героев игры такого масштаба выглядят мелкими и

незначительными. Единственным подлинным героем игры оказывается город. И этот город - Рим, а Вы как бы его верховное божество.



Это отличная игра и ее можно рассматривать даже как обучающую по основам политической теории, но у нее есть огромный недостаток - она очень медленна. Порядка двух часов уходит на то, чтобы сделать три-четыре хода, пока компьютер перебирает раз за разом все подвластные Вам провинции и принимает от Вас управляющие воздействия. Впрочем, в какой-то степени это выглядит реалистичным в том смысле, что и реальные политические лидеры проводят ежедневно многие часы, изучая политические карты и размышляя. Хотя в игре это может быстро наскучить.

Игра не имеет логического финала. Пользователь сам решает, где ему остановиться; когда его империя начинает разваливаться и он не имеет уже реальных шансов на будущее. Кроме того, будущему пользователю, начинающему игру, предстоит потратить немало времени для того, чтобы разобраться, что делает какая клавиша и что обозначают те или иные значки на экране.

## GALLIPOLI

"CCS"



Обозреватель: "ИНФОРКОМ."

Сражение на полуострове Гелиболу относится к тем военным кампаниям, которые блестяще прорабатываются в теории, но совсем по-другому происходят на практике. Это же можно отнести и к этой игре, которой в значительной степени не хватает реализма, хотя играбельность ее оказалась высокой.

К началу 1915 г. на западном фронте сложилась напряженная ситуация сдерживаемого равновесия и английское командование предприняло попытку нарушить стратегический баланс сил путем высадки своих войск на полуострове, с которого открывался бы доступ к столице Турции, находившейся в союзе с Германией.

Идея состояла в том, чтобы вывести Турцию из войны и тогда открылась бы возможность значительного расширения коммуникаций между западными союзными державами и Россией. На выполнение этой задачи был брошен десант в составе английских, новозеландских и австралийских частей общей численностью более 60 тыс. человек.

К несчастью, турки оказались намного более серьезными соперниками, чем кто-либо

на западе до этого мог предположить, а условия местности на полуострове не шли ни в какое сравнение с условиями на европейском ТВД. Так что 22 тысячи турецких войск стойко держали оборону против превосходящих сил противника. Их главной целью стало продержаться до подхода подкреплений, после чего чисто логически английскому десанту остается только утопиться в море.

Бывают стратегические игры, в которых основу составляет умелое маневрирование войсками, бывают игры, в которых основой является управление снабжением, здесь же в основу игры создатели заложили инженерную подготовку операций. Если говорить просто, основной упор здесь сделан на ... рытье окопов в каменистой местности. Что может быть противнее?!!



Вы можете создавать целые линии окопов, соединять их ходами сообщения, подводить минные ходы под укрепления противника, ставить мины и контрмины. В общем, если все было бы так, как задумали игру ее создатели, это была бы одна из скучнейших игр.

К счастью, с военно-исторической точки зрения в игре был допущен серьезный промах. Полагая, что в сражении мало использовалась артиллерия, программисты создали для союзников слишком простые условия для победы. На самом деле артиллерия использовалась широко, а открытая скалистая местность очень этому благоприятствовала.

Упущение этого важного фактора из виду привело к тому, что искушенный игрок не будет в поте лица копать километры окопов, а сделает все тривиально просто. Собрав все силы в один кулак и, невзирая ни на какие чудовищные потери (войска-то ведь не живые), он прорвет фланг турок а затем, выйдя на равнины, прокатится тяжелым катком по их тылам. Захват же баз снабжения в Тылу турецких войск оказывается достаточным условием победы.

Фирма CCS редко делает подобные ошибки и эта игра, к счастью, является исключением. Тем не менее, если забыть об исторических реалиях, то при таком антиисторичном развитии игры, она оказывается очень играбельной.

# СОВЕТЫ ЭКСПЕРТОВ

Emlyn Hughes International Soccer  
AUDIOGENIC SOFTWARE LTD



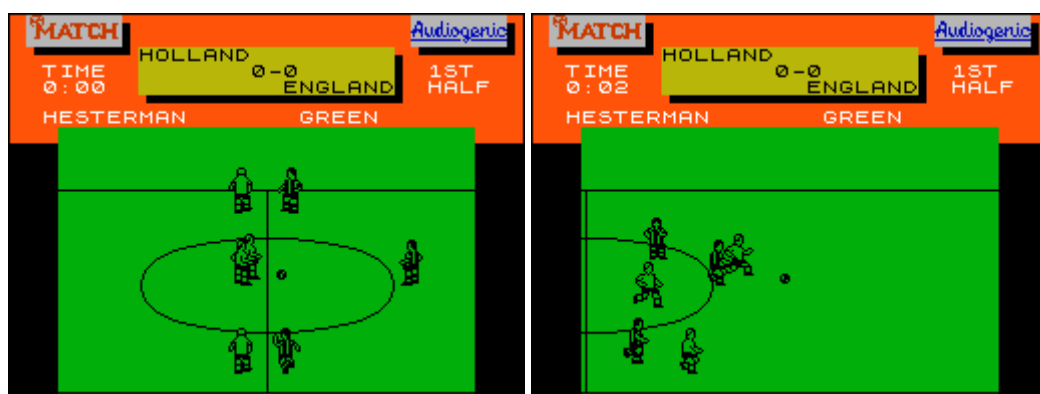
Эксперт: Матвеев Ю. А. г. Москва

До конца игры оставалось две минуты. Сборная России проигрывала со счетом 1:2 в Финальном матче кубка Лиги. Английские болельщики, предвкушая победу своей команды, дружно и весело распевали национальные песни и бросали с трибун длинные бумажные ленты. Полотнища английских флагов гордо развевались над ликующими англичанами.

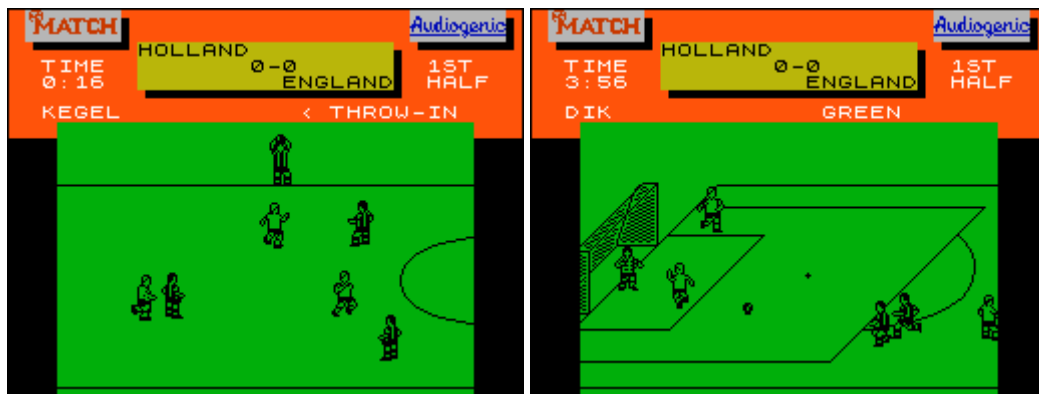
Не желая мириться с поражением, сборная России отчаянно атаковала, пытаясь спасти игру. Судья посматривал на секундомер и, казалось, вот-вот прозвучит финальный свисток.

Иванов получил пас из глубины поля и на мгновение притормозил: второй номер англичан Смит тараном шел на него, рискуя заработать штрафной удар. Из-под ноги Смита Иванов мягко переправил мяч Сидорову и тот, четко его обработав, мощно пробил в правый верхний от вратаря угол. Мяч, словно снаряд, влетел в ворота. Огромный стадион замер, в то время как немногочисленная группа российских болельщиков взорвалась криками и аплодисментами. 2:2! Судья указал на центр поля и, спустя несколько секунд, дал финальный свисток. Теперь зависело от того, как команды сыграют в дополнительное время.

После игры Президент ФИФА должен был торжественно вручить золотой Кубок Лиги команде победителей. Один мяч - вся жизнь, проигрывать никто не хотел...







\* \* \*

В этой игре вам предоставляется возможность участвовать в международном турнире по футболу среди лучших команд Европы. В отличие от многих игр подобного типа "Международный футбол" привлекает к себе внимание неплохой мультипликационной графикой и прекрасной динамикой. Зрелищность - тоже одно из качеств этой игры. Ситуации, возникающие на игровом поле, напоминают сцены реальных футбольных матчей. Игроки передвигаются осмысленно и точно. Это особенно заметно на усложненных уровнях игры. Как оппонент, компьютер играет достаточно сильно, и Вам не раз придется начинать с центра поля.

В игру заложены практически все правила футбола: штрафные удары, выбросы из-за боковой, угловые, удары от ворот и, что особенно интересно, пенальти. Вы можете во время матча произвести замены, а также переставить игроков, усиливая тем самым атаку или защиту.

В "Международном футболе" есть возможность отдать управление игроками компьютеру, а самому заняться исключительно менеджментом, составляя команду из игроков, наиболее подготовленных к матчу. Кто и на каком месте будет играть, Вы решаете сами. В этом случае игра превращается в деловую, а Вы ощущаете себя в роли тренера, который может лишь переживать за свою команду и вовремя производить замены и перестановки.

После загрузки программы компьютер спросит Вас о выборе управления для первого игрока: "КЛАВИАТУРА ИЛИ ДЖОЙСТИК (K/J)?"

Если Вы выбираете джойстик, то:

"КЕМПСТОН ИЛИ СИНКЛЕР (K/S)?"

Если Вы решили использовать джойстики для обоих игроков, то рекомендуется назначить Синклер-джойстик для первого игрока, а Кемпстон - для второго, поскольку замечено, что на некоторых моделях "Спектрум"-совместимых компьютеров при другом выборе Синклер-джойстик не работает. После того, как первый игрок выбрал управление, компьютер попросит подтвердить выбор:

"ВСЕ ПРАВИЛЬНО (Y/N)?"

После нажатия "Y" компьютер предлагает выбрать управление второму игроку. Если выбирается клавиатура, Вам нужно будет задать клавиши управления. Затем Вы вновь подтверждаете свой выбор и переходите к главному меню. С помощью курсора Вы можете выбрать один из четырех разделов:

OPTIONS  
COLOURS  
GAME  
DISPLAYS

### OPTIONS

В этом разделе Вы можете задать основные параметры игры:

DURATION - (продолжительность матча) - от 2 до 90 минут.

1 OR 2 Vs C - (один или два игрока против компьютера)

EXTRA TIME - (дополнительное время) - во время кубковых матчей в случае ничьей по

окончании основного времени будет назначено дополнительное.

VIEW C Vs C - (просмотр, как компьютер играет сам с собой).

HOME & AWAY - (домашние и выездные игры) - при включении этой опции компьютер будет играть сильнее в домашних матчах.

PRACTICE - (тренировка)

SKILL LEVEL - (выбор уровня игры) - 1...10

EQUAL SKILL - (равный уровень) - при такой установке все команды, управляемые компьютером, играют на одном уровне.

BACKHEELS - (пас пяткой)

KICK DIRECTIONS - (направление полета мяча) - 1,3,5,

AUTO GOALIE - (автоматическое управление вратарем) - при включении этой опции Ваш вратарь управляется компьютером.

AUTO RESELECT - (автоматический выбор игроков) - в этом режиме компьютер переключает управление на того игрока, который находится ближе всех к мячу. Выключив это режим, Вы можете подключить к управлению любого футболиста, находящегося в поле зрения экрана. Переключение осуществляется клавишей "FIRE".

Pts FOR WIN - (очки за победу) - 2 или 3.

SUBSTITUTES - (замены) - 2 или 1.

EXIT MENU - (выход) - после настройки игры Вы возвращаетесь в основное меню.

## COLOURS

Выбрав этот раздел, Вы получите возможность изменить цвет поля опцией PITCH COLOUR или изменить цвет формы игроков: PLAYER COLOUR. Далее вновь EXIT MENU.

## GAME

В этом разделе задаются команды и игроки, а также сохраняется и загружается отложенная игра.

PLAY MATCH - этой опцией открывается очередной матч. Сначала Вы знакомитесь с составом играющих команд, а затем, после нажатия "FIRE", переходите непосредственно к игре. В случае, если Ваша команда не участвует в очередном матче, Вы можете посмотреть, как играют команды, управляемые компьютером. Если Вы не желаете наблюдать за этим матчем, то в разделе "OPTIONS" нужно выключить режим "VIEW C Vs C". Перед игрой с участием Вашей команды Вы можете переключить управление с Кемпстон-джойстика на Синклер-джойстик и наоборот клавишей "SPACE".

ARRANGE FRIENDLY - (товарищеская встреча) - Вы можете назначить товарищескую встречу с любой командой из предлагаемого списка.

PICK TEAM - (перестановки в команде) - при выборе этой опции Вы можете производить тактические перестановки в команде, которая выбрана в опции "TEAM" (см. ниже)

POSTPONE MATCH - (отложить встречу) - при желании Вы можете перенести очередную встречу на более позднее время.

START CUP - (открытие кубковых встреч) - выбирая эту опцию Вы сможете участвовать только в кубковых матчах.

START LEAGUE - (открытие чемпионата) - здесь Вы участвуете только в играх на первенство лиги (встречи турнирные, а не кубковые).

START SEASON - (открытие сезона) - в этом случае Вы играете по расписанию чемпионата и Кубка лиги.

| Options                           | Colours | Game                | Displays |     |     |     |      |
|-----------------------------------|---------|---------------------|----------|-----|-----|-----|------|
| ENGLAND                           |         | Played by: COMPUTER |          |     |     |     |      |
|                                   |         | Spd                 | Def      | Att | Fit | Mat | Gl's |
| MILLER                            | 1       | Goalkeeper          | 99       | 0   | 0   | 0   | 0    |
| SMITH                             | 2       | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| PEARSON                           | 4       | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| ROBINSON                          | 7       | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| BROWN                             | 8       | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| ALDERSON                          | 9       | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| PALMER                            | 3       | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| HARDY                             | 5       | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| JACKSON                           | 6       | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| GREEN                             | 10      | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| JONES                             | 11      | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| WILSON                            | 12      | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| ROBINS                            | 14      | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| BLACK                             |         | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| CARTER                            |         | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| INNES                             |         | 000 000 000         | 99       | 0   | 0   | 0   | 0    |
| NEXT MATCH -> ASL LEAGUE - WEEK 1 |         |                     |          |     |     |     |      |
| HOLLAND vs ENGLAND                |         |                     |          |     |     |     |      |

EDIT TEAM - (редактирование команды) - войдя в редактор, Вы можете отредактировать команду, выбранную в опции "TEAM". После нажатия "FIRE", в левом верхнем углу Вы увидите название команды, а в правом - имя тренера. В начальном состоянии все команды управляются компьютером, и поэтому в правом верхнем углу будет написано "PLAYED by: COMPUTER". Ниже дается список игроков с краткой характеристикой каждого. Например:

- ROBINSON - фамилия игрока
- 7 - номер игрока
- SPD - скорость
- DEF - уровень игры в защите
- ATT - уровень игры в атаке
- FIT - степень готовности к мячу - высшая степень 99
- MAT - количество сыгранных игр
- GLS - количество забитых мячей.

Характеристики SPD, DEF, ATT заданы графически: один синий, два белых кружка - низкий уровень; два синих, один белый - средний уровень; три синих кружка - высокий уровень. Например: SPD - два синих кружка; DEF - три синих кружка; ATT - один синий кружок.

Это означает, что данный игрок обладает средней скоростью, высоким уровнем игры в защите и низким уровнем в нападении, следовательно, это защитник, и его вряд ли стоит ставить в нападение.

В игре местоположение футболиста на поле задается его номером:

со 2 по 5 - защитники, с 6 по 8 - средняя линия, с 9 по 11 - атака, 12 и 14 - запасные игроки, 1 - вратарь (не редактируется и не переставляется)

Управляя курсором, Вы можете отредактировать любую из восьми команд по Вашему желанию. Редактируются название команды, фамилии игроков и их характеристики.

ВНИМАНИЕ! Компьютером управляются только те команды, у которых в редакторе в строке "PLAYED BY:" впечатаны восемь букв - COMPUTER. В противном случае за команду играет Вы. Названия команд могут быть любыми.

Подтвердив свою редакцию команды, Вы возвращаетесь в подменю "GAME"

TEAM - (команда) - Вы можете выбрать любую из восьми команд, предназначенных к редакции или к тактической перестановке игроков.

SAVE TEAM - сохранить на ленте результаты редактирования команды.

LOAD TEAM - загрузка с ленты ранее сохраненной команды.

SAVE ALL - сохранить всю игру.

LOAD ALL - загрузить всю игру.

EXIT MENU - выход в основное меню.

## DISPLAYS

В этом разделе находится вся справочная информация и статистика игр.

SHOW TEAMS - (показать все команды) - компьютер дает список команд и фамилии тренеров.

SHOW PLAYERS - (показать игроков) - компьютер выводит на экран фамилии игроков команды, заданной в опции "TEAM".

TEAM - (выбор команды).

SHOW FIXTURES - (показать предстоящие матчи)

SHOW RESULTS - (показать результаты матчей)

SHOW TABLE - (показать таблицу чемпионата)

BY TEAM - (на команду) - включив эту опцию, Вы после запроса о предстоящих матчах увидите список игр одной команды на весь сезон.

BY WEEK - (на неделю) - аналогично предыдущей опции, но для просмотра всех игр на предстоящей неделе.

WEEK - (номер недели)

EXIT MENU - (выход)

Если у Вас еще нет опыта в этой игре, то Вам не обойтись без тренировки. После загрузки игры и ее настройки, включите в разделе "OPTIONS" опцию "PRACTICE". Опцию "VIEW C vs C" выключите, затем войдите в редактор и, выбрав любую команду, впишите в строчку "PLAYED BY:" свое имя. После этого в опции "ARRANGE FRIENDLY" назначьте товарищескую встречу. Затем "PLAY MATCH" - и Вы уже на тренировке. Компьютер не будет Вам сопротивляться, и Вы сможете отработать удары по воротам и технику ведения мяча. После тренировки Вы можете перейти к чемпионату ("START SEASON").

Во время игр на первенство Лиги Вы не сможете отредактировать свою команду или загрузить ее с ленты. Опции "EDIT TEAM" и "LOAD TEAM" в разделе "GAME" Вам уже будут недоступны.

#### ЗАМЕНА ИГРОКА:

Во время игры Вы можете заменить одного или двух игроков. Замена производится следующим образом: когда мяч уйдет за боковую линию (OUT), нужно нажать "CAPS SHIFT" - игра остановится (режим пауза). Затем клавиша "B" - и Вы переходите в основное меню. Здесь Вы выбираете раздел "GAME".

В опции "TEAM" установите название своей команды. Перейдите к опции "PICK TEAM". С помощью курсора выберите игрока, которого следует заменить и переставьте его на место игрока из запаса. Затем курсор отводится на позицию "GAME", подтверждается произведенная замена и Вы вновь возвращаетесь к опции "PLAY MATCH". После появления картинки матча нажмите "CAPS SHIFT", и Вы можете продолжать игру.

Прервать игру Вы можете клавишей "T".

Есть в программе и такой "жучок": в режиме паузы (CAPS SHIFT) нажать "V", а затем "SPACE" и вновь "CAPS SHIFT" для выхода из паузы, и Вы сразу переходите ко второму тайму. Если же такое переключение было произведено во втором тайме, то игра заканчивается. Таким образом, Вы можете обеспечить себе беспроигрышный вариант игры на любых уровнях. Однако настоящим бойцам этот "жучок" не потребуется.

#### УПРАВЛЕНИЕ ФУТБОЛИСТАМИ:

Управление каждым игроком является двухступенчатым. Во время ведения мяча Вы управляете футболистом как обычно, но нажав "FIRE", Вы автоматически переходите к выбору направления удара и траектории полета мяча. Удар наносится после того, как Вы отпустите клавишу "FIRE". Сила удара зависит от продолжительности нажатия на клавишу "FIRE".

#### УДАР НИЗОМ:

Положение джойстика вперед по ходу футболиста, затем нажать и отпустить "FIRE".

#### УДАР ВЕРХОМ:

Положение джойстика вперед по ходу футболиста, затем "FIRE", не отпуская "FIRE" перевод джойстика в противоположное положение и только после этого Вы отпускаете "FIRE".

#### ПАС ПЯТКОЙ:

Положение джойстика вперед по ходу футболиста, затем "FIRE", не отпуская "FIRE"

перевод джойстика в противоположное положение, затем вновь вперед по ходу и только после этого отпустить "FIRE".

При установке в опции "KICK DIRECTIONS" раздела "OPTIONS" цифры "5" Вы добавите к прямым ударам диагональные передачи.

Научившись виртуозно управлять своими футболистами, Вы начнете получать истинное удовольствие от этой игры. Занять первое место в чемпионате и выиграть Кубок Лиги, при условии, что компьютер играет в полную силу (10 уровень), будет очень трудно. Этого смогут добиться лишь те, кто играет в красивый комбинационный футбол и обладает качествами настоящего бойца. Удачи!

# СДЕЛАЙТЕ САМИ

## BLOCK BUSTER

Вниманию читателей, любящих осваивать свой компьютер в практической работе по набору и отладке игровых программ, сегодня мы предлагаем программу английского программиста Адама Булла "BLOCK BUSTER". Программа нами переведена на русский язык и оттестирована.

Игра представляет собой одну из версий распространенных в свое время игр типа "TIME BOMB". Ваше игровое поле состоит из прямоугольных блоков и некоторых специальных полей (премиальные поля выглядят как флажки, а незадействованные бомбы изображаются черепом со скрещенными костями).

Время от времени одна из бомб включается и начинает мигать. Ваша задача поставить своего сапера на это поле до того, как истечет время горения запала.

Сапер перемещается с помощью курсорных клавиш. Блок, по которому он прошел один раз исчезает. Второй раз наступить на эту же клетку смертельно опасно. Не менее опасно наступить на заминированную незадействованной бомбой клетку.

По мере того как все меньше и меньше блоков будет оставаться на экране, Вам все труднее будет подобраться к очередной бомбе. Здесь вам на помощь может прийти одна дополнительная возможность. С помощью клавиш "1" и "2" Вы можете сдвинуть влево или вправо ряд блоков, в котором находится Ваш герой.

Начиная со 2-го уровня в игре начнут попадаться ряды цветных блоков, которые не сдвигаются, да и количество бомб на каждом уровне будет все больше и больше.

Еще одна полезная возможность связана с циклической организацией игрового поля. Покинув экран вправо, Ваш герой появится слева, покинув экран сверху, он появится снизу и, соответственно, наоборот.

Теперь несколько слов о русификации. Она подробно описана нами в прошлом выпуске ZX-РЕВЮ при разборе программы "AFRICAN SEEDS". Здесь следует применить ту же самую технику. Более того, в наших последующих работах мы всегда будем ссылаться на нее, как на стандартную. Это позволит Вам не тратить время на русификацию и всегда использовать одни и тот же программный блок.

А теперь, когда есть определенность с вопросом русификации, можно переходить непосредственно к тексту программы "BLOCK BUSTER". Буквы с "А" по "Р" в кавычках в строках 60, 110, 160, 200, 1010, 1040, 4030, 4040, 4530, 6055 - это символы UDG-графики.

Основные переменные, использованные в программе, имеют следующее назначение:

- HI - лучший результат.
- SC - текущий счет игры.
- L - текущий уровень.
- L\$ - строковая переменная, в которой хранится количество оставшихся попыток.
- A,B - текущие координаты сапера.
- AA,BB - последние координаты сапера
- F1,F2 - координаты премиального флага.
- B1,B2 - координаты активизированной бомбы.
- M\$ - массив, в котором хранится конфигурация игрового поля.
- B0 - количество бомб на каждом уровне.
- T - время, оставшееся до взрыва бомбы.
- D - номер ряда для неперемещаемых блоков.
- X\$ - переменная, используемая в функции INKEY\$ и т. п.
- X, Y, Z - переменные общего назначения.



```

1 GO TO 10
2 CLEAR 64599: RANDOMIZE USR 15619: REM : LOAD "chr" CODE 64600
3 GO SUB 8: GO SUB 5000: GO TO 0
5 RANDOMIZE USR 15619: REM : ERASE "BUSTER"
6 RANDOMIZE USR 15619: REM : SAVE "BUSTER" LINE 2
7 STOP
8 POKE 23606,88: POKE 23607,251: RETURN: REM RUS
9 POKE 23606,0: POKE 23607,60 : RETURN : REM LAT
10 LET HI = 0: GO SUB 8
20 PAPER 7: BORDER 7: INK 0: BRIGHT 0: CLS : PRINT AT 11,0; "НАЖМИТЕ КЛАВИШУ <I>, ЕСЛИ НУЖНА
ИНСТРУКЦИЯ"
30 IF INKEY$<>" " THEN GO TO 30
40 IF INKEY$="" THEN GO TO 40
50 IF INKEY$="i" OR INKEY$="I" THEN GO SUB 6000
60 RANDOMIZE : LET SC=0: LET L=1: LET L$="PPP"
70 LET A=19: LET B=15: LET AA=A: LET BB=B
80 LET D=-1+(INT (RND*10)*2+2 AND L>2): GO SUB 4000
90 LET F1=Y: LET F2=Z: LET T=-1: LET B0=1: LET B1=T: LET B2=T
100 REM *MAIN GAME *
110 PRINT AT F1,F2; INK 6; "DE";AT F1+1,F2;"EG";AT A,B; INK 7;"LM"; AT A+1,B;"NQ"
120 IF AA<>A OR BB<>B THEN LET SC=SC+2: PRINT AT AA,BB;" ";AT AA+1,BB;" ": LET M$(AA,BB TO
BB+1)=" ": LET M$(AA+1,BB TO BB+1)=" ": IF M$(A,B)=" " AND (F1<>A OR F2<>B) THEN GO
TO 1000
130 LET AA=A: LET BB=B
140 IF A=F1 AND B=F2 THEN BEEP .02,20: BEEP .02,25: BEEP .02,30: GO SUB 4500: LET F1 = Y:
LET F2=Z: LET SC=SC+20
150 IF A=B1 AND B=B2 THEN LET B0=B0+1: BEEP .03,30: BEEP .03,25: BEEP .03,20: LET
SC=SC+L*15: LET T=-1: LET B1=T: LET B2=T
160 IF M$(A,B)="H" OR T=0 THEN GO TO 1000
170 IF B0>L*3+3 THEN GO TO 3000
180 PRINT AT 0,5; INK 7;SC
190 IF T>0 THEN LET T=T-1: PRINT AT B1,B2; FLASH 1; INK 6; PAPER 2;"**";AT B1+1,B2; "**"; AT
B1+ (INT(T/2)=T/2),B2;T: BEEP .01,T+10: GO TO 500
200 LET Y=M(B0): LET Z=INT (RND *15)*2+1: IF M$(Y,Z)="H" THEN LET T=50-L*4: LET B1=Y: LET
B2=Z: LET M$(Y,Z TO Z+1)="A": LET M$(Y+1,Z TO Z+1)="BC"
500 LET X$=INKEY$: IF X$<"5" THEN GO TO 540
510 LET A=A+(2 AND X$="6")-(2 AND X$="7"): LET B=B+(2 AND X$="8")-(2 AND X$="5")
520 LET A=A+(20 AND A<1)-(20 AND A> 19): LET B=B+(30 AND B<1)-(30 AND B>29)
530 GO TO 100
540 IF X$<>"1" AND X$<>"2" OR D=A THEN FOR X=1 TO 15: NEXT X: GO TO 100
600 IF X$="2" THEN GO TO 700
610 FOR X=0 TO 1: LET M$(A+X)=M$(A+X,3 TO )+M$(A+X): NEXT X
620 IF A=F1 THEN LET F2=F2-2
630 IF A=B1 THEN LET B2=B2-2
640 GO TO 800
700 FOR X=0 TO 1: LET M$(A+X)=M$(A+X,29 TO )+M$(A+X): NEXT X
710 IF A=F1 THEN LET F2=F2+2
720 IF A=B1 THEN LET B2=B2+2
800 FOR X=0 TO 1: PRINT AT A+X,1;M$(A+X): NEXT X
810 LET F2=F2+(30 AND F2=-1)-(30 AND F2=31)
820 LET B2=B2+(30 AND B2=-1)-(30 AND B2=31)
900 GO TO 100
1000 REM * LOSE A LIFE *
1010 FOR X=7 TO 0 STEP -.2: PRINT AT A,B; INK X;"LM"; AT A+1,B;"NQ": NEXT X

```

```

FOR X=50 TO 10 STEP -1: BEEP .01,X: BEEP .01,X+3: NEXT X
1030 LET L=L$(2 TO ): IF L$="" THEN PRINT AT 0,20;" ": GO TO 2000
1040 IF M$(A,B)="H" OR M$(A,B)="" THEN GO SUB 4500: LET A=Y: LET B=Z: LET AA=A: LET BB=B:
    IF M$(A,B)="" THEN GO TO 1040
1050 IF T=0 THEN LET B0=B0+1: LET T=-1: LET B1=T: LET B2=T
1060 GO SUB 4060: GO TO 170
2000 REM * GAME OVER *
2010 PRINT AT 9,0; PAPER 1; FLASH 1;"
    <<<<<<<< ИГРА ОКОНЧЕНА >>>>>>>>
    <<<<<<<< ЕЩЕ РАЗ (Y/N)? >>>>>>>>"
2020 IF SC>HI THEN LET HI=SC: GO SUB 4070
2030 FOR X=1 TO 50: BEEP .004, X
2040 IF INKEY$="y" OR INKEY$="Y" THEN GO TO 60
2050 IF INKEY$="n" OR INKEY$="N" THEN STOP
2060 NEXT X: GO TO 2030
3000 REM * NEXT LEVEL *
3010 LET L=L+(L<9): LET X$="СУПЕР ПРИЗ! == СУПЕР ПРИЗ! =="
3020 FOR X=1 TO 50: PRINT AT 21,0; INK 6; FLASH 1;X$: BEEP .01,X: LET X$=X$(2 TO )+X$
3030 LET X$=X$( TO 32): NEXT X
3040 LET SC=SC+55*L: GO TO 70
4000 REM * SET UP SCREEN *
4010 PAPER 0: INK 5: BORDER 0: BRIGHT 1: CLS
4020 DIM M$(20,30): DIM M(L*3+3)
4030 FOR X=1 TO 20 STEP 2: LET M$(X) = "■A■A■A■A■A■A■A■A■A■A■A■A": LET M$(X+1)=
    "BCBCBCBCBCBCBCBCBCBCBCBCBCBCBC": NEXT X
4040 FOR X=1 TO L*3+3: GO SUB 4500: LET M$(Y,Z TO Z+1) = "HI": LET M$(Y+1,Z TO Z+1) = "JK"
4050 LET M(X)=Y: NEXT X
4060 FOR X=1 TO 20: PRINT AT X,1; INK 5-(2 AND (X=D OR X=D+1));M$(X): NEXT X
4070 PRINT AT 0,0; BRIGHT 0; INK 7; "ЧЕТ:";SC;TAB 10;"МАКС. ";HI;TAB 20;L$;TAB 24;
    "УРОВЕНЬ";L
4500 REM * RANDOM POSITION *
4510 LET Y=INT (RND*10)*2+1
4520 LET Z=INT (RND*15)*2+1
4530 IF M$(Y,Z)="H" OR Y=19 AND Z=15 THEN GO TO 4510
4540 RETURN
5000 REM * U.D.G. DATA *
5010 POKE 23675,88: POKE 23676,255: FOR X=0 TO 127: READ Y: POKE USR "A"+X,Y: NEXT X: RETURN
5020 DATA 248,244,242,242,242,242,242,242,255,255,255,255,64,32
5030 DATA 31,0,242,242,242,242,10,6,254,0,0,0,15,63,127,127,127
5040 DATA 127,60,60,24,248,216,216,216,216,127,112,64,0,0,0,0,0
5050 DATA 216,248,24,24,24,24,24,0,96,231, 255,63,57,57,63,30
5060 DATA 12,206,254,248,56,56,248,240,30,15,28,60,247,227,96,0
5070 DATA 240,224,112,120,222,142,12,0,7,7,63,15,31,61,57,63,192
5080 DATA 192,248,224,240,120,56,248,123,92,79,39,12,88,112,0
5090 DATA 188,116,228,200,96,52,28,0,60,255,90,126,165,24,36,66
6000 CLS : PRINT "

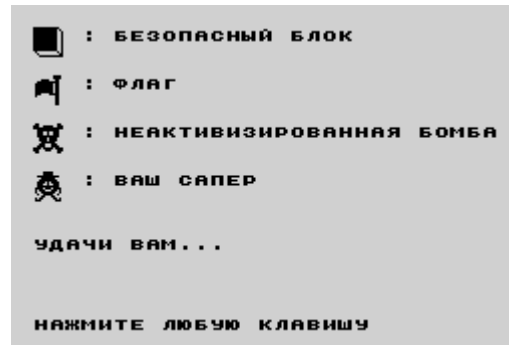
```



```

        КЛАВИШАМИ <1> ИЛИ <2>. ЭТО ПОМО-
        ЖЕТ ВАМ ПОДОЙТИ К БОМБЕ И Т.Д."
6040 PRINT "
        ЕСЛИ ВЫ БЕРЕТЕ ФЛАГ, ТО ВАМ
        ВЫДАЕТСЯ ПРЕМИЯ."
6050 GO SUB 9: PRINT "": GO SUB 8
6055 PRINT "
        ■ A : БЕЗОПАСНЫЙ БЛОК"" BC"" DE : ФЛАГ"" FG"" HI : НЕАКТИВИЗИРОВАННАЯ
        БОМБА"" JK"" LM : ВАШ САПЕР"" NO"" УДАЧИ ВАМ..."" NA : НАЖМИТЕ ЛЮБУЮ КЛАВИШУ"
6060 IF INKEY$="" THEN GO TO 6060
6070 RETURN

```



Набирая программу, можете периодически пользоваться RUN 5 для сохранения на ленте того, что уже набрано.

После автостарта программы со строки 2, происходит загрузка символьного набора "chr" CODE, далее происходит формирование символов UDG-графики при помощи подпрограммы 5000. После этого выполняется переход на начало программы. После остановки во время отладки, программу можно опять запускать командой RUN.

В строке 20 следует запрос на вывод инструкции. Если есть подтверждение для этого - нажата клавиша "I" - то происходит вывод инструкции при помощи подпрограммы 6000.

В строках 60-90 происходит задание исходных параметров игры. При помощи подпрограммы со строки 4000 выполняется прорисовка основного игрового поля. Задание случайных позиций для расстановки "мин" при этом происходит в строках с 4500.

Со строки 100 начинаются действия, связанные непосредственно с игрой. В строке 120 проверяется одно из условий, при которых происходит потеря "жизни". Это попадание "сапера" на пустое поле. Если оно выполняется, то программа переходит на строку 1000 для выполнения действий, связанных с этим.

Другое условие потери жизни - это попадание "сапера" на поле, где установлена неактивизированная мина. Оно проверяется в строке 160.

Строка 170 выполняет проверку того, все ли мины обезврежены. Если да, то программа переводит на строку 3000 - это переход на следующий уровень.

Опрос клавишей управления происходит начиная со строки 500. Здесь выполняется расчет новых координат "сапера", если нажата одна из курсорных клавиш. Если же нажата клавиша "1" или "2", то выполняются действия по сдвиганию горизонтального ряда блоков влево или вправо. При нажатии клавиши "1" это выполняется в строках с 610, а при нажатии "2" - в строках с 700.

В подпрограмме потери жизни со строки 1000 происходит уменьшение на 1 количества оставшихся жизней. Если их больше не осталось, что проверяется в строке 1030, то происходит переход на подпрограмму "конец игры", которая находится в строках с 2000. здесь вам будет предложено сыграть еще раз. Если да, то игра повторится сначала (при этом сохранится максимальный счет, набранный в предыдущей игре), если нет - то остановка и выход в Бейсик.

# Компьютерная новелла

Матвеев Ю. А.



## ПОСЛЕДНЯЯ СХВАТКА

(по игре Колина Свинбурна "Джо Блэйд-3")

Здание военно-промышленной компании было оцеплено. Десятки полицейских автомашин наглухо перекрывали все подъезды к небоскребу. На дальних подступах ожидали своего часа бронетранспортеры и танки, готовые по первому приказу выступить на штурм главной резиденции Кракса Бладфингера.

Мощные прожекторы, оперативно установленные на соседних домах, освещали осажденный небоскроб с первого по последний этаж. У главного входа в здание, перед запертыми изнутри бронированными дверями, стояло несколько человек в штатском. Они что-то оживленно обсуждали. Чуть в стороне от них плотный человек небольшого роста с мегафоном на плече отдавал какие-то распоряжения полицейским.

На противоположном конце площади, прилегавшей к небоскребу, в окружении автоматчиков, стоял лимузин мэра города. За темными стеклами автомобиля шли трудные переговоры по радиотелефону с самим Краксом. Мэр сидел на переднем сидении. Во влажных руках он держал телефонную трубку и молчал. Говорил, в основном, Кракс. Он держался настолько уверенно, что главе города казалось бесполезным перебивать его.

- ... Ваши идеи не стоят и ломаного гроша, - назидательно гремел в трубку Бладфингер. Подумайте лучше о том, что будет, когда за Вас возьмутся разъяренные мамы этих детей. Материнский инстинкт сильнее любой бомбы. - Кракс на секунду затих, как бы предлагая собеседнику оценить всю силу приведенных аргументов. Затем он продолжил: - Всем будет лучше, если мои условия будут выполнены. Раскинь мозгами и ты поймешь, что я предлагаю оптимальный вариант: мне нужен всего лишь самолет. Зато какое я оставляю наследство - целый небоскроб, набитый техникой...

- ...и толпами головорезов, - с иронией в голосе добавил мэр.

- Ну, это уже ваши проблемы, - ухмыльнулся Кракс. - Только дайте мне уйти без шума. Буду весьма благодарен, если вы пришлете на крышу вертолет, который доставит меня до аэропорта, а уж там Ваш самолет долго не задержится. Главное, чтобы хватило горючего для перелета через океан. Даю слово, что больше ничего у вас не попрошу и мы распрощаемся. О'кей? И давайте без самодеятельности. Только при этих условиях счастливые мамы увидят своих ненаглядных чад. В противном случае, сам знаешь, здание заминировано и будет уничтожено вместе с заложниками. Держу пари, моя техника не подведет. Вы согласны?

Мэр представил язвительную улыбку на лице Бладфингера и брезгливо поморщился:

- Нет, Кракс, я все равно найду способ выкурить тебя из твоей крепости. Советую сдаться, так ты, по крайней мере, сохранишь себе жизнь.

- Неужели? - захохотал Кракс. - А я не думал, что вы настолько кровожадны, чтобы смотреть на жареное человеческое мясо.

- Подонок, - прошептал мэр и выключил связь.

В воздухе пахло грозой. Ситуация осложнилась еще утром, когда мэр приказал арестовать Кракса Бладфингера. Но тот что-то пронюхал и успел захватить шестерых детей в качестве заложников. Они были похищены прямо из школы, когда там только закончились занятия. Операция оказалась на грани краха. Вечером того же дня Кракс с доброй сотней своих головорезов, забаррикадировался в здании военно-промышленной компании и теперь диктовал свои условия. Мэр всячески пытался оттянуть время, чтобы просчитать все возможные варианты. Военные, как всегда, предлагали штурм и обещали за десять минут захватить здание, но это было чревато человеческими жертвами. В такой ситуации можно было не сомневаться, что Кракс выполнит свои угрозы. К тому же мэр еще не вполне доверял военным, полагая, что в их рядах наверняка остались верные Краксу люди, несмотря на проведенную в последнее время чистку.

Президент военно-промышленной компании Кракс Бладфингер стал терять вес в обществе сразу после того, как вскрылись факты о применении им гипнотизаторов для давления на членов городского магистрата. С большим трудом, на свой страх и риск, мэр собрал вокруг себя людей, готовых опрокинуть трон, на котором восседал Кракс. Однако, в последний момент Бладфингер перехитрил всех. Но мэр знал, что безвыходных положений не бывает. Принимать условия Кракса он и не думал. Просто предстояло найти путь, ведущий к победе при новых правилах игры. Его мысли крутились вокруг одной и той же идеи, воплотить в жизнь которую мог только один человек. Его звали Джо Блэйд.

Мэр вызвал по радиации командира группы охраны:

- Вы знаете Джо Блэйда?

- Да, сэр, - ответил офицер.

- Он, скорее всего, сейчас дома. Езжайте туда, объясните ему ситуацию и доставьте его ко мне.

- Есть! - ответил охранник, и мэр выключил радио.

Не прошло и получаса, как дверь бронированного лимузина распахнулась и на заднее сидение бесшумно приземлился Джо. Он оставался верен своему стилю: военные камуфляжные штаны, майка-безрукавка и излюбленная кепка. Если бы не личное знакомство, мэр принял бы его за головореза из банды Кракса.

- Здравствуй, - мэр развернулся лицом к Джо и протянул через сиденье руку.

- Здравствуйте, - вяло поздоровался Блэйд.

Заметив недовольство собеседника, мэр сразу поинтересовался в чем дело.

- Ваша охрана забрала у меня распылитель, - ответил Джо, поглядывая в окошко автомобиля на офицера охраны, завладевшего его оружием.

- Это меры предосторожности, - сказал мэр, - но все поправимо. Секунду... - он распахнул дверь, вышел из машины и что-то сказал охраннику. Тот передал ружье Блейда мэру и куда-то удалился. Мэр вернулся в машину, но оружие сразу не отдал, а положил его на свободное сиденье водителя.

- Потерпи немного, - сказал он.

Джо кивнул и полез в карман брюк за сигаретами.

- Я надеюсь, что ты уже в курсе, - начал мэр.

- Еще бы, - кивнул Джо, затягиваясь сигаретой. - Весь город стоит на ушах.

- План такой. По моим данным, в этот небоскреб, - мэр ткнул пальцем в лобовое стекло, - ведет подземный ход из вентиляционной шахты, расположенной под зданием городского магистрата. Об этом подземном ходе знаю только я, несколько доверенных мне лиц и Кракс. Так как небоскреб полностью блокирован. Уйти оттуда можно лишь через подземный ход. Судя по всему, Кракс это и собирается сделать в случае невыполнения его условий. В то время, когда из-под обломков будут вытаскивать трупы, Кракс уже успеет пересечь границу.

- А почему бы ему не сделать это сейчас? - резонно заметил Джо.

- Понимаешь, - мэр наклонился ближе, - Он рискует и попытается ухватиться за все возможные варианты. Он, наверняка, подозревает, что я знаю об этом подземном ходе, хотя строился он еще в начале века и был строго засекречен от посторонних. Я сам наткнулся на

него случайно, когда прорабатывал варианты захвата Кракса. Мы тогда планировали взять его в здании магистрата и осматривали подвалы. Я думаю, что Кракс использует этот вариант только в самом крайнем случае, когда отступить будет некуда. Он же не может дать полной гарантии, что в конце хода его не поджидают мои люди.

- Логично, - Джо покосился на запищавший радиотелефон. Мэр снял трубку:

- Ну, что Вы решили? - ледяным голосом спросил Кракс.

- Ответ будет через час, - сухо сказал мэр. - Мне нужно время, чтобы отвести людей и технику и обсудить кое-какие детали, связанные с вертолетом.

- А самолет уже готов? - деланно удивился Кракс.

- Все будет, как только очистим площадь, - отрубил мэр.

- Хорошо, я жду еще час, - и Кракс отключился.

- Нас перебили... Ну так вот, - мэр города посмотрел на Джо Блэйда, сохранявшего железное спокойствие, - Я, как всегда, надеюсь только на тебя. Людей с твоим уровнем подготовки нет даже среди моей охраны и лучше тебя это никто не сделает. - Мэр улыбнулся. - В общем, я решил так: ты через подземный ход проникаешь в резиденцию Кракса и освобождаешь детей. Кстати, их как и тогда, шесть человек. Объясни им, что они могут смело прыгать из окон вниз. Мы натянем брезентовое пожарное полотно. Опытные люди будут ловить их.

Джо Блэйд прикинул высоту небоскреба:

- А если они все на двадцатых этажах? Не высоковато ли прыгать?

- Жить захочешь - прыгнешь, - ответил мэр. - Если будет возможность, пусть спустятся на лифте пониже. Одним словом, смотри по обстановке. Мы с тобой можем рассчитывать только на внезапность. А потом меня интересует Кракс. Я, конечно, не настаиваю, но если будет возможность схватить его, не упusti свой шанс. Я даю тебе право действовать от моего имени, а если удастся уничтожить здание компании, то буду очень признателен. Мне не хотелось бы чтобы от Кракса оставалось какое-либо наследство.

В стекло боковой двери автомобиля постучал офицер охраны. Мэр приоткрыл дверцу и тот протянул ему какой-то предмет в брезентовом чехле. Захлопнув дверь, глава города повернулся к Джо и передал сверток ему. Джо сдернул чехол. Он держал в руках новенький блестящий автомат пульсирующего действия с полным магазином. Он удивленно поднял глаза.

- Получше твоего ружья, - сказал мэр, довольный произведенным впечатлением. - Действует как распылитель, да к тому же не оставляет вредных продуктов распыления. Отменная вещь!

- Откуда это?

- Мы тоже кое-что умеем, - и мэр, включив рацию, стал отдавать распоряжения.

\* \* \*

Спустя полчаса Джо Блэйд, вооруженный автоматом, выслушав последние наставления мэра, нырнул в люк подземного хода. Путь к небоскребу оказался недлинным. Джо освещал себе дорогу карманным фонариком. Он шел по тесному и сырому кирпичному коридору лишь с одной мыслью: поскорее добраться до цели. Повернув за угол, неожиданно для себя он увидел выход.

Джо Блэйд осторожно толкнул широкую дверь и та, со скрипом, подалась. Проскользнув в щель, он осмотрелся. Подвал небоскреба показался огромным. Высокие кирпичные стены и бесконечный коридор. Впереди на бетонном полу аршинными буквами было написано: "НЕ ЗАГРОМОЖДАЙТЕ ПРОХОД". Внезапно он услышал чьи-то шаги и увидел двоих. Впереди шел охранник в военной форме с автоматом наперевес, а чуть сзади кто-то в штатском. Блэйд только успел разглядеть его бандитскую физиономию, как палец непроизвольно нажал на курок. Распылитель сработал безотказно. В воздухе повисли два белых облачка. Джо надавил плечом на дверь, через которую только что вошел. Она была заперта с той стороны. И этого следовало ожидать: путь к отступлению был уже предусмотрительно закрыт людьми мэра.

Джо прошел коридор, нашел какую-то невзрачную дверь и снова наткнулся на охрану.

Бой был короткий. Расчистив себе путь, он быстро осмотрел подсобное помещение. В одной из комнат он нашел ключ. Нигде не задерживаясь, Джо обошел весь подвал, но кроме кем-то подложенной мины и ручной гранаты больше ничего не обнаружил. Через мину Джо благополучно перепрыгнул, а вот гранату оставил себе. Потом в конце длинного коридора он увидел лифт. Джо Блэйд вошел в кабину и сердце его слегка екнуло: этот лифт мог подняться только до третьего этажа. (Названия этажей приводятся по правилам, принятым в английском языке. - прим. автора) Для того, чтобы попасть выше, требовался специальный пропуск. Но где его искать? Джо поднялся этажом выше. Людей здесь оказалось гораздо меньше, чем он ожидал. Праздно шатающиеся охранники встретились ему пару раз, когда он осматривал комнаты в поисках заложников. Джо предпочитал беречь патроны: если по коридору кто-то шел, он на время прятался за дверь.

Обойдя весь этаж, Джо пополнил карманы ключами. В одной из комнат он нашел взрывное устройство. Вероятно, Кракс приготовил его для уничтожения здания. Внимательно осмотрев устройство, Джо понял, что до полной комплектации не хватает магнитного детонатора, приводящего эту штуковину в действие. "Потом разберемся", - подумал Джо и на всякий случай запомнил расположение комнаты на этаже.

На выходе он столкнулся с каким-то бандитом, который заглянул в дверь. Джо отскочил в сторону и выстрелил. Проход освободился.

По пути к лифту Джо Блэйд снял со стены еще один ключ. Он понимал, что чем больше найдет ключей, тем больше сможет открыть дверей в этих лабиринтах. У лифта Джо отправил на небеса еще одного охранника и вошел в кабину. Здесь некоторое время можно было отдохнуть и обдумать дальнейшие действия.

Естественно, Кракс развел заложников по разным этажам и, чтобы добраться до них, было необходимо обзавестись пропусками на все этажи. Джо уже представил себе, сколько он потратит времени и сил, прежде чем сможет беспрепятственно перемещаться по небоскребу.

Наземный этаж Джо осмотрел быстро, благо здесь охраны было немного. Лишь один раз он на долю секунды замешкался, проходя через дверь из одного коридора в другой и за это поплатился: рябой мордоворот, вышедший навстречу, нанес чувствительный удар всем корпусом, да такой, что у Джо Блэйда поплыли круги перед глазами. Потеряв ориентацию, Джо успел пару раз нажать на курок, но промахнулся. Охранник опять двинулся на него. Не успев прийти в себя, Джо посмотрел вперед по коридору и увидел, как к нему движутся еще четыре охранника, двое из которых были вооружены. "Граната!" - мелькнуло у Джо, и он, подпрыгнув, метнул в толпу легкую ручную гранату, которую нашел в подвале. Когда рассеялся дым, он не увидел окровавленных и изуродованных тел - граната Кракса имела свойства распылителя. Охранники бесследно исчезли. Джо как вихрь промчался по коридору, поочередно заглядывая во все двери. В одной из комнат он заметил заложника.

Ребенок сидел на полу, связанный по рукам и ногам, с кляпом во рту. Глаза, полные слез, испуганно смотрели на Джо Блэйда.

- Свои, - тихо сказал Джо Блэйд и принялся освобождать ребенка от веревок.

Они обменялись несколькими фразами. Оказалось, что в рядах охраны царит паника и многие уже думают о побеге. Краем уха мальчишка слышал, что у большинства охранников автоматы не заряжены и носят они их только для показухи. Военная машина Бладфингера где-то дала сбой и ему, видимо, не удалось вооружить всех своих людей. Самые верные Краксу охранники находятся на верхних этажах, а внизу всю основную работу делают патрульные роботы.

Джо изложил вкратце план побега и мальчишка утвердительно кивнул.

- Будь осторожен. Это наземный этаж - высота небольшая. Так что не бойся прыгать в окно. Я тебя прикрою. - Хлопнув парня по плечу, Джо вышел в коридор. Здесь никого не было. Быстро проскочив несколько комнат, он заглянул в приоткрытую дверь в самом конце коридора. На полу у дальней стены лежал какой-то предмет. "Детонатор." - подумал Джо сразу, как только взял вещицу в руки. Теперь он мог зарядить одно взрывное устройство, найденное на нижнем наземном этаже. Однако он понимал, что таких взрывных устройств гораздо больше и для полного уничтожения здания необходимо было зарядить их все.

Следовательно, нужно постараться найти как можно больше детонаторов. С этой мыслью Джо Блэйд вышел из комнаты. Хлопнула дверь, и из соседней комнаты появился охранник. Увидев Джо Блэйда, охранник собирался что-то спросить, но тот без промедления нажал на курок. Облако еще не растворилось в воздухе, а Джо уже был во второй половине этажа. Здесь он собрал какие то компьютерные дискеты и на всякий случай прихватил их с собой.

Возвращаясь к лифту, Джо заглянул в последнюю комнату, оставшуюся неосмотренной на этом этаже. Он чуть было не споткнулся о мину, которая лежала прямо у входа, ибо в этот момент его внимание привлекла табличка с надписью "БЕЗОПАСНОСТЬ" над дверью в соседнее помещение. Джо осторожно толкнул дверь, но вдруг погас свет, раздался какой-то звук и перед глазами засветился шифрозамок с восьмью цифрами. Отступать было некуда. Сейчас здесь будет вся охрана, и тогда ему - конец. Секретный замок отводил ровно одну минуту на подбор кода. Счетчик лихо стал отстукивать секунды. Джо в панике беспорядочно нажимал кнопки на панели замка, надеясь на случайное совпадение цифр. С такой системой кода он встретился впервые. Нужной комбинации не получалось. Когда до срабатывания сигнализации оставалось около двадцати секунд, Джо понял систему работы кода. Взяв себя в руки, он начал методично подбирать нижний ряд цифр. В течение следующих десяти секунд ему удалось выставить правильный шифр. Раздался булькающий звук, включился свет и дверь распахнулась. Комната была пустыня. На письменном столе у окна стояла бутылка молока, а на тарелке лежал гамбургер. Видно, хозяин кабинета собирался перекусить, но был срочно куда-то вызван. Джо не упустил возможности подкрепиться. Усталость как рукой сняло. Вероятно, продукты содержали специальные вещества, влияющие на жизнедеятельность человеческого организма и восстанавливали силы в кратчайшие сроки. Кракс заботился о своем персонале.

Не прошло и десяти минут, а Джо уже заканчивал осмотр первого этажа. Здесь он отыскал еще два взрывных устройства, несколько ключей и магнитный детонатор. Зная, что с детонатором нельзя приближаться к динамиту ближе, чем на один метр, он перепрыгнул через взрывное устройство и направился к лифту.

Он расстрелял охранника, поджидавшего кабину и только сейчас заметил, что запас патронов на исходе. Сумеет ли он найти их для своего автомата в логове Кракса?

На втором этаже Джо долго не задержался. В одной из многочисленных комнат он нашел еще одного связанного заложника и, переговорив с ним, заставил его быстро покинуть здание через окно. В соседних комнатах Джо нашел взрывные устройства. Запомнив расположение комнат, Джо Блэйд перешел во вторую половину этажа. Несколько ключей и граната были весьма кстати, поскольку патронов для своего автомата он пока так и не нашел.

Нажимая в лифте кнопку с цифрой "три", Джо с горечью думал о пропуске на следующие этажи, которого у него не было. Когда двери лифта распахнулись, в глаза ударил яркий голубой неоновый свет. Джо Блэйд оказался в другом мире. После полумрака подвальных помещений обстановка казалась ему почти домашней.

Не успел Джо выйти из лифта, как к нему бросился охранник, требуя предъявить какие-то документы. Блэйд не стал вступать в спор, да и патронов оставалось совсем немного. Он рванул в противоположную от охранника сторону и, пробежав несколько метров, толкнул первую попавшуюся дверь. И только тут Джо Блэйд заметил прикрепленные к стенам видеокамеры, которые следили за каждым его шагом. Похоже, его засекли. Только бы успеть зарядить автомат. Он выбежал в коридор.

Отперев ключом застекленную дверь, Джо оказался в небольшом кабинете. Патронов здесь не было, однако какая-то коробка, с нарисованной на ней театральной маской, сразу привлекла его внимание. Джо обнаружил в коробке полный комплект обмундирования для службы охраны. За несколько секунд Джо Блэйд преобразился до неузнаваемости и стал как две капли воды похож на человека из охраны Бладфингера. Единственное, о чем жалел Джо, так это о недолговечности этой формы, ибо в ткань материи были вшиты волокна микродатчиков, которые были способны улавливать опознавательные сигналы от передающих устройств, вживленных в мочку уха каждого сотрудника компании. Джо не знал всех тонкостей работы этой системы, однако понимал, что чужака, одевшего форму, легко

запеленговать. Поэтому носить униформу более двух-трех минут было опасно. Неспеша Джо Блэйд вышел в коридор и осмотрелся. Мимо прошли четыре сотрудника безопасности, не обратив на него никакого внимания.

За то время, пока Джо чувствовал себя достаточно спокойно в форме охранника, ему удалось тщательно осмотреть весь этаж. Уже не веря в удачу, он обнаружил ящик с патронами нужной системы. Затем, где-то в глубине длинного кабинета, нашелся и пропуск на верхние этажи, а чуть позже, осматривая компьютерный зал, Джо решил поэкспериментировать с дискетами.

Включив компьютер, он быстро отыскал нужный пароль и, войдя в меню, запросил машину об этажах, на которых находятся заложники. "6 этаж", - выдал компьютер и отключился. Блэйд понял, что по одной информационной дискете он может получить только один ответ, однако в его запасе было несколько дискет, поэтому уходить он не торопился. Компьютер выдавал информацию по любому вопросу. Джо удалось узнать все, что ему было нужно. Записав ответы компьютера, он вернулся к лифту. Мимо, тяжело дыша, пробежал здоровенный охранник. Джо Блэйд на всякий случай отвернулся лицом к лифту. Вероятно, охрана уже заподозрила что-то неладное, хотя он старался не оставлять свидетелей.

На четвертом этаже Джо пришлось столкнуться сразу с двумя патрульными роботами. Спасительная форма осталась в лифте: Джо не мог более рисковать. Роботы чем-то походили на пылесосы - невысокая пластиковая платформа на небольших гусеницах венчалась продолговатой металлической шеей с телекамерой, заменявшей голову. Несмотря на свою неуклюжесть, роботы передвигались довольно быстро и Блэйду пришлось разложить их на атомы, как только те стремительно покатались на него. "Интересно, - подумал Джо. - как там сейчас мэр? Удалось ли ему оттянуть время?"

\* \* \*

- Я уничтожу всех, - кричала трубка голосом Кракса. - Предупреждаю: еще полчаса - и вы будете жалеть, что не выполнили мои требования.

Мэр посмотрел на часы: Джо Блэйд должен успеть. Двух заложников уже передали заплаканным матерям, но еще четыре семьи ждали своей очереди.

- Я отдал приказ об отводе техники, - сурово произнес мэр. - Вертолет готовится к вылету, так что вы можете успокоиться.

Кракс ничего не ответил и отключился.

Площадь перед зданием военно-промышленной компании заметно опустела. Под стенами небоскреба с туго натянутыми полотнищами стояли в напряженной ожидании пожарные и несколько полицейских. Жителей прилегающих домов спешно эвакуировали. Вокруг осажденного здания образовали километровую запретную зону. Танки и бронетранспортеры отодвинулись вглубь города.

Мэр, не покидая своего лимузина, наблюдал за этой картиной. "Только бы Джо успел найти всех заложников," - думал он. Подземный ход надежно перекрыт. О Блэйде он тоже позаботился: вертолет, который якобы предназначался для Кракса, на самом деле до последнего будет ожидать Джо на крыше небоскреба. Пилот уже получил необходимые инструкции. Что еще?

Мэр уже в который раз просчитывал возможные варианты. Запищал радиотелефон:

- Слушаю.

- Еще в одном танке обнаружена бомба, - хрипло отозвался голос на другом конце.

- Жертв нет? - нахмурился мэр.

- Нет.

- Номер танка?

- Ноль восемь.

- Держите в курсе, - отрезал глава города и положил трубку.

Сохранять конспирацию было необходимо, поскольку Кракс мог прослушивать любые каналы связи. После этого короткого разговора у мэра полегчало на душе - Джо освободил третьего заложника, который находился на восьмом этаже. "Молодец парень!" - мэр

откинулся на спинку сидения и облегченно вздохнул.

\* \* \*

Его загнали в угол. Сразу три охранника приближались к нему. "Будь проклят этот этаж!", - подумал Джо, доставая из кармана гранату. Патроны закончились еще на выходе из лифта, когда пришлось распылить почти целый взвод охраны и пару свирепых роботов.

Вспышка на время ослепила, но медлить было нельзя. Джо бросился по коридору обратно к лифту, спустился двумя этажами ниже за патронами, которые он оставил про запас. Однако спокойствия не прибавилось, ибо боеприпасов хватило лишь на половину магазина.

Чувствуя легкую слабость в ногах, Джо поднялся на одиннадцатый этаж. Здесь, по сообщению компьютера, можно было найти патроны и легко перекусить.

Помещения службы безопасности Джо мог осматривать уже без лишней нервозности, связанной с подборкой кода. Ему помогали специальные пластиковые пропуска с магнитным покрытием, которые он нашел, осматривая какие-то служебные помещения. Места заточения оставшихся троих детей Джо уже знал. Ближайший заложник находился на пятнадцатом этаже. Пропуск у Блэйда был до девятнадцатого. Полностью вооружившись и приведя себя в порядок, Джо Блэйд вошел в лифт и решительно нажал кнопку пятнадцатого этажа.

Через несколько секунд кабина лифта остановилась и распахнулись автоматические двери. В холле Джо осмотрелся. Вдоль стен стояли кожаные диваны и кресла, полы покрывали мягкие ковровые дорожки, а на стенах висели дорогие картины. Посмотрев налево, он увидел человека в военной форме, который быстро шел к нему. Джо развернулся и побежал в противоположную сторону.

- Стой! Стрелять буду! - крикнул охранник, вскидывая автомат.

Джо Блэйд видел впереди себя спасительную дверь. Оставалось сделать несколько шагов, но вслед прогремели выстрелы. Правое плечо обожгло, кровь тонкой струйкой побежала по руке. Джо выстрелил в ответ и не промахнулся, но вслед за охранником, жужжа гусеницами, двигался патрульный робот. Блэйд нырнул в кабинет и, придерживая ногой дверь, отдышался.

- Черт возьми, здесь ребята стрелять умеют, - процедил он, зажав рукой рану.

Вдруг в углу комнаты кто-то всхлипнул. Джо обернулся и увидел ребенка, сидевшего на полу за креслом.

"Ну, что ж, - успокаивал себя Джо, развязывая веревки, - это хорошая компенсация за царапину от пули".

Парень оказался на редкость смышленным и Джо Блэйду не пришлось долго объяснять, что к чему. Прикрывая мальчишку, он проводил его до лифта и объяснил, где лучше выпрыгнуть из окна. Сам же пересек коридор и открыл массивную дверь, ведущую в просторный кабинет. Здесь под кадкой с фикусом лежал брошенный кем-то бронежилет. "Во-время," - подумал Джо, застегивая замки пластиковой защиты. Чувствуя себя намного спокойнее, он прошел коридор до лифта и поднялся на девятнадцатый этаж. Тут он должен был найти пропуск на верхние этажи небоскреба и заложника. Компьютер не обманул. Пять минут понадобилось на то, чтобы отыскать в одной из комнат плачущего ребенка, а по пути прихватить с собой сразу два пропуска на лифт. Но Джо Блэйду пришлось расстаться с бронежилетом, который долго носить было так же опасно, как и форму.

Спустившись на шесть этажей ниже, Джо основательно подкрепился предусмотрительно оставленным здесь гамбургером и зарядил магазин распылителя. Машинально проверив карманы, он еще раз убедился в сохранности шести детонаторов, которые он собрал. С их помощью Джо собирался уничтожить здание. Замысел был прост. Оставалось только найти последнего ребенка.

И тут его словно ударило током: а Кракс? Взорвать здание и до конца своих дней думать, что этот паук погиб вместе со всеми? А если он успеет уйти? Ну уж нет!

Он решительно направился к лифту и спустился на третий этаж. В компьютерном зале никого не было. У Джо оставалась последняя неиспользованная дискета. Блэйд включил



компьютер. Короткий сигнал. Минута на подбор кода. По запросу о местонахождении Кракса компьютер выдал: двадцать пятый этаж.

- Что ж, увидимся! - вслух сказал Джо и направился к выходу.

Но сначала Джо Блэйд поднялся на двадцать первый этаж за заложником. Все прошло, как по маслу. Ему повезло - ребенок сидел в коридоре недалеко от лифта. Освобождая его от веревок, Джо думал о Краксе...

\* \* \*

- Я не буду ждать ни минуты, - Кракс нервно теребил пуговицу на манжете пиджака.

- Вертолет уже подают на крышу, - тихо урчала трубка голосом мэра.

- В таком случае я могу идти туда?

- Пожалуй...

- Я не верю Вам! - сорвался в крик Кракс. - Вы лжете!

Но он уже кричал в пустоту. Мэр выключил связь. Кракс Бладфингер швырнул трубку Радиотелефона на пол, поднялся с кресла и, быстро накинув плащ, распахнул дверь в коридор. Неожиданно тишину офиса нарушил треск автоматных очередей и грохот взорвавшейся гранаты. На секунду все стихло. Кракс в панике отшатнулся к стене, когда увидел широкоплечего мускулистого человека в прокопченной майке и камуфляжных штанах, который появился из облака дыма.

- Поговорим, - спросил Джо Блэйд, приближаясь. Дуло распылителя смотрело прямо в лицо Краксу.

Бладфингер рванул по коридору. Надо заметить, что бегал он неплохо, несмотря на свой возраст. Джо бросился за ним.

Кракс знал все входы и выходы и прекрасно ориентировался в своих апартаментах. Джо еле успевал за ним. В одном из коридоров пришлось воспользоваться гранатой, чтобы убрать людей Кракса. "Уйдет, гад," - думал Джо Блэйд, глядя на то, как Бладфингер распахивает очередную дверь. И тут сдали нервы. Джо нажал на курок. Но что это? Фигура Кракса, чуть задержавшись, исчезла за дверью. "Промахнулся?" - с удивлением подумал Джо, врываясь следом.

Это была ловушка. Кракс прятался за спинами сразу трех вооруженных охранников. Джо метнул гранату, но его успели задеть. Скрипя зубами от боли, он надвигался на Кракса, который, как ни странно, был цел и невредим. Отступить Бладфингеру было некуда. Прижавшись спиной к стене, с мертвенно бледным лицом, он с ужасом смотрел на супермена.

- Знаем мы твои штучки, - тяжело дыша сказал Джо, хватая за грудки президента компании. Кракс не сопротивлялся. Вдруг что-то хрустнуло под железными пальцами Блэйда и тихо запищал миниатюрный приборчик, спрятанный во внутреннем кармане Бладфингера. Рванув плащ на груди Кракса, Джо вытащил на свет маленькую черную коробочку с цифровым индикатором. Ох уж этот любитель головоломок! Блэйду некогда было рассуждать о том, что произойдет, если он не успеет подобрать код. Посеребший Кракс медленно сполз по стенке, когда Джо удалось выставить нужную комбинацию цифр. Защитное поле вокруг Бладфингера было разрушено.

- Ну, вот и все, - выдохнул Джо. Взвалив Кракса на спину, он донес его до ближайшего окна. - Высоковато будет. Но тебя внизу примут в лучшем виде.

Джо Блэйд распахнул окно. Внизу на площади по-прежнему стояли люди с натянутым брезентом, который большим серым квадратом выделялся на фоне темного асфальта.

- Принимайте! - выкрикнул в ночь Джо и спихнул испуганного Кракса с подоконника. Несколько секунд тишины. Обмякшее тело потерявшего сознание Бладфингера удачно приземлилось по центру полотнища и, пару раз спружинив, осталось лежать без движения.

Расстреляв в коридоре охрану захваченного президента, Джо Блэйд пошел к лифту.

Зарядить взрывные устройства на нижних этажах было делом техники. Охрана почти не сопротивлялась. Самые умные наверняка уже сдались в руки правосудия и в здании остались лишь фанатики. Зарядив последнее шестое взрывное устройство, Джо вошел в кабину лифта и нажал кнопку двадцать восьмого этажа. "Мэр что-то говорил о вертолете, - думал Джо, - если, конечно, меня дождутся". Ему необходимо было отыскать путь на крышу

небоскреба. Время поджимало. Пришлось пробежать по коридору, не обращая внимания на крики людей Кракса и автоматные выстрелы. Он открывал двери одну за другой. Где же лестница на крышу? Вскоре, в кабинете службы безопасности он увидел то, что искал. Через несколько секунд Джо уже был на крыше. Какие-то отчаянные автоматчики встретились у него на пути, но на этот раз он не пожалел патронов. На другом конце крыши его ожидал вертолет. Спотыкаясь и еле передвигая ноги, Джо Блэйд приближался к цели. Вслед трещали автоматы последних фанатиков, а впереди он видел блестящий бронированный борт вертолета, отражавший тусклый свет неполной луны. Еще один шаг и он спасен.

Джо пришел в себя, когда упал в мягкое кресло рядом с пилотом, и вертолет, резко поднявшись вверх, стал удаляться от небоскреба. Площадь внизу была пустынна. Джо посмотрел на пилота. Тот хитро подмигнул и подтолкнул плечом уставшего супермена. Блэйд вяло улыбнулся. Пилот включил рацию:

- Все по плану, - доложил он. - Иду на базу.

Джо посмотрел на часы: час ночи. Ему сильно хотелось спать.

- Как же тебе удалось? - спросил пилот и посмотрел на Блэйда.

- Сам не знаю, - вздохнул Джо и закрыл глаза.

Вертолет шел на посадку.

ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ МУЗЕЙ  
И ФИРМА "ИНФОРКОМ"  
учреждают  
АССОЦИАЦИЮ ПОЛЬЗОВАТЕЛЕЙ ДОМАШНИХ КОМПЬЮТЕРОВ  
а также  
"ФОНД ЛИТЕРАТУРЫ, ОПИСАНИЙ И ПРОГРАММ" ("ФЛОП")

|                               |                                                                                                    |
|-------------------------------|----------------------------------------------------------------------------------------------------|
| Задачи АССОЦИАЦИИ<br>и ФОНДА: | Оказание содействия в приобретении компьютеров,<br>периферии, программ и литературы всем желающим. |
|-------------------------------|----------------------------------------------------------------------------------------------------|

ОСНОВНЫЕ НАПРАВЛЕНИЯ ДЕЯТЕЛЬНОСТИ:

1. Открыта **ВЫСТАВКА - КОНСУЛЬТАЦИЯ - ПРОДАЖА** учебно-игровых домашних компьютеров. Выставка работает ежедневно, кроме понедельника с 11 до 17 часов.

Адрес: Москва, Новая площадь, д. 3/4, подъезд 1, Государственный Политехнический музей. Проезд до станции метро "Лубянка", "Китай-город".

Приглашаем всех производителей домашних  
компьютеров и периферии к участию в  
**ВЫСТАВКЕ - ПРОДАЖЕ**

2. Авторам программ открывается возможность для реализации собственных разработок.

В качестве первого шага создается система защиты авторских прав. Авторы, желающие заявить о своих правах должны выслать два описания программы (краткое для публикации и подробное для архива) со своими данными в адрес "ИНФОРКОМа".

Описания программ, заслуживающих внимание будут опубликованы в "ZX-РЕВЮ", для чего открывается новый раздел "АВТОРСКАЯ ПРОГРАММА". Если в течение трех месяцев со дня публикации краткого описания в АССОЦИАЦИЮ не поступят претензии на авторское право со стороны иных лиц, авторство подтверждается сертификатом и АССОЦИАЦИЯ предложит автору заключить договор на реализацию его программы.

АССОЦИАЦИЯ также планирует выдавать заказа на разработку обучающих программ.

Кроме авторов программ приглашаются к работе авторы технических разработок, доработок, усовершенствований и т.п.

3. АССОЦИАЦИЯ берет на себя функции по тестированию технических разработок в области компьютеров и других устройств, развивающих интеллект.

4. АССОЦИАЦИЯ приглашает к сотрудничеству всех заинтересованных и инициативных лиц, способных заняться организацией региональных представительств АССОЦИАЦИИ и заняться коммерческой деятельностью по распространению компьютеров, периферии, литературы.

5. Продажа литературы и подписка на "ZX-РЕВЮ" производится на корпункте "ИНФОРКОМа" по адресу: Москва, "Новый Арбат", д.2, 19-е отделение связи (1-ый этаж операционного зала). Ежедневно, кроме воскресенья с 10 до 17 часов, перерыв с 14 по 15 часов.

Для переписки с АССОЦИАЦИЕЙ просим пока временно использовать адрес "ИНФОРКОМа".

121019, Москва, а/я 16. На конверте делать пометку "ФЛОП". Эти письма будут переданы в АССОЦИАЦИЮ.

## ИНФОРКОМ СООБЩАЕТ

1. Подготовлен к печати 2-ой том четырехтомника по графике компьютера "ZX-Spectrum". Книга называется "Прикладная графика". Рассмотрена графика растровая, векторная и блочная. Показаны приемы организации трехмерной графики в растровом и векторном исполнении, особо рассмотрены актуальные вопросы борьбы с "клэшингом цветовых атрибутов" и работа с нестандартными шрифтами размером менее, чем 8\*8 и более, чем 8\*8. Книга является логическим продолжением тома 1 "Элементарная графика", но может быть использована и самостоятельно. Ее основные положения и выводы войдут в т. 3 "Динамическая графика" в качестве исходных.

2. Начата продажа лицензии на издание "Прикладной графики" в регионах. Книга поставляется на дискете 1,2 Мб (формат MS DOS). Принимаются заявки. Продолжается продажа лицензий на т. 1 "Элементарная графика" и на учебник по программированию в машинных кодах (бывший трехтомник, расширенный и дополненный). Напишите нам или дайте телеграмму. Мы сообщим контактный телефон для переговоров.

3. Выход книги из типографии для розничной продажи ожидается в октябре 1993 г.

4. С 1-го сентября 1993 г. начинается прием подписки на "ZX-РЕВЮ-94" и на четырехтомник "Графики СПЕКТРУМА". Желающие получить бланк-заказ подписных изданий могут прислать заполненный конверт с наклеенными почтовыми марками на сумму 6 руб. для России (30 руб. для стран СНГ) по нашему адресу: 121019, Москва, а/я 16



"ИНФОРКОМ" 121019, Москва, Г-19, а/я 16

Вниманию читателей!

Мы продолжаем публиковать адреса пунктов, в которых можно приобрести наши материалы.

- г. МОСКВА, ул. Новый Арбат, д.2. 19-е отделение связи, 1-ый этаж операционного зала.
- г. МОСКВА, радиорынок в Митино, проезд до ст. метро Тушинская или поездом до пл. Трикотажная (Рижское направл.). Суббота, воскр, 9 - 14.
- г. МОСКВА, радиорынок в Царицыно, проезд до ст. метро Царицыно. Суббота, воскр, 9 - 14.
- г. БЕЛГОРОД, магазин "РАДИОТОВАРЫ", ул. Ленина, 32.
- г. ВОРОНЕЖ, Студия компьютерных игр SAN-SAN. Магазин-салон "ЭЛЕКТРОНИКА", тел. 14-00-73.
- г. ДНЕПРОПЕТРОВСК, ул. Шевченко, 34. Фирма "ЭКОС".
- г. ЕКАТЕРИНБУРГ, магазин "СПЕКТРУМ", Главный проспект, 99.
- г. КЕМЕРОВО, магазин "ТЕХНИЧЕСКАЯ КНИГА", ул. Весенняя, 24.
- г. КЕМЕРОВО, магазин "ОРБИТА", пр. Ленина, 133.
- г. КИРОВ, "Дом науки и техн.", Магазин-салон "МАРС", ул. Производственная, д. 27.
- г. КРАСНОЯРСК, радиорынок, проезд до ост. "Затон", субб., вскр.
- г. НИЖ. НОВГОРОД, ИМА "Ф-ПЛЮС", магазин "ФОТОЛЮБИТЕЛЬ", ул. Горького, 146.
- г. ОРЕНБУРГ, магазин "ВОЕННАЯ КНИГА", ул. Советская.
- г. РЫБИНСК, ул. Гоголя, 1. ТТЦ "ГНОМ".
- г. ЧЕБОКСАРЫ, маг. "ЭКСПРЕСС", НПК Фирма "НОВА", ул. Привокзальная, д.6.

Вниманию дистрибуторов! Адреса наших оптовых покупателей публикуются бесплатно.

## СПЕКТРУМ В ШКОЛЕ

Дорогие друзья!

В прошлом месяце мы "играючи" научились отыскивать решение сложных алгебраических уравнений численным методом простой итерации.

Это, как Вы понимаете, не единственный возможный численный метод и сегодня мы попробуем с Вами сыграть в другую компьютерную игру, которая тоже позволяет находить корни алгебраических уравнений. Назовем эту игру "Вилка". Имеется в виду "артиллерийская вилка", хотя можно было бы назвать игру и "Перелет - Недолет".

Если Вы не знаете, что такое артиллерийская вилка, то лучше бы Вам этого и не знать, потому что ситуация, когда Вы в нее попадаете, очень неприятна. Представьте себе теплое летнее утро в полевом блиндаже. Хорошо позавтракав, противник начинает лениво высевать дневную норму снарядов и мин по вашим позициям. Бах!!! Снаряд разорвался где-то далеко впереди. "Недолет", - ухмыляетесь Вы и продолжаете спокойно уплетать консервы. Бах!!! Одинокий снаряд разрывается где-то в сотне метров сзади. "Так, теперь перелет".

Вот теперь Вы должны понимать, что попали в "вилку" и, если у противника есть хоть какие-то мозги в голове, то следующим выстрелом он накроет Ваш блиндаж, будьте покойны. Двумя выстрелами он "пристрелял" орудие и теперь у него есть математические таблицы, прицельные приспособления и большой опыт, чтобы в третий раз не промахнуться. Более того, пристреляв одно орудие и использовав эти данные и для других орудий, он может ударить теперь и всей батареей. Бросайте свою тушенку и двигайте в окоп, - хорошо,

если Вы не пожалели сил и заранее откопали запасную линию, где и отсидитесь во время арналета.

Вот так же можно попасть "выстрелом" и в корень алгебраического уравнения, если предварительно сделать несколько "выстрелов".

Причем Вам совершенно все равно, если первые "выстрелы" будут очень неточными. Единственное, что должно быть сделано - в одном выстреле должен быть "перелет", а в другом "недолет", иначе "вилка" не получится.

Рассмотрим простое уравнение:

$$3x^2 + 5x - 22 = 0$$

Сделаем первый выстрел - пусть  $x=0$ . Подставим  $x$  в уравнение, оно естественно не сойдется и возникнет "невязка"  $d = -22$ .

Раз невязка имеет знак "минус", будем считать, что это "недолет".

Сделаем второй выстрел  $x=5$ . Теперь  $d=78$  - это большой перелет. То, что мы хотели, произошло. Образовалась артиллерийская вилка:

$x=0$  - мало

$x=5$  - много.

Если бы она не образовалась, нам пришлось бы еще "пострелять". А теперь все просто - ударим посередине  $x=2.5$ . Подсчитаем невязку  $d = 9.25$  - это опять перелет, но уже ближе к цели, мы получили новую вилку

$x=0$  - мало

$x=2.5$  - много.

Вновь ударим посередине  $x=1.25$ . Невязка  $d$  равна примерно  $-12$  -это недолет.

$x=1.25$  – мало

$x=2.5$  - много.

Следующий удар нанесем по пункту  $1.875$ . И так далее. Точное решение  $x=2$  и с каждым "выстрелом" мы все ближе к нему подбираемся.

Вы, конечно поняли, что на каждом шаге мы берем "недолет" и "перелет" и стреляем посередине, потом опеределаем, что это было - "недолет" или "перелет" и от предыдущего шага берем противоположное значение, чтобы всегда оставалась "вилка".

Теперь нам осталось только поручить все эти дела компьютеру.

1 REM здесь Вы введете свое уравнение.

10 DEF FN A(X) - .....

20 LET b=.01 : REM допустимое значение "невязки"

29 REM теперь вводим пару значений  $x$  так, чтобы получить "вилку"

30 INPUT "Xmin?":x0

40 INPUT "Xmax?":x1

49 REM рассчитаем невязку

50 LET d0 = FN A(x0)

60 LET d1 = FN A(x1)

70 LET print = 500 80 GO SUB print

90 IF SIGN (d0) = SIGN (d1) THEN GO TO 30: REM "вилка" не получилась"

100 LET x= (x0 + x1)/2: REM "вилка получилась"

110 LET d = FN A(x)

120 IF ABS(d)<b THEN GO TO 200

130 IF SGN (d) = SGN (d0) THEN GO TO 170

140 LET x1 = x: LET d1 = d

150 GO SUB print

160 GO TO 100

170 LET x0 = x: LET d0 = d

180 GO SUB print

190 GO TO 100

200 PRINT "!!! x= ";x,"d= ";d

210 STOP

500 PRINT "x0= ";x0,"d0= ";d0

510 PRINT "x1= ";x1,"d1= ";d1

520 RETURN

Этот метод определения корней в сложных алгебраических уравнениях тоже известен

не одну сотню лет и называется методом Ньютона. Нам с Вами просто очень повезло, что ко времени появления первых ЭВМ уже были известны десятки и сотни подобных численных методов, легко адаптируемых для компьютеров. Впрочем, может быть ЭВМ именно потому и появились, что потребовалось упростить работу сотен расчетчиков, сидевших месяцами с арифмометром над баллистическими таблицами для расчета траекторий полета первых ракет.

# SINCLAIR LOGO

(Продолжение) Начало см. на с. 69-74, 90-96.

## ГЛАВА 5 ПОВТОРЯЮЩИЕСЯ ОПЕРАЦИИ

### Условия.

Под "условием" в ЛОГО понимается операция, которая дает в результате логическое значение TRUE ("истина") или FALSE ("ложь"), в ответ на вопрос типа: "Это число равно другому?"

В ЛОГИКЕ, а также в языках программирования, имеющих дело с обработкой списков (ЛИСП, ПРОЛОГ, ЛОГО), такие операции называют ПРЕДИКАТами. Может быть поэтому в ЛОГО все примитивы, относящиеся к этой категории, оканчиваются на "-Р"

Попробуйте набрать:

```
PRINT EQUALP 5 5
```

- получите в результате:

```
TRUE
```

Здесь условие EQUALP проверяет, равны ли два параметра. Если да, то выдается результат TRUE, иначе выдается результат FALSE. Например:

```
PRINT EQUALP 5 7
```

```
FALSE
```

### Математические условия.

Как и в других математических операциях здесь используют символы, которые размещают между двумя аргументами (так называемая индексная форма записи). Это следующие символы:

> - больше

< - меньше

= - равно.

Примеры:

```
PRINT 1 > 2
```

```
FALSE
```

```
PRINT 5 < 8
```

```
TRUE
```

```
MAKE "FACT 5-5
```

```
PRINT :FACT
```

```
TRUE
```

Знак "=" используется для сравнения двух объектов любого типа. Это могут быть числа, слова, списки. Эта операция дает значение TRUE тогда и только тогда, когда оба объекта одинаковы. Эта операция тождественна операции EQUALP, которую мы рассмотрели ранее. Разница, как видите, только в форме записи, поскольку EQUALP записывается не "между", а "перед" двумя параметрами (префиксная форма записи).

```
MAKE "SHOPPING[BREAD BUTTER MILK STAMPS]
```

```
PRINT :SHOPPING =[BREAD BUTTER MILK STAMPS]
```

```
TRUE
```

```
MAKE "GIRL "MARY
```

```
PRINT :GIRL=MARY
```

```
TRUE
```

```
MAKE "NAME "MARY
```

```
PRINT :GIRL = :NAME
```

```
TRUE
```



```
PRINT (7-5)-2 TRUE
PRINT EQUALP(7-5) 2
TRUE
```

Условия очень часто используют совместно с оператором IF. IF после себя требует два, а иногда и три операнда. Первый - это условие, а второй - список инструкций, который должен быть выполнен, если условие справедливо. Например:

```
MAKE "TODAY "FRIDAY
IF :TODAY = "FRIDAY [PRINT "TGIF]
TGIF
```

Если оператор IF имеет три операнда, то в третьем операнде записывается список инструкций, которые должны выполняться, если условие "ложно".

```
MAKE "TODAY "MONDAY
IF :TODAY= "FRIDAY [PRINT "TGIF]
[PRINT [NOT FRIDAY YET]]
NOT FRIDAY YET
```

Как и конструкции REPEAT, команда IF может иметь достаточно длинную запись и ее приходится переносить со строки на строку.

А теперь мы рассмотрим некоторые примитивы, являющиеся условиями, и обсудим, в каких случаях они могут применяться.

### **MEMBERP**

Проверяет, является ли первый операнд (а он может быть числом, словом или списком) частью второго (который должен быть списком).

```
TO CHECKSHOP: STUFF
MAKE "SHOPPING[BREAD BUTTER MILK STAMPS]
IF MEMBERP: STAFF:SHOPPING
[PRINT [GOT IT]] [PRINT [OH DEAR FORGOTTEN IT]]
END
```

```
CHECKSHOP "MILK
GOT IT
CHECKSHOP "BACON
OH DEAR FORGOTTEN IT
```

### **NUMBERP**

Эта программа выдает значение TRUE, если ее аргумент является числом. В противном случае выдается FALSE.

Вам нередко приходится проверять, является ли то, что ввел пользователь, числом, прежде, чем приступить к обработке этого ввода. Согласитесь, нелепо задать вопрос о том, сколько Вам лет ("How old are you?") и пытаться как-то обсчитывать полученный результат, если в ответе было:

NOT VERY MUCH ("не очень много")

```
TO CHECKNUM
MAKE "ANSWER FIRST READLIST
IN NUMBERP: ANSWER
[PRINT [[THANK YOU]]]
[PRINT [[I WANTED A NUMBER]]]
END
```

Обратите внимание на то, что мы используем FIRST READLIST. Если бы мы использовали только READLIST, то результат был бы списком. Даже если в нем было бы

всего одно число, например 5, он все равно распознавался бы только как список, а не как число.

### **WORDP**

Эта операция дает в результате TRUE, если ее операндом является слово. Вспомните, что числа тоже являются словами, поэтому они тоже дают в результате TRUE.

```
PRINT WORDP 3
TRUE
```

```
PRINT WORDP "WISE
TRUE
```

```
PRINT WORDP [MEN]
FALSE
```

В последнем случае операнд является списком, в который входит только одно слово, но все же это список, а не слово и потому в результате выдано FALSE.

### **LISTP**

Дает в результате TRUE, если операнд является списком.

```
PRINT LISTP:SHOPPING
TRUE
```

В практической работе Вам придется много раз писать процедуры для обработки списков. Очень хорошей практикой будет устраивать в начале каждой процедуры проверку: "А является ли входной параметр списком?"

### **EMPTYP**

Проверяет, не пустой ли операнд. Ищет пустое слово " или пустой список [] и если то, либо другое найдено, выдает TRUE. Этот оператор имеет очень широкое применение. Длинные списки анализируются путем разбиения их на элементы и последовательной обработки элементов одного за другим, пока список не станет пустым. Точно также слова анализируют, разбивая их на символы. Мы в дальнейшем будем очень часто использовать выражение вида:

```
IF EMPTYP : ALIST[STOP]
```

для того, чтобы останавливать процесс.

### **NAMEP**

Дает в результате TRUE, если операндом является имя чего-либо, т.е. этому имени соответствует некоторое содержание.

```
MAKE "MONTH "APRIL
PRINT NAMEP "MONTH
TRUE
PRINT NAMEP "FRED
FALSE
```

## **Рекурсия.**

Мы уже рассматривали такие процедуры, как

```
TO TRI
FORWARD 50
RIGHT 120
TRI
END
```

Запустите эту процедуру. Сама она не остановится. Придется для этого использовать

BREAK (CAPS SHIFT + SPACE). Самое важное в этой процедуре - это то, что она входит в бесконечный цикл повторов без использования повтора REPEAT, а вызовом самой себя TRI. Итак, TRI исполняет первых две инструкции - FORWARD 50 и RIGHT 120, после чего снова вызывается TRI, которая исполняет первые две инструкции и т.д. и т.п.

Рекуррентные процедуры могут иметь входные параметры, как и любые другие процедуры. В простейшем виде эти входные параметры всегда одни и те же при каждом вызове процедуры. Мы можем легко изменить процедуру TRI и задать любой многоугольник.

```
TO POLY :SIDE :ANGLE
  FORWARD :SIDE
  RIGHT :ANGLE
  POLY :SIDE :ANGLE
END
```

Эта процедура отличается от той, которую мы рассматривали в главе 2 (с инструкцией REPEAT). Она может изображать некоторые такие вещи, которые той не под силу. Попробуйте:

```
POLY 50 144
```

Когда процедура вызывает саму себя, это называется рекурсией. В ЛОГО это основной способ организации повторяющихся процессов.

В процедурах TRI и POLY мы использовали очень простой вид рекурсии, когда процедура не знает, где же следует остановиться.

Рекурсия используется не только для того, чтобы организовать повторяющиеся действия. Здесь могут быть достигнуты и некоторые дополнительные цели.

Предположим, что Вы покупаете тюбик зубной пасты и, вскрыв его, находите внутри ваучер, дающий право на снижение цены на 20 центов при покупке очередного тюбика, по сравнению с этим. Спрашивается: "Когда Вам надо идти за следующим тюбиком?" Ответ: "Немедленно". Ведь и в нем будет такой же ваучер и т.д. и скоро Вы завалите квартиру почти бесплатной зубной пастой.

Аналогичный прием может быть использован и в ЛОГО:

```
TO ADDON :NUMBER
  PRINT :NUMBER
  ADDON :NUMBER+1
END
```

```
ADDON 1
1
2
3
...
и т. д.
```

При первом вызове ADDON печатается 1 и во второй вызов передается параметр 2 и т. д.

Рассмотрим еще один пример. Предположим, что нам надо найти сумму чисел от 1 до N, где число N начинается с единицы и непрерывно возрастает. Может быть, мы примем решение, что процесс надо остановить, когда сумма превысит одну тысячу. И тогда мы вводим команду STOP, служащую для того, чтобы останавливать текущую процедуру. В принципе, STOP действует так же, как и END, но может размещаться в любом месте процедуры.

Обратите внимание на то, что STOP останавливает только ту процедуру, внутри которой находится. Если эта процедура вызывалась другой, то та не останавливается и продолжает работать.

```
TO ADDUP :NUMBER
```

```

IF :TOTAL >1000 [STOP]
MAKE "TOTAL :TOTAL + :NUMBER
PRINT :TOTAL
ADDUP :NUMBER+1
END
MAKE "TOTAL 0

```

```

ADDUP 1

```

```

1

```

```

3

```

```

6

```

```

10

```

```

15

```

```

...

```

```

и т. д.

```

Обратите внимание на то, что нам приходится выставлять исходное значение TOTAL, равное нулю, вне процедуры, поскольку нам не надо, чтобы TOTAL принимало бы нулевое значение всякий раз, как происходит исполнение процедуры.

Рекурсия очень часто оказывается очень удобной при работе со списками. Например, поэлементная печать списка, которую мы рассматривали в гл. 1, может быть сделана более элегантно с помощью рекурсии. Например, так:

```

TO LBL :ALIST
PRINT FIRST :ALIST
LBL BUTFIRST :ALIST
END

```

Все очень просто. Процедура распечатывает содержимое начала списка, а затем точно так же поступает с его остатком. Когда же она закончит свою работу? Когда список станет пустым.

```

TO LBL :ALIST
IF EMPTY? :ALIST [STOP]
PRINT FIRST :ALIST
LBL BUTFURST :ALIST
END

```

Проверьте эту процедуру на каком-нибудь списке, а затем попробуйте ее видоизменить так, чтобы она печатала список в обратном порядке.

Конечно, в некоторых случаях рекуррентные выражения могут быть весьма сложными, но простейшие применения рекурсии поистине просты. Кроме бесконечных рекурсий (которые на самом деле тоже бесконечными не являются, ведь при каждом вызове процедуры ЛОГО фиксирует факт этого вызова в оперативной памяти и рано или поздно эта память исчерпывается) существуют еще две возможности. В приведенных примерах процедура начинается с проверки. В языках, предназначенных для структурного программирования, таких как Паскаль или скажем, в развитой версии БЕЙСИКа Бета-БЕЙСИК, эта конструкция называется циклом WHILE. В ЛОГО это выглядит так:

```

TO WHILE <параметр>
IF <условие> [STOP]
... <тело процедуры> ...
WHILE <параметр>
END

```

В этой процедуре <тело> не исполняется, если <условие> принимает значение TRUE. В частности, эта процедура вообще не исполняется, если исходное значение параметра таково, что <условие> справедливо.

Другой прием состоит в проверке условия в конце серии повторяющихся инструкций. Это соответствует команде UNTIL Паскаля или БЕТА БЕЙСИКа.

```

TO UNTIL <параметр>
... <тело процедуры>
IF <условие> [UNTIL<параметр>]
END

```

Здесь тело процедуры всегда выполняется до проверки условия, так что по крайней мере один раз процедура будет исполнена.

Очень часто, когда мы используем рекурсию для организации повторяющихся вычислений, нам необходимо создать стартовую процедуру, которая задаст начальные значения переменных, сделает первый вызов рекуррентной процедуры и сможет обработать полученный из нее результат. В нижеприведенном примере, который должен отыскать сумму, последовательности введенных с клавиатуры чисел, процедура DOTOTAL сначала устанавливает ноль в качестве текущей суммы, а затем вызывает рекуррентную процедуру TOTALUP. Когда работа закончена, DOTOTAL печатает итоговый результат.

```

TO DOTOTAL
MAKE "TOTAL 0
TOTALUP
PRINT SENTENCE [THE TOTAL IS] :TOTAL
END

```

Процедура TOTALUP демонстрирует использование процедуры NUMBERP для того, чтобы проверять те числа, которые пользователь вводит с клавиатуры. Она позволяет пользователю ввести числовую последовательность и закончить ее словом END или любым другим нечисловым вводом. Процедура организована по схеме UNTIL, рассмотренной выше. Единственное отличие состоит в том, что вводится дополнительная инструкция

```
MAKE "TOTAL :TOTAL + :NUM
```

если условие справедливо (TRUE). Эту инструкцию можно было бы поместить и в начало процедуры, но только в том случае, если бы первое значение NUM задавалось бы вышележащей процедурой DOTOTAL.

```

TO TOTALUP
PRINT [TYPE IN A NUMBER]
MAKE "NUM FIRST READLIST
IF NUMBERP :NUM[MAKE "TOTAL :TOTAL +NUM TOTALUP]
END

```

Когда с клавиатуры будет введено какое-то слово, не являющееся числом, вызов TOTALUP будет прерван. В итоге, будет остановлен и весь процесс и управление будет передано в вышележащую процедуру в точку после первого вызова TOTALUP.

### **Генератор случайной последовательности.**

Идеи, изложенные в этой главе, можно применить к генерации случайных последовательностей. Рассматривайте это как первый шаг к программам, с помощью которых компьютер будет писать стихи.

Давайте рассмотрим сначала как ЛОГО может выдавать случайные числа. Вообще-то эти числа рассчитываются и потому не являются вполне случайными, но маловероятно, что Вы заметите их неслучайность.

Так, RANDOM 5, например, выдаст одно число из последовательности 0,1,2,3,4. Точно также RANDOM 6 даст число от 0 до 5.

Попробуйте:

```
REPEAT 50 [PRINT RANDOM 6]
```

- и посмотрите, что из этого выйдет.

Если Вам надо иметь случайное число не из диапазона [0;4], а из диапазона [1;5], то Вы тоже можете воспользоваться оператором RANDOM 5, но к полученному результату надо прибавить единицу. Это делается так:

```
1 + RANDOM 5.
```

Обратите внимание на то, что запись RANDOM 5+1, будет неправильной, поскольку сначала будет исполнено сложение 5 + 1, а затем процедура RANDOM с параметром 6.

Давайте используем эту процедуру для того, чтобы извлечь из списка случайный элемент. Прежде всего введем переменную ITEMNO - случайное целое число от 1 до значения COUNT, а затем используем эту переменную для выбора элемента списка.

```
TO CHOOSE :ALIST
  MAKE "ITEMNO 1 + RANDOM COUNT :ALIST
  MAKE "CHOICE ITEM :ITEMNO :ALIST
END
```

Попробуйте эту процедуру с каким-либо списком. Например, для списка PRESENT:

```
CHOOSE :PRESENT
```

Для того, чтобы генерировать предложения, нам нужна сначала какая-то канва, какая-то скелетная схема, например:

```
THE SMALL DOG QUICKLY BITES THE CARELESS MAN
```

Рассмотрим структуру этого предложения, учитывая, что не все наши читатели могут быть знакомы с английской грамматикой.

- Артикль "the" может иметь в качестве альтернативы артикль "a".
- Прилагательное SMALL (маленький, -ая) может быть заменено на иное, например: [BIG BROWN SPOTTED CARELESS]
- Существительное "DOG" (собака) является таким же объектом, как и [MAN TREE TABLE KENNEL]
- Наречие QUICKLY (быстро) описывает, как происходит событие. Вместо него можно употребить [VICIOUSLY SLOWLY CAREFULLY HAPPILY]
- Глагол BITES (кусает) - указывает на происходящее действие точно так же, как и глаголы: [LIKES CHASES CARRIES HIDES].
- Еще один артикль "THE".
- Еще одно прилагательное "CARELESS" (беспечный)
- Еще одно существительное "MAN" (человек, мужчина).

Итак, если мы введем сокращения:

AST - артикль;

ADJ - прилагательное;

ADV - наречие;

VER - глагол;

KOU - существительное, то получим следующую скелетную схему:

```
[AKT ADJ NOU ADV VER ART ADJ SOU]
```

Приравняем этот список переменной PATTERN, а с именами AKT, ADJ, NOU и пр. свяжем свои списки глаголов, наречий, существительных и т.п. Так, например, список с именем "ADJ" будет содержать:

```
[SMALL BIG BROWN SPOTTED CARELESS]
```

Теперь мы можем конструировать случайные предложения.

```

TO RANDSENTENCE
  MAKE "PATTERN [ART ADJ NOU VER ART ADJ NOU]
  MAKE "ART [SMALL BIG BROWN SPOTTED CARELESS]
  MAKE "KOU [MAK TREE TABLE KENNEL DOG]
  MAKE "ADV [QUICKLY VICIOUSLY SLOWLY CAREFULLY HAPPILY]
  MAKE "VER [BITES LIKES CHASES CARRIES HIDES]
  REPEAT 50 [DORANDOM]
END

TO DORANDOM
  MAKE "OUTLINE []
  FOLLOW :PATTERN
  PRINT "OUTLINE
  PRINT "
END

```

Процедура FOLLOW сначала проверяет, не является ли ее параметр (а это список, в котором закодирована скелетная схема нашей фразы) пустым. Если это так, происходит выход из процедуры. Иначе, берется первый элемент скелета, а он является именем списка слов и из этого списка делается случайный выбор. Выбранное слово добавляется к собираемому предложению OUTLINE.

```

TO FOLLOW :APATTERN
  IF EMPTY? :APATTERN [STOP]
  MAKE "THISWORD FIRST :APATTERN
  CHOOSE THING :THISWORD
  MAKE "OUTLINE SENTENCE :OUTLINE :CHOICE
  FOLLOW BUTFIRST :APATTERN
END

```

## ГЛАВА 6. ЧЕРЕПАШЬЯ ГРАФИКА-2.

Идеи, рассмотренные в двух предыдущих главах, могут быть проиллюстрированы с помощью "черепашьей графики". Возьмем процедуру RESPI из гл. 3 и модифицируем ее так, чтобы "черепашка" в одних случаях поворачивала бы направо, а в других - налево. Сделаем это, обеспечив дополнительный входной параметр, который будет списком чисел, значений COUNT, для которых "черепашка" должна поворачивать влево. Когда COUNT является числом, входящим в этот список, выполняется поворот налево, иначе - направо. Вставлять всю эту логику в оператор REPEAT, по-видимому, слишком длинно, поэтому проще создать отдельную процедуру:

```

TO RESPI2 :SIZE :ANGLE :MAX :ALIST
  MAKE "COUNT 1
  REPEAT :MAX [SUBSPI]
  RESPI2 :SIZE :ANGLE :MAX :ALIST
END

TO SUBSPI
  FORWARD :SIZE * :COUNT
  IF MEMBERP :COUNT :ALIST [LEFT :ANGLE] [RIGHT :ANGLE]
  MAKE "COUNT :COUNT+1
END

```

Проверьте эту процедуру со следующими параметрами:

```

RESPI2 5 120 6 [1 3]
RESPI2 5 90 11 [3 4 5]

```

На рис. 1 показан пример работы процедуры.

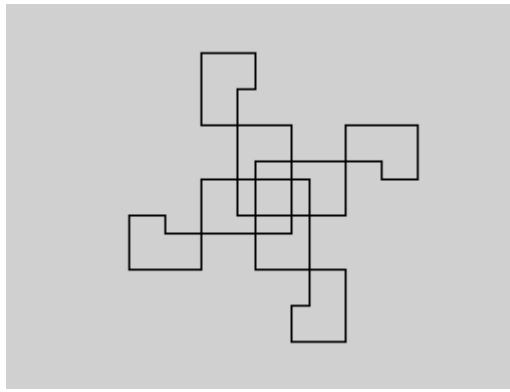


Рис.1 RESPI2 5 90 7 [1 2 3 4]

Мы рассмотрели выше способ остановки рекуррентных процедур. Теперь интересно было бы рассмотреть, как можно остановить рекуррентную процедуру, изображающую замкнутую диаграмму (такую, в которой "черепашка" возвращается в ту точку, с которой начинала работу. Прежде всего, нам надо узнать, вернулась ли "черепашка" в точку старта. В ЛОГО есть процедура-примитив, которая вычисляет координаты "черепашки" в любой момент времени. Это процедура `position`, она выдает список из двух чисел. Первое число - горизонтальная координата, отмеренная от центра экрана; второе число - вертикальная координата, она тоже измеряется от центра экрана и для верхней его половины является положительной, а для нижней - отрицательной.

ЛОГО имеет три команды для перемещения "черепашки" в указанное положение:

`SETX 50` - переводит ее на вертикаль, имеющую координату X, равную 50.

`SETY -50` - переводит ее на горизонталь, имеющую координату Y, равную -50.

`SETPOS [-20 30]` - устанавливает "черепашку" в точку, с указанными в списке координатами.

Если перо "черепашки" опущено, то при исполнении каждой из этих трех операций будет прорисована линия от исходной точки к конечной.

В главе 3 мы говорили о том, что знак "минус" перед числом имеет двойственное значение. ЛОГО применяет простое правило к оценке знака "минус", если он находится в списке. Если перед знаком есть пробел или нет ничего и число следует непосредственно за знаком (без пробела), то знак «минус» рассматривается, как часть числа и это означает, что число отрицательное. Во всех прочих случаях этот знак интерпретируется, как самостоятельное слово.

Итак, `[- 10 10]` - это три слова, точно так же и `[10 - 10]` и `[10-10]` и ни одна из этих записей не может быть использована в качестве входного параметра для процедуры `SETPOS`.

С другой стороны, `[10 -10]` и `[-10 10]` - состоят из двух слов и могут быть использованы при вызове `SETPOS`.

Следует внимательно относиться и к другим процедурам, принимающим список входных параметров в виде чисел, среди которых могут быть и отрицательные.

Кроме процедур, позволяющих выставить X и Y-координаты "черепашки", существуют две функции `XCOR` и `YCOR`, позволяющие определить и выдать текущие координаты X и Y. Эти функции не могут выдать эти координаты на экран, но могут предоставлять их другим процедурам для обработки.

Если мы хотим определить момент, когда "черепашка" сделает замкнутый контур, то кроме координат нас должно интересовать еще и направление, в котором она смотрит. Этот результат может быть получен с помощью функции `HEADING`. - она выдает результат от 0 до 360 (можете рассматривать его, как показание стрелки компаса).

Север - 0

Восток - 90

Юг - 180

Запад - 270



Таким образом, поворачивая "черепашку" направо, Вы увеличиваете значение HEADING, а поворачивая ее налево - уменьшаете его. Направление, в котором "смотрит" "черепашка" можно не только измерить, но и задать с помощью команды SETHEADING, после которой должно идти число от 0 до 360.

Вернемся к нашей проблеме определения момента, когда "черепашка" закончит замкнутый контур и пойдет по второму разу.

Нам надо зафиксировать координаты "черепашки" в исходной точке и направление ее движения. Для удобства мы можем собрать эти три числа в один список:

```
MAKE "STATE LPUT HEADING POSITION
```

А после этого мы можем регулярно проверять положение "черепашки", дабы уловить момент, когда она будет находиться в той же точке и смотреть при этом в том же направлении.

```
IF EQUALP :STATE LPUT HEADING POSITION [STOP]
```

Конечно, не следует делать такую проверку до того, как "черепашка" сделает первый шаг.

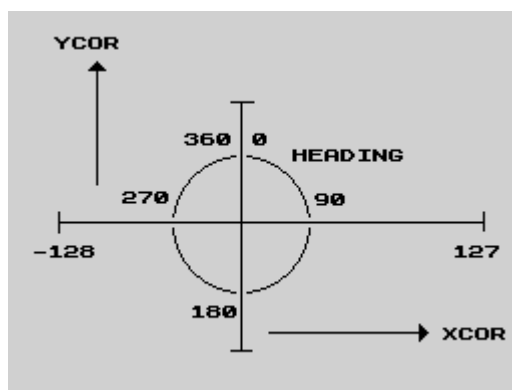


Рис. 2 Координаты "черепашки"

Можно использовать процедуры, задающие координаты X и Y и для того, чтобы рисовать на экране графики. Предположим, что у Вас есть список из двенадцати чисел (ежемесячные расходы) и его надо передать в качестве входного параметра в процедуру, которая нарисует на экране график. Пока для простоты изложения мы предположим, что все числа не превосходят 160 и не являются очень малыми, по сравнению с этим числом.

Сначала надо нарисовать пару осей. Начиная из левого нижнего угла из координаты [-125 -80] нарисуем горизонтальную линию с двенадцатью вертикальными метками, соответствующими 12 месяцам.

```
TO XAXIS
  PENUP
  SETPOS [-125 -80]
  PENDOWN
  SETHEADING 90
  FORWARD 5
  REPEAT 12 [FORWARD 20 SETY -82 SETY -80]
END
```

Вертикальные метки идут через каждые 20 шагов "черепашки", т.е. 20 шагов соответствуют одному календарному месяцу.

Аналогично рисуется и вторая ось. Ее тоже следовало бы разметить делениями шкалы, но поскольку они зависят от тех количеств, которые мы измеряем и тех единиц, которые мы применяем, это лучше делать для каждого конкретного случая по-своему и наш читатель сможет сделать это самостоятельно, по вкусу.

```
TO YAXIS
```

```

PENUP
SETPOS [120 -85]
PENDOWN
SETY 80
END

```

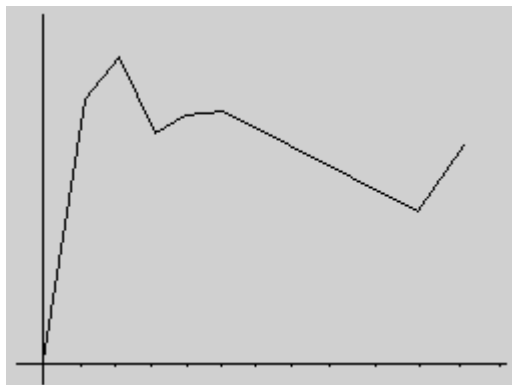


Рис. 3 График ежемесячных расходов.

Оси пересекаются в т.  $[-120 -80]$ , поэтому это и будет исходная точка нашего графика. Просмотрим список, используя переменную COUNT и для каждого значения нарисую линию, соединяющую предыдущую точку с текущей.

Поскольку исходная точка у нас находится не в нуле, а отстоит от него на 120 единиц влево и на 80 единиц вниз, мы должны будем соответственно вычесть 120 из горизонтальной координаты и 80 из вертикальной координаты каждой текущей точки.

```

TO PLOT
  MAKE "COUNT :COUNT + 1
  MAKE "XCOR :COUNT*20-120
  MAKE "YCOR ITEM :COUNT :VALUELIST-80
  SETPOS LIST :XCOR :YCOR
END

```

COUNT необходимо в начале работы выставлять в нуль с тем, чтобы когда процедура выполнялась бы в первый раз, это значение было бы равно единице. Необходимо также подготовить список VALUELIST значений, для которых должен строиться график.

В итоге мы получим главную процедуру:

```

TO GRAPH :VALUELIST
  XAXIS
  YAXIS
  SETPOS [-120 -80]
  PENDOWN
  MAKE "COUNT 0
  REPEAT 12[PLOT]
  PENUP
  SETPOS [-120 -80]
END

```

Если среди данных есть значения, большие чем 160, или весьма малые значения, то либо "черепашка" выйдет за пределы экрана, либо график будет плохо различим. В этих случаях возникает необходимость масштабировать данные. Для этого их умножают на какое-либо число, в результате чего получают график приемлемых размеров.

Итак, научились использовать рекурсию при построении графических фигур. Приведем в качестве примеров еще несколько фигур, при построении которых тоже используется рекурсия.

Во-первых, рассмотрим бесконечную кривую, которую называют "снежинкой".

Построим равносторонний треугольник. Разделим каждую сторону на три части и на среднем отрезке построим еще по равностороннему треугольнику и т.д. (Рис. 4)

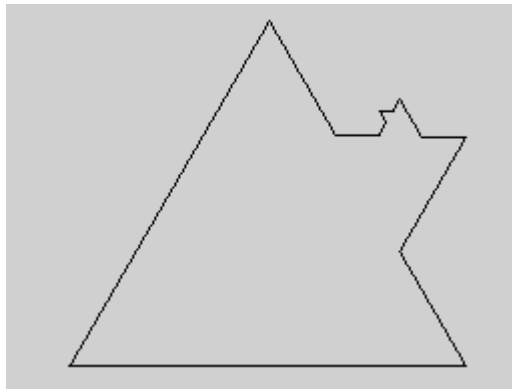


Рис. 4 "Снежинка"

Для рисования треугольника мы поступаем просто - три раза:

```
FORWARD :LENGTH  
RIGHT 120
```

Для того же, чтобы изобразить "снежинку", мы должны заменить процедуру FORWARD процедурой изображения треугольника с размером 1/3 предыдущего.

```
TO SNOWFLAKE :LENGTH  
  REPEAT 3[SNOWLINE :LENGTH RIGHT 120]  
END
```

Для того, чтобы нарисовать сторону "снежинки", служит процедура SNOWLINE. "Черепашка" проходит треть текущей стороны, поворачивается влево на 60 градусов, рисует линию длиной 1/3 от предыдущей, поворачивается направо на 120 градусов и т.д.

```
TO SNOWLINE :LENGTH  
  FORWARD :LENGTH/3  
  LEFT 60  
  FORWARD :LENGTH/3  
  RIGHT 120  
  FORWARD :LENGTH/3  
  LEFT 60  
  FORWARD :LENGTH/3  
END
```

Это создает тот эффект, который нам нужен, но СТОП! Вместо каждой из сторон треугольника мы получили по четыре отрезка. Но ведь на каждом из них должен быть построен свой треугольник. Поэтому всякая команды FORWARD в процедуре SNOWLINE должна быть фактически заменена вызовом самой этой процедуры SNOWLINE.

Очевидно, что с каждым шагом стороны этих треугольников будут все меньше и меньше и их трудно будет рассмотреть.

Мы можем заранее установить какой-то предел на этот размер и когда длина стороны становится меньше этого предела, прекращаем эту рекурсию и идем дальше. Если этот предел включить в качестве параметра в процедуру SNOWLINE, то можно начинать с довольно больших треугольников и постепенно вычерчивать "снежинку" все детальнее и детальнее.

```
TO SNOWFLAKE :LENGTH :LIMIT  
  REPEAT 3 [SNOWLINE :LENGTH RIGHT 120]  
END  
  
TO SNOWLINE :LENGTH  
  IF :LENGTH < :LIMIT [FORWARD :LENGTH STOP]  
  SNOWLINE :LENGTH/3  
  LEFT 60  
  SNOWLINE :LENGTH/3
```

```

RIGHT 120
SNOWLINE :LENGTH/3
LEFT 60
SNOWLINE :LENGTH/3
END

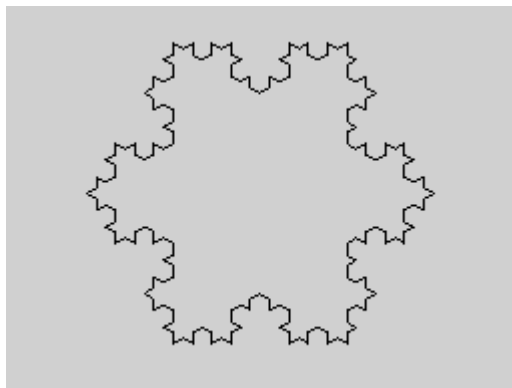
```

Хорошим демонстрационным примером будет:

```

PENUP
SETY -30
PENDOWN
SNOWFLAKE 100 5

```



И напоследок кривая, которую называют "драконом". Она рисуется двумя процедурами, которые вызывают друг друга по-очереди. Между вызовами одна процедура обеспечивает поворот направо, а другая - налево. Длина шага в рекуррентной последовательности одна и та же, поэтому здесь нам нужен иной метод для того, чтобы знать, когда остановиться. Можно, например, сделать так, что когда процедура дойдет до определенного уровня глубины рекурсии, то процесс должен прерываться. Для этого введем счетчик, который назовем LEVL. Он начнет свою работу, например со значения 5 и при каждой рекурсии будет убывать на единицу, пока не обнулится.

Вот эти процедуры:

```

TO LEFTDRAGON :SIZE :LEVL
  IF :LEVL = 0 [FORWARD :SIZE STOP]
  LEFTDRAGON :SIZE :LEVL-1
  LEFT 90
  RIGHTDRAGON :SIZE :LEVL-1
END

TO RIGHTDRAGON :SIZE :LEVL
  IF :LEVL = 0 [FORWARD :SIZE STOP]
  LEFTDRAGON :SIZE :LEVL-1
  RIGHT 90
  RIGHTDRAGON :SIZE :LEVL-1
END

```

Не имеет значения, с какой процедуры Вы начнете работу. Рекомендуется начинать с небольших значений SIZE и LEVL. Начинать работу можно, например, так:

```

CLEARSCREEN
PENUP
SETH 90
SETX 100
PENDOWN
LEFTDRAGON 2 20

```

(Продолжение следует)

# ПРИМЕНЕНИЕ АСSEMBЛЕРА ДЛЯ СОЗДАНИЯ БЫСТРОРАБОТАЮЩИХ ПРОГРАММ

Перевод с английского Пашорина В.И.  
Продолжение. Начало см. с. 9-12, 48-61, 97-103

## 7. ДВИЖЕНИЕ

Теперь, когда мы познакомились со способами вывода информации на экран и способами считывания состояния клавиатуры, мы можем рассмотреть способы задания перемещения объектов на экране дисплея. Это, безусловно, важная часть любой игровой программы. В этой главе мы рассмотрим все возможные способы задания перемещения.

Для организации движения изображения по экрану большинство программистов используют знакоместа, что позволяет ускорить выполнение программы и оперировать с любым из возможных цветов. Весь экран представляет собой прямоугольную матрицу 24x32, т.е. всего 768 знакомест и поэтому перемещение символа из одного знакоместа в другое, соседнее с ним, выглядит не совсем плавным. При этом, если необходимо показать перемещение только одного пиксела, приходится переписывать весь символ из одного знакоместа в другое. Однако при этом, в отличие от BASIC-команды PLOT, нет ограничений на выбор цвета.

### Перемещение знакоместа переднего плана экрана

Для начала необходимо определить стартовую позицию шаблона и затем, используя способы, рассмотренные в главе 5, установить шаблон в это знакоместо. Обычно начальное положение дублируется, поскольку полностью задать положение шаблона - это значит установить его текущее и последующее положения на экране дисплея. Для исключения мигания при перемещении шаблона необходимо сразу же выводить его в новой позиции целиком. Такой способ используется для задания режима OVER 1, при котором перемещение шаблона осуществляется на фоне заднего плана без изменения последнего.

Программа 7.1 демонстрирует этот способ. Здесь шаблон в виде космического корабля перемещается по экрану вверх/вниз при нажатии клавиш 1/Q и вправо /влево при нажатии клавиш 9/0. Клавиша "SPACE" используется для возврата в БЕЙСИК.

Листинг 7. 1.

| АДРЕС | МЕТКА  | МАШ. КОД                     | АСSEMBLER      | КОММЕНТАРИЙ                                                                                                                                                               |
|-------|--------|------------------------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       |        |                              | ORG 23760      |                                                                                                                                                                           |
| 23760 |        | F3                           | DI             | ; Отключение прерываний.                                                                                                                                                  |
| 23761 |        | 18 18                        | JR START       | ; Переход на начало работы.                                                                                                                                               |
| 23763 | DATA_1 | 0 0 15 24 49 226 225 224     |                | ; Шаблон символа CHR\$ 144.                                                                                                                                               |
| 23771 |        | 24 60 255 255 153 255 231 60 |                | ; Шаблон символа CHR\$ 145.                                                                                                                                               |
| 23779 |        | 0 0 240 24 140 71 135 7      |                | ; Шаблон символа CHR\$ 145.                                                                                                                                               |
| 23787 | START  | ED 5B 7B 5C                  | LD DE, (23675) | ; Начало области UDG.                                                                                                                                                     |
| 23791 |        | 21 D3 5C                     | LD HL, DATA_1  |                                                                                                                                                                           |
| 23794 |        | 01 18 00                     | LD BC, 24      |                                                                                                                                                                           |
| 23797 |        | ED B0                        | LDIR           | ; Задание трех первых символов<br>; графики UDG.                                                                                                                          |
| 23799 |        | 18 0A                        | JR BEGIN       | ; Переход на начало работы.                                                                                                                                               |
| 23801 | DATA_2 | 22 0 0 16 8 17 8 144 145 146 |                | ; Эта последовательность ко-<br>; дов является символьной стро-<br>; кой, которую можно трактовать<br>; как AT 0,0; PAPER 8; INK 8;<br>; CHR\$ 144; CHR\$ 145; CHR\$ 146. |
| 23811 | BEGIN  | FD 36 0E 28                  | LD (IY+14), 40 | ; Выставляем цвет бордюра.                                                                                                                                                |
| 23815 |        | FD 36 53 0F                  | LD (IY+83), 15 | ; Выставляем INK и PAPER.                                                                                                                                                 |

|          |             |                |                                  |
|----------|-------------|----------------|----------------------------------|
| 23819    | 3E 02       | LD A, 02       |                                  |
| 23821    | CD 01 16    | CALL 5633      | ; Открыли канал печати на экран. |
| 23824    | CD 6B 0D    | CALL 3435      | ; Очистка экрана.                |
| 23827    | 3E 02       | LD A, 02       |                                  |
| 23829    | CD 01 16    | CALL 5633      |                                  |
| 23832    | 11 F9 5C    | LD DE, DATA_2  | ; Начало символьной строки.      |
| 23835    | 01 0A 00    | LD BC, 10      | ; Длина символьной строки.       |
| 23838    | CD 3C 20    | CALL 8252      | ; Печать символьной строки.      |
| 23841 L7 | 11 00 5B    | LD DE, 23296   | ; Начало буфера принтера.        |
| 23844    | 21 F9 5C    | LD HL, DATA_2  | ; Начало символьной строки.      |
| 23847    | 01 0A 00    | LD BC, 10      | ; Длина символьной строки.       |
| 23850    | ED B0       | LDIR           | ; Переброска символьной строки   |
|          |             |                | ; DATA_2 в начало буфера прин-   |
|          |             |                | ; тера.                          |
| 23852    | 2A 01 5B    | LD HL, (23297) | ; Координаты текущей позиции     |
|          |             |                | ; печати.                        |
| 23855    | 01 FE EF    | LD BC, 61438   |                                  |
| 23858    | ED 78       | IN A, (C)      | ; Ввод данных с порта клавиату-  |
|          |             |                | ; ры. Согласно тому числу, ко-   |
|          |             |                | ; торое установлено в BC, здесь  |
|          |             |                | ; принимается сигнал с правой    |
|          |             |                | ; половины верхнего ряда клавиш  |
|          |             |                | ; (клавиши 6...0).               |
| 23860    | CB 47       | BIT 0, A       | ; Проверка клавиши "0".          |
| 23862    | 20 01       | JR NZ, L1      | ; Переход, если не нажата.       |
| 23864    | 24          | INC H          | ; Если нажата.                   |
| 23865 L1 | CB 4F       | BIT 1, A       | ; Проверка клавиши "9".          |
| 23867    | 20 02       | JR NZ, L2      | ; Переход, если не нажата.       |
| 23869    | 25          | DEC H          | ; Если нажата.                   |
| 23870 L2 | 01 FE F7    | LD BC, 63486   |                                  |
| 23873    | ED 78       | IN A, (C)      | ; Опрос полуряда (1...5).        |
| 23875    | CB 47       | BIT 0, A       | ; Проверка клавиши "1".          |
| 23877    | 20 01       | JR NZ, L3      | ; Переход, если не нажата.       |
| 23879    | 2D          | DEC L          | ; Если нажата.                   |
| 23880 L3 | 01 FE FB    | LD BC, 64510   |                                  |
| 23883    | ED 78       | IN A, (C)      | ; Опрос полуряда (Q...T).        |
| 23885    | CB 47       | BIT 0, A       | ; Проверка клавиши "Q".          |
| 23887    | 20 01       | JR NZ, L4      | ; Переход, если не нажата.       |
| 23889    | 2C          | INC L          | ; Если нажата.                   |
| 23890 L4 | 7D          | LD A, L        | ; Координата по вертикали.       |
| 23891    | FE 16       | CP 22          | ; Проверка на выход за пре-      |
|          |             |                | ; дела экрана снизу.             |
| 23893    | 20 03       | JR NZ, L5      | ; Переход, если все в порядке.   |
| 23895    | 2D          | DEC L          | ; Иначе уменьшаем L              |
| 23896    | 18 05       | JR L6          | ; и переходим на метку L6.       |
| 23898 L5 | FE FF       | CP 255         | ; Проверка на выход за пре-      |
|          |             |                | ; дела экрана сверху.            |
| 23900    | 20 01       | JR NZ, L6      | ; Переход, если все в порядке.   |
| 23902    | 2C          | INC L          | ; Иначе увеличиваем L.           |
| 23903 L6 | 7C          | LD A, H        |                                  |
| 23904    | FE 1E       | CP 30          | ; Проверка на выход за пре-      |
|          |             |                | ; дела экрана справа.            |
| 23906    | 20 03       | JR NZ, L8      | ; Переход, если все в порядке.   |
| 23908    | 25          | DEC H          | ; Иначе уменьшаем H.             |
| 23909    | 18 05       | JR L9          | ; и переходим на метку L9.       |
| 23911 L8 | FE FF       | CP 255         | ; Проверка на выход за пре-      |
|          |             |                | ; дела экрана слева.             |
| 23913    | 20 01       | JR NZ, L9      | ; Переход, если все в порядке.   |
| 23915    | 24          | INC H          | ; Иначе увеличиваем H.           |
| 23916 L9 | ED 5B 01 5B | LD DE, (23297) | ; Координаты старой позиции      |
|          |             |                | ; печати.                        |
| 23920    | AF          | XOR A          | ; Обнуление флага переноса (это  |
|          |             |                | ; важный момент при подготовке   |
|          |             |                | ; к исполнению команды SBC).     |
| 23921    | E5          | PUSH HL        | ; Запомнили координаты новой     |

|           |             |                  |                                                                                                                             |
|-----------|-------------|------------------|-----------------------------------------------------------------------------------------------------------------------------|
| 23922     | ED 52       | SBC HL, DE       | ; позиции печати на стеке.<br>; Проверяем изменилась ли по-<br>; зиция печати.                                              |
| 23924     | E1          | POP HL           | ; Восстановили новую позицию.                                                                                               |
| 23925     | 28 2C       | JR Z, L12        | ; Если позиция печати не не-<br>; нялась, делаем обход на L12.                                                              |
| 23927     | 22 01 5B    | LD (23297), HL   | ; Запомнили координаты новой<br>; позиции печати в буфере<br>; принтера.                                                    |
| 23930     | FD 36 57 03 | LD (IY+87), 3    | ; Включение OVER 1.                                                                                                         |
| 23934     | 11 F9 5C    | LD DE, DATA_2    |                                                                                                                             |
| 23937     | 01 0A 00    | LD BC, 10        |                                                                                                                             |
| 23940     | CD 3C 20    | CALL 8252        | ; Печать в старых координатах.<br>; а поскольку включен режим<br>; OVER 1, то это на самом деле<br>; не печать, а стирание. |
| 23943     | 11 00 5B    | LD DE, 23296     |                                                                                                                             |
| 23946     | 01 0A 00    | LD BC, 10        |                                                                                                                             |
| 23949     | CD 3C 20    | CALL 8252        | ; Печать в новой позиции.                                                                                                   |
| 23952     | 11 F9 5C    | LD DE, DATA_2    |                                                                                                                             |
| 23955     | 21 00 5B    | LD HL, 23296     |                                                                                                                             |
| 23958     | 01 0A 00    | LD BC, 10        |                                                                                                                             |
| 23961     | LD B0       | LDIR             | ; Копирование новой позиции<br>; в старую.                                                                                  |
| 23963     | 21 88 13    | LD HL, 5000      |                                                                                                                             |
| 23966 L11 | 2B          | DEC HL           |                                                                                                                             |
| 23967     | 7C          | LD A, H          | Цикл задержки.                                                                                                              |
| 23968     | B5          | OR L             |                                                                                                                             |
| 23969     | 20 FB       | JR NZ, L11       |                                                                                                                             |
| 23971 L12 | 01 FE 7F    | LD BC, 32766     | Опрос нажатия                                                                                                               |
| 23974     | ED 78       | IN A, (C)        | клавиши                                                                                                                     |
| 23976     | CB 47       | BIT 0, A         | "ПРОБЕЛ".                                                                                                                   |
| 23978     | C2 21 5D    | JP NZ, L7        | ; Если не нажата, переход на L7                                                                                             |
| 23981     | FD 36 57    | 00 LD (IY+87), 0 | ; Восстановление режима OVER 0.                                                                                             |
| 23985     | FB          | EI               | ; Разрешение прерываний.                                                                                                    |
| 23986     | C9          | RET              | ; Возврат в БЕЙСИК.                                                                                                         |

Космический корабль может перемещаться одновременно в двух направлениях без изменения изображения заднего плана. Корабль выводится в цветах PAPER 8 и INK 8, чтобы оставаться контрастным на любом фоне заднего плана изображения. Скорость перемещения корабля регулируется с помощью цикла задержки. Если же корабль остается неподвижным, то процедура стирающая старое изображение и печатающая новое обходится, чтобы избежать мерцания.

Процедура хоть и выглядит немалой по своим размерам, тем не менее очень проста. Просмотрите ее, в ней все должно быть понятно и доступно.

Вы можете усовершенствовать эту программу так, чтобы космический корабль при выходе его за пределы экрана, появлялся с противоположной стороны.

Процедура в машинных кодах выполняется довольно быстро и если в цикле задержки записать LD HL, 1, то за время перемещения корабля его изображение не успеет полностью восстановиться на новой позиции. Поэтому для большинства программ необходимы циклы задержки, чтобы сохранить изображение на время перемещения.

### Перемещение изображения заднего плана экрана

Для задания перемещения заднего плана необходимо организовать перемещение не одного знакоместа, а всего экрана или его части. Это перемещение обычно выполняется в одном из 4 направлений: вверх, вниз, вправо, влево; причем перемещаются одновременно и символы и их атрибуты. Если смещение всего экрана происходит на одну символьную позицию, то необходимо решить, как поступить с символами, вышедшими за пределы экрана. Мы можем либо потерять их и выводить на противоположной стороне экрана пустые

знакоместа, либо запомнить их вместе с атрибутами и затем выводить на экран в соответствующем месте на противоположной стороне. Такой эффект за цикливания изображения весьма полезен для игровых программ.

Программы с 7.2 по 7.5 демонстрируют смещение атрибутов без возврата с противоположной стороны для всех четырех направлений (такое движение принято называть "протяжкой" - SCROLL). Программы с 7.6 по 7.9 демонстрируют смещение атрибутов по всем четырем направлениям с возвратом (назовем это движение "прокруткой" - ROLL).

"Прокрутку" и "протяжку" символов для всех четырех направлений демонстрируют программы 7.10 ... 7.18.

## Листинг 7.2

### SCROLL вверх для атрибутов.

| АДРЕС    | МЕТКА | МАШ. КОД | АССЕМБЛЕР     | КОММЕНТАРИЙ                                                                     |
|----------|-------|----------|---------------|---------------------------------------------------------------------------------|
|          |       |          | ORG 23760     |                                                                                 |
| 23760    |       | 11 00 58 | LD DE, 22528  | ; Начало области экранных<br>; атрибутов.                                       |
| 23763    |       | 21 20 58 | LD HL, 22560  | ; Начало области атрибутов<br>; для второго ряда знакомест.                     |
| 23766    |       | 01 E0 02 | LD BC, 736    | ; Длина перебрасываемого блока<br>; (23 нижних строки по 32 зна-<br>; коместа). |
| 23769    |       | ED B0    | LDIR          | ; Перенос атрибутов на одну<br>; строку вверх.                                  |
| 23771    |       | 06 20    | LD B, 32      | Заполнение нижней строки                                                        |
| 23773    |       | 3A 8D 5C | LD A, (23693) | экрана постоянными атрибута-                                                    |
| 23776 L2 |       | 12       | LD (DE), A    | ми, взятыми из системной пе-                                                    |
| 23777    |       | 13       | INC DE        | ременной ATTR_P(23693)                                                          |
| 23778    |       | 10 F2    | DJNZ L2       |                                                                                 |
| 23780    |       | C9       | RET           | ; Возврат.                                                                      |

Эти программы легко усовершенствовать для организации ROLL и SCROLL только части экрана. Весь экран условно разбит на 3 сегмента, по 8 рядов в каждом. Каждый ряд, в свою очередь, состоит из 8 линий. Каждая линия в дисплейной области памяти занимает 32 байта. Причем записывается в эту область памяти сначала первая линия первого ряда, затем первая линия второго ряда и т.д. последовательно до первой линии восьмого ряда, включительно.

Затем записываются вторые линии первых восьми рядов, затем третьи и так последовательно весь сегмент, после чего записывается второй и за ним последовательно третий сегмент экрана. Каждый сегмент занимает 2046 байтов в дисплейной области памяти.

Обратите внимание, что в процедурах 7.3...7.18 используются команды передачи блока байтов - LDIR и LDDR. После выполнения этих команд в регистровых парах HL и DE остается записанный конечный адрес блока +1 (для LDIR) и конечный адрес - 1 (для LDDR), Это можно использовать для задания адреса продолжения выполнения программ процедуры, а не перезаписывать адреса в HL и DE.

Программы 7.2...7.18 написаны так, чтобы не вносить в них изменения при смене адреса загрузки. Программа 7.19 является усовершенствованной версией программы SCREEN UP SCROLL. Если Вам все понятно в ней, попробуйте сами усовершенствовать программу SCREEN DOWN SCROLL.

Сравнивая программы 7.15 и 7.19 Вы заметите что вторая занимает уже не 99, а только 87 байтов, хотя и это не абсолютно компактная программа. Весьма полезные детали искусства программирования, а также способы уменьшения объема памяти и времени работы показаны в программе 7.15. Эта программа перемещает вверх каждый символ ряда, линия за линией.



### Листинг 7.3

#### SCROLL вниз для атрибутов

```
L7 LD DE, 23295
LD HL, 23253
LD BC, 736
LDDR
LD B, 32
LD A, (23593)
LD (BE), A
INC DE
DJNZ L7
RET
```

### Листинг 7.4

#### SCROLL влево для атрибутов

```
L12 LD B, 24
LD HL, 22528
PUSH HL
POP DE
PUSH BC
INC HL
LD BC, 31
LDIR
LD A, (23693)
LD (DE), A
POP BC
DJNZ
L12 RET
```

### Листинг 7.5

#### SCROLL вправо для атрибутов

```
L14 LD B, 24
LD HL, 23295
PUSH HL
POP DE
PUSH BC
DEC HL
LD BC, 31
LDDR
LD A, (23693)
LD (DE), A
POP BC
DJNZ, L14
RET
```

### Листинг 7.6

#### ROLL вверх для атрибутов.

```
LD HL, 22528 ; Начало ат-
; рибутов.
PUSH HL ; Копирование
POP DE ; через стек.
LD B, 32 ; 32 знако-
; места ряда.
L16 LD A, (HL) ; Запомнили
PUSH AF ; атрибуты на
; стеке.
INC HL ; Знакомес-
; то справа.
DJNZ L16 ; Повтор для
; всего ряда.
LD BC, 736 ; Протяжка для
LDIR ; 23-х нижних
; рядов.
LD B, 32 ; Восстанавли-
L17 POP AF ; ваем атрибуты
DEC HL ; верхнего ряда
LD (HL), A ; в нижнем ряду
DJNZ L17 ; экрана.
RET ; Возврат.
```

### Листинг 7.7

#### ROLL вниз для атрибутов

```
LD HL, 23295
PUSH HL
POP DE
LD B, 32
L24 LD A, (HL)
PUSH AF
DEC HL
DJNZ L24
LD BC, 736
LDDR
LD B, 32
L25 POP AF
INC HL
LD (HL), A
DJNZ L25
RET
```

### Листинг 7.8

#### ROLL влево для атрибутов

```
L32 LD B, 24
LD HL, 22528
PUSH HL
POP DE
PUSH BC
LD A, (HL)
INC HL
LD BC, 31
LDIR
LD (DE), A
POP BC
DJNZ L32
RET
```

### Листинг 7.9

#### ROLL вправо для атрибутов

```
L33 LD B, 24
LD HL, 23295
PUSH HL
POP DE
PUSH BC
LD A, (HL)
DEC HL
LD BC, 31
LDDR
LD (DE), A
POP BC
DJNZ L33
RET
```

### Листинг 7.10

#### ROLL влево для символов.

```
L1 LD B, 192 ; Высота экрана.
LD HL, 16384 ; Начало экран-
; ной области.
PUSH HL ; Копирование
POP DE ; через стек.
PUSH BC ; Запомнили BC.
LD A, (HL) ; Запомнили те-
; кущую линию
; самого левого
; знакоместа.
INC HL ; Адрес линии
; справа.
LD BC, 31 ; 31 знакоместо
; в ряду.
LDIR ; Копирование.
LD (DE, A) ; Копирование
; линии самого
; левого знако-
; места в самое
; правое.
POP BC ; Восстановление
; счетчика линий
DJNZ L1 ; Если не все
; линии скопиро-
; ваны, то воз-
; врат на L1.
RET ; Выход.
```

### Листинг 7.11

#### ROLL вправо для символов

```
L1 LD B, 192
LD HL, 22527
PUSH HL
POP DE
PUSH BC
LD A, (HL)
DEC HL
LD BC, 31
LDDR
LD (DE), A
POP BC
DJNZ L1
RET
```

### Листинг 7.12

#### SCROLL влево для символов.

```
L13 LD HL, 16384
XOR A
LD B, 192
PUSH BC
PUSH HL
POP DE
INC HL
LD BC, 31
LDIR
LD (DE), A
POP BC
DJNZ L13
RET
```

### Листинг 7.13

#### SCROLL влево для символов. (второй вариант)

```
L1 LD HL, 16384
LD DE, 32
LD B, 192
LD (HL), 0
ADD HL, DE
DJNZ L1
LD HL, 16385
LD DE, 16384
LD BC, 6143
LDIR
EX DE, HL
LD (HL), 0
RET
```

### Листинг 7.14

#### SCROLL вправо для символов.

```
L14 LD HL, 22527
XOR A
LD B, 192
PUSH BC
PUSH HL
POP DE
DEC HL
LD BC, 31
LDDR
LD (DE), A
POP BC
DJNZ L14
RET
```

### Листинг 7.15

**SCROLL** вверх для символов.

```
L3 LD DE, 16384
LD HL, 16416
LD BC, 2016
LDIR
LD B, 8
LD HL, 18432
LD DE, 16608
PUSH BC
PUSH HL
PUSH DE
LD BC, 32
LDIR
POP DE
POP HL
POP BC
INC D
INC H
DJNZ L3
LD DE, 18432
LD HL, 18464
LD BC, 2016
LDIR
LD B, 8
LD HL, 20460
LD DE, 18656
L4 PUSH BC
PUSH HL
PUSH DE
LD BC, 32
LDIR
POP DE
POP HL
POP BC
INC D
INC H
DJNZ L4
LD DE, 20480
LD HL, 20512
LD BC, 2016
LDIR
LD B, 8
LD HL, 20704
L5 PUSH BC
PUSH HL
LD B, 32
L6 LD(HL), 0
INC HL
DJNZ L6
POP HL
POP BC
INC H
DJNZ L5
RET
```

### Листинг 7.16

**ROLL** вверх для символов.

```
LD HL, 16384
L18 LD B, 32
L19 LD A, (HL)
PUSH AF
INC HL
DJNZ L19
XOR A
LD L, A
INC H
LD A, H
CP 72
JR NZ, L18
LD DE, 16384
LD HL, 16415
LD BC, 2016
LDIR
LD B, 8
LD HL, 18342
LD DE, 16608
L20 PUSH BC
PUSH HL
PUSH DE
LD BC, 32
LDIR
POP DE
POP HL
POP BC
INC D
INC H
DJNZ L20
LD DE, 18432
LD HL, 18464
LD BC, 2016
LDIR
LD B, 8
LD HL, 20480
LD DE 18656
L21 PUSH BC
PUSH HL
PUSH DE
LD BC, 32
LDIR
POP DE
POP HL
POP BC
INC D
INC H
DJNZ L21
LD DE, 20480
LD HL, 20512
LD BC, 2016
LDIR
LD HL, 22527
L22 LD B, 32
L23 POP AF
LD (HL), A
DEC HL
DJNZ L23
LD A, 255
LD L, A
DEC H
LD A, H
CP 79
JR NZ, L22
RET
```

### Листинг 7.17 SCROLL вниз для символов.

L8 LD DE, 22527  
LD HL, 22495  
LD BC, 2016  
LDDR  
LD B, 8  
LD HL, 18656  
LD DE, 20480  
PUSH BC  
PUSH HL  
PUSH DE  
LD BC, 32  
LDIR  
POP DE  
POP HL  
POP BC  
INC D  
INC H  
DJNZ L6  
LD DE, 20479  
LD HL, 20447  
LD BC, 2016  
LDDR  
LD B, 8  
LD HL, 16608  
LD DE, 18432

L9 PUSH BC  
PUSH HL  
PUSH DE  
LD BC, 32  
LDIR  
POP DE  
POP HL  
POP BC  
INC D  
INC H  
DJNZ L9  
LD DE, 18431  
LD HL, 18399  
LD BC, 2016  
LDDR  
LD HL, 16384  
LD B, 32  
L10 LD(HL), 0  
L11 INC HL  
DJNZ L11  
XOR A  
LD L, A  
INC H  
LD A, H  
CP 72  
JR NZ, L10  
RET

### Листинг 7.18 ROLL вниз для символов

|     |                                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                           |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| L26 | LD HL, 22527<br>LD B, 32                                                                                                                                                                                                             | ; Конец дис-<br>; плейной об-<br>; ласти.<br>; 32 знакоместа<br>; в строке,<br>; Запомнили<br>; нижнюю линию<br>; текущего зна-<br>; коместа.<br>; Знакоместо<br>; слева.                                                                                                                                                                                                 |
| L27 | LD A, (HL)<br><br>PUSH AF<br><br>DEC HL<br><br>DJNZ L27<br>LD A, 255<br><br>LD L, A<br>DEC H<br>LD A, H<br><br>CP 79<br><br><br>JR NZ, L26<br><br>LD DE, 22527<br>LD HL, 22495<br>LD BC, 2016<br>LDDR<br><br>LD B, 8<br>LD HL, 18656 | ; И так 8 раз.<br>; Возврат к<br>; предыдущей<br>; линии послед-<br>; него знако -<br>; места ряда.<br>; Проверка на<br>; выход в 23-ю<br>; строку экрана<br>; Если нет, то<br>; возврат назад<br>; Копирование<br>; информации в<br>; третьем сег-<br>; менте на ряд<br>; вниз.<br>; Восемь линий.<br>; Адрес начала<br>; первой линии<br>; последнего<br>; ряда второго |

|     |              |                 |
|-----|--------------|-----------------|
|     | LD DE, 20480 | ; сегмента.     |
|     |              | ; Адрес начала  |
|     |              | ; первой линии  |
|     |              | ; первого ряда  |
|     |              | ; третьего сег- |
|     |              | ; мента.        |
| L28 | PUSH BC      | ; Сохранили PP- |
|     | PUSH HL      | ; гистры от     |
|     | PUSH DE      | ; порчи коман-  |
|     |              | ; дой LDIR.     |
|     | LD BC, 32    | ; 32 байта в    |
|     |              | ; каждой линии. |
|     | LDIR         | ; Перенос одной |
|     |              | ; линии через   |
|     |              | ; границу сег-  |
|     |              | ; ментов.       |
|     | POP DE       | ; восстановле-  |
|     | POP HL       | ; ние регистров |
|     | POP BC       |                 |
|     | INC D        | ; Переход к     |
|     | INC H        | ; очередной ли- |
|     |              | ; нии ряда.     |
|     | DJNZ L28     |                 |
|     |              | ; Возврат для   |
|     |              | ; переноса 8-ми |
|     |              | ; линий через   |
|     |              | ; границу сег-  |
|     |              | ; мента.        |
|     | LD DE, 20479 | ; Движение ин-  |
|     | LD HL, 20447 |                 |
|     |              | ; формации во   |
|     | LD BC, 2016  | ; втором сег-   |
|     | LDDR         | ; менте на ряд  |
|     |              | ; вниз.         |
|     | LD B, 8      |                 |
|     | LD HL, 16608 |                 |
|     | LD DE, 18432 | Перенос         |
| L29 | PUSH BC      | информации      |
|     | PUSH HL      | через           |
|     | PUSH DE      | границу         |
|     | LD BC, 32    | между           |
|     | LDIR         | первым          |
|     | POP DE       | и вторым        |
|     | POP HL       | сегментами.     |
|     | POP BC       |                 |
|     | INC D        |                 |
|     | INC H        |                 |
|     | DJNZ L29     |                 |
|     | LD DE, 18431 | ; Движение ин-  |
|     | LD HL, 18399 | ; формации в    |
|     | LD BC, 2016  | ; первом сег-   |
|     | LDDR         | ; менте на ряд  |
|     |              | ; вниз.         |
|     | LD HL, 16384 | ; Начало экрана |
| L30 | LD B, 32     |                 |
| L31 | POP AF       | Восстановле-    |
|     | LD (HL), A   | ние со стека    |
|     | INC HL       | информации      |
|     | DJNZ L31     | для восьми      |
|     | XOR A        | линий первого   |
|     | LD L, A      | ряда (на стек   |
|     | INC H        | она поступила   |
|     | LD A, H      | из восьми       |
|     | CP 72        | нижних линий    |

|            |          |
|------------|----------|
| JR NZ, L30 | экрана). |
| RET        | ; Выход. |

## Листинг 7.19

### SCROLL вверх для символов (2-ой вариант)

|    |              |                 |
|----|--------------|-----------------|
|    | LD HL, 16384 | ; Начало дис-   |
|    |              | ; плейной об-   |
|    |              | ; ласти памяти. |
|    | CALL SEGM    | ; Движение пер- |
|    |              | ; вого сегмента |
|    | EX DE, HL    | Перенос         |
|    | LD HL, 18438 | ряда через      |
|    | PUSH HL      | границу         |
|    | CALL UP      | сегментов.      |
|    | POP HL       |                 |
|    | CALL SEGM    | ; Движение вто- |
|    |              | ; рого сегмента |
|    | EX DE, HL    | Перенос ряда    |
|    | LD HL, 20480 | через границу   |
|    | PUSH HL      | сегментов.      |
|    | CALL UP      |                 |
|    | POP HL       |                 |
|    | CALL SEGM    | ; Движение тре- |
|    |              | ; тьего сегмен- |
|    |              | ; та.           |
| L4 | LD HL, 20704 | Обнуление       |
|    | PUSH HL      | (стирание)      |
|    | LD 8, 32     | информации      |
| L3 | LD (HL), 0   | в последнем     |
|    | INC HL       | ряду            |
|    | DJNZ L3      | третьего        |
|    | POP HL       | сегмента.       |
|    | INC H        |                 |
|    | LD A, H      |                 |
|    | CP 88        |                 |
|    | JR NZ, L4    |                 |
|    | RET          | ; Выход         |

### Продолжение Л. 7.19.

Нижеследующая процедура копирует все ряды (кроме верхнего) в текущем сегменте на один ряд выше.

|      |            |                   |
|------|------------|-------------------|
| SEGM | LD B, 7    | ; Первый ряд сег- |
|      |            | ; мента не пере-  |
|      |            | ; носим, потому 7 |
| L1   | PUSH BC    | ; Сохранили от    |
|      |            | ; "порчи".        |
|      | LD BC, 32  | ; Соответствующая |
|      |            | ; линия следующе- |
|      |            | ; го ряда отстоит |
|      |            | ; на 32 байта от  |
|      |            | ; текущей.        |
|      | PUSH HL    | ; Копирование ад- |
|      | POP DE     | ; реса через стек |
|      | ADD HL, BC | ; HL указывает на |
|      |            | ; ряд ниже        |
|      | PUSH HL    | ; Сохранили HL и  |
|      |            | ; DE от "порчи"   |
|      | PUSH DE    | ; командой LDIR   |
|      | CALL UP    | ; Копирование ря- |

|         |  |                  |
|---------|--|------------------|
|         |  | ; да вверх.      |
| POP DE  |  |                  |
|         |  | Восстановление   |
| POP HL  |  | регистров.       |
| POP BC  |  |                  |
| DJNZ L1 |  | ; Если не все 7  |
|         |  | ; рядов скопиро- |
|         |  | ; ваны, возврат. |
| RET     |  | ; Выход.         |

Нижеследующая процедура копирует содержимое экранного ряда на один ряд выше.

|    |           |                    |
|----|-----------|--------------------|
| UP | LD B, 8   | ; 8 линий в ряду.  |
| L2 | PUSH BC   | ; Сохранили регис- |
|    | PUSH HL   | ; тры от "порчи"   |
|    | PUSH DE   | ; командой LDIR.   |
|    | LD BC, 32 | ; 32 байта в линии |
|    | LDIR      | ; Перенос на один  |
|    |           | ; ряд вверх.       |
|    | POP DE    |                    |
|    | POP HL    | ; Восстановление   |
|    | POP BC    | ; регистров.       |
|    | INC D     | ; Переход к оче-   |
|    |           | ; редной линии     |
|    | INC H     | ; текущего ряда.   |
|    | DJNZ L2   | ; Если не все 8    |
|    |           | ; линий скопирова- |
|    |           | ; ны, возврат.     |
|    | RET       | ; Выход.           |

Вместо программы, выполняющей перемещение сразу же целого ряда, можно написать программу, выполняющую перемещение только одного столбца. Чтобы переместить весь ряд, эта программа должна повториться 32 раза. В то же время, такое решение позволит перемещать не весь ряд, а только заданное количество столбцов и, более того, при таком решении нет необходимости хранить в памяти все 256 байтов верхнего ряда экрана (как, например, в программе 7.18), а только 8 байтов одного столбца ряда. Программа 7.20 показывает, как достигается такое уменьшение объема памяти. Здесь адрес первого байта смещаемого столбца записывается в регистровую пару DE, а общее число смещаемых столбцов записывается в регистр B. Эта программа может быть размещена в любом участке памяти.

#### Листинг 7.20.

##### SCROLL вверх по столбцам.

|    |              |                  |
|----|--------------|------------------|
|    | XOR A        | ; Сброс ак-ра.   |
|    | LD DE, 16388 | ; Адрес верхней  |
|    |              | ; линии 4-го     |
|    |              | ; столбца.       |
|    | LD B, 10     | ; Двигаем вверх  |
|    |              | ; 10 столбцов.   |
| L5 | PUSH DE      | ; Сохранение ре- |
|    |              | ; гистров        |
|    | PUSH BC      | ; от корропции.  |
|    | PUSH DE      | ; Копирование из |
|    | POP HL       | ; DE в HL.       |
|    | LD B, 8      | ; 8 линий ряда.  |
| L6 | LD A, (HL)   | ; Запоминаем на  |
|    | PUSH AF      | ; стеке восемь   |
|    | INC H        | ; линий верхнего |
|    | DJNZ L6      | ; ряда.          |
| L4 | LD B, 7      |                  |
|    |              | ; 7 рядов сег-   |
|    |              | ; мента.         |

|     |                                                                                      |                                                                                                                                                                  |
|-----|--------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| L2  | PUSH BC<br>LD HL, 32<br>ADD HL, DE                                                   | ; Указание на<br>; следующую ли-<br>; нию ряда.                                                                                                                  |
|     | PUSH HL<br>LD B, 8                                                                   | ; Восемь линий<br>; в ряду.                                                                                                                                      |
| LS  | LD A, (HL)<br>LD (DE), A<br>INC H<br>INC D<br>DJNZ L1<br>POP DE<br>POP BC<br>DJNZ L2 | ; Копирование<br>; через акк-р<br>; для одного<br>; ряда<br>; Восстановление<br>; регистров.<br>; Повтор для<br>; очередного ря-<br>; да в сегменте.             |
|     | LD HL, 1824<br>ADD HL, DE<br>LD A, H<br>CP 88<br>JR Z, END<br>PUSH HL<br>LD B, 8     | ; Переход через<br>; границу сегм.<br>; Проверка на<br>; конец экран-<br>; ной области.                                                                          |
| L3  | LD A, (HL)<br>LD (DE), A<br>INC H<br>INC D<br>DJNZ L3<br>POP DE<br>JR L4             | ; Перенос восьми<br>; линий через<br>; границу сег-<br>; мента.<br>; Возврат для<br>; работы со сле-<br>; дующий сегм.                                           |
| END | LD DE, 32<br>SBC HL, DE<br>LD B, 8                                                   | ; Восстановле-<br>; ние со стека в<br>; нижнем ряду                                                                                                              |
| L7  | POP AF<br>LD (HL), A<br>DEC HL<br>DJNZ L7<br>POP BC<br>POP DE<br>INC DE              | ; столбца того,<br>; что запомнили<br>; в самом верх-<br>; нем ряду.<br>; Восстановление<br>; регистров.<br>; Переход к оче-<br>; редному столб-<br>; цу экрана. |
|     | DJNZ L5                                                                              | ; Если не все 10<br>; столбцов об-<br>; служены, воз-<br>; врат на L5.                                                                                           |
|     | RET                                                                                  | ; Выход.                                                                                                                                                         |

Дальнейшее уменьшение объема памяти, занимаемого программой, может быть достигнуто, если выполнять перемещение одновременно не всех рядов, а только заданного их количества. При этом можно реализовать смещение на экране определенного окна с заданными размерами. Программа 7.31, занимающая всего 58 байтов, демонстрирует это решение. Для проверки выхода на границу сегмента экрана здесь используется команда BIT 0, H. Если нулевой бит регистра H не равен 0, то мы находимся на границе сегмента экрана и вместо добавления числа 32 к числу, записанному в регистровой паре DE, необходимо добавить число 1824 для перевода "первого" ряда в следующий сегмент. Число смещаемых рядов определяется числом шагов цикла L4. Адрес ячейки верхнего левого угла смещаемого окна может быть определен по формуле:

$$16384 + (1824 * \text{INT}(\text{ROW}/8)) + ((\text{ROW} - \text{INT}(\text{ROW}/8)) * 32 + \text{COLUMN})$$



Здесь ROW - номер экранной строки (в знакоместах).

COLUMN - номер экранного столбца (в знакоместах).

Для смещения всего экрана в регистровой паре DE должно быть записано число 16384, в регистре В перед циклом L6 число 32 и здесь же, но перед циклом L4 - число 23. Эту программу можно переделать для организации смещения вверх/вниз.

#### Листинг 7.21.

```

XOR A
LD DE, 16515      ; Левый верхний
                  ; угол прокручи-
                  ; ваемого "окна"
                  ; имеет коорди-
                  ; наты X=3;
                  ; Y=4.
LD B, 7           ; Ширина "окна"
                  ; - 7 знакомест.
L5    PUSH DE
      PUSH BC
      PUSH DE
      POP HL
      LD B, 8
L1    LD A, (HL)
      PUSH AF
      INC H
      DJNZ L1
      LD B, 13    ; Высота "окна"
                  ; - 13 знакомест
L4    PUSH BC
      LD HL, 32
      ADD HL, DE
      BIT 0, H
      JR Z, L2
      LD HL, 1824
      ADD HL, DE
L2    PUSH HL
      LD B, 8
L3    LD A, (HL)
      LD (DE), A
      INC D
      INC H
      DJNZ L3
      POP DE
      POP BC
      DJNZ L4
      LD B, 8
L5    DEC H
      POP AF
      LD (HL), A
      DJNZ L5
      POP BC
      POP DE
      POP DE
      INC DE
      DJNZ L6
      RET
```

В ПЗУ компьютера ZX Spectrum имеется одна из наиболее полезных процедур - SCROLL n, позволяющая смещать n строк вместе с их атрибутами. Число n должно быть больше или равным 2 (но, конечно, не больше, чем 24).

Программа 7.22 показывает, как можно использовать эту процедуру для очистки строк с 16-ой по 24-ю включительно. Число строк отсчитывается от 24-й, при этом в регистр В записывается число строк-1 и девять раз вызывается процедура SCROLL n, находящаяся по

адресу 3584.

#### Листинг 7.22

```
LD B, 9
L1  PUSH BC
    LD B, 8
    CALL 3584
    POP BC
    DJNZ L1
    RET
```

Ну, и в конце этой главы разберем процедуру, позволяющую одновременно смещать влево (или вправо) сразу же половину символа. Эта процедура хоть и короткая, но довольно сложная, т.к. здесь используются новые команды. Например, команда RLD перемещает биты с 0-го по 3-й в биты с 7-го по 4-й в ячейке, адрес которой находится в регистровой паре HL. Биты же с 7-го по 4-й из этой ячейки переписываются в биты с 0-го по 3-й в регистр A. А данные, которые были в этом регистре в битах с 0-го по 3-й переписываются в ячейку, адрес которой находится в регистровой паре HL, и занимают здесь соответственно 0...3 биты. Эта команда не влияет на флаги состояния процессора.

#### Листинг 7.23

```
DI
LD HL, 22527
LD B, 196
L1  PUSH BC
    XOR A
    LD B, 32
    PUSH HL
L2  RLD
    DEC HL
    DJNZ L2
    POP DE
    EX DE, HL
    ADD A, (HL)
    LD (HL), A
    EX DE, HL
    POP BC
    DJNZ L1
    EI
    RET
```

В программе 7.23 для каждой строки экрана используется команда RLD для перемещения правой половины каждого символа в левую часть. Левая половина первого символа при этом остается пустой, т.к. для каждой строки изначально в регистре A записан 0. Чтобы организовать циркулярное смещение, необходимо временно запоминать значение левой половины первого символа, а затем выводить ее, как правую половину последнего символа.

# НАША ПРЕЗЕНТАЦИЯ

Сегодня мы представляем Вам нашу новую книгу "30 часов БЕЙСИКа для начинающих". Надо сказать, что это наш первый опыт в том смысле, что до сих пор мы издавали только книги, написанные собственными руками. Сейчас, увы, дело обстоит так, что Вы, уважаемые друзья, читаете быстрее, чем мы пишем и потому нам пришлось обратиться к классике. Поводом для этого стали многочисленные обращения дать что-то для начинающих или хотя бы повторить наше самое первое издание, с которым мы дебютировали в 1989 г. - "Большие возможности Вашего "Спектрума".

Мы сразу отказались от идеи повтора нашей первой книги, ведь за эти годы она была перепечатана многими организациями, где слово в слово, где с добавлениями и, вполне возможно, она Вам встречалась под другим наименованием. Вместо этого мы сделали перевод с книги Клайва Пригмора (Clive Prigmore) и Пола Шрива (Paul Shreeve), которая является методическим пособием по "Спектруму" для Британского Национального колледжа заочного повышения квалификации (National Extension College).

Книга готова и ее оригинал-макет на дискете предлагается всем желающим самостоятельно заняться печатью по нашей лицензии и распространением. Мы же ждем ее выход из типографии не раньше ноября текущего года.

В этой книге нам больше всего понравился ход мысли автора. Все основные операторы БЕЙСИКа даются читателю на основе конкретных практических задач. Сегодня в качестве презентации новой книги мы даем одну маленькую главу, которая, как нам кажется, хорошо иллюстрирует методический подход автора. Глава посвящена членению стрингов оператором (...ТО...).

## 5.6 Расчленение стрингов.

Мы никогда не задумывались о том, что такое дата? Например, как компьютер представляет себе 23 июня 1973 года?

Мы знаем, что компьютер умеет хранить данные. И делает это двумя способами - в виде чисел или в виде символьных строк (стрингов). В виде чисел он уделяет по 5 байтов памяти на каждое число, а для стрингов - по одному байту на каждый символ. Так что же такое дата? Число или стринг?

23 июня 1973 года - вроде бы стринг, а если мы запишем то же самое как 230673 - то вроде бы это число. Давайте разбираться.

Если 230573 это число, то почему 110593 (11 мая 1993 года) меньше, ведь 93-ий год был позже 73-го? А если это стринг, то спрашивается, как компьютер сможет рассчитать, сколько лет будет в 93-м году молодому человеку, родившемуся в 73-м, да и вообще как тогда оперировать с календарем?

Ответ очень прост. Даты - это особый вид информации и работать с ними надо особо. Их надо вводить, как стринги, а хранить в памяти и обрабатывать надо как числа. И вводят их именно как стринги еще и потому, что не все элементы даты нам одинаково важны.

Например, если наша организация ежегодно в декабре месяце повышает заработную плату тем, кто отработал у нас более десяти лет, то для этого важен только год, в котором работник был зачислен. Если же наша организация занимается настройкой пианино и роялей, то каждые три месяца мы рассылаем всем своим зарегистрированным клиентам письмо-напоминание о том, что надо бы обслужить инструмент. В этом случае нас интересует только месяц, когда был последний раз обслужен данный клиент, а год и день не имеют никакого значения.

Наконец, для отдела жалоб и претензий, которому полагается разбираться с каждым заявлением в 30-ти дневный срок, важен день и месяц поступления заявления.

Итак, мы видим, что стринг, содержащий дату, имеет важное значение как единое целое, но есть серьезные причины для того, чтобы вырезать из него отдельные части и обрабатывать их отдельно. Таким образом, после того, как дата была введена в формате:

ДДММГГ или

ДД.ММ.ГГ или

ММ.ДД.ГГ или как угодно, нам нужно иметь возможность "вырезать" из нее отдельно дни, месяцы и годы и работать с ними порознь, но теперь уже как с числами.

Для этого нужен специальный оператор, и он есть. На "Спектруме" это делает оператор

(...TO...).

Его полная форма, например, такая:

x\$ (a TO b)

Здесь из строки x\$ вырезается его часть, начинающаяся с символа, номер которого равен a и до символа с номером b.

Если x\$ = "СПЕКТРУМ", то

x\$ (5 TO 7) = "ТРУ"

Вы можете посмотреть, как это происходит, на примере следующей программы:

```
12 INPUT "Введите исходное слово"; m$
16 PRINT m$
20 INPUT "Начало вырезки"; a
40 PRINT a
50 INPUT "Конец вырезки"; b
60 PRINT b
80 LET n$=m$(a TO b)
90 PRINT m$;"(";"a;" TO ";"b;"")=";"n$
100 STOP
```

В некоторых других языках программирования, а также в некоторых других версиях БЕЙСИКа Вы найдете еще несколько операторов для вычленения субстроки из строки. Нам бы не стоило на них и останавливаться, но Вы должны знать, как Вам реализовывать те же возможности, раз этих операторов у Вас нет.

LEFT\$ (x\$,k) - дает Вам k левых символов строки. На "Спектруме" его аналогом будет:

x\$(1 TO k).

RIGHT\$ (x\$,k) - дает k правых символов строки. На "Спектруме" аналог подобрать труднее, приходится пользоваться функцией LEN, которую мы рассмотрели ранее:

x\$(LEN (x\$)-k+1 TO LEN (x\$))

MID\$(x\$,k,m) - дает Вам m символов, начиная с k-го символа. С этим оператором дело обстоит проще:

x\$(k TO k+m)

\* \* \*

Книга написана в стиле дружеского диалога, имеет большое количество вопросов для самоконтроля и упражнений с решениями. Она доставит удовольствие не только тем, кто сам осваивает БЕЙСИК, но и тем, кто передает свой опыт другим.

Общий объем книги довольно велик, но мы "уложили" его в 208 стр. стандартного формата.

Спешите приобрести лицензию, а мы в это время планируем издание собственной книги "БЕЙСИК для подготовленного пользователя".

Ваш "ИНФОРКОМ".

# FORUM

В прошлом выпуске "ZX-РЕВЮ" мы объявили о создании совместно с Государственным Политехническим музеем АССОЦИАЦИИ ПОЛЬЗОВАТЕЛЕЙ ДОМАШНИХ КОМПЬЮТЕРОВ и Фонда "ФЛОП" (фонд литературы, описаний и программ) и читатели уже с интересом откликнулись на это начинание.

Нам пишут Д. Палтусов и В. Кутин из г. Екатеринбурга:

Здравствуйте, господа!

Мы хотим выразить вам глубокую признательность. Дело, начатое Вами позволит сформировать наконец-то в России рынок программ, периферии и литературы для машин класса "Spectrum". Теперь люди, использующие компьютер не только для игр, смогут рассчитывать, что "Родина их не забудет".

ИФК: Большое спасибо на добром слове. Мы передаем все письма с пометкой "ФЛОП" в Политехнический музей. Надо сказать, что это их инициатива в создании такого фонда. Мы же только оказываем посильную помощь в его становлении, пользуясь тем авторитетом, который "ZX РЕВЮ" имеет по стране.

И главное, что мы хотели бы видеть в итоге это не совсем рынок программ и периферии. Скорее нам видится будущее, как свободный рынок талантливых людей. А программы, устройства и прочее - это вроде как визитная карточка этих талантов, способных делать большие дела и получать достойную оплату за свой труд.

KOPP: Мы готовы предложить некоторые свои работы. Это годы работы и килобайты кода:

1. Base - игровая программа (92).
2. Constructor - конструктор для создания игр (92).
3. Generator - редактор спрайтов (91).
4. Instal - дисассемблер (93).
5. Mega-gen - редактор спрайтов (93).
6. Multiedit+Utility - редактор мультфильмов (92).
7. Shareholder - игра (92).
8. Turbo-Copy - копировщик (93).

ИФК: Вашими разработками мы открываем раздел "Авторская программа" (см. соответствующий раздел).

KOPP: Вместе с тем, у нас есть несколько разработок, которые нельзя считать авторскими, поскольку мы просто адаптировали или доработали фирменные программы. Мы полагаем, что авторские права на них мы заявлять не можем. Может быть Вы что-то подскажете?

Это русифицированная и доработанная версия графического редактора "ARTSTUDIO", русифицированные аналоги игр серии DIZZY Оливера Твинса, доработанная версия АССЕМБЛЕРА "ZEUS".

ИФК: Дорогие друзья, мы не сильны в юриспруденции, но знаем, что кроме имущественного авторского права есть еще и неимущественное моральное. А это, как нам кажется, то, что всегда останется с Вами, если Вы знаете, что сделали большое и нужное дело, то почему бы об этом не знать и другим?

Никто не претендует на авторские права Шекспира, когда переводит "Ромео и Джульетту", но разве мы бы познакомились с его произведениями, если бы Маршак и Пастернак не приложили к делу свой талант и переводчика и поэта? И разве сегодня мы не чтим память и того и другого?

А знаете, сколько добавляют и отнимают режиссеры, когда ставят пьесу или снимают фильм по тому же Шекспиру? Конечно, имя Шекспира всегда стоит на афише, но выдающихся режиссеров народ тоже знает и помнит.

Нам кажется, что адаптация программ вещь безусловно нужная, полезная и

стимулировать ее расширение желательно. Тем более, что большинство программистов начинают именно с этого и занимаются этим до тех пор, пока наберутся достаточного опыта для создания собственных программ.

Если создать вокруг адаптации ореол "второсортности", то можно сильно подорвать желание будущих творцов двигаться вперед, что-то изучать, создавать и творить. Интересы нашего общего дела от этого только пострадают. Поэтому мы думаем, что для авторских адаптаций надо ввести свой раздел, что мы и сделаем.

Поэтому пожалуйста пришлите "объективки" на Ваши адаптации (название, автор, дата и т.п.) и мы включим их в раздел.

\* \* \*

Нам пишет Скворцов Владимир Николаевич из города Чебоксары:

Ассоциация пользователей домашних компьютеров - это как раз то, что надо, это здорово! Ведь сколько любителей компьютеров предоставлены сами себе. И у каждого есть свои находки, свои открытия в любимых программах и компьютерах. И как часто хочется поделиться со своими единомышленниками находками и открытиями. Поэтому готов сотрудничать по любым вопросам, связанным с деятельностью Ассоциации.

\* \* \*

Нам пишут из г. Ташкента:

Привет, ИНФОРКОМ!

Мы, Сметанин Юрий и Ильичев Андрей, проведя некоторые исследования, хотели бы внести свой небольшой вклад в досье игры ELITE. К сожалению, мы не получаем "ZX-РЕВЮ" за 1993 год, а хотелось бы, поэтому не знаем, может быть эта работа кем-то уже сделана.

ИФК: Сейчас, когда работа почты стабилизировалась, мы вновь принимаем на себя ответственность отправлять "ZX РЕВЮ" за пределы России и подписку от читателей из республик СНГ принимаем и на 93-ий год и на 94-ый, хотя, правда, для них это стоит существенно дороже. Мы надеемся, что наши читатели скооперируются, ведь "синклеристы" не водятся поодиночке, а ведут социально-активный образ жизни и выписывать "РЕВЮ" они смогут.

КОРР: ... Итак, мы выяснили значение почти всех кодов в символьном наборе программы. (Список кодов прилагается).

ИФК: Сведений о полном исследовании кодов к нам не поступало, хотя Вы вероятно знаете о том, что Д.П. Шилин проводил исследование кодировки слогов текстовых сообщений (ZX-РЕВЮ N5,6 1992г.)

КОРР: ... Значение некоторых кодов, а именно 2,3,4,127 выяснить не удалось. При попытке их распечатать программа "зависает", а некоторые коды 7,10,95... не имеют печатного символа. Возможно это какие-то управляющие коды.

Интересны коды 163,167,192. Их значения непостоянны. На место знака (#) вставляется имя планеты, на которой Вы находитесь. В коде 199 - Galactic Chart - дальше печатается номер галактики, в которой Вы находитесь.

Как видно из анализа набора кодов, абсолютно все сообщения состоят из 1-2 кодов, отсюда становится возможной русификация программы. Над этим мы сейчас и работаем.

|       |                                    |     |                     |
|-------|------------------------------------|-----|---------------------|
| Код   | Значение                           | 115 | DOCKING COMPUTER    |
| 1     | LAVE                               | 116 | GALACTIC HYPERSPACE |
| 2'    |                                    | 117 | MILITARY LASER      |
| 3'    |                                    | 118 | MINING LASER        |
| 4'    |                                    | 119 |                     |
| 5     | FUEL:0.0 LIGHT YEARS               | 120 | INCOMING MISSILE    |
| 6     | CASH 7                             | 121 | ENERGY              |
| 8     | 0.0                                | 122 | GALACTIC            |
| 9     | CR                                 | 123 | DOCKING COMPUTER ON |
| 10    |                                    | 124 | ALL                 |
| 11    |                                    | 125 | CLOAKING DEVICE     |
| 12    | LL                                 | 126 | COMMANDER           |
| 13    |                                    | 127 |                     |
| 14    |                                    | 128 | AL                  |
| 15    | MAY DAY! INVADED                   | 129 | LE                  |
| 16    | ON                                 | 130 | XE                  |
| 17    | EQUIPMENT:                         | 131 | GE                  |
| 18    | : CLEAN                            | 132 | ZA                  |
| 19    | : OFFENDER                         | 133 | CE                  |
| 20    | : FUGITIVE                         | 134 | BI                  |
| 21    | HARMLESS                           | 135 | SO                  |
| 22    | MOSTLY HARMLESS                    | 136 | US                  |
| 23    | POOR                               | 137 | ES                  |
| 24    | AVERAGE                            | 138 | AR                  |
| 25    | ABOVE AVERAGE                      | 139 | MA                  |
| 26    | COMPETENT                          | 140 | IN                  |
| 27    | DANGEROUS                          | 141 | DI                  |
| 28    | DEADLY                             | 142 | RE                  |
| 29    | --- ELITE ---                      | 143 | A                   |
| 30    | PRESENT                            | 144 | ER                  |
| 31    | FUEL LEAK                          | 145 | AT                  |
| 32    |                                    | 146 | EN                  |
| 33    | 32 38                              | 147 | BE                  |
| 34    |                                    | 148 | RA                  |
| 35    | символы                            | 149 | LA                  |
| 36    |                                    | 150 | VE                  |
| 37    | псевдографики                      | 151 | TI                  |
| 38    |                                    | 152 | ED                  |
| 39-58 | Совпадает со стандартным набором   | 153 | OR                  |
| 59-62 | Графика                            | 154 | QU                  |
| 63    | ? (вопросит. знак)                 | 155 | AV                  |
| 64    | _ (подчеркивание)                  | 156 | TE                  |
| 65-90 | Совпадает со стандартным алфавитом | 157 | IS                  |
| 91-94 | Графика                            | 158 | RI                  |
| 95    |                                    | 159 | DN                  |
| 96    | FRONT                              | 160 | FUEL SCOOPS ON      |
| 97    | REAR                               | 161 | CHART               |
| 98    | LEFT                               | 162 | GOVERNMENT          |
| 99    | RIGHT                              | 163 | DATA ON (#)         |
| 100   | ENERGY LOW                         | 164 | INVENTORY           |
| 101   | RIGHT ON COMMANDER                 | 165 | SYSTEM              |
| 102   | EXTRA                              | 166 | PRICE               |
| 103   | PULSE LASER                        | 167 | (#) MARKET PRICES   |
| 104   | BEAM LASER                         | 168 | INDUSTRIAL          |
| 105   | FUEL                               | 169 | AGRICULTURAL        |
| 106   | MISSILE                            | 170 | RICH                |
| 107   | LARGE CARGO BAG                    | 171 | AVERAGE             |
| 108   | E.C.M. SYSTEM                      | 172 | POOR                |
| 109   | EXTRA PULSE LASERS                 | 173 | MAINLY              |
| 110   | EXTRA BEAM LASERS                  | 174 | UNIT                |
| 111   | FUEL SCOOPS                        | 175 | VIEW                |
| 112   | ESCAPE POD                         | 176 | QUANTITY            |
| 113   | ENERGY BOMB                        | 177 | ANARCHY             |
| 114   | ENERGY UNIT                        | 178 | FEUDAL              |
|       |                                    | 179 | MULTI-GOVERNMENT    |

180 DICTATORSHIP  
181 COMMUNIST  
182 CONFEDERACY  
183 DEMOCRACY  
184 CORPORATE STATE  
185 SHIP  
186 PRODUCT  
187 LASER  
188 HUMAN COLONIAL  
189 HYPERSPACE  
190 SHORT RANGE CHART  
191 DISTANCE  
192 HYPERSPACE (#)  
193 HYPERSPACE RANGE!  
194 MAY DAY! CORIOLIS IN DANGER !  
195 LIGHT YEARS  
196 TECH. LEVEL  
197 CASH  
198 BILLION  
199 GALACTIC CHART  
200 TARGET LOST  
201 MISSILE JAMMED  
202 RANGE  
203 ST  
204 QUANTITY OF  
205 SELL  
206 CARGO  
207 EQUIPT  
208 FOOD  
209 TEXTILES  
210 RADIOACTIVES  
211 SLAVES  
212 LIQUOR/WINES  
213 LUXURIES  
214 NARCOTICS  
215 COMPUTERS  
216 MACHINERY  
217 ALLOYS

218 FIREARMS  
219 FURS  
220 MINERALS  
221 GOLD  
222 PLATINUM  
223  
224 GEM STONES  
225 ALIEN ITEMS  
226 Cr  
227 LARGE  
228 FIERCE  
229 SMALL  
230 GREEN  
231 RED  
232 YELLOW  
233 BLUE  
234 BLACK  
235 HARMLESS  
236 SLIMY  
237 BUG-EYED  
238 HORNED  
239 BONY  
240 FAT  
241 FURRY  
242 RODENT  
243 FROG  
244 LIZARD  
245 LOBSTER  
246 BIRD  
247 HUMANOID  
248 FELINE  
249 INSECT  
250 AVERAGE RADIUS  
251 COM  
252 COMMANDER  
253 DESTROYED  
254 RO  
255 UNIT QUANTITY



Коды, отмеченные знаком ('), вызывают "зависание" программы при попытке их вывести на печать. В символах 163,167,192 на место знака (#) вписывается имя планеты, на которой Вы находитесь.

Символы 7, 10, 11, 13, 14, 35, 119 не имеют печатного обозначения и печатаются как пробел.

До свидания! Успехов Вам!

\* \* \*

Нам пишет Сороченко Сергей Владимирович из г. Норильска:

Здравствуйте, редакция "ZX РЕВЮ"! Я большой поклонник компьютера "SPECTRUM" и Вашего журнала. Я занимаюсь исследованием сложных имитаторов, адвентюрных и стратегических игр. В этом письме я хочу предложить мое описание игры FIGHTER BOMBER, сделанной фирмами Vector Graphics и Activision.

FIGHTER BOMBER - это имитатор самолета-штурмовика. В этой игре прекрасная графика, идентичная графике программы HARD DRIVING, сделанной фирмой DOMARK.

ИФК: Огромное спасибо Сергей Владимирович и за теплые слова и за Ваш труд. Вашу работу мы напечатали в разделе "Советы экспертов". Вы знаете, что мы очень любим программы именно этих жанров и попали в самое сердце.

В Вашей работе нам особенно понравилось то, что Вы не просто "раскрутили" порядок работы с программой, но и критически обсудили некоторые особенности ее работы. Всегда приятно иметь дело с творческим подходом.

KOPP: ... В заключение примите, пожалуйста, мои поправки к Вашему описанию игры ТОМАНАВК. Я не знаю, что на карте обозначено цифрами, но это не Ваши базы, так как при выходе на них я не обнаружил на земле ничего, а при посадке по координатам никакого ремонта не было. Ремонт, заправку и перевооружение производят только на вертолетных площадках, которых в каждом квадрате четыре.

И еще. Повреждение TADS - это не повреждения хвостового оперения, а повреждение прибора, показывающего, какая цель находится в перекрестке прицела.

ИФК: Ну вот, Вы нас и прокололи. А может, мы специально такую пенку сделали? Вдруг они там в Пентагоне тоже "ZX-РЕВЮ" читают? Может, мы хотели оставить полки "Томагавков" без ремонтной базы, а теперь они обо всем догадаются. Ну да ладно, сдерем с них пару сотен миллионов ихних папуасских долларов за это "ноу-хау" и купим себе новый джойстик.

KOPP:... В следующем письме я хочу предложить описание игры OCEAN CONQUEROR фирмы Digital Integration.

ИФК: Огромное спасибо. От души поиграем, как раз и новый джойстик опробуем.

Военнослужащий Похилук М.П. из г. Славгород Алтайского края задает следующий вопрос:

"В нашем небольшом городе образовалось нечто вроде клуба любителей "Спектрума" и у некоторых его членов имеются компьютеры с ПЗУ "ТУРБО-90", у большинства же, как и у меня, ПЗУ стандартное. Преимущества "ТУРБО-90" налицо, и мы хотели бы иметь такое у себя, но специалистов в этой области у нас в городе мы не знаем, а вот сможем набрать программу, которая делала бы то же самое, что ПЗУ "ТУРБО-90", но работала бы где-нибудь в свободной области ОЗУ мы смогли бы".

ИФК: Сделать то же самое, что делает встроенный монитор этой прошивки в принципе можно. Можно "вытащить" монитор из ПЗУ и расположить его в ОЗУ, изменив адресацию в машинных кодах. Но это теряет смысл, так как любая программа-монитор, расположенная в ОЗУ, может находиться как раз в той же области памяти, в которой находится исследуемая программа. Ведь заранее совершенно неизвестно, какая область памяти останется "свободной" после загрузки интересующей программы. Прелесть "ТУРБО-90" в том и состоит, что монитор не в ОЗУ, а в ПЗУ. При этом все ОЗУ свободно и при помощи монитора, находящегося в ПЗУ Вы можете "забираться" в любые уголки памяти и исследовать любые программы, загруженные в ОЗУ. Извлекать же монитор из ПЗУ для

размещения его в ОЗУ нет необходимости, ведь это не самый лучший монитор. Существует достаточно много программ-мониторов, выполняющих все те же функции, что и монитор "ТУРБО-90" плюс еще много других. Например, "MONS-4" ("MONS-3") из пакета программ "DEVPAС". Это довольно старая программа, но она до сих пор является непревзойденной по своим возможностям. "MONS", кстати, устроен так, что может работать в любых адресах. То есть, если Ваша исследуемая программа расположена в верхней части памяти, Вы можете загрузить "MONS" в нижнюю часть и наоборот. К тому же, при помощи монитора "ТУРБО-90" Вы не имеете возможности дизассемблирования программы, а это очень немаловажная функция при исследовании кодов программ.

К сказанному следует добавить еще один любопытный факт. В свое время мы сообщали о мониторе, работающем в ОЗУ, который находится в игре "EQUINOX". По своим функциям и управляющим клавишам он поразительно похож на монитор "ТУРБО-90" (или монитор "ТУРБО-90" похож на монитор "EQUINOX"). Если Вас заинтересует эта информация, то подробности смотрите "ZX-РЕВЮ" N11-12, за 1991 г. на стр. 254.

И, наконец, последнее. Мы очень рады, что в Вашем городке образовалось "нечто вроде клуба любителей "Спектрума". Знаем мы, что то же самое есть и в сотнях других больших и малых городов. Мы много раз уже писали, что мы приветствуем неформальное объединение синклеристов, что это одна из наших главных целей, что мы всегда готовы абсолютно бесплатно и без очереди давать объявления о создании и работе клубов, но почему-то никто этим не пользуется. Разве наши читатели не хотят наладить связь между клубами напрямую? Разве это не нужно? Конечно, кое-какую информацию о своем компьютере Вы получаете из "ZX-РЕВЮ", но мы ведь не можем объять необъятное. Объединяйтесь, обменивайтесь информацией напрямую, мы не боимся конкуренции, а наоборот окажем любую помощь для свободной циркуляции информации между клубами.

\* \* \*

По поводу ПЗУ "ТУРБО-90" пришло еще одно письмо от нашего читателя из Ульяновска - Силантьева М.

Он исследовал эту прошивку методом "теневого ПЗУ". Таким способом это делать очень удобно, если в Вашем компьютере предусмотрена такая возможность - теневое ПЗУ, загружаемое в ОЗУ. Для тех, кто не знаком с этим, сообщаем, что такая возможность есть у компьютеров, имеющих две "линейки" микросхем памяти - отдельно основное ОЗУ и отдельно экранная память. При этом, если после рестарта компьютера, сделать PEEK ADR, где ADR - 0...16383, то читаться будет ПЗУ, но если подать команду POKE ADR, то будет сделана запись в область памяти ОЗУ, являющейся "теневой" для ПЗУ (в данный момент она неактивна и никак себя не проявляет). Можно даже сделать SAVE "SYS"CODE 0,16384 на компьютере, имеющем, например, монитор "ТУРБО-90". А потом загрузить его в свой компьютер, (имеющий стандартный монитор) командой LOAD "SYS"CODE 0. Для того, чтобы переключиться на эту "теневую" прошивку, наш читатель Силантьев М. пользуется тумблером. В "Новосибирском" варианте такое переключение выполняется командой OUT 15,0. При этом переключается триггер, отключающий основное ПЗУ и подключающий теневую область ОЗУ. При этом активным становится загруженный монитор, а основное ПЗУ - пассивным. Теперь можно воспользоваться дополнительными возможностями, заложенными в этой прошивке. Для того, чтобы переключиться обратно на основное ПЗУ, надо опять выполнить команду OUT 15,0. Определить, какое ПЗУ в данный момент является активным, можно, например, по начертанию символов: В "ТУРБО-90" оно несколько отличается. Можно подать команду NEW, тогда будет выведено фирменное сообщение об используемой версии.

После этого отступления мы возвращаемся к письму нашего корреспондента.

KOPP: "У меня обнаружилась одна неприятность. Первые 5 байтов теневого ПЗУ самопроизвольно изменяются после первого же обращения к ним. Если попытаться вывести значение этих ячеек командой:

```
FOR f=0 TO 10: PRINT PEEK f: NEXT f
```

то первые 5 байтов имеют нулевые значения. А если поочередно давать команды:

```
PRINT PEEK 0  
PRINT PEEK 1
```

то печатаются неизвестно откуда взявшиеся числа. При попытке изменить их они самопроизвольно восстанавливаются. Дизассемблирование этих первых 5 байтов дает какую-то абракадабру. Из-за этого неправильно обрабатывается подпрограмма RESET (RANDOMIZE USR 0)

ИФК: Мы проверили то, о чем пишет наш читатель на "Новосибирском" варианте Спектрума и убедились, что этот компьютер ведет себя в точности так же. В чем же тут дело? Может, кто-нибудь из читателей сталкивался уже с подобной проблемой? Или хотя бы знает причину этого явления? Если такие есть, откликнитесь, пожалуйста.

\* \* \*

Дорогие читатели. Вы помните заметку "МАЛЕНЬКИЕ ХИТРОСТИ" в 5-6 выпуске "РЕВЮ" за этот год, где на стр.96 было опубликовано сообщение Радзевича А. А. из г.Нальчика? Оно касалось любопытных эффектов, возникающих при исполнении оператора DRAW. Пока никто не ответил на вопрос "Как это происходит?" Но мы со своей стороны провели небольшой розыск по зарубежным журналам и оказалось, что этой проблемой занимаемся не только мы. Вот что мы нашли в июньском номере журнала "SINCLAIR USER" за 1986 г. (No. 51) на стр. 15. Там приводится несколько иной вариант программы, воспроизводящей такие эффекты.

```
5 LET n=1  
10 PAPER 0: INK 4: BORDER 0: CLS  
20 FOR a=51 TO 10000 STEP 100  
25 LET n=n+1  
30 PLOT 65, 30  
40 DRAW INVERSE n; 120, 120, PI*a  
50 NEXT a
```

Были и другие публикации в этом журнале, например N5,1986, но суть не меняется - все это разные модификации той же идеи. К сожалению, намеков на причины происходящих на экране эффектов там нет. Итак, проблема остается открытой, хоть и приобрела международный характер.

\* \* \*

Нам пишет Лярский С.С. из пос. Березовский Свердловской обл:

Уважаемые коллеги!

Разрешите мне поделиться с Вами своими соображениями по поводу дешифрации портов в "Ленинграде". Если человек впервые собирает такую конструкцию как компьютер и еще если он его сумеет довести до того состояния, в котором он мало-мальски работает, - этот человек, конечно, радуется тому, что у него есть компьютер, что игрушки играют, что программы работают и т. д. Но это только первое время. Дальше наступает непреодолимое желание усовершенствовать свою машину, расширить ее возможности, а если еще у Вас такая "сырая" штука, как первый выпуск "Ленинграда" - бери в руки паяльник, напрягай мозги и "вперед с песней".

Усовершенствований хватит на всю оставшуюся жизнь. Это, конечно же, не упрек авторам Ленинградской версии.

Очевидно, их задачей было сделать наиболее простую, дешевую и более-менее "Спектрум"-совместимую машину. Возможно, им и удалось решить задачу простоты, но что касается дешевизны и совместимости - это большой вопрос. Вот и мне, после сборки и наладки такого "Ленинграда", пришлось столкнуться с тем, что не все игры, а конкретно "BRUCE LEE" работают от джойстика. Я стал советоваться с друзьями и те, кто раньше меня собрали и немного освоились с этим компьютером, посоветовали подать логический "0" на три старших разряда порта джойстика. Но после того, как я это проделал, некоторые игры,

например, "EXOLON", зависали после загрузки и запуска Бейсик-загрузчика. Тогда я стал разбираться с Монитором, зашитым в ПЗУ и понял, что при существующей дешифрации портов в "Ленинграде" таких накладок мне не избежать. Исходя из этих соображений и еще прочитав разработку НТК "ПЛЮС" по интерфейсам "Спектрума", родилась схема, которую я Вам и предлагаю (приведена на следующей странице).

В своем "Спектруме" я исключил две микросхемы 555КП1, которые использовались для чтения клавиатуры, магнитофона и джойстика. Микросхему 555ТМ9 я оставил как выводной порт с адресом "FE", добавив две микросхемы 555АП5 для чтения клавиатуры и магнитофона и с учетом дальнейшего расширения возможностей моего компьютера добавил параллельный порт 580ВВ55, включив его по рекомендациям НТК "ПЛЮС", только с более жесткой дешифрацией портов. В этой схеме я использовал свободные инверторы "Ленинграда", а также элементы, которые выделяли сигналы IORD' и IOWR' (D14.1, D14.4).

Сигналы по схеме:

C2 - запись в порт "FE", C3 - чтение порта "FE", сигнал C1 - вспомогательный.

Если не устраивает жесткая дешифрация, можно исключить элементы D41.1 и D42.1 и тогда вместо сигнала C1 на входы 4 - D42.2 и 12 - D42.4 подать сигнал IORQ'. Адреса внешних портов MC 580ВВ55 будут следующие:

"1F" - чтение, запись порта "А"(порт джойстика);

"3F" - чтение, запись порта "В";

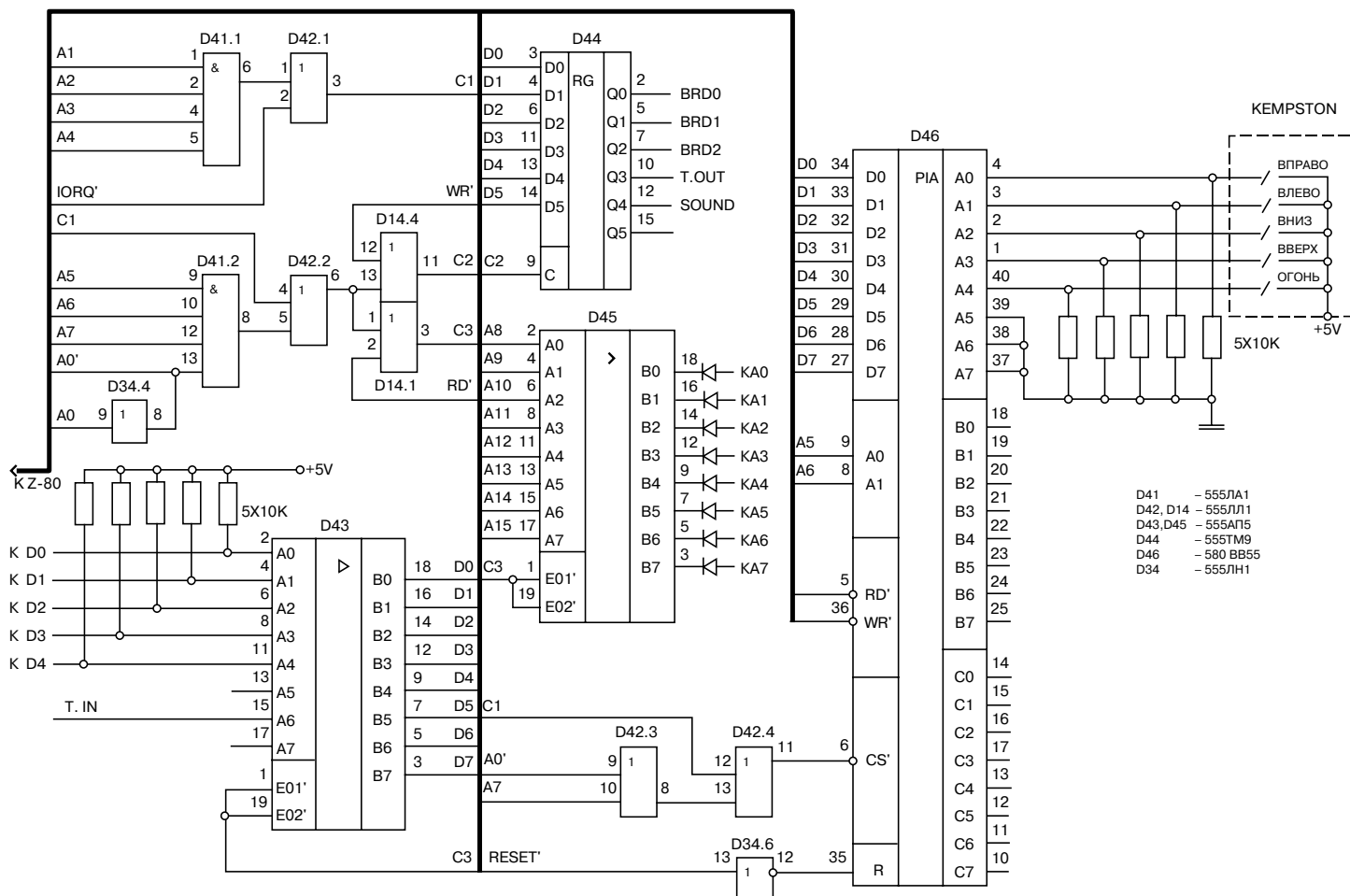
"5F" - чтение, запись порта "С";

"7F" - регистр управляющего слова.

Вот, пожалуй, и все по поводу этой схемы.

ИФК: Нас, как и большинство наших читателей, очень интересует все, что касается совершенствования компьютеров в плане повышения их совместимости с фирменными моделями. Таким работам мы даем "зеленую улицу" и принимаем их с особой благодарностью.

Приносим свои извинения нашим читателям за нестандартный вид схемы (мы уже говорили о сложностях исполнения графических работ в текстовом редакторе), а для работы в графической среде у нас не хватает кадров и времени. Так, например, инвертированные сигналы вместо общепринятой черты над обозначением сигнала у нас отмечены знаком (') - апостроф.



Нам пишет Стас Рубцов из г. Улан-Удэ.

Здравствуйте, Инфорком!

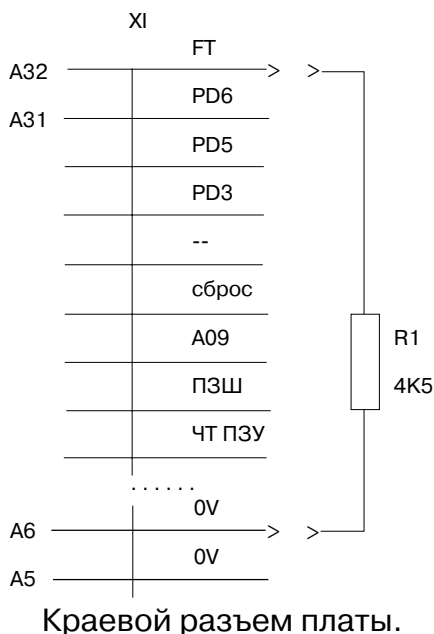
Большое спасибо за очень нужный и интересный журнал. Подписчиком я не являюсь, но регулярно пользуюсь Вашими разработками и работаю с журналом. Беру я его у знакомого, "товарища по Спектруму", если можно так сказать. Он является Вашим подписчиком с самого первого выпуска. Так что мне, можно сказать, повезло.

ИФК: Спасибо на добром слове, Стас. Мы очень рады, что у Вас есть хороший друг. На Вашем примере еще раз посоветуем всем, кто нас читает, объединяться и еще раз объединяться. Новые повышения цен на печать в конце этого года поставят Вас перед необходимостью существенных затрат на вздорожавшее "РЕВЮ" и только единение поможет выжить и Вам и нам.

КОРР: ... Работаю я на ZX совместимой машинке "Дельта-С". У нас в городе "Дельты" разошлись по рукам за две недели - по самым предварительным сведениям - три партии по 800 штук. Были ли еще партии, я не знаю. Во всех Ваших журналах нет ни одного слова о компьютере "Дельта-С" и это может говорить либо об ограниченном его распространении либо о его полной совместимости с фирменной машиной. Ни то ни другое не соответствует действительности. "Дельта-С" распространена и в Томске, и в Иркутске и в других городах, в Москве я видел в фирменном "Московском" (на площади трех вокзалов). Работаю я с "Дельтой-С" год, и хочу сказать несколько слов об этой машинке. По моим сведениям, "Дельта-С" выпускается по меньшей мере на трех заводах. У меня вариант Курского завода (305040, Курск, ПО "Прибор"). Оформление машины не очень хорошее, в частности, "жесткая" контактная клавиатура, не очень качественно сделан вывод звука, неважный внешний вид. Однако технически машина прекрасная. Имеет встроенный ZX-интерфейс II, буферизированный выход на ZX-принтер. Несовместима "Дельта-С" с фирменным "Спектрумом" только в одном. При наличии в программе вариаций на тему "POKE 23570,16" - вывод заголовков типа "BYTES..." и др. на принтер, из-за ошибки в схеме не проходит

номер с подпрограммой ПЗУ "COPY-LINE" по адресу 0F0C-0F13 (COPY-L-2), т.е. если принтер не подключен, программа "виснет".

Выход очень прост. Надо "подключить" принтер, дать ответ из принтера по линии "BUSY" - "занят", что принтер не готов к приему информации. Но так как не у всех есть фирменный ZX-принтер, советую всем столкнувшимся с этой проблемой просто подать на вход "BUSY" уровень "ЛОГ 0". Я поступаю так: соединяю на разъеме вывод A32 "ГТ" (готовность принтера), соответствующий сигналу "BUSY" с выводом A5(A6) т.е. с общим проводом, через резистор 4.5 кОм, при этом не надо разбирать компьютер. После такой переделки все программы идут, как на фирменной машине.



Звук можно вывести на усилитель, для этого на свободном месте (на задней стенке компьютера) монтируем гнездо СГ-5, вскрываем компьютер, находим единственный проводник, ведущий к звукоизлучателю (от транзистора на плате) и в тоже самое место припаиваем минусовый вывод конденсатора К50-3 10 мкФ х 12 V, плюсовой вывод припаиваем к контактам "1" и "4" на СГ-5, а к контакту "2" на том же разъеме припаиваем "массу" компьютера (перемычку от контакта "2" разъема "+5В" либо от контакта "2" разъема RGB).

Хотелось бы, чтобы изготовители "Дельты-С" учли эти вопросы в своих разработках.

ИФК: От всей души присоединяемся к этому пожеланию и думаем, что независимо ни от чего Ваши рекомендации помогут многим любителям повысить совместимость своих машин.

КОРР: ... Я прошу извинить меня за то, что получилось так, как будто у "Дельты-С" есть существенные недостатки. Это прекрасная машина и это видно даже по печатной плате, жалко только, что к нему не прилагается принципиальная схема. Может быть его производители учтут и этот момент?

\* \* \*

По вопросам совместимости различных версий "Спектрума" нам пишет из Йошкар-Олы Москвин Денис Егорович:

Адресую свое письмо в раздел "Форум" журнала "ZX-РЕВЮ", так как хочу поделиться своим открытием в области совместимости компьютера "Пентагон 48" с фирменным "ZX-Spectrum".

В последнее время 48-е машины вытесняются 128-ми, но мне еще раз хотелось бы остановиться на 48-м "Пентагоне".

В ваших журналах опубликовано множество статей, посвященных совместимости "Пентагона", но среди них не оказалось статьи о том, почему не работают на "Пентагоне"

такие всем известные программы, как "Cobra Stalone", "Renegade-1", "Str. Hawk" и т.п. Мне бы хотелось, чтобы этот пробел был восполнен.

Эти программы и им подобные не "идут" потому, что в "Пентагоне" не коммутируется сигнал WAIT (ожидание). Этот сигнал, как оказалось, в фирменном компьютере используется для разрешения конфликтов памяти, вызванных регенерацией нижней области ОЗУ, где хранится содержимое экрана.

Скоммутировать сигнал WAIT можно с помощью триггера. В "Пентагоне" это микросхема TH2 (DD11), расположенная в нижней части платы компьютера рядом с микросхемой ЛП5, с которой выводятся цвета (R, G, B) и яркость (Y).

Если сигнал WAIT подключить согласно приведенной ниже схеме, то можно будет смело загружать "Renegade", т.к. она будет работать!

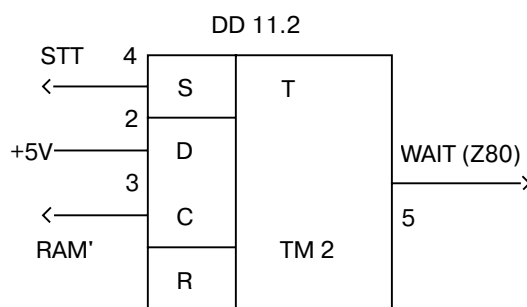


Рис. 2

Здесь на схеме:

STT - это вывод 12 м/с DD1.3 (ЛН1 - на плате она одна);

RAM' - это вывод 11 DD18.1 (ЛА3 - ее легко найти, т.к. именно к выводу 11 подключена перемычка, уходящая под процессор);

WAIT - это вывод 24 микропроцессора.

Таким образом, если подытожить все исправления "Пентагона", опубликованные в Ваших журналах, то "пентагоновская" версия 48-го станет полным аналогом фирменного SPECTRUMa.

Данная доработка опробована на 5-ти компьютерах. Опыт показал, что при подключении сигнала WAIT к процессорам КР 1858 ВМ1 (советский аналог Z80) программы Cobra и Renegade все равно не работают. Видимо, это происходит из-за неправильной системы прерываний у данного процессора.

\* \* \*

Нам пишет Гасин Руслан Владимирович из Магадана.

Здравствуйтесь, уважаемая редакция!

Я являюсь недавним Вашим читателем (подписчиком). Основной целью подписаться на Ваш журнал было желание научиться работать с машинными кодами, описание программ (в основном системных), взлом (а вернее техника защиты) программ.

Ну что ж. В принципе я удовлетворен. Мне понравилась информация, представленная Михаиленко В.С. Все это хорошо. Но я думаю, что многие согласятся со мной.

Я предлагаю хоть чуть-чуть рассмотреть технику защиты программ на диске (в системе TR-DOS). Конечно, можно заявить, что с дисководом работают не так уж и много пользователей ZX-SPECTRUM. Я сам только подумываю о нем (дисковом). Но я и не требую доскональных данных о защите (на дискетах). Отталкиваясь от Ваших же принципов (дать общую методику, идею, направление, а люди сами додумают), можно дать лишь общие сведения. Например, такую. На диске есть директорий, он "говорит" о длине, расположении файлов и т.д. Так почему бы не сделать так, чтобы на диске был файл, но операционная система не могла ни вывести его название, длину в директорий, ни запустить на выполнение без разрешения пользователя. Конечно же, все это выполнимо, и что-то можно сделать? Но

не хватает сведений о TR-DOS (ее переменных и т. д.)

Естественно, это не совсем защита от взлома, а скорее от несанкционированного доступа к программе.

Очень прошу, напечатайте мое письмо. Вдруг, кто-то откликнется и расскажет, что знает. А от Вас, если это произойдет, зависит только вопрос - напечатать или нет. Но я думаю, что одну страницу не грех будет отдать.

Надеюсь, что мое письмо не оставит Вас равнодушными.

ИФК: Уважаемый Руслан Владимирович, здесь двух мнений быть не может. Голосуем "За" единогласно и безусловно дадим все, что сочтут нужным сообщить наши читатели. Но, кажется на горизонте появилось кое-что поинтереснее. Мы имеем в виду операционную систему IS-DOS. Мнение знакомых специалистов, опробовавших ее, единодушное - ЭТО БЛЕСК, который действительно превращает "Спектрум" в полупрофессиональную ЭВМ. Впрочем, пока СТОП... мы находимся в начале пути, информация пока скудна, но не исключено, что следующий год пройдет под знаком именно этой системы.

\* \* \*

Нам пишет Алексей Жильцов из Самары:

Я хочу сообщить Вам об особенностях игры "Death Wish-3". Если встать в доме под окно и нажать "W", то на экране появится план улицы и прицел, двигая которым и стреляя, можно расстреливать пробегающих бандитов. Только нужно следить, чтобы сзади никто не подошел, иначе будут проблемы.

\* \* \*

Многие наши читатели жалуются, что мы мало внимания уделяем аппаратной части. Это так, но если Вы вспомните, мы уже писали, что у истоков "ИНФОРКОМа" и "ZX-РЕВЮ" стоял коллектив по своей квалификации скорее преподавательский, чем технический. Поэтому аппаратные вопросы мы всегда отдавали другим. Сначала нам помогал в этом НТК "ПЛЮС", а сейчас, когда эта фирма увы больше не работает, мы можем только опираться на Ваши письма, уважаемые читатели. Писем с программами или программными вопросами намного больше, чем с аппаратными. И тем ценнее для нас каждое письмо, в котором так или иначе затрагиваются аппаратные проблемы. И коль уж сегодня мы дали некоторые рекомендации по возможным доработкам "Дельты-С", то очень кстати будет и сообщение нашего читателя из г. Норильска - Галушака Александра. Вот что он пишет:

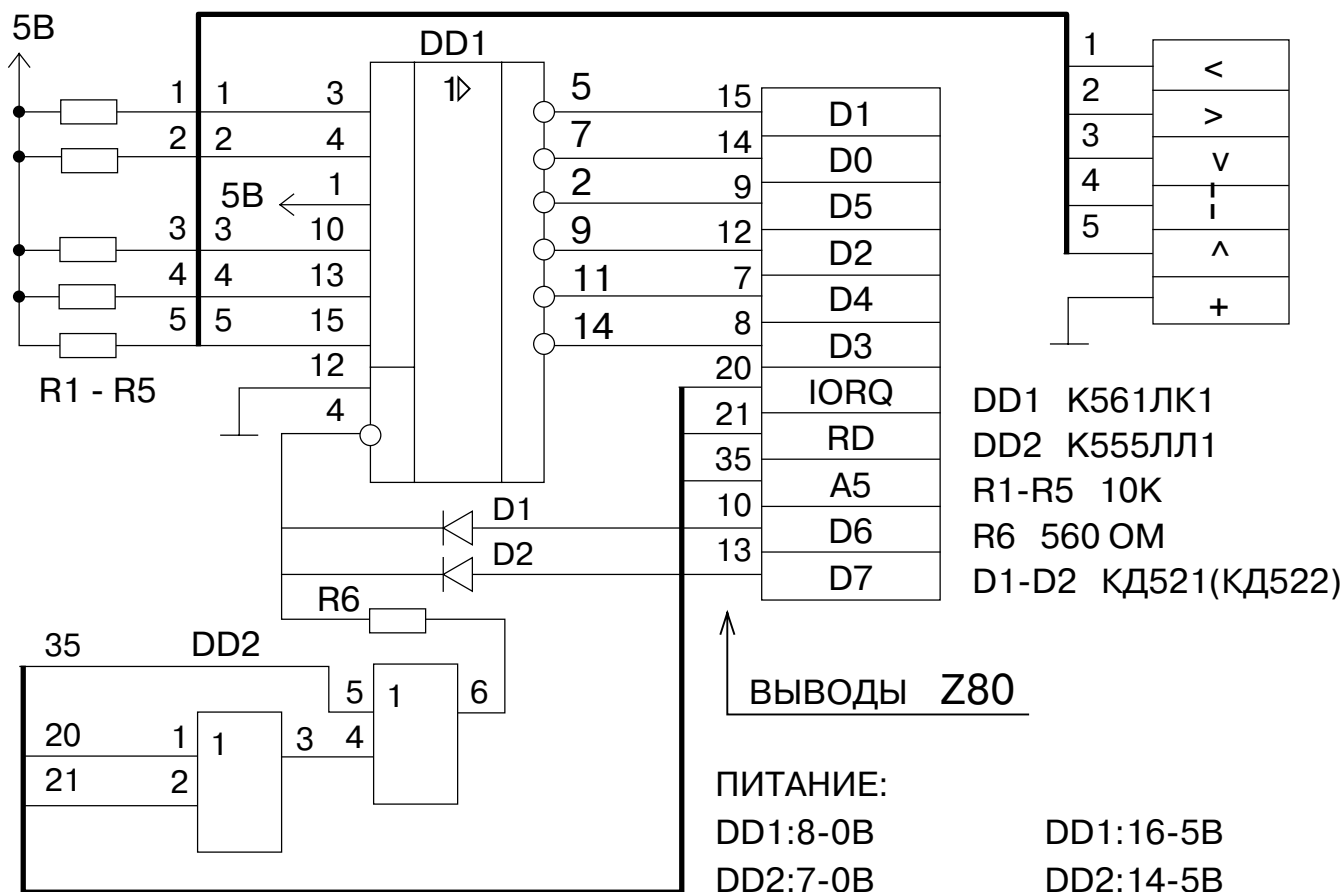
"В Норильске много владельцев ПК "Дельта-С", но многим из них не нравится, что в "Дельте" нет порта KEMPSTON-джойстика. Я подозреваю, что в СНГ существует немало типов компьютеров без KEMFSTON-джойстика.

Я подключил джойстик к "Дельта-С" своему товарищу и хочу поделиться опытом с читателями "ZX-РЕВЮ".

Собран интерфейс KEMPSTON-джойстика на двух микросхемах, установленных на небольшой печатной плате, и соединяется с Z80 тринадцатью проводами, включая питание.

Я обратил внимание, что на "Дельте-С" вместо двух микросхем 2764 используется одна микросхема 27128, а место для второй микросхемы - свободно. Поскольку на ПЗУ выводятся почти все необходимые для интерфейса джойстика сигналы (кроме IORQ и RD), то я разработал печатную плату в расчете на то, что она будет установлена на место неиспользуемой микросхемы 2764 (ПЗУ-1) и соединяться с основной платой компьютера короткими гибкими проводами. Расположенный таким образом интерфейс размещается внутри корпуса компьютера ничему не мешая. С разъемом KEMPSTON-джойстика дополнительная плата соединяется шлейфом (6 жил) необходимой длины.





Возможны и другие решения. В частности, если в ПК используются 2 микросхемы 2764, то можно использовать эту же печатную плату, расположив ее над процессором Z80, и соединив непосредственно с выводами Z80.

ИФК: Александр готов ответить на любые вопросы, возникшие при изготовлении интерфейса и дает свой адрес:

Просьба вкладывать конверт с обратным адресом.

Еще, если есть такие вопросы, могу выслать рисунок печатной платы для ZX-LPRINT-3 (описание которого было в "РЕВЮ-91" N 7-8). Этот интерфейс принтера мною собран и благополучно работает уже год".

\* \* \*

652692, Кемеровская обл. , Гурьевский р-н, с. Горскино, Матвееву Ю.Н.

В заключение передаем уважаемому корреспонденту пламенный привет от редактора раздела "К.Н." Матвеева Ю.А.

\* \* \*

Как добро и зло, как плюс и минус не могут существовать друг без друга, так не могут существовать отдельно друг от друга вопросы защиты и взлома программ. Ситуация складывается примерно так. Появляются новые разновидности защиты, существующие какое-то время, затем они раскалываются, появляется литература на тему о защите и ее снятии. Новые варианты защиты разрабатываются уже на основе систематизированного материала по снятию защиты. Появляются новые, более изощренные способы взлома и процесс продолжается дальше, замыкая очередной виток эволюционной спирали. Кто сделает следующий ход в этой бесконечной игре?

Мы уже печатали материалы по защите программ, хорошо осознавая тот факт, что этим материалом пользуются как те, кто защищает свои программы, так и те, кто их после этого взламывает. Сегодня мы предлагаем материал, присланный Григорием Борисовичем Лупповым из города Кирова. На наш взгляд им как раз могут результативно воспользоваться разработчики защиты, в то же время для "взломщика" - здесь всего лишь общие фразы.

Вот что пишет наш корреспондент: "Я в общем-то серьезно "Спектрумом" не занимаюсь - больше спец по "Вектору 06Ц". Однако имею "Пентагон 48К" и с удовольствием читаю "ZX-РЕВЮ", когда удастся выпросить его у знакомых "спектрумовщиков". В номере 1-2 за 1992 г. увидел статью по защите, а защита программ - дело одинаково актуальное для всех бытовых персональных ЭВМ в конфигурации с магнитофоном. Хочу предложить Вашему вниманию свой взгляд на этот вопрос.

В ходе долгих размышлений и поисков вариантов защиты программ для "Вектора", я пришел к выводу, что действительно, для магнитофонных версий ПЭВМ надежной защиты нет! Все способы защиты, которые существуют на "Спектруме", это своего рода искусные, красивые трюки авторов, крайне недолговечные. Эти трюки в целом проблемы не решают. Внешне это выглядит так: автор, петляя и изворачиваясь, как заяц-русак, старается сбить со следа волка-взломщика. Результат один - заяц погибает.

Нужно применять другой подход. Защита должна быть простая, относительно несложно ставиться, но самое главное - она должна быть трудоемкой для снятия и легко модифицируемой (на случай, если одну модификацию сняли).

Предлагаю классическую схему: основная программа в своем формате (отличном от стандартного), впереди идет закодированный загрузчик с декодировщиком в формате "Спектрума" и не обязательно с автозапуском.

Слабость схемы очевидна - декодер всегда можно просмотреть и разобраться в принципе его работы. Поэтому процесс декодирования должен включать в себя несколько основных положений.

1. Декодирование с использованием всего адресного пространства компьютера.
2. Декодирование с подвязкой по времени.
3. Перевод на декодированный загрузчик должен быть неочевидным (чтобы нельзя было вставить переход на взламывающую подпрограмму после декодировки), он должен быть включен в закодированный текст и декодироваться последним, то есть декодирование идет от старших адресов к младшим.

4. При декодировании используются псевдослучайные числа. В начале работы декодер забивает все свободное пространство ОЗУ (и видео-ОЗУ в том числе) псевдослучайными числами. Генератор таких чисел должен быть:

- а) последовательным - это значит, что n-ое число можно получить только после получения всех предыдущих чисел;
- б) параметрическим - это значит, что получаемая последовательность будет зависеть от параметров генератора, то есть генератор легко перенастраивается.

Числа в память должны заноситься не подряд, а либо через какое-то смещение (не

равное 1) либо по случайным адресам (это лучше), например, первые два байта из генератора определяют адрес, третий байт - данные. Естественно декодер и загрузчик меняться не должны.

Затем декодер начинает непосредственно свою работу. Декодирование идет сверху - вниз. Для получения очередного байта над ним нужно проделать ряд операций с использованием данных из ОЗУ, адресуемых несколькими маркерами. Маркер - циклически бегающий по адресу декодера; маркер - циклически бегающий по всему ОЗУ и ПЗУ (кроме декодера и загрузчика); случайный маркер - обеспечивающий определенную защиту от резидентных взломщиков. Время-зависимые данные (регистр R или ячейка-таймер) обеспечивают защиту от эмулирования декодера пиратским взломщиком.

После декодирования управление передается на загрузчик. Так как содержимое памяти загрузчику известно, то он может загружать (декодировать) программу, используя содержимое памяти (однако снижается при этом технологичность и модифицируемость защиты). Загрузчик должен иметь гибкую структуру, чтобы загружаемая программа легко перехватывала управление в любое время. Лучше всего использовать блочный формат: вся программа разбивается на блоки по 32, 64 или 256 байтов; в начале каждого блока идет заголовок с адресом загрузки, в конце - байт-паритет КС.

Программа может иметь и внутреннюю защиту, включающую проверку наличия в памяти меток о работе декодера и загрузчика и т. д. Это можно сделать на авторских программах, но не на чужих.

Для большей модифицируемости можно применять плавающую константу чтения, которая либо изменяется по определенному алгоритму, либо содержится в таблице.

Технологичность защиты обеспечивается тем, что автору нужно только раз получить закодированный особым образом загрузчик и использовать его затем на разных программах. Как только защиту вскроют (ничто не вечно под Луной), можно изменить параметры генератора случайных чисел и таблицу констант чтения. "Пирату" придется повторять весь трудоемкий процесс вскрытия. Трудоемкость определяется тем, что нужно эмулировать ОЗУ так, чтобы там был и набор псевдослучайных чисел и пират-эмулятор декодера. Это можно сделать с использованием дисководов или дополнительного ОЗУ (в 128К версии), однако при этом будут изменяться время-зависимые данные и процесс декодировки нарушится. Если программа рассчитана на версию компьютера с музыкальным процессором, то последний можно использовать в качестве генератора время-зависимых данных.

Не могу ничего посоветовать относительно борьбы с "MAGIC", т.к. не знаю тонкостей работы в этом режиме. Однако многие программы после обработки кнопкой "MAGIC" не работают, значит придумать что-то можно".

Материал этот чисто теоретический, у многих может вызвать вопрос типа: "Это все складно звучит, но как же все это практически реализовать? Где конкретные примеры, демонстрирующие изложенные принципы?" Примеров действительно нет, так как любой практический пример приведет к снижению во много раз долговечности защиты, сведя на нет все усилия автора программы. Хорошая теория - это уже больше половины дела. А как это конкретно воплотить в жизнь - как говорится, думайте сами, решайте сами...

ИФК: Но Григорий Борисович, видимо предвидя подобные вопросы, все же продолжает письмо конкретным примером, предлагая защитный трюк, связанный с блоками без заголовка.

KOPP... Копиры определяют конец такого блока, если при выходе из подпрограммы LOAD\_BYTES установлен флаг Z. Подпрограмма загрузки бита имеет вид:

```
05E3 CALL #05E7
05E6 RET NC
05E7 LD A,#16
05E9 DEC A
05EA JR NZ, #05E9
05EC AND A
05ED INC B
05EE RET Z
```

Если долго не появляется смена уровней (фронт сигнала), то подпрограмма досрочно возвращает управление. Такая ситуация возникает как раз при загрузке файлов без заголовка, если задана фиктивная (большая) длина. Трюк заключается в том, чтобы выгружать такой блок как можно более длинным. То есть сначала мы рассчитываем байт-паритет (КС), идущий в конце каждого блока, и записываем его в конец файла в виде данных, а выгружаем файл с максимальной длиной. При загрузке все будет нормально, так как мы знаем реальную длину блока и загрузим его без проблем. У тех же утилит, которые не знают реальную длину (программы-копировщики), возникнут проблемы со свободной памятью. Конечно, можно обрезать такой файл, однако нет гарантии, что обрезали только ненужные данные, или что загрузчик не проверяет целостность файла.

ИФК: Но и на этом письмо Григория Борисовича не заканчивается. Он предлагает читателям еще несколько слов насчет формата записи байта на магнитофон.

Он пишет: "В "Векторе" используется следующий формат: 1 заменяется на 01, а 0 - на 10 (так называемый каскадный код). При этом длина битов одинаковая, кроме того, не надо так точно выдерживать междубитовую и междубайтовую длину. Я уже пробовал эмулировать такой формат на "Спектруме". Результаты - положительные. Они хороши еще и тем, что на "Спектруме" можно при этом формате достичь в 2-3 раза большей скорости записи, по сравнению со стандартной, без каких-либо особых проблем (нужен только загрузчик). Это готовый турбо-формат, причем с изменяемой константой записи, то есть Вы можете регулировать ее в широких пределах. Константа чтения автоматически определяется при загрузке. Для защиты от помех используется дублирование. Каждый блок (по 256 байт) задублирован. И помехоустойчивость можно улучшить, если дублировать не отдельный блок, как сейчас (по принципу: блок 1, дубль 1, блок 2, дубль 2, ...), а всю программу (блок 1, блок 2, блок n, дубль 1, дубль 2, ... дубль n). Кроме того, возможно сделать (а на ВЕКТОРЕ уже сделано), что сбой в каком-то блоке не вызывает сброс всей программы (на "Спектруме" одна помеха вызывает сбой всего файла), так как программа разбита на мелкие блоки (как отсеки в подводной лодке). Вы можете остановить магнитофон, отмотать чуть назад и повторно загрузить сбойное место. Процесс загрузки отображается на экране в виде карты загрузки. Очень наглядно видно, где произошел сбой. Кроме того, есть возможность, пожертвовав сбойным участком в 32 байта, загрузить оставшуюся информацию.

На "Векторе" загрузчик занимает 256\*2 байт, на "Спектруме" его можно сделать короче".

ИФК: Если у читателей есть интерес к тому, что говорилось об этом формате, то Григорий Борисович предлагает выслать текст загрузчика и текст подпрограммы записи в таком формате на магнитофон.

Уважаемый Григорий Борисович! Мы искренне благодарим Вас за предоставленный материал. Думаем, что это мнение разделят наши читатели.

# ПРОФЕССИОНАЛЬНЫЙ ПОДХОД

(С) Збитнев В. А., 1992, 1993. (г. Новосибирск)

## Применение статического генератора случайных чисел на примере программы "ELITE"

Вероятно, уже все знают о том, что такое RND. Во многих игровых и учебных программах этот оператор используется для того, чтобы поведение компьютера в некоторых ситуациях было непредсказуемым. Эту роль прекрасно исполняет генератор случайных чисел на ZX-SPECTRUMе. Он относится к типу динамических, то есть таких, которые изменяют свое состояние независимо от того, когда вызывается этот генератор. Свое значение "спектрумовский" RND вычисляет, используя переменную SEED (23670 DEC), которая изменяется с течением времени.

Но так бывает не во всех компьютерах. Многие, наверное, знают работу RND на других версиях Бейсика для компьютеров типа "Ямаха" (MSX), IBM. Сколько бы раз Вы не вызывали бы RND в своей программе (например для рисования звездного неба), он выдает одни и те же числа после команды RUN, RANDOMIZE (в случае рисования звездного неба одна и та же картина будет на экране, если Вы остановите программу и снова запустите ее через RUN).

Такой вид RND-генератора называется статическим. Он плох для непредсказуемых действий, но он прекрасно может работать в качестве хранилища огромного объема данных, не требующих в памяти места. Но самое главное - эта информация будет постоянной, сколько бы раз эти данные не были бы затребованы. Простейший пример такого RND - генератора на ассемблере

```
RND: LD     A, (R1)
      LD     D, A
      LD     A, (R2)
      LD     (R1), A
      ADD    A, D
      LD     D, A
      LD     A, (R3)
      LD     (R2), A
      ADD    A, D
      RLCA
      LD     (R3), A
      RET
```

R1, R2, R3 - переменные

В регистре A будет содержаться случайное число (0-255). Но подобный генератор случайных чисел все время будет генерировать очередные случайные числа. Теперь создадим программу, подобную RANDOMIZE в Бейсике.

```
RAND: LD     A, 5
      LD     (R1), A
      LD     A, 90
      LD     (R2), A
      LD     A, 37
      LD     (R3), A
      RET
```

Теперь, после вызова подпрограммы RAND некий указатель встаёт на начало некой таблицы, в которой хранится бесконечное количество байтов, которые и извлекаются из неё

подпрограммой RND. Где это можно использовать? К примеру, в хорошо известной игре "ЖИЗНЬ" ("LIFE") нужна матрица со случайно расположенными в ней клетками. Вводить через DATA весь массив неудобно, медленно и не практично. Если заполнять массив через обыкновенный RND, то возникает проблема с тем, как снова воспроизвести картину, если это понадобится. Опять прибегаем к DATA. Но есть более практичный способ: задание массива через статический RND, который будет выдавать те же самые значения, что и в первый раз, стоит только вызвать RAND.

В игровых программах с большим количеством комнат таким путем можно запоминать все комнаты, как комбинацию трех (или другого числа байтов), в данном случае - 5, 90, 37. Стоит их только изменить и Вы имеете опять новую комнату. Программа "Demons" использует как раз такой способ, она чем-то напоминает "ЖИЗНЬ", но выглядит в 7 раз красивее и эффектнее. Попробуйте, не пожалеете!

### Программа 1 Demons

|       |      |           |                                                        |
|-------|------|-----------|--------------------------------------------------------|
|       | ORG  | 30000     |                                                        |
|       | CALL | RAND      | ;Сброс генератора случайных чисел в исходное положение |
|       | CALL | RNDSC     | ;Заполнение экрана атрибутами через RND                |
| NXTPG | LD   | IX, 22528 | ;Старый массив                                         |
|       | LD   | HL, 31000 | ;Новый массив                                          |
|       | LD   | BC, 768   | ;768 байтов - размер области экранных атрибутов        |
| LOOKS | LD   | A, (HL)   |                                                        |
|       | INC  | A         |                                                        |
|       | AND  | 7         |                                                        |
|       | CP   | (IX+1)    | ;Если слева, справа, сверху или снизу может кто-то     |
|       | JP   | Z, SEAT   | ; "съесть" этот атрибут, то пусть "съедает".           |
|       | CP   | (IX-1)    |                                                        |
|       | JP   | Z, SEAT   |                                                        |
|       | CP   | (IX+32)   |                                                        |
|       | JP   | Z, SEAT   |                                                        |
|       | CP   | (IX-32)   |                                                        |
|       | JP   | Z, SEAT   |                                                        |
| EATED | INC  | HL        | ;Продолжить обработку массива                          |
|       | INC  | IX        |                                                        |
|       | DEC  | BC        |                                                        |
|       | LD   | A, B      |                                                        |
|       | OR   | C         |                                                        |
|       | JP   | NZ, LOOKS |                                                        |
|       | CALL | COPY      |                                                        |
|       | RET  |           |                                                        |
| SEAT  | LD   | (HL), A   | ; "Съедаем" текущий атрибут                            |
|       | JR   | EATED     |                                                        |
| COPY  | HALT |           |                                                        |
|       | LD   | HL, 31000 | ;Копирование нового массива в старый                   |
|       | LD   | DE, 22528 |                                                        |
|       | LD   | BC, 768   |                                                        |
|       | LDIR |           |                                                        |
|       | RET  |           |                                                        |
| RNDSC | LD   | HL, 31000 | ;Заполняем новый массив байтами с числовым значением   |
|       | LD   | BC, 768   | ;от 0 до 7                                             |
| FILL  | CALL | RND1      |                                                        |
|       | AND  | 7         |                                                        |
|       | LD   | (HL), A   |                                                        |
|       | INC  | HL        |                                                        |
|       | DEC  | BC        |                                                        |
|       | LD   | A, B      |                                                        |
|       | OR   | C         |                                                        |
|       | JP   | NZ, FILL  |                                                        |
|       | CALL | COPY      | ;Копирование массива в старый массив                   |
|       | RET  |           |                                                        |

```

RAND      LD      A, 5           ; 5, 90, 37 - байты, означающие параметры экрана
          LD      (R1), A
          LD      A, 90
          LD      (R2), A
          LD      A, 37
          LD      (R3), A
          LD      HL, 16384
          LD      DE, 16385
          LD      BC, 6144
          LD      (HL), 255
          LDIR
          RET

RND1      LD      A, (R1)        ; Генератор случайных чисел
          LD      D, A
          LD      A, (R2)
          LD      (R1), A
          ADD     A, D
          LD      D, A
          LD      A, (R3)
          LD      (R2), A
          ADD     A, D
          RLCA
          LD      (R3), A
          RET

R1         DEFB  0              ; Исходные значения параметров
R2         DEFB  0
R3         DEFB  0

```

Следующая область применения - формирование статичной (неменяющейся) информации. Возьмем к примеру игру "ELITE". Откуда компьютер с таким небольшим объемом памяти берет названия и координаты всех 256 звезд, в каждой из восьми галактик, да ещё и неизвестно сколько всего этих галактик. Где хранятся данные о ценах на каждой из планет, её техническом уровне, общественном строе и населении.

Разумеется, это работа статического RND. Если Вы наберете программу 2 на Ассемблере или 2.1 на БЕЙСИКе, то сможете получить данные о всех названиях планет и их координатах. Остальная информация каким-то образом хранится в 6 основных байтах и может быть образована одним из 4 RND, вызываемых последовательно для каждой звезды. Может быть, удастся расшифровать и остальные данные.

Итак, статический RND может хранить в себе информацию. Можно, к примеру, сделать слоги русскими, задействовать остальные байты для русских имен, для пола, возраста, адреса и пожалуйста: у Вас есть картотека на 10000 человек, работающих на космическом корабле. Статический RND может быть применим и для динамической информации. Примером может служить игра SENTINEL, в которой все ландшафты имеют свой номер, а задание их в памяти невозможно - слишком расточительно. Но в процессе игры Вы можете некоторым образом изменять эту информацию уже в матрице, созданной этим RND.

#### Программа 2.1

Заполняет массив данными о планете, начиная с адреса 40000 в следующем формате:

```

| Имя звезды | X | Y |
| 8 байтов  |16.|16.| = 10 байтов на планету

```

```

          ORG     30000
          CALL    RAND
          LD      B, 0           ; 256 звезд
STARS     PUSH    BC
          CALL    LDNAME        ; Считать имя в NAME

```

|        |                |                                                      |
|--------|----------------|------------------------------------------------------|
|        | CALL SAVNM     | ; Записать имя из NAME в массив, на который          |
|        | POP BC         | ; указывает MASSV                                    |
|        | DJNZ STARS     |                                                      |
|        | RET            |                                                      |
| LDNAME | LD HL, NAME    | ; Позиция в имени указывается переменной POS         |
|        | LD (POS), HL   |                                                      |
|        | LD B, 9        | ; Очистили 9 байтов                                  |
| CLEAR  | LD (HL), 0     | ; Имя будет равно 8 символам                         |
|        | INC HL         |                                                      |
|        | DJNZ CLEAR     |                                                      |
|        | LD A, (D0+1)   |                                                      |
|        | LD (D1), A     | ; Запишем координату Y                               |
|        | LD A, (D2+1)   |                                                      |
|        | LD (D3), A     | ; Координата X звезды                                |
|        | LD A, (D0)     | ; Условие, при котором 6-ой бит указывает на то,     |
|        | AND 64         | ; в слове нет последнего слога                       |
|        | PUSH AF        |                                                      |
|        | CALL LDSLG     | ; Три слога                                          |
|        | CALL LDSLG     |                                                      |
|        | CALL LDSLG     |                                                      |
|        | POP AF         |                                                      |
|        | JR Z, RND2     | ; Если нет последнего слога, то уход на холостой RND |
| LDSLG  | LD A, (D4+1)   | ; Условие отсутствия слога:                          |
|        | AND 31         | ; номер слога = 0, т.е. слог "AL" - невозможен       |
|        | JR Z, RND2     |                                                      |
|        | LD HL, SLOGS   |                                                      |
|        | ADD A, A       | ; Номер слога*2                                      |
|        | LD E, A        |                                                      |
|        | LD D, 0        |                                                      |
|        | LD HL, DE      | ; Находим позицию слога в SLOGS                      |
|        | LD A, (HL)     | ; Считываем первый символ слога                      |
|        | INC HL         |                                                      |
|        | PUSH HL        |                                                      |
|        | LD HL, (POS)   |                                                      |
|        | LD (HL), A     | ; Записываем первый символ по адресу POS             |
|        | INC HL         | ; Следующий символ слога                             |
|        | EX DE, HL      |                                                      |
|        | POP HL         |                                                      |
|        | LD A, (HL)     | ; Если символ равен "?", то не записывать            |
|        | CP 63          | ; урезанный слог                                     |
|        | JP Z, SAVPS    |                                                      |
|        | LD (DE), A     | ; Записываем 2-й символ слога                        |
|        | INC DE         |                                                      |
| SAVPS  | LD (POS), DE   | ; Сохраняем позицию                                  |
| RND2   | LD HL, (D0)    | ; Сама процедура RND                                 |
|        | LD DE, (D2)    |                                                      |
|        | ADD HL, DE     |                                                      |
|        | EX DE, HL      |                                                      |
|        | LD HL, (D2)    |                                                      |
|        | LD (D0), HL    |                                                      |
|        | LD HL, (D4)    |                                                      |
|        | LD (D2), HL    |                                                      |
|        | ADD HL, DE     |                                                      |
|        | LD (D4), HL    | ; D0<---D2<---D4<---D0+D2+D4                         |
|        | RET            |                                                      |
| SAVENM | LD HL, NAME    | ; Записываем имя в массив                            |
|        | LD DE, (MASSV) |                                                      |
|        | LD BC, 8       |                                                      |
|        | LDIR           |                                                      |
|        | LD A, (D1)     | ; Считываем Y                                        |
|        | SRL A          | ; Преобразуем в нормальный вид                       |
|        | XOR 127        |                                                      |



```

LD      (DE), A      ; Записываем в массив
INC     DE
LD      A, (D3)      ; Считываем X и записываем в массив
LD      (DE), A
INC     DE
LD      (MASSV), DE   ; Запоминаем позицию в массиве
RET

RAND     LD      HL, RNDDAT ; Копируем данные о галактике в массив RND
LD      DE, D0
LD      BC, 6
LDIR     LD      HL, 40000
LD      (MASSV), HL
RET

D0        DEFW  00      ;
D1        DEFW  00      ;
D2        DEFW  00      ; Переменные
D3        DEFW  00      ;
D4        DEFW  00      ;
MASSV     DEFW  00      ;
NAME      DEFW  "      "
SLOGS     DEFM  "ALLEXEGEZA"
          DEFM  "CEBISOUSES"
          DEFM  "ARMAINDIRE"
          DEFM  "A?ERATENBE"
          DEFM  "RALAVETIED"
          DEFM  "ORQUANTEIS"
          DEFM  "RION"

RNDDAT    DEFB  74      ; 61...66 байты в отгружаемом блоке состояния ELITE
          DEFB  90
          DEFB  72
          DEFB  2
          DEFB  83
          DEFB  183

```

## Программа 2.1

### Печать имён планет всех галактик ELITE.

```

10  PAPER 0: INK 6: BORDER 0
20  CLS: PRINT "ELITE SPACE""1-STAR NAMES""2-PLOTS OF STARS"
30  LET k$=INKEY$: IF k$="1" THEN GO TO 60
40  IF k$="2" THEN GO TO 200
50  GO TO 30

```

### Подпрограмма вывода названий.

```

60  LET n=0: GO SUB 1070: GO SUB 1010: REM установка генератора случайных чисел в исходное
    положение и установка стринга n$, содержащего слоги всех названий планет.
70  CLS: PRINT INK 5; "NO NAME"
80  FOR f=1 TO 20
90  LET r=d0-INT (d0/256)*256
100 GO SUB 2000: REM В строках 100...120 и 150 содержатся четыре вызова генератора для одной
    звезды.
110 GO SUB 2000
120 GO SUB 2000
130 IF r>127 THEN LET r=r-128
140 IF r-64<0 THEN GO SUB 1030: REM Если 6-ой бит равен нулю, то производится холостой
    вызов RND и слово становится короче на один слог.
150 GO SUB 2000
170 LET n=n+1: IF n=256 THEN STOP: REM Если число звёзд равно 256, то конец работы.
180 NEXT f: BEEP .1, 30: PAUSE 0: GO TO 70: REM Формируется следующее имя

```

## Вывод позиций звёзд

```
200 CLS: PLOT 0,23: DRAW 255,0: DRAW 0,129
205 DRAW -255,0: DRAW 0,129
210 GO SUB 1010
220 FOR f=0 TO 256
230 LET y=INT (d0/256): LET y=INT (y/2)+24
240 LET x=INT (d2/256)
250 PLOT OVER 1;x,175-y: REM Вывод звезды
260 GO SUB 1030: GO SUB 1030
270 GO SUB 1030: GO SUB 1030
280 NEXT f: STOP

1000 REM *Подпрограмма инициализации статического генератора*
1010 RESTORE: READ d0, d2, d4: RETURN

1020 REM *Статический генератор случайных чисел*
1030 LET s=d0+d2: IF s>65535 THEN LET s=s-65536
1040 LET s=s+d4: IF s>65535 THEN LET s=s-65536: REM s=d0+d2+d4 (двухбайтное логическое
    сложение)
1050 LET d0=d2: LET d2=d4
1060 LET d4=s: RETURN

1065 REM *Генерация списка возможных слогов*
1070 LET n$="ALLEXEGEZACEBISOUSESARMAINDIREA?"
1080 LET n$=n$+"ERATENBERALAVETIEDORQUANTEISRION"
1090 RETURN

2000 REM *Выбор случайного слога*
2010 LET h=INT (d4/256): LET h=h-INT (h/32)*32
2020 IF h=0 THEN GO SUB 1030: RETURN
2030 LET h=h*2: PRINT n$(h+1)
2040 LET h=h+1
2050 IF n$(h+1)="?" THEN GO SUB 1030: RETURN
2060 PRINT n$(h+1): GO SUB 1030: RETURN

3000 DATA 74+90*256
3010 DATA 72+2*256
3020 DATA 83+183*256
3030 REM этот блок данных соответствует галактике с параметрами 74, 90, 72, 2, 83, 183
    (байты 61...66 в блоке состояния программы ELITE)
```

## ВОЗВРАЩАЯСЬ К НАПЕЧАТАННОМУ

В прошлом выпуске "Форума" наши читатели обращались за помощью по поводу программ "Swords & Sorcery" и "Dun Darach". Пока никто не откликнулся на эти письма, мы на скорую руку пробежались по своим архивам, выудили кое-что из импортных журналов и составили нечто вроде эссе из разных источников. Это, конечно, капля в море, на серьезный поиск пока не было времени, в дальнейшем может быть дадим больше информации.

Мы настолько рады, что начинает разгораться интерес к жанру ADVENTURE, что спешим раздуть первую искру, надеясь, что скоро вспыхнет и костер.

### Swords & Sorcery.



Вот несколько подсказок по прохождению первого квадранта (Quadrant 1).

Когда Вы проходите тренировку, то проведите один день с мастером Yama, шесть дней с мастером Bog и 7 дней с Мерлоком (Merlock). Это даст Вам 22 пункта магии.

Обязательно съешьте все пирожки (pies), сколько сможете себе позволить (до 5 штук) и во время своего путешествия прежде чем исследовать что-либо (Explore) включайте режим Attack на случай неожиданного нападения.

В первом квадранте самым первым делом пойдите в комнату 6 и возьмите там меч (sword). Комнаты мы нумеруем слева направо таким образом, что левая верхняя комната имеет номер 1. Меч позволит Вам наносить удары (режим ATTACK HACK). Все имущество, какое соберете, относите в эту комнату. Капкан (Room Trap) здесь лучше не разряжать (disarm), т.к. после этого на Вас нападают 5...6 монстров.

Мы приносим извинения за возможную неадекватность перевода, т.к. эту программу, мы сами не пускали, а в английском языке такие слова, как Room Trap могут иметь много разных значений и при переводе адекватное выбирается только по смыслу, другого пути нет.

После комнаты 6 идите в комнату 18 и там разрядите (disarm) капкан (trap). Тогда Вы сможете достать staff (волшебную палочку) из мешка (sack). Это даст Вам дополнительно 16 пунктов магии и, кроме того, два новых заклятья, но работают они только в том случае, если Ваши магические способности (magic skill) выше, чем боевые (fighting skill).

Владеть этой волшебной палочкой хорошо, но дорого, т.к. при этом быстро расходуется много магической энергии. Есть оригинальный выход - ее надо проглотить (eat). Тогда и энергия не расходуется, т.к. ее у Вас с собой больше нет, и магические способности сохраняются, ведь она всегда при Вас.

Соберите все мечи, которые Вам встретятся, и отнесите их в комнату 6. В комнате 15 есть Кольцо Прыгучести (Ring of Jumping), а ключ N1 (Key 1) Вы найдете в комнате 14. Вам следует также взять Sabaton в комнате 22 (что это такое мы не знаем, может быть какое-то оружие. В английском языке с давних пор принято давать имена собственные оружию. Вспомните меч Эскалибур короля Артура). Если у Вас к этому времени еще нет магического меча (Magic sword), убедитесь, что Вы выбрали ATTACK FOR A DEFENCE, иначе с монстром, который там обитает, Вам не справиться, его не берет заклятье FIREBOLT, не берет его и обычное оружие. Выход из этого квадранта находится в комнате 4.

Теперь "Dun Darach".

В этой игре полно потайных дверей. Их можно обнаружить тщательной проверкой стен. Обычно на наличие такой двери указывает факел на стене.

Первым делом Вам надо приобрести книгу (book) в магазине (shop) на улице King Street. Ее надо отдать леди Mgor (Ledy Mhor), а она Вам даст взамен другую. С ее помощью откроется потайная дверь на улице Myre Street. Внимательно поищите здесь. Если Вам удастся найти щит (shield), то его можно посвятить (offer) богине Дану (Goddess Danu).

На Древнем Холме (Old Hill) Вы найдете лиру (Lyre). С ней идите к Дейну (Dan), которой обитает за потайной дверью на улице Park Row. Он даст Вам взамен за эту лиру щит, посвященный богу Мидиру (Midir). Отнесите его богу и он даст Вам ключ.

В этой игре можно неплохо зарабатывать. Например так. Возьмите золото у Ассэйера (Assayer) на улице West Way и продайте его брокерам на улице Cross Street. Существует еще очень много разных способов, но этот наипростейший. Но опасайтесь воров, они часто похищают то, на чем находится Ваша "звезда". Поэтому никогда не помещайте ее на предмет, которым дорожите, например держите ее на деньгах (Iridi), которые сравнительно легко восстанавливаются.

В игре множество головоломок. В частности, многие встают в тупик перед вопросом, как взять жемчужину (Pearl) из комнаты Strong Room, что на северной стене и куда потратить свои деньги.

Обе проблемы можно решить одновременно. Набрав 10 000 Iridi, отнесите деньги в гильдию воров на улице Silver Street. Они продадут Вам лицензию (Licence), с помощью которой Вы и похитите эту жемчужину.

А вот пример еще более крутой головоломки. Чтобы войти в замок, Вам необходим свиток (Scroll of Scar) и специальный ключ (B-Key). В замок Вы войдете, но первая же дверь не откроется никакими стараниями. Вам надо иметь при себе еще какой-нибудь предмет (любой) и пройти с ними в комнату, где есть три вращающихся колеса - это система кодового замка. Ваша задача выложить предметы на стол (по одному) так, чтобы колеса остановились в определенной комбинации. Код этой комбинации - DPE, хотя чтобы его получить, надо бы решить головоломку, выцарапанную на столе. После того, как колеса остановятся, надо взять свиток со стола и идти в открывшуюся дверь.

И после этого Вас еще будет ожидать очень много разных загадок и непонятных подсказок. Так, например,  $2^{**}25$  - означает 2 в двадцать пятой степени. Если с помощью калькулятора Вы найдете, чему равно это число, то результат поможет выработать систему, с помощью которой можно будет пройти через длинные ряды дверей в замке.

В общем, друзья, это серьезная игра. Не удивляйтесь, если Вам придется поблуждать по улицам старинного города год - полтора, не меньше.

\* \* \*

И, в заключение, обращение ко всем! В той части, которая касается адвентюрных программ, у нас большие архивы и возможности. Если Вы где-то "застряли" и Вам нужна оперативная помощь, пишите, не стесняйтесь. Может быть мы сами дадим полезный совет, может быть кто-то из наших читателей пройдет Вашим путем и решит проблему, но ведь под лежащий камень вода не потечет, давайте вместе решать эти задачи.

И предложение к тем, кто досконально разобрался с какими-либо адвентюрами. Если Вы можете стать внештатными консультантами, сообщайте нам, мы дадим Ваш адрес и напишем, по каким программам к Вам можно обращаться за помощью. Если будет достаточное количество писем, этот раздел может стать и постоянным.

## HEAVY ON THE MAGIC



(с) Троеглазовы П. и Г. 1993  
Хабаровский край, с. Гайтер.

Здравствуйте!

В №6 за 1992 год Вы давали описание программы "HEAVY ON THE MAGIC". Каракашев А.Г. неплохо поработал с программой, но раскрывая листинг программы, он не догадался раскрыть и проанализировать отгружаемый блок, а именно здесь можно было найти путь к завершению программы или, по крайней мере, приблизиться вплотную к решению главной задачи - найти выход из замка.

Сначала о неточностях в описании. Дух в котле не AU, а AI. Магических званий - 9, но о них позже. В программе 188 активных предметов. Корреспондент пишет, что побывал во всех комнатах. Это не так.

Составив карту, мы заметили, что на первом этаже угловые комнаты справа не имеют входов. Ясно, что одна из них FURNACE. Мы быстро выяснили, что эта "пекарня" находится в верхнем правом углу, под лестницей. А вот в комнату рядом с Циклопом можно попасть, видимо, только при определенных условиях. Может, нужно разрушить стену или здесь поможет пресловутый ERLSTONE? Во всяком случае, нам удалось побывать в этой комнате не с игры, а искусственно задав соответствующие координаты Аксилу. В комнате - абсолютная темнота. Ее можно осветить на долю секунды, используя любое из заклятий и при вспышке можно увидеть 2 предмета - прах от убитого монстра и книгу Гримуар. Выход из комнаты один - "W" и он выводит Вас на 4-й этаж в район PIT (?). Комната полна странностей, например, все вызванные предметы исчезают из нее, как в любой другой комнате. Апекс, вызванный ранее, не уходит, сколько бы Вы ему не говорили "APEX, THANKS", он отвечает - "GLAD TO HELP" и остается в комнате. Мы можем поднять Гримуар, у нас будет два Гримуара, а на том же месте будет стоять еще один, поднимем его у нас три книги, а на полу еще одна, - и так без конца. Каждый из поднятых Гримуаров содержит столько же заклинаний, сколько у Вас было до появления в этой комнате. Лишние предметы выкладываются и стоят во всех углах комнаты, кроме ее центра, где лежит прах. Если выложить что-либо на прах, предмет исчезает. Заклятье BLAST в этой комнате не действует на предметы. Выйдя из этой комнаты, Аксил уже не может вернуться назад - позади стена.

Всего комнат 256. Счет ведется с первого этажа, слева направо - левая верхняя - N1, FURNACE - N8 и т.д. Счет комнат правого нижнего угла замка сдвинут на этаж вверх -

2 этаж - 64,  
3 этаж - 128,  
4 этаж - 192,  
1 этаж - 0.

Видимо, в нулевую комнату можно попытаться войти с помощью заклинаний или каких-либо других действий из комнаты N255 (или N63?).

Теперь об отгружаемом блоке. Рассказывать о каждом байте долго, а их 1342, но вот основные:

Первые 27 байтов - сведений об Аксиле, их можно отгружать и загружать отдельно, используя опции главного меню 4 и 5.

2-й байт - стойкость (STAMINA) 3-й байт - ила магии (MAGIC) 4-й байт - удача (LUCK) 5-й байт - звание (GRADE) В этом байт такое соответствие цифр:

- 1 - NEOPHYTE (новичок)
- 2 - ZELATOR (соискатель)
- 3 - FRACTICUS (практикант)
- 4 - PHILOSOPHUS (философ)
- 5 - ADEPTUS MINOR (младший эксперт)
- 6 - ADEPTUS MAJOR (старший эксперт)
- 7 - ADEPTUS EXEPT (высший эксперт)
- 8 - MAGISTR TEMPLI (магистр храма)
- 9 - MAGUS (маг, чародей)

6-й байт - набранные очки (points)

Далее идут пять блоков по 4 байта - предметы, которые находятся у Аксила. И 27-й байт - наличие заклинаний:

- 224 - INVOKE, FREEZE, BLAST
- 240 - INVOKE, FREEZE, BLAST, CALL
- 248 - INVOKE, FREEZE, BLAST, CALL, TRANSFUSION

94-й байт - комната, в которой находится Аксил. Например, 8 - FURNAGE, 43 - STINGS, 61 - CLAWS, 108 - MISERI (старт) и т.д.

Блок байтов со 188 по 217 - монстры, постоянно обитающие в 30-ти комнатах, в эти комнаты не может заходить никто, кроме Аксила. Пока не будет убит монстр, в эту комнату нельзя вызывать никого, даже Апекса, иначе Аксил погибнет. Два монстра в экране - мгновенная смерть. Убив монстра, можно спокойно, не боясь, что кто-нибудь затопчет Аксила у входа, задать дальнейший маршрут или обдумать дальнейшие действия. В комнату не заходит никто даже после смерти монстра, но теперь сюда можно вызвать любого из обитателей замка. Пока монстры живы, соответствующие байты равны нулю. Смерть монстра обозначается номером комнаты, в которой он обитает, например:

- 189-й байт - 150 (тролль)
- 193-й байт - 138 (призрак)
- 197-й байт - 18 (оборотень)
- 200-й байт - 49 (вампир)
- 205-й байт - 63 (циклоп)
- 210-й байт - 82 (ленивец)
- 217-й байт - 253 (медуза) и т.д.

С 541-го до 1292-го идут 188 четырехкилобайтных блока - это "активные" предметы. Активными мы называем предметы, непосредственно участвующие в игре, их можно брать, переносить, в крайнем случае - осматривать и изучать или уносить. Пассивные - фон. Все активные предметы, как правило, участвуют в отгрузках.

Из 4-х байтов, обозначающих какой-либо предмет, важны 2-й и 4-й: 2-й - комната, в которой находится предмет, 4-й - его координаты в комнате.

В игре 19 монстров. Каракашев, видимо, забыл упомянуть об одном из них - это голова дракона "RABAK". Она не пропустит Вас, пока Вы не скажете - "RABAK, WATER". На все вопросы у "дракоши" один ответ, но вот на фразу - "RABAK, DRAGON" он ответил нам "Мне горячо?". "Ах, тебе горячо!!!" - подумали мы и предложили ему водички и RABAK открыл нам дорогу к последнему "танцевальному залу" в "LIGHTGATE".

Апекс не то, чтобы "ходячая энциклопедия", но все-таки знает немало. У него в запасе около 40 вариантов ответов и, заметьте, в разных местах замка он может ответить по-разному на один и тот же вопрос. В некоторых случаях он прямо подсказывает, как вызвать птицу Феникс, как пройти воду или как пройти через пропасть. Подсказывает, как подобрать ключи к дверям, ведь над каждым ключом нарисован знак и на вопрос "APEX, SIGN", он

отвечает - "Это звездный знак", ключей 12, знаков Зодиака тоже 12. Остается сравнить знаки, нарисованные в звездах над ключами с названиями комнат и станет ясно, каким ключом открывать какую дверь (смотри рис. 1).



Рис.1

Про знак с двумя кольцами Апекс говорит - "Это знак пошлости". К этим дверям не нужны ключи, просто нужно поставить на стол мешочек с золотом "BAG". Про знак, у дверей с односторонним проходом, он говорит - "Знак - хода нет".

Остальных обитателей замка он явно недолюбливает:

WYVERN и TROLL - "Лучше их убить!"

CYCLOPS - "Один глаз и полный идиот!" и т.п.

Апекс, даже ничего не говоря, подсказывает слово SORONOROS. Обратите внимание на его живот, видите знак? Теперь прикиньте его к надписи на стене...



Рис.2

А самое главное - он подсказывает, где искать выход - "APEX, EXIT", ответ - "PILE TOMBS AND PARADISE". Такие комнаты есть на 2-ом и 4-ом этажах - N65 и N242.

Но нам все-таки кажется, что та самая странная "темница" на первом этаже должна дать ключ к выходу на свободу, ведь попасть в нее еще пока не удавалось никому (естественным путем).

У нас еще осталось 10 неиспользованных предметов. Каракашев пишет о 4-х, но кроме 3 костей (BONE) и одной "RIB" куда-то нужно пристроить "FOOT", "GRIMOIRE" (в нулевой комнате), 2 оставшиеся сумочки с золотом (BAG) и 2 золотых камня, похожих на SNAKE - "ROCK". Камни тяжелые, когда Аксил их поднимает, он теряет 10 единиц энергии, но не вздыхает "О-Н-Н-Н...", как если бы он брал отравленный предмет.

Волчья лапа "FOOT" поднимает на несколько единиц цифру в графе "Удача". (Это же происходит, когда мы берем FLASK).

Может быть, какие-то из этих вещей нужно принести в комнату PILE TOMBS и тогда откроется вход в эту странную комнату с прахом в центре и бесконечным Гримуаром рядом с прахом, а уже потом идти в PARADISE. Но для того, чтобы проверить эти догадки нужно еще найти или хотя бы узнать, что такое EARLSTONE, ведь он должен подсказать, как

открыть дверь в LIGHTGATE. Тот путь, что подсказывает Каракашев А.Г., видимо неправильный, т.к. попав в комнату с рубином с помощью Астарота, разрушив дверь с помощью Асмодея, мы входим в дверь. Аксил исполняет танцевальный номер, но очки... убывают. Значит, недостаточно этого, чтобы найти вход - необходимо открыть эту последнюю дверь.

У "темницы" есть еще одна особенность - принесенные с собой предметы можно множить до бесконечности (эффект Гримуара), нужно только поставить на пол необходимый предмет, затем поднять его и т.д. Мы только не проверили, что будет, если набрать этих "фантомов", войти в обычную комнату, выставить их, затем выйти из комнаты и снова войти - скорее всего они исчезнут - проверьте.

Недостаточно ясна и роль FURNAGE. Обычно туда забрасывают Аксила демоны, если он им надоест, отбирая у него все предметы и магию. Но... Возьмите с собой CLASP, другие необходимые предметы и попросите Астарота отправить Вас туда. CLASP предохранит Аксила от огня, он может ходить по комнате, исследовать ее, вызывать предметы и демонов и, если они будут недовольны, то дальше этой комнаты не бросят Аксила. Вот только загвоздка - как оттуда выбраться, может вызвать "фантом" SWORD и снова использовать Астарота?...

Да, кстати, нулевая комната приготовила для нас еще один сюрприз - если мы будем брать лежащий там Гримуар и снова ставить его, у нас будет накапливаться сила магии. Таким образом, ее можно довести до 99 единиц.

В предыдущем описании был неправильно назван один из трех предметов, необходимых для вызова духа - AI. Это - SKULL, а не HEAD. Оба черепа HEAD непригодны к использованию. Только поняв, что на предметах, имеющих важное значение, как правило, есть какие-то знаки или цифры, нам удалось найти третий необходимый предмет для вызова AI.

Каракашев пишет, что он набрал 99 единиц в графе "очки" и, мол, странно, что Аксил не получил свободы - игра то сделана! Дело в том, что очки начисляются не только за прохождение маршрута и выполнение необходимых условий. За каждого убитого монстра начисляются очки, например, за Тролля - 1, за Змея - 2, за Медузу - 8 и т.д. Чем сильнее противник, тем больше очков Вы набираете. Можно набрать 99 очков не выходя из какой-либо комнаты, где часто появляются монстры. Количество очков важно для применения TRANSFUSION. Если у Аксила кончается энергия, он может воспользоваться этим заклинанием и десять единиц перекочают из графы "очки" в графу "стойкость". Вы можете повысить стойкость Аксила до 99-ти, но в этом случае у Вас может оказаться 0 очков. Поэтому пользуйтесь "TRANSFUSION" экономно, ведь для выхода на свободу Аксилу несомненно нужен полный комплект и выполненных действий и очков.

В начале игры Вы можете изменить (перетасовать) цифры в графах "стойкость", "магия", "удача". Лучший вариант, когда меньшая цифра в графе "стойкость", большая в "магии" и средняя в "удаче". Просто после старта нужно быстренько пробежать через 3 комнаты и подкрепиться хлебом (10 единиц), а затем возвратиться к старту и взять Гримуар. После того, как Вы подберете недостающие заклинания CALL и TRANSFUSION, Ваша сила магии повышается до 50 единиц и Вы будете уничтожать слабых и средних противников с первого удара. В этом случае Ваше путешествие значительно облегчается. Кроме этого, можно повысить силу магии, переноса с собой магические предметы - MANTIS, SWORD, SNAKE и т.п.

Мы подозреваем, что повысить звание выше PHILOSOPHUS в этой программе невозможно, хотя искусственным путем (изменив 5-й байт в отгружаемом блоке) мы смогли их получить. Но, возможно, каким-то путем их можно получить в комнатах PILE COLLODON, PILE TOMBS, PARADISE и в той же самой темнице на первом этаже, а уж звание Мага - только на выходе на свободу. Возможно...

К тому же, в программе есть еще масса не используемых слов - зачем они?

BRONZE, BIRD, CRYPT, CHROMA, COPPER, COBALT, FOUNTAIN, IPSISSIMUS, MAGNAM, MONSTER, NICKEL, NEARBY, OGRE, PASSWORD и т.д.



# ЧИТАТЕЛЬ ЧИТАТЕЛЮ

Своими успехами в области программирования на Бейсике делится с нами т. Дронов из Читы. Его опыт пригодится начинающим программистам.

"Высылаю Вам на суд свою первую программу и хочу сказать Вам большое спасибо за "ZX-РЕВЮ". Благодаря Вам я, изучив за месяц два годовых комплекта "ZX-РЕВЮ", узнал очень много хитростей и секретов Бейсика. Подтверждением этого является эта программа, написанная за один вечер. Я знаю, Бейсик - легкий язык, но без хорошей книги его не ПРОСТО освоить. А Ваша литература лучшая из всех, которую я видел. Информация излагается очень понятно и доступно. "Сейчас я, с помощью Вашей книги, приступаю к изучению Ассемблера. Надеюсь овладеть и этим языком, хотя он намного сложнее.

Теперь несколько слов о программе. Это аналог широко известной настольной игры "Морской бой", в которую мы частенько играли в детстве. Маленькое отличие заключается в том, что в моей игре корабли можно располагать как угодно и где угодно в пределах своего квадрата. Так же делает и компьютер."

ИФК: Спасибо за теплые слова. От себя хотим добавить несколько слов. Корабли в этой игре - только одноклеточные, их можно ставить и поодиночке, и вплотную друг к другу как угодно. После попадания корабль сразу же считается потопленным. Всего их 20 у Вас и столько же у компьютера. Посмотрите на листинг программы: действительно, это занятие на один вечер. Если организовать в программе ограничения на расположение кораблей, на их конфигурацию (одно- или многоклеточные и т.д.), программа получилась бы как минимум, раз в пять длиннее. Но можем Вас уверить, что набрав самостоятельно эту программу, Вы с не меньшим азартом будете вести сражения с компьютером. А может быть, вскоре у Вас появится желание внести в программу ограничения на конфигурацию и расположение кораблей - что же, пробуйте смело. Лучшее обучение программированию - это обучение "в бою".

Автор предлагает русифицировать программу при помощи символов UDG-графики. Однако, поскольку мы уже привыкли к постоянно используемому нами методу русификации, мы приводим, как обычно, вариант для загружаемого русско-латинского символьного набора. Думаем, что автор простит нам такое вторжение в его программу и еще несколько непринципиальных штрихов. Итак, предлагаем программу всем читателям.



Рис. 1

```
1 GO TO 10
2 BORDER 7: PAPER 7: INK 0: CLEAR : LOAD "chr" CODE 64600
4 GO TO 0
5 SAVE "SEA" LINE 2: SAVE "chr"CODE 64600,768: STOP
8 POKE 23606,88: POKE 23607,251: RETURN
9 POKE 23606,0: POKE 23607,60: RETURN
10 BORDER 5: PAPER 7: INK 0: CLS : POKE 23658,8: GO SUB 8
15 REM РИСУЕМ ИГРОВОЕ ПОЛЕ ===
20 FOR N=0 TO 9
30 LET Z$=CHR$ (N+65)
```

```

40 PRINT AT 7,N+4;N;AT 7,N+19;N;AT N+8,3;Z$;TAB 18; Z$
50 NEXT N
60 PRINT AT 0,1; PAPER 4;" DRONOV * TCHITA * 1993 "
70 PRINT AT 3,10; PAPER 7; INK 1; " МОРСКОЙ БОЙ "
80 DRAW 255,0: DRAW 0,175: DRAW -255,0: DRAW 0,-175
90 PLOT 112,112: GO SUB 630: PLOT 232,112: GO SUB 630
100 REM ФОРМИРУЕМ КООРДИНАТЫ КОРАБЛЕЙ КОМПЬЮТЕРА =====
110 PRINT #0;"ПОДОЖДИТЕ МИНУТОЧКУ, ПОЖАЛУЙСТА."
120 DIM A$(20,4): DIM B$(20,4)
130 FOR N=1 TO 10
140 LET Y=8+INT (RND*10): LET X=4+INT (RND*10)
150 LET A$(N)=CHR$ 22+CHR$ Y+CHR$ X
160 IF N=1 THEN GO TO 200
170 FOR M=1 TO N-1
180 IF A$(N)=A$(M) THEN GO TO 140
190 NEXT M
200 NEXT N
210 INPUT ;: PRINT #0;"Я ГОТОВ! РАСПОЛОЖИМ ВАШИ КОРАБЛИ": BEEP .5,40: PAUSE 50
220 REM РИСУЕМ КОРАБЛИ =====
230 FOR N=1 TO 20
240 GO SUB 690: LET Y=Y-57: LET X=X-29
250 LET G$=CHR$ 22+CHR$ Y+CHR$ X: LET B$(N)=G$
260 IF N=1 THEN GO TO 300
270 FOR M=1 TO N-1
280 IF B$(N)=B$(M) THEN GO SUB 740: GO TO 240
290 NEXT M
300 PRINT INK 1; G$+CHR$ 143
310 NEXT N
320 REM НАЧАЛО ИГРЫ =====
330 LET W=0: LET Q=0: BEEP .5,20
340 PRINT AT 20,4;20;TAB 19;20
350 LET N=INT (RND*2): REM РОЗЫГРЫШ ПЕРВОГО ХОДА =====
360 IF N THEN GO TO 520
370 PRINT #0;TAB 10;"ВАШ ВЫСТРЕЛ"
380 PAUSE 40
390 IF Q=20 THEN INPUT ;: PRINT #0;TAB 4;"ПОЗДРАВЛЯЮ! ВЫ ПОБЕДИЛИ": FOR K=10 TO 15: BEEP
    .1,K: NEXT K: GO TO 790
400 GO SUB 690: LET Y=Y-57: LET X=X-44
410 IF SCREEN$ (Y,X)="+" THEN GO TO 620
420 IF SCREEN$ (Y,X)="X" THEN GO TO 620
430 GO SUB 680
440 FOR N=1 TO 20
450 IF F$=A$(N) THEN PRINT #0; TAB 4;"БРАВО! ПРЯМОЕ ПОПАДАНИЕ": LET A$(N)=C$+"X": PRINT
    PAPER 2;A$(N): LET Q=Q+1: PRINT AT 20,4;20-Q;" ": BEEP .5,1: PAUSE 20:GO TO 390
460 NEXT N
470 PRINT C$+"+": BEEP .1,40
480 PRINT #0;TAB 14;"МИМО!"
490 REM ПЕРЕХОД ХОДА =====
500 PAUSE 40: INPUT ;
510 IF W=20 THEN PRINT #0;TAB 6; "ВАША ФЛОТИЛИЯ РАЗБИТА": BEEP .8,2: BEEP .6,2: BEEP .2,2:
    BEEP .8,2: GO TO 760
520 PRINT #0;TAB 10; "МОЙ ВЫСТРЕЛ": PAUSE 20
530 LET Y=8+INT (RND*10): LET X=19+INT (RND*10)
540 IF SCREEN$ (Y,X)="+" THEN GO TO 530
550 IF SCREEN$ (Y,X)="X" THEN GO TO 530
560 GO SUB 680
570 FOR N=1 TO 20
580 IF F$=B$(N) THEN LET B$(N)=C$+"X": INPUT ;: PRINT #0; TAB 6;"ВАШ КОРАБЛЬ ПОТОПЛЕН":
    PRINT INK 1;B$(N): LET W=W+1: PRINT AT 20,19;20-W;" ": BEEP .5,1: GO TO 500
590 NEXT N
600 INPUT ;: PRINT #0;TAB 5;"ВАМ ПОВЕЗЛО, Я НЕ ПОПАЛ": PRINT C$+"+"
610 BEEP .1,40: GO TO 380
620 PRINT #0; "ВЫ СЮДА УЖЕ СТРЕЛЯЛИ": GO TO 380
630 RESTORE
640 FOR N=0 TO 8

```

```
650 READ Y,X: DRAW Y,X
660 NEXT N
670 RETURN
680 LET C$=CHR$ 22+CHR$ Y+CHR$ X: LET F$=C$+" ": RETURN
690 INPUT "ВВЕДИТЕ КООРДИНАТЫ " ;D$
700 IF LEN D$<2 THEN GO SUB 740: GO TO 690
710 LET Y=CODE D$(1): LET X=CODE D$(2)
720 IF Y<65 OR Y>74 OR X<48 OR X>57 THEN GO SUB 740: GO TO 690
730 RETURN
740 PRINT #0;TAB 12; "ОШИБКА! "
750 BEEP .5,20: PAUSE 40: RETURN
760 FOR N=1 TO 20
770 IF A$(N,4)=" " THEN PRINT PAPER 2;A$(N)
780 NEXT N
790 PAUSE 70: INPUT ;: PRINT #0;TAB 4;"ХОТИТЕ ЕЩЕ РАЗОК? (Y/N)": PAUSE 0
800 IF INKEY$="N" THEN GO SUB 9: STOP
810 RUN
820 DATA 0,-81,-81,0,0,81,81,0,0,-2,2,0,0,-81,-81,0,0,2
```

# СОВЕТЫ ЭКСПЕРТОВ

Fighter Bomber

\* VECTOR GRAPHICS \*

\* ACTIVISION \*

Эксперт: Сороченко С. В.

г. Норильск



## Загрузка игры

После загрузки первого блока Вам предложат ввести имя пилота и выбрать один из четырех самолетов:

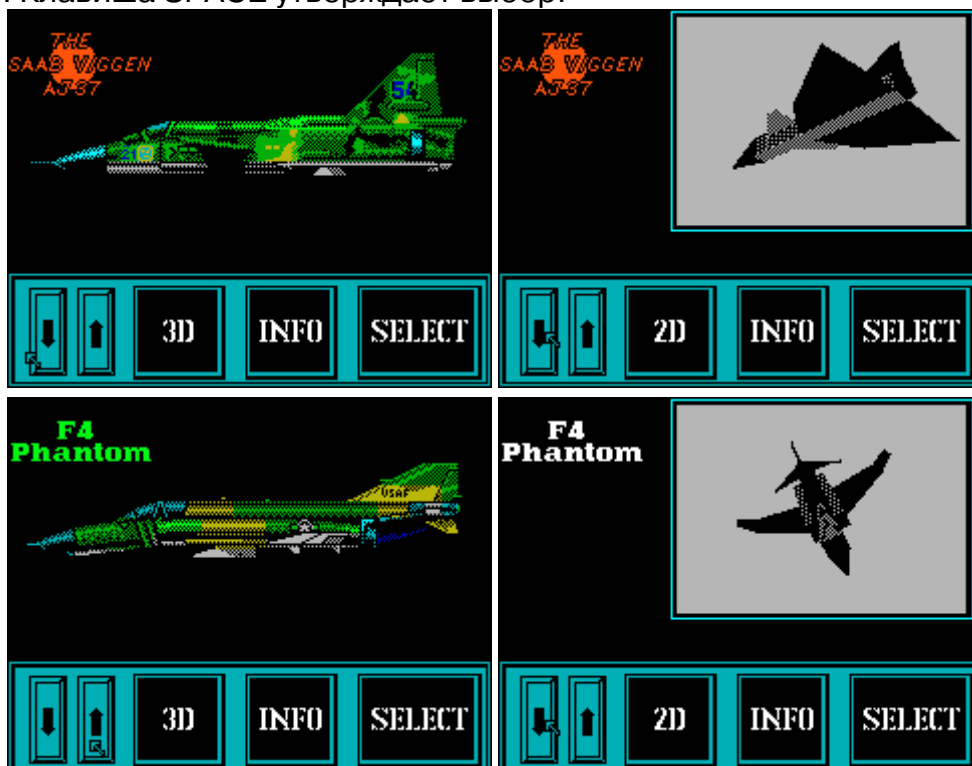
SAAB VIGGEN AJ 37 - самолет совместного производства шведских и британских авиастроителей.

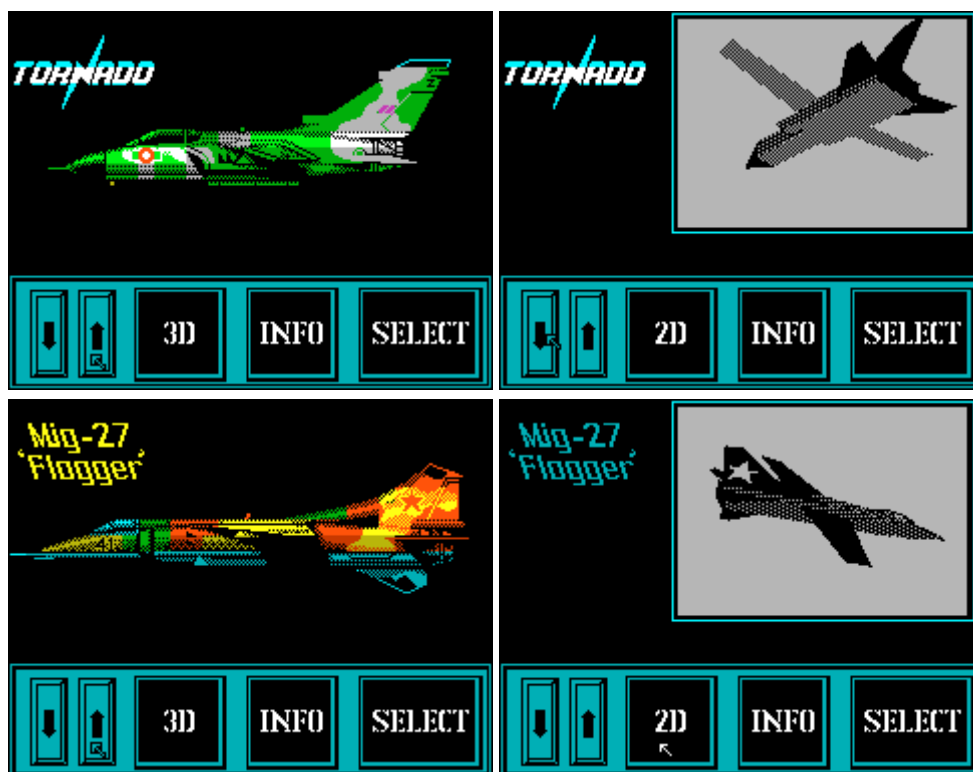
F-4 PHANTOM - ВВС США

TORNADO - Британские ВВС

MIG-27 ("FLOGGER") - советские ВВС. (Кстати, на рисунке изображен вовсе не MIG-27 FLOGGER, а MIG-23 FLOGGER - это я проверил в справочнике. Информация в программе также дана на MIG-23, а не на MIG-27).

Итак, на экране показан SAAB. Клавишами A,Q,O,P можно перемещать курсор по пиктограммам. Клавиша SPACE утверждает выбор.





Пиктограммы "ВВЕРХ" и "ВНИЗ" - это просмотр самолетов.

3D - это самолет в трехмерном виде.

INFO - информация об этом самолете.

SELECT - выбор штурмовика. После нажатия SELECT загружаются миссии, которые Вы можете выбрать. На экране появятся надписи:

Free Flight - свободный полет

Covert - военная миссия.

Ниже размытым шрифтом написаны остальные военные миссии, которые Вы сможете выбрать только после выполнения предыдущей миссии. В каждой миссии по два задания.

Сначала для ознакомления с аспектами полета следует выбрать Free Flight - появляется новое меню:

On Runway - на взлетной полосе.

At 30000 ft - на высоте 30000 футов.

Above Bridge - над мостом.

Lined Up - приземление.

ENTER выход в предыдущее меню.

Выберем On Runway и загружаем собственно саму игру. Внимание! В зависимости от типа выбранного самолета, после загрузки этого блока нужно загрузить еще два маленьких блока. Например:

"TORNADO" - игра запускается сразу без подзагрузки маленьких блоков.

"SAAB" - первые два блока.

"MIG" - вторые два блока.

"FANTOM" - третья пара блоков.

После этого на экране появится вид из кабины самолета.

### Управление самолетом

Клавиши:

Q,A - тангаж.

O,P - крен.

W - торможение с помощью шасси.

B - воздушные тормоза.

CAPS SHIFT + 1...0 - изменение тяги двигателя (1 - наименьшая, 0 - наибольшая), при двукратном нажатии CAPS + 0 двигатели переходят в форсированный режим.

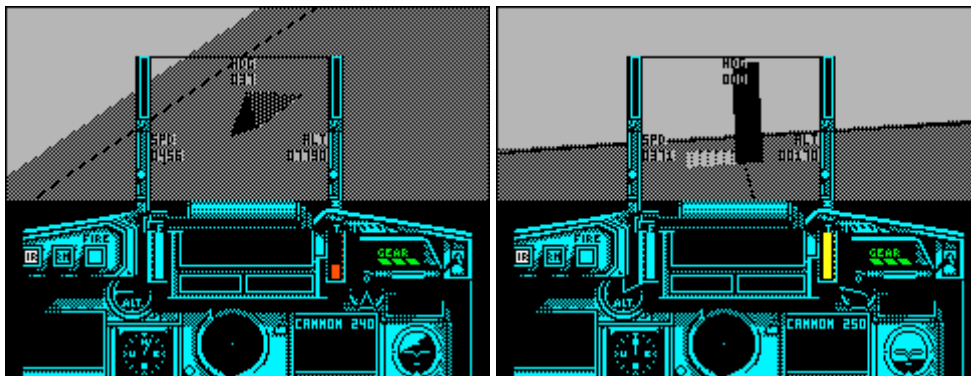
ENTER - производит переключение на один из трех видов вооружения.

CANNON - пушка;

MAVERICK - ракеты "воздух-земля" с наведением по лучу радара.

SEWINDER - ракеты "воздух-воздух" с инфракрасным наведением.

Клавиша G убирает и выпускает шасси (только в воздухе).



### Приборная доска

В центре - радар с цифрой, указывающей дальность его действия в милях. Эта цифра изменяется нажатием клавиши R.

Так как в разных самолетах расположение остальных приборов разное, то описывать их расположение не буду, а назову только функции.

Квадратик с надписью IR становится красным, если по Вам выпустили ракету (только в боевых миссиях).

Прибор с буквой N в верхней части - компас.

Прибор, показывающий тангаж и крен выглядит характерно для всех имитаторов.

Столбик с буквой T рядом - указатель тяги.

Столбик с буквой F - горючее.

Также имеются окно вооружения, которое показывает, сколько и какого у Вас боезапаса осталось, и окно бортового компьютера, предупреждающего об опасности и других важных вещах.

Прибор с надписью ALT - альтиметр (Ваша примерная высота).

Прибор с цифрами 0, 1, 2, 3 - Ваша скорость в MAX-ах.

Ваши высота, скорость, а также направление показаны в центре основного экрана в точных цифрах.

Основная прелесть этой игры, на мой взгляд - это возможность обозревать свой самолет во время игры с самых различных позиций.

Итак,

SYMBOL SHIFT+1 - обзор из кабины самолета.

SS+2 и SS+3 - неизвестно зачем.

SS+4 - посмотреть на самолет с вышки, находящейся на ВПП. Скорее всего, Вы увидите просто точку в небе, но клавишами K и J можно увеличить или уменьшить изображение.

SS+5 - посмотреть назад.

SS+6 - посмотреть влево.

SS+7 - посмотреть вправо.

SS+8 - посмотреть сверху, работают клавиши K и J

SS+9 - посмотреть сзади, клавиши K и J функционируют.

SS+0 - посмотреть на самолет с южной стороны, то есть с той стороны, в которую он направлен в начале игры. Это полезно, чтобы оценить Ваше положение относительно ВПП. Этот режим позволяет кроме увеличения и уменьшения изображения, вращать изображение по горизонтали клавишами I и U, а также по вертикали, - D и E.

### Полет

Полет в принципе ничем не отличается от других имитаторов, но маневрирование, при взгляде со стороны, производит большое впечатление.

Однако, наряду с достоинствами, есть несколько досадных недоработок, которые

снижают впечатление, как об имитаторе.

Так, например, даже на самой большой тяге можно лететь с выпущенными шасси и ничего плохого, кроме снижения скорости не случится.

Самолет почему-то может садиться в любом месте, а не только на ВПП, поэтому посадка не представляет никакой сложности. Можно сесть где-то в районе ВПП, а потом подрулить к ней.

Но самый, на мой взгляд, грубый просчет состоит в том, что можно рулить хвостовым оперением (клавиши Z и X) не только на маленькой скорости на ВПП, но и при скорости 1200 миль/час в воздухе. Где это видано, чтобы самолет разворачивался в воздухе без крена?! Такого не выдержит никакое хвостовое оперение.

### **Военные миссии**

Я не стану описывать все миссии - это займет слишком много места, а описание получилось и так слишком длинным.

Первая миссия, которую Вы можете выбрать - это COVERT. После этого на экране появятся две подмиссии. Вы можете выбрать только первую, так как вторая для Вас пока закрыта.

Первая подмиссия - SLEEPER.

Вам покажут участок карты США. Обязательно надо ознакомиться с текстом миссии, иначе не будете знать полетного задания.

Пиктограмма INFO предложит Вам выбрать один из пунктов на карте. По нажатии "FIRE", в окне появится надпись, что это за объект.

Пиктограмма "OK" означает, что Вы ознакомились и решили начать действия.

В первой миссии SLEEPER Вам нужно уничтожить лагерь террористов, а затем вернуться на базу.

Это самая простая миссия. В ней нет самолетов противника, но если Вы пролетите слишком близко к цели, по Вам могут выпустить ракеты.

Целей в этой миссии две, хотя уничтожить нужно только одну. Вторая цель во внимание не принимается.

Итак, Вы стоите на аэродроме.

В "окне" показаны координаты цели. Клавишей N можно переключать на координаты Вашей базы.

HDG - направление,

RGE - расстояние в милях.

Переключите радар на 25 миль.

Справа появится точка - это цель. Взлетайте и ложитесь на курс. Когда до цели останется меньше 20 миль, переключайтесь на VP "Мейверик". На экране появится крест. Нажмите S и головка наведения ракеты захватит цель. При этом на экране бортового компьютера появится надпись LOCKED. Не рекомендую захватывать цель на расстоянии больше 20 миль - велика вероятность промаха. После захвата цели нажмите "пробел" - ракета попадет в цель. Следить за ней можно по радару. Если ракета попала, компьютер выдаст TARGET DESTROYED - цель уничтожена. Если компьютер выдал MISS TARGET DESTROYED - значит Вы уничтожили главную или одну из главных целей. После того, как Вы уничтожите все главные цели, можете вернуться на базу, приземлиться на взлетную полосу и выключить двигатели.

После этого Вас спросят, хотите ли Вы повторить миссию. Если нет, то нужно загрузить блок, после загрузки которого Вам снова покажут карту и курс, которым Вы летали, а также скажут, сколько целей Вы подбили и засчитывается Вам эта миссия или нет. После этого запросят, желаете ли Вы принять следующую миссию или начать игру с самого начала. Если Вы хотите сыграть в следующую миссию, снова загружайте блок миссий и выбирайте COVERT. Вы увидите, что открылась вторая подмиссия - BRIDGE END. Это уже намного более сложное задание, так как появляются самолеты противника. Самолеты можно сбить пушкой, что очень сложно. Но можно и с помощью ракет "Сайдуиндер". Если на радаре появился самолет - переключайтесь на "Сайдуиндер". При этом в экране

вооружения появятся расстояние до самолета, его высота, направление и скорость. Теперь нужно поймать его в центральный экран. После появления сообщения LOCKED - пускайте ракету и вражескому самолету не миновать уничтожения.

Ваш самолет в отличие от вражеских имеет возможность выпускать радиолокационные и тепловые ловушки против вражеских ракет. Радиолокационная - клавиша C, тепловая - клавиша F.

Если то, что Вы прочитали, Вам интересно, смело стартуйте в небо и пусть не отвернется от Вас Удача!

## Mercenary "NOVAGEN" 1987

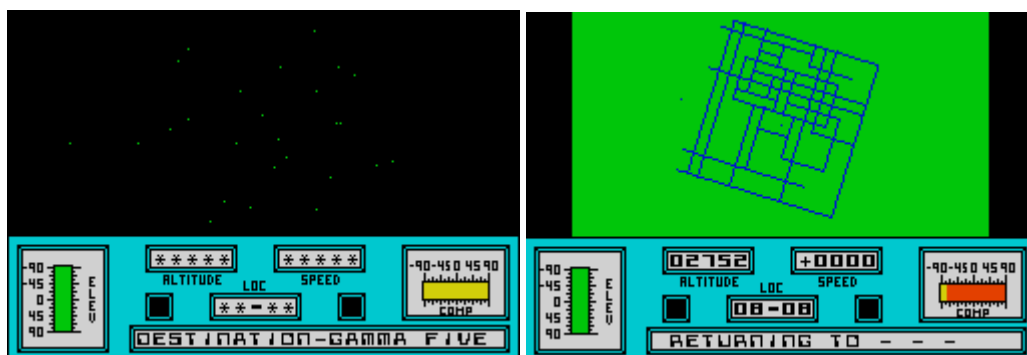


Эксперт: Кустов Г.В.  
Г. Львов

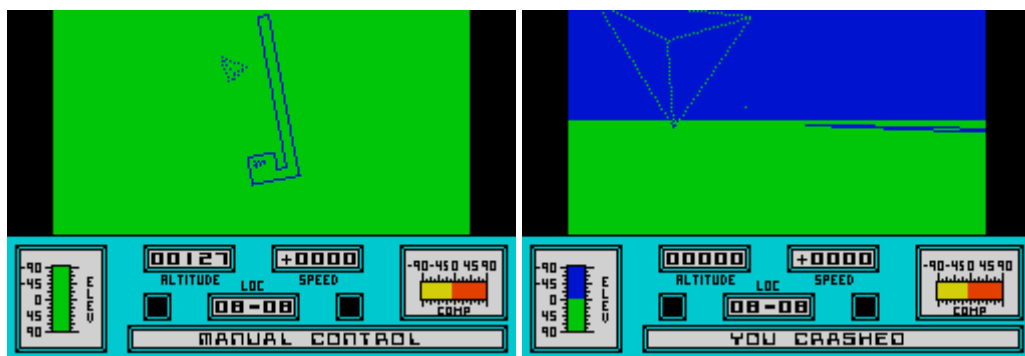
Я хотел бы выступить в качестве неофициального эксперта и рассказать про очень интересную, на мой взгляд, программу, которую можно было бы поставить рядом с "ELITE". Называется она "MERCENARY". Про эту программу я нигде ничего не слышал и в Вашем журнале она тоже не затрагивалась. Думаю, мои исследования пригодятся любителям этой программы.

Итак, "MERCENARY ESCAPE FROM TARG" можно отнести к жанру ARCADE/ADVENTURE с элементами PUZZLE, так как главное - это умение ориентироваться в подземных лабиринтах и даже в надземных. В этой программе использована очень необычная объемная графика и для простоты видимые объекты нарисованы только из ребер, т.е. есть уникальная возможность проходить сквозь объекты и рассматривать их с любой из шести сторон света. Этот эффект настолько сильно захватывает, что забывается отсутствие геометрических плоскостей у нарисованных объектов. В программе возможно пользование несколькими видами транспортных средств. Управление транспортом имеет разные особенности для каждого вида транспорта.

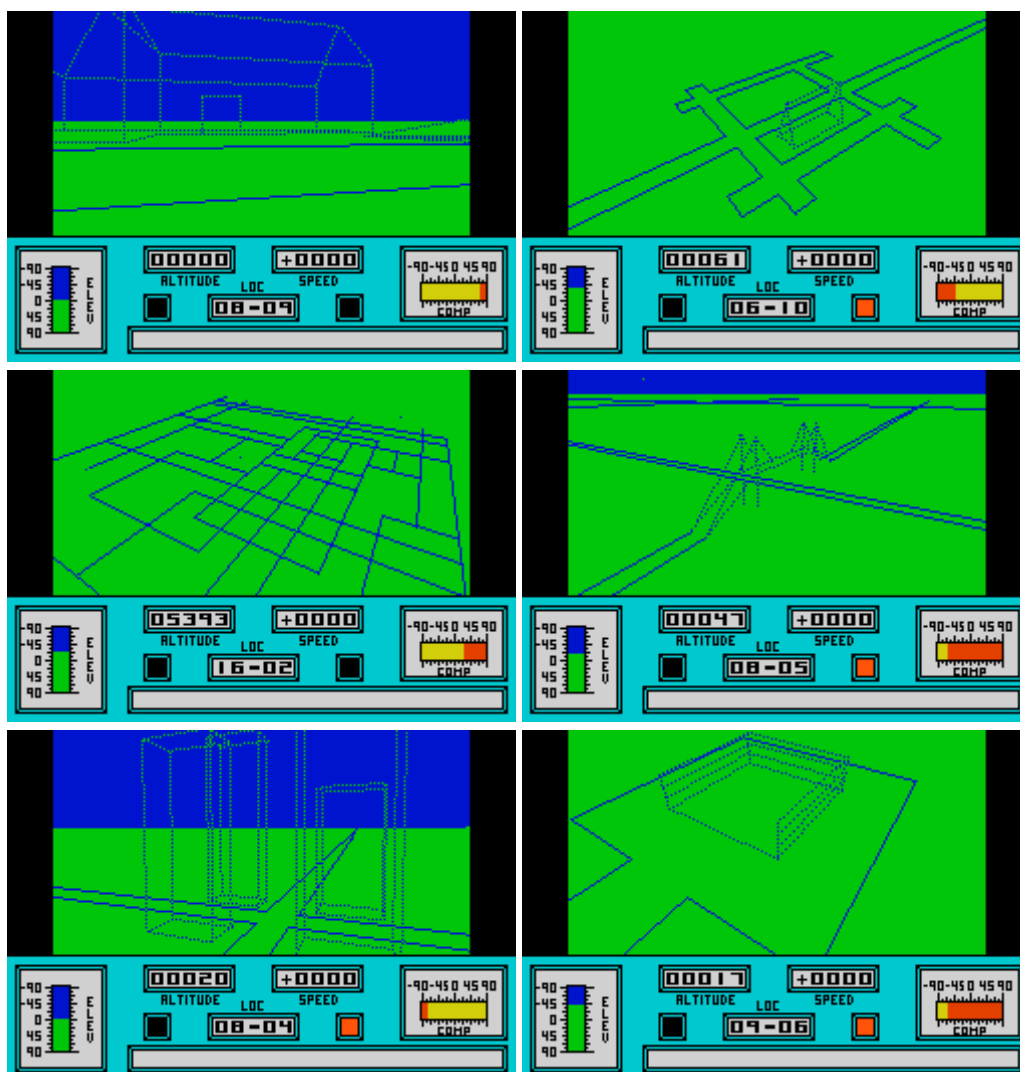
Действие начинается с того, что Ваш звездный корабль стартует и берет курс на систему "GAMMA FIVE". После этого бортовой компьютер сообщает о повреждении системы управления и корабль сбивается с курса, который невозможно откорректировать. Вы летите прямо на какую-то планету, катастрофа неминуема и... Вы разбиваетесь.







Оказавшись на поверхности, Вы получите сообщение - о ходе войны между PLYAR и MECHANOID (PLYAR - это люди, MECHANOID - роботы). Вы находитесь возле аэробазы, где стоит самолет. У Вас с собой 9000 кредитов, а самолет стоит 5000. Все же, его желательно купить, иначе вряд ли Вы далеко уйдете пешком.



Итак, Ваша миссия: как наемник, работая на людей (хотя можно работать и на роботов) Вы должны найти главного робота и продать его людям для предотвращения войны, а также найти и доставить различные предметы. В виде премии за них Вы должны получить более 1000000 кредитов, на которые можно купить новый корабль и покинуть эту негостеприимную планету, завершив тем самым свою миссию.

### Управление игрой.

Выбор управления возможен один раз, в начале игры - Вы выбираете один из трех возможных типов джойстика. Кроме выбранного джойстика есть управляющие клавиши на клавиатуре:

L - выход из транспорта,

V - вход в транспорт,

T - взять предмет,

D - положить предмет,

E - подъем/опускание лифта,

1,2,3,4 - переключение скоростей соответственно 1,2,3 и 4 скорости,

BREAK - сброс скорости до 0,

K - плавное увеличение скорости,

J - плавное уменьшение скорости,

CS+S (CAPS SHIFT + S) - вывод на ленту игровой ситуации,

CS+L - ввод с ленты отгруженной игры,

CS+Q - экстренный вызов самолета.

В случае управления героем, джойстик действует так:

"ВВЕРХ" - вперед,

"ВНИЗ" - назад,

"ВПРАВО" - вправо,

"ВЛЕВО" - влево,

"ОГОНЬ" - не действует. Если Вы управляете автомобилем или TURBO-автомобилем, джойстик действует так:

"ВВЕРХ" - не действует,

"ВНИЗ" - не действует,

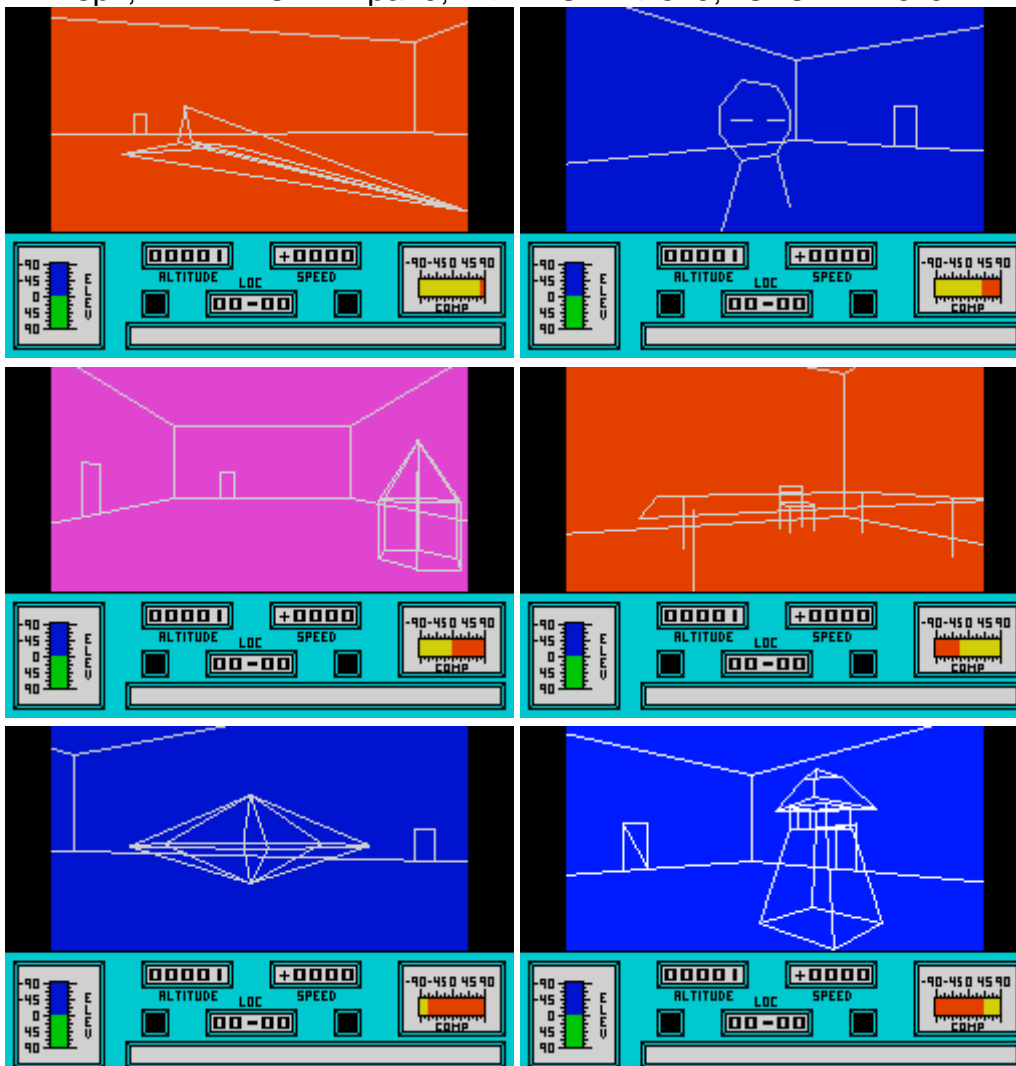
"ВПРАВО" - вправо,

"ВЛЕВО" - влево,

"ОГОНЬ" - огонь.

В самолетах и в летающей тарелке - следующее соответствие: "ВВЕРХ" - вниз,

"ВНИЗ" - вверх, "ВПРАВО" - вправо, "ВЛЕВО" - влево, "ОГОНЬ" - огонь.



### Работа с лентой.

Выгрузку игровой ситуации осуществляют нажатием CS+S. После этого в строке для сообщений выдается запрос, какой блок выгружать 1...4. Это позволяет вывести на ленту четыре разных игровых ситуации. У каждого блока свой номер, который проверяется в момент загрузки. Пригодиться выгрузка под несколькими номерами может, когда Вы идете, к примеру, на рискованную операцию. Тогда Вы выгружаете игровую ситуацию до операции под одним номером, а после операции - под другим. Загрузка игровой ситуации происходит так же, как и выгрузка, только в начале надо нажать CS+L вместо CS+S.

### Экран программы

После запуска программы в нижней части экрана появится таблица, отражающая текущую информацию.

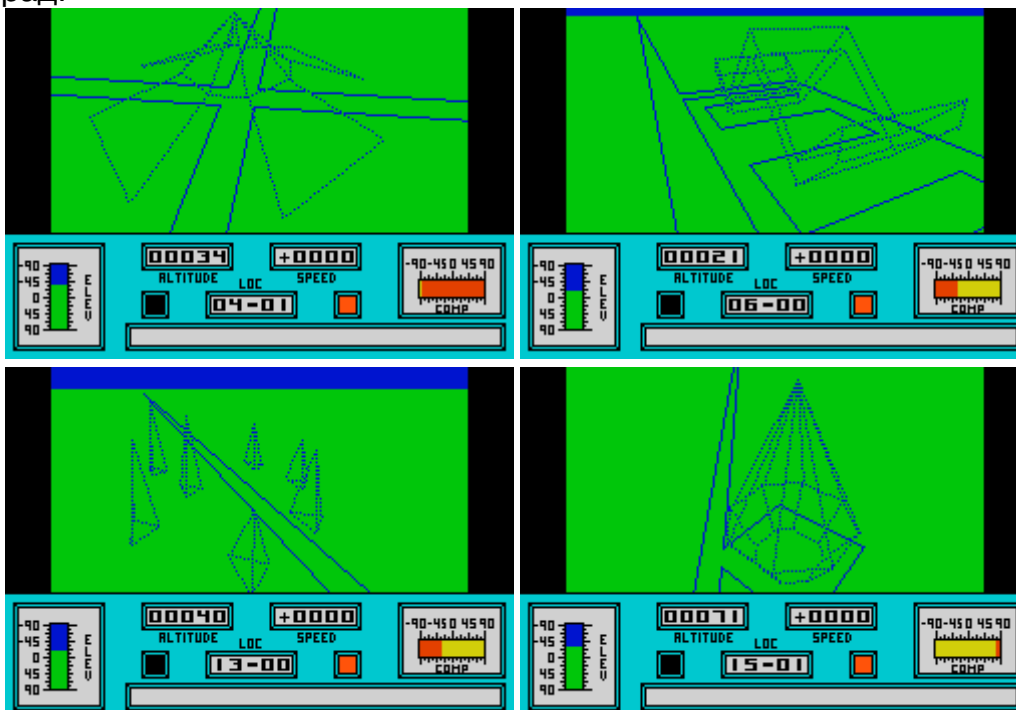


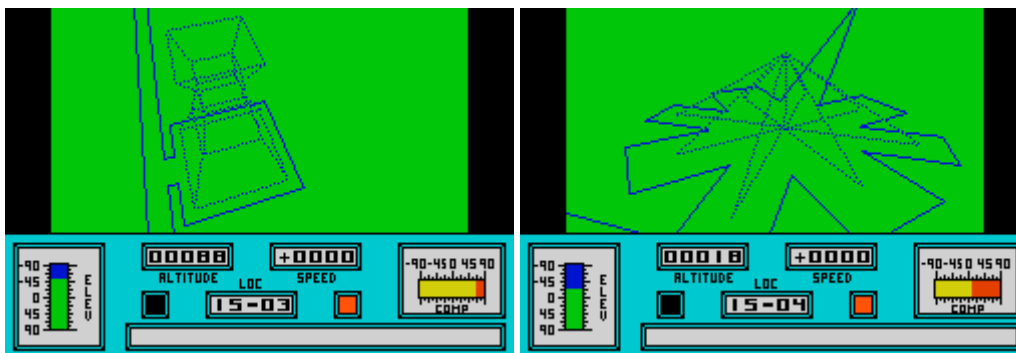
Рис. 1

- 1 - прибор, показывающий угол тангажа
- 2 - высотомер
- 3 - локатор
- 4 - спидометр
- 5 - компас
- 6 - строка информационных сообщений
- 7 - индикатор снижения высоты (действует, когда высота меньше 300 метров).
- 8 - индикатор металлоискателя.

### Описание транспортных средств и предметов.

Итак, очутившись на поверхности незнакомой планеты, Вы получили предложение купить мобильный самолет, что и делаете нажатием клавиши Y. Этот мобильный самолет имеет фантастические возможности. Он за секунду разгоняется до 1500 км/час (!) и за столько же сбрасывает скорость до нуля. Можно плавным увеличением 4-ой скорости разогнаться и до 4900 км/час. Можно даже зависнуть на любой высоте и, кувыряясь во все стороны, как внутри шара, осматривать местность под любыми углами. Управление очень простое. Единственно, при поездке нужно выбирать 1-ую или 2-ую скорость с углом тангажа не более 15 град.





Имеется еще один самолет. Он находится в ангаре с координатами по локатору "\*\*\*-\*\*\*". Внешне он похож на современный истребитель. Его отличительная особенность - большая скорость (более 7000 км/час) и он позволяет взлетать на высоту до 99999.

В квадрате 03-00 в ангаре находится летающая тарелка. Основное отличие от самолетов - низкая скорость (1800 км/час), отсутствие крена при поворотах и почти полное отсутствие инерции.

В квадрате (координаты по локатору 09-05) в ангаре находится TURBO-автомобиль, который развивает скорость по земле до 1800 км/час.

В квадрате 11-13 возле ангара у дома стоит автомобиль. Он может разгоняться до 300 км/час.

Всего в игре имеется 7 ангаров с лабиринтами. Шесть подземных ангаров в квадратах: 09-06, 09-05, 03-00, 11-13, 81-35, \*\*-\*\* и один ангар находится на высоте 64997 метров над квадратом 06-08.

Метод отыскания ангаров. Вход в ангар выглядит, как маленькая площадка, обнесенная с трех сторон забором. Если ангар в черте города, то находится он возле дороги.

Примерно через две минуты после покупки самолета Вам сообщают, что нужно лететь в квадрат 09-06. Наберите высоту примерно 100 метров и, маневрируя самолетом, войдите в этот квадрат. Следите за высотой, можно разбиться. Попад в нужный квадрат, погасите скорость до нуля и осмотритесь, в поисках входа в ангар. Осторожно садитесь на землю и подкатите на колесах ко входу. Можно оставить самолет (или другой транспорт) на поверхности в удобном месте, а можно вместе с самолетом заехать на лифтовую площадку и опуститься в ангар. В ангаре и в лабиринтах транспорт не действует, поэтому нужно из него выйти и дальше идти пешком.

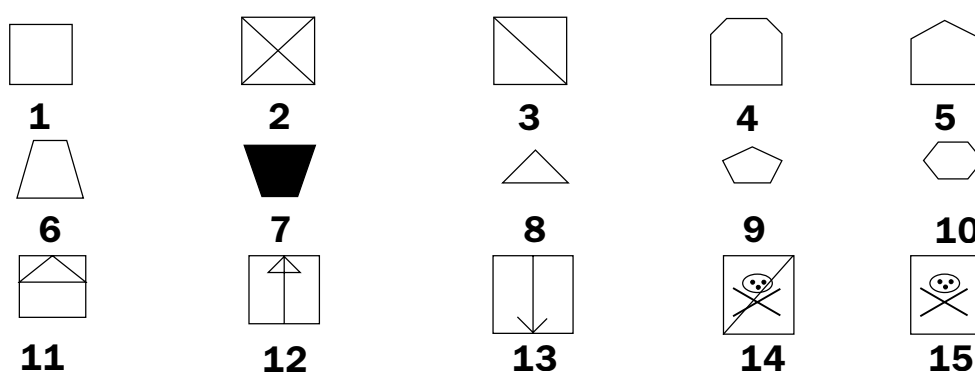


Рис.2

| Обозначение | Назначение                                                                                                                      |
|-------------|---------------------------------------------------------------------------------------------------------------------------------|
| 1           | Прямоугольная дверь. Можно войти без ключа.                                                                                     |
| 2           | Телепортатор двухстороннего действия (туда и обратно).                                                                          |
| 3           | Телепортатор одностороннего действия (туда)                                                                                     |
| 4           | При попытке в нее войти будет сообщение LOCKED (закрыто). Чтобы пройти нужно иметь с собой кристалл такой же формы как и дверь. |
| 5           | то же                                                                                                                           |
| 6           | то же                                                                                                                           |

|    |                                                                                                                                                                                                                                                                                                                                     |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7  | то же                                                                                                                                                                                                                                                                                                                               |
| 8  | то же                                                                                                                                                                                                                                                                                                                               |
| 9  | то же                                                                                                                                                                                                                                                                                                                               |
| 10 | то же                                                                                                                                                                                                                                                                                                                               |
| 11 | При входе в нее будет очень темно, поэтому нужно выйти задом. Необходимо иметь с собой PHOTON EMITER (фотонный прожектор) для прохождения в эту дверь                                                                                                                                                                               |
| 12 | Вход в лифт, поднимающий вверх.                                                                                                                                                                                                                                                                                                     |
| 13 | Вход в лифт, опускающий вниз.                                                                                                                                                                                                                                                                                                       |
| 14 | Войдя в нее Вы телепортируетесь и перепутается структура лабиринта относительно компаса (запад на восток), войдя во 2-ой раз, ситуацию можно исправить.                                                                                                                                                                             |
| 15 | Если эта дверь находится в лабиринте, который находится на высоте 64997, то Вы вывалитесь и упадете на землю и растеряете все предметы которые были с Вами. Если эта дверь находится в подземном лабиринте, тогда из этой комнаты нет обратного пути. И придется вызывать самолет с потерей всех предметов или заново вводить игру. |

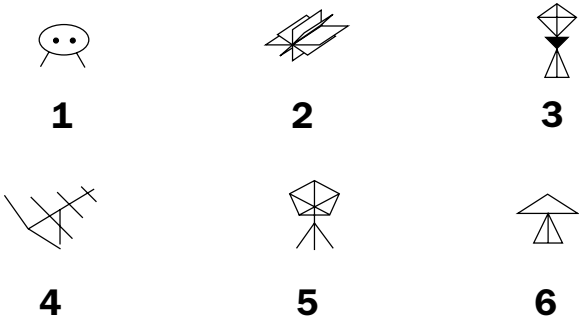
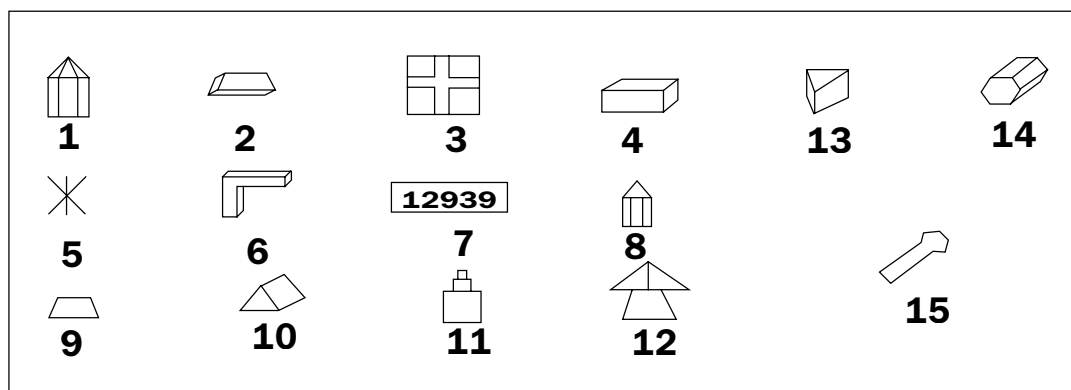


Рис. 3

| Обозначение | Название                               | Назначение                                                                                                                                                                         | Где найти |
|-------------|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|
| 1           | SIGHTS<br>зрение                       | Включает прицел для точного наведения на объект                                                                                                                                    | 09-06     |
| 2           | POWERAMP<br>турбина                    | Дает возможность мобильному самолету развивать скорость до 9500 км/час и взлетать на высоту 99999 м.                                                                               | 09-06     |
| 3           | ANTIGRAV<br>антигравитатор             | С его помощью можно брать с собой тяжелые предметы: такие как "нейтронное топливо" и все виды транспорта.                                                                          | 09-05     |
| 4           | ANTENNA<br>антенна                     | Имея ее в COMMUNICATION ROOM (квадрат 09-06) Вы можете купить ракету за 999999 кредитов Вам дадут информацию о ее местонахождении для того, чтобы на этой ракете завершить миссию. | 09-05     |
| 5           | METAL<br>DETECTOR<br>металлоискатель   | Обнаруживает над землей вход в любой ангар                                                                                                                                         | 09-05     |
| 6           | PHOTON<br>EMITER<br>фотонный прожектор | Необходим для прохождения комнат с дверями (11)                                                                                                                                    | 09-05     |



| Обозначение | Название                                         | Где взять | Где продать                             |
|-------------|--------------------------------------------------|-----------|-----------------------------------------|
| 1           | ENERGY CRYSTAL<br>энергетический кристалл        | 09-06     | либо в 03-00, либо в POWER ROOM (08-08) |
| 2           | CATERING PROVISION<br>провиант                   | 09-06     | в KETCHIN (08-08)                       |
| 3           | MEDICAL SUPPLIES<br>медикаменты                  | 09-06     | в INFIRMARI (08-08)                     |
| 4           | LARGE BOX<br>большой ящик                        | 09-06     | в INFIRMARI (08-08)                     |
| 5           | NEUTRON FUEL<br>нейтронное топливо               | 09-05     | в ENGINE ROOM (08-08)                   |
| 6           | USEFUL ARMAMENT<br>оружие                        | 11-13     | в ARMOUERY (03-00) или (08-08)          |
| 7           | ESSENTIAL 12939<br>SUPPLY PEPSI                  | 03-00     | в CONFERENCE ROOM (08-08)               |
| 8           | MECHANOID робот                                  | 03-00     | в INTERVIEW ROOM (08-08)                |
| 9           | GOLD золото                                      | 81-35     | в EXCHEQVER (08-08)                     |
| 10          | DATA BANK банк данных                            | **_**     | в CONTROL ROOM (08-08)                  |
| 11          | WINCHESTER<br>винчестер                          | **_**     | в LABORATORY (03-03) или (08-08)        |
| 12          | ANTI TIME BOMB<br>антибомба с часовым механизмом | 08-08     | ?                                       |
| 13          | CHEESE сыр                                       | 11-13     | ?                                       |
| 14          | NOVADRIVE<br>управление                          | **_**     | ?                                       |
| 15          | COFFIN гроб                                      | 09-05     | ?                                       |

Ангар в квадрате \*\*\_\*\* обнаруживается так. Сначала надо долететь до квадрата 99-99, потом, при появлении квадрата \*\*\_\*\* сбросить скорость и с высоты около 4000 метров посмотреть, где внизу вход в ангар.

Точка, висящая над городом это тоже ангар. Он находится на высоте 64997 метров. Нужно взлететь на эту высоту. Вы увидите в воздухе большую плиту, садитесь на нее. В одном из углов плиты есть квадратная площадка, являющаяся входом.

Теперь о дверях, которые встречаются в лабиринтах. В таблице (рис. 2) даны краткие сведения о дверях:

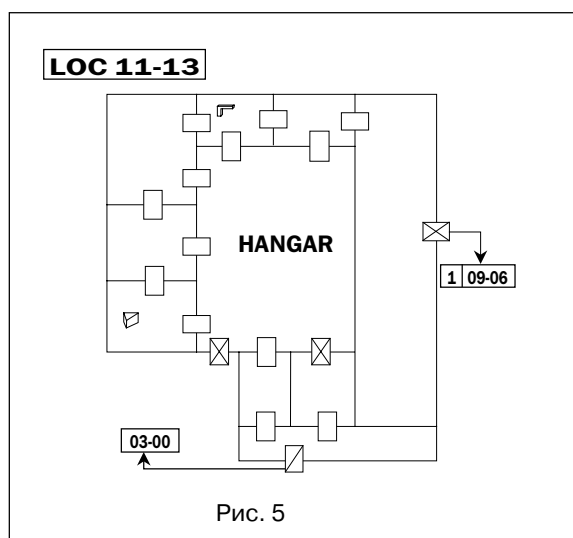
Все предметы, с которыми Вы столкнетесь по ходу игры, можно условно разделить на два класса. В одном классе предмет - товар, в другом классе предмет - навигационное средство.

В таблицу (рис. 3) внесены предметы, необходимые для навигации.

А теперь рассмотрим предметы, интересующие покупателей. Их можно использовать для продажи. В следующей таблице (рис. 4) дана информация о том, где взять товар и где его продать.

О предметах можно сообщить еще такую особенность. Если на базе (квадрат 03-00) взять MECHANOID (робота), то после этого нельзя здесь продавать предметы. С собой можно одновременно иметь не более десяти предметов. В таблице против некоторых предметов стоят знаки "?". Пока значение этих предметов невыяснено. Может быть, кто-нибудь выяснит этот вопрос?

В заключение прилагаю карты лабиринтов (рис. 5 ... 12).



На картах лабиринтов есть односторонние и двусторонние телепортаторы, не отмеченные куда они телепортируют. Некоторые из них телепортируют каждый раз в другое место а некоторые еще не изучены. Есть и другие неисследованные вопросы, например: почему в городе все строения можно сбивать выстрелами своего оружия? Что за самолет, постоянно летающий над городом по одному и тому же курсу на высоте 500 метров? Можно ли увидеть самого себя?

Успеха Вам! GOOD LUCK!

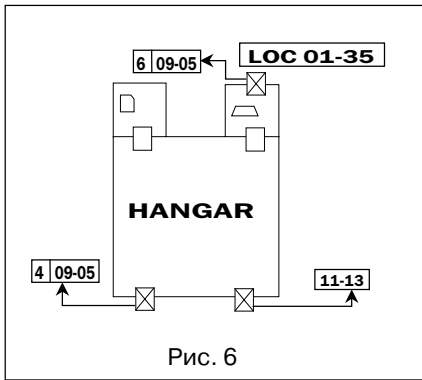


Рис. 6

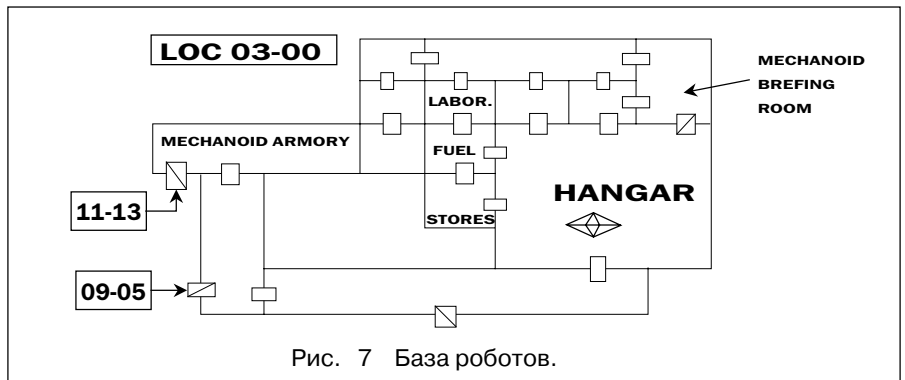


Рис. 7 База роботов.

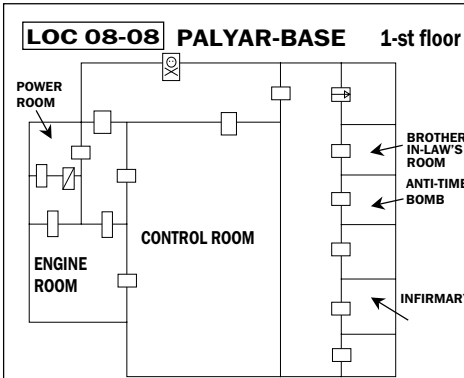


Рис. 8a

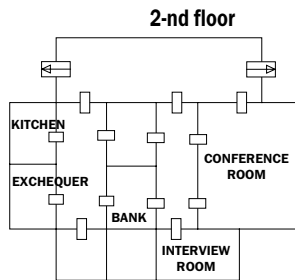


Рис. 8b

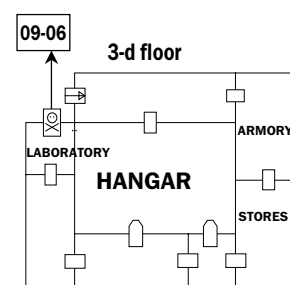


Рис. 8c

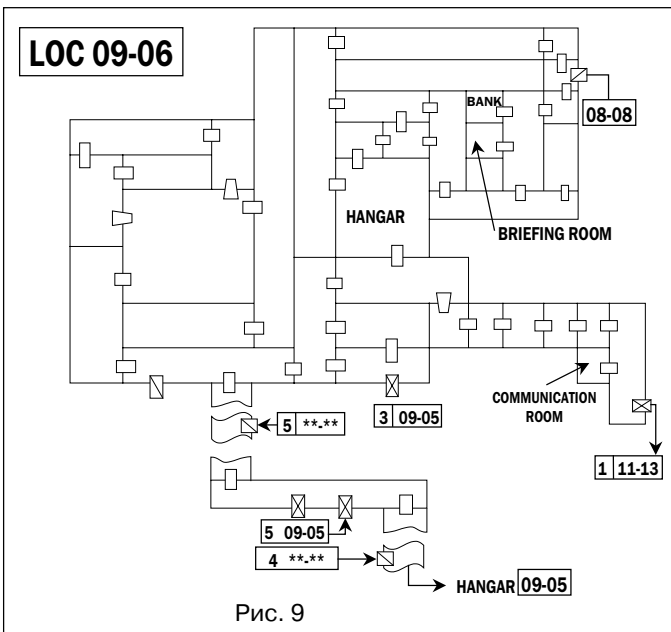


Рис. 9

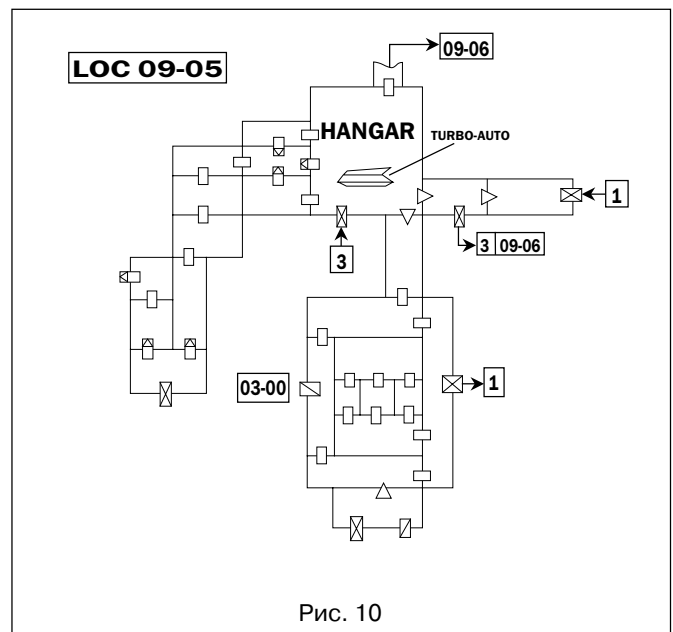


Рис. 10

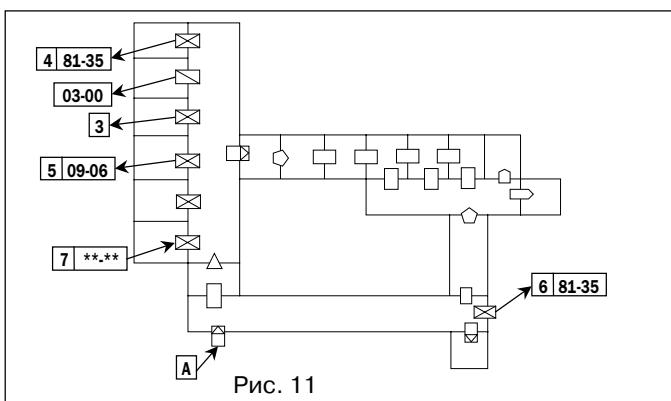


Рис. 11

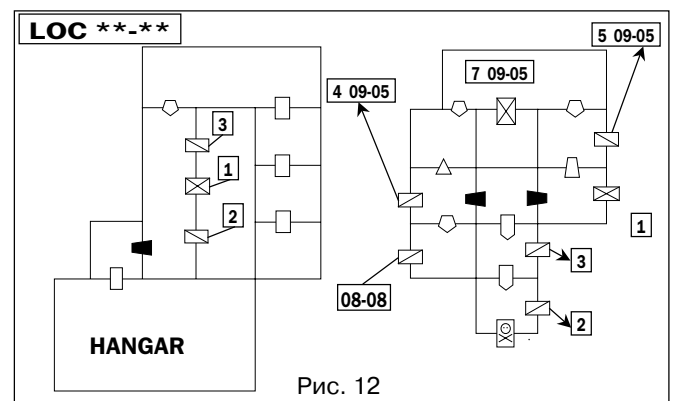


Рис. 12



# Сделайте сами

## DICEY

Сегодня в рубрике "Сделайте сами" мы предлагаем интересную настольную игру в кости. Это вариация на тему известного "покера". Тем, кто знает, что это такое, наверное ничего объяснять не надо, для тех же, кто с ним не знаком, поясним, что бросая пять игровых костей, нужно стараться создать на них одну из комбинаций выпавших очков. Для того, чтобы это было возможно, после первого броска разрешается перебросить те кости, которые Вас не устраивают. О том, что это за комбинации и как они оцениваются, мы говорить не будем, в листинге программы эта информация есть.

Покер требует в определенной степени умения мыслить комбинаторно, принимать правильные решения. Известно, что против теории вероятности не пойдешь, но тем не менее, есть игроки, которые играют лучше, а есть играющие хуже. Очевидно, первые придерживаются какой-то выигрышной стратегии, а вторые ее еще не выработали.

Данная игра дает двойную возможность для реализации Вашей стратегии. Дело в том, что за набранные комбинации Вы не просто получаете очки, а имеете возможность передвигать по результатам броска фишки на игровом поле. Надо еще хорошо подумать, прежде чем добиваться той или иной комбинации, к чему Вам следует стремиться. Это зависит от конкретной ситуации на игровом поле.

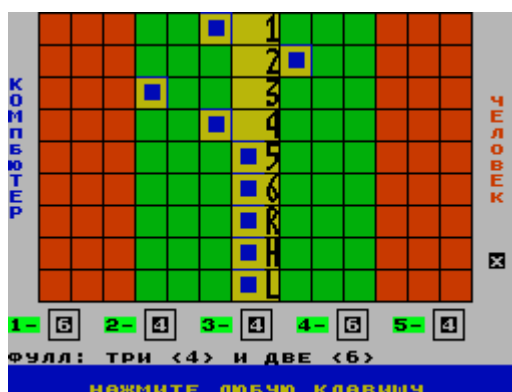
Игра нравится даже тем, кому вообще игровые программы не нравятся, и мы надеемся, что и Вам она доставит много радости.

Хочется сказать несколько слов о проблеме "ошибок в листингах". Это бич всех компьютерных журналов и у нас в "ZX-РЕВЮ" это случалось. Трудоемкость подготовки статьи с игрой для Вас, уважаемые читатели, примерно в три раза выше, чем любой другой аналогичной статьи в связи с необходимостью многочисленных проверок, перепроверок и отладок и, тем не менее, Вы сами знаете, что ошибки проникали в окончательный текст. Это досадно, хотя и не вредно для тех, кто хочет не просто набрать программу, но при этом еще и освоить программирование.

Сейчас мы изменили технологию подготовки программ для самостоятельного набора и ошибки практически исключены. Можете смело набирать и запускать программу. Если что-то не работает, проверяйте правильность ввода.

Наша технология такова: сначала программа набирается, отлаживается, русифицируется на "Спектруме". Когда она полностью работоспособна, она "сбрасывается" на диск (TR-DOS), после чего программными средствами перекодируется "символика" и конвертируется формат записи в MSDOS. В виде файла MSDOS она верстается и распечатывается на IBM-компьютере. Поскольку все операции автоматические, то листинг, который Вы видите перед собой, точно соответствует тому, что было в памяти "Спектрума", когда программа была отлажена и работала, и ошибок здесь просто не должно быть.

Несколько слов о русификации. Мы придерживаемся ранее принятого нами способа русификации. Если Вы с ним не знакомы, Вам понадобятся выпуски ZX-РЕВЮ N3, 4 за этот год и, может быть, N 1,2 тоже за этот год.



```

1 GO TO 3
2 CLEAR 64599: LOAD "chr"CODE 64600
3 BORDER 1: PAPER 1: CLS : POKE 23658,8: POKE 23606,88: POKE 23607,251:
4 PRINT AT 10,5; INK 3; BRIGHT 1;"ВАМ НУЖНА ИНСТРУКЦИЯ?";AT 12,6;"НАЖМИТЕ <Y> ИЛИ <N>": BEEP
   .5,3: BEEP .35,0
5 IF INKEY$="" THEN GO TO 5
6 IF INKEY$="Y" THEN GO SUB 9100: GO TO 8
7 GO SUB 9270
8 CLS : BORDER 1: INK 0: PAPER 7
9 PRINT AT 10,0; FLASH 1; INK 1; PAPER 6; "СЕЙЧАС Я ПОДГОТОВЛЮ ИГРОВОЕ ПОЛЕ"
10 GO SUB 9900: REM U.D.G. ====
15 RANDOMIZE
20 DIM T(5):DIM R(5):DIM Z$(6) :LET T$=CHR$ 132+CHR$ 136:LET B$ =CHR$ 129+CHR$ 130
30 DIM D(5,2,1): FOR I=1 TO 5: FOR N=1 TO 2: READ P: LET D(I,N ,1)=P: NEXT N: NEXT I
34 REM ПОДГОТОВКА КООРДИНАТ ИГРОВОГО ПОЛЯ И ИЗОБРАЖЕНИЯ ФИШЕК=
35 DIM Y(9): DIM X(13)
40 FOR I=0 TO 8: LET Y(I+1)=2*I: NEXT I
50 FOR I=1 TO 7: LET X(I)=2*I: NEXT I: FOR I=8 TO 13: LET X(I) =2*I+1: NEXT I
60 LET E1=0: LET E2=0: DIM F(9): FOR I=1 TO 9: LET F(I)=7: NEXT I
65 CLS : IF PLAY=2 THEN LET GO=1: PRINT AT 10,6; INK 1;"ПЕРВЫЙ ИГРОК НАЧИНАЕТ": PAUSE 75
66 IF PLAY=1 AND GO=1 THEN PRINT AT 10,7; INK 1;"КОМПЬЮТЕР НАЧИНАЕТ": PAUSE 75
67 IF PLAY=1 AND GO=-1 THEN PRINT AT 10,4; INK 1;"О.К. - ВЫ ХОДИТЕ ПЕРВЫМ": PAUSE 75
70 GO SUB 9000: REM ИГРОВОЕ ПОЛЕ =====
80 GO SUB 2000: REM ИЗОБРАЖЕНИЕ КОСТЕЙ И ФИШЕК =====
90 GO SUB 4500: REM ПРОВЕРКА НА ПОБЕДУ =====
100 LET GO=GO*-1
200 PRINT AT 21,0; INK 1; PAPER 6; "                ПЕРВЫЙ БРОСОК                "
202 FOR N=1 TO 10: FOR I=1 TO 5: LET R(I)=1+INT (RND*6): NEXT I
205 BEEP .01,INT (RND*10)
210 GO SUB 3500: NEXT N
230 IF PLAY=2 OR GO=1 THEN GO SUB 3000
231 IF PLAY=1 AND GO=-1 THEN GO SUB 3600: GO SUB 5000: GO SUB 5500
234 IF M=0 THEN GO TO 310
235 PRINT AT 21,0; INK 1; PAPER 6;"                ВТОРОЙ БРОСОК                "
236 FOR N=1 TO 10
240 GO SUB 2400
250 GO SUB 3500
255 NEXT N
270 IF PLAY=2 OR GO=1 THEN GO SUB 3000
271 IF PLAY=1 AND GO=-1 THEN GO SUB 3600: GO SUB 5000: GO SUB 5500
274 IF M=0 THEN GO TO 310
275 PRINT AT 21,0; INK 1; PAPER 6;"                ТРЕТИЙ БРОСОК                "
276 FOR N=1 TO 10
280 GO SUB 2400
290 GO SUB 3500
295 NEXT N
310 GO SUB 3600
315 IF PLAY=1 AND GO=-1 THEN GO SUB 6300
320 PRINT AT 21,0;X$
330 PRINT #1; INK 6; "                НАЖМИТЕ ЛЮБУЮ КЛАВИШУ                "
340 PAUSE 0: INPUT ;
350 IF GO=-1 THEN GO TO 370
360 IF PLAY=2 THEN PRINT INK 1; AT 21,0;"                БРОСАЕТ ПЕРВЫЙ ИГРОК                "
361 IF PLAY=1 THEN PRINT INK 1; AT 21,0;"                БРОСАЕТ КОМПЬЮТЕР                "
365 GO TO 70
370 IF PLAY=2 THEN PRINT INK 2; AT 21,0;"                БРОСАЕТ ВТОРОЙ ИГРОК                "
371 IF PLAY=1 THEN PRINT INK 2; AT 21,0;"                БРОСАЕТ ЧЕЛОВЕК                "
380 GO TO 70
2000 REM ИЗОБРАЖЕНИЕ КОСТЕЙ И ФИШЕК
2010 FOR I=1 TO 5: PLOT D(I,1,1),D(I,2,1) : DRAW 15,0: DRAW 0,-15: DRAW -15,0: DRAW 0,15:
   PRINT BRIGHT 1; INK 0; PAPER 4;AT 19,6*(I-1);I;"-": NEXT I
2020 FOR I=1 TO 9
2025 IF I=E1 OR I=E2 THEN NEXT I
2026 IF I=10 THEN GO TO 2040
2030 PRINT OVER 1; INK 1; PAPER 6;AT Y(I),X(F(I));T$;AT Y(I)+1, X(F(I));B$: NEXT I

```

```

2040 IF NOT E1 AND NOT E2 THEN RETURN
2050 LET DISP1=DISP1*G0: LET DISP2=DISP2*G0
2060 LET F(E1)=F(E1)+DISP1
2070 IF F(E1)<1 THEN LET F(E1)=1
2080 IF F(E1)>13 THEN LET F(E1)=13
2090 PRINT OVER 1; INK 1; PAPER 6; AT Y(E1),X(F(E1));T$;AT Y(E1)+1,X(F(E1));B$
2100 IF NOT E2 THEN RETURN
2110 LET F(E2)=F(E2)+DISP2
2120 IF F(E2)<1 THEN LET F(E2)=1
2130 IF F(E2)>13 THEN LET F(E2)=13
2140 PRINT OVER 1; INK 1; PAPER 6;AT Y(E2),X(F(E2));T$;AT Y(E2)+1,X(F(E2));B$
2150 RETURN
2400 REM СБРОС ЗНАЧЕНИИ КОСТЕЙ==
2410 FOR I=1 TO M: LET R(T(I))=1+INT (RND*6): NEXT I
2420 BEEP .01,INT (RND*10)
2430 RETURN
2999 REM ПОВТОРНЫЙ БРОСОК =====
3000 LET R$="": INPUT AT 0,0;"ВВЕДИ Y ЕСЛИ О.К. ИЛИ N ЕСЛИ НЕТ";LINE R$
3010 IF R$<>"Y" AND R$<>"N" THEN GO TO 3000
3020 IF R$="Y" THEN GO TO 310
3030 INPUT "СКОЛЬКО КОСТЕЙ ПЕРЕБРОСИТЕ? ";M
3040 LET M=INT M: IF M<1 OR M>5 THEN GO TO 3030
3045 PRINT #1; FLASH 1; INK 2; PAPER 7;"БУДУТ ПЕРЕБРАСЫВАТЬСЯ КОСТИ ... ": PAUSE 50
3050 FOR I=1 TO M: INPUT "КОСТЬ НОМЕР : ";T(I): LET T(I)=INT T(I)
3055 IF T(I)<1 OR T(I)>5 THEN GO TO 3050
3060 NEXT I: RETURN
3499 REM ПЕЧАТЬ ЗНАЧЕНИЙ КОСТЕЙ=
3500 FOR I=1 TO 5: PRINT AT 19,6*I-3; INK 7; PAPER 0;R(I): NEXT I
3510 RETURN
3599 REM СОРТИРОВКА ЗНАЧЕНИЙ КОСТЕЙ =====
3600 FOR N=1 TO 4: FOR I=1 TO 5-N
3610 IF R(I)<R(I+1) THEN LET A=R(I): LET R(I)=R(I+1): LET R(I+1)=A
3620 NEXT I: NEXT N
3629 REM HOLD HOW MANY OF EACH==
3630 DIM V(5,2): LET COUNT=1: LET A=0
3640 FOR N=1 TO 5
3650 IF N=5 THEN GO TO 3700
3660 FOR I=N TO 4
3670 IF R(I)=R(I+1) THEN LET COUNT=COUNT+1: GO TO 3690
3680 GO TO 3700
3690 NEXT I
3700 LET A=A+1: LET V(A,2)=COUNT: LET V(A,1)=R(N)
3710 LET N=N+COUNT-1: LET COUNT=1
3720 NEXT N
3769 REM НАЗНАЧАЕМ ПЕРЕМЕННОЙ X$ КОМБИНАЦИЮ КОСТЕЙ И УСТАНОВЛИВАЕМ СЧЕТЧИКИ ПЕРЕМЕЩЕНИЯ
      ФИШЕК ==
3790 DIM X$(32): LET E1=0: LET E2=0: LET DISP1=0: LET DISP2=0
3800 IF V(1,2)<>5 THEN GO TO 3820
3801 REM ПЯТЕРКИ =====
3802 LET X$="ПЯТЕРКА <"+STR$ V(1,1)+"> !!!"
3804 LET DISP1=V(1,2)-1: LET E1=V(1,1): GO TO 3930
3820 IF V(1,2)<>4 AND V(2,2)<>4 THEN GO TO 3830
3821 REM ЧЕТВЕРКИ =====
3822 LET DISP1=3: IF V(1,2)=4 THEN LET E1=V(1,1)
3823 IF V(2,2)=4 THEN LET E1=V(2,1)
3824 LET X$="ЧЕТВЕРКА <"+(STR$ V(1,1) AND V(1,2)=4)+(STR$ V(2,1) AND V(2,2)=4)+">": GO TO
      3930
3830 IF V(1,2)<>3 AND V(1,2)<>2 OR V(2,2)<>3 AND V(2,2)<>2 OR V(1,2)=V(2,2) THEN GO TO 3850
3834 REM ФУЛЛ =====
3835 LET X$="ФУЛЛ: ТРИ <"+(STR$ V(1,1) AND V(1,2)=3)+(STR$ V(2,1) AND V(2,2)=3)+"> И ДВЕ
      <"+(STR$ V(1,1) AND V(1,2)=2)+(STR$ V(2,1) AND V(2,2)=2)+">"
3838 FOR I=1 TO 2
3840 IF V(I,2)=3 THEN LET DISP1= V(I,2)-1: LET E1=V(I,1)
3843 IF V(I,2)=2 THEN LET DISP2= V(I,2)-1: LET E2=V(I,1)
3845 NEXT I: GO TO 3930

```

```

3849 REM ТРОЙКИ =====
3850 FOR I=1 TO 3: IF V(I,2)=3 THEN LET DISP1=V(I,2)-1: LET E1=V(I,1): LET X$="ТРОЙКА
    <" + STR$ V(I,1) + ">"
3851 NEXT I
3854 REM ПАРА ИЛИ ДВЕ ПАРЫ =====
3855 LET A=0: FOR I=1 TO 5: IF V(I,2)=2 THEN LET A = A+1: LET B=I
3856 NEXT I
3860 IF A=0 THEN GO TO 3930
3870 IF A=2 THEN GO TO 3900
3880 LET E1=V(B,1): LET DISP1=V(B,2)-1
3895 LET X$="ПАРА <" + STR$ V(B,1) + ">": GO TO 3930
3900 LET A=0: FOR I=1 TO 5
3905 IF V(I,2)=2 AND NOT A THEN LET A=I
3910 IF V(I,2)=2 AND A THEN LET B=I
3915 NEXT I
3920 LET X$="ДВЕ ПАРЫ, <" + STR$ V(A,1) + "> И <" + STR$ V(B,1) + ">"
3925 LET DISP1=V(A,2)-1: LET E1=V(A,1): LET DISP2=V(B,2)-1: LET E2=V(B,1)
3929 REM ИНИЦИАЛИЗАЦИЯ СУММЫ И МАРКЕРОВ БОЛЬШОЙ И МАЛОЙ СУММЫ==
3930 LET TOTAL=0: LET HIGH=0: LET LOW=0: FOR I=1 TO 5
3940 LET TOTAL=TOTAL+R(I): NEXT I
3950 IF TOTAL>12 AND TOTAL<23 THEN GO TO 3995
3955 IF TOTAL>25 THEN LET HIGH=3
3960 IF TOTAL>22 AND TOTAL<26 THEN LET HIGH=2
3965 IF TOTAL<10 THEN LET LOW=3
3970 IF TOTAL>9 AND TOTAL<13 THEN LET LOW=2
3975 IF CODE X$(1)<>32 THEN GO TO 3985
3979 REM БОЛЬШАЯ ИЛИ МАЛАЯ СУММА
3980 GO SUB 4010: RETURN
3984 REM ВЫБОР КОМБИНАЦИИ =====
3985 IF PLAY=1 AND GO=-1 THEN GO TO 3994
3986 PRINT AT 21,0;"ВЫБИРАЙТЕ, НА ЧТО ВЫ ИДЕТЕ..... ": PAUSE 100 : PRINT AT 21,0;"СУММА /
    ";X$(1 TO 22);"?
3990 INPUT "S" ЕСЛИ СУММА, ИНАЧЕ "X": "; LINE R$
3991 IF R$<>"S" AND R$<>"X" THEN GO TO 3990
3993 IF R$="S" THEN GO SUB 4010
3994 REM СТРИТ =====
3995 FOR I=1 TO 4: IF R(I)<>R(I+1)+1 THEN GO TO 3998
3996 NEXT I
3997 LET DISP1=(3 AND R(1)=6)+(2 AND R(1)=5): LET E1=7: LET X$=("БОЛЬШОЙ" AND
    R(1)=6)+("МАЛЫЙ" AND R(1)=5)+" СТРИТ"
3998 IF CODE X$(1)=32 THEN LET X$="          Н И Ч Е Г О !"
4000 RETURN
4010 LET X$=("БОЛЬШАЯ СУММА =" AND HIGH)+("МАЛАЯ СУММА =" AND LOW)+STR$ TOTAL
4020 LET DISP1=(HIGH AND HIGH)+(LOW AND LOW): LET DISP2=0
4030 LET E1=(8 AND HIGH)+(9 AND LOW): LET E2=0
4040 RETURN
4500 REM ПРОВЕРКА ВЫИГРЫША =====
4510 LET A=0: LET B=0
4520 FOR I=1 TO 9
4530 IF F(I)<4 THEN LET A=A+1
4540 IF F(I)>10 THEN LET B=B+1
4550 NEXT I
4560 IF A>2 OR B>2 THEN GO TO 4575
4570 RETURN
4575 FOR I=1 TO 30
4580 IF PLAY=2 THEN PRINT AT 21,5; FLASH 1; BRIGHT 1; INK 1; PAPER 6; "ПОБЕДИЛ "+("ПЕРВЫЙ"
    AND A>2)+("ВТОРОЙ" AND B>2)+" ИГРОК"
4581 IF PLAY=1 THEN PRINT AT 21,5; FLASH 1; BRIGHT 1; INK 1; PAPER 6; ("ПОБЕДИЛ
    КОМПЬЮТЕР!!!" AND A>2)+("ВЫ ВЫИГРАЛИ! ПОЗДРАВЛЯЮ." AND B>2)
4585 BEEP .01,I: NEXT I
4590 LET R$="": INPUT "СЫГРАЕМ ЕЩЕ? ('Y' ИЛИ 'N') "; LINE R$
4600 IF R$="" THEN GO TO 4590
4610 IF R$="Y" THEN GO TO 4660
4620 LET R$="": INPUT "КОНЕЦ? ('Y' ИЛИ 'N') "; LINE R$
4630 IF R$="" THEN GO TO 4620

```

```

4640 IF R$="Y" THEN GO TO 9998
4650 GO TO 4590
4660 CLS : PRINT ""ВЫ МОЖЕТЕ ИЗ
МЕНИТЬ РЕЖИМ ИГРЫ.""СНАЧАЛА ВВ
ЕДИТЕ ""1"" ИЛИ ""2""""ПО ЧИСЛ
У ИГРАЮЩИХ; ЗАТЕМ""(ЕСЛИ ВЫ ИГ
РАЕТЕ С КОМПЬЮТЕРОМ)""ВВЕДИТЕ
""-1"" ЕСЛИ ХОТИТЕ ХОДИТЬ""ПЕР
ВЫЙ. ИНАЧЕ ВВЕДИТЕ ""1""
4670 INPUT "ОДИН ИЛИ ДВА-ИГРОКА?"; PLAY
4680 LET PLAY=INT PLAY: IF PLAY<1 OR PLAY>2 THEN GO TO 4670
4690 IF PLAY=2 THEN GO TO 60
4700 INPUT " -1, ЕСЛИ ХОДИТЕ ПЕРВЫМ, ИНАЧЕ 1:"; GO
4710 LET GO=INT GO: IF GO=0 OR GO>1 OR GO<-1 THEN GO TO 4700
4720 GO TO 60
4999 REM УСТАНОВКА МАРКЕРОВ ПРИОРИТЕТОВ =====
5000 DIM P(9): LET A1=0: LET A2=0
5010 FOR I=1 TO 9: REM ЕСТЬ ЛИ ФИШКИ В КРАСНОЙ ЗОНЕ? =====
5020 IF F(I)<4 THEN LET A1=A1+1
5030 IF F(I)>10 THEN LET A2=A2+1
5040 NEXT I
5050 IF A1<>2 OR A2<>2 THEN GO TO 5210
5060 LET B1=7
5070 FOR I=1 TO 9
5080 IF F(I)<B1 AND F(I)>3 THEN LET B1=F(I): LET B2=I
5090 NEXT I
5100 IF B1=7 THEN GO TO 5130
5110 IF B2<7 THEN LET P(B2)=1: RETURN
5120 LET P(B2)=B2: RETURN
5130 LET B1=14
5140 FOR I=1 TO 6
5150 IF F(I)<B1 AND F(I)>10 THEN LET B1=F(I): LET B2=I
5160 NEXT I: IF B1=14 THEN GO TO 5180
5170 LET P(B2)=1: RETURN
5180 FOR I=7 TO 9
5190 IF F(I)<B1 AND F(I)>10 THEN LET B1=F(I): LET B2=I
5200 NEXT I: LET P(B2)=B2: RETURN
5210 IF A1<>2 THEN GO TO 5280
5220 LET B1=14
5230 FOR I=1 TO 9
5240 IF F(I)<B1 AND F(I)>3 THEN LET B1=F(I): LET B2=I
5250 NEXT I
5260 IF B2>=7 THEN LET P(B2)=B2: RETURN
5270 LET P(B2)=1: RETURN
5280 IF A2<>2 THEN GO TO 5380
5290 LET A2=0: FOR I=7 TO 9
5300 IF F(I)>10 THEN LET A2=A2+1
5310 NEXT I: IF A2<2 THEN GO TO 5350
5320 FOR I=7 TO 9
5330 IF F(I)>10 THEN LET P(I) = 1
5340 NEXT I: RETURN
5350 LET B1=14: FOR I=1 TO 6
5360 IF F(I)>10 AND F(I)<B1 THEN LET B1=F(I): LET B2=I
5370 NEXT I: LET P(B2)=1: RETURN
5380 FOR I=1 TO 9
5390 IF F(I)=1 THEN LET P(I)=3
5400 IF F(I)=2 OR F(I)=3 THEN LET P(I)=2
5410 NEXT I: RETURN
5499 REM РАЧЕТ ПРИОРИТЕТА=1 ===
5500 LET A=0: LET M=0: FOR I=1 TO 6
5510 IF P(I)=1 THEN LET A=I
5520 NEXT I: IF A=0 THEN GO TO 5570
5530 FOR I=1 TO 5
5540 IF R(I)<>A THEN GO SUB 6210
5550 NEXT I: IF M=0 THEN GO TO 310

```

```

5560 RETURN
5570 IF P(7)>I THEN GO TO 5640: REM НЕТ СМЫСЛА ДЕЛАТЬ СТРИТ ===
5580 IF E1=7 THEN GO TO 310
5590 IF E1=0 AND E2=0 THEN LET M=1: LET T(1)=5: RETURN
5600 IF P(7)<>7 THEN GO TO 5640 : REM СТРИТ НЕРЕАЛЕН =====
5610 FOR I=1 TO 4
5620 IF R(I)=E1 THEN LET M=1: LET T(1)=1: RETURN
5630 NEXT I
5640 IF P(8)<>8 OR TOTAL<20 THEN GO TO 5720: REM БОЛЬШАЯ СУММА СОМНИТЕЛЬНА
      =====
5650 IF TOTAL>23 THEN GO TO 5690
5660 FOR I=3 TO 5
5670 IF R(I)<4 THEN GO SUB 6210
5680 RETURN
5690 FOR I=4 TO 5
5700 IF R(I)<3 THEN GO SUB 6210
5710 RETURN
5720 IF P(9)<>9 OR TOTAL>15 THEN GO TO 5800: REM МАЛАЯ СУММА СОМНИТЕЛЬНА
      =====
5730 IF TOTAL<12 THEN GO TO 5770
5740 FOR I=1 TO 3
5750 IF R(I)>3 THEN GO SUB 6210
5760 NEXT I: RETURN
5770 FOR I=1 TO 2
5780 IF R(I)>2 THEN GO SUB 6210
5790 RETURN
5800 IF DISP1=1 AND DISP2=1 THEN GO TO 5860
5810 IF DISP1<>1 THEN GO TO 5980 : REM НЕТ ПАР =====
5820 FOR I=1 TO 5
5830 IF R(I)<>E1 AND P(E1)=0 THEN GO SUB 6210
5840 IF P(E1)>1 THEN LET M=4: IF I>1 THEN LET T(I)=I
5850 NEXT I: RETURN
5860 IF P(E1)>1 AND P(E2)>1 THEN GO TO 5910: REM ДВУХ ПАР НЕТ ==
5870 IF P(E1)=0 AND P(E2)=0 THEN GO TO 5940: REM ПРИНИМАЕТСЯ ДВЕ ПАРЫ И ПЕРЕБРАСЫВАЕТСЯ
      НЕПАРНАЯ КОСТЬ =====
5879 REM 2 ПАРЫ, ПЕРЕБРАСЫВАЕТСЯ 1 ПАРА И НЕПАРНАЯ КОСТЬ =====
5880 FOR I=1 TO 5
5890 IF R(I)<>E2 AND P(E2)=0 OR R(I)<>E1 AND P(E1)=0 THEN GO SUB 6210
5900 NEXT I: RETURN
5910 FOR I=1 TO 5
5920 IF R(I)=E1 OR R(I)=E2 THEN GO SUB 6210
5930 NEXT I: RETURN
5940 FOR I=1 TO 5
5950 IF R(I)<>E1 AND R(I)<>E2 THEN GO SUB 6210
5970 NEXT I: RETURN
5980 IF DISP1<>2 AND DISP2<>2 THEN GO TO 6110: REM НЕ ТРОЙКА ===
5990 IF DISP1=1 OR DISP2=1 THEN GO TO 6050
6000 FOR I=1 TO 5
6005 IF E1=0 THEN GO TO 6025
6010 IF DISP1=2 AND F(E1)>5 AND R(I)>E1 THEN GO SUB 6216
6015 IF P(E1)=3 AND R(I)=E1 THEN GO SUB 6210
6020 IF F(E1)<6 AND P(E1)<3 AND R(I)<>E1 THEN LET M=1: LET T(1) = I
6025 IF E2=0 THEN GO TO 6045
6030 IF DISP2=2 AND F(E2)>5 AND R(I)<>E2 THEN GO SUB 6210
6035 IF P(E2)=3 AND R(I)=E2 THEN GO SUB 6210
6040 IF F(E2)<6 AND P(E2)<3 AND R(I)<>E2 THEN LET M=1: LET T(1)=I
6045 NEXT I: RETURN
6049 REM ФУЛЛ =====
6050 IF E1=0 AND E2 = 0 THEN GO TO 310
6060 FOR I=1 TO 5
6070 IF P(E1)>1 AND P(E2)>1 THEN GO SUB 6210
6080 IF P(E1)>1 AND P(E2)=0 AND R(I)<>E2 THEN GO SUB 6210
6090 IF P(E1)=0 AND P(E2)>1 AND R(I)<>E1 THEN GO SUB 6210
6100 NEXT I: RETURN
6110 IF DISP1<>3 AND DISP2<>3 OR E1>6 THEN GO TO 6150: REM НЕ ЧЕТВЕРКА =====

```

```

6120 FOR I=1 TO 5
6130 IF R(I)<>E1 AND DISP1=3 OR R(I)<>E2 AND DISP2=3 THEN GO SUB 6210
6140 NEXT I: RETURN
6150 IF HIGH=3 OR LOW=3 OR DISP1=4 OR E1=7 AND P(7)=0 THEN GO TO 310: REM БЕЗ ПЕРЕБРОСКИ
      =====
6160 IF HIGH=2 AND P(8)=0 THEN LET M=1: LET T(1)=5: RETURN
6170 IF LOW=2 AND P(9)=0 THEN LET M=1: LET T(1)=1: RETURN
6180 LET M=5: FOR I=1 TO 5
6190 LET T(I)=I: NEXT I: RETURN
6210 LET M=M+1: LET T(M)=I
6220 RETURN
6300 IF NOT HIGH AND NOT LOW THEN RETURN
6310 IF F(8)<4 AND HIGH OR F(9)<4 AND LOW THEN RETURN
6320 IF DISP1=2 AND DISP2=1 AND (F(8)-HIGH)>3 AND HIGH THEN RETURN
6321 IF DISP1=2 AND DISP2=1 AND (F(9)-LOW)>3 AND LOW THEN RETURN
6322 IF DISP1=1 AND DISP2=2 AND (F(8)-HIGH)>3 AND HIGH THEN RETURN
6323 IF DISP1=1 AND DISP2=2 AND (F(9)-LOW)>3 AND LOW THEN RETURN
6330 IF DISP1>HIGH AND HIGH OR DISP1>LOW AND LOW THEN RETURN
6340 IF (F(E1)-DISP1)<4 THEN RETURN
6344 IF E2=0 THEN GO TO 6350
6345 IF (F(E2)-DISP2)<4 THEN RETURN
6350 IF F(E1)>10 AND F(8)<10 AND HIGH THEN RETURN
6354 IF E2=0 THEN GO TO 6360
6355 IF F(E2)>10 AND F(8)<10 AND HIGH THEN RETURN
6360 IF F(E1)>10 AND F(9)<10 AND LOW THEN RETURN
6364 IF E2=0 THEN GO TO 6370
6365 IF F(E2)>10 AND F(9)<10 AND LOW THEN RETURN
6370 GO SUB 4010: RETURN
8999 REM ГРАФИКА = = = = =
9000 FOR I=0 TO 17: PRINT PAPER 6; AT I,14; " ": NEXT I
9005 FOR I=0 TO 17: PRINT PAPER 6; AT I,16;CHR$(144+I): NEXT I
9010 FOR I=0 TO 17: PRINT PAPER 2;AT I,2;Z$;AT I,23;Z$: NEXT I
9020 FOR I=0 TO 17: PRINT PAPER 4;AT I,8;Z$;AT I,17;Z$: NEXT I
9040 PLOT 15,31: FOR I=1 TO 4: DRAW 216,0: DRAW 0,16: DRAW -216, 0: DRAW 0,16: NEXT I: DRAW
      216,0 : DRAW 0,16: DRAW -216,0
9050 FOR I=2 TO 14 STEP 2: PLOT I*8-1,175: DRAW 0,-143: NEXT I:FOR I=17 TO 29 STEP 2: PLOT
      I*8-1,175: DRAW 0,-143: NEXT I
9055 IF PLAY=1 THEN GO TO 9070
9060 PRINT INK 1;AT 2,0;"П";AT 3,0;"Е";AT 4,0;"Р";AT 5,0;"В";AT 6,0;"Ы";AT 7,0; "Й";AT
      8,0;" ":AT 9,0;"И";AT 10,0; "Г"; AT 11,0;"П";AT 12,0;"О"; AT 13,0; "К"
9065 PRINT INK 2; AT 2,30;"В";AT 3,30;"Т";AT 4,30;"О";AT 5,30;"Р";AT 6,30;"О";AT
      7,30;"Й";AT 8,30;" ":AT 9, 30; "Й"; AT 10,30; "Г"; AT 11,30; "Р"; AT 12,30; "О"; AT
      13,30;"К": GO TO 9080
9070 PRINT INK 1;AT 4,0;"К";AT 5,0;"О";AT 6,0;"М";AT 7,0;"П";AT 8,0;"б";AT 9,0;"Ю";AT
      10,0;"Т"; AT 11,0; "Е";AT 12,0; "Р"
9075 PRINT INK 2; AT 5,30;"Ч";AT 6,30; "Е"; AT 7,30; "Л";AT 8,30;"О";AT 9,30;"В";AT
      10,30;"Е";AT 11,30;"К"
9080 IF GO=1 THEN PRINT AT 15,0; INK 7; PAPER 0; FLASH 1;"X";AT 15,30; PAPER 7; " "
9085 IF GO=1 THEN PRINT AT 15,0; PAPER 7;" ":AT 15,30; INK 7;PAPER 0; FLASH 1;"X"
9090 RETURN
9100 REM ИНСТРУКЦИЯ =====
9110 BORDER 1: INK 6: PAPER 1: PRINT : CLS
9120 PRINT AT 3,0;
      " ЦЕЛЬ ЭТОЙ ИГРЫ СОСТОИТ В ТОМ,
      ЧТОБЫ ПРОВЕСТИ ТРИ ФИШКИ В КРАС-
      НУЮ ЗОНУ НА СВОЕЙ ПОЛОВИНЕ ПОЛЯ
      ДО ТОГО КАК ЭТО СДЕЛАЕТ ВАШ СО-
      ПЕРНИК."
9130 PRINT '
      ИГРА ВЕДЕТСЯ ДЕВЯТЬЮ ФИШКАМИ.
      СНАЧАЛА ОНИ РАСПОЛОЖЕНЫ В ЦЕНТРЕ
      ПОЛЯ НА ДОРОЖКАХ, ОБОЗНАЧЕННЫХ
      ""1"", ""2"",...""6"", ""R"", ""
      H"", ""L"".
      " ЗА ОДИН ХОД ВЫ МОЖЕТЕ ПЕРЕ-

```

ДВИГАТЬ ДО ДВУХ ФИШЕК. ВЕЛИЧИНА  
ХОДА ОПРЕДЕЛЯЕТСЯ РЕЗУЛЬТАТОМ  
БРОСАНИЯ ПЯТИ КОСТЕЙ."

9140 GO SUB 9300

9150 PRINT AT 5,0;

" ПОСЛЕ ПЕРВОГО БРОСКА ВЫ МОЖЕ-  
ТЕ ДВАЖДЫ ПЕРЕБРОСИТЬ СТОЛЬКО  
КОСТЕЙ, СКОЛЬКО ХОТИТЕ. ПРИ ВТО-  
РОМ И ТРЕТЬЕМ БРОСКАХ РАЗРЕШАЕТ-  
СЯ ИСПОЛЬЗОВАТЬ ЛЮБЫЕ КОСТИ."

9160 PRINT "

" ВАША ЦЕЛЬ ПРИ ПЕРЕБРОСКЕ КОС-  
ТЕЙ - СОСТАВИТЬ ТАКУЮ КОМБИНА-  
ЦИЮ, КОТОРАЯ ДАСТ ВАМ ВОЗМОЖ-  
НОСТЬ ЛУЧШЕГО ХОДА ФИШКАМИ."

9170 GO SUB 9300

9180 PRINT AT 1,0;

" ПРАВО НА ХОД ДАЮТ СЛЕДУЮЩИЕ  
КОМБИНАЦИИ:"

"- ДВЕ ИЛИ БОЛЬШЕ КОСТЕЙ ОДИНАКО-  
ВОГО ЗНАЧЕНИЯ 1...6 (ПЕРЕДВИ-  
ГАЮТСЯ ФИШКИ НА ДОРОЖКЕ С СО-  
ОТВЕТСТВУЮЩИМ НОМЕРОМ);"

"- ""СТРИТ"" - ПОСЛЕДОВАТЕЛЬНОСТЬ  
ИЗ ПЯТИ ИДУЩИХ ПОДРЯД ЗНАЧЕНИЙ  
КОСТЕЙ (ПЕРЕДВИГАЕТСЯ ФИШКА ПО  
ДОРОЖКЕ ""R"" - ""RUN"");"

9190 PRINT "

"- БОЛЬШАЯ СУММА КОСТЕЙ - БОЛЬШЕ  
22 (ПЕРЕДВИГАЕТСЯ ФИШКА ПО ДО-  
РОЖКЕ ""H"" - ""HIGH"");"

"- МАЛАЯ СУММА - СУММА ВСЕХ КОС-  
ТЕЙ МЕНЬШЕ 13 (ПЕРЕДВИГАЕТСЯ  
ФИШКА ПО ДОРОЖКЕ ""L"" - ""LOW"")."

9200 GO SUB 9300

9205 PRINT AT 2,0;

" КОМБИНАЦИИ ПОЗВОЛЯЮТ СДЕЛАТЬ  
ХОД НА ОПРЕДЕЛЕННУЮ ВЕЛИЧИНУ:"

9210 PRINT TAB 17;"ПАРА - 1 ШАГ" TAB 15;"ТРОЙКА - 2 ШАГА" TAB 13;"ЧЕТВЕРКА - 3 ШАГА" TAB 14;  
"ПЯТЕРКА - 4 ШАГА" TAB 2;"БОЛЬШОЙ СТРИТ (2-6) - 3 ШАГА"

9220 PRINT TAB 4;"МАЛЫЙ СТРИТ (1-5) - 2 ШАГА" TAB 2;"БОЛЬШАЯ СУММА (>25) - 3 ШАГА" TAB 4;"БОЛЬШАЯ  
СУММА (23-25) - 2 ШАГА" TAB 4;"МАЛАЯ СУММА (<10) - 3 ШАГА" TAB 2;"МАЛАЯ СУММА (10-12)  
- 2 ШАГА"

" ЗАМЕЬТЕ, ЧТО ЕСЛИ У ВАС ЕСТЬ  
ДВЕ ПАРЫ, ИЛИ ФУЛЛ (ПАРА+ТРОЙКА)  
ТО ВЫ ИМЕЕТЕ ПРАВО ПЕРЕДВИНУТЬ  
ДВЕ ФИШКИ."

9230 GO SUB 9300

9240 PRINT "" TAB 11;"ЗАМЕЧАНИЯ" ;AT 2,11; OVER 1;"\_\_\_\_\_"

"1) ВЫ НЕ ИМЕЕТЕ ПРАВА ХОДИТЬ ОД-  
НОВРЕМЕННО ПО ДОРОЖКАМ ""H"" (""L"" ) И ""1""...""6""."

"2) ФИШКА НЕ МОЖЕТ ВЫХОДИТЬ ЗА  
ВНЕШНЮЮ ГРАНИЦУ КРАСНОЙ ЗОНЫ  
- ПРОПУЩЕННЫЙ ХОД ПРОПАДАЕТ."

9245 PRINT "

"3) ИГРОК, ДЕЛАЮЩИЙ ХОД, ОТМЕЧА-  
ЕТСЯ КУРСОРОМ "; FLASH 1; INK 7; PAPER 0;"X"; FLASH 0; INK 6; PAPER 1;"."

9250 PRINT "

"4) НОМЕРА КОСТЕЙ ИЗОБРАЖАЮТСЯ  
ЧЕРНЫМ ПО ЗЕЛЕНОМУ, НАПРИМЕР:

"" TAB 16; INK 0; PAPER 4;"5-":

GO SUB 9300

9255 CLS : PRINT AT 5,0;

" В НАЧАЛЕ ИГРЫ ФИШКИ УСТАНОВ-  
ЛЕНЫ НА НЕЙТРАЛЬНЫХ ОТМЕТКАХ."



```

ОНИ ИЗОБРАЖАЮТСЯ ТАК:``TAB 15; INK 0; PAPER 6;CHR$ 132;CHR$ 136; PAPER 1 ` TAB 15;
PAPER 6;CHR$ 129;CHR$ 130
9260 PRINT `
"-----
    ЕСЛИ ВЫ ХОТИТЕ ПРОЧИТАТЬ ЭТУ
    ИНСТРУКЦИЮ ЕЩЕ РАЗ, НАЖМИТЕ <I>,
    ИНАЧЕ НАЖМИТЕ ЛЮБУЮ КЛАВИШУ...": GO SUB 9300: IF INKEY$="I" THEN GO TO 9110
9270 BORDER 1: PAPER 1: INK 6: CLS : PRINT AT 10,0;
"    ЕСЛИ ВЫ ИГРАЕТЕ С КОМПЬЮТЕРОМ,
    ВВЕДИТЕ ""1"", А ЕСЛИ С ПАРТНЕРОМ,
    ТО ВВЕДИТЕ ""2"". "
9271 INPUT "ВВЕДИТЕ 1 ИЛИ 2 : ";PLAY
9272 LET PLAY=INT PLAY: IF PLAY>2 OR PLAY<1 THEN GO TO 9271
9275 IF PLAY=2 THEN GO TO 9290
9280 CLS : PRINT AT 7,0;
"    ПОЖАЛУЙСТА ИМЕЙТЕ ВВИДУ, ЧТО
    КОГДА КОМПЬЮТЕР ПРИНИМАЕТ РЕШЕ-
    НИЕ О ТОМ, КАКУЮ КОСТЬ ЕМУ ПЕРЕ-
    БРОСИТЬ, ОН РАСПОЛАГАЕТ КОСТИ В
    ПОРЯДКЕ ИХ ЗНАЧЕНИЙ.````
9285 PRINT ``
"    ЕСЛИ ВЫ ХОТИТЕ ХОДИТЬ ПЕРВЫМ,
    ТО ВЕДИТЕ ""-1"" ИНАЧЕ ВЕДИТЕ ""1""
9286 INPUT "ВВЕДИТЕ 1 ИЛИ -1 : ";GO
9287 LET GO=INT GO: IF GO=0 OR GO>1 OR GO<-1 THEN GO TO 9286
9290 BEEP .25,3: BEEP .15,0: BEEP .15,0: BEEP .25,3: CLS : RETURN
9300 PRINT #1;
"    НАЖМИТЕ ЛЮБУЮ КЛАВИШУ...": PAUSE 0: BEEP .25,3: BEEP .15,0:CLS : RETURN
9900 POKE 23675,88: POKE 23676,255: RESTORE 9900
9901 FOR I=0 TO (18*8-1): READ A
9902 POKE USR "A"+I,A: NEXT I
9904 RETURN
9905 DATA 0,8,24,56,120,88,88,24,24,24,24,24,126,126,0: REM A&B=1
9910 DATA 0,0,60,126,70,2,6,12,12,24,24,48,48,126,126,0: REM C&D=2
9915 DATA 0,126,126,6,12,24,48,96,124,126,6,6,102,124,120,0: REM E&F=3
9920 DATA 0,6,14,30,54,102,102,102,102,127,127,6,6,6,6,0: REM G&H=4
9925 DATA 0,126,126,96,96,96,126,126,6,6,6,12,24,112,96,0: REM I&J=5
9930 DATA 0,6,12,24,24,48,48,120,108,108,102,102,54,60,24,0: REM K&L=6
9935 DATA 0,56,124,102,102,108,108,120,112,120,108,108,102,102,102,0: REM M&N=R
9940 DATA 0,102,102,102,102,102,102,126,126,102,102,102,102,102,102,0: REM O&P=H
9945 DATA 0,96,96,96,96,96,96,96,96,96,96,126,126,0: REM Q&R=L
9950 DATA 20,27,68,27,116,27,164,27,212,27: REM DICE PLOT CO-ORDS
9980 SAVE "DICEY" LINE 2: SAVE "chr" CODE 64600,768: VERIFY "DICEY": VERIFY "chr" CODE: GO
    TO 9998
9990 RANDOMIZE USR 15619: REM :ERASE "DICEY"
9992 RANDOMIZE USR 15619: REM :SAVE "DICEY" LINE 2
9994 LET ERR=USR 15619: REM : VERIFY "DICEY"
9996 IF ERR<>0 THEN PRINT FLASH 1; "ERROR!": BEEP 1,0: STOP
9998 BEEP .1,26: BEEP .1,20
9999 BORDER 7: PAPER 7: INK 0: BRIGHT 0

```

Автостарт программы происходит со строки 2, в которой загружается символьный набор.

В строке 3 происходит переключение курсора с [L] на [C] для упрощения опроса клавишей, а также переключение на загруженный русско-латинский символьный набор.

В строках 4...6 организован вывод инструкции при нажатии клавиши "Y". Инструкция выводится при помощи подпрограммы в строках с 9100. Прочитайте ее, Вам понятнее будут многие моменты при отладке программы.

После этого при помощи подпрограммы 9270 происходит задание числа играющих. Возможна игра "ЧЕЛОВЕК-КОМПЬЮТЕР" или "ИГРОК 1-ИГРОК 2". Это определяет переменная PLAY, соответственно имеющая значение 1 или 2. В том случае, если Вы играете с компьютером, можно предоставить право первого хода Вам или компьютеру. Это

определяется переменной GO в строке 9286. В том случае, если играют два партнера, первым всегда ходит игрок 1.

В строках 8...10 происходит задание UDG-графики. Это выполняет подпрограмма со строк 9100.

Со строки 15 начинается инициализация переменных, участвующих в программе. Переменные B\$ и B\$ в строке 20 определяют изображение фишек на игровом поле.

Основной цикл игры представлен в строках 70...380. Остальные строки программы (с 2000) являются различными фрагментами и подпрограммами, необходимыми для игры.

После прорисовки игрового поля в строках 70 и 80 и проверки условия победы в игре (строка 90) происходит передача хода противнику в строке 100.

В строках 200... 210 организован первый бросок. Значение костей изображается при помощи подпрограммы со строки 3500. Далее (строка 230), если бросок выполнил человек, выполняется подпрограмма со строки 3000 - это пауза для оценки результата и принятия решения о том, нужна ли переброска. Если не нужна, то программа делает скачок на строку 310. А если нужна, то следует запрос о перебрасываемых костях: сколько их и какие будут перебрасываться. Далее - возврат из подпрограммы.

В строке 231 если был ход компьютера, им выполняются действия по оценке хода и определению приоритетного варианта при помощи подпрограмм со строк 3600, 5000, 5500. Если в результате выполнения этих проверок определится, что переброска не нужна, то произойдет скачок со строки 234 на строку 310. Иначе программу продолжает группа строк 235...255 это - выполнение второго броска для тех костей, которые требуют переброски.

Строки 270...274 - это копия строк 230...234 - здесь выполняются те же действия по оценке результатов броска.

Аналогично выполняется и третий бросок при помощи строк 275...295, после чего программа продолжается со строки 310. При помощи подпрограммы со строки 3600 происходит выявление результативной комбинации костей, то есть той, которая приводит к выполнению хода игроком.

В строке 320 выявленная результативная комбинация выводится на экран. После нажатия любой клавиши (строки 330,340) происходит передача хода противоположной играющей стороне (строки 350...380), после чего программа закидывается на строку 70, то есть на повторение всех игровых действий.

Строки 9980 и 9990 служат для самозаписи программы соответственно для магнитофонной и дисковой версии. Не забудьте также о строке 2, которая в случае дисковой версии должна быть следующей:

```
2 CLEAR 64599: RANDOMIZE USR 15619: REM : LOAD "chr"CODE 64600
```

На этом мы прощаемся с Вами, желаем с пользой и интересом провести время за этой игрой, а мы пока готовим крупный сюрприз для ближайших выпусков - программный комплекс для самостоятельного создания адвентюрных игровых программ - Adventure Building System, сокращенно - ABS.

Вы увидите, что с его помощью можно разрабатывать и увлекательные обучающие программы для детей.

Идет длительный процесс по адаптации и отладке этого пакета. Мы надеемся, что к следующему выпуску все работа будет закончена и мы вынесем его для широкого применения.

Искренне Ваш, "ИНФОРКОМ".

# Компьютерная новелла

(С) Матвеев Ю. А., 1993

## Стратегия капитана Кренона.

(По мотивам программы Rebel Star)

### Часть 1. Лунная база "Дельта".

Скоростной космический лайнер "Викинг", следующий рейсом Земля-Венера, вышел на основную трассу в 0.15 по Гринвичу, когда пассажиры уже готовились ко сну. В двухместной каюте номер 29 при тусклом свете ночника возле обзорного экрана, заменявшего иллюминатор, в креслах сидели двое: седой крупный мужчина лет шестидесяти, одетый в серый строгий костюм, и светловолосый юноша в форме курсанта Высшей Космической школы. Оба молча смотрели на желто-серую горбушку удаляющейся Луны.

- Дед, а дед, - тихо прошептал юноша, не отрывая глаз от экрана. - Ты помнишь свою первую высадку на Луну?

- Конечно. Это было еще в конце прошлого века.

- Интересно было наверное, правда?

Бывший командир Группы космических десантников Элмер Кренон медленно перевел взгляд на внука:

- Интересно? Вряд ли... - и, чуть помолчав, добавил: - Скорее страшно.

Юноша внимательно посмотрел на деда и легонько тронул его за рукав:

- Ты обещал рассказать.

Кренон ничего не ответил, а только поднялся с кресла и, неспеша расстегивая пуговицы на пиджаке, тяжело вздохнул.

Ущербная Луна сместила в левую часть экрана и постепенно пряталась за его краем. Кренон снял пиджак и аккуратно повесил на спинку кресла. Потом подошел к внуку и положил тяжелые теплые ладони ему на плечи.

- Ну, пожалуйста, дед, - протянул юноша. - Я давно хотел услышать эту историю.

Кренон посмотрел на экран и тихо сказал:

- Хорошо, только потом сразу спать.

Юноша встал, потянулся, разминая мышцы, и с легкостью начал стаскивать с себя курсантскую форму. Тем временем Кренон плеснул в стакан вишневого сока из прозрачной пластиковой бутылки и залпом осушил его.

Вспоминать историю высадки спасательной бригады было тяжело. Тогда он только принял командование десантной группой патрульного корабля "Радий" и, конечно, не подозревал, что судьба уготовит ему такую жестокую проверку на зрелость. Потом многие его хвалили за удачно проведенную операцию, но он-то знал, что, будь в его распоряжении хотя бы два часа лишнего времени, все могло оказаться по-другому. Не пришлось бы жертвовать людьми. Многие из подчиненных были его друзьями. Они заканчивали одну учебку, все вместе пошли служить в одно десантное подразделение, и район им достался, казалось бы, тихий - Луна. Экваториальная зона, сектор А-3. Некоторые были даже недовольны и пытались перевестись в другие десантные группы. Да и сам, молодой тогда еще, Кренон хоть и убеждал иных остаться, понимал, что Луна - это не огнедышащий Меркурий и не коварный Марс.

Служба шла размеренно и спокойно, пока серьезно не заболел командир десантников Петр Ковальски. В срочном порядке его отправили на землю, а командиром патрульного корабля был назначен Элмер Кренон. Неприятности начались на следующий день...

Он разминался в спортивном зале, когда неожиданно по внутреннему телеселектору его вызвал радист Билли Брэг. Освободив кисти рук от ремней тренажера, Кренон поднялся и неспеша подошел к пульту.

- Ну, что там у тебя?

- Только что получили сигнал SOS. - хрипло отозвался Брэг.

- Сейчас буду. - Кренон шлепнул ладонью по выключателю и тихо выругался. Без Ковальски он чувствовал себя достаточно беспокоино и неудобно, словно цыпленок без матери курицы.

Накинув поверх спортивного костюма китель, Кренон направился в командный отсек. Билли Брэга он застал возле дисплея бортового компьютера. На экране светилась карта-сетка района бедствия. Краем глаза Кренон заметил в верхнем правом углу экрана название лунной базы, с которой пришел сигнал.

- "Дельта?" - удивленно спросил он.

Брэг молча кивнул, переключая масштаб изображения.

- Что могло случиться у селенологов? - вслух подумал Кренон.

Брэг оторвался от экрана и повернулся к командиру:

- Они не отвечают, передатчик постоянно посылает сигнал SOS и больше ничего.

Кренон почувствовал, как лоб покрывается испариной:

- Объявляй общую тревогу. Да, и еще: свяжись с "Селеной" и доложи по форме.

В то время, как Билли Брэг выполнял поручения командира, Кренон опустился в кресло перед компьютером и запросил файл данных на лунную базу "Дельта".

Изучая колонки цифр, он думал о том, что могло стать причиной бедствия на этой тихой, забытой Богом базе. Построенная во времена второй волны колонизации и с самого начала отданная селенологам, база имела мощную буровую установку, небольшую, по нынешним временам, исследовательскую лабораторию, атомный реактор, сад с фонтаном под прозрачным центральным куполом и много всякого рода мелочей, необходимых для работы и отдыха персонала. Конструкция базы и ее основные элементы были сделаны на совесть. Малейшее отклонение от нормы в работе важнейших агрегатов четко отслеживалось центральным компьютером. Автоматизированная система контроля во главе с огромным вычислительным комплексом "Исаак", заслужившая самые высокие оценки на всех базах подобного типа, на этот раз, видимо, не сумела предотвратить бедствие. Вероятно, все произошло внезапно и последствия оказались необратимыми.

Командный отсек быстро заполнили созванные по тревоге десантники. Весь экипаж патрульного корабля "Радий" собрался у центрального пульта управления, чтобы послушать указания орбитального комплекса "Селена". Выслушав сообщение Билли Брэга, оператор "Селены" некоторое время не выходил на связь.

- Как всегда, - заметил Джаспер Прюн, - Теперь дебаты устроят...

- Симпозиум соберут, - согласился с ними Леон Троцкий, который еще в школе космодесантников получил прозвище "Революционер" из-за своей известной фамилии.

Однако ответ с "Селены" пришел быстро. Уверенный голос оператора внушал оптимизм:

- Это Ваш район, так что действуйте по инструкции.

К каналу связи подключился Кренон:

- Специальных указаний нет?

- Нет, - ответил оператор.

- Понял, - задумчиво отозвался командир "Радия". - Отбой.

Он медленно встал с кресла. Все в ожидании повернулись к нему.

- Готовность номер один и по полной форме, - выдохнул Кренон.

Десантники удивленно переглянулись. Все слышали, что "Селена" не дала специальных указаний. Следовательно, можно было бы не брать с собой боевых роботов, полный боекомплект вооружений и не надевать военные скафандры. В таких случаях обычно ограничивались медицинской аппаратурой и легкими скафандрами.

- Береженого Бог бережет, так что ли? - нарушил молчание Леон Троцкий.

- Особенно революционеров, - иронично отозвался Джим Дигриз.

- Выполняйте приказ, - отрезал Кренон.

Десантники стали молча расходиться. Курт Левин, - заместитель командира патрульного корабля, пилот первого класса, - расположился за пультом управления. Только сейчас Кренон посмотрел на главный экран. Лунное Море Дождей без единой капельки воды медленно выплывало навстречу, открывая бесконечные пространства застывшей миллионы лет назад магмы. До лунной базы "Дельта" оставалось около двухсот километров.

- Включаю орбитальные двигатели, - доложил Курт Левин.

Патрульный корабль лег на правый борт и медленно стал приближаться к поверхности Луны, удерживая курс на ""Дельту".

Корабль они посадили недалеко от базы селенологов. Прозрачный центральный купол переливался в ослепительных лучах не по-земному яркого солнца. Служебные постройки, примыкавшие к куполу, выглядели обыкновенно. Никаких следов разрушений замечено не было. На связь по-прежнему никто не выходил.

Кренон построил отряд в шлюзовом отсеке "Радия". Двенадцать вооруженных десантников в скафандрах со снятыми гермошлемами в ожидании стояли напротив него.

- Разбиваемся на четыре группы, - начал Кренон. - Первая группа - Нельсон Смит и Джаспер Прюн. Ваша задача осмотреть цех переработки. Обо всех обнаруженных неисправностях сразу докладывать.

Механик Джаспер Прюн утвердительно кивнул, но тут же, сняв с плеча лазерное ружье, продемонстрировал его на всеобщее обозрение:

- А это тогда для чего? Гвозди забивать?

Все засмеялись, лишь один Кренон с каменным лицом молча посмотрел на Прюна.

- Вторая группа, - продолжил Кренон, - Вилли Брэг, Леон Троцкий и Илэйн Фрейх. Вы осмотрите центральный вход, ангар и прилегающие помещения. Третья группа - Дон Кейлони, Сларти Бартфаст и Рита Рампол. Ваша задача проверить жилые отсеки. Пойдете через служебный вход. Старший группы Дон Кейлони.

Непалец Рита Рампол хотел что-то сказать, но Кренон жестом остановил его.

- Четвертая группа - Курт Левин, Джим Дигриз и курсанты Джил и Джой. Проверьте запасной выход и присоединяйтесь ко второй группе. Старший - Курт Левин. Задача ясна? - спросил Кренон, придирчиво осматривая каждого десантника.

Вопросов ни у кого не было. Кренон посмотрел на Нельсона Смита:

- Боевые дроиды готовы?

- Да.

- Приступайте к разгрузке. Каждой группе - по дроиду.

Смит подошел к автоматической двери, ведущей в дроидный отсек и нажал на панели электронного замка несколько кнопок. С мягким гудением, дверь отъехала в сторону, освободив проход. Из глубины помещения бесшумно вышли один за другим четыре боевых дроида. Десанникам пришлось потесниться: боевые машины цилиндрической формы на массивных стальных ногах-шарнирах заняли почти все свободное пространство. Кренон ободряюще кивнул на вопросительный взгляд Нельсона Смита и махнул рукой.

- Выходим, - сказал он и закинул на плечо фотонный излучатель.

Застегивая колпак гермошлема, Кренон прокрутил в голове все параграфы инструкций для спасательных отрядов и остался доволен.

Они оставили корабль на дне небольшого древнего кратера и, разделившись на четыре группы, быстро добрались до базы.

Первые опасения появились, когда они остановились напротив наглухо запертых шлюзовых ворот. Автоматика полностью заблокировала вход и открыть двери не было никакой возможности. Командиры групп доложили по общей связи о том, что и у них ничего не получается. База молчала, сигнал SOS больше не поступал. В эфире воцарилась тишина.

- Похоже, нас здесь совсем не ждут, - прозвучал в шлемофонах голос Курта Левина. - Командир, придется применить дронов.

- Возможно, - задумчиво отозвался Кренон, - но лучше не тратить их заряды. У нас

есть светосабли, давайте их и попробуем.

Светосабля взрывного контактного действия позволяла прорубать проходы в любых завалах и поэтому давно была на вооружении спасательных отрядов. Практически любой, даже самый прочный, металл сдавался под натиском мощных электро-лазерных разрядов.

- А что нам прикажете делать? - прозвучал недовольный голос Джаспера Прюна. - У нас нет с собой ни одной светосабли.

- Я иду к Вам, - ответил Кренон. Он посмотрел на отливающий солнцем стеклянный колпак гермошлем Леона Троцкого. - За старшего - Троцкий.

- Есть, - откликнулся Леон. - Билли, где будем рубить?

- Революция, не засоряй эфир, - пошутил кто-то из десантников.

Кренон легко преодолел несколько метров до входа в цех переработки, где его ждали Нельсон Смит и Джаспер Прюн.

- А что, если у них отключена герметизационная защита - увидев приближающегося Кренона спросил Смит. Нас же сдует, как пылинки!

Кренон ничего не ответил. Он вплотную приблизился к шлюзу и, отметив середину двери, достал из ножен светосаблю. В этот момент на связь вышел Дон Кейлони, старший группы, которая шла через служебный вход:

- Командир, у нас ничего не выходит. Прошу разрешить использовать дроида.

- Разрешаю.

В шлемофоне что-то затрещало, затем радался звук, напоминающий взрыв хлопушки, а следом прогремел бас Курта Левина:

- У меня получилось, я в тамбуре. Гермозащита работает. Можно смело входить.

Кренону хватило четырех ударов, чтобы прорубить внешнюю шлюзовую дверь, ведущую в цех переработки. Он вошел в довольно просторное помещение шлюзового тамбура. Сработала система герметизации и прорубленный вход закрылся прозрачной световой пленкой, временно защищающей базу от вакуума. Следом вошли Прюн и Смит, затем медленно вошел дROID. Кренон остановился перед внутренней шлюзовой дверью, чтобы отдышаться. Надо сказать, что работать светосаблей было нелегко и приходилось прилагать большие усилия для хорошего контакта сабли с металлом.

- Старшие групп, - сказал Кренон, - доложите обстановку.

Первым откликнулся Леон:

- Вошли в предбанник.

- Аналогично, - отозвался Курт Левин.

Дон Кейлони после недолгого молчания поведал всем о том, что его дROID разрезал вторую перегородку и они вошли в гардеробную.

- Что-нибудь наблюдаете? поинтересовался Кренон.

- Ничего подозрительного, ответил Кейлони, - хотя... Черт!!!

Послышался треск и на время все стихло. Потом все услышали дрожащий голос Рампола, перекрываемый шумом:

- Внимание! Нападение на группу... - Опять треск и шум.

- Что там у вас?

- Дон, ответь!

- Говорит Дон Кейлони. Нас обстреляли какие-то дроиды. Мы отступаем.

- Говорит Кренон. Сколько их?

- Не знаю, много. Очень много!

- Дон, без паники... - Кренон, замахнувшись саблей, рубанул твердый металл перегородки. Ничего не получилось. Перегородка устояла.

- Говорит Леон. Мы вскрыли центральный вход, я вижу с десяток дроидов, которые движутся к служебному входу. Меня пока не замечают. Впрочем, подождите...

Они открыли о... - речь Троцкого оборвалась на полуслове. Эфир наполнился непрекращающимся треском.

- Всем назад! - кричал в микрофон Кренон, нанося светосаблей сильнейшие удары упрямой перегородке. Наконец, она подалась.

В образовавшемся проходе он увидел высокого дроида с круглой головой и

остроносой лазерной пушкой, который очень быстро двигался в сторону служебного входа. Затаив дыхание, Кренон смотрел на удалявшегося робота.

- Ого! - выглядывая из-за плеча командира, воскликнул Нельсон Смит, - откуда они здесь?

Кренон отступил назад:

- Дон, отзовись!

- На связи, - тяжело дыша сказал Кейлони.

- Я надеюсь, все обошлось?

- Сларт ранен, а у нас нет даже медицинского зонда.

- Дон, они все движутся к вам.

- Мы отошли, - успокоил всех Кейлони. - Прячемся за своим железным болваном.

Командир, какие будут указания?

Кренон на секунду замешкался. Такого поворота событий он, конечно, не ожидал. Хорошо, что внутреннее чутье его не подвело и он полностью вооружил отряд. Но кто мог знать, что безобидная захолустная база селенологов буквально кишит военными дроидами. Конечно, на базах подобного типа всегда есть пять или шесть дроидов. Это, как правило, механические роботы, которые занимаются погрузкой и разгрузкой кораблей, а также добычей полезных ископаемых. Что здесь делают военные машины? Сомнительно, что их привезли с Земли. Зачем? Да и контрольную проверку ни один корабль с подобным грузом не смог бы пройти, а если бы это произошло, то на всех постах стало бы известно о существовании такой странной базы селенологов. С "Селены" никаких специальных указаний не было. Ясно, что там ничего не знали. Правда, военные дроиды это не их профиль. Вопросы нужно задавать Комитету Оборона. Тогда, какого дьявола нас не предупредили? Остается одно: дроидов собирали на этой базе. Причем, быстро и в тайне от всех. Видимо, выполняли заказ какой-нибудь могущественной военной корпорации.

И вдруг он почувствовал, что знает, где лежит ключ к разгадке тайны лунной базы "Дельта".

- Смит, - позвал Кренон. - А как управляются эти дроиды?

Из тени развороченной шлюзовой двери появился роботехник Нельсон Смит:

- Обычно по радиосвязи. Но, в принципе, программа может быть занесена и в оперативную память.

Смита слышали все, но слово снова взял Кренон:

- Дело в том, что этих дроидов собирали здесь. Для каких целей - не важно. Но я хочу сказать, что выполняя спецзаказ какой-то корпорации, селенологи вряд ли были посвящены в детали и не знали, как и где дроиды будут применяться, так? Поэтому готовый продукт, скорее всего, был без рабочей программы.

- Возможно, - протянул Смит.

- Кто может управлять роботами с базы?

- Что? - Нельсон вздрогнул и посмотрел сквозь стекло гермовлема на Кренона. - Хочешь сказать, сами селенологи?

- Допустим...

- А кто тогда посылал сигнал бедствия?

- Вот именно, - выдохнул Кренон, - я думаю, этих селенологов дроиды и убрали.

- Выходит, бунт?

- Да.

- Кто же во главе?

- Тебе видней... Мне, например, кажется, что это "Исаак" - компьютер. Больше никому.

У Нельсона округлились глаза:

- Похоже, очень похоже на правду. В комплексе "Исаак" есть передающие и принимающие устройства. Ему вполне под силу управлять сотней таких дроидов.

- Хорошо, - Кренон снял с плеча фотонный излучатель. - Эту догадку мы можем проверить?

- Конечно, нужно только настроиться на их частоту.

- Займись этим, Смит. Билли Брэг, ты меня слышишь?

- Я на связи.

- Что с Леоном?

- Легкое ранение, Илэйч им сейчас занимается.

- Хорошо. Дон Кейлони!

- Слышу тебя.

- Пока ничего не предпринимайте. Ждите указаний.

- Мы за границей шлюза. Нас не трогают.

- Обращение к старшим групп! До моего особого указания всем находиться за пределами базы.

- Я поймал сигнал "Исаака"! закричал Смит, работавший с личным передатчиком. - Все сигналы закодированы, ничего понять невозможно. Но это точно он!

Кренон доверял Нельсону Смиту как самому себе.

- Судя по карте, - сказал он, - через цех переработки по тоннелю мы сможем попасть в операторскую, а там и компьютерный центр недалеко.

- Верно, - подтвердил Смит.

- Так он вас туда и пустит, вмешался Джаспер Прюн, поглаживая блестящую поверхность военного дроида. - У него там наверняка охрана.

- Мы их отвлечем, - уверенно сказал Кренон. - Ты видел, как они все помчались к служебному входу?

Джаспер Прюн пожал плечами. Кренон снова вышел на связь:

- Дон, ты меня слышишь?

- Я все понял. Берем огонь на себя, - вздохнул он.

- Дон, прошу, держитесь как можно дольше. Успеха! - заключил Кренон, а для остальных добавил: - группы Курта Левина и Леона Троцкого ждите указаний. Я с Прюном и Смитом попробую пробраться к "Исааку".

- Мы выдвигаемся, - доложил Дон Кейлони.

Кренон выглянул в проем и увидел, как еще несколько дроидов, словно по приказу, направились к служебному входу.

- Какое у них может быть оружие? - шепотом поинтересовался Кренон у Смита.

- Какое угодно: от слабеньких зикеров до мощных лазерных установок.

В эфире опять что-то затрещало и послышались легкие хлопки. Стало ясно, что у служебного входа началась схватка. Кренон внимательно осмотрел цех переработки, который предстояло пересечь, чтобы войти в тоннель. По центру цеха располагался огромный конвейер, который непрерывно работал. Дальше он видел мощную буровую установку, а за ней вход в заветный тоннель, ведущий к цели. Вдоль конвейера располагались процессоры, которые перерабатывали поступающий с конвейерной линии грунт. Повсюду были разбросаны груды щебня и остатки обработанных ископаемых.

Кренон махнул рукой и, пробежав вперед несколько шагов, остановился под прикрытием конвейера. Следом примчались Прюн и Смит. Военный дROID немного поотстал, зато надежно защищал их с тыла. Шум в эфире прекратился и десантники отчетливо слышали чей-то стон.

- Докладывает Слартти Бартфаст. Первая атака отбита. Кейлони ранен. Рита Рампол убит.

- Группы Троцкого и Левина вперед! - срывающимся голосом командовал Кренон.

- Осторожней, командир!

Кто это сказал, Кренон так и не понял: ослепительный яркий, чуть голубоватого цвета луч пронесся слева и ударился в процессор, рассыпавшись фонтаном искр. Стреляли из тоннеля. Конечно, не все дроиды противника ушли к служебному входу. Прюн был прав. "Исаак" обеспечил себе надежную защиту.

Совсем рядом прошипел еще один луч. Кренон рванулся вперед и, преодолев несколько метров свободного пространства, спрятался за грудой щебня.

В эфире все трещало, стреляло и взрывалось. Были слышны чьи-то крики и стоны. Кренон заметил, что справа от него в коридоре, ведущем к нейтральному входу, где



находилась группа Леона Троцкого, скопилось с десятков неприятельских дроидов. Часть роботов отделилась от остальных и двигалась теперь к буровой установке

- По-моему, это к нам, - заметил Джаспер Прюн, который взобрался на ленту конвейерной линии, чтобы оценить обстановку.

Кренон не видел, что произошло у конвейера - ему мешала гора щебня, за которой он прятался, но он отчетливо услышал крик Смита:

- Ну, иди сюда, гад!

Яркие вспышки озарили весь цех. Металл и камни плавились под натиском смертоносных лучей. Этот кошмар продолжался несколько секунд, но Кренону показалось, что прошла целая вечность. Когда все стихло, он увидел андроида двухметрового роста с лазерным пистолетом в железной руке. "Здесь еще и андроиды есть", - только и успел подумать Кренон, когда луч лазера расплавил добрую половину груды щебня в полуметре от него. Он несколько раз выстрелил в сторону андроида, но, похоже, промахнулся. Необходимо было сменить укрытие, иначе в следующий раз, вместо этой бурой корки оплавленного камня, будет лежать он сам. Кренон бросился к тоннелю, который казался таким близким и ему было уже все равно, что совсем недавно оттуда стреляли. В глазах черным пятном стояла вспышка луча, бившего в щебень. Кто-то уже бежал за ним. Кренон обернулся. Это был Джаспер Прюн. В эфирной многоголосице было невозможно разобрать, что кричал ему механик. Из глубины тоннеля под ноги ударил луч, но на этот раз Кренон успел выстрелить ответной очередью. Где-то в глубине коридора вспыхнул голубым пламенем патрульный дROID и с грохотом разлетелся на куски.

В мимолетном затишье Кренон вдруг разобрал слова Прюна:

- Смита убили. Но он положил трех дроидов.

Комок подкатил к горлу. Пытаясь справиться с неприятным ощущением, Кренон прокричал в микрофон:

- Все группы к служебному тоннелю!

Отозвался, почему-то, только один Леон Троцкий:

- Я у входа в цех переработки.

- А остальные?

- Билли Крэг убит, а Илэйн Фрейх тяжело ранен и уже не может идти. Он отстреливается у входа в ангар.

- Где Курт Левин? - спросил Кренон.

- Я его не видел, - ответил Леон.

Сквозь шум и свист в шлемофоне отчетливо прозвучал голос Курта:

- Я не могу пересечь ангар. Веду круговую оборону.

- Где твои люди?

- Курсанты у соседнего лунохода, а Джим Дигриз с военным дроидом направился к вам.

- Посмотри, там где-то Илэйн Фрейх. Он ранен. Помоги ему, - подключился к разговору Леон Троцкий.

- Ладно. А, черт! Ах ты, электронный выродок!...

Курт Левин больше не отвечал. Кренон и Прюн уже шли по тоннелю. Сзади, принимая на себя основной огонь, двигался военный дROID.

Сквозь перекрестный огонь в тоннель проник Леон Троцкий, неизвестно как сумевший преодолеть этот страшный путь.

Кренон понимал, что основной удар приняли на себя группы Курта Левина и Дона Кейлони и им удалось оттянуть часть дроидов.

Дверь операторской была заперта изнутри. Кренон решил использовать один из четырех мощнейших зарядов своего боевого робота. Вспышка, фейерверк огня и двери как не бывало.

ДROID въехал в образовавшийся проход. Следом вошел Кренон, а Джаспер Прюн не успел. С противоположной стороны раздался выстрел. Шипящий луч ударил в грудь Прюну и тот успел лишь вскинуть лазерное ружье, когда прозвучал еще один выстрел. Джаспер Прюн упал навзничь с прожженной насквозь грудью.

Кренон едва успел отскочить в сторону, как из бокового коридора, ведущего в компьютерный центр, ударил встречный луч. Заряд лазера приняла на себя броня боевого дроида десантников. Узкий коридор наполнился дымом.

В этой сумасшедшей свалке трудно было разобраться. Их зажали со всех сторон. Сразу три огромных дроида вели непрерывный огонь и Кренону пришлось прятаться в операторской. Улучшив момент, он попытался связаться с остальными, но ему никто не ответил. Лишь один раз, откуда-то, словно с далекой Земли, до него донесся слабый голос Дона Кейлони, которого он уже давно мысленно похоронил:

- Командир, мертвые не простят, если мы не победим.

Послышалось ему это или нет, но он почувствовал второе дыхание. Потратив всю обойму на дроида, охранявшего вход в компьютерный центр, Кренон ворвался внутрь. Посередине зала, мигая лампочками и переливаясь всеми цветами радуги, мирно гудел компьютер "Исаак". Кто бы мог подумать, что эта машина сумеет принести столько бед. Кренон перезарядил фотонный излучатель и начал методично расстреливать управляющие консоли компьютера. С противоположного конца зала к нему приближался дроид с лазерной головой. Кренону показалось, что "Исаак" буквально раскалился, предчувствуя свой близкий конец. И теперь командир десантников понимал, что этот единственный дроид последняя надежда компьютера. Он представил себе, как электронная душа "Исаака" переселяется в военного дроида. И вот уже компьютер - не беспомощная железяка, набитая микросхемами, а грозный и опасный противник. Да вот же он: совсем рядом!

"О чем это я?" - одернул себя Кренон, уворачиваясь от лазерного луча, - по-моему, начинаю сходить с ума". Он с удивлением отметил, что вот уже несколько минут его не покидает желание прекратить стрельбу, да и вообще бросить фотонный излучатель на пол. Ему стало страшно. Дроид был совсем близко. Но прятаться за "Исааком" Кренону почему-то совсем не хотелось. Он хорошо видел, как шарообразная голова дроида поворачивается в его сторону, лазерная пушка смотрела прямо в лицо.

- Не-е-т! - закричал Кренон и в последнее мгновение успел отпрыгнуть в сторону.

Дроид дернулся следом и выстрелил. Лазерный луч, шипя, стал вгрызаться в непрочную обшивку "Исаака". Кренон видел, как плавились электронные мозги агрессора и как тут же застыл без движений вражеский дроид. Это была победа! Раб сам убил своего хозяина. Мгновение спустя в зал ворвался Леон Троцкий:

- Неужели конец?

- Конец, - выдохнул Кренон.

В гермошлемах затрещало и на связь вышел Курт Левин:

- Леон, ты еще спрашиваешь? Иди и посмотри, сколько вокруг меня собралось застывших истуканов.

У Кренона не было сил даже улыбнуться.

\* \* \*

Когда Элмер Кренон закончил свой рассказ, его внук еще долго лежа в постели смотрел в потолок и молчал. Потом он спросил:

- Дед, а сколько человек вас осталось тогда в живых?

- Пять человек: я, Леон Троцкий, Курт Левин, Джим Дигриз и Дон Кейлони.

- А вы узнали, почему компьютер "Исаак" взбунтовался?

- Когда вскрыли черный ящик, то выяснилось, что переполненный военной информацией, компьютер просто не выдержал экспериментов этих, так называемых, "селенологов". Почему это произошло, мы так и не узнали, потому что дроид очень сильно повредил даже черный ящик. Мы можем лишь догадываться, какая каша заварилась в недрах базы перед нашим прибытием, только с той поры компьютеры "Исаак" были сняты с производства.

- А вы еще летали вместе?

- Со времен "Дельты" мы стали неразлучными друзьями и побывали в таких передрягах, что события на лунной базе покажутся детской забавой.

- Это когда? - У внука загорелись глаза.

- Ну-у, - протянул Кренон, - например, в системе Альфа Центавра, когда мы наткнулись на поселение совершенно чуждых нам существ.

- Дед, расскажи!

- В следующий раз, - усмехнулся Кренон. - Давай спать...

# АВТОРСКАЯ ПРОГРАММА

Развивая инициативу, объявленную в прошлом выпуске "РЕВЮ", мы начинаем публикацию заявок на авторское право по программным и аппаратным разработкам для домашних персональных компьютеров системы "Синклер-Спектрум".

"ИНФОРКОМ" рассчитывает, что кроме чисто "заявочного" характера этот раздел будет служить еще и справочным для тех, кто хочет и может сам что-то создать. Здесь Вы можете увидеть над чем работают Ваши коллеги, может быть найдете полезные идеи и для себя.

Мы надеемся, что публикации этого раздела явятся своеобразным генератором новых идей.

Раздел открыт для Вас, уважаемые авторы.

## 1. СЕРВИСНЫЕ ПРОГРАММЫ (УТИЛИТЫ).

ЗАЯВКА: У001.

ПРОГРАММА: INSTALL (инсталлятор).

АВТОР: В. Кутин, г.Екатеринбург.

ДАТА РАЗРАБОТКИ: март 1993 г.

ЯЗЫК ПРОГРАММИРОВАНИЯ: Ассемблер.

НАЛИЧИЕ ЗАЩИТЫ: нет.

ДОКУМЕНТАЦИЯ: - не указано.

КРАТКОЕ ОПИСАНИЕ:

Программа позволяет создать дисассемблер под тот адрес, который нужен, при необходимости встроит в дисассемблер и монитор. Полученным дисассемблером можно пользоваться независимо от инсталлятора. Он обладает следующими особенностями:

- особая компактность;
- цветность;
- озвученность;
- многооконность (4 окна для одновременной работы с разными фрагментами);
- наличие DISPLACE-системы для мониторинга программ, написанных в области экрана или в области системных переменных;
- 16 функций монитора;
- 16 функций дисассемблера;
- имеются процедуры, написанные специально для хаккеров.

\* \* \*

ЗАЯВКА: У002.

ПРОГРАММА: TURBO-COPY (копировщик)

АВТОР: В. Кутин, г.Екатеринбург.

ДАТА РАЗРАБОТКИ: июль 1993 Г.

ЯЗЫК ПРОГРАММИРОВАНИЯ: Ассемблер.

НАЛИЧИЕ ЗАЩИТЫ: нет.

ДОКУМЕНТАЦИЯ: - не указано.

КРАТКОЕ ОПИСАНИЕ:

Внешне выглядит, как TURBO COMP, ZX-COPY, TF-COPY, но имеет ряд преимуществ:

- работа с двумя скоростями - стандартной и аналогичной ПЗУ-90.
- в отличие от TURBO-COMP имеет функцию VERIFY;
- выдает информацию о компрессии блоков в процентах;
- имеет современное оформление (стилизованный шрифт);
- работает на тех машинах, на которых TURBO COMP не работает;
- особо полезна для машин с ПЗУ-90.

\* \* \*

ЗАЯВКА: У003.

ПРОГРАММА: MEGA-GEN (генератор/редактор спрайтов)

АВТОР: Д. Палтусов, г.Екатеринбург

ДАТА РАЗРАБОТКИ: апрель 1993 г.

ЯЗЫК ПРОГРАММИРОВАНИЯ: Ассемблер.

НАЛИЧИЕ ЗАЩИТЫ: нет.

ДОКУМЕНТАЦИЯ: - не указано.

КРАТКОЕ ОПИСАНИЕ:

Предназначена для создания спрайтов любого размера (по вертикали - в пикселах, по горизонтали - в знаках). Имеет развитую систему меню, удобна в работе, возможно управление джойстиком (Sinclair и Kempston), работает с группой спрайтов, результаты сводит в файл, удобный для использования в пользовательской программе ("подшивает" файл-указатель размещения спрайтов).

\* \* \*

ЗАЯВКА: У004.

ПРОГРАММА: GENERATOR (генератор/редактор спрайтов)

АВТОР: Д.Палтусов, г.Екатеринбург.

ДАТА РАЗРАБОТКИ: март 1992 г.

ЯЗЫК ПРОГРАММИРОВАНИЯ: Ассемблер, БЕЙСИК

НАЛИЧИЕ ЗАЩИТЫ: есть.

ДОКУМЕНТАЦИЯ: не указано.

КРАТКОЕ ОПИСАНИЕ:

Рассчитан на начинающих программистов, позволяет создавать спрайты 16 X 16. Готовые спрайты можно использовать как из машинного кода, так и из Бейсика ("зашиваются" в область UDG). Возможно управление джойстиком (Sinclair).

\* \* \*

ЗАЯВКА: У005.

ПАКЕТ: CONSTRUCTOR

АВТОР: Д.Палтусов, г.Екатеринбург.

ДАТА РАЗРАБОТКИ: сентябрь 1992 г.

ЯЗЫК ПРОГРАММИРОВАНИЯ: Ассемблер.

НАЛИЧИЕ ЗАЩИТЫ: нет.

ДОКУМЕНТАЦИЯ: - не указано.

КРАТКОЕ ОПИСАНИЕ:

Пакет процедур, достаточных для создания игровой программы - карточной игры.

Состав пакета:

OPENDAUM - демонстрационная карточная игра, написанная с помощью "конструктора".

CONSTRUCTOR - блок подпрограмм в машинном коде.

TXT.FIL - текстовый файл программы OPENDAUM, написанный под ассемблер "ZEUS". Позволяет изучать логику работы игры и технологию применения "конструктора".

\* \* \*

ЗАЯВКА: У006.

ПАКЕТ: MULTIEDIT UTILITY1

АВТОРЫ: Д.Палтусов, В.Кутин, г.Екатеринбург.

ДАТА РАЗРАБОТКИ: апрель 1992 г.  
ЯЗЫК ПРОГРАММИРОВАНИЯ: Ассемблер.  
НАЛИЧИЕ ЗАЩИТЫ: есть.  
ДОКУМЕНТАЦИЯ: - не указано.  
КРАТКОЕ ОПИСАНИЕ:

Аналог аниматора FANTA VISION (для IBM совместимых машин). Служит для создания мультфильмов в четыре этапа:

- определение формата кадра;
- создание фаз анимации;
- моделирование экранного изображения;
- компиляция мультфильма.

В пакет входит мощный компилятор, позволяющий сгенерировать файл, независимый от материнской программы под любой адрес загрузки и программа UTILITY1 для "вырезания" исходного спрайта из заставок, созданных в других графических редакторах.

\* \* \*

## 2. ПРИКЛАДНЫЕ ПРОГРАММЫ

Заявок нет.

## 3. ИГРОВЫЕ ПРОГРАММЫ

ЗАЯВКА: И001.  
ПРОГРАММА: SHAREHOLDER  
ЖАНР: TRADITIONAL/MANAGEMENT  
АВТОРЫ: Д.Палтусов, В.Кутин, г.Екатеринбург  
ДАТА РАЗРАБОТКИ: август 1992 г.  
ЯЗЫК ПРОГРАММИРОВАНИЯ: Ассемблер.  
НАЛИЧИЕ ЗАЩИТЫ: есть.  
КРАТКОЕ ОПИСАНИЕ:

Компьютерный аналог известной за рубежом настольной игры (освещалась в журнале "Наука и жизнь").

Количество играющих - до 4-х человек, за них может играть компьютер по одному из двух возможных алгоритмов.

Цель игры - повышение состояния за счет рациональной игры с акциями на бирже.

Управление: клавиатура, CURSOR, SINCLAIR, KEMPSTON.

Экранный интерфейс: типа "ARTSTUDIO".

\* \* \*

ЗАЯВКА: И002.  
ПРОГРАММА: BASE  
ЖАНР: PUZZLE/STRATEGY  
АВТОРЫ: Д.Палтусов, В.Кутин, Р.Мельников, г.Екатеринбург.  
ДАТА РАЗРАБОТКИ: Июнь 1992 Г.  
ЯЗЫК ПРОГРАММИРОВАНИЯ: Ассемблер.  
НАЛИЧИЕ ЗАЩИТЫ: есть.  
КРАТКОЕ ОПИСАНИЕ:

Тема игры: освобождение космической базы, захваченной пришельцами.

Логика работы: противник уничтожается, будучи запертым в безвыходном пространстве.

Экранный интерфейс - типа "BOULDER DASH" и "ROCKFORD".

Игра многоуровневая и имеет встроенный редактор уровней, позволяющий развивать ее далее.

Управление: клавиатура, CURSOR, SINCLAIR, KEMPSTON.

## Внимание, СИНКЛЕРИСТЫ!

То, о чем многие месяцы просят некоторые наши читатели, наконец свершилось! Мы объявляем о дне рождения нового журнала, посвященного играм для IBM-совместимых компьютеров, а также всему, что с ними связано.

1 января 1994 года выйдет в свет  
первый номер журнала "PC-РЕВЮ"

Папой этого издания будет "ИНФОРКОМ", а мамой - новая фирма, которую мы сейчас регистрируем - акционерное общество закрытого типа "ИНФОРКОМ - ПРЕСС".

Наших старых, добрых и верных читателей и почитателей просим не беспокоиться. День рождения "PC-РЕВЮ" не станет похоронным для "ZX-РЕВЮ", мы работали и будем работать над улучшением его содержания и не только не собираемся сворачивать это дело, но и будем развивать его дальше. Более того, мы рассчитываем на то, что и многие из Вас подключатся к созданию и распространению этого журнала.

| ЧТО ТАКОЕ "PC-РЕВЮ"?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | СОСТАВ "PC-РЕВЮ"                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>1. "PC-РЕВЮ" будет выпускаться и распространяться электронным способом. Уход от полиграфии снимет массу преград с его свободного распространения.</p> <p>2. Планируется выпускать по одному номеру ежемесячно. Объем каждого выпуска - 2,0...2,5 мегабайта. В заархивированном виде объем выпуска составит 1080 К в расчете на то, что он будет поставляться на одной дискете HD (1,2М) или на трех дискетах DD (3X360К) по выбору заказчика.</p> <p>3. "PC-РЕВЮ" будет поставляться без защиты от копирования. Мировая практика показала, что самая лучшая "защита" - это отсутствие защиты, но при оперативном обновлении информации. При таком подходе выигрывает и издатель и потребитель.</p> <p>4. Журнал готовится так, чтобы с ним можно было работать на IBM-совместимых машинах любого класса. Единственное ограничение - графические иллюстрации рассчитаны на мониторы VGA. Пользователи, не имеющие их, смогут работать с журналом, но не смогут просматривать сопровождающие картинки.</p> <p>5. У читателя будет возможность распечатать любые понравившиеся ему статьи журнала на имеющемся у него оборудовании.</p> <p>6. "PC-РЕВЮ" будет распространяться через сети дистрибуторов. Порядок распространения следующий: "ИНФОРКОМ-ПРЕСС" направляет ограниченное количество мастердисков своим дистрибуторам по регионам с лицензией на неограниченное копирование и распространение. Локальные дистрибуторы могут сами продавать выпуски "PC-РЕВЮ" на своих дискетах или копировать журнал на дискету заказчика, а могут создать свою дистрибуторскую сеть, в этом случае они станут Генеральными дистрибуторами.</p> | <p>1. Обзоры новейших игровых программ с иллюстрациями.</p> <p>2. Исследование и анализ известных и широкораспространенных игровых программ.</p> <p>3. Переписка с читателями. Каждый, кто любит игровые программы, сможет написать о своих открытиях и проблемах. Любое грамотно и интересно написанное письмо будет опубликовано для обсуждения.</p> <p>4. Обзоры зарубежной прессы: интервью с выдающимися программистами и издателями, обзоры новых аппаратных средств, расширяющих возможности стандартной машины, рассказы об известных фирмах.</p> <p>5. Раздел "Компьютерная новелла".</p> <p>6. Игровые этюды. К каждому номеру журнала мы будем прилагать отгруженные модули состояния тех или иных известных игр, представляющие собой некий этюд (по типу шахматного), решение которого доставит читателю удовольствие. Лучшие решения будут поощряться ценными призами.</p> <p>7. Рекламный блок. Этот блок не займет более 10% объема журнала и, если получит достойное развитие, позволит значительно снизить Ваши затраты на получение отдельных выпусков.</p> <p>6. Справочный блок. В него войдут:</p> <ul style="list-style-type: none"><li>- указатель наших дистрибуторов (люди во всех уголках страны должны знать, к кому им надо обращаться для приобретения очередного выпуска);</li><li>- указатель игр (для быстрого поиска, где и когда освещалась та или иная игра);</li><li>- указатель авторов (для фиксации авторских прав наших внештатных корреспондентов, статьи которых будут публиковаться в журнале);</li><li>- указатель самых низких цен по фирмам г. Москвы на радиоэлектронную продукцию на текущий месяц.</li></ul> |

Итак, для создания собственной первичной сети Генеральных дистрибуторов мы объявляем ТЕНДЕР на распространение "РС-РЕВЮ". Тендер - это как бы закрытый аукцион и он применяется в тех случаях, когда открытый аукцион провести невозможно.

1. ТЕНДЕР завершится к 15.12.93

2. Подать заявку каждый желающий может уже сейчас. Заявка подается в произвольной форме, в заявке Вы должны указать МАКСИМАЛЬНО ВОЗМОЖНУЮ ДЛЯ ВАС цену, по которой Вы готовы принимать мастердиск одного выпуска журнала для лицензированного копирования и распространения, а также форму оплаты (нал., б/нал.). При определении экономически обоснованной максимальной цены ориентируйтесь на свои возможности по распространению копий, а также на возможность организации Вами региональной дистрибуторской сети.

Обращаем Ваше внимание на то, что указание вами предельно допустимой цены НИ К ЧЕМУ ВАС НЕ ОБЯЗЫВАЕТ. В 99 случаях из 100 Вам придется платить значительно меньше, в данном случае Ваша цифра явится только свидетельством серьезности намерений и наличия потенциальных возможностей как будущего дистрибутора, что нам и необходимо знать при создании первичной сети.

3. 15.12.93 мы отберем 100 наиболее весомых заявок и установим единую цену мастердиска. Она будет равна размеру самой слабой из ста отобранных заявок.

4. Все победители ТЕНДЕРА получат извещение с указанием установленной цены и дальнейшие указания.

5. Итоги ТЕНДЕРА будут объявлены в последнем выпуске "ZX-РЕВЮ" этого года.

ПРИМЕЧАНИЕ:

1. Если на ТЕНДЕР поступит менее 100 заявок, ТЕНДЕР считается несостоявшимся и мы примем решение об изменении порядка распространения "РС-РЕВЮ".

2. Всех держателей "Зеленых пакетов", заинтересованных в распространении "РС-РЕВЮ", просим обратиться к нам по телефону или письмом. Им, как уже зарегистрированным дистрибуторам, будут предоставлены некоторые льготы.





"ИНФОРКОМ" 121019, Москва, Г-19, а/я 16

## СПЕКТРУМ в школе

Здравствуйте, дорогие друзья!

Так уж повелось в последнее время на наших страницах, предназначенных для школьников, что мы играючи осваиваем интересные численные методы решения математических уравнений, и сегодня мы хотели бы предложить Вам еще одну игру, из которой вытекают совсем не игровые последствия.

Скажите, что Вы подумаете о человеке, который предложит Вам решить ОДНО уравнение с двумя неизвестными? Наверное, что у него, как сейчас говорят, "крыша поехала". А если это будет одно уравнение с пятью неизвестными? Это невозможно, скажете Вы? А если он Вам скажет, что его не интересует абсолютно точное решение и его устроит решение с какой-то допустимой погрешностью? О, это совсем другое дело - тогда можно как-то попробовать УГАДАТЬ более-менее приличное решение, особенно если это игра.

Итак, возьмем для игры следующие значения:

$$X = 5; \quad Y = 2;$$

Составим из них уравнение, например такое:

$$X + X*Y + X*X + Y*Y - 44 = 0$$

И предложим его решить кому-либо из знакомых, не открывая истинных значений X и Y. Единственное, что можно подсказать - так это то, что X находится в пределах от 0 до 20, а Y - от 0 до 10, и пусть он теперь попробует решить такое уравнение.

Что будет делать Ваш партнер? Во-первых, он попробует взять в качестве первого приближения средние значения X и Y из их возможного интервала и посмотреть, что из этого получится:

$$X = 10; \quad Y = 5$$

$$10 + 50 + 100 + 25 - 44 = 139$$

Мы получили "невязку, равную 139, поскольку не угадали X и Y.

Надо их корректировать. Поскольку "невязка" больше нуля, значит мы где-то перебрали. Ударим опять посередине:

$$X = 5; \quad Y = 2.5.$$

Получим:

$$5 + 12.5 + 25 + 6.25 - 44 = 4.75$$

"Невязка" стала значительно меньше, значит, мы на верном пути. Попробуем еще раз уменьшить X и Y.

$$X = 2.5; \quad Y = 1.25$$

$$2.5 + 3.13 + 6.25 + 1.56 - 44 = -30.56$$

На сей раз мы ушли не в ту сторону. Попробуем еще раз и опять посередине:

$$X = 3.75; \quad Y = 1.85$$

$$3.75 + 6.98 + 14.06 + 6.98 - 44 = -12.23$$

Снова "недолет", стрельнем еще последний разок и пока хватит.

$$X=4.38; \quad Y=2.18$$

$$4.38+9.55+19.18+4.75-44 = -6.14$$

Итак, мы получили результат:

$$X=4.38, \text{ а } Y=2.18$$

Истинный же результат, как мы помним, был:

$$X=5, \text{ а } Y=2.$$

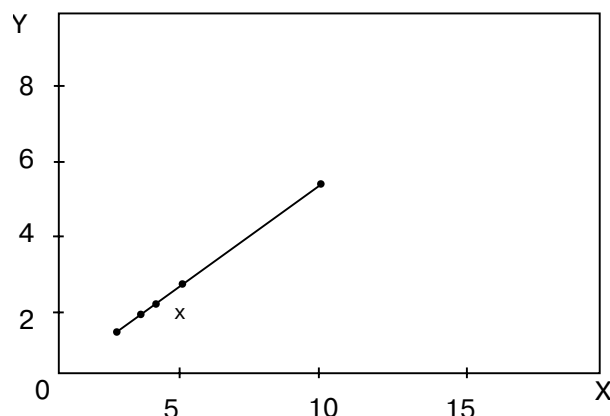
Выходит, мы решили задачу с точностью до примерно 12 процентов. Хорошо это или плохо? Ну, во-первых, это еще не предельная наша точность, ведь считали-то мы на бумажке, а если применить компьютер, то ее можно бы и повысить и как это сделать, мы еще покажем. А во-вторых, надо подумать, где такие расчеты могут использоваться?

Оказывается, они могут применяться, как правило, в двух случаях. Во-первых, в технике, когда какие-то параметры конструкции неизвестны или известны плохо. Их, конечно, можно устанавливать с помощью испытаний, экспериментов и т.п. и так и делают, но это довольно дорого, трудоемко и долго. Такой расчет позволяет во много раз снизить затраты на испытания и сэкономить месяцы и годы. К тому же, как правило, все технические устройства имеют не менее, чем 100-процентный запас надежности и точность наших расчетов вовсе не так уж и плоха.

Вторая область, когда приходится решать уравнения, в которых неизвестных больше, чем самих уравнений - экономические расчеты, типа того, как организовать транспортировку пива, нефти и подсолнечного масла между десятью регионами, спрос в которых на эти продукты точно не известен, но затратить при этом минимум средств на перевозку, хранение и обслуживание цистерн. В этих расчетах точность порядка 12 процентов - просто блестящая. В рыночной экономике надо уметь прогнозировать результаты своих действий.

Итак, мы умеем теперь решать уравнения с большим количеством неизвестных, хотя вопросов еще осталось немало. Во-первых, как приспособить к этому делу компьютер, а во-вторых, надо ему дать четкий алгоритм, потому что мы с Вами "стреляли" по собственному разумению - чуть ближе, чуть дальше, а компьютеру надо точно объяснить, куда "стрелять" в поисках нужного решения.

Давайте посмотрим на графике (рис. 1), как мы двигались к решению, начиная от  $X=10, Y=5$ .



Как видите, мы все время "бегали" по одной прямой, пока не нашли на ней точку, близко лежащую к истинному решению. Да это и не удивительно, ведь мы все время делили отрезок пополам. Но так дело не пойдет. Это хорошо, что у нас только два неизвестных и наш график имеет две координаты  $X$  и  $Y$  (его можно изобразить на даже на плоскости). А если бы неизвестных было штук пять? Вы не представляете, что такое пятимерное пространство? Мы тоже не представляем, и как попасть прямой линией в таком пространстве в нужную точку? Это действительно трудно представить.

Для того, чтобы "освободить" себя от блуждания по одной прямой и выйти за ее пределы (например так, как показано на Рис. 2), надо "отпустить"  $X$  и  $Y$  от необходимости делить отрезок пополам при поиске нового приближения и разрешить компьютеру самому

выбирать, какое приращение на  $X$  и  $Y$  он будет делать на каждом шагу. Можно даже разрешить ему это делать случайным образом. Пусть бегают где хочет, все равно мы примем только те его шаги, которые ведут к уменьшению "невязки", а все прочие отбросим.

Так наш компьютер "выйдет" за пределы роковой прямой. Если у нас пять неизвестных, то он начнет "блуждать" в пятимерном пространстве, медленно но верно подходя к приемлемому (хоть и не точному) решению.

Такой метод расчета известен в математике под названием "случайного" поиска.

Наметим алгоритм:

1. Ввод уравнения.
2. Ввод предельных значения неизвестных.
3. Выбираем первую точку в качестве средней между предельными значениями неизвестных.
4. Расчет "невязки".
5. С помощью генератора случайных чисел определяем "приращение" для каждой из координат.
6. Получаем новую точку.
7. Проверяем, не выходит ли она за пределы допустимых значений по координатам. Если да, то возвращаемся на п. 5.
8. Рассчитываем "невязку" и сравниваем ее с предыдущей "невязкой". Если она по абсолютной величине больше, чем предыдущая, то такой шаг нам не нужен, возвращаемся на п. 5.
9. Делаем новый шаг в том же "направлении", что и предыдущий случайный шаг. Теперь нам генератор случайных чисел не нужен, не стоит тратить время на случайный поиск направления, раз мы его уже нашли.
10. Проверяем не вышли ли мы за пределы допустимой области. Если да, то возврат на п. 5 (снова включаем случайный поиск).
11. Проверяем значение "невязки". Если она возросла, то шаг отменяется и возвращаемся на п. 5 для поиска нового направления, а если уменьшилась, то возвращаемся на п. 9 и шагаем в том же направлении еще раз.

Можно сделать кое-что еще. Во-первых, мы имеем здесь бесконечный цикл, поэтому надо предусмотреть окончание расчетов и выход из него по достижении какого-то значения "невязки".

Во-вторых, можно ускорить работу, если после удачного шага делать следующий шаг в том же направлении например в два раза длиннее. А после неудачного шага не бежать сразу к случайному определению нового направления, а попробовать шагнуть еще раз туда же, но в десять раз короче.

Можно повысить надежность наших расчетов, запуская одну и ту же программу несколько раз. Поскольку у нас неизвестных больше, чем количество уравнений, то может быть получено много разных пар  $X$  и  $Y$ , которые удовлетворяют решению задачи. Определив случайным образом несколько таких пар, можно подумать и выбрать среди них оптимальную.

Можно предложить еще множество разных усовершенствований и ухищрений и каждый, кому понадобится использовать метод случайного поиска для решения своей задачи сможет доработать его так, как ему надо.

От себя же добавим, что у метода случайного поиска, хоть он и работает довольно медленно, есть одна интересная особенность. Дело в том, что при применении других численных методов расчет хоть происходит и быстрее, но, встречаются ситуации, когда он заходит в тупик. Математики говорят, что расчёт "попал в яму" или "в овраг", и обычными способами из такого оврага не выбраться. А с помощью случайного поиска - пожалуйста. Поэтому применяют иные, быстроработающие методы, но когда оказывается, что "невязка" начинает уменьшаться слишком медленно, то подключают случайный поиск и он выводит из тупика.

Представьте себе, что на неровную холмистую лужайку бросили мяч. Ваша задача -

закатить его в самую низкую точку этой лужайки (минимальная невязка). Если он попал на холм, то он бодро покатится туда, куда надо (работает быстрый численный метод), но если по пути он закатится в какую-либо ямку, рытвину или овраг - дело плохо, сам он оттуда не выкатится. Его надо "встряхнуть", еще раз подкинуть, чтобы он мог катиться дальше - это и сделает вовремя подключенная процедура случайного поиска.

```

10 REM *** Метод случайного поиска***
20 DEF FN A(X,Y) = X+X*Y+X*X+Y*Y - 44: REM Ваше уравнение.
30 LET xmax = 20: LET xmin=0: LET ymax = 10: LET ymin=0: REM ввод граничных условий.
40 LET x=(xmax+xmin)/2: LET y=(ymax+ymin)/2: REM исходная точка поиска
45 GO SUB 400:REM расчет невязки.
48 LET dif1 = dif: REM запомнили невязку.
49 GO SUB 300: REM печать.
50 LET xstep = (xmax-xmin)/10: LET ystep = (ymax-ymin)/10: REM стандартный шаг поиска.
60 GO SUB 500: REM определение случайного направления.
70 LET x1=x+dx: LET y1=y+dy: REM Поиск новой точки.
80 IF x1 > xmax OR x1 < xmin OR y1 > ymax OR y1 < ymin THEN GO TO 60: REM При выходе за
    границы зоны надо повторить поиск новой точки.
90 LET x=x1: LET y=y1: REM переход к новой точке.
100 GO SUB 400: REM расчет новой невязки.
110 IF ABS(dif) < ABS(dif1) THEN GO TO 130
120 LET x=x-dx: LET y=y-dy: GO TO 60: REM если невязка оказалась больше, чем была, то эта
    точка нам не нужна.
130 LET dif1 = dif: REM запомнили новую невязку.
140 GO SUB 300: REM печать
150 IF ABS(dif1) < 1.0 THEN STOP: REM если достигнута заданная точность, то конец работы.
160 GO TO 70: REM Возврат для следующего шага.
300 PRINT "X= ";x; TAB 10;"Y= ";y; TAB 20;"DIFF= ";dif1
310 RETURN
400 LET dif = FN A(x,y)
410 RETURN
500 LET dx=xstep*(2*RND-1):LET dy=ystep*(2*RND-1)
510 RETURN

```

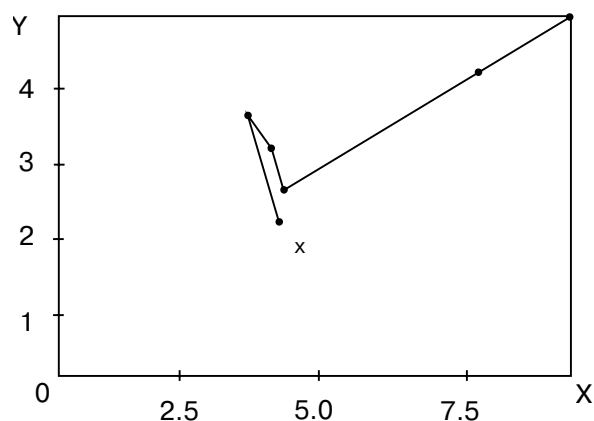
Примечание. Здесь мы используем не функцию RND, а  $2 \cdot \text{RND} - 1$ . Это сделано потому, что RND дает результат в диапазоне  $0 \dots 1$ , а нам нужно в диапазоне  $-1 \dots +1$ , иначе наш поиск будет не случайным, а все время будет уводить нас вправо и вверх.

В строке 50 мы задали параметры xstep и ystep - это как бы сила того толчка, с которой застрявший мяч выбрасывается из ямки. Мы взяли эту величину, равной одной десятой размеров "игрового поля". Если взять мало, то силы может и не хватить и мячик не найдет более удачного продолжения своему пути, а если взять слишком много, то он будет слишком долго метаться по игровому полю, а то и вообще выходить за его пределы. Можно подобрать эту величину экспериментально, а можно и сделать ее плавающей, в зависимости от того, как близко к финалу мы подошли.

Проведя расчеты по приведенной выше программе мы получили следующую траекторию поиска.

| N шага | X   | Y   | Невязка |
|--------|-----|-----|---------|
| 1      | 10  | 5   | 141     |
| 2      | 6   | 4.2 | 78      |
| 3      | 4   | 2.5 | -7.5    |
| 4      | 3.8 | 3.1 | -4.9    |
| 5      | 3.5 | 3.7 | -1.7    |
| 6      | 4.3 | 2.8 | -1.2    |
| 7      | 5.1 | 2.0 | 0.66    |

График поиска показан на рис. 2.



Результат очень близок к задуманному. Но для проверки его точности мы еще пять раз повторили поиск и получили еще пять возможных ответов.

| X   | Y   |
|-----|-----|
| 1.3 | 5.8 |
| 4.8 | 2.2 |
| 4.0 | 3.3 |
| 2.9 | 4.4 |
| 2.6 | 4.7 |

Как видите, здесь результаты уже не столь близки к задуманным, но зато мы можем сузить диапазон "возможных" значений для X и Y. Например так:

XMAX = 6: XMIN = 0:

YMAX = 6: YMIN = 0

И теперь "прогон" программы еще восемь раз дал другую серию результатов:

| X            | Y    |
|--------------|------|
| 4.55         | 2.61 |
| 4.25         | 2.98 |
| 4.56         | 2.60 |
| 5.21         | 1.69 |
| 4.54         | 2.61 |
| 4.57         | 2.57 |
| 4.58         | 2.57 |
| 4.69         | 2.43 |
| Среднее 4.65 | 2.51 |
| Точность 7%  | 25%  |

А теперь, в заключение этой статьи, мы предлагаем Вам пару легенд, из которых Вы поймете, что такие "детские игры" могут в реальной жизни приводить к экономии тысяч рублей личных и миллиардов рублей государственных денег.

## 1.

Дело было почти 20 лет назад. СССР произвел запуск очередной межпланетной станции в дальний космос. Как на грех, сразу после пуска выясняется, что одна из ответственных систем вообще не работает. Причем, пока она и не нужна, а вот когда через несколько месяцев станция дойдет до конечной цели своего путешествия, то без нее не обойтись. Но меры можно предпринимать только сейчас, потом будет поздно. Надо что-то срочно делать.

Было несколько путей выхода из создавшегося положения и окончательный выбор решения лег на плечи молодого специалиста, стаж работы которого был около двух лет. Поставили срок - три дня и поручили обосновать принятое решение необходимыми расчетами.

Уже через под-дня грубых расчетов оказалось, что предсказать, что даст то или иное решение можно только решив пару уравнений размером в пол-страницы, в котором к тому же было пять неизвестных параметров. Конечно, эти параметры можно было "вытаскивать"

ставя практические эксперименты, но на это нужны были бы годы и такая работа оказалась бы на уровне десятка кандидатских диссертаций.

Беглый поиск в технической библиотеке показал, что можно попробовать воевать с таким уравнением. Еще пол-дня ушло на изучение основ вариационного исчисления и теории оптимального управления, а еще через два дня после отсева всего лишнего, почерпнутого из научных трудов, родился алгоритм, немного похожий на предложенный Вам. Решение было найдено и подтверждено расчетами, благо ЭВМ ЕС-1022 в те годы уже существовала.

Дело оставалось за малым - защитить это решение перед Государственной комиссией и убедить ее в правильности предложенных мероприятий.

Начальника отдела чуть не хватил удар, когда он увидел, что решение получено из двух уравнения с кучей неизвестных. Его реакцию словами не передать, и вот тогда и началась игра, которую мы Вам описали в самом начале. Задумывались числа, составлялись уравнения, а молодой специалист "угадывал" то, что задумал начальник. Пришло еще более высокое руководство и увлеченно включилось в игру. Одним словом, вдоволь наигравшись, они вынуждены были перекреститься и согласиться на рискованное с их точки зрения решение. Их поддержка на заседании Госкомиссии решила дело. Конечно, вопрос о том, каким путем решалось уравнение там не поднимался. Военные - люди серьезные, им не до таких мелочей. Им важно знать, что будет и как будет и кто будет отвечать, если все будет не так, а иначе.

Можно ли использовать положительные результаты кабинетной игры в обоснование сложных расчетов для ответственной техники? Во всяком случае, акты и протоколы были подписаны и ответственность легла на плечи отважных начальников - их убедила именно игра!

Через четыре месяца после этого станция полностью выполнила поставленную перед ней задачу, а поведение всех систем по данным телеметрической информации совпало с расчетным с точностью до полутора процентов, так были спасены миллиарды государственных рублей, затраченных на программу полета. Восторги молодого специалиста по этому поводу можно не описывать, а отважные начальники лукаво усмехались, принимая поздравления на самом высоком уровне.

История получила, кстати, неожиданное продолжение. Мы ведь сказали, что в уравнение входило много неизвестных параметров, для определения которых требовались длительные и дорогие испытания техники. Но раз уж приближенное решение было найдено, то интересно стало поглядеть и на то, чему эти параметры оказались равны. Так вот, два из них оказались настолько далеки от того, что предсказывала теория, что это вызвало массу сомнений. Раз за разом случайным поиском искали новые решения, но эта два параметра все равно отличались от теории раза в два. Впоследствии, когда экспедиция закончилась успешно и подтвердила расчет, стали проверять теорию и выяснили, что в ее основу была положена модель начала 60-х годов, которая успела устареть. Скорректировав модель и построив новую теорию, с удивлением обнаружили, что эти параметры оказались почти такими, как показало решение уравнения со многими неизвестными.

## 2.

Год 1990-ый. Только что появился на свет "ИНФОРКОМ" и сразу предложил своим клиентам первый десяток методических разработок.

Встал щепетильный вопрос определения цены на каждую. Большую цену назначить нельзя. Деньги, затраченные на их подготовку и печать не вернутся и фирму ждет мгновенная смерть. Малую цену назначить нельзя, ибо если спрос превысит очень ограниченные физические возможности коллектива, то не хватит подготовленного тиража, не хватит сил и времени на транспортировку и отправку, не хватит места для хранения, сорвется подготовка новых работ. Пойдут срывы заказов, потеря доверия клиентов и в итоге тоже смерть.

Два десятка неизвестных (цена на каждую из разработок и их тиражи) вошли в

несколько уравнений, увязывающих физические возможности коллектива, полиграфические мощности дружественного предприятия, почтовые возможности дружественного отделения связи и потенциальные финансовые возможности неизвестных будущих клиентов.

Расчет проводился на этот раз уже не на ЕС-1022, а на "Спектруме", что намного удобнее, по той же известной теперь Вам методике и в результате его были получены цены, которые надо назначить на ту или иную разработку и примерный ожидаемый тираж (для каждой - свой). По прошествии года исключительно ради интереса было проведено сравнение того, что дал компьютер с тем, что оказалось на самом деле. Самая грубая ошибка составила всего 35 процентов. Так еще раз удачно сработала эта увлекательная, но очень полезная игра.

В заключение хочется еще раз напомнить тем, кто нас читает - игра дело серьезное! Играйте на здоровье, но старайтесь при этом одновременно и обогащаться знаниями - кто знает, когда это пригодится!

# SINCLAIR LOGO

(Продолжение)

Начало см. на с. 69-74, 90-95, 178 - 182.

## 7. И снова об обработке списков.

Мы уже видели с Вами некоторые процедуры, выдающие какой-то результат и назвали их операциями. С помощью команды OUTPUT Вы можете сами создавать такие операции. В этом случае команда OUTPUT определяет то значение, которое должно быть возвращено в качестве результата. Например, если мы хотим создать процедуру, которой нам не хватает в стандартной версии ЛОГО, такую, которая будет вычислять разность двух чисел аналогично тому, как процедура SUM вычисляет их сумму, то мы делаем так:

```
TO DIFFERENCE: NUM1: NUM2
  OUTPUT :NUM1 - :NUM2
END
```

и тогда:

```
PRINT DIFFERENCE 7 5
даст результат:
2
```

Когда ЛОГО встречает команду OUTPUT, он передает то значение, которое находится справа от слова OUTPUT и останавливает процедуру. Поэтому DIFFERENCE 7 5 имеет значение 7-5.

Вы можете в одной процедуре иметь несколько команд OUTPUT, но в этом случае они должны стоять в разных ветвях IF, поскольку первая же встреченная команда OUTPUT прервет исполнение процедуры, действительно, нам ведь нужен только один результат ее работы.

Например, Вы можете сделать так, чтобы процедура вычисления разности двух чисел всегда выдавала положительный результат. Для этого надо, чтобы оба числа сравнивались между собой и из большего вычиталось меньшее.

```
TO DIFFERENCE :NUM1 :NUM2
  IF :NUM1> :NUM2 [OUTPUT :NUM1-:NUM2]
  OUTPUT :NUM2 - :NUM1
END
PRINT DIFFERENCE 10 3
7
PRINT DIFFERENCE 2 6
4
```

Можно, конечно, было записать и так:

```
IF :NUM1> :NUM2 [OUTPUT :NUM1-:NUM2][OUTPUT :NUM2-:NUM1]
```

но в этом нет особой необходимости, если исполняется первая команда OUTPUT, то и процедура закончится, а до второй команды OUTPUT дело и не дойдет.

Точно так же можно сконструировать и процедуру, выдающую по команде OUTPUT слово или список. Например, мы хотим узнать, начинается ли слово с гласной буквы или нет. Если да, то в качестве артикля перед ним надо ставить по правилам английской грамматики не "A", а "AN".

Поскольку нас интересует только первая буква в слове, то неплохо проверить, не является ли слово пустым списком.

Вот пример, который выдает значение "AN", если слово начинается с гласной, или выдает "A", если слово начинается с иной буквы или выдает пустой символ, если слово -



пустое.

```
TO ARTICLE :AWORD
  IF EMPTY? :AWORD [OUTPUT ""]
  IF MEMBER? FIRST :AWORD[A E I O U][OUTPUT "AN"]
  OUTPUT "A"
END
```

Попробуйте сами изменить процедуру ARTICLE так, чтобы она могла работать не только с прописными, но и со строчными буквами тоже. Бывают случаи, когда операциями надо пользоваться очень осторожно. Прежде всего, некоторые операции могут иметь побочные эффекты, т.е. они могут вести себя как команды. Например, мы можем создать операцию, которая запросит имя пользователя и выдаст первое из введенных в нее слов в качестве результата.

```
TO GETNAME
  PRINT [WHAT IS YOUR NAME]
  OUTPUT FIRST READLIST
END
```

Поэтому:

```
MAKE "YOU GETNAME
```

даст сообщение

```
WHAT IS YOUR NAME
```

и будет ждать ввода с клавиатуры, прежде чем передаст первое прочитанное слово в качестве результата операции. Все это выглядит достаточно просто, но что будет, если Вы попробуете:

```
PRINT GETNAME
```

Обратите внимание на то, что внутри процедуры GETNAME тоже имеется команда печати PRINT. ЛОГО вычисляет результат работы процедуры справа налево и потому сначала исполнит GETNAME, а затем передаст результат команде PRINT.

То же самое произойдет и в команде:

```
PRINT SENTENCE "HELLO GETNAME
```

Здесь и GETNAME и "HELLO обеспечивают входные данные для команды SENTENCE, которая передает свой результат команде PRINT.

Попробуйте:

```
PRINT (SENTENCE GETNAME "ALIAS GETNAME)
```

и посмотрите, получится ли в итоге именно тот результат, которого Вы ждете.

Второй момент, о котором нужно всегда помнить, заключается в том, что всякий раз, когда вызывается исполнение операции, она работает, как в первый раз.

```
PRINT DIFFERENCE 7 5
2
PRINT DIFFERENCE 7 5
2
```

Так все и должно быть и ничего иного мы и не ждем (но при многократных повторных вычислениях на это уходит много времени). Сколько бы раз Вы не вызывали GETNAME, эта процедура всегда будет делать запрос и ждать ввода с клавиатуры. Если же Вам надо запомнить то имя, которое было введено в самый первый раз, то Вам надо присвоить его символьной переменной и обращаться к ней всякий раз, когда Вам будет необходимо:

```
MAKE "YOU GETNAME
PRINT SENTENCE "HELLO :YOU
PRINT SENTENCE [I AM GLAD TO MEET YOU]: YOU
```

## Уровни процедур.

Когда одна процедура вызывает другую, мы говорим, что мы перешли на второй уровень процедуры. Если и эта процедура, в свою очередь, вызывает еще одну, то это уже третий уровень и т. д. Вот, например, три процедуры:

```
TO PROC1
  PRINT [LEVEL 1]
  PROC2
END
TO PROC2
  PRINT [LEVEL 2]
  PROC3
END
TO PROC3
  PRINT [LEVEL 3]
END
```

Здесь PROC1 - это первый уровень. Когда она вызывает PROC2, мы переходим на второй уровень. Когда же она вызывает PROC3, то это уже третий уровень, после чего Вам надо выполнить три команды END, STOP или OUTPUT, прежде чем мы вновь вернемся на верхний уровень (нулевой уровень).

Нулевой уровень в ЛОГО - это командный уровень, именно на нем Вы находитесь, когда перед Вами на экране изображено приглашение "?". Есть и специальная команда TOPLEVEL, которая позволяет сразу вернуться на верхний уровень, независимо от того, где Вы находитесь. Впрочем, этой командой пользоваться не рекомендуется, разве что в случае крайней необходимости.

Уровень, на котором Вы находитесь, никак не зависит от содержания тех процедур, которые Вы исполняете, а зависит только от того, сколько процедур, вложенных друг в друга, Вы исполнили. Представьте себе, что каждая встреченная на Вашем пути процедура - это ступенька вниз, а каждая встреченная команда END, STOP или OUTPUT - это ступенька вверх. Поэтому, если находясь на самом верхнем (командной) уровне Вы вызовете PROC3, то перейдете на первый уровень, хотя по своему содержанию PROC3 устроена так, что выдаст сообщение <LEVEL3>.

Интересны рекуррентные процедуры. Они могут уходить в глубину на очень много уровней.

```
TO DOLEVEL :N
  PRINT SENTENCE "LEVEL :N
  IF :N<10[DOLEVEL :N+1]
END
```

Тогда команда DOLEVEL1 даст следующую печать:

```
LEVEL1
LEVEL2
LEVEL3
LEVEL4
LEVEL5
LEVEL6
LEVEL7
LEVEL8
LEVEL9
```

Когда же, в конце концов, N станет равно 10, мы выполним команду END на 9-ом уровне, потом на 8-ом и так далее до верхнего уровня, в этом можно и убедиться самому, если изменить DOLEVEL так:

```
TO DOLEVEL :N
  PRINT SENTENCE -LEVEL :N
  IF :N<10 [DOLEVEL:N+1]
  PRINT "FINISH
END
```

```
DOLEVEL 1
  LEVEL1
  LEVEL2
  LEVEL3
  LEVEL4
  LEVEL5
  LEVEL6
  LEVEL7
  LEVEL8
  LEVEL9
  FINISH
  FINISH
  FINISH
  FINISH
  FINISH
  FINISH
  FINISH
  FINISH
```

### **Глобальные и локальные переменные.**

Когда слово используется в качестве входного параметра в процедуру (при определении процедуры), тогда при вызове процедуры это слово должно принимать то значение, которое было задано при вводе. Простой пример:

```
TO SAYNUM :N
  PRINT SENTENCE [N IS] :N
END
```

**И тогда:**

```
SAYNUM 5
  дает ожидаемый результат:
  N IS 5
```

**Теперь попробуйте так:**

```
MAKE "N 7
SAYNUM 5
  N IS 5
```

**А теперь попробуйте:**

```
PRINT N
```

На этот раз результат будет неожиданным - 7. ЛОГО запомнит то значение, которое имела переменная N до вызова процедуры и отложит его до тех пор, пока процедура не закончится.

Итак, мы можем считать, что переменная N в процедуре SAYNUM является локальной для данной процедуры, т.е. SAYNUM имеет свое собственное значение N, которое может быть использовано только внутри этой процедуры. Те же слова, которые не указаны при определении процедуры в качестве ее параметров, являются глобальными (т.е. значения этих переменных одинаковы во всех частях программы). Отсюда вытекает одно важное следствие для того случая, когда используем несколько уровней вложения процедур. Мы покажем его на примере трех процедур PROC1, PROC2, PROC3.

```
TO PROC1 :N
  PRINT SENTENCE "PROC1 :NUM
  PROC2 5
  PRINT SENTENCE [PROC1 AGAIN] :NUM
END
TO PROC2 :N
  PRINT SENTENCE "PROC2 :NUM
  PROC3 7
  PRINT SENTENCE [PROC2 AGAIN] :NUM
END
TO PROC3 :N
```

```
PRINT SENTENCE "PROC3 :NUM
END
```

Здесь процедуры 1 и 2 печатают значение своего параметра прежде, чем вызывать следующую процедуру, поэтому:

```
PROC1 3
PROC2 5
PROC3 7
PROC2 AGAIN 5
PROC1 AGAIN 3
```

Итак, ЛОГО "помнит", какому уровню процедур, какое значение NUM соответствует. Проще всего, по-видимому, представлять себе локальные переменные, как совершенно новые переменные, когда Вы находитесь внутри процедуры, несмотря на то, что они могут иметь то же имя, что и переменные, встречающиеся вне этой процедуры.

Если локальная переменная изменяется внутри процедуры, то это не имеет никаких эффектов на все одинаковые переменные вне этой процедуры. Если же изменяется глобальная переменная, то это изменение действует всюду, например:

```
TO TEST :NUM
  MAKE "NUM :NUM+1
  PRINT :NUM
  MAKE "OTHER :OTHER+1
END
MAKE "NUM 5
MAKE "OTHER 7
TEST 5
6
PRINT :NUM
5
PRINT :OTHER
8
```

Это имеет важное значение для рекурсии, Вы можете положиться на ЛОГО, когда он закончит рекуррентную процедуру в том смысле, что ЛОГО "помнит", где он был и каковы были там значения локальных переменных.

### **Команда OUTPUT для рекуррентных процедур.**

Вы можете использовать процедуры рекуррентно, поставив в команду OUTPUT имя самой этой же процедуры, но если Вам надо, чтобы эта процедура все-таки закончилась и при этом выдала бы какой-то результат. Вам надо иметь еще хотя бы один OUTPUT, в котором не происходит рекуррентного вызова этой процедуры еще раз.

Чтобы все это стало ясным, мы изменим ранее использованную процедуру GETNAME так, чтобы она запрашивала имя пользователя и выдавала бы первое слово введенного имени, если это возможно, а если нет, то печатала бы соответствующее сообщение и вызывала бы саму себя для повторения попытки. Правда, имейте в виду, что это приведет и к повторной печати запроса. Мы не будем здесь пользоваться конструкцией FIRST READLIST.

Поскольку оператор FIRST не любит пустые списки (и это естественно) и потому сначала проверим, не является ли введенный список пустым, а только потом будем к нему применять оператор FIRST.

```
TO GETNAME
  PRINT [ПОЖАЛУЙСТА НАЗОВИТЕ ВАШЕ ИМЯ]
  MAKE "YOU READLIST
  IF EMPTY? :YOU[PRINT[ХОТЯ БЫ ОДНО ИМЯ У ВАС ДОЛЖНО БЫТЬ]
  MAKE "YOU FIRST :YOU
  IF NUMBERP:YOU[НОМЕР ВМЕСТО ИМЕНИ БЫВАЕТ ТОЛЬКО У ЗАКЛЮЧЕННЫХ] OUTPUT GETNAME]
  OUTPUT :YOU
END
```

Это же можно проиллюстрировать на примере процедуры, входными параметрами которой будут число и слово, а в качестве результата будет слово, в которое будут вводить

столько символов, сколько указано входным числом. Итак:

```
START 5 "RECURSIVE
```

```
  выдаст
```

```
  RECUR.
```

Если :N - это число, а :INWORD - входное слово, то мы хотим получить от него :N первых символов. Теперь мы можем выделить из слова первый символ. После этого можно рекуррентно вызывать процедуру START до тех пор, пока еще не будут выданы :N-1 символов от оставшейся части входного слова :INWORD, т.е.

```
START :N-1 BUTFIRST :INWORD,
```

а затем поместить первый символ в новое слово:

```
WORD FIRST :INWORD
```

```
START :N-1 BUTFIRST :INWORD,
```

после чего выдать его в качестве результата с помощью OUTPUT.

Пока все идет хорошо, но встает традиционный вопрос: "Где же остановить рекурсию?" Поскольку N постепенно уменьшается и указывает на то, сколько еще символов мы хотим получить, то остановиться надо тогда, когда :N равно нулю и выдать "пустое слово" для обеспечения перехода START вверх на предыдущий уровень.

```
TO START :N :INWORD
```

```
  IF :N= [0 OUTPUT"]
```

```
  OUTPUT WORD FIRST :INWORD
```

```
  START :N-1 BUTFIRST :INWORD
```

```
END
```

```
PRINT START 5 "RECURSIVE
```

```
  RECUR
```

Давайте проследим ход работы процедуры с помощью следующей таблицы (табл. 1):

Таблица 1

| Уровень | :N | : INWORD  | FIRST<br>: INWORD | BUTFIRST<br>: INWORD | OUTPUT            |
|---------|----|-----------|-------------------|----------------------|-------------------|
| 1       | 5  | RECURSIVE | R                 | ECURSIVE             |                   |
| 2       | 4  | ECURSIVE  | E                 | CURSIVE              |                   |
| 3       | 3  | CURSIVE   | C                 | URSIVE               |                   |
| 4       | 2  | URSIVE    | U                 | RSIVE                |                   |
| 5       | 1  | RSIVE     | R                 | SIVE                 |                   |
| 6       | 0  |           |                   |                      | "                 |
| 5       | 1  | RSIVE     | R                 |                      | R=WORD"R"         |
| 4       | 2  | URSIVE    | E                 |                      | UR=WORD"U"R       |
| 3       | 3  | CURSIVE   | C                 |                      | CUR=WORD"C"UR     |
| 2       | 4  | ECURSIVE  | U                 |                      | ECUR=WORD"E"CUR   |
| 1       | 5  | RECURSIVE | R                 |                      | RECUR=WORD"R"ECUR |

### Вложенные списки.

В качестве элементов списков в ЛОГО тоже могут быть списки. В этом нет ничего удивительного. Если мы составим список того, что лежит у нас в кармане, то можем получить:

```
[МОНЕТА КАРАНДАШ ЛАСТИК СПИСОК ПОКУПОК]
```

Обратите внимание на то, что объекты, входящие в список покупок вовсе не являются элементами списка того, что лежит у нас в кармане, таким образом, список содержимого кармана имеет 4 элемента, несмотря на то, что список покупок сам является списком:

```
[ХЛЕБ МОЛОКО МЯСО МАРКИ]
```

а ХЛЕБ не является вещью, находящейся в кармане.

Более того, ЛОГО даже допускает создать такой список содержимого кармана:

```
[МОНЕТА КАРАНДАШ ЛАСТИК [ХЛЕБ МОЛОКО МАСЛО МАРКИ]]
```

и в этом списке по-прежнему всего 4 элемента, несмотря на то, что четвертый элемент представляет из себя список из 4-х элементов.

Сколько же элементов в следующем списке?

```
[ [ХЛЕБ МОЛОКО МАСЛО МАРКИ] [ДЖЕЙН ДЭВИД РОЗМАРИ] [СТИРКА ГЛАЖКА]  
[8_МАЯ 20_ИЮНЯ 1_ИЮЛЯ 10_СЕНТЯБРЯ 23_НОЯБРЯ] ]
```

В этом списке 4 элемента. Все они - тоже списки. Проще было бы дать им имена:

```
[ПОКУПКИ ДРУЗЬЯ ЗАБОТЫ ДАТЫ]
```

Вы составили такой список для себя, теперь Вы можете составить аналогичный список для друга и еще раз для кого-то и тогда получите список списков:

```
[МОЙ ТВОЙ ЧУЖОЙ]
```

Здесь МОЙ - это [ПОКУПКИ ДРУЗЬЯ ЗАБОТЫ ДАТЫ], точно так же ТВОЙ и ЧУЖОЙ. Так можно продолжать и дальше, вкладывая список один в другой, но наверное стоит остановиться и предоставить такую работу ЛОГО.

### Процедуры создания списков

Мы уже использовали команду SENTENCE для того, чтобы создать список из двух и более элементов, которые могут быть словами или списками. Эта команда объединяет их в один список, снимая внешние квадратные скобки и переписывая весь список заново. Например:

```
(SENTENCE [ВОТ ВАМ][ПРИМЕР ДЕЙСТВИЯ][ОПЕРАТОРА SENTENCE])
```

выдает в качестве результата:

```
[ВОТ ВАМ ПРИМЕР ДЕЙСТВИЯ ОПЕРАТОРА SENTENCE]
```

Сейчас же мы с Вами рассмотрим три новых оператора, по-разному создающих списки.

FPUT (это First PUT). Имеет два входных параметра. Второй параметр - список. Команда FPUT создает новый список, в котором первый элемент берется из первого входного параметра, а остальная часть списка из второго.

Например:

```
FPUT "РАЗ[ДВА ТРИ ЧЕТЫРЕ]
```

выдаст список:

```
[РАЗ ДВА ТРИ ЧЕТЫРЕ]
```

В отличие от SENTENCE здесь не происходит слияния внешних квадратных скобок при объединении списков, например:

```
FPUT [РАЗ ДВА][ТРИ ЧЕТЫРЕ]
```

дает:

```
[ [РАЗ ДВА] [ТРИ ЧЕТЫРЕ]
```

Можете считать, что FPUT берет свой первый параметр и ставит его в первую позицию второго параметра.

Другой оператор, который мы рассмотрим, - это LPUT (Last PUT). Он работает совершенно аналогично, но с одним исключением. Первый параметр LPUT ставится последним элементом в списке, выраженном вторым входным параметром.

```
LPUT [ПОСЛЕДНИЕ СЛОВА][ЭТО]
```

дает:

[ЭТО [ПОСЛЕДНИЕ СЛОВА]]

Третий оператор - LIST. Он имеет два или более входных параметров и выдает их в виде списка, независимо от того, являются ли они списками или словами.

```
LIST "БАНК [ХЛЕБ МОЛОКО МАСЛО МАРКИ]
```

имеет два входных параметра и выдает список следующего содержания:

```
[БАНК [ХЛЕБ МОЛОКО МАСЛО МАРКИ]
```

Если LIST имеет более двух входных параметров, то они должны быть заключены в круглые скобки.

```
(LIST "БАНК [ХЛЕБ МОЛОКО МАСЛО МАРКИ][ДЖЕИН ДЭВИД РОЗМАРИ]).
```

выдает список:

```
[БАНК [ХЛЕБ МОЛОКО МАСЛО МАРКИ] [ДЖЕИН ДЭВИД РОЗМАРИ]]
```

Можете полагать, что команда LIST просто берет все свои параметры и ставит вокруг них дополнительную пару квадратных скобок.

**ВНИМАНИЕ!** Когда LIST используется более, чем с двумя параметрами и один из них равен самому списку LIST, то в этих случаях оператор не всегда работает правильно. Он также может вызвать проблемы, если включен в квадратные скобки при операторе REPEAT или в список IF. Очень сложные конструкции, таким образом, следует создавать постепенно.

### Сортировка чисел.

Теперь мы посмотрим, как можно использовать списки для создания сложных структур данных и для некоторых других практических целей. Сначала мы рассмотрим вопрос сортировки последовательности чисел, введенных с клавиатуры, сортировка данных это очень важный вопрос, в котором проявляется мощь компьютеров. Существует много методов для выполнения сортировки, но мы остановимся на одном из них.

Мы будем хранить числа в структуре, называемой "деревом". На рис. 1 показан пример такого дерева.

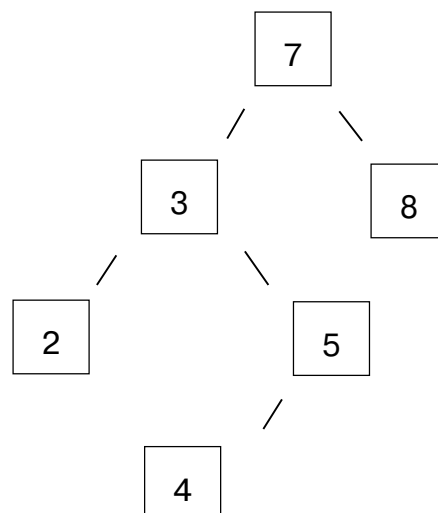


Рис. 1

Если не очевидно, почему эта структура получила такое название, переверните рисунок вверх ногами. Каждое обведенное число называется "узлом", в каждом из узлов дерево может ветвиться (максимум на две ветви).

Посмотрите, как это организовано. Возьмем, например, число 6 и введем его в наше дерево. Начнем с вершины и в каждом узле будем сравнивать наше число с тем, что

содержится в данном узле. Если наше число меньше, то из узла пойдём по левой ветви, а если больше - то по правой.

Представим, что число 6 помещается на вершину дерева, поскольку  $6 < 7$ , то идем вниз по левой ветви к числу 3. Здесь у нас  $6 > 3$  и продолжаем путь по правой ветви к числу 5. Раз  $6 > 5$ , то и здесь надо пойти по правой ветви, но на сей раз у нас правой ветви нет и ее надо сделать. Просто пририсовываем правую ветвь к узлу 5 и поставим в ее конце узел 6.

Итак, каждый узел может иметь две ветви, одну влево для чисел, которые меньше, чем значение в узле и одну вправо для чисел, которые больше, чем значение в узле, простой вопрос "наше число больше, чем значение в узле или меньше?" определяет, по какой ветви нам идти.

Точно так же можно построить дерево и для списков. Каждый узел может быть представлен списком из трех элементов.

Средний элемент - это номер узла, а два других элемента - тоже списки. Первый элемент определяет левую ветвь дерева, а последний - правую, каждая ветвь выстраивается аналогично. Если из узла ветвь не исходит, то ей соответствует пустой список. Таким образом, пустые списки служат как бы маркерами, показывающими, что мы дошли до конца данной ветви.

Вы, наверное, не удивитесь, когда узнаете, что такие конечные узлы называют "листьями", а исходный узел - "корнем".

Дерево для структуры списков показано на рис. 2.

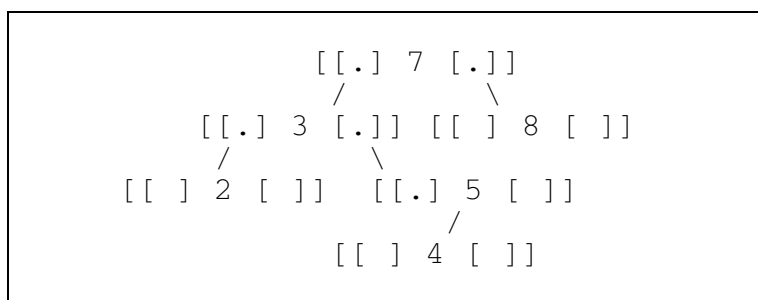


Рис.2

В тех узлах, где список не пустой, стоит "точка" [.] и стрелка, показывающая содержимое данного списка, поэтому несмотря на то, что списки находятся один внутри другого, такой "развернутый" вид действительно демонстрирует похожесть на настоящее дерево.

Давайте еще раз посмотрим, как число 6 вписывается в это дерево. Рассмотрим корневой узел на вершине диаграммы, он имеет 3 элемента, из которых средний - число 7. Новое число 6 меньше и потому мы идем по левой ветви, рассматривая левый список. Этот список имеет средний элемент, равный 3. Поскольку 6 больше, то мы ищем правое продолжение, новый список имеет средним элементом 5. Раз 6 больше, чем 5, мы ищем опять правое продолжение. На этот раз список пуст, мы дошли до конца ветви и теперь нашли место, в которое должно быть помещено число 6. Поскольку это последняя точка, то мы отмечаем ее пустыми списками: `[[6[]]]`.

Итак, существуют только два правила для того, чтобы поместить новое число в дерево.

1) Если число помещается в узел, в центре которого уже есть число, то эти числа сравниваются и, если новое число меньше, то оно помещается в начальный список, а если больше, то в конечный.

2) Если число помещается в пустой список, оно становится в средний элемент, а справа и слева добавляются по пустому списку.

Если мы начнем с пустого списка и будем вводить числа в порядке 7,3,5,8,... то у нас будет создаваться список так, как показано в табл. 2. На каждом шагу мы показываем только новые вошедшие в список элементы, дабы сделать структуру проще и нагляднее.



```

ШАГ
1      [                               ]
2      [       ] 7 [       ]
3      [ ] 3 [       ]
4      [ ] 5 [       ]
5      [ ] 8 [       ]

```

Как видите, числа ложатся в список в правильном порядке.

Теперь рассмотрим, как сделать это на языке ЛОГО. Сначала нам нужна процедура, которая выдаст пользователю необходимые инструкции и создаст исходный пустой список, затем выполнит ввод чисел и распечатает упорядоченный список.

```

TO SORT
  PRINT [Введите числа]
  PRINT [Когда закончите, наберите END]
  MAKE "NUMLIST []
  INPUT
  PRINT INORDER :NUMLIST
END

```

Теперь рассмотрим процедуру INPUT. Она должна принять с клавиатуры первое слово и проверить, не число ли это. Мы используем тот факт, что это не число для того, чтобы сигнализировать о том, что ввод закончен. Если же это число, то его надо ввести в список и посмотреть, куда оно ложится, после чего сформировать новый список.

```

TO INPUT
  MAKE "NUM FIRST READLIST
  IF NOT NUMBERP :NUM [STOP]
  MAKE "NUMLIST RENEW :NUMLIST
END

```

Здесь мы впервые использовали операцию NOT. Она берёт то условие, которое стоит в качестве ее параметра и изменяет полученный результат с TRUE на FALSE или, соответственно, наоборот. Так что она работает достаточно просто и выдает результат TRUE (ИСТИНА), если :NUM числом не является.

Теперь нам нужно записать процедуру RENEW, которая проверит введенное число по всем имеющимся спискам и вставит его на свое законное место. Эта процедура должна создавать списки из трех элементов, обычно по результатам IF. Как было указано в предыдущей главе, LIST нельзя использовать с круглыми скобками внутри, так что мы начнем с того, что напишем процедуру, которая примет три параметра и сделает из них список:

```

TO LIST :A :B :C
  OUTPUT (LIST :A :B :C)
END

```

Теперь, когда мы проверяем введенное число по имеющемуся списку (назовем процедуру ALIST), у нас могут быть три возможности исхода:

- список может быть пустым, в этом случае мы заменяем его новым списком из трех элементов, внешние из которых являются пустыми списками, а средний - введенным числом;

```
LIST 3 []:NUM[]
```

- если вводимое число больше, чем средний элемент списка, то новый список замещает тот, который был. В нем первые два элемента остаются неизменными, а в качестве третьего вводится новый список, содержащий вводимое число:

```
LIST3 FIRST :ALIST ITEM2 :ALIST
```

```
RENEW LAST :ALIST
```

- если вводимое число меньше, чем средний элемент, то в новом списке меняется первый элемент:

```
LIST3 RENEW FIRST :ALIST ITEM2 :ALIST LAST :ALIST
```

Теперь мы пришли к процедуре RENEW:

```
TO RENEW :ALIST
  IF EMPTY :ALIST [OUTPUT LIST3[] :NUM[]]
  IF :NUM>ITEM2 :ALIST
    [OUTPUT LIST3 FIRST :ALIST
     ITEM2 :ALIST
     RENEW LAST .ALIST]
  [OUTPUT LIST3
   RENEW FIRST :ALIST
   ITEM2 :ALIST
   LAST :ALIST]
END
```

Как мы уже видели, в результате наши числа будут упорядочены по старшинству. Проблемой, однако, является большое количество квадратных скобок. К счастью, здесь помогает сила рекурсии. Для того, чтобы упорядочить список, состоящий из списка, числа и еще одного списка, нам надо распечатать первый список в упорядоченном виде, число и второй список в упорядоченном виде. Если список пуст, то вообще ничего печатать не надо:

```
TO PRINTINORDER :ALIST
  IF EMPTY :ALIST [STOP]
  PRINTINORDER FIRST :ALIST
  PRINT ITEM2 :ALIST
  PRINTINORDER LAST :ALIST
END
```

Когда Вы закончите ввод процедур, дайте команду SORT. Введите несколько чисел, по одному на строке. Заканчивайте ввод чисел словом END или любым другим удобным словом, но не числом.

После того, как числа будут распечатаны в отсортированном порядке. Вы можете попробовать:

```
PRINT :NUMLIST
```

Теперь Вы увидите, как список хранится в структуре "дерева". Если же Вы хотите увидеть, как создается новый отсортированный список. Вы можете попробовать вставить эту команду в процедуру INPUT.

### **Самообучающаяся игра.**

Сейчас мы попробуем использовать структуру "дерева" для того, чтобы написать игру. Суть игры состоит в том, что один играющий задумывает некоторое животное, а другой должен его угадать. При этом он может задавать любые вопросы, но ответ может быть только "да" или "нет". Но самая интересная часть игры состоит в том, что программа способна самообучаться. В начале игры она знает только двух животных, а по мере того, как Вы с ней играете, она становится все более и более образованной.

Вся информация хранится в виде дерева. Каждый узел - это список, состоящий из трех элементов. Обычно средний элемент списка - вопрос, который надо задать игроку. Если ответ "да", то программа использует первый элемент списка (а он тоже список) для следующего хода. Если же ответ "нет", то программа использует последний элемент списка.

Мы неизбежно дойдем до последней точки. Она выявляется по тому факту, что и первый и последний элемент являются пустыми списками. В данном случае средний элемент - это уже не вопрос к играющему, а название животного.

Когда имя животного выдается игроку, то у него есть две возможности:

- если компьютер угадал верно, то программу не надо обучать;
- но если пользователь задумал другое животное, то программа запрашивает вопрос, с помощью которого можно различить то животное, которое задумал пользователь, и то, которое назвала программа. Это вопрос теперь становится средним элементом в новом узле, а названия животных - в узлах справа и слева. Опять же к каждому из этих названий добавляется справа и слева по пустому списку. Входная процедура должна ввести пользователя в суть игры и создать корневой список перед тем, как игра начнет свою работу.

```
TO ANIMAL
  PRINT " PRINT [Добро пожаловать на викторину]
  PRINT [Отвечайте только "ДА" или "НЕТ"]
  MAKE "NODE [[[]][УТКА][[]]
    [ОНО ПЛАВАЕТ?]
    [[]][СВИНЬЯ][[]]]
  GO
END
```

Рабочая процедура GO просит играющего задумать животное, проверяет элементы своего списка и обеспечивает повтор всего процесса для другого животного.

```
TO GO
  PRINT [ЗАДУМАЙ ЖИВОТНОЕ]
  PRINT "
  MAKE "NODE TRY :NODE
  PRINT [ЕЩЕ РАЗ?]
  MAKE "YN FIRST READLIST
  IF YN= "YES [GO]
END
```

Процедура GO проверяет текущий узел. Это может привести к изменению списка, поэтому процедура TRY выдает новое содержание узла.

Необходимо предусмотреть две возможности. Средний элемент в узле может быть либо вопросом, который надо задать игроку, либо готовым ответом.

Программа различает эти два случая, проверив первый элемент. Если он пуст, то во втором элементе содержится ответ.

```
TO TRY :ANODE
  IF EMPTY FIRST :ANODE
    [OUTPUT GUESSANSWER]
    [OUTPUT PUTQUESTION]
END
```

Теперь процедура GUESSANSWER должна угадать ответ и, если он верен, то изменений делать не надо, а если нет, то надо сконструировать новый узел?

```
TO GUESSANSWER
  PRINT "
  PRINT SENTENCE [ЭТО]
  ITEM2 :ANODE
  MAKE "YN FIRST READLIST
  IF :YN= "YES
    [PRINT [Я ТАК И ДУМАЛ]
    OUTPUT :ANODE]
    [OUTPUT NEWNODE]
END
```

Процедура NEWNODE запрашивает правильный ответ и конструирует новый узел для двух животных, используя при этом полученный от пользователя вопрос. "Новое" животное должно быть помещено в новый список между двумя пустыми списками. "Старое" животное уже организовано таким образом в текущем узле.

Здесь нам вновь будет нужна процедура LIST3, о которой мы говорили выше:

```
TO NEWNODE
  PRINT [Сдаюсь! что это за животное?]
  MAKE "ANIMAL1 READLIST
  MAKE "NEWPART[LIST[1:ANIMAL1[]]
  MAKE "ANIMAL2 ITEM2 :ANODE
  PRINT [SENTENCE[Дайте мне вопрос, чтобы отличить] :ANIMAL1 "и :ANIMAL2]
  MAKE "QUESTION READLIST
  PRINT [SENTENCE[и для ] :ANIMAL1 [ответ будет да?]
  MAKE "YN FIRST READLIST IF :YN="ДА[OUTPUT LIST3 :NEWPART :QUESTION :ANODE] [OUTPUT LIST3
    :ANODE :QUESTION :NEWPART]
END
```

Это самая сложная процедура в программе, но ключ к ее пониманию лежит в двух операциях OUTPUT, записанных в ее конце.

```
TO PUTQUESTION
  PRINT "
  PRINT ITEM2 :ANODE
  MAKE "YN FIRST READLIST
  IF :YN=" ДА [OUTPUT LIST3
  TRY FIRST :ANODE
  ITEM2 :ANODE
  LAST :ANODE]
  [OUTPUT LIST3 FISTST :ANODE
  ITEM2 :ANODE
  TRY LAST :ANODE]
END
```

И здесь тоже основная смысловая часть процедуры заключена в двух последних операциях OUTPUT.

Теперь, если все сделано правильно. Вы можете дать команду ANIMAL и, работая с программой, увидеть, как происходит ее самообучение.

## 8. Некоторые математические операции.

ЛОГО различает два различных вида чисел: целые (например 1, -23, 245 и т.д.) и десятичные (0.67, 28.91 и т.п.). Обычно Вам не надо думать о том, что между ними есть разница, вероятно, единственный случай, когда различие между ними существенно, это тогда, когда Вы пытаетесь ввести десятичное число в процедуру, которая ожидает в качестве своего параметра целое число. И даже в этом случае, как правило, ЛОГО предупредит Вас.

Операция INT (INTEGER) принимает в качестве входного любое число, а свой результат выдает в виде целого. Она, таким образом, отрезает десятичную часть числа.

```
PRINT INT 3,14
3
PRINT
6
PRINT INT -2.1
-1
```

Обратите внимание на то, что хотя 6,999 очень близко к числу 7, результатом все-таки будет число 6. Этот процесс всегда ведет к округлению вниз.

В некоторых случаях такая особенность может быть полезной. Например, иногда бывает важным найти результат деления одного числа на другое и образующийся при этом остаток.

Может это быть полезным, когда Вы проверяете, является ли данное число целым или нет. Если оно целое, то применение к нему функции INT уже не изменит его.

```
TO TESTINT1 :NUM
  IF :NUM-INT :NUM
```

```
[PRINT "YES] [PRINT "NO]
END
```

Таким же способом можно определить и делится ли одно число на другое нацело.

```
TO TESTDIV :NUM1 :NUM2
  IF (NUM1/NUM2) = INT(:NUM1/:NUM2)
    [PRINT "YES] [PRINT "NO]
END
```

Если же Вам на самом деле нужно ближайшее целое число, то Вы можете воспользоваться операцией **ROUND**. Она тоже может принимать любое число в качестве входного и выдает ближайшее целое число.

```
PRINT ROUND 3.14
3
PRINT ROUND 6.99
7
PRINT ROUND -2.9
-3
```

Эта операция берет свое название из термина rounding ("округление"). Она может быть использована для того, чтобы выдавать ответ с заданной степенью точности, с небольшими усилиями ее можно использовать и для получения результата с заданным количеством десятичных знаков.

Так, например, если Вам надо округлить 3.14159 до двух десятичных знаков, то Вы можете умножить это число на 100 (314.159), округлить его (314) и снова разделить на 100 (3.14). Эту операцию можно конвертировать в рекуррентную процедуру, на каждом шагу умножая исходное число на 10. Если количество десятичных чисел равно нулю, то число просто округляется.

```
TO CORRECT :NUM :DECPLS
  IF DECPLS=0 [OUTPUT ROUND :NUM]
  OUTPUT (CORRECT :NUM*10 :DECPLS -1)/10
END
```

Есть еще одна операция, которая всегда выдает в результате своей работа целое число. Это операция **REMAINDER**. Она требует два входных параметра и выдает остаток от деления нацело первого числа на второе.

```
PRINT REMAINDER 12 5
2
```

Эта операция обеспечивает еще один путь для проверки делится ли одно число на другое нацело. Если это так, то результат операции равен нулю.

### **ИСТИНА и ЛОЖЬ (TRUE и FALSE)**

Мы можем писать процедуры, которые проверяют условия и выдают в качестве результата **TRUE** или **FALSE**, это можно сделать одним из двух возможных способов, Вы можете выдавать результат в виде слова **TRUE**, **FALSE** и т.п., например:

```
TO DIVISIBLE :A :B
  IF 0=REMAINDER :A :B
    [OUTPUT "TRUE] [OUTPUT "FALSE]
END
```

Второй способ - получение ответа сразу в результате проверки и немедленной выдачи результата:

```
TO DIVISIBLE :A :B
  OUTPUT 0=REMAINDER :A :B
END
```

Здесь, если действительно остаток от деления :A на :B равен нулю, результат равен TRUE, а если нет - то результат равен FALSE.

Такой результат можно напрямую использовать для оператора IF.

```
IF DIVISIBLE 20 5 [PRINT "YES] [PRINT "NO]
```

Фактически Вы только что определили предикативную процедуру и чтобы это подчеркнуть, ее можно назвать DIVISIBLEP.

Иногда мы сталкиваемся с тем, что будет ли результат операции иметь значение TRUE или FALSE, зависит от одного или большего количества условий, ЛОГО обеспечивает три операции, работающих с предикатами (условиями), мы уже рассмотрели NOT, изменяющую результат условия на противоположный. Есть еще операция AND, которая дает в результате значение TRUE, если все входящие выражения имеют значение TRUE. Обычно операция AND обрабатывает два условия, но их может быть и больше, если применить круглые скобки.

```
PRINT AND 1<2 3<7
```

```
TRUE
```

Другая операция - OR (ИЛИ). Она дает в результате TRUE, если хотя бы одно из входящих выражений имеют значение TRUE. Так же, как и AND, операция OR может иметь более двух входных выражений.

При аккуратной работе можно объединять AND и OR в одном выражении. Например, Вы можете прийти ко мне на день рождения, если Вас зовут Фред или Вы женщина и к тому же красивая.

### **Возведение в степень и извлечение корня.**

Узнать квадрат числа можно умножив его само на себя, другие степени числа легко находятся с помощью рекурсии.

```
TO POWER :A :B
  IF :B<2 [OUTPUT :A]
  OUTPUT (POWER :A :B -1)*A
END
```

Для вычисления корней квадратных из числа, ЛОГО имеет специальную функцию SQRT.

```
PRINT SQRT 25
5
```

Мы можем воспользоваться теоремой Пифагора для вычисления расстояния между двумя точками с известными координатами (координата X и координата Y). В учебниках по математике Вы найдете, что расстояние между двумя точками с координатами [X1, Y1] и [X2, Y2] можно найти, как

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Рис. 3

```
TO DISTANCE :POS1 :POS2
  MAKE "XDIF FIRST :POS1-FIRST :POS2
  MAKE "YDIF LAST :POS1-LAST :POS2
  OUTPUT SQRT(XDIF*XDIF+YDIF*YDIF)
END
```

### **Поиск простых чисел.**

Простое число - это число, которое нацело делится только на единицу или само на себя. Проблема поиска простых чисел волнует математиков уже несколько тысяч лет. Пока не найдена формула, по которой можно было бы найти все простые числа, а те формулы, которые уже известны, выдают определенную последовательность из простых чисел, но начиная с какого-то числа выдают и непростые числа (См. ZX-РЕВЮ-91, №7,8. с. 133).

Хоть и не существует вполне надежной формулы для простых чисел, зато существует метод. Мы можем просто брать числа по очереди и проверять, делятся ли они нацело на другие числа, которые меньше, чем оно само. Задача упрощается тем, что нужно проверять не все числа, а только до корня квадратного из исходного числа. Ведь если оно делится на что-то, что больше, чем квадратный корень, то в результате получим частное, которое меньше, чем квадратный корень, а эти числа мы уже проверили.

```
TO TESTPRIME :NUM :N
  IF :N>SQRT :NUM[OUTPUT "TRUE]
  IF DIVISIBLE :NUM :N[OUTPUT "FALSE]
  OUTPUT TESTPRIME :NUM :N+1
END
```

Проверку надо начинать с числа 2 - это наименьшее простое число.

```
TO PRIMESFORM :NUM
  IF TESTPRIME :NUM2[PRINT :NUM]
  PRIMESFORM :NUM+1
END
```

Теперь команда PRIMESFORM 2 будет распечатывать простые числа, начиная с числа 2 и выше.

### Тригонометрические операции.

Если Вы уже знакомы с тригонометрическими функциями, то можете пропустить один абзац. Если же нет, то читайте дальше.

Если наша "черепашка" имеет значение HEADING, равное углу :ANGLE, то на каждом шаге она проходит разные расстояния вдоль экрана и поперек. Если "черепашка" пройдет в направлении своего движения единицу пути, то ее перемещение по горизонтали равно синусу угла :ANGLE, а перемещение по вертикали - равно косинусу угла :ANGLE. Если путь, пройденный "черепашкой" равен единице, то очевидно, что путь, пройденный ею по горизонтали, меньше единицы, то же и путь, пройденный по вертикали.

Может быть, Вас интересует не один шаг, пройденный "черепашкой", а сразу 10. В этом случае ее перемещение по горизонтали:

```
10*SINE :ANGLE
  а путь по вертикали:
10*COSINE :ANGLE
```

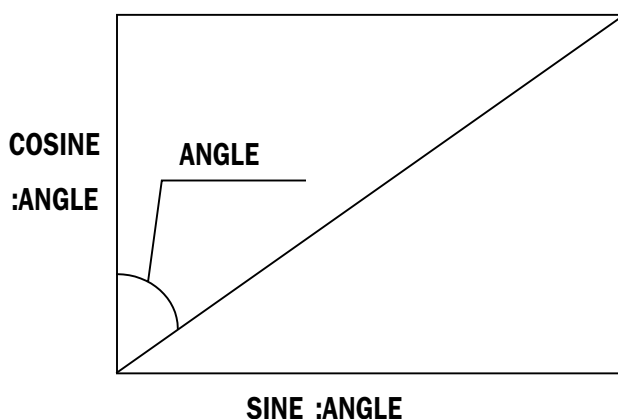


Рис. 4 Синус и косинус угла.

Математики пользуются еще двумя функциями: тангенс (это частное от деления синуса угла на его косинус) и котангенс (частное от деления косинуса на синус).

ЛОГО предоставляет возможность использования четырех операций: SINE, COSINE, TANGENT, COTANGENT, в сокращенной форме SIN, COS, TAN, COT. У ЛОГО есть большое преимущество перед другими языками программирования в том смысле, что ЛОГО принимает углы для этих операции в градусной мере, в то время как другие языки

программирования предпочитают иметь дело с радианами. Поэтому, если Вы привыкли к другим языкам и к радианной мере, то теперь вздохнете с облегчением. Если же Вы к ней не привыкли и никогда о ней не слышали, то тем более вздохнете с облегчением, поскольку можно о ней и не думать.

Кроме 4-х описанных выше операций, есть еще четыре: ARCSIN, ARCCOS, ARCTAN, ARCCOT, которые имеют прямо противоположное действие.

В качестве входного параметра они принимают число, а на выходе выдают угол в градусах, который имеет соответствующий синус (косинус, тангенс, котангенс).

Так, PRINT ARCCOS 0.5 даст Вам угол, косинус которого равен 0.5, т.е. 60 градусов.

Эти операции имеют огромное значение в изучении электроники, волновых явлений и вибраций. С помощью этих функций можно также исполнять очень красивые и интересные кривые. Рассмотрим, например, пару следующих процедур, которые изображают график функции косинус для углов от 0 до 360 градусов. Координаты X и Y здесь распечатываются через угол и его косинус, соответственно. Масштаб избран таким, чтобы изображение хорошо вписывалось в экран. Когда "черепашка" дойдет до границы экрана, Вы получите сообщение "turtle out of bounds".

```
TO SHOWCOS
  PENUP
  SETPOS [120 0] PENDOWN
  SETPOS [120 0] DO COS 0
END
```

```
TO DOCOS :ANGLE
  MAKE "X 2* :ANGLE/3-120
  MAKE "Y 80*COS :ANGLE
  SETPOS SE :X :Y
  DOCOS :ANGLE+15
END
```

Вы увидите колебания в вертикальной плоскости. Что произойдет, если мы добавим сюда еще и колебания по горизонтали? Вы получите фигуры, названные именем их первооткрывателя - фигуры Лиссажу.

```
TO LISSAJOU :A :B :C :D :INC
  SHOWTURTLE
  CLEARSCREEN
  PENUP
  LISS 0
END

TO LISS :ANGLE
  MAKE "X 120*COS(:A*:ANGLE+:B)
  MAKE "Y 80*COS(:C*:ANGLE+:D)
  SETPOS SE :X :Y
  PENDOWN
  LISS :ANGLE+:INC
END
```

Перо опустится после первой команды SETPOS (и после этого останется опущенным), поэтому "черепашка" пойдет в стартовую позицию без изображения линии. Чем выше значение INC, тем более угловатой будет полученная графика.

Попробуйте:

```
LISSAJOU 11 20 21 40 1
```

Введение в графику цвета позволяет получать очень интересные эффекты. Вот еще несколько примеров "черепашьей" графики:

```
TO CYCLO :K :ANGLE :SIZE
  FORWARD :SIZE*SIN(:K*HEADING)
```



```
RIGHT :ANGLE
CYCLO :K :ANGLE :SIZE
END
```

### Попробуйте

```
CYCLO 1 10 10
CYCLO 4 5 20
```

### И еще одна процедура:

```
TO LOBE :A :B :INC :ANGLE
  FORWARD 1
  RIGHT :A :B :INC :ANGLE+:INC
END
```

### Попробуйте:

```
LOBE 2 3 9 0
LOBE 10 8 6 0
LOBE 2 8 6 0
LOBE 2 9 3 0
```

(Окончание в следующем номере).

# ПРИМЕНЕНИЕ АССЕМБЛЕРА ДЛЯ СОЗДАНИЯ БЫСТРОРАБОТАЮЩИХ ПРОГРАММ

Перевод с английского Пашорина В.И.

Продолжение. Начало см. с. 9-12, 48-61, 97-103, 183-188

## 8. Команды ATTRIBUTE, SCREEN\$ и POINT

Если Вы программировали игры на БЕЙСИКе, то вероятно сталкивались с тем, что Вам необходимо знать в определенной игровой ситуации текущее расположение заданного символа на экране. Это, например, нужно при проверке условия - достигла ли ракета цели, либо при выполнении посадки на Луну. В БЕЙСИКе для этого существуют три способа:

- использовать оператор ATTRIBUTE, который проверяет значение INK, PAPER, FLASH и BRIGHT в заданном знакоместе экрана и выдает число, соответствующее этим параметрам;

- использовать оператор SCREEN\$, который определяет код символа в заданном знакоместе и выдает код этого символа, если он входит в диапазон 32...127. В противном случае считается, что ячейка пуста;

Примечание: процедуру в машинных кодах, которая может проверить не только символы ASCII, но и символы UDG-графики и символы блочной графики, т.е. из диапазона 32...164 мы привели в первом томе, посвященном графике для "Спектрума" - "Элементарная графика" на с. 99 под названием SCANNER. - ИНФОРКОМ.

- использование оператора POINT, который проверяет состояние одного пиксела экрана и выдает 1, если пиксел включен и 0, если он выключен.

Все эти три способа могут быть реализованы и в программах в машинных кодах с использованием специальных процедур ПЗУ.

### 8.1. ATTRIBUTE (строка, столбец).

Процедура ATTR, аналогичная по действию оператору ATTRIBUTE, находится в ПЗУ по адресу 9603. Перед ее вызовом необходимо записать координаты исследуемого знакоместа в регистровой паре BC (номер строки - в регистре B, а номер столбца - в регистре C). Результат проверки в виде числа 0...255 записывается на вершину стека калькулятора.

Определить атрибуты знакоместа, используя это число, можно по формуле:

ЧИСЛО=128\*FLASH + 64\*BRIGHT + 8\*PAPER + INK значение

Это число можно переписать со стека калькулятора в регистр A, используя процедуру ПЗУ FP\_TO\_AA, находящуюся по адресу 11733, а затем сравнить его с заданным числом N с помощью команды CP N.

Пример реализации этого способа показан в программе 8.1. Программа проверяет каждое знакоместо экрана до тех пор, пока не найдет ячейку, атрибуты которой соответствуют PAPER 1, INK 7, FLASH 1, т.е. числу 143. После запуска этой программы на экране появляется ракета, медленно перемещающаяся по направлению к человечку. При совмещении ракеты с этим человечком происходит возврат в БЕЙСИК. Вы можете вместо команды возврата RET подключить процедуру, создающую звуковой эффект, сопровождающий попадание ракеты в человечка.

Такой способ проверки атрибутов символьных ячеек безусловно хорош, но требует, чтобы, например, все поражаемые объекты имели одинаковые атрибуты.

Листинг 8. 1.

| АДРЕС | МЕТКА | МАШ. КОД    | АССЕМБЛЕР                   | КОММЕНТАРИЙ             |
|-------|-------|-------------|-----------------------------|-------------------------|
| 23760 |       | ED 5B 7B 5C | ORG 23760<br>LD DE, (23675) | ; Адрес первого символа |

|       |         |                     |               |                                 |
|-------|---------|---------------------|---------------|---------------------------------|
| 23764 |         | 21 42 5D            | LD HL, DATA 3 | ; графики пользователя.         |
| 23767 |         | 01 08 00            | LD BC, 8      | ; Конструкция символа UDG.      |
| 23770 |         | ED B0               | LDIR          | ; 8 байтов в символе.           |
| 23772 |         | 3E 02               | LD A, 2       | ; Установка символа UDG-"A"     |
| 23774 |         | CD 01 16            | CALL 5633     | ; Канал экрана - 2.             |
|       |         |                     |               | ; Открываем канал печати на     |
|       |         |                     |               | ; экран.                        |
| 23777 |         | CD 6B 0D            | CALL 3435     | ; Очистка экрана.               |
| 23780 |         | 3E 02               | LD A, 2       | ; См. выше.                     |
| 23782 |         | CD 01 16            | CALL 5633     |                                 |
| 23785 |         | 11 2E 5D            | LD DE, DATA1  | ; Данные для печати - DATA1.    |
| 23788 |         | 01 10 00            | LD BC, 16     | ; Символов - 16.                |
| 23791 |         | CD 3C 20            | CALL 8252     | ; Вызов подпрограммы ПЗУ для    |
|       |         |                     |               | ; печати символьной строки.     |
| 23794 |         | FD 36 57 03         | LD (IY+87), 3 | ; (IY+87) указывает на          |
|       |         |                     |               | ; системную переменную PFLAG    |
|       |         |                     |               | ; (23697). Установка в ней      |
|       |         |                     |               | ; числа 3 означает включение    |
|       |         |                     |               | ; режима INVERSE.               |
| 23798 |         | AF                  | XOR A         | ; Обнуление аккумулятора.       |
| 23799 |         | 32 40 5D            | LD (COL), A   | ; Обнуление столбца COL.        |
| 23802 | L1      | 11 3E 5D            | LD DE, DATA2  | ; Данные для печати - DATA2.    |
| 23805 |         | 01 04 00            | LD BC, 4      | ; Символов - 4.                 |
| 23808 |         | CD 3C 20            | CALL 8252     | ; Печать строки DATA2.          |
| 23811 | DELAY   | 21 FF FF            | LD HL, 65535  | Организация цикла               |
| 23814 | L2      | 2B                  | DEC HL        | задержки для того, чтобы        |
| 23815 |         | 7C                  | LD A, H       | изменения на экране не          |
| 23816 |         | B5                  | OR L          | происходили слишком             |
| 23817 |         | 20 FB               | JR NZ, L2     | быстро.                         |
| 23819 |         | 11 3E 5D            | LD DE, DATA2  | ; Данные для печати - DATA2.    |
| 23822 |         | 01 04 00            | LD BC, 4      | ; Символов - 4.                 |
| 23825 |         | CD 3C 20            | CALL 8252     | ; Печать строки DATA2.          |
| 23828 |         | 3A 40 5D            | LD A, (COL)   | ; Номер столбца позиции печати. |
| 23831 |         | 3C                  | INC A         | ; Переход к соседней позиции.   |
| 23832 |         | 32 40 5D            | LD (COL), A   | ; Запомнили новую позицию.      |
| 23835 |         | ED 4B 3F 5D         | LD BC, (LINE) | ; Координаты позиции печати.    |
| 23839 |         | CD 83 25            | CALL 9603     | ; Вызов процедуры ПЗУ для опре- |
|       |         |                     |               | ; деления атрибутов в текущей   |
|       |         |                     |               | ; координате.                   |
| 23842 |         | CD D5 2D            | CALL 11733    | ; Перенос результата со стека   |
| 23845 |         | FE 8F               | CP 143        | ; калькулятора в регистр A.     |
| 23847 |         | 20 D1               | JR NZ, L1     | ; Проверка на PAPER 1: INK 7:   |
|       |         |                     |               | ; FLASH 1.                      |
| 23849 |         | FD 36 57 00         | LD (IY+87), 0 | ; Возврат, если нужно знако-    |
| 23853 |         | C9                  | RET           | ; место еще не найдено.         |
| 23854 | DATA1   | DEFB 22 0 20 17     |               | ; Выключение режима INVERSE.    |
|       |         | 1 16 7 18           |               | ; Возврат в БЕЙСИК.             |
|       |         | 1 144 18 0          |               | ; Эти данные эквивалентны:      |
|       |         | 17 7 16 0           |               | ; ...AT 0, 20; PAPER 1; INK 7;  |
|       |         |                     |               | ; FLASH 1; "A"; PAPER 0; INK 7; |
|       |         |                     |               | ; FLASH 0. Здесь "A" - это сим- |
|       |         |                     |               | ; вол UDG; размещенный на кла-  |
|       |         |                     |               | ; више A, т.е. CHR\$(144).      |
| 23870 | DATA2   | DEFB 22             |               | ; Эти данные эквивалентны:      |
| 23871 | LINE    | DEFB 0              |               | ; ...AT 0, COL; CHR\$(62).      |
| 23872 | COL     | DEFB 0              |               | ; Символ 62 - это ">", он и     |
| 23873 | DEFB 62 |                     |               | ; изображает "ракету".          |
| 23874 | DATA3   | DEFB 24 153 126 153 |               | ; Это данные для символа UDG,   |
|       |         | 24 36 36 102        |               | ; изображающего человечка.      |

## 8.2 SCREEN\$

Точно так же, как и при вызове процедуры ATTR, перед вызовом этой процедуры необходимо в регистровую пару BC записать координаты символьной ячейки. Процедура

SCREEN\$ находится по адресу 9526. Параметры символа исследуемой символьной ячейки при вызове этой процедуры будут записываться на калькуляторный стек в пятибайтной форме. Использование калькуляторного стека для хранения параметров символа для Вас, наверное, открытие, поскольку до сих пор речь шла о хранении на стеке только чисел в пятибайтной форме.

Коды калькулятора при этом оперируют не с самими символами, а с их параметрами, записанными в пятибайтной форме. Для того, чтобы передать параметры символа на стек, необходимо записать в регистровую пару BC размер символа в байтах, а в регистровую пару DE - начальный адрес участка памяти, где хранится вся информация о символе. Для стандартных символов в регистр A записывается 0.

Для передачи этих параметров на стек используется процедура, находящаяся в ПЗУ по адресу 10929. Процедура же по адресу 11249 выполняет обратную функцию - передает параметры символа со стека в соответствующие регистры A, B, C, D, E. При этом размер символа будет записан в регистровую пару BC, а начальный адрес участка памяти с информацией о символе - в регистровую пару DE.

Программа 8.2 показывает, как можно использовать эти процедуры. После выполнения программы, на экране появятся два символа @, соответственно в позициях 0,20 и 0,21. Вместо вызова процедуры PRINT STRING (адрес 6252), можно использовать команду LD A,(DE) для проверки значения символа в позиции 0,20, а затем выполнить необходимые действия.

Листинг 8.2.

| АДРЕС | МЕТКА | МАШ. КОД | АССЕМБЛЕР  | КОММЕНТАРИЙ                                                                                                                                   |
|-------|-------|----------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
|       |       |          | ORG 23750  |                                                                                                                                               |
| 23760 |       | 3E 02    | LD A, 2    | ; Канал экрана - 2.                                                                                                                           |
| 23762 |       | CD 01 16 | CALL 5633  | ; Открываем канал печати на экран.                                                                                                            |
| 23765 |       | CD 6B 0D | CALL 3435  | ; Очистка экрана.                                                                                                                             |
| 23768 |       | 3E 02    | LD A, 2    | ; Канал экрана - 2.                                                                                                                           |
| 23770 |       | CD 01 16 | CALL 5633  | ; Открываем канал печати на экран.                                                                                                            |
| 23773 |       | 3E 16    | LD A, 22   |                                                                                                                                               |
| 23775 |       | D7       | RST 16     |                                                                                                                                               |
| 23776 |       | 3E 0     | LD A, 0    | Аналог ...AT 0,20; "@"                                                                                                                        |
| 23778 |       | D7       | RST 16     |                                                                                                                                               |
| 23779 |       | 3E 14    | LD A, 20   |                                                                                                                                               |
| 23781 |       | D7       | RST 16     |                                                                                                                                               |
| 23782 |       | 3E 40    | LD A, 64   |                                                                                                                                               |
| 23734 |       | D7       | RST 16     |                                                                                                                                               |
| 23785 |       | 06 14    | LD B, 20   | ; Номер экранного столбца.                                                                                                                    |
| 23787 |       | 0E 00    | LD C, 0    | ; Номер строки экрана.                                                                                                                        |
| 23789 |       | CD 38 25 | CALL 9528  | ; Вызов процедуры ПЗУ для определения того, что находится в данном знакоместе.                                                                |
| 23792 |       | CD F1 2B | CALL 11249 | ; Передача параметров символа со стека калькулятора в регистры процессора.                                                                    |
| 23795 |       | CD 3C 20 | CALL 8252  | ; Печать той символьной строки, на местоположение которой указывает пара DE, а длина которой находится в BC. В нашем случае - это символ "@". |
| 23798 | C9    |          | RET        |                                                                                                                                               |

При использовании БЕЙСИК-оператора SCREEN\$ имеет место одно ограничение. Код распознаваемого символа на экране должен быть в диапазоне от 32 до 127. В противном случае считается, что это пустой символ. Такое ограничение существенно для игровых программ, где используются UDG, распознать которые в данном случае не удастся. Для программ в машинных кодах такой проблемы не существует, поскольку UDG можно включить в стандартный символьный набор. Постоянно этот набор хранится в ПЗУ, начиная

с адреса 15360. Однако начальный адрес этого набора можно менять, меняя значение системной переменной CHARS (23606/7). Она указывает на 256 байтов ниже, чем начало символьного набора. Если переписать символьный набор из ПЗУ в ОЗУ таким образом, чтобы в этот набор вышел участок с информацией об UDG и чтобы UDG "A" соответствовал код 32, UDG "B" - код 33 и т.д., то можно вызывать процедуру SCREEN\$ для идентификации UDG на экране.

Программа 8.3 работает точно так же, как и программа 8.1, но эта программа способна распознавать на экране символ, код которого равен 144. Выполнив копирование стандартного символьного набора в ОЗУ и переписав значение символьной переменной CHARS можно совсем отказаться от UDG. Теперь можно часть стандартных символов (например, строчные буквы и некоторые знаки пунктуации) заменить на символы, форму которых можем задать сами (это будет около 50 символов). Процедура SCREEN\$ будет легко распознавать на экране любой символ, если он входит в символьный набор. И еще одно важное замечание. Необходимо всегда иметь свободный участок памяти, используемый как рабочий для стека. Если такого участка нет или он есть, но периодически не очищается, то при выполнении программы может появиться сообщение о переполнении памяти. Поэтому перед каждым вызовом процедуры SCREEN\$ необходимо очищать специально зарезервированный рабочий участок.

Листинг 8.3.

| АДРЕС | МЕТКА | МАСШ. КОД   | АССЕМБЛЕР      | КОММЕНТАРИЙ                    |
|-------|-------|-------------|----------------|--------------------------------|
| 23760 |       |             |                |                                |
| ..... | СМ.   | ЛИСТИНГ 8.1 |                |                                |
|       |       |             | ORG 23760      |                                |
| 23833 |       |             |                |                                |
| 23836 | 2A    | 7B 5C       | LD HL, (23675) | ; Адрес первого символа UDG.   |
| 23839 | 25    |             | DEC H          | ; Отступили на 256 байтов.     |
|       |       |             |                | ; Мы помним, что CHARS указы-  |
|       |       |             |                | ; вает на 256 байтов ниже, чем |
|       |       |             |                | ; начало шрифта.               |
| 23840 | 22    | 36 5C       | LD (23606), HL | ; Переустановили CHARS так,    |
|       |       |             |                | ; она теперь указывает на      |
|       |       |             |                | ; область UDG.                 |
| 23843 | CD    | 38 25       | CALL 9528      | ; Вызов процедуры SCREEN\$.    |
| 23846 | CD    | F1 2B       | CALL 11249     | ; Передача параметров симво-   |
|       |       |             |                | ; ла со стека в регистры.      |
| 23849 | 1A    |             | LD A, (DE)     | ; Проверка номера символа      |
|       |       |             |                | ; в текущем знакоместе.        |
| 23850 | FE    | 20          | CP 32          | ; Проверка "не пробел ли это?" |
| 23852 | 21    | 00 3C       | LD HL, 15360   | ; Расположение стандартного    |
|       |       |             |                | ; шрифта минус 256 б.          |
| 23855 | 22    | 36 5C       | LD (23606), HL | ; Восстановили стандартное     |
|       |       |             |                | ; значение CHARS.              |
| 23858 |       |             |                |                                |

..... Далее, как в программе 8. 1

Процедура очистки рабочего участка имеется в ПЗУ по адресу 5823. Вызов этой процедуры в программе 8.3 можно производить после команды LD A,(DE), однако предварительно необходимо сохранить в памяти последнее значение регистра A.

### 8.3 POINT

Оператор POINT x,y в БЕЙСИК-программах выдает 1, если пиксел экрана с координатами x,y включен и выдает 0, если он выключен. Процедура ПЗУ, соответствующая этому оператору (адрес 8910), может быть вызвана, если предварительно записать в регистр В у-координату, а в регистр С, соответственно х координату пиксела. При этом результат зашлется на стек, а т.к. он равен 0 или 1, то мы можем переписать его в регистр А и сравнить с заданным значением. В программе 8.4 показан способ проверки пиксела верхнего левого угла (координаты 0,175).

Если в начале в символьную ячейку с координатами 0,0 занести черный квадрат, то

результат работы процедуры будет равен 1, т.к. пиксел с координатами 0,175 входит в этот квадрат.

#### Листинг 8.4.

| АДРЕС | МЕТКА | МАШ. КОД | АССЕМБЛЕР  | КОММЕНТАРИЙ                                            |
|-------|-------|----------|------------|--------------------------------------------------------|
|       |       |          | ORG 23760  |                                                        |
| 23760 |       | 3E 02    | LD A, 2    | ; Канал экрана - 2.                                    |
| 23752 |       | CD 01 16 | CALL 5633  | ; Открываем канал печати на экран.                     |
| 23765 |       | CD 6B 0D | CALL 3435  | ; Очистка экрана.                                      |
| 23768 |       | 3E 02    | LD A, 2    | ; Канал экрана - 2.                                    |
| 23770 |       | CD 01 16 | CALL 5633  | ; Открываем канал печати на экран.                     |
| 23773 |       | 3E 8F    | LD A, 143  | ; "Черный квадрат".                                    |
| 23775 |       | D7       | RST 16     | ; Печать квадрата.                                     |
| 23776 |       | 0E 00    | LD C, 0    | ; Координата x.                                        |
| 23778 |       | 06 AF    | LD B, 175  | ; Координата y.                                        |
| 23780 |       | CD CE 22 | CALL 8910  | ; Вызов процедуры POINT из ПЗУ.                        |
| 23783 |       | CD D5 2D | CALL 11733 | ; Перенос результата со стека калькулятора в регистры. |
| 23786 |       | FE 01    | CP 1       | ; Проверка включен пиксел или нет.                     |
| 23788 |       | C8       | RET Z      | ; Выход, если да.                                      |

## 9. ПРИНТЕР

Не так уж много существует программ, для работы которых необходим принтер. Всегда можно ограничиться для вывода информации только экраном дисплея. Но иногда, например, необходимо показать свой уровень, достигнутый в какой-нибудь игре друзьям или знакомым. В этом случае нужно сделать копию изображения экрана на принтере. Поэтому в этой главе мы рассмотрим три BASIC-команды: COPY, LPRINT, LLIST и их аналоги в машинных кодах, обеспечивающих работу с принтером.

### 9.1. COPY

При вызове этой процедуры только двадцать две строки экрана передаются на принтер. Программа в машинных кодах позволит Вам улучшить эту процедуру так, чтобы можно было передавать на принтер произвольное число строк, причем, начиная с любой строки на экране, например, передать на принтер три строки, начиная с седьмой.

В программе 9.1 показан способ передачи на принтер всех 24 строк экрана, поскольку для некоторых игр достигнутый уровень показан в нижних строках экрана и при обычном вызове процедуры COPY получить необходимую информацию с двух последних строк не удастся.

#### Листинг 9.1

|       |          |              |
|-------|----------|--------------|
|       |          | ORG 23760    |
| 23760 | 21 00 40 | LD HL, 16384 |
| 23763 | 06 C0    | LD B, 192    |
| 23765 | F3       | DI           |
| 23756 | CD B2 0E | CALL 3762    |
| 23769 | C9       | RET          |

Как Вы заметили, в этой программе перед вызовом процедуры, находящейся в ПЗУ по адресу 3762, необходимо в регистровую пару HL записать начальный адрес экранной области ОЗУ (т.к. нам необходима копия, начиная с первой строки, то этот адрес 16384), а в регистр B - число копируемых строк, умноженное на 8.

### 9.2 LPRINT

Эта команда имеет сходство с командой PRINT. Различие состоит в том, что не работают по этой команде управляющие коды и, конечно, используется другой канал для вывода информации на принтер, который необходимо открыть в программе в машинных

кодах, прежде чем использовать команду RST 16. Например, чтобы вывести на принтер литеру "A", т.е. реализовать в машинных кодах LPRINT "A", необходимо первоначально открыть канал 3, затем записать в регистр A код литеры "A", ну а затем использовать команду RST 16.

Программа 9.2 демонстрирует этот способ для вывода на принтер символьной строки.

#### Листинг 9.2

```

                                ORG 23760
23760      3E 03      LD A,3
23762      CD 01 16  CALL 5633
23765      11 DF 5C  LD DE,DATA
23768      01 09 00  LD BC,9
23771      CD 3C 20  CALL 8252
23774      C9      RET
23775 DATA DEFM PROGRAM 2
```

Для выполнения табуляции выводимой на принтер информации, необходимо предварительно в системную переменную PRCC (23680) через POKE записать соответствующее число.

### 9.3 LLIST

Это третья БЕЙСИК-команда для работы с принтером. Реализовать ее в машинных кодах довольно просто. Достаточно только записать CALL 6133 и будет выполняться листинг строк BASIC-программы на принтере.

# СТРАНИЦА iS-DOS

(C) Самыловский С.В., 1993.

(C) SLOT Co.Ltd., Moscow, 1993.

Новинка сезона - дисковая операционная система для SPECTRUM "iS-DOS"

Еще несколько месяцев назад это казалось невероятным. Разрыв между желанием иметь дома профессиональный персональный компьютер и возможностью купить его становился все больше и больше. Красавцы PC достойно заняли свое место в институтах, банках, офисах и пока очень редко перешагивают порог нашего дома. Да и нужен ли каждому из нас дома, в школьном классе, в офисе небольшой частной фирмы супер-компьютер типа IBM PC, стоимостью более миллиона рублей? Зададим себе вопрос: что бы мы хотели поручить своему электронному помощнику? При всем кажущемся многообразии ответов на данный вопрос, все они могут быть сведены к следующим:

- игры и развлечения для детей и взрослых;
- подготовка, хранение и печать текстов и документов;
- выполнение текущих расчетов, решение конкретных задач;
- обучение, тренировка, управление какими-либо устройствами;
- профессиональное использование компьютера, составление программ на различных языках программирования, их отладка и т.п.;
- коммуникационные функции - домашний секретарь, телефонный автоответчик, прием и передача текстовых и графических документов по телефонным линиям связи.

Итак, сможет ли каждый из нас в обозримом будущем иметь у себя дома такой компьютер? Посмотрите внимательно на свой SPECTRUM! Если у него предусмотрена возможность подключения накопителя на гибком магнитном диске (дисковода) и печатающего устройства (принтера), то уже завтра Вы можете стать обладателем вычислительной системы, которая решит большинство Ваших домашних проблем, поможет в учебе и работе, позволит в спокойной домашней обстановке приобщиться к такому загадочному и такому интересному миру настоящих компьютерных технологий! Это маленькое чудо сделает для Вас и Вашего SPECTRUM-совместимого компьютера новая дисковая операционная система iS-DOS!

О чем собственно речь?

Речь идет об открытой многокаталоговой операционной системе iS-DOS, результате многолетнего труда коллектива разработчиков из Санкт-Петербургской фирмы ISKRASOFT, под руководством Ильяшенко Э.Н. Само название операционной системы iS-DOS - это сокращение от IskraSoft Disk Operating System. Оно, как и товарный знак системы закреплено законодательно за данной фирмой и без ее согласия не может быть использовано кем либо в коммерческих и рекламных целях.

На Российском рынке iS-DOS появилась в конце 1992 года и с этого момента уже сумела завоевать многочисленные симпатии пользователей SPECTRUM-совместимых компьютеров. Подробнее об истории создания системы и основных идеях, положенных в ее основу мы расскажем на страницах первых выпусков "ZX-РЕВЮ-94".

Компьютер, который мы выбираем.

Важнейшим достоинством системы iS-DOS является ее высокая мобильность и "неприхотливость" в требованиях к компьютеру. Она будет работать на любом SPECTRUM-



совместимом компьютере, имеющем в своем составе хотя бы один накопитель на гибком магнитном диске (дискет). Для работы iS-DOS требуется всего 48 Кбайт оперативной памяти. Остальная имеющаяся в компьютере оперативная память может быть использована для буферизации ввода-вывода или создания электронного диска, что примерно на порядок может повысить быстродействие работающих программ. iS-DOS обеспечивает использование до 4-х накопителей на гибких магнитных дисках, электронного диска и одного жесткого диска типа винчестер одновременно. Работа с различными клавиатурами от обычной 40-клавишной до XT-клавиатуры поддерживается набором специальных драйверов клавиатуры.

К Вашему компьютеру подключен принтер? В богатом перечне драйверов принтеров Вы наверняка быстро найдете нужный. Если принтер обеспечивает печать символов псевдографики, то высокое качество печатаемых документов, таблиц, графиков Вам обеспечено. Если принтер старенький то iS-DOS учтет и это!

Имея "ZX-модем" в составе компьютера и установив iS-DOS, Вы легко сможете связаться по телефону со своим коллегой и передать ему несколько файлов, а если Ваш компьютер обеспечивает последовательный интерфейс RS-232, то подключив к нему стандартный для IBM PC модем, Вы, с помощью специальной коммуникационной программы сможете выйти в настоящие компьютерные сети.

Работа с цветным монитором несомненно доставит Вам удовольствие от общения с системой. Если Ваш компьютер подключен к цветному или черно-белому телевизору, не переживайте! Вы можете работать в режиме 41 или 61 символ в строке, а возможность настройки цвета используемых программ сделает работу с компьютером максимально удобной.

Итак, все зависит только от наших возможностей, а начать можно и с малого! В любом случае возникает вопрос:

Так где же взять iS-DOS систему?

Кому-нибудь это может показаться странным, но iS-DOS придется купить. Переписать ее у товарища не удастся, так как она неплохо защищена от копирования и в лучшем случае "не захочет" загрузиться. Не лучшим вариантом будет и покупка системы с рук у знакомого или (что гораздо хуже) у незнакомого умельца хэккера. Ряд интересных программ вероятнее всего не смогут работать корректно или вообще не будут запускаться, а это так обидно...

Официально приобрести iS-DOS можно у самой фирмы ISKRASOFT в С. Петербурге, у ее Генерального дистрибьютора - московского предприятия "Slot" Co.,Ltd или у их официальных дилеров в Вашем городе или области. Заказав систему, можно также получить каталог прикладных программ и литературы по iS-DOS почтой по адресам:

117330 Москва, а/я 707, фирма "Slot"

или

121019 Москва, а/я 16, фирма "ИНФОРКОМ".

В Москве iS-DOS, различные прикладные программы для нее и документацию Вам предложит фирма "Slot", контактный телефон (095) 143-11-91, в следующих местах:

- на корпункте "ИНФОРКОМА" по адресу: Москва, Новый Арбат д.2, 19-е отделение связи (1-й этаж операционного зала);
- в здании Политехнического музея в отделе "Вычислительная техника" по адресу: Москва, Новая пл., д.3/4 п.1, 3-й эт.
- на московском радиорынке в выходные дни у официальных дилеров фирмы "Slot".

Система и все прикладные программы к ней доставляются на качественных дискетах, в фирменной упаковке фирмы "Slot", со специальной фирменной маркировкой и комплектуются кратким описанием покупаемого продукта, необходимого для начала работы HELPa и "Регистрационной карточки пользователя" которая дает ее обладателю право на 50% скидку при покупке новой версии программы и бесплатное использование "горячей

линии помощи".

Зарегистрированным пользователям фирма в первую очередь предоставляет информацию о новых поступлениях программ и документации, проводит с ними индивидуальные консультации, организует обучение и семинары по обмену опытом.

### **Первое знакомство с iS-DOS.**

iS-DOS выгодно отличается от существующие дисковых операционных систем для компьютеров семейства "ZX-SPECTRUM" и обладает более широкими возможностями, чем, например, традиционная TRDOS. При разработке ОС iS-DOS использованы стиль и идеология известной операционной системы MS-DOS и оболочки "Norton Commander" для ЭВМ IBM PC, а также стремление к максимальному удобству для пользователя. Такой подход к созданию системы обладает определенным преимуществом - пользователь, работающий на ПЭВМ типа "ZX-SPECTRUM" с iS-DOS при переходе на IBM PC попадает в знакомую, привычную уже среду "NORTON COMMANDER".

Операционная система iS-DOS позволяет работать с разнообразными внешними устройствами, такими, как электронный диск, дисковод 5.25 дюйма, винчестер, модем и локальная сеть, принтеры различных типов.

Полноценный текстовый редактор "Editor", программы печати текстовых и графических файлов, поддержка модема превращают обыкновенный "ZX-Spectrum" в полноценное рабочее место современного делового человека. iS-DOS значительно расширяет возможности компьютеров типа "ZX-Spectrum" для использования их в научной деятельности, в сфере бизнеса и управления, для создания прикладных и коммерческих программных продуктов.

iS-DOS ориентирована на пользователя, не обладающего навыками программирования, но не оставит равнодушным и профессионального программиста. И первое, с чем Вам предстоит познакомиться - это файловая оболочка системы.

### **ФАЙЛОВАЯ ОБОЛОЧКА. ( SHELL )**

Файловая оболочка iS-DOS предназначена для хранения, просмотра, редактирования и запуска файлов. В нее Вы попадаете после запуска операционной системы. Для этого необходимо вставить дискету с ос iS-DCS в дисковод и нажать клавишу <Reset>, т.е. перезапустить компьютер. Загрузчик iS-DOS перехватывает обращение TRDOS к диску и передает управление iS-DOS. После загрузки Вы увидите две панели с содержимым Вашего диска, а верхняя строка показывает системные утилиты, которыми Вы можете пользоваться при работе в оболочке. Эти утилиты вызываются цифровыми клавишами <1>,...,<0> соответственно и выполняют следующие функция:

|        |                                                                     |
|--------|---------------------------------------------------------------------|
| HELP   | - справочная информация.                                            |
| USER   | - вызов пользовательского меню.                                     |
| VIEW   | - просмотр файла.                                                   |
| EDIT   | - вызов текстового редактора.                                       |
| COPY   | - копирование файлов.                                               |
| RENAME | - переименование файлов, каталогов.                                 |
| crDIR  | - создание подкаталога.                                             |
| DELETE | - удаление файлов, каталогов.                                       |
| MENU   | - системное меню.                                                   |
| MASK   | - установка имен, расширений, параметров, файлов для отображения на |

панели.

На верхней строке правой и левой панели указано текущее устройство и имя диска или подкаталога. На правой и левой панели - содержимое главного каталога. Чтобы оказаться в нужном Вам подкаталоге подведите курсор к нему и нажмите <Enter>. Движение курсора осуществляется клавишами:

<A> - вниз

<Q> - вверх  
<O> - на левую панель  
<P> - на правую панель

Здесь работают также стандартные клавиши управления курсором.

<CS/Q> - в начало каталога.  
<CS/A> - в конец каталога.  
<CS/O> - в корневой каталог левой панели.  
<CS/P> - в корневой каталог правой панели.  
<SS/CS>- обмен панелей.

Выбор нового устройства:

<CS/1>- на левой панели.  
<CS/2>- на правой панели.

При выборе нового устройства появится меню:

|               |   |   |   |   |   |  |
|---------------|---|---|---|---|---|--|
| Choose Drive: |   |   |   |   |   |  |
| A             | B | C | D | E | F |  |
|               |   |   |   |   |   |  |

Управляя курсором клавишами <O>, <P> можно выбрать новое устройство с помощью нажатия <Enter>. Отказаться от выбора и выйти в оболочку можно с помощью <SS/A> или <Space>. Подсказки по программам iS-DOS вызываются с помощью клавиши <1>. Для этого необходимо подвести курсор к интересующей Вас программе и нажать <1>. Для просмотра текстового файла подведите курсор к нему и нажмите клавишу <3>. Для управления оболочкой задействованы два типа клавиш - зарезервированные для оболочки и переопределяемые пользователем.

К первому типу относятся: клавиши <O>, <P>, <A>, <Q> и их комбинации с <CS>, клавиш <5>, <6>, <7>, <8>, <9> и комбинации <CS/1> <CS/2>, а также <Enter> и <Space>. Клавиши <O>, <P>, <A>, <Q>, а также <5>, <6>, <7>, <8> в сочетании с <CS> - управляют движением курсора (такое управление является стандартным для iS-DOS).

Второй тип клавиш и их функции заданы в файле extkey.txt, в котором Вы можете сами определить функции клавиш так, как Вам удобно. По умолчанию поддерживаются следующие определения (один из вариантов):

c - сравнение файлов на панелях;  
C - калькулятор;  
SS+c - копировщик каталогов;  
d - запрос даты;  
D - вывод данных о файле на экран;  
e - просмотр и удаление резидентов;  
f - поиск файла;  
F - копирование файла с сегментацией;  
h - распечатка текста или картинки;  
l - свободное место на диске;  
m - монитор командной строки;  
M - резидентный монитор ком. строки (необходимо сначала установить);  
SS+k - маркировать все файлы в каталоге (+);  
SS+j - отменить маркировку всех файлов (-);  
n - копирование файла с удалением (move);  
r - удаление целого каталога с файлами;  
S - вывод системной информации;  
s - сортировка по имени файла;  
SS+s - сортировка по расширению файла;

- t - вывод 'дерева' каталогов;
- u - восстановление удаленных файлов.

### Начальная загрузка системы.

При начальной загрузке операционная система ищет в корневом каталоге диска, с которого она загружена, файл `autoexec.bat`. Если этот файл будет найден, то он выполняется.

Файл `autoexec.bat` содержит команды, которые должны выполняться каждый раз при начальной загрузке системы. Эти команды осуществляют необходимую настройку операционной системы, устанавливают удобное для работы окружение, запрашивают текущую дату, устанавливают список каталогов в которых производится поиск выполняемых программ (команда `path`), устанавливают текущий каталог рабочей панели и выполняют другие полезные действия.

### ПОДПРОГРАММА ВЫЗОВА СПРАВОЧНОЙ ИНФОРМАЦИИ (ПОДСКАЗОК) ( HELP )

Программа вызова справочной информации в основном меню системы (фигурирует как " HELP ") вызывается клавишей <1>.

Для получения справочной или дополнительной информации по интересующей Вас программе найдите файл этой программы курсором и нажмите <1>. Что Вы увидите? Это зависит от содержания файла с именем интересующей Вас программы, но с расширением `.hlp`, находящегося в каталоге `HELP`. Если Вы увидите:

" Welcome to Shell for iS-DOS-92 ... "

то это означает, что в каталоге `HELP` отсутствует соответствующий файл с расширением `.hlp` и Вам предложена справка по оболочке.

### МЕНЮ ПОЛЬЗОВАТЕЛЯ ( USER )

Программа предназначена для удобства работы пользователя в `SHELL` (аналог программирования реакции на нажатие клавиши с большими возможностями, удобна для использования при быстрых переходах из каталога в каталог, для часто вызываемых программ, для вызова прикладных пакетов и т.д.). Программа вызывается клавишей <2>.

Для работы программы необходим текстовый файл `menu.txt`. При вызове программа вначале ищет файл `menu.txt` в текущем каталоге, затем (при его отсутствии там) в каталоге `SHELL`, таким образом пользователь может иметь в каждом каталоге свое меню.

### ПРОГРАММА ПРОСМОТРА ТЕКСТОВЫХ ФАЙЛОВ ( VIEW )

Для просмотра текстового файла удобно воспользоваться функцией `VIEW`. Для этого подведите к нему курсор и назовите клавишу <3>. На экране появится большое окно, в котором распечатается содержимое файла. По файлу можно двигаться с помощью клавиш управления курсором, а также в режиме непрерывного скроллинга. Можно переключить режим отображения текста на экран: 41/61 буква в строке.

### ТЕКСТОВЫЙ РЕДАКТОР ( EDIT )

Мощный и удобный в работе текстовый редактор `EDIT` несомненно придется по душе даже профессионалу. По своим возможностям он ненамного уступает знаменитому текстовому редактору `LEXICON` для `IBM PC`. Кроме обычных функций удаления, вставки и построчного редактирования в нем реализованы возможности контекстного поиска и замены, блочные операции над текстом, режим использования псевдографики, разбивка

текста на страницы, автоматическое или построчное форматирование текстовых абзацев с выполнением переносов слов по правилам орфографии, возможность одновременной работы с несколькими текстовыми файлами, работа с макросами.

Для вызова текстового редактора подведите курсор к файлу, который Вы желаете отредактировать, и нажмите клавишу <4>. Если тип выбранного Вами файла относится к допустимому, то редактор предложит Вам его отредактировать, создать новый файл, или отказаться и от того и от другого. Вам будет также предоставлена возможность копировать файл в резервный файл расширением .bak. В противном случае Вам будет предложено лишь создать новый файл.

### КОПИРОВАНИЕ ФАЙЛОВ ( COPY )

Удобно и понятно в системе реализована функция копирования файлов. Для того, чтобы скопировать файл, надо подвести к нему курсор и нажать клавишу <5>. Для копирования группы файлов их надо предварительно выделить с помощью клавиши пробел. Копирование всегда происходит с текущей панели на противоположную. Включение режима работы с буферизацией заметно ускорит выполнение больших операций копирования.

### РЕДАКТИРОВАНИЕ АТРИБУТОВ ФАЙЛА ИЛИ КАТАЛОГА ( RENAME )

Изменить имя файла или его расширение поможет Вам функция RENAME. Она работает с атрибутами как файла, так и каталога. Установив курсор на файл, с которым Вы желаете работать, нажмите клавишу <6>. При этом загорится 6-ий слева индикатор "RENAME" и возле выбранного Вами файла, мигает курсор строкового редактора. Изменив имя или расширение файла нажмите ENTER.

### СОЗДАНИЕ НОВОГО КАТАЛОГА ( MKDIR )

Для создания нового каталога (подкаталога) нажмите клавишу <7> MKDIR. В ответ на появившийся запрос введите имя создаваемого каталога.

### УДАЛЕНИЕ ФАЙЛОВ и КАТАЛОГОВ ( DELETE )

Удаление файлов или пустых каталогов выполняется клавишей <8>. Подведите курсор к удаляемому файлу и нажмите <8>. Для удаления группы файлов их надо предварительно выделить с помощью клавиши <пробел>. После ввода подтверждения файлы будут уничтожены.

### СИСТЕМНОЕ МЕНЮ ( MENU ) ВВОД МАСКИ-ШАБЛОНА ФАЙЛА. УСТАНОВКА РЕЖИМОВ ПЕЧАТИ КАТАЛОГА ( MASKA )

Две последние функции файловой оболочки MENU <9> и MASKA <0> помогут Вам настроить систему для быстрого поиска и удобной индикации в текущем каталоге файлов с необходимыми именами и ( или ) расширениями.

### РАБОТА С TR-DOS ДИСКАМИ В СРЕДЕ IS-DOS.

Серьезное внимание разработчики системы уделили вопросам копирования информации из TRDOS в iS-DOS и обратно. Копирование файлов с TRDOS-дисков выполняется программой FROM\_TRD и запускается клавишей <f> или из "USER"-меню. Текстовые файлы TLW с помощью программы FROM\_TLW после копирования можно преобразовать в формат редактора EDIT системы iS-DOS.

Копирование на TRDOS-диски выполняется программой TO\_TRDOS и вызывается клавишей <t> или из "USER"-меню. Отметив копируемые файлы пробелом или подведя

курсор к копируемому файлу нажмите <t>. Вам будет предложено меню, в котором можно отказаться от копирования, изменить текущие параметры копирования или выполнить его, нажав <ENTER>.

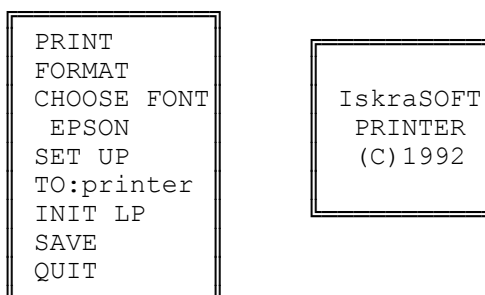
### РАБОТА С MS-DOS ДИСКАМИ В СРЕДЕ iS-DOS.

Работая в iS-DOS, Вы получаете возможность прочитать дискету в формате MS-DOS емкостью 360к, просмотреть каталоги и подкаталоги на ней, удалить файлы и каталоги, переименовать файлы и каталоги, создать новые каталоги, скопировать файл или группу файлов на диск iS-DOS и обратно, перекодировать скопированный из MS-DOS текстовый файл в формат редактора EDIT системы iS-DOS и выполнить обратную операцию.

Обращает на себя внимание отсутствие со стороны iS-DOS ограничения на размер обрабатываемых файлов MS-DOS. Это наверняка заинтересует тех, кто хотел бы результаты домашней работы перенести потом на более мощный компьютер для загрузки в базу данных или распечатки на лазерном принтере.

### ПРОГРАММА ПЕЧАТИ ТЕКСТОВЫХ ФАЙЛОВ.

Программа печати текстовых файлов в iS-DOS выполнена в виде отдельной программы PRINT, которая вместе с редактором EDIT позволит Вам получить качественный документ, используя для этого все возможности подключенного к компьютеру принтера. Для вызова программы печати (предварительно загрузив драйвер печати) установите курсор на текстовом файле и нажмите клавишу <H>. Загорится индикатор "VIEW", и Вы окажетесь в меню программы печати текстовых файлов iS-DOS:



До начала печати можно определить формат печатаемого документа, количество столбцов печати, необходимость отступов, нумерации листов, плотность печатаемых символов, качество шрифта и другие параметры. Для настройки на конкретный тип принтера имеется специальное меню, которое предлагает Вам на выбор: EPSON, MC6312, MC6313, MC6337. Если в этом перечне Вы не нашли подходящий для себя принтер, имеется возможность описать в специальном файле свой принтер самостоятельно.

### ПРОГРАММА ПЕЧАТИ ГРАФИЧЕСКИХ ИЗОБРАЖЕНИЙ И РИСУНКОВ ИЗ ФАЙЛОВ ТИПА \*.scr.

Программа печати графических изображений практически эмулирует возможности известного графического пакета STORY BOARD FOR MS-DOS. Возможность масштабирования изображения, режим ЛУПА, свободное размещение на листе, печать в полутонах полностью удовлетворяют Ваши потребности.

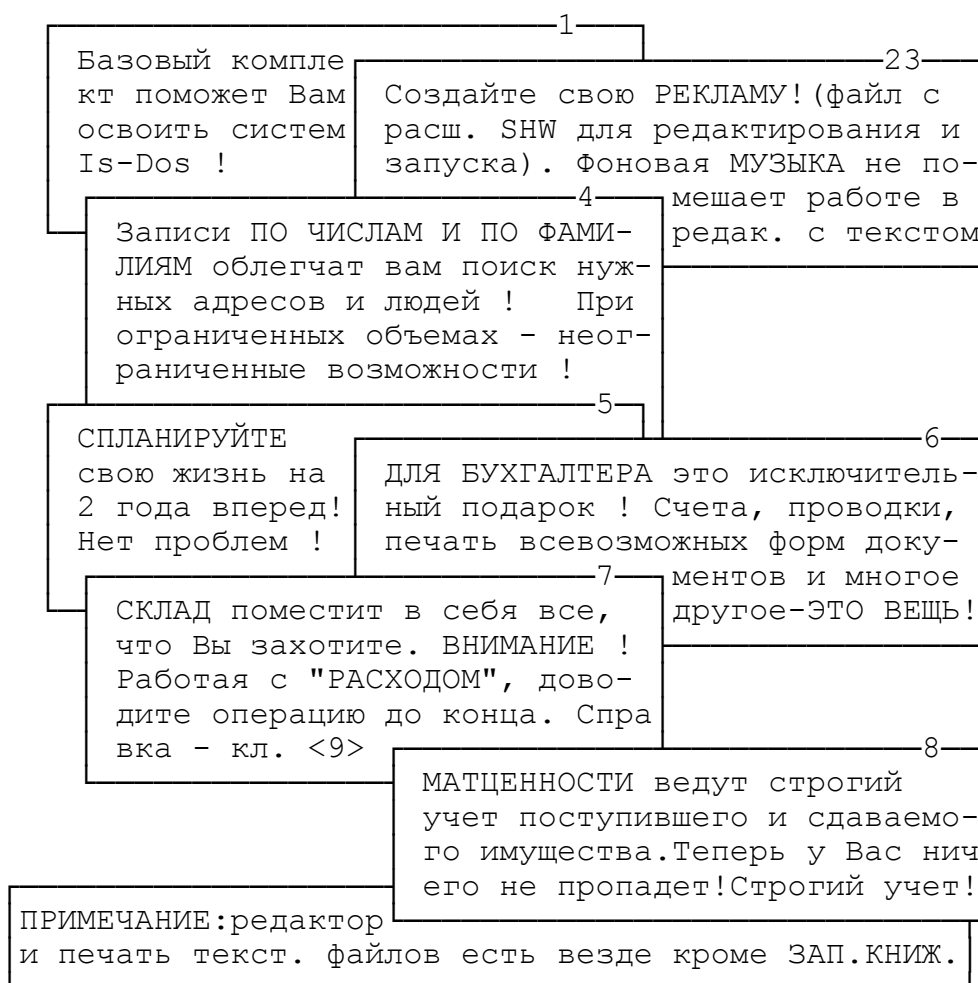
Перед запуском программы должен быть инсталлирован драйвер Вашего принтера. Допустимый тип печатаемых файлов \*.scr. При переброске графических файлов из TRDOS, не забывайте их переименовывать. Далее подведите курсор к файлу с расширением scr, из которого Вы желаете распечатать графические изображения и нажмите клавишу H.

Продолжение следует.

КОМПЛЕКТ ДИСКЕТ Is-Dos  
поставляемый фирмой "Slot" г. Москва

| N  | НАИМЕНОВАНИЕ                | СОДЕРЖАНИЕ                                                                                                                                                     |
|----|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 01 | БАЗОВЫЙ КОМПЛЕКТ            | ОС is-DOS; Утилиты; MS-Dos; Текстовый редактор; Программа печати текстовых файлов;                                                                             |
| 23 | PRINTLUX & MUSIC            | ОС is-Dos; Печать; Редактор;                                                                                                                                   |
|    | RECLAM SHOW                 | Программа печати экранных файлов + .SCR файлы; Пакет фонового музыкального сопровождения на базе AY-процес. Программа создания динамических рекламных роликов; |
| 04 | ЗАПИСНАЯ КНИЖКА             | ОС is-Dos; Пакет Записная книжка"                                                                                                                              |
| 05 | ДЕЛОВОЙ КАЛЕНДАРЬ           | ОС is-Dos; Печать; Редактор; Пакет "Деловой календарь"                                                                                                         |
| 06 | АРМ "ФИНАНСЫ БЕЗ ПРОБЛЕМ"   | ОС is-Dos; Печать; Редактор; Пакет"Финансы без проблем"                                                                                                        |
| 07 | АРМ "СКЛАД"                 | ОС is-Dos; Печать; Редактор; Пакет "Склад"; "Настройщик"                                                                                                       |
| 08 | АРМ "МАТЕРИАЛЬНЫЕ ЦЕННОСТИ" | ОС is-Dos; Печать; Редактор; Пакет "Матценности"                                                                                                               |

Distributed by SLOT Co.Ltd Moscow Tel:(095)143-11-91



При первом взгляде на то, о чем Вам сегодня рассказали, у пользователя возникает обычно два вопроса:

- НЕУЖЕЛИ ЭТО СДЕЛАНО У НАС?
- ПОЧЕМУ Я ОБ ЭТОМ ЕЩЕ НИЧЕГО НЕ ЗНАЮ?

То же произошло и с нами. Где-то начиная с июля месяца 1993 года в нашей почте все чаще стали проскакивать сообщения о том, что где-то кто-то что-то видел или слышал. Люди просили рассказать, что мы знаем об iS-DOS.

Комментарии, которыми сопровождалось подобные сообщения, обычно соответствовали тем, которыми сопровождают сообщения об НЛО. Основная суть: это невероятно, этого просто не может быть, потому что этого не может быть никогда.

Мы начали "раскручивать" этот вопрос. Сначала приобрели пару систем и проверили их в деле. Оказалось, что слухи не вполне правильно описывали действительность. **ДЕЙСТВИТЕЛЬНОСТЬ ОКАЗАЛАСЬ ЕЩЕ НЕВЕРОЯТНЕЕ.**

И сейчас мы делаем однозначный вывод: "Господа, пока Вы читаете эти строки, Ваш компьютер переживает второе рождение, а у Вас начинается новая жизнь! О том, что сделали Ваши коллеги, российские программисты, К. Синклер даже и не мечтал.

Да, это действительно сделано у нас в России! И мы еще расскажем на страницах "ZX-РЕВЮ" о том, как это было сделано и расскажем о тех людях, которые стояли у истоков новой системы.

Самое же потрясающее, что это действительно многолетний труд, который не вышел за стены лаборатории до тех пор, пока система не была полностью подготовлена и обкатана и пока она не "обросла" необходимым пакетом утилит, драйверов и прикладных программ, то есть всем тем, без чего ее нельзя было бы взять такую, как она есть и сразу использовать на полную мощность. Это уникальный пример грамотного, но очень трудного с инженерной и экономической точки зрения маркетинга. Вы только представьте, через что пришлось пройти разработчикам, которые несколько лет работали в поисках и сомнениях, не имея возможности ни оценить спрос на свой труд, ни получить за него минимальную компенсацию. И если коллектив выжил в таких условиях, довел свое дело до конца, то ему просто надо ставить памятник при жизни.

Из статьи руководителя фирмы "SLOT", Самыловского Сергея Владимировича Вы узнали самые первые сведения об iS-DOS. По должности автор говорит в этой статье о том, что сейчас есть реально и что может быть взято Вами немедленно.

У нас же руки в этом смысле развязаны и мы можем вместе с Вами, уважаемые читатели, заглянуть в ближайшее будущее.

Система открыта, а это значит, что она представляет неограниченный простор для творчества системных программистов, которые могут уже сейчас работать над ее развитием, над разработкой новых системных утилит и драйверов любых внешних устройств. Основное достоинство состоит в том, что каждый, кто способен сделать нечто значимое, сделает это не для корзины и не для узкого круга близких друзей, а для тысяч счастливых обладателей новейшей системы и его труд получит достойное моральное и материальное вознаграждение.

Программисты-прикладники, опирающиеся на возможности новой системы, получают необходимую поддержку от аппаратной среды (несколько дисководов, винчестер, средства коммуникации и т.д. и т.п.). Они смогут не просто работать, а работать "под систему" и быть уверенными, что их труд не останется невостребованным.

Поддержка из системы iS-DOS сетевых возможностей даст неограниченный простор для творчества всем желающим. Это и возможность работы с BBS и обмен информацией через модем. Возможно, уже в 1994 году в разных уголках страны появятся электронные журналы по "Синклеру". Каждый желающий сможет стать и автором и главным редактором и издателем.



Совместимость файлов с форматом MSDOS фактически превращает Ваш "Спектрум" в IBM-совместимую машину (на уровне системы). Работая дома, по сети Вы получите доступ к ресурсам IBM-компьютера, установленного у Вас на работе. Вы сможете по-новому организовать свой рабочий день и перейти к тому стилю, который уже давно практикуется в развитых странах.

Ресурсы IBM-совместимой техники смогут быть использованы на благо "Спектрума". Появятся редакторы и АССЕМБЛЕРЫ для "Спектрума", работающие на IBM. Открывается возможность создания программ КРОСС-ассемблеров для перевода машинного кода программ, написанных для одного компьютера на машинный код другого. Все это возможно, но требует больших ресурсов памяти и большого дискового пространства для размещения функциональных библиотек. Теперь эти возможности начнут приходить в Ваш дом.

Об уникальных возможностях использования "Спектрума" в качестве офисного оборудования мы уже и не говорим, они очевидны.

Новая система предоставляет необычные возможности для заводов-изготовителей компьютеров. В своих производственных планах они уже сейчас могут учесть включение в принципиальную схему ПК узлов, которые позволят взять от машины максимум возможного под управлением iS-DOS. Не исключено, что будут заключены лицензионные соглашения между заводами и авторами системы о включении в архитектуру компьютеров ПЗУ с системой iS-DOS.

Все это перспективы ближайших нескольких месяцев. А что же для этого делается сейчас?

Между фирмами "ИНФОРКОМ" и "SLOT" заключено соглашение о совместных действиях по продвижению системы iS-DOS на рынок СНГ. Отныне, начиная с января 1994 г. "ИНФОРКОМ" начинает эксклюзивное освещение новой системы. Принято решение о пятикратном увеличении тиражности "ZX-РЕВЮ" в 1994 году и о существенном улучшении его полиграфического исполнения. Мы надеемся, что все желающие получать это издание наконец-то смогут это сделать и не остаться за бортом стремительного распространения iS-DOS.

Программа действий согласована с авторами системы, фирмой Iskra-Soft (С.Петербург) и получила полную поддержку.

Предполагается, что по мере распространения системы и завоевания ею популярности, будет открываться и доступ к ее работе изнутри (для системных программистов). Со временем будут опубликованы точки входа в систему и закрытая часть информации о ее работе, короче говоря, распространение системы будет сопровождаться ее "открыванием" в расчете на то, что ее распространение будет поддержано дальнейшим ее развитием. Возможно, будет создана сертификационная группа для включения в систему разработок сторонних специалистов.

Подготовлена и сдана в печать первая книга об iS-DOS, выход из печати ожидается в конце ноября - начале декабря 1993 года. Книгу можно будет и приобрести как в фирме "SLOT", так и по каналам "ИНФОРКОМа".

Одним словом, всех нас ждут в 1994 году большие события. Настраивайтесь на серьезную работу, будущее обещает быть интересным.

Ваш "ИНФОРКОМ".

# ПРОФЕССИОНАЛЬНЫЙ ПОДХОД

(С) Збитнев В.А., г. Новосибирск, 1993г.

## КОМПЬЮТЕР И ЗВУК

Вы уже несколько знакомы с генерацией звука на Спектруме. Самый простой пример: оператор ВЕЕР в БЕЙСИКе. С остальными звуковыми эффектами Вы знакомы лишь понаслышке из игровых программ. Что представляет из себя звук на компьютере? На рис. 1 представлено классическое изображение гармонической частоты.

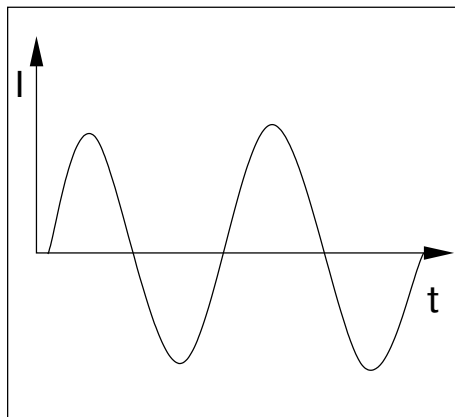


Рис. 1

На компьютере такая синусоида неформируема. Мы не сможем на "Спектруме" сформировать плавное изменение положения диффузора динамика, т.к. доступ к динамике возможен только через бит 4 (здесь четвертый бит является пятым по порядку или числом 16, если все остальные считать нулями) порта номер 254.

Положением динамика управляет один бит, значит возможны только два положения, где 0 - начальное положение, 1 - максимальное удаленное от начального положения. Графиком частоты, образованной таким образом, служит рис. 2.

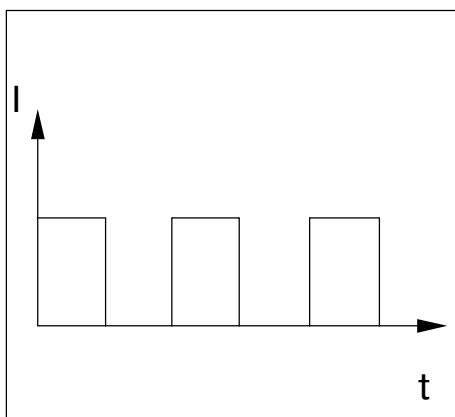


Рис. 2

Подобным образом информация считывается и записывается на магнитофон. Попробуйте набрать программу 1, в которой данные, считываемые с магнитофона, запоминаются в памяти. Информация может представлять собой человеческую речь, музыку и прочую звуковую информацию. Программа 2 проигрывает информацию, считанную с магнитофона и записанную в память, выводя ее на динамик компьютера.

В данном случае информация записывается в верхние две трети экрана с задержкой 15. Чем меньше эта задержка, тем быстрее заполнится память и более четко воспроизведение. Так что подбирайте ее по вкусу.

### Программа 1.

```
TAPBUF    LD    HL, 16384
          LD    DE, 4096
SBUF      LD    C, 1
LDBYT     LD    A, 15
DELAY1    DEC    A
          JR    NZ, DELAY1
          IN    A, (254)
          RLCA
          RLCA
          RL    C
          JR    NC, LDBYT
          LD    (HL), C
          INC    HL
          DEC    DE
          LD    A, D
          OR    E
          JP    NZ, SBUF
          RET
```

### Программа 2.

```
Bufdin    LD    HL, 16384
          LD    DE, 4096
STBYT     LD    A, (HL)
          LD    B, A
          LD    C, 128
PLAY      LD    A, 15
DELAY2    DEC    A
          JR    NZ, DELAY2
          LD    A, B
          AND    C
          JP    NZ, BEEP1
          XOR    A
          OUT    (254), A
BP1Q      RRC    C
          JR    NC, PLAY
          INC    HL
          DEC    DE
          LD    A, D
          OR    E
          JR    NZ, STBYT
          RET    BEEP1
          LD    A, 16
          OUT    (254), A
          JR    BP1Q
```

Но даже при минимальном значении задержки звук не будет достаточно реалистичным. Из рис. 3 видно, что выделяются не все частоты, многие из них, находясь ниже 0, приравниваются к 0, а находящиеся выше 0, приравниваются к 1.

Над данными, которые считываются с магнитофона, можно выполнять некоторые операции, например, удаление низкой частоты, подавление шума, выявление преобладающей частоты. Выявление частот используется в цветомузыке. Набрав программу 3, Вы можете узнать, сколько импульсов (точнее перепадов с 1 на 0 или с 0 на 1) произошло за некоторый интервал времени.

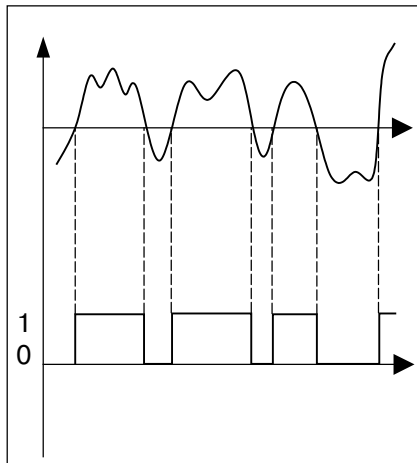


Рис. 3

Число импульсов возвращается в регистровой паре BC, т.е. Вы можете вызывать эту программу Бейсик-программой. По числу таких импульсов можно узнать, какая частота преобладает (в данном случае за 127 тактов). Чем выше частота, тем меньше значение, содержащееся в BC. Можно на основе этой программы создать цветомузыку или программу, распознающую мелодию, проигрываемую с магнитофона. Но эта программа не сможет Вам выдать точное значение частоты, идущей с магнитофона. Это значение весьма приближенно, т.к. часть частот пропадает, часть наоборот появляется в виде шума.

#### Программа 3.

```
BCDELAY  LD    B, 127
          LD    H, A
          LD    C, 0
IMPV      18    A, (254)
          XOR    H
          AND    64
          CALL   Z, INCC
          DJNZ   IMPV
          RET
INCC      LD    A, H
          CPL    LD
          H, A   INC
          C      RET
```

Если вы, к примеру, запустив программу 1, предварительно нажмете клавишу "воспроизведение" и "пауза" на магнитофоне, то увидите, что с магнитофона, который молчит, идет шум, который весьма заметен (вполне возможно, что у Вас прекрасный магнитофон, тогда шума не будет, но урезание частот останется).

Итак, Вы представляете уже, какое качество звука можно получить, используя Спектрум, какие недостатки имеет звуковое решение на компьютере.

Но возможен резонный вопрос: "Каким же образом регулируется громкость в мелодиях игровых программ?" Что же такое громкость? С помощью рис. 4 можно увидеть, что громкость - это амплитуда звуковой частоты. Но как же можно изменять амплитуду на Спектруме?

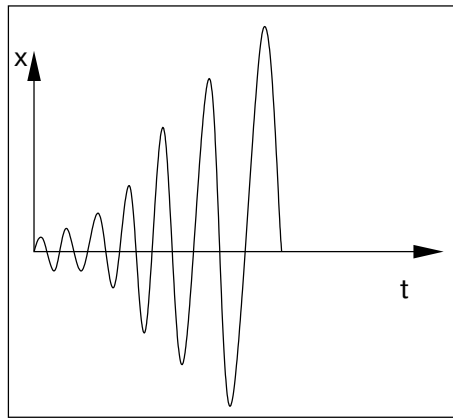


Рис. 4

Представьте себе, что на динамик подана 1. Динамик начинает поднимать диффузор. А потом тут же подадим на него 0. Диффузор опять примет начальное положение. А что, если мы будем увеличивать значение задержки между подачей 1 и 0. Диффузор поднимется уже несколько выше, и так можно увеличивать эту задержку до тех пор, пока за промежуток времени между 1 и 0 диффузор не будет успевать достичь наивысшего положения.

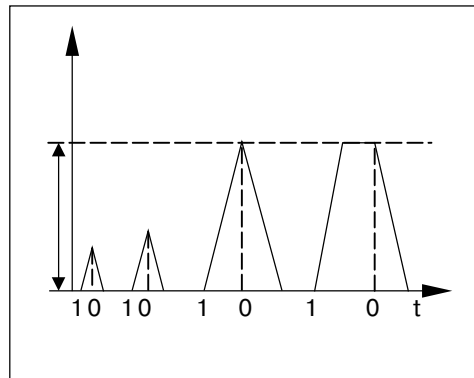


Рис. 5

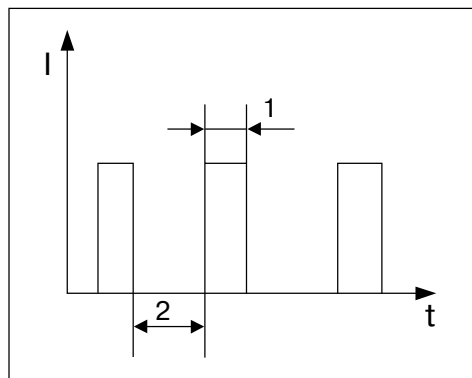
Если он будет успевать, то, как такового, изменения громкости происходить не будет. Вы, наверное, уже замечали, что в играх, в которых используется затухание, слышно, что при уменьшении громкости мелодии появляется высокая частота. Это происходит именно из-за этого эффекта. Подобные мелодии с затуханиями почти невозможно слушать на маленьких (высокочастотных) динамиках, так что наиболее благородно мелодия звучит на низкочастотных (больших) динамиках. Итак, мы выяснили, что для изменения громкости нужно изменять задержку между 1 и 0.

#### Программа 4.

| МЕТКА  | АССЕМБЛЕР      | КОММЕНТАРИЙ                          |
|--------|----------------|--------------------------------------|
|        | ORG 30000      |                                      |
|        | DI             | ; Отключение прерываний.             |
|        | LD IX, 31000   | ; Начало данных.                     |
| CONTIN | LD A, (IX+0)   | ; Прием данных в аккумулятор.        |
|        | CP 255         | ; 255 - конец мелодии.               |
|        | JP Z, EXI      | ; Выход.                             |
|        | OR A           | ; Проверка флагов в аккумуляторе.    |
|        | JR Z, N01      | ; Если ноль, то обход.               |
|        | LD C, A        | ; Временное сохранение аккумулятора. |
|        | AND 128        | ; Проверка старшего (7-го) бита.     |
|        |                | ; Если он =1, то                     |
|        | LD (BARB+1), A | ; включается признак ударника.       |
|        | LD A, C        | ; Восстановление аккумулятора.       |
|        | AND 127        | ; Гашение 7-го бита.                 |
|        | LD D, A        | ; Временное сохранение аккумулятора. |
|        | LD (NT1+1), A  | ; Установка звука в первом канале.   |
|        | SRL A          | ; Деление числа на                   |
|        | SRL A          | ; восемь для определения             |

|        |      |              |                                                |
|--------|------|--------------|------------------------------------------------|
|        | SRL  | A            | ; задержки громкости.                          |
|        | INC  | A            | ; Уход от возможного нуля.                     |
|        | LD   | (VL1+1), A   | ; Установили задержку громкости в              |
|        |      |              | ; первом канале.                               |
|        | LD   | A, 1         | ; Установка компенсационной задержки,          |
|        | LD   | (DEL1 +1), A | ; чтобы громкость не влияла на частоту.        |
| N01    | INC  | IX           | ; Переход к очередному байту данных            |
|        | LD   | A, (IX+0)    | ; (второй канал).                              |
|        | OR   | A            | ; Проверка флагов в аккумуляторе.              |
|        | JR   | Z, N02       | ; Обход, если 0 (играть не надо).              |
|        | LD   | E, A         | ; Временно сохранили аккумулятор.              |
|        | LD   | (NT2+1), A   | ; Установка звука во втором канале.            |
|        | SRL  | A            | ; Деление                                      |
|        | SRL  | A            | ; на                                           |
|        | SRL  | A            | ; восемь.                                      |
|        | INC  | A            | ; Плюс единица (уход от нуля).                 |
|        | LD   | (VL2+1), A   | ; Установили задержку громкости во             |
|        |      |              | ; втором канале.                               |
|        | LD   | A, 1         | ; Установка компенсационной                    |
|        | LD   | (DEL2+1), A  | ; задержки.                                    |
| N02    | INC  | IX           | ; Переход к очередному байту данных.           |
|        | LD   | C, 2         | ; Темп                                         |
| BEPS   | LD   | B, 0         | ; мелодии.                                     |
|        | CALL | BEEP         | ; Выдача сигнала на динамик.                   |
|        | CALL | BEEP         | ; Выдача сигнала на динамик.                   |
|        | CALL | DECVOL       | ; Уменьшение громкости.                        |
|        | DEC  | C            |                                                |
|        | JP   | NZ, BEPS     | ; 2 раза.                                      |
|        | JP   | CONTIN       | ; Возврат для приема очередных данных.         |
| ; BEEP |      |              |                                                |
|        |      |              | ; Регистр D - первый канал.                    |
|        |      |              | ; Регистр E - второй канал.                    |
|        |      |              | ; VL1 - громкость в первом канале.             |
|        |      |              | ; VL2 - громкость во втором канале.            |
|        |      |              | ; DEL1 - компенсационная задержка 1-го канала. |
|        |      |              | ; DEL2 - компенсационная задержка 2-го канала. |
| BEEP   | DEC  | D            |                                                |
|        | JR   | NZ, S2       |                                                |
| NT1    | LD   | D, 0         |                                                |
|        | LD   | A, 16        | первый                                         |
|        | OUT  | (254), A     |                                                |
| VL1    | LD   | A, 0         |                                                |
| S0     | DEC  | A            | канал                                          |
|        | JR   | NZ, S0       |                                                |
|        | OUT  | (254), A     |                                                |
| DEL1   | LD   | A, 32        |                                                |
| S1     | DEC  | A            |                                                |
|        | JR   | NZ, S1       |                                                |
| S2     | DEC  | E            |                                                |
|        | JR   | NZ, BARB     |                                                |
| NT2    | LD   | E, 0         |                                                |
|        | LD   | A, 16        | Второй                                         |
|        | OUT  | (254), A     |                                                |
| VL2    | LD   | A, 12        | канал                                          |
| S3     | DEC  | A            |                                                |
|        | JR   | NZ, S3       |                                                |
|        | OUT  | (254), A     |                                                |
| DEL2   | LD   | A, 32        |                                                |
| S4     | DEC  | A            |                                                |
|        | JR   | NZ, S4       |                                                |
| BARB   | LD   | A, 0         | ; Признак удара.                               |
|        | OR   | A            |                                                |
|        | JR   | Z, WAIT      |                                                |
|        | LD   | A, 16        |                                                |

|        |      |             |                                      |
|--------|------|-------------|--------------------------------------|
|        | OUT  | (254), A    |                                      |
|        | LD   | A, 0        | Удар                                 |
| STUK   | DEC  | A           |                                      |
|        | JR   | NZ, STUK    |                                      |
|        | LD   | (BARB+1), A |                                      |
| WAIT   | LD   | A, 20       |                                      |
| S5     | DEC  | A           |                                      |
|        | JR   | NZ, S5      |                                      |
|        | OUT  | (254), A    |                                      |
|        | LD   | A, 0        |                                      |
|        | IN   | A, (254)    |                                      |
|        | AND  | 31          |                                      |
|        | XOR  | 31          |                                      |
|        | JR   | NZ, EXIT    |                                      |
|        | DJNZ | BEEP        |                                      |
|        | RET  |             |                                      |
| EXIT   | POP  | BC          |                                      |
| EXI    | EI   |             |                                      |
|        | RET  |             |                                      |
| DECVOL | LD   | A, (VL1+1)  | ; Уменьшение громкости в             |
|        | DEC  | A           | ; первом канале.                     |
|        | JR   | Z, N03      | ;                                    |
|        | LD   | (VL1+1), A  | ; Соответствующее увеличение         |
|        | LD   | A, (DEL1+1) | ; компенсационной задержки в первом  |
|        | INC  | A           | ; канале.                            |
|        | LD   | (DEL1+1), A | ;                                    |
| M03    | LD   | A, (VL2+1)  | ; Уменьшение громкости во            |
|        | DEC  | A           | ; втором канале.                     |
|        | RET  | Z           |                                      |
|        | LD   | (VL2+1), A  | ; Соответствующее увеличение         |
|        | LD   | A, (DEL2+1) | ; компенсационной задержки во втором |
|        | INC  | A           | ; канале.                            |
|        | LD   | (DEL2+1), A |                                      |
|        | RET  |             |                                      |



1. - изменение громкости.

2. - изменение частоты.

Рис.6

Теперь перейдем к формированию частоты. Обычно звуковая частота формируется так, как показано на рис. 2. Но можно формировать и другую форму частоты. Пример на рис. 6.

Теперь уже более понятно, как формировать частоту с изменяющейся амплитудой. Если Вы сделаете программу вывода звука по этому принципу, то Ваша программа будет работать нормально. Единственный недостаток: зависимость частоты от громкости. Предположим, задержка частоты равна 10, а задержка громкости равна 20. При уменьшении громкости будет слышно только изменение частоты и лишь в конце будет слышно некоторое изменение громкости.

Перейдем к более точному методу. Чтобы частота сохранялась при изменении громкости, введем еще один параметр: задержка для сохранения частоты (b). Ее суть состоит в том, чтобы расстояние между импульсами (d) оставалось постоянным, при

изменении громкости (a). Значит, мы должны взять некоторое максимальное значение ( $a_{\max}$ ), откуда  $b = a_{\max} - a$ .

Т.е. чем больше значение задержки громкости (a), тем меньше значение компенсирующей задержки (b). К примеру, для затухания (a) уменьшается на 1, а (b) увеличивается на 1 до тех пор, пока  $(a) > 1$ . При  $(a) = 1$  затухание останавливается.

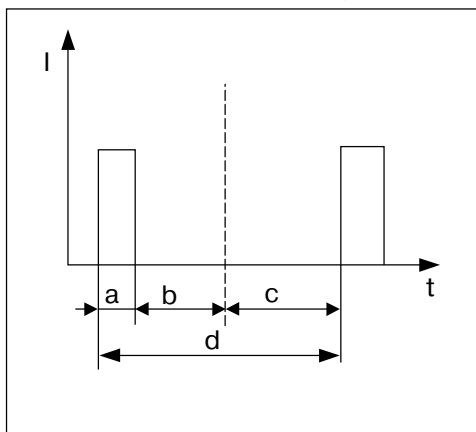


Рис. 7

Следующий вопрос: "Как сделать, чтобы одновременно могли работать несколько звуковых каналов?"

Представим себе некую сумму двух частот (рис. 8). Здесь график I - уменьшающаяся частота, график II - постоянная частота, а график S - результирующая (суммарная частота).

Каждый такой импульс представляет собой отрезок (ab) см. рис. 7. Правда, у первой и второй частоты (a) и (b) могут отличаться. Способ, предлагаемый на рис. 7 здесь не проходит. Отрезок (c) между импульсами одной частоты превращается в счетчик, который определяет: через сколько тактов должен произойти импульс. Этот способ вывода звуковой частоты реализован в программе 4.

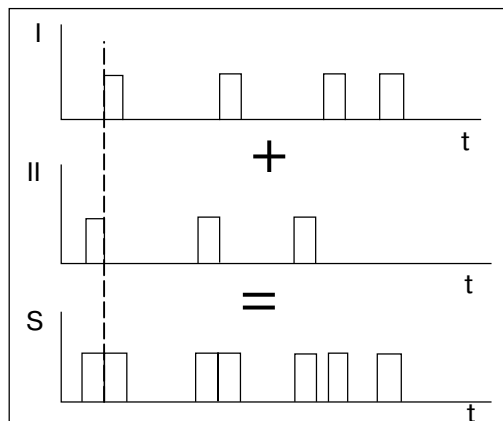


Рис. 8.

Еще одно важное замечание: отрезок (ab) занимает  $1/8$  часть отрезка (c). Это необязательно принимать во внимание, но это обеспечивает нормальную регулировку громкости при более высокой частоте. А так как  $256/8=32$ , то задержка (ab) принята равной 32.

Как работает программа. В памяти с адреса 31000 лежит массив данных. В каждом четном байте лежит значение частоты для I канала, в нечетном - значение ноты II канала. Если номер ноты равен 0, то данные канала не иницируются. Если четный байт более 127 (установлен 7 бит), то удар в барабан. Если четный байт равен 255, то мелодия считается оконченной. Выход из программы осуществляется также нажатием на любую клавишу. Звук барабана формируется продолжительной установкой 1 на динамик и сброс затем. В конце программы подпрограмма DECVOL уменьшает значение громкости обоих каналов. Но можно сделать один из каналов с постоянной громкостью или, наоборот, с ее увеличением.



# ЧИТАТЕЛЬ - ЧИТАТЕЛЮ

В силу объективных причин так сложилось, что "ИНФОРКОМ" серьезно не занимается аппаратными вопросами, связанными со "Спектрумом". Кстати, ведь каждый знает, что нельзя объять необъятное и как знать, что бы из нас получилось, если бы мы начали заниматься несвойственными нам делами, может быть и ZX-РЕВЮ уже не было бы.

Но то, что мы этим вопросом не занимаемся вовсе не означает, что мы его игнорируем. Наоборот, мы очень высоко ставим эту проблему, просто не чувствуем себя мало-мальски компетентными, чтобы самим писать на эту тему или давать какие-то рекомендации. И чем меньше мы в этих вопросах разбираемся, тем ценнее для нас каждое письмо, приходящее от наших читателей на эту тему. Особенно, если в ней затрагиваются глубинные проблемы стандартизации и обеспечения совместимости разных машин.

Сегодня своими мыслями и пожеланиями с Вами делится наш корреспондент из г. Казани Дмитрий Киселев. Это далеко не окончательная точка в проблеме. Это приглашение к свободному диалогу, к обмену идеями и мыслями, а на первых порах и это тоже большое дело. И мы полагаем, что даже тем читателям, которые и не помышляют о доработках и совершенствовании своих машин, эта статья будет интересна, поскольку они смогут узнать больше о том, в какой среде работают их коллеги, каково положение дел на сегодняшний день.

(С) Киселев Д., Казань, 1993г.

## К ВОПРОСУ О СТАНДАРТИЗАЦИИ

В связи с созданием Ассоциации Пользователей Домашних компьютеров и Фонда "ФЛОП" представляю Вашему вниманию мои соображения по поводу стандартизации дальнейших расширений к ПК "Спектрум". Не секрет, что многие системные программисты уже сейчас недовольны скоростными и информативными качествами этой в общем-то неплохой машины. Пример тому - создание многочисленных режимов, не существующих в стандартном "Спектруме 48К" и даже в "128+2".

В некоторых моделях, таких как "Балтика", "АТМ" и "Пентагон профи" даже были попытки внедрить ОС CP/M (ZX-РЕВЮ 5,6-93 и "Радиолобитель N4 за 1993 г.). Однако и здесь наблюдается неразбериха в адресном пространстве этих режимов, от внешнего порта для пользователя (программатор и т.д.) на ВВ55 до разных режимов графики и цвета. Даже на ПК "ОРИОН-128", который недавно начал развиваться, уже существует стандартная таблица портов расширения ("Радио", N4, 1993 г.) Настала пора подумать о подобной таблице и для ПК типа "Спектрум". Я понимаю, что "ИНФОРКОМ" аппаратными вопросами не занимается, однако хотелось бы узнать о программных решениях этого вопроса. В частности, было бы интересно узнать о карте памяти и портах в таких ПЭВМ как "Таймекс 48 и 68" "Энтерпрайз", "Амстрад 464" и им подобных (см. Примечание "Инфоркома" в конце статьи). На Западе ведь тоже люди не только целыми днями в "Элиту" играют, но и пытаются иногда сменить картриджи и приблизиться к более серьезной среде программирования. Неплохо было бы также услышать мнение специалистов по поводу удобства программирования и логической законченности подобных расширений. Само собой разумеется, что в одном письме всех аспектов этой проблемы не отразишь, поэтому я попросил бы вкратце ознакомить читателей с этим вопросом и пусть они выскажутся и предложат свои решения. Со своей стороны я предлагаю в качестве затравки стандартизацию следующих расширений.

Во-первых, расширение памяти свыше 128К удобно адресовать через 2 старших бита в системном порту SP128 (ZX-РЕВЮ 2/91). Таким образом можно адресовать до  $2^5 \cdot 16K = 512K$  ОЗУ. Однако здесь встают по крайней мере 2 проблемы:

1) Необходимо так пронумеровать страницы, чтобы пользователь мог наращивать ОЗУ постепенно: сначала 1 линейка РУ5 (я предлагаю страницы 5,2,0 закрепить за ней, а в

качестве первой сделать страницу 4, но об этом ниже). Далее при установке 2 линейки РУ5 адресовать к ней остальные 4 страницы, а при установке во 2-ую линейку РУ7 изменять только банки этих 4 страниц.

2) Так как в режиме 320К будут существовать 4 страницы с номером 7, необходима договоренность об использовании только одного SCREEN1 (ZX-РЕВЮ 2/91) или нескольких, что потребует дополнительных условий для переключения. В любом случае следует достичь максимальной совместимости с режимом 128К, а также возможности легкой доработки копировщиков, текстовых редакторов и т.д. на режим 320К. Все, кто может предложить какое-нибудь решение, пусть напишет о нем в "ИНФОРКОМ".

Теперь про теневую страницу. Номер 4 я выбрал потому, что по физическим адресам она является предыдущей к 5-й и нижние 32К переадресовывать не нужно. В теневой странице можно размещать доп. экраны, отладчики и даже альтернативные версии ПЗУ, будь то Бейсик-система с 3-х скоростной загрузкой-выгрузкой, которой я пользуюсь, или TR-DOS. Эта возможность прекрасно реализована в ПК "Балтика". Порт, расположенный по 7ЕН может не только отключать ПЗУ, но и выбирать 4 разновидности экранного пространства, о чем будет рассказано ниже.

К сожалению, при включенном ПЗУ отключается всякое обращение к этой странице, что затрудняет загрузку в нее. С помощью небольшой доработки (нужно от элемента D32 (ЛЛ1), отключить выход D33.2(ЛЕ4 выв. 6) и подключить выход D33.3 (ЛЕ4 выв. 8)) можно загружать любые данные с адреса 0000Н с помощью стандартного загрузчика в ПЗУ. Способность записывать данные в ОЗУ через ПЗУ пригодится также и для мониторов и отладчиков, имеющих собственный экран в этой области.

Кстати об экранах. В том же ПК "Балтика" реализовано 4 экрана, о чем говорилось выше. Первый экран - стандартный, расположен с адреса 4000Н. Второй экран - такой же по формату, но расположен с адреса 2000Н и может использоваться отдельным отладчиком, после этого экрана остается 2К на матрицы шрифтов 16x16. Третий и четвертый экраны заслуживают особого описания. Дело в том, что при формировании адресов их атрибутов отключается частота, делящая экран на 24 знакоместа в столбце и атрибуты адресуются так же, как и графика.

Таким простым фокусом, который легко сделать и на "Ленинграде" и на "Пентагоне 48", достигается графика не только высокого графического, но и цветного изображения (один байт пикселей на один байт цвета). Третий экран расположен с 0000Н, а четвертый с 4000Н. эти 4 экрана коммутируются битами 0 и 1, а ПЗУ отключается битом 7 порта 7ЕН. Таким образом, еще остается место для других расширений.

Я предлагаю в качестве стандартного расширения экран с 2000Н и экран высокого цветного разрешения с 4000Н, т.к. он позволяет загружать стандартные картинки, а атрибуты в крайнем случае можно или пересчитывать, или убирать совсем. Конечно, при этом будет затрагиваться область буфера принтера, но в этих областях "сорят" и многие фирменные программы. Наличие же загружаемого "ПЗУ" позволит сместить эти области вверх и полностью решить проблему.

Конечно, найдутся люди, которые скажут: - "зачем нам эти 12К графики, у нас и с 6К Спектрум не успевает работать". Однако, если стандартизировать также и турбо-режим процессора, который кстати можно прикрепить к одному из свободных битов порта 7ЕН, то соотношение 12К/7(8) Mhz будет даже лучше стандартного, да и проблема с "клэшингом" атрибутов почти решится. Остается, правда, вопрос о том, хватит ли быстродействия у ОЗУ, но это уже проблема технологов, а не программистов.

Следующий вопрос, который подымался на страницах "ZX-РЕВЮ" - это вопрос стандартизации русского шрифта. И если создать доп. матрицы знаков и разместить их в удобном месте - это не проблема, то привязку русских букв к латинским клавишам, а также вопрос о выборе способа и кода переключения нельзя оставлять без внимания. Здесь я тоже предлагаю не изобретать велосипед, а пойти навстречу рядовым пользователям - секретарям, писателям и машинисткам. Достаточно взглянуть на решение этой проблемы

со стороны фирмы IBM, которая выпускает клавиатуру именно в стандарте QWERTY (ЙЦУКЕН), а не в стандарте QWERTY (ЯВЕРТЫ), который нравится лишь программистам-системщикам, а не машинисткам. Что же касается недостающих клавиш, то здесь можно предложить ввести дополнительные линии данных в порту клавиатуры FEN. Это легко аппаратно реализуемо и используется в таких ПК, как "Оризон", "Оризон-микро", "Магик" (изг. в г. Ижевске), "Байт", а также, вероятно, в фирменном SP128+2, где есть разъем KEYPAD. Нужно лишь стандартизировать определенную привязку этих клавиш к линиям адресов и внести небольшие изменения в ПЗУ, связанные также с кодами переключения РУС/LAT, в качестве которых я предлагаю использовать коды 02/03, которые в Спектруме не используются, а в ASCII именуются как начало и конец текста соответственно.

Необходимо также подправить синтаксис БЕЙСИКа, чтобы он не интерпретировал коды переключения внутри чисел и т.п. Впрочем, мне будет интересно узнать и другие мнения по этому поводу.

Следующая доработка не встречалась мною ни в одном компьютере и потому я рискну назвать ее своей собственной. Речь идет о расширении музыкальных возможностей Спектрума. Сегодня существует множество программ, использующих музыкальный процессор 8912(10) в режиме 48K. Может быть, они используют также и порт динамика и записи на магнитофон для имитации ударников, хотя я лично таких о программ не слышал. Во всяком случае, я предлагаю использовать этот порт для создания ритм-фона и обращаю ваше внимание на следующий факт: в 8912(10) существуют 1(2) порта, которые предназначены для обмена данными с внешними устройствами и находятся внутри адресного пространства регистров 8912(10) и которые не используются.

Я предлагаю поставить на порт А (который есть на обоих процессорах) два 4-х разрядных ЦАПа, которые будут делить сигналы динамика и магнитофона на 16 уровней, что дополнительно расширит диапазон музыкальных возможностей ПК. Конечно, ЦАП это слишком громко сказано, я имею в виду всего лишь известную R-2R цепочку. Кстати, 16 уровней выходного сигнала вполне достаточно (в крайнем случае можно сделать эти 16 уровней промежуточными между "0" и "1", т.е. получим "0" плюс еще 16 уровней от +5в лог. 1), т.к. 8912(10) тоже имеет 16 уровней громкости и модуляции, во что я поверил только тогда, когда прочел его техническое описание.

Из написанного видно, что желательно проводить комплексную стандартизацию расширений для наиболее полного их использования. Теперь слово за остальными читателями. Вообще нужно активнее работать в этом направлении, чтобы потом не было мучительно больно за то, что наш любимые компьютеры - это прыжок в будущее из позапрошлого. А основными критериями для стандартизации я предлагаю сделать в порядке важности

- 1) минимальную несовместимость на программном уровне;
- 2) удобство программирования, близость в адресном пространстве одностипных устройств и расположение "опасных" портов по четным адресам, меньшим 128;
- 3) легкость аппаратной доработки на наиболее распространенных моделях.

Внедрение этих доработок позволив всколыхнуть необъятные программирующие умы наших Родин, которые наделают на них (компьютерах) такие крутые программы, что на Западе (я надеюсь) люди начнут потихоньку собирать в гаражах "СНГовские Спектрумы".

\* \* \*

#### КОММЕНТАРИЙ "ИНФОРКОМа".

Читатель поставил вопрос о том, что ему интересен подход зарубежных производителей ПК "Энтерпрайз", "Амстрад-464", "Таймекс-2048" и "Таймекс-2068". Это уже вопрос по нашей части, кое-какой информацией мы располагаем.

Сразу должны сказать, что опыт "Энтерпрайза", и "Амстрада" нам мало пригодится, поскольку эти машины по архитектуре существенно отличаются от "Спектрума", несмотря

на то, что собраны на том же процессоре Z-80. То же можно сказать и о "Шарпе" и о стандарте MSX ("Ямаха", "Хит-Бит", "Спектравидео" и пр.). Все эти машины (кроме м.б. "Шарпа") уже при своем "зачатии" планировались под работу в дисковой системе, чего нельзя сказать о "Спектруме", в котором ничего для этого не было сделано, и потому это скорее промежуточные прототипы для более поздних действительно интересных (по своему) машин, и их архитектура сейчас представляет скорее академический, чем практический интерес для наших условий.

Совсем другое дело - "Таймекс". Эта фирма долгое время (до 1985 г.) работала по прямому заказу К.Синклера. Основная цель, стоявшая перед этими машинами была такая - "Обеспечить проникновение "Спектрума" на американский континент" (было открыто представительство Синклера в Бостоне). Все-таки существовала определенная разница между Европейским и Американским рынками. У американцев и требования к машинам покрупнее, да и звонкой монеты в карманах всегда было побольше. То, что эта цель достигнута не была, другой вопрос, а вот как это делалось, нам действительно может быть интересно, ведь мы тоже сейчас находимся в аналогичной позиции - обеспечиваем захват "Спектрумом" рынка СНГ.

Модель 2048 - почти полностью идентична стандартному "Спектруму". Естественно, для американцев сделали красивый дизайн, дешевая, надежная и очень удобная клавиатура и сейчас может служить отличным образцом для большинства наших производителей. Встроили порт джойстика и даже ВЫКЛЮЧАТЕЛЬ ПИТАНИЯ! Но это нам не так важно, важнее скорее то, что был применен новый контроллер экрана - на нестандартной, специально для этого случая сделанной БИС - единственное значительное изменение в архитектуре.

Благодаря ей, стало возможным (теоретически) введение дополнительных экранных режимов и второго экрана, что делалось управлением по 255-ому порту. Практически же, надо сказать, это нигде не использовалось.

Переключение экранов выполнялось командой OUT 255,1.

Кроме того, был введен дополнительный экранный режим (54 знака на 22 вместо 32X22), что исполнялось командой OUT 255,n, где n - число, которое определяет цвет символов и фона, например, OUT 255,62 - печать черным по белому.

Появилась возможность реализации цветной графики высокого разрешения. На каждое знакоместо отводилось по 8 байтов атрибутов (вместо 1), т.е. в каждой линии знакоместа можно было задавать свой цвет INK и свой цвет PAPER. Эта возможность исполнялась командой OUT 255,2.

Команда OUT 255,0 - возврат к стандартному "Спектрумовскому" экрану.

Никакой поддержки в ПЗУ для этих нестандартных режимов работы не было и они могли реализовываться только программированием в машинных кодах. Стандартное программное обеспечение, поддерживающее эти режимы также неизвестно.

Существенным шагом по изменению архитектуры компьютера была модель 2068. Она имела два встроенных порта джойстика, звуковой процессор и тот же оригинальный видеоконтроллер. Существенные изменения имело и базовое ПЗУ. Здесь действительно появилась новая карта памяти, новые системные переменные и новые порты ввода/вывода.

БЕЙСИК приобрел несколько новых команд: DELETE, FREE, STICK, ON ERROR и SOUND.

Команда DELETE позволяла удалять строку по ее номеру или группу строк (DELETE.....).

Команда FREE позволяла узнать объем свободной памяти БЕЙСИКА, например (PRINT FREE).

Команда STICK читала положение одного из двух джойстиков и работала аналогично команде INKEY\$. Ее формат, например:

PRINT STICK (m,n)

Здесь m=1, если интересует положение рукоятки или m=2, если интересует состояние кнопки "ОГОНЬ". Параметр n=1 или 2 (соответственно левый или правый джойстик).

Команда ON ERROR обеспечивала переход в заданное место программы в случае возникновения ошибки. Команда SOUND служила для управления звуковым процессором, который имел три музыкальных и три шумовых канала, из БЕЙСИКа.

Как и в "Таймексе-2048" здесь возможны те же режимы управления экраном, тоже возможно иметь два экрана или высокое разрешение атрибутов или 64 символа в экранной строке.

Соответственно, была изменена карта памяти (звездочками отмечены участки, совпадающие со стандартным "Спектрумом").

#### Карта памяти с одним экраном.

|                 |                                                           |
|-----------------|-----------------------------------------------------------|
| 0000H           | Системное ПЗУ. (*)                                        |
| 1000H           | Графический экран. (*)                                    |
| 5800H           | Атрибуты. (*)                                             |
| 5B00H           | Буфер принтера. (*)                                       |
| 5C00H           | Системные переменные.                                     |
| 6000H           | Машинный стек.                                            |
| 6200H           | Диспетчер прикладных программ в машинных кодах.           |
| 6315H           | Код управления банками памяти.                            |
| 6840H           | Блок переменных для прикладных программ в машинных кодах. |
| ARSBUF<br>CHANS | Информация о каналах.                                     |
| PROG            | БЕЙСИК - программа.                                       |
| VARs            | Переменные БЕЙСИКа.                                       |
| ELINE           | Буфер редактируемой строки.                               |
| WORKSP          | Буфер INPUT.                                              |
| STKBOT          | Стек калькулятора.                                        |
| STKEND          | Свободно.                                                 |
| RAMTOP          | Верхний предел БЕЙСИКа.                                   |
| UDG             | Графика пользователя.                                     |
| P_RAMT          | Физический конец ОЗУ.                                     |

Как видно из этой карты памяти, во многом ее структура совпадает с картой стандартного "Спектрума". В то же время, выделение зарезервированных областей для машинного стека, блока диспетчера прикладных программ в машинных кодах и блока машиннокодовых переменных, говорит о том, что во-первых, была попытка стандартизировать программы пользователя в кодах и, во-вторых, по-видимому, предполагалось в дальнейшем надстраивать архитектуру компьютера, оставив эти области

неизменными для обеспечения совместимости с ранее выпущенным программным обеспечением.

Переключение графических режимов выполнялось теми же командами OUT, что и на Таймексе-2048, но здесь это поддержано новой картой памяти и новыми системными переменными.

#### Карта памяти с двумя экранами.

|                 |                                                           |
|-----------------|-----------------------------------------------------------|
| 0000H           | Системное ПЗУ. (*)                                        |
| 4000H           | графический экран-1. (*)                                  |
| 5800H           | Атрибуты-1. (*)                                           |
| 5B00H           | Буфер принтера. (*)                                       |
| 5C00H           | Системные переменные.                                     |
| 6C00H           | Графический экран-2.<br>Атрибуты - 2.                     |
| 7B00H           | Блок переменных для прикладных программ в машинных кодах. |
| ARSBUF<br>CHANS | Информация о каналах.                                     |
| PROG            | БЕЙСИК-программа.                                         |
| VARs            | Переменные БЕЙСИКа.                                       |
| ELINE           | Буфер редактируемой строки.                               |
| WORKSP          | Буфер INPUT.                                              |
| STKBOT          | Стек калькулятора.                                        |
| STKEND          | Свободно.                                                 |
| RAMTOP          | Верхний предел БЕЙСИКа.                                   |
| UDG             | Графика пользователя.                                     |
| F7C0H           | Машинный стек.                                            |
| F9C0H           | Диспетчер прикладных программ в машинных кодах.           |
| FAD5H           | Код управления байтами памяти.                            |
| P_RAMT          | Физический конец ОЗУ.                                     |

#### Новые порты I/O.

В Таймексе-2068 было задействовано несколько новых портов ввода/вывода для обслуживания во-первых двух джойстиков и, во-вторых, для обслуживания музыкального процессора.

Управление музыкальным процессором осуществлялось командами OUT по портам F6H - регистр данных и F5H - индексный регистр. Сначала командой OUT по порту F5H получим доступ к одному из регистров музыкального процессора, а потом командой OUT по

порту F6H -вводим в него нужное число.

Для чтения джойстиков сначала включался контроллер командой  
OUT F5H,DEH

а затем их положение определялось анализом портов:

F5FEH - левый

F6FDH - правый.

#### Дополнительные возможности.

Из-за существенных изменений в ПЗУ, в системных переменных и в карте памяти стала проблематичной совместимость с программами, написанными для фирменного "Спектрума". В принципе, эта совместимость возможна, но это скорее исключение, чем правило. Для того, чтобы все-таки добиться полной совместимости, фирма "Таймекс" начала выпускать дополнительные картриджи ПЗУ, для подключения которых компьютер имеет специальный отсек.

После подключения такого картриджа компьютер становится полностью аналогичен фирменному варианту, сохраняя свои дополнительные возможности для тех, кто обращается к ним через внешние порты из машинного кода.

Для удобства пользователей на подобных картриджах выпускались и игровые и прикладные программы, например текстовый редактор, а с учетом того, что американцы испытывали в то время явную ностальгию по CP/M, был выпущен и картридж "Зебра", превращающий компьютер в CP/M-совместимую машину.

У наших программистов, имеющих со старых времен большие заделы работ в CP/M тоже явно прослеживается ностальгия по этой системе, и вообще во многом пути развития могут быть схожи. Сейчас не хотелось бы делать выводы и рекомендации, еще не время, но опыт предшественников может дать пищу для размышлений...

Единственное, что можно было бы сказать тем, кто думает над доработками и стандартизацией. Опыт показывает, что это не должно быть самоцелью и обязательно должно быть **ПОДДЕРЖАНО ЭКОНОМИЧЕСКИ**.

Пусть мы скажем грубо, но прямо. Вы можете быть альтруистом сколько хотите и можете делать какие угодно открытия просто за спасибо, из любви к работе. Но если то, что Вы придумали, не даст другим людям заработать ни копейки, Ваше дело не приживется. И наоборот, если Вы открываете новый путь, по которому пойдут люди и увидят для себя в этом возможность не только поработать, но и **ЗАРАБОТАТЬ**, они подхватят, разовьют и увековечат Вашу идею и Вас вместе с ней. На этом построен весь многолетний опыт К. Синклера.

Он дал возможность развернуться тысячам фирм, выпускающим программы и сотням фирм, выпускающим периферию. Те сделали свой капитал, а его компьютер стал самым популярным в мире. То же самое происходит и у нас.

Впрочем, последнее слово останется конечно за Вами, дорогие читатели и мы будем ждать Ваши письма и Ваше мнение.

От редакции.

Представляя читателям эту интересную авторскую работу, мы должны сказать несколько слов о "Спектруме", о нашем взгляде на этот компьютер и вообще на "Синклеровское" движение в целом.

"Спектрум" занимает совершенно особое место в деле компьютеризации нашей страны. Количество его поклонников давно перевалило за миллионы. Это движение, если говорить честно, стало всенародным и уникальным. Уникальность его состоит в том, что огромная часть самого талантливого, способного и одаренного населения одной шестой земного шара, не спрашиваясь ни советов, ни разрешений у высокого начальства, сделала то, что оказалось не под силу никаким госкомитетам и институтам. В рамках скромных 48 килобайт каждый день во всех уголках страны решаются интересные проблемы, делаются настоящие открытия.

Отличительной чертой нашего "синклериста" стала необычайная гордость за свою машину и уважение к ее возможностям. Ни для кого не секрет, что это маленькое чудо может очень многое из того, что и предположить себе не могут фирмы, торгующие суперсовременными 386-ыми и 486-ми IBM-совместимыми машинами. Мы знаем множество примеров, когда способные программисты, намаявшись за рабочий день с IBM-совместимой техникой, приходят домой и отдыхают наедине с любимым "Спектрумом". Они ни на что его не променяют.

Умение реализовать свои идеи, найти новые решения, сделать свой компьютер профессиональным, оставаясь в пределах 48 килобайт, дают такой большой простор для творчества и самоутверждения, какого не во всяком деле найдешь. Мы можем восхищаться гигантскими прокатными станами, ловко расправляющимися с тоннами металла, но всегда будет в почете профессия ювелира, способного так обработать несколько граммов, что они будут радовать глаз и вызывать всеобщее восхищение.

Все это имеет непосредственное отношение к данной статье. Астрономические расчеты достаточно сложны, но это одна из тех областей, в которых Ваш "Спектрум" смело может потягаться с гораздо более мощными потомками. Редакцию "ZX-РЕВЮ" особенно радуют такие материалы, которые выводят эту машинку на новый уровень, позволяют взять от него то, для чего она не предназначалась.

Мы никогда не сталкивались с астрологией и были приятно обрадованы тем фактом, что это, как оказалось, не мертвая наука былых лет, а постоянно развивающееся учение и занимаются им люди серьезные. Они непрерывно растут, повышают свою эрудицию, компьютерную грамотность, расширяют круг общения. Короче говоря, если Вы растете и вместе с вами растут возможности Вашего "Спектрума", то это то, что надо - основная цель "ZX-РЕВЮ" достигнута!

Для тех же из Вас, кто захочет заняться этим делом основательно, эта статья сможет стать первой вешкой на длинном и трудной пути. Мы далеки от мысли, что прочитав и разобрав ее, Вы сразу выйдете на некоторый достаточный уровень. Нет, это будет только началом. Может быть, даже трудным началом. Вам еще предстоит узнать и понять очень и очень многое. Статья достаточно сложна для неподготовленного читателя, поэтому кое-что, самое необходимое, в процессе редактирования мы вынесли в отдельный комментарий в ее заключении.

Автор статьи - Сергей Крушинский. Он не является ни профессиональным астрологом, ни профессиональным программистом. И то и другое - хобби. По профессии он журналист. В то же время, его программы по астрологии у нас распространены не меньше, чем западные аналоги и, если Сергей достиг значительных успехов в своем деле, то благодаря огромной любви к астрологии и, самое главное, к своему SPECCY, как он нежно называет "Спектрум".



## SPECCY + АСТРОЛОГИЯ



Доводилось ли когда-нибудь читателю заглядывать в кабинет современного астролога? Если да, то могу спорить: первое, что бросилось в глаза, было не магическим кристаллом и не чучелом крокодила. На самом почетном месте светился дисплей! Как-то у моего знакомого, считающегося по праву одним из ведущих московских астрологов, сломался его "Коммодор". "Такое чувство, словно выключилось левое полушарие моего мозга", - жаловался он. Ничего удивительного, я сам вспоминаю как кошмар возню с таблицами и циркулем, длившуюся порой до полуночи ради того, чтобы построить единственный гороскоп, а под утро, напрягая взгляд, пытаешься его прочитать.

Дело решительным образом изменилось после того, как я купил "Спектрум" и написал первую программу - она стала делать за меня "грубую" работу.

Что такое компьютерная астрология? Это эффективная работа с большим количеством информации, возможность экспериментировать и исследовать. Чего только нет в современных астрологических программах, написанных для машин класса IBM: базы данных, атласы мира, графика, статистическая обработка информации... Даже самому отъявленному скептику небезынтересно взглянуть на аккуратные распечатки, представляющие его персону с новой точки зрения.

Может показаться, что "Спектруму" с его 48-ью килобайтами не под силу тягаться с куда более мощными компьютерами. Ничего подобного. Даже если за дело возьмется начинающий программист, не имеющий понятия ни о чем, кроме стандартного Бейсика - каким я был 2 года назад, когда написал первую программу, - SPECCY хватит на то, чтобы сделать по крайней мере самое важное, а именно: построить гороскоп рождения.

### **Что такое гороскоп.**

Астрология покоится на аксиоме, согласно которой состояние мира в данном месте и в данный момент времени есть нечто уникальное. Когда человек рождается, он настолько восприимчив, что это состояние мира накладывает на него неповторимый отпечаток, впоследствии в большей или меньшей степени определяющий его реакции, а также тип ситуаций, с которыми приходится сталкиваться.

Самая большая роль отводится Солнцу, Луне и планетам Солнечной системы. Их расстановка - основа астрологического "сюжета" - никогда не повторяется во всех подробностях, так же, как не повторяется ни одна шахматная партия.

Сравните три рисунка. На них изображено по сути одно и то же: так называемая "небесная сфера". Вторая картинка имеется в любом астрономическом учебнике. За ее разъяснением можете обратиться и в энциклопедию. А третья - традиционный гороскоп.

Положение любого тела на небесной сфере определяется парой координат: долготой или угловым расстоянием от точки весеннего равноденствия до проекции тела на эклиптику ( $^{\circ}/1$ ), и широтой - расстоянием "вверх" или "вниз" от нее, со знаком "+" к Северному полюсу эклиптики и "-" к южному.

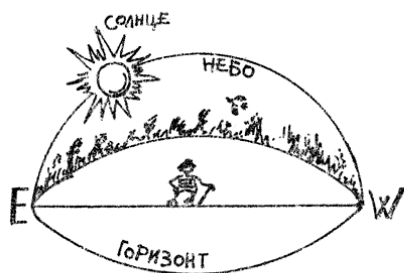


Рис. 1

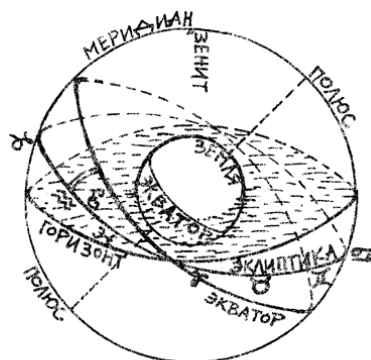
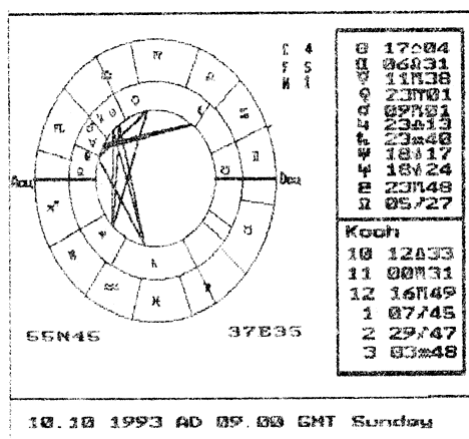


Рис. 2



Внешний круг - это пояс Зодиака. Он делится на 12 знаков:

- |              |              |
|--------------|--------------|
| ♈ - Овен     | ♉ - Телец    |
| ♊ - Близнецы | ♋ - Рак      |
| ♌ - Лев      | ♍ - Дева     |
| ♎ - Весы     | ♏ - Скорпион |
| ♐ - Стрелец  | ♑ - Козерог  |
| ♒ - Водолей  | ♓ - Рыбы     |

Внутреннее кольцо разделено на 12 домов. Их тоже 12. Их отсчет начинается с линии горизонта (Асц-Дсц) и нумеруются против часовой стрелки, как и знаки.

В этом же кольце расположены Солнце ☉, Луна ☾ и планеты:

- |            |          |
|------------|----------|
| Меркурий ☿ | Венера ♀ |
| Марс ♂     | Юпитер ♃ |
| Сатурн ♄   | Уран ♅   |
| Нептун ♆   | Плутон ♇ |

Кроме того, Северный или Восходящий Лунный узел ♋.

Зодиак - это пояс созвездий, тянущийся вдоль эклиптики. По нему совершают свой видимый путь Солнце, Луна и планеты. Астрологи делят его на 12 знаков по 30 градусов: Овен, Телец, Близнецы и т.д. Начало Овна - точка весеннего равноденствия. Астрологи говорят: "Марс - 12 градусов Тельца", а не "42 градуса", хотя это то же самое. В наше время знаки не совпадают с одноименными созвездиями.

Другая система координат называется экваториальной. В ней две мерки - прямое восхождение и склонение.

В астрологии есть еще такое понятие, как "дома". Их, как и знаков, двенадцать, только получаются они путем деления не зодиака, а пространства вокруг горизонта. Начало 1-го дома совпадает с точкой горизонта, где восходит солнце. Если знак говорит о том, как проявляет себя находящаяся в нем планета, то дом - о сфере жизни, на которую это действует. Сами же планеты - символы определенных жизненных принципов.

Солнце - источник сил, достоинства, принцип власти: глава государства и "царь в голове" (этот ряд можно продолжать до бесконечности); Луна - инстинкты, привычки, чувства, Марс - борьба, самоутверждение, Венера - гармония, любовь и т.д. Изучив положение планет в знаках, домах и угловые расстояния между планетами, астролог пытается построить характерный сюжет.

... Итак, надо научить "Спектр" делать следующее:

1) Рассчитывать эклиптические геоцентрические  $(^*/2)$  координаты Солнца, Луны и планет;

2) Определять границы домов в той же эклиптической системе координат и выводить результат на экран или принтер хотя бы в виде таблицы. Впрочем, места вполне хватит и на приличный рисунок.

Основой для расчетов будут служить дата, время рождения и, кроме того, географическая широта и долгота. Рассмотрим подробнее эти этапы.

### Основной расчет.

Вначале определяется положение Солнца на эклиптике. Рассчитываем ZM (среднюю аномалию), ZV (истинную аномалию), ZR (радиус-вектор). Затем довольно простым способом SL, долготу солнца. Широта его всегда равна нулю  $(^*/3)$ . Две величины, полученные на данном этапе, ZR и ZM надо сохранить как SR и SM - они понадобятся в дальнейшем. Чтобы найти положение планеты, необходимо:

1) Рассчитать тем же способом, что и для Солнца, ZM, ZV, ZR;

2) Найти гелиоцентрическую широту  $(^*/4)$  и долготу планеты (ZB и ZL);

3) Используя полученные ранее величины, перейти к геоцентрическим координатам.

Местонахождение Луны определяется совершенно особым способом, для нее придется предусмотреть отдельный программный блок. Вместе с Луной будет рассчитываться и долгота лунного узла - точки, в которой Луна пересекает эклиптику, двигаясь с юга на север. Она важна для гороскопа, поскольку говорит о том, насколько гармонично личность вписывается в социум. Все сказанное можно проиллюстрировать схемой (табл. 1):

Таблица 1

| Солнце                                                | Планеты (от J=2 до J=9)  | Луна                                     |
|-------------------------------------------------------|--------------------------|------------------------------------------|
| SL, SR, SM<br>A(1,1)=SL<br>A(1,2)=0<br><br>ZM, ZR, ZV | ZL, ZB    A(J,1), A(J,2) | A(10,1), A(10,2)<br>A(11,1)<br>A(11,2)=0 |

Входом в этот блок процедур послужит переменная, выражающая искомый момент времени - T и таблица, где содержатся характеристики орбит на некоторый известный момент, так называемую начальную эпоху T0. Их нетрудно пересчитать на эпоху T. Выход - массив A(2,11), долгота и широта 10 небесных тел и лунного узла.

Формулы, которые используются для решения почти всех поставленных здесь задач, - это классические формулы небесной механики, основывающиеся на законах, открытых Иоганном Кеплером. Смешно подумать, они уместаются в десяток Бейсик-строк. Есть, правда, нюанс. Тела Солнечной системы влияют друг на друга, что нарушает в некоторой мере "правильность" их движения. В астрономии это называется "возмущениями". Сатурн, например, может оказаться в соседнем градусе. Для достижения точности, на разных этапах расчетов приходится вносить поправки.

Чтобы вычислить положения планет с точностью до угловой секунды, рекомендую

такой алгоритм:

- а) отключить телефон, запереть дверь;
- б) выпить крепкого кофе;
- в) набирать тригонометрические выражения, страницу за страницей; вернуться к шагу б).

Выход из этого цикла надо предусмотреть по одному из следующих условий:

- если не хватило памяти компьютера;
- если у программиста нервное истощение;
- если последний убедился, что его старания бесполезны, т.к. все равно для 8-разрядного компьютера сверхвысокая точность недостижима (или достижима, но сверхвысокой ценой)!

К счастью, в астрологии, в отличие от навигации, не требуется такая точность. Плюс-минус 10 минут вполне достаточно. Поэтому надо будет всего лишь внести маленькую поправку в гелиоцентрическую долготу Сатурна, Юпитера, а по желанию и Нептуна и Урана. Эта точность будет почти неизменной не только для нашего столетия, но и, скажем, для 16-го века.

### Календарь.

Период Т говорит о том, сколько столетий прошло от начальной эпохи. Примем за начальную эпоху 31 декабря 1889 г., 12 часов 00 минут по Гринвичу. Рассчитаем, сколько суток и долей суток прошло с этого момента. Переведем их в столетия, поделив на 36525. Если наша дата приходится на один из прошлых веков, число будет отрицательным.

Определение интервала в сутках - дело довольно хлопотное, если браться за него "в лоб". Поэтому астрономы применяют так называемую систему юлианских дат. За точку отсчета взят средний Гринвичский полдень 1 января 4713 г. до н.э. В следующий полдень наступила юлианская дата 1, потом -2 и т.д. Таким образом, начальной эпохе соответствует ю.д. 2415020, а 1 февраля 1965 г. 11 ч. 46 мин. - 2438792.99. Для определения ю.д. существует формула:

$$\begin{aligned} \text{JD} = & \text{TRUNC}(365.25 * Y) + \\ & \text{TRUNC}(30.6001 * (M+1)) + \\ & D + A + 1720994.5 \end{aligned}$$

Если месяц - январь или февраль, то  $Y = (\text{год} - 1)$ ,  $M = (\text{месяц} + 12)$ . Иначе  $Y = \text{году}$ ,  $M = \text{месяцу}$ .

$$D = \text{число} + (\text{час по Гринвичу} + \text{мин} / 60) / 24$$

$$A = 0, \text{ если дата дана по старому стилю. Иначе } A = 2 - X + \text{TRUNC}(X/4), \text{ где } X = \text{TRUNC}(Y/100)$$

Дальше просто: вычитаем из ю.д. 2415020 - получается число суток от начальной эпохи. Между прочим, в подавляющем большинстве программ, занимающихся определением дня недели и биоритмов, используется система юлианских дат. Так, если известна ю.д. в 0 часов по Гринвичу, то номер дня недели:

$$N = (\text{JD} + 1.5) \text{ MOD } 7.$$

0 - воскресенье, 6 - суббота.

Примечание: Здесь мы применили две функции - TRUNC( ) и MOD, которых нет в стандартном Бейсике, хотя они имеются среди кодов калькулятора "Спектрума", поэтому надо дать некоторые пояснения.

Функция TRUNC выделяет целую часть дробного числа. Дотошный читатель может предположить, что то же самое делает и стандартная функция INT, но это не совсем так. Они совпадают только для положительных чисел, для отрицательных же они различаются.

Дело в том, что INT всегда округляет число "вниз", а TRUNC - всегда округляет к "нулю".

Так, если число равно 3.14, то  $\text{INT}(3.14) = 3$  и  $\text{TRUNC}(3.14)$  тоже равно 3. С другой стороны, если число равно -3.14, то  $\text{INT}(-3.14) = -4$ , а  $\text{TRUNC}(-3.14)$  равно -3.

Задать в стандартном БЕЙСИКе функцию  $\text{TRUNC}$  можно так:

```
DEF FN T(X)=SGN(X)*INT(ABS X)
```

Теперь функция  $\text{MOD}(X,Y)$  - это остаток от деления  $X/Y$ :

```
DEF FN M(X,Y)=X-Y*INT(X/Y).
```

Итак, помимо блока с основными расчетами, в программу надо включить календарные вычисления. На входе дата и время по Гринвичу, на выходе - T (период) и JD (юлианская дата, она может еще пригодиться).

### Ввод исходных данных.

Если не претендовать на очень уж товарный вид программы, то подойдет стандартное INPUT. Обозначим данные, которые должны быть получены в результате так: YE, MO, DA, HO, MI, LA, LO - год; месяц; день; часы; минуты; градусы широты (положительное число, если она северная, отрицательное, если южная); градусы долготы ("+" к западу, "-" к востоку, это может не соответствовать некоторым стандартам, но так удобнее).

Способ организации ввода, т.е. вводить ли поясное время, а потом пересчитывать его в Гринвичское или сразу последнее; записывать долготу в виде градусов/минут или в виде десятичной дроби, как вводить для нее знак - дело вкуса. Главное - разобраться во всем заранее, в противном случае появятся ошибки, которые трудно будет выловить. И, разумеется, надо своевременно сообщить пользователю о выбранном формате.

В англо-американских программах принято записывать не "23:15", а "11:15", затем "PM" - "после полудня". Потом - номер часового пояса ("TIME ZONE"). 16 июня 1930 г. почти на всей территории бывшего СССР стрелки были переведены на 1 ч. вперед, что получило название "декретное время". А с 1980-го действует летнее. Поэтому к номеру часового пояса надо еще прибавлять 1 или 2 часа.

Координаты Москвы в большинстве "фирменных" программ записываются так: 037E35; 55N45 ("N" означает север, "E" - восток).

Я вполне солидарен с программистами, которые считают хорошим тоном проверять, не ошибся ли пользователь. Широта не может выходить за пределы -90, +90 градусов. В феврале 1993 г. 28, а не 29 дней... Что еще хорошо бы учесть? В самом начале я писал о домах гороскопа. Сегодня чаще всего применяют 2 системы: Планидуса и Коха, и в том и в другом случае при расчете идет в ход тригонометрия. Так вот, если задана заполярная широта и одна из этих систем, происходит остановка с сообщением "INVALID ARGUMENT".

Определить, сколько дней в любом месяце любого года можно по формуле:

$$X=28+\text{ABS}(\text{INT}(\text{MO}/8) - (\text{MO}/2 - \text{INT}(\text{MO}/2)) * 2) + 2 * \text{SGN}(\text{ABS}(\text{MO}-2)) + \\ (1 - \text{SGN}((Y/4 - \text{INT}(Y/4)) * 4)) * (1 - \text{SGN}(\text{ABS}(\text{MO} - 2)))$$

В тех же фирменных программах, как правило, имеется банк данных, в котором хранятся во-первых имена и даты рождения клиентов, а во вторых, координаты городов. Дисковая операционная система TR-DOS позволяет сделать то же самое на "Спектруме".

### Астрологические дома.

Если полный оборот вокруг своей оси Земля совершает примерно за 24 часа, то каждые 4 минуты, или около того, над горизонтом восходит новый градус эклиптики. В астрологии этот участок называется Асцендентом и считается одним из самых важных факторов гороскопа, пожалуй даже более важным, чем солнечный знак (о котором идет речь на последних страницах газет). Примерно так же меняется кульминирующий градус или середина неба (она занимает самое высокое положение в небе по сравнению с остальной частью эклиптики). Противоположные им точки называются "десцендент" и "Основание неба", а все вместе - "углами гороскопа", вот вкратце их значения:

Асцендент - темперамент, внешность, самооценка, самые характерные черты

личности и судьбы... вообще, по одному нему можно сказать очень многое о человеке. Вся астрология, если угодно - не что иное, как учение о больших и малых циклах. Асцендент - это начало (в этой точке планета восходят, "рождаются"), а по началу любого цикла можно спрогнозировать его дальнейшее развитие. Еще его можно сравнить с оглавлением книги, где коротко указано, чего от нее ждать.

Середина Неба - социальная реализация личности, карьера, общественное положение (здесь небесные тела кульминируют; Солнце пересекает это место в полдень, образно говоря, достигает расцвета), С.Н. в решительном, инициативном знаке Овна - человек готов идти на риск и борьбу, добиваясь профессионального успеха; в знаке Рыб - наоборот, пассивен, плывет по течению, зато шансы нередко сами приходят к нему в руки...

Десцендент - говорит о других людях (в противовес "Я" - Асценденту), с которыми приходится считаться. Прежде всего, это партнеры: коллеги, супруг.

Основание Неба - "источник": родительский дом, детство, родина, "семейное гнездо".

Асцендент - начало 1-го дома. О.Н. - 4-го, Десцендент - 7-го, С.Н. - 10-го. Границы остальных домов рассчитываются в разных системах по-разному. На протяжении 2 тысячелетий нашей эры одни системы уступали место другим. Сегодня широко известны несколько. Любопытно, что одна из них принадлежит перу Иоганна Молиториса де Кунигсберга (1436-1476), признанного математика, внесшего вклад в сферическую тригонометрию. Его имя, как и псевдоним - Региомонтанус - говорит о том, что родом он из нынешнего Калининграда.

Другая система приписывается т. Кампанелле... Я приведу систему Вальтера Коха (1895-1977). Считается, что она особенно хорошо работает в сфере бытовых прогнозов.

Расположение домов неодинаково не только в разное время, но и на разных широтах. Представьте себе апельсин, разрезанный наискосок.

От угла, под которым прошел нож, будет зависеть внешний вид содержимого. Помимо широты и времени суток для решения задачи надо знать еще период  $T$  и наклон эклиптики в эпоху (угол  $\epsilon$ , см. рис. 2) Обозначим его  $EC$ .

$$EC = 23.452292 - 1.30125E-02 * T - 1.6388889E-06 * T * T + 5.0305556E-07, \text{ градусов.}$$

Сперва определяется прямое восхождение Небесного меридиана  $RAMC$ , затем прямое восхождение пересчитывается в долготу (т.е. переходим от одной системы координат к другой) - получается С.Н. Находим асцендент. Наконец, зная "углы", одну за другой рассчитываем границы домов 11-го, 12-го, 2-го и 3-го. Остальные дома расположены в противоположных градусах. На выходе - массив  $H(1...12)$ , дома.

### **"Дебют".**

А.Г.Алексеев в одной из статей для "ZX-РЕВЮ" удачно назвал этим словом фрагменты, которые повторяются в разных программах - с них удобно начинать. В мой "дебют" входит несколько процедур:

#### **1. "INTERVAL".**

В астрономических вычислениях часто приходится иметь дело с большими углами. Например, угол  $ZM$ , рассчитанный по стандартной формуле, может оказаться равным 29345.672 градуса или 512.17858 радианам. Это то же самое, что 185.672 градуса или 3.2405876 радиан. Мало того, что последние два числа нагляднее. Бывают случаи, когда вычисляя тригонометрические функции больших углов, калькулятор "Спектрума" ошибается.

Попробуйте, например, `PRINT SIN(512.17858)`.

Получится -.098815626.

Затем: `PRINT SIN(3.2405876)`.

Результат: -.098833433! Между прочим, IBM показывает: -0,0988334620170090.

Десяток таких ошибок, и результат окажется столь далеким от точного, что духи-

управители планет разгневаются на астролога, который им воспользуется... дабы избежать их немилости, предлагаю процедуру, которая приводит число A в интервал 0 - AMAX:

```
95 REM: вход A-число, AMAX-верхний предел
96 REM: X-результат, Y-локальная переменная
100 LET Y=INT(ABS(A)/AMAX)*AMAX
110 LET X=A-Y*(A>=AMAX)+(AMAX+Y)*(A<0): REM: Слава Джорджу Булю!
120 RETURN
```

## 2. "POLYNOME"

Как Вы заметили, ряд формул содержит умножение последовательности чисел на степени времени (T). Процедуру для этого на Бейсике удобно оформить в виде функции пользователя:

```
129 REM POLYNOME
130 DEF FN P(A,B,C,T)=A+T*(B+C*T)
```

## 3. "ARCTANGENS(A/B)"

При использовании функции ATN возникает неопределенность: неизвестно, в каком квадранте лежит вычисленный угол. Его можно установить по знакам A и Y:

```
139 REM на входе A,B (радианы), на выходе X
140 IF NOT B THEN LET B=1E-6: REM на случай, если B=0
150 LET Y=ATN(A/B)
160 LET A=Y+(Y<0)*PI+(A<0)*PI
170 LET B=PI+PI: GOSUB 100: REM к процедуре INTERVAL, которая вернет результат X
180 RETURN
```

Кроме этих подпрограмм, я включаю в "дебют" объявление констант.

```
10 RESTORE 9000
20 READ ZE, ON, L6, K
30 LET S=PI/K: REM величина, приближающаяся к 1 радиану. Чтобы переводить градусы в радианы
    и наоборот, удобнее всего умножать, и делить число на S, а не на (PI/180)
40 LET K2=K+K: REM 360
50 LET P2=PI+PI
9000 DATA 0,1,60,180: REM константа
```

Этот ход значительно сокращает программу. Все же, из соображений читаемости, я не буду использовать константы в примерах.

Неплохо было бы написать процедуры "дебюта" в кодах калькулятора и разместить их, например, в REM-строках. Параметры удобно передавать туда через функции пользователя. (См. "Элементарная графика", гл.3. или "Программирование в машинных кодах и на языке АССЕМБЛЕРА", ч. 2. гл. 4.6). Вопрос в том, как возвращать в Бейсик результат, он ведь находится на стеке калькулятора в интегральном представлении. Я знаю несколько способов. Увы, ни один из них не назовешь вполне изящным. Самый примитивный - завести особую переменную, допустим Q, затем написать на Ассемблере процедуру, которая будет искать в области Бейсик-переменных нашу Q и переносить в отведенные ей 5 ячеек памяти 5 байт с калькуляторного стека. Процедура, которую я привожу - это слегка измененная программа FIND из N-3,4 "ZX-РЕВЮ" за 1993 год ("Применение ассемблера для создания быстроработающих программ").

|              |                                                                                                     |
|--------------|-----------------------------------------------------------------------------------------------------|
| CALL FIND    | ;Вызов процедуры, которая ищет адрес переменной и помещает его в HL.                                |
| INC HL       | ;После кода имени переменной идут 5 байтов, туда и надо переместить 5 байтов со стека калькулятора. |
| EX DE,HL     | ;Чтобы выполнить команду LDIR, этот адрес должен быть в DE.                                         |
| LD HL,(5C65) | ;Теперь надо найти начало стека калькулятора. Помещаем в HL                                         |

; содержимое системной переменной STKEND. Число находится в ячейках от  
; (STKEND)-5 до (STKEND)-1.

```
DEC HL
DEC HL
DEC HL
DEC HL
DEC HL      ; Теперь в HL - адрес первого байта искомого числа.
LD BC, 5     ; Его длина - 5 байт.
LDIR         ; Передаем его переменной.
RET
```

```
-----
FIND      ; Поиск переменной.
LD HL, (5C4B) ; Адрес системной переменной VARS.
X1 LD A, 51H   ; Код буквы "Q"
CP(HL)    ; Сравнение и выход, если переменная найдена. Ее адрес - в HL.
RET Z
BIT 5, (HL) ; Если 5-й бит в имени равен 0, то это либо массив,
            ; либо символьная строка.
JR NZ, X2  ; Следующие строки следует пропустить.
INC HL
LD E, (HL)
INC HL
LD D, (HL)
ADD HL, DE
INC HL
JR X1

X2 BIT 6, (HL) ; Аналогичным способом проверяем, не является ли переменная
            ; многосимвольной, параметром цикла, наконец, если ни тем, ни
            ; другим - соответствует ли ее имя "Q".

JR NZ, X3
INC HL
LD A, (HL)
BIT 7, (HL)
JR Z, -6
LD DE, 6
ADD HL, DE
JR X1

X3 BIT 7, (HL)
JR Z, -10
LD DE, 13
ADD HL, DE
JR X1
```

**Конструкция получится такая:**

#### **1. В БЕЙСИКе: Объявить переменную Q**

```
LET Q=0
```

**Объявить функцию пользователя:**

```
DEF FN A(A, B, ...) =USR адрес
```

**Вызывать ее:**

```
LET X=FN A(...)
```

**X** здесь не имеет ровно никакого значения: он показывает лишь содержимое регистровой пары BC. Важно то, что ответ содержится в Q.

Читателям, не имеющим опыта работы с калькулятором "Спектрума", предлагаю воспользоваться отлаженной процедурой INTERVAL. Перед тем, как вызывать ее, надо поместить параметры на калькуляторный стек. Первый параметр на стеке (сверху) - AMAX, следующий - число A (листинг 1).

При вычислении Y неплохо было бы воспользоваться командой mod/div, которой располагает калькулятор, но, к сожалению, она не во всех случаях работает корректно.



| Дес. код | Мнемоника  | Комментарий. Содержимое стека.             |
|----------|------------|--------------------------------------------|
| 6        |            | ;Поскольку будет использоваться операция   |
| 10       | LD B,0A    | ; ">=", надо подготовить регистр B, помес- |
|          |            | ; тив туда ее код (10)                     |
| 239      | RST 28H    | ; Включение калькулятора.                  |
| 192      | store M0   | ; Сохраняем AMAX в ячейке M0               |
| 2        | delete     | ; и удаляем его со стека.                  |
| 193      | store M1   | ; Сохраняем A в ячейке M1                  |
| 43       | abs        | ; ABS(A)                                   |
| 224      | recall M0  | ; Вызов AMAX на стек                       |
| 5        | divide     | ; ABS(A)/AMAX                              |
| 50       | truncate   | ; TRUNC(ABS(A)/AMAX)                       |
| 224      | recall M0  | ; AMAX, TRUNC(ABS(A)/AMAX)                 |
| 4        | multiply   | ; AMAX*TRUNC(ABS(A)/AMAX)=Y                |
| 194      | store M2   | ; Y - в ячейку M2                          |
| 225      | recall M1  | ; A, Y                                     |
| 49       | duplicate  | ; A, A, Y                                  |
| 224      | recall M0  | ; AMAX, A, A, Y                            |
| 10       | >=         | ; A >= AMAX ? Результат: 0 или 1           |
| 226      | recall M2  | ; Y, [0 ИЛИ 1], A, Y.                      |
| 4        | multiply   | ; Y*(A>=AMAX), A, Y.                       |
| 3        | subtract   | ; Y*(A>=AMAX)-A, Y                         |
| 56       | end calc   | ; Выход из калькулятора.                   |
| 6        |            | ; Теперь надо подготовить регистр B к      |
| 13       | LD B,0B    | ; операции "<".                            |
| 239      | RST 28H    |                                            |
| 224      | recall M0  | ; AMAX на стек                             |
| 226      | recall M2  | ; Y на стек                                |
| 15       | add        | ; AMAX+Y                                   |
| 225      | recall M1  | ; A на стек                                |
| 160      | const zero | ; 0 на стек                                |
| 13       | <          | ; A < 0 ?                                  |
| 4        | multiply   | ; (AMAX+Y)*(A<0)                           |
| 15       | add        | ; A-Y*(A>=AMAX)+(AMAX+Y)*(A<0) - что и     |
|          |            | ; требовалось...                           |
| 1        | exchange   | ; Лучше не оставлять на стеке "мусора".    |
| 2        | delete     | ; В данной случае за результатом идет лиш- |
|          |            | ; нее число, оно удаляется.                |
| 56       | end calc   | ; После чего - выход из процедуры.         |
|          |            | ; Результат на стеке.                      |
| 201      | RET        |                                            |

### Апология Бейсика.

Вот и зашла речь о наболевшем вопросе: на каком языке программировать? Теперь, когда структура программы более-менее ясна, самое время порассуждать на эту тему.

Не стоит торопиться с ответом: "конечно же, в кодах". И вот почему. При всех бесспорных преимуществах - скорости, экономии памяти, возможности создавать нестандартные эффекты - даже отлаженная машинно-кодовая программа останется как бы замороженной. Изменить какие-либо детали еще можно, но структуру... легче писать заново, чем резать по живому. Если же автор серьезно интересуется астрологией, его взгляды на предмет не стоят на месте. То и дело возникает желание попробовать новую методику или изменить привычную. То и дело говоришь себе: "Вот этого я раньше не учитывал, а надо бы".

Хочу подчеркнуть, структура, которую я описал, подходит для скромной программы, не претендующей на то, чтобы удовлетворить неслабые потребности профессионального астролога. А методов, помимо гороскопа рождения - хоть пруд пруди...

Совершенствуешься и в умении решать вычислительные задачи. Я, например, совсем недавно узнал, как проще всего рассчитывать орбиту Плутона (это запутанный вопрос в астрономии, поскольку Плутон был открыт сравнительно недавно). И до сих пор не нашел

простого способа выполнять на восьмиразрядном "Спектруме" математические операции с двойной точностью, как, например, в GW-BASICe: задал "DEFDBL A-T" - и все переменные от А до Т становятся числами с удвоенной разрядностью. Но рано или поздно, ведь, узнаю, быть может не без помощи "ZX-РЕВЮ". Кстати, не занимался ли кто-нибудь из читателей этим вопросом? Не думаю, чтобы это было архисложной задачей. В компьютере MSX, построенном на том же процессоре Z-80, все числа по умолчанию 16-разрядные! Это очень пригодилось бы в календарных расчетах (9-я цифра в юлианской дате не обязательна, но ее присутствие повысило бы точность).

... Еще один "минус" кодовой программы: мониторы-отладчики, по крайней мере, те, что мне известны, не "берут" кодов калькулятора. Попробуйте сами и увидите такую фантазмагорию! Дабы уменьшить мучения, мне пришлось делать специальный отладчик; он позволял писать мнемониками и макро-командами, переводил числа в упакованную форму - вобщем, выполнил всякую рутину, но при всем этом был лишен удобств пакета GENS/MONS. Другое дело, игра - там калькулятор используется не так часто, задачи иные, а потому и подход не тот.

Недостатки стандартного БЕЙСИКа общеизвестны. Ладно, пускай, вычисления. Хуже то, что ради экономии памяти приходится идти на уловки, делающие порой программу нечитаемой. Отсутствуют некоторые команды, очень полезные для создания хорошей структуры, типа ON...GOTO, зато спектрумовский БЕЙСИК один из самых демократичных. Как-то я попробовал на пижонском компьютере MSX написать нечто вроде:

```
GOSUB(VAL"1000" AND X=SGN PI)+(VAL"2000" AND X=BIN).
```

Беднягу, кажется, стошнило...

Есть ли другие варианты? Первое, что приходит в голову - ПАСКАЛЬ, однако, повозившись недельку-другую с версией фирмы Hisoft, приходишь к выводу, что авторы ее заботились в первую очередь о том, как облегчить себе жизнь. Вещественные числа там имеют всего 6 значащих цифр (\*5). Чтобы получить результат такой же точный, как на БЕЙСИКе, приходится проделывать разные арифметические трюки. Из-за этого и по другим причинам катастрофически не хватает памяти. Нет полной ясности относительно того, как Паскаль взаимодействует с системой - распространенные руководства говорят об этом крайне скупо. Так что далеко не все машинно-кодовые фрагменты, которые пытаешься вставить в программу, используя оператор INLINE - особенно, если в них есть обращения к ПЗУ, дают ожидаемые результаты.

Программу написать можно, в этом я убедился. Работать она будет исключительно быстро, читать ее текст и отлаживать - одно удовольствие - ПАСКАЛЬ есть ПАСКАЛЬ. Но считать она будет максимум в рамках двух столетий, делать сможет очень немного... и никаких красот. Может быть, есть смысл сделать на Паскале графический табель-календарь, показывающий, как планеты движутся в течение, скажем, заданного месяца - благо, разрешение экрана не настолько высоко, чтобы на графике были видны минуты, зато важно, чтобы расчеты выполнялись быстро.

С другой стороны, ничуть не худшего эффекта можно добиться с помощью TOBOS-FP, пожалуй, единственного из Бейсик-компиляторов, который оправдывает надежды. С ним пришлось немало повозиться. Часть того, что о нем написано - не только в "таблюдных" инструкциях, но и в такой серьезной книге, как "Диалекты Бейсика для ZX-SPECTRUM" - оказалось неточным. А кое о чем важном там вообще не сказано, например, о том, что объектный код является релоцируемым. Несмотря на то, что непосредственно после компиляции он располагается с адреса 24000, его можно загружать и запускать с другого адреса - скажем, с 26000, расположив ниже собственные кодовые подпрограммы. Лишь бы он не затирал рабочие процедуры и стек компилятора.

Напомню, что для компиляции больших программ в TOBOSe предусмотрен режим "компиляции с уничтожением исходного текста". Оказалось, этот режим не работает с дисковой системой, которая сдвигает БЕЙСИК вверх. Так что приходится инициализировать стандартную SINCLAIR-систему страшной командой RANDOMIZE USR 0, загружать с ленты компилятор и исходный текст, компилировать, записывать полученный объектный код на

ленту, потом загружать его с ленты в новый адрес и только после этого переписывать на диск (\* /6). Удовольствия мало. Тем не менее, мне удалось откомпилировать таким образом довольно объемную программу, включавшую помимо расчетов еще и графику, сервис, разные виды таблиц - большую часть того, что необходимо профессиональному астрологу - и знакогенератор, печатающий 60 символов в строке. Больше всего поразила скорость. Хотите верьте, хотите нет, она была почти такой же, как у программы "PRIMA", написанной Б.Богдановым для IBM на Турбо-Паскале!

Однако, увы, за все приходится платить. Недостаток TOBOSa - пониженная точность расчетов (он работает с 7-значными числами); памяти, несмотря на все, он съедает много: хочешь добавить в программу какой-нибудь милый пустячок - не тут-то было...

Вариант, который мне кажется почти идеальным для некоммерческой астрологической программы - это БЕТА-БЕЙСИК. О его преимуществах было достаточно написано в "ZX-РЕВЮ". Он как будто нарочно создан для подобных программ. Позже пойдет речь об уравнении Кеплера. Оно очень красиво вписывается в структуру DO...UNTIL. Оператор ON...GOSUB в паре мест просто необходим. Функции SINE, COSE, знакогенератор 4/7 пикселей, возможность гибко манипулировать сеткой графических координат, окна - все это помогает наглядно иллюстрировать результаты.

### Параметры орбит.

Я уже писал, что для расчета координат небесных тел нужно знать во-первых, период Т, а во-вторых, параметры их орбит в начальную эпоху Т0. Для каждой из планет таких параметров 6: средняя аномалия, эксцентриситет, большая полуось, долгота перигелия, долгота восходящего узла и наклон орбиты. Обозначим их переменными ZM, ZE, ZA, ZW, ZQ, ZI.

Допустим, дата у нас 24 июня 1976 г. Время - 0 ч. по Гринвичу. Тогда юлианская дата:

JD= 2443683. 5,

а

$T = (2443683.5 - 2415020) / 36525 = 0.78476386.$

Найдем среднюю аномалию ZM для Меркурия. В таблице даны три числа:

102,27938,

149472,52,

7E-6.

Обозначим их X1, X2, X3.

$ZM = X1 + X2 * T + X3 * T * T$

(или  $FN P(X1, X2, X3, T) = 117402.91$  градусов =  $42.91004$  градусов =  $0.74892085$  радиан.

Точно так же находим остальные значения.

Все величины, кроме ZA (астрономические единицы) и ZE нужно перевести в радианы!

Поскольку каждая орбита представляет собой эллипс, данные параметры описывают именно эту геометрическую фигуру, а также ее положение относительно орбиты Земли.

### Положения Солнца и планет.

Средняя аномалия ZM показывает, где бы находилась планета, если бы она двигалась равномерно по кругу. Зная ее и эксцентриситет эллипса, можно найти эксцентрическую аномалию EA и истинную аномалию ZV, т.е. перевести планету на эллипс. Для определения EA служит уравнение Кеплера:

$EA = ZE * \sin(EA) = ZM.$

Очевидно, решить его не так-то просто, Но компьютеру все нипочем! Вот как это делается.

```
197 REM HELI01
```

```
198 REM
```

```
199 REM Уравнение Кеплера. На входе ZE,ZM. На выходе EA
```

```
200 LET EA=ZM: REM начальное значение EA
```

```
202 LET EPSYLON=1E-6: REM задаем точность
```

```
205 REM начало цикла
```

```

210 LET DELTA=EA-ZE*SIN(EA)-ZM: REM невязка.
220 LET EA=EA-DELTA/(1-ZE*COS(EA)): REM очередное значение EA
230 IF ABS(DELTA)>EPSYLON THEN GOTO 210: REM к началу цикла, если не достигнута требуемая
    точность

```

А вот как то же самое выглядит в кодах калькулятора (к моменту вызова этой процедуры на вершине калькуляторного стека должно находиться ZM, за ним - ZE. Сама процедура может быть расположена в любом месте оперативной памяти). См. листинг-2.

## Листинг-2

| Дес. код | Мнемоника    | Комментарий                              |
|----------|--------------|------------------------------------------|
| 239      | RST28H       | ; Вкл. калькулятора                      |
| 195      | store M3     | ; Начальное значение EA, т.е. ZM -       |
|          |              | ; в 3-ю ячейку памяти                    |
| 195      | store M4     | ; ZM в 4-ю ячейку.                       |
| 1        | exchange     | ; Меняем местами ZM и ZE                 |
| 197      | store M5     | ; ZE в 5-ю ячейку                        |
| -----    |              |                                          |
| 2        | delete       | ; Удаляем верхнее число со стека.        |
|          |              | ; Это начало цикла.                      |
| 49       | duplicate    | ; Копирование вершины стека.             |
| 229      | recall M5    | ; Вызываем ZE                            |
| 1        | exchange     | ; Теперь стек выглядит так: EA, ZE, EA   |
| 31       | sin          | ; sin(EA), ZE, EA                        |
| 4        | multiply     | ; ZE*sin(EA), EA                         |
| 3        | subtract     | ; EA-ZE*sin(EA)                          |
| 226      | recall M4    | ; вызываем ZM                            |
| 3        | Subtract     | ; EA-ZE*sin(EA)-ZM, т.е. невязка DELTA.  |
| 49       | duplicate    | ; DELTA, DELTA                           |
| 161      | const one    | ; 1, DELTA, DELTA                        |
| 227      | recall M3    | ; EA на стек                             |
| 32       | cos          | ; COS(EA)                                |
| 229      | recall M5    | ; ZE на стек                             |
| 4        | multiply     | ; cos(EA)*ZE                             |
| 3        | subtract     | ; 1-cos(EA)*ZE, DELTA, DELTA             |
| 5        | divide       | ; DELTA/(1-cos(EA)*ZE), DELTA            |
| 227      | recall M3    | ; EA на стек                             |
| 1        | exchange     | ;                                        |
| 3        | subtract     | ; EA минус последний результат. Получено |
|          |              | ; очередное значение EA                  |
| 195      | store M3     | ; сохраняем его в 3-й ячейке             |
| 1        | exchange     | ; меняем местами EA и DELTA              |
| 42       | abs          | ; abs(DELTA)                             |
| 52       |              | ; Это ввод                               |
| 221      |              | ; числа                                  |
| 6        |              | ; 1E-5 в упакованной форме               |
| 55       |              | ; (допустимая                            |
| 189      |              | ; величина                               |
| 5        |              | ; невязки)                               |
| 3        | subtract     | ; abs(DELTA)-1E-6                        |
| 55       | gt zero      | ; >0?                                    |
| 0        |              |                                          |
| 225      | jump true-31 | Если да, то переход к началу цикла       |
| -----    |              |                                          |
| 56       | end calc.    | ; Выключение калькулятора.               |
| 201      | RET          | ; Результат на вершине стека.            |

### Теперь можно найти истинную аномалию ZV:

```

235 REM истинная аномалия. На входе EA, ZE, ZA (большая полуось).
240 LET A=ZA*SIN(EA)*SQR(1-ZE*ZE)
250 LET B=ZA*COS(EA)-ZE)
260 GOSUB 140: REM к процедуре ARCTANGENS
270 LET ZV=X

```

### Затем радиус-вектор, ZR:

```
275 REM радиус-вектор. На входе ZA, EA, ZE
290 LET ZR=ZA*(1-ZE*COS(EA))
```

Таким образом, величины ZR, ZV найдены, что и было намечено в первоначальной схеме. На этом можно было бы и закончить подпрограмму HELIO1, однако, здесь же удобнее вычислить еще одну промежуточную величину, используя очередной параметр орбиты. Обозначим ее ZU.

```
290 LET A=ZV+ZW: LET AMAX=PI+PI
300 GOSUB 100: REM к процедуре INTERVAL
310 LET ZU=X
320 RETURN: REM конец подпрограммы HELIO1
```

Долготу солнца (SL) найти теперь просто. В случае с ним ZU -это долгота Земли. Так что  $SL=ZU+PI$ . Полученную величину надо "прогнать" через процедуру INTERVAL. С планетами сложнее. Вот как выглядит расчет для них.

```
321 REM HELIO2
322 REM гелиоцентрические координаты планет.
323 REM на входе ZQ,ZI,ZU, на выходе ZL, ZB - гелиоц. широта и долгота
324 REM
325 REM *** ZI.
330 LET A=COS(ZI)*sin(ZU):LET B=COS(ZU)
340 GOSUB 140: REM к процедуре ARCTANGENS
350 LET A=X+ZQ: LET AMAX=PI+PI
360 GOSUB 100: REM к процедуре INTERVAL
370 LET ZL=X
380 LET ZB=ASN(SIN(ZU)*SIN(ZI))
390 RETURN
395 REM GEOCENTRIC
396 REM переход от гелио- к геоцентрическим долготе и широте (LO,LA)
397 REM на входе: ZR, ZL, ZB, а также SR, SL - радиус-вектор и долгота Солнца.
398 REM Z, A, M, C, N, D - локальные переменные
399 REM
400 LET A=ZL-SL:LET M=ZR*COS(ZB):LET C=ZR*SIN(ZB)
410 LET N=M*SIN(A): LET D=M*COS(A)+SR
415 REM *** LO
420 LET A=N: LET B=D: GOSUB 140:REM ARCTANGENS
425 LET A=X+SL: LET B=PI+PI:GOSUB 100: REM INTERVAL
430 LET LO=X
435 REM *** расстояние от Земли
440 LET Z=SQR(M*N+D*D+C*C)
445 REM *** LA
450 LET LA=ASN(ZR/Z*SIN(ZB))
460 RETURN
```

Осталось перевести LO и LA из радиан в градусы и передать массиву A( ).

### Луна.

Вместе с Солнцем и Асцендентом, Луна - важнейший показатель в любом гороскопе, а в женском часто самый главный. Достаточно вспомнить, с чем она связывалась в представлениях разных народов, чтобы понять ее значение в астрологии. В древних религиях ее почитаем, как богиню зачатий и материнства. Если Солнце - активное, мужское начало, то Луна - уравнивающая его пассивная, женская сторона природы, инстинкт самосохранения, чувства... Известно, что лунные циклы оказывают влияние на многие процессы в животном и растительном мире, на здоровье человека. Древние славяне лечились, выполняли земледельческие работы и даже солили капусту сообразно с определенными фазами Луны.

Луна движется через Зодиак быстро, проходя полный круг за месяц, а один знак -

примерно за двое суток. Точно вычислить ее положение в заданный момент времени непросто - слишком много факторов приходится учитывать. На каждом шагу надо вносить поправки. Целиком все занимает пару книжных страниц. Тем, кто желает вычислить положение Луны с точностью до минуты, рекомендую обратиться к книге Ж.Месса "Астрономические формулы для калькуляторов". Здесь же приведу упрощенный расчет, занимающий немного меньше места и дающий точность плюс-минус 10 минут дуги. Это не так страшно, как может показаться на первый взгляд - ведь видимый диск Луны "растянут" на целых 30 минут, полученная же долгота указывает положение центра диска.

```

495 REM MOON
496 REM на входе T,JD,SL,SN. На выходе X,Y,ZQ - долгота, широта Луны и долгота Лунного узла
498 REM C,D - вспомогательные локальные переменные
499 REM параметры орбиты
500 LET D=JD-2415020: REM число дней, прошедших от начальной эпохи
505 DIM P(5): REM для поправок
510 LET ZL=270.434164+D*13.176397: REM средняя долгота
520 LET ZW=334.32956+D*0.11140408: REM долгота перигея
530 LET ZM=ZL-ZW: REM средняя аномалия
540 LET ZQ=259.18328-D*0.052953923 REM долгота узла
543 LET ZI=5.145396: REM наклон орбиты
545 REM приведение параметров в интервал 0...360 и перевод в радианы.
550 LET AMAX=350
500 LET A=ZM: GOSUB 100: LET ZM=X*(PI/180)
570 LET A=ZL: GOSUB 100: LET ZL=X*(PI/180)
580 LET A=ZQ: GOSUB 100: LET ZQ=X*(PI/180)
590 LET ZI=ZI*(PI/180)
595 REM расчет основных поправок
600 LET C=ZL-SL
610 LET P(1)=1.2739*SIN(C+C-M)
620 LET P(2)=0.1858*SIN(SM)
630 LET P(3)=0.37*SIN(SM)
640 LET ZM=ZM+(P(1)-P(2)-P(3))*(PI/180)
650 LET P(4)=6.2686*SIN(ZM)
660 LET P(5)=0.214*SIN(ZM+ZM)
670 LET ZL=ZL+P(1)+P(4)-P(8)+P(5))*(PI/180)
680 LET ZL=ZL+0.6583*SIN(2*ZL-SL))*(PI/180)
690 LET ZQ=ZQ-0.16*SIN(SM)*(PI/180)
700 LET C=ZL-ZQ
705 REM *** долгота
710 LET A=SIN(C)*COS(ZI): LET B=COS(C)
720 GOSUB 140: REM ARCTANGENS
725 LET A=X+ZQ: LET AMAX=PI+PI: GOSUB 100: REM INTERVAL
730 REM *** широта
740 LET Y=ASN(SIN(C)*SIN(ZI))
750 LET X=X/(PI/180): LET Y=Y/(PI/180): LET ZQ=ZQ/(PI/180)
760 RETURN

```

Разумеется, вместо (PI/180) лучше использовать константу S, которая равна 1.7453292E-2, т.е. приблизительно 1 радиану. А массив P(5) объявить в начале программы.

### ***Поправки на возмущения планетных орбит.***

Больше всего отклоняются от правильного в геометрическом смысле движения Сатурн и Юпитер. К их средним долготам (ZL) надо прибавить:

**Для Юпитера:**

```

((0.3314-0.0103*P(1))*SIN(P(4))-
0.0644*P(1)*COS(P(4)))*(PI/180)

```

**Для Сатурна:**

```

((0.1609*P(1)-0.0105)*COS(P(4))+
(0.0182*P(1)+0.8142)*SIN(P(4))-
0.1488*SIN(P(5))-0.0408*SIN(P(5)+
P(5))+0.0856*SIN(P(5))*COS(P(3))+
0.0813*COS(P(5))*SIN(P(3)))*(PI/180).

```

где:

$$P(1)=T/5+0.1$$

$$P(2)=(237.47555+3034.9061*T)*(PI/180)$$

$$P(3)=(265.91650+1222.1139*T)*(PI/180)$$

$$P(4)=5*P(3)-2*P(2) \quad P(5)=P(3)-P(2)$$

Как всегда, полученные величины лучше привести в интервал  $0 \dots PI*2$ .

### Расчет астрологических домов по системе В.Коха.

```
885 REM
886 REM KOCH HOUSES
887 REM
888 REM на входе: GMT-время по Гринвичу в часах и десятых долях часа, L0 - долгота в
      градусах, "-" к востоку. LA-широта в градусах, "-" к югу, EC- наклон эклиптики в
      градусах, T. Выход: массив H(1...12) - эклиптическая доля гота границ домов.
892 REM
893 REM Прямое восхождение меридиана
894 REM
900 LET RAMC=FN P(6.6460656,2400.0513,2.581E-5,T):REM POLYNOME
910 LET A=(RAMC+GMT)*15-L0
920 LET LA1=LA*(PI/180)
930 IF LA>(90-EC) THEN LET A=RAMC+180: LET LA1=-LA1
940 LET A=RAMC: LET AMAX=360:GOSUB 100: LET RAMC=X: REM INTERVAL
945 LET RAMC=RAMC*(PI/180): LET EC1=EC*(PI/180):REM перевод в радианы
950 REM середина неба
960 LET A=SIN(RAMC):LET B=COS(RAMC)*COS(EC1): GOSUB 140: REM ARCTANGENS
965 LET MC=X
970 REM Асцендент
980 LET A=COS(RAMC):LET B=-(SIN(EC)*TAN(LA1)*COS(EC)*SIN(RAMC):GOSUB 140
985 LET AS=X
990 LET L=1: LET X=AS:GOSUB 1140
993 LET L=10: LET X=MC:GOSUB 1140
995 REM Вершины домов
1000 LET X0=SIN(EC1)*TAN(LA1)/COS(EC1)*SIN(RAMC)
1010 RESTORE 1020
1020 DATA 0.52350879,1.0471976,2.0943952,2.617994
1030 LET X4=-3
1040 FOR J=1 TO 4
1050 READ X: LET X4=X4+1
1060 IF X4=0 THEN LET X4=1
1070 GOSUB 1100
1080 NEXT J
1090 RETURN
1095 REM вычисления
1100 LET X3=RAMC+X+ATN(X0/SQR(1-X0*X0))*X4/3
1110 LET A=SIN(X3):LET B=COS(X3)*COS(EC1)-TAN(LA1)*SIN(EC1)
1120 LET L=L+1: IF L>12 THEN LET L=L-11
1130 GOSUB 140: REM ARCTANGENS
1140 LET H(L)=X/(PI/180):LET A=X+PI: LET AMAX=PI+PI:GOSUB 100: REM INTERVAL
1150 LET H(L+6-12*(L>7))=X/(PI/180): REM противоположный дом
1160 RETURN
```

### Управляющая программа.

Поскольку для ряда простых расчетов я давал лишь формулы, то нет смысла подробно описывать главный блок, показывая номера строк для переходов, важнее сам алгоритм. Выглядит он так:

1) Ввести исходные данные. Проверить, не допустил ли пользователь ошибку. Если да, то начать сначала. Распечатать эти данные на экране и спросить у него самого, все ли в порядке. Если нет, начать сначала. Перевести все в формат, который будет использовать программа.

2) Рассчитать юлианскую дату, соответствующую заданной дате и времени, вычислить период T.

3) Рассчитать по T и данным из таблицы исходные параметры орбиты Солнца: ZM, ZE,

ZA, ZW. Найти положение Солнца (см. HELIO1). Сохранить среднюю долготу, радиус-вектор и среднюю аномалию Солнца в особых переменных.

4) Найти положение Луны и Северного Лунного узла -  $M(1,1)$ ,  $M(2,1)$ ,  $M(11,1)$ ,  $M(12,1)$

5) Вычислить значения  $P(1..5)$ , которые понадобятся при расчете поправок на долготы Сатурна и Юпитера

6) Для планет:

Цикл для J от 2 до 9

Определить ZM, ZE, ZA, ZW, ZQ, ZI.

Найти гелиоцентрические координаты. Если J=5 или 6 (Юпитер, Плутон), вычислить поправки и прибавить их к ZL.

Перейти к геоцентрическим координатам.

Перевести результат в градусы и присвоить массиву M( ).

7) Найти наклон эклиптики EC.

8) Рассчитать границы домов H( ).

9) Вывести результат на дисплей или принтер.

10) Вернуться к началу.

Я намеренно не касался пункта (9) только потому, что тема эта слишком обширна. Сюда входят вопросы дизайна и графики, форматирования чисел... возможно даже несколько нестандартного прерывания, если Вы желаете, чтобы в углу экрана имелись часы - для всего этого потребовалась бы отдельная статья. Хотелось бы только обратить внимание читателей на следующее: все результаты, хранящиеся в массивах H( ) и M( ) - вещественные числа, долготы лежат в диапазоне 0...360 градусов. Выводить на экран лучше градусы и минуты. Если же придерживаться астрологической традиции, то надо указывать знак Зодиака. Так, например, 338.25002 преобразуется в строку "08:04 Рыб". Один из возможных способов вывода представлен на рис.3 (получен из зарубежной программы фирмы Protheus software).

В заключение прошу извинить меня за слишком поверхностное изложение астрономии. Всем, кому такой уровень покажется недостаточным, горячо рекомендую книги: Ж.Меес - "Астрономические формулы для калькуляторов", М., Мир, 1988; П. Дэффет-Смит - "Практическая астрономия с калькулятором", М., Мир, 1982. Что же касается тонкостей астрологии, то среди моря доступной сегодня литературы я бы выделил учебники Сефариала, Джорджа Ллевеллина, Ф. Сакоян, Хет Монстера и философские работы Редиара.

\* \* \*

#### КОММЕНТАРИИ "ИНФОРКОМа".

/1. "Эклиптика" - правильнее говорить - "плоскость эклиптики".

Это плоскость, которая совпадает с плоскостью орбиты земли. Поскольку Земля вращается вокруг Солнца, то естественно, что Солнце всегда лежит в этой плоскости.

/2. Геоцентрические координаты - это координаты в системе, в которой начало отсчета привязано к центру Земли. Эклиптические геоцентрические координата - координаты в системе, в которой начало координат совпадает с центром Земли, а главная плоскость XY совпадает с плоскостью эклиптики.

/3. Поскольку Солнце естественно всегда лежит в плоскости эклиптики, то его широта в эклиптической геоцентрической системе координат всегда равна нулю.

/4. Гелиоцентрическая система координат - в отличие от геоцентрической связана не с Землей, а с Солнцем. Центр этой системы находится в центре Солнца, а главная плоскость XY - совпадает с плоскостью эклиптики.

/5. Это действительно так, но надо признать, что это не столько облегчает жизнь создателям ПАСКАЛЯ, сколько пользователям. Ведь благодаря этому в ПАСКАЛе HP4T удастся использовать только четыре восьмиразрядных регистра для хранения действительных чисел (вместо пяти). Это значительно упрощает интерфейс программ на



ПАСКАЛе с собственными процедурами пользователя в машинном коде. Из м/к легко находятся и эксплуатируются переменные ПАСКАЛЯ.

/6. По-видимому, используя информацию, приведенную в статье Алексеева А.Г. "Секреты TRDOS" (см. ZX-РЕВЮ N1,2 за 1993 г.) этот недостаток компилятора можно было бы и устранить. Не исключено, что кто-то это уже и сделал.

Справочные таблицы для расчетов элементов орбит планет.

#### SUN

|    |           |           |          |
|----|-----------|-----------|----------|
| ZM | 358.47583 | 35999.05  | -.00015  |
| ZE | .01675104 | -.0000418 | -1.26E-7 |
| ZA | 1.0000002 | 0         | 0        |
| ZV | 101.22083 | 1.71918   | .00045   |

#### MERCURY

|    |            |           |           |
|----|------------|-----------|-----------|
| ZM | 102.27938  | 149472.52 | 7E-6      |
| ZE | 0.20561421 | .00002046 | -3E-8     |
| ZA | 0.3870986  | 0         | 0         |
| ZV | 28.753753  | 0.3702806 | .0001208  |
| ZQ | 47.145944  | 1.1852083 | .0001739  |
| ZI | 7.002881   | .0018608  | -.0000183 |

#### VENUS

|    |           |            |           |
|----|-----------|------------|-----------|
| ZH | 212.60322 | 58517.804  | .001286   |
| ZE | .00682069 | -.00004774 | 9.1E-8    |
| ZA | 0.7233316 | 0          | 0         |
| ZU | 54.384186 | 0.5081861  | -.0013664 |
| ZQ | 75.779647 | 0.89985    | .00041    |
| ZI | 3.393631  | .0010058   | -1E-6     |

#### MARS

|    |           |            |          |
|----|-----------|------------|----------|
| ZM | 319.51913 | 19139.855  | .000181  |
| ZE | .0933129  | .000092064 | -7.75-8  |
| ZA | 1.5236883 | 0          | 0        |
| ZW | 285.43176 | 1.0697667  | .0001313 |
| ZQ | 48.786442 | 0.7709917  | -1.4E-6  |
| ZI | 1.850333  | -.000675   | .0000126 |

#### JUPITER

|    |           |           |           |
|----|-----------|-----------|-----------|
| ZM | 225.32833 | 3034.692  | -.000722  |
| Z8 | .04833475 | .00016418 | -4.876E-7 |
| ZA | 5.2025613 | 0         | 0         |
| ZV | 273.27756 | 0.5994317 | .00070405 |
| ZQ | 99.443414 | 1.01053   | .00035222 |
| ZI | 1.308736  | -.0056961 | 3.9E-6    |

#### SATURN

|    |           |           |            |
|----|-----------|-----------|------------|
| ZM | 175.46622 | 1221.5515 | -.000502   |
| ZE | .05589232 | -.0003455 | -7.28E-7   |
| ZA | 9.554747  | 0         | 0          |
| ZV | 338.3078  | 1.0852207 | .00097854  |
| ZQ | 112.79041 | 0.8731951 | -.00015218 |
| ZI | 2.492519  | -.0039189 | -.00001549 |

#### URANUS

|    |           |            |           |
|----|-----------|------------|-----------|
| ZM | 72.64878  | 428.37911  | .000079   |
| ZE | .0463444  | -.00002658 | 7.7E-8    |
| ZA | 19.21814  | 0          | 0         |
| ZW | 99.071581 | 0.985765   | -.0010745 |
| ZQ | 73.477111 | 0.4986678  | .0013117  |
| ZI | 0.772464  | .0006253   | .0000395  |

#### NEPTUNE

|    |           |           |           |
|----|-----------|-----------|-----------|
| ZM | 37.73063  | 218.46134 | -.00007   |
| ZE | .00899704 | 6.338-6   | -2E-9     |
| ZA | 30.10957  | 0         | 0         |
| ZW | 276.04597 | 0.3256394 | .00014095 |
| ZQ | 130.68139 | 1.098935  | .00024987 |
| ZI | 1.779242  | -.0095436 | -9.1E-6   |

#### PLUTO

|    |           |           |        |
|----|-----------|-----------|--------|
| ZM | 229.94722 | 144.91306 | 0      |
| ZE | .24864    | 0         | 0      |
| ZA | 39.51774  | 0         | 0      |
| ZW | 113.52139 | 0         | 0      |
| ZQ | 108.95444 | 1.39601   | .00031 |
| ZI | 17.14678  | 0         | 0      |

# СДЕЛАЙТЕ САМИ

(С) Т.Д. Фрост (T.D. FROST), 1986.

(С) Адаптация, Алексеев А.Г., 1993.

(С) Авторизованный перевод, редактирование, "ИНФОРКОМ". 1993 г.

## ADVENTURE GAMES

### **Введение.**

Те, кто интересуются адвентюрными играми, знают о существовании программ для автоматизированного создания таких игр - QUILL, GAC и др. Интерес к этим программам поддерживается прежде всего тем, что очень часто любителям игровых программ приходят в голову отличные сюжеты, рождаются готовые сценарии и человек хочет сам создать карту своего выдуманного мира, населить пещеры троллями, попрыгать золото в сундуках и насытить игру хитроумными головоломками. Что делать в такой ситуации? Если Вы не программист, то дело ваше почти безнадежное.

Но на помощь Вам может прийти "ABS" - это система, которая позволит сделать первый шаг. Она может пригодиться и опытным программистам, ведь для того, чтобы проверить свою задуманную концепцию вовсе не нужно садиться и писать игровую программу. На это могут уйти месяцы, а окажется, что концепция не интересна, сценарий скучен и все труды затрачены зря. Гораздо удобнее иметь такой гейм-дизайнер, как ABS и тогда проверить свою концепцию можно в течение нескольких часов.

Эта программа будет полезна и тем, кто просто интересуется вопросом "Как это делается?" Как устроены адвентюрные игры, как они работают - все это Вы узнаете, разобрав эту систему.

"ABS", если вдуматься, готовый инструмент для создания обучающих программ для детей типа "Путешествие Васи Кляксина в страну Невыученных Уроков". Когда наберете необходимый навык, сможете делать такие программы своим детям хоть по пять штук в неделю. Вы можете также быстро создать игру-иллюстрацию к той книжке, которую только что прочитал Ваш ребенок или ученик.

И, наконец, нет предела творческому воображению наших читателей - опытные хакеры смогут развить и доработать эту программу - сократить ее размер, сделать генерируемые программы более насыщенными и оживленными, более "приспособленными" для русского языка - дерзайте, творите.

Если дело пойдет, то мы с удовольствием поддержим его развитие и в разделе "Возвращаясь к напечатанному" будем сообщать о новых идеях и приемах, открытых нашими читателями.

### **Структура системы.**

Система ABS довольно объемна. Листинг 1 - это сама программа ABS. Листинги 2...5 - демонстрационная адвентюрная программа, показывающая, как пользоваться программой ABS. В них содержится вся необходимая для корневой программы информация - количество комнат, количество объектов, описания комнат и т.п. Под "комнатами" здесь понимаются конечно не физические комнаты, а локации, т.е. участки, из которых состоит игровое пространство - это может быть и лес и горы и борт космического корабля и действительно комнаты.

Во время работы демонстрационная программа выдает на экран описание текущей локации (а может выводиться графика, если она есть) и список объектов, видимых в данном месте. Программа задает вопрос: "ВАШИ ДЕЙСТВИЯ?" и ждет от Вас команды. Ваша команда типа: "ВЗЯТЬ ФАКЕЛ" печатается на экране, затем анализируется сначала глагол, потом существительное и исполняется программой.

Главная программа (Листинг 1) содержит кодировку часто встречающихся действий,

например "С" или "ИДТИ С" означает "ИДТИ НА СЕВЕР".

### Листинги.

Порядок работы с листингами, их ввода и их выгрузки на ленту - довольно сложен, но если Вы аккуратно будете следовать всем указаниям, то сможете не только установить головной блок, но и ввести и запустить демонстрацию.

Программа русифицирована нашим традиционным способом с использованием утолщенного русско-латинского символьного набора в кодировке ASCII КОИ-7, сформированного в файл "chr" CODE.

Сначала введите Листинг\_1. Он сохраняется на ленте прямой командой RUN 9990. Это "Мастер-копия" головной программы. Вы будете ею пользоваться всякий раз, когда захотите создать новую игру, а пока спрячьте эту кассету в надежном месте, сейчас она нам больше не понадобится.

#### Листинг\_1. Программа "ABS". Мастер-копия. Файл "ABS".

```
1 GO TO 100
2 CLEAR 49999: LOAD "chr" CODE 50000
4 GO TO 0
5 GO TO 9990
8 POKE 23606,80: POKE 23607,194: RETURN
9 POKE 23606,0: POKE 23607,60: RETURN
40 CLS : POKE 33658,8: GO SUB 8
42 PRINT PAPER 0; INK 6; BRIGHT 1; AT 1,3;" ADVENTURE BUILDER SYSTEM "
44 RETURN
50 PRINT INK 2; BRIGHT 1; FLASH 1; AT 8,6;"ПОДОЖДИТЕ НЕМНОГО..."; AT 10,5;"ИДЕТ ГЕНЕРАЦИЯ
  ДАННЫХ."
52 RESTORE 50: FOR X=64008 TO 64047: READ A: POKE X,A: NEXT X
54 DATA 254,226,40,15,254,195,40,15,254,204,40,15,254,203,40,15,195,48,250,62,126,24,10,62,
  124,24,6,62,123,24,2,62,125,195,72,252,121,222,133,201
58 RESTORE 60: FOR X=64430 TO 65077: READ A: POKE X,A: NEXT X
60 DATA 205,190,251,62,21,50,60,250,24,33,58,137,92,254,2,192,62,3,50,137,92,229,6,0,205,0,
  14,225,201,58,137,92,71,62,24,144,254,22,40,216
61 DATA 50,60,250,33,61,250,6,55,54,0,35,16,251,33,61,250,54,1,35,54,1,35,54,0,33,106,92,54,
  0,62,22,215,58,60,250,215,62,0,215,62
62 DATA 62,215,62,22,215,58,60,250,215,58,61,250,215,60,50,61,250,62,42,215,1,0,0,205,61,31,
  205,191,2,33,8,92,78,62,13,185,202,181,252,62
63 DATA 12,185,202,132,252,62,8,185,202,132,252,58,61,250,254,32,202,48,250,121,254,32,40,
  13,254,230,212,44,250,254,128,210,8,250,79,24,9,79,58,61,250
64 DATA 254,3,218,48,250,65,62,22,215,58,60,250,215,58,61,250,61,215,60,50,61,250,120,245,
  215,241,42,62,250,17,63,250,25,119,42,62,250,35,34
65 DATA 62,250,17,50,0,33,200,1,205,181,3,195,243,251,58,61,250,254,3,218,48,250,61,50,61,
  250,62,22,215,58,60,250,215,58,61,250,215,61,50,61,250
66 DATA 62,32,215,42,62,250,1,62,250,9,54,0,58,62,250,61,50,62,250,195,120,252,58,61,250,
  254,3,218,48,250,62,22,215,56,60,250,215,58,61,250
67 DATA 61,215,62,32,215,17,25,0,33,250,0,205,181,3,33,63,250,54,32,35,229,6,31,126,254,48,
  56,3,35,16,248,225,17,100,250,52,31,144
68 DATA 6,0,184,250,251,252,126,18,35,19,16,250,24,17,79,71,126,18,35,19,16,250,62,0,145,71,
  175,19,18,16,252,33,93,250,6,31,126,254,0,32
69 DATA 3,43,16,248,126,254,32,40,3,43,16,248,5,72,6,0,33,64,250,9,6,0,17,107,250,126,18,35,
  19,16,250,33,0,0,17,100,250,6,0,26,190
70 DATA 32,10,35,19,16,248,126,50,114,250,24,17,72,6,0,9,35,62,199,190,40,2,24,225,33,114,
  250,54,200,33,0,0,17,107,250,6,0,26,190,32
71 DATA 10,35,19,16,248,126,50,115,250,24,17,72,6,0,9,35,62,199,190,40,2,24,225,33,115,250,
  54,201,58,115,250,71,62,0,184,216,33,24,251,58
72 DATA 115,350,6,0,79,9,62,1,190,192,58,115,250,6,0,128,50,115,250,201,33,0,0,6,0,54,0,35,
  16,251,17,0,0,6,0,58,24,251,33,124,251
73 DATA 190,40,4,35,16,250,201,79,62,0,144,18,19,35,121,24,243,33,0,0,6,0,54,0,35,16,251,17,
  0,0,6,0,62,99,33,124,251,190,40
74 DATA 4,35,16,250,201,79,62,0,144,18,19,35,121,24,243,17,7,0,33,0,0,6,0,58,24,251,190,40,
  3,25,16,250,58,115,250,71,35,16,253,126
```

```

75 DATA 50,115,250,201,33,24,251,17,130,250,1,150,0,237,176,201,33,130,250,17,24,251,1,150,
    0,237,176,201,33,24,251,6,150,54,0,35,16,251,1,24,251
76 DATA 62,0,2,33,99,0,9
80 RESTORE 80: FOR X=64048 TO 64059
82 READ Z: POKE X,Z: NEXT X
84 DATA 17,5,1,33,125,1,205,181,3,195,18,252
89 RETURN
100 BORDER 7: PAPER 7: INK 0: CLS : POKE 23658,8: GO SUB 8
105 GO SUB 40: GO SUB 50: GO SUB 40
106 FOR X=1 TO 11: BEEP .1,X: NEXT X
107 FOR X=10 TO 1 STEP -1: BEEP .1,X: NEXT X
110 PRINT AT 2,0;"
    ВВЕДИТЕ ПОЛНОЕ КОЛИЧЕСТВО ГЛА-
    ГОЛОВ, ИСКЛЮЧАЯ:
    ИДТИ, ВЗЯТЬ (БРАТЬ), ПОЛОЖИТЬ
    (БРОСИТЬ), ПРОВЕРЯТЬ, СМОТРЕТЬ,
    ИНВЕНТАРЬ (И), СТОП (КОНЕЦ), СО-
    ХРАНИТЬ, ЗАГРУЗИТЬ;
    А ТАКЖЕ БУКВ, ЗАДАЮЩИХ НАПРАВЛЕ-
    НИЕ: С(ЕВЕР), Ю(Г). В(ОСТОК),
    З(АПАД), (ВВЕР)Х, (В)Н(ИЗ) - ВСЕ
    ЭТО УЖЕ ВВЕДЕНО С ПОМОЩЬЮ КОДОВ:"
115 PRINT TAB 8 "ИДТИ.....0"
116 PRINT TAB 8 "ВЗЯТЬ.....1"
117 PRINT TAB 8 "ПОЛОЖИТЬ...2"
118 PRINT TAB 8 "ПРОВЕРИТЬ..3"
119 PRINT TAB 8 "СМОТРЕТЬ...4"
120 PRINT TAB 8 "ИНВЕНТАРЬ..5"
121 PRINT TAB 8 "КОНЕЦ.....6"
122 PRINT TAB 8 "СОХРАНИТЬ..7"
123 PRINT TAB 8 "ЗАГРУЗИТЬ..8"
125 INPUT V
126 LET V=V+28
127 GO SUB 40
128 PRINT INK 7; PAPER 2; BRIGHT 1; AT 5,2; "СИМВОЛЫ ИДЕНТИФИКАЦИИ КОМАНДЫ"
129 PRINT AT 8,0;"
    ВВЕДИТЕ КОЛИЧЕСТВО СИМВОЛОВ,
    КОТОРЫЕ СЛУЖАТ ДЛЯ СРАВНЕНИЯ
    ВВЕДЕННОГО ГЛАГОЛА С ОРИГИНАЛОМ."
130 PRINT INK 2; AT 13,4; "ВВЕДИТЕ ЧИСЛО ОТ 3 ДО 7"
131 INPUT VL
132 IF VL<3 OR VL>7 THEN BEEP .5,20: GO TO 127
133 LET VP=VL+1: LET VPP=VP+1
134 LET DAV=64000-(V*VP)
140 GO SUB 40
150 PRINT AT 8,0;"
    ВВЕДИТЕ КОЛИЧЕСТВО ОБЪЕКТОВ,
    КОТОРЫЕ БУДУТ ВХОДИТЬ В ИНВЕН-
    ТАРЬ ИГРЫ:"
155 INPUT MOB
156 IF MOB>50 THEN PRINT INK 2; AT 12,1; "ИЗВИНИТЕ, НО ЭТО СЛИШКОМ МНОГО!": BEEP .5,20: PAUSE
    50: GO TO 140
160 GO SUB 40
161 PRINT INK 7; PAPER 1; BRIGHT 1; AT 5,2; "СИМВОЛЫ ИДЕНТИФИКАЦИИ ОБЪЕКТА"
162 PRINT AT 8,0;"
    ВВЕДИТЕ КОЛИЧЕСТВО СИМВОЛОВ,
    ПО КОТОРЫМ СУЩЕСТВИТЕЛЬНОЕ СРАВ-
    НИВАЕТСЯ С ОРИГИНАЛОМ"
163 PRINT INK 1; AT 13,4; "ВВЕДИТЕ ЧИСЛО ОТ 3 ДО 7"
164 INPUT NL
165 IF NL<3 OR NL>7 THEN BEEP .5,20: GO TO 160
166 LET NP=NL+1: LET NPP=NP+1
170 GO SUB 40
171 PRINT AT 8,0;"    ВВЕДИТЕ МАКСИМАЛЬНУЮ ДЛИНУ ОПИСАНИЯ ОБЪЕКТА:"
175 INPUT L

```

```

176 IF L>32 THEN PRINT INK 2;AT 12,1;"МАКСИМАЛЬНАЯ ДЛИНА - ТОЛЬКО 32!": BEEP .5,20: PAUSE
    50:GO TO 170
179 LET AYAS=0
180 GO SUB 40
190 PRINT AT 8,0;"
    ВВЕДИТЕ ЧИСЛО ОБЪЕКТОВ, КОТО-
    РЫЕ МОГУТ ИЗМЕНЯТЬ СВОЕ ОПИСАНИЕ
    ПО ХОДУ ИГРЫ, НАПРИМЕР:" INK 1'TAB 5;"ФАКЕЛ - ГОРЯЩИЙ ФАКЕЛ" INK 0'"ЕСЛИ ЭТО НЕ
    ТРЕБУЕТСЯ, ВВЕДИТЕ 0"
195 INPUT MCOB
196 IF AYAS=0 AND MCOB>7 THEN PRINT INK 2; AT 19,5;"ВЫ АБСОЛЮТНО УВЕРЕНЫ?": BEEP .5,20:
    PAUSE 50: LET AYAS=1: GO TO 180
200 GO SUB 40
210 PRINT AT 8,0;"
    ВВЕДИТЕ ЧИСЛО ПРОЧИХ СЛОВ,
    КОТОРЫЕ ПО ВАШЕМУ ЖЕЛАНИЮ ДОЛЖНЫ
    РАСПОЗНАВАТЬСЯ ПРОГРАММОЙ:"
215 INPUT MON
220 LET N=MOB+MON+22
225 LET DAN=DAV-(N*NP)-5
230 GO SUB 40
235 PRINT AT 8,0;"
    ВВЕДИТЕ МАКСИМАЛЬНОЕ ЧИСЛО
    ОБЪЕКТОВ, КОТОРЫЕ ОДНОВРЕМЕННО
    МОЖНО ВЗЯТЬ:"
240 INPUT MAX
245 GO SUB 40
255 PRINT AT 8,0;"ВВЕДИТЕ КОЛИЧЕСТВО ЛОКАЦИЙ:"
260 INPUT NOR
265 GO SUB 40
270 PRINT PAPER 2; INK 7; BRIGHT 1;AT 5,8; "ФАКТОР СКРОЛЛИНГА"
272 PRINT AT 8,0;"
    ВЫ МОЖЕТЕ СДЕЛАТЬ ТАК, ЧТОБЫ
    НЕСКОЛЬКО ВЕРХНИХ СТРОК НЕ
    СКРОЛЛИРОВАЛИСЬ ПРИ ПЕЧАТИ НА
    ЭКРАНЕ ДЛИННЫХ СООБЩЕНИЙ. СКОЛЬ-
    КО СТРОК ОСТАВИТЬ НЕПОДВИЖНЫМИ,
    РЕШИТЕ САМИ."
273 PRINT '"ЕСЛИ ТАКОЙ ЭФФЕКТ НЕ НУЖЕН - 0"
277 INPUT SCRO
279 IF SCRO>15 THEN PRINT INK 2; AT 18,7; "ЭТО СЛИШКОМ МНОГО!": BEEP .5,20: PAUSE 50: GO TO
    265
280 LET SCRO=23-SCRO
285 GO SUB 40
290 PRINT AT 8,0;"В КАКОЙ ЛОКАЦИИ НАЧИНАЕТСЯ ИГРА?"
295 INPUT BEG
330 POKE 64821,DAV-256*INT (DAV/256): POKE 64822,INT (DAV/256)
335 POKE 64860,DAN-256*INT (DAN/256): POKE 64861,INT (DAN/256)
340 POKE 64903,MCOB: POKE 64924,(MOB-MCOB): POKE 64934, MOB: POKE 54944,MOB
345 POKE 64960,(MOB+1): POKE 64972,MOB: POKE 64997,(MOB+1)
350 POKE 64982,MOB: POKE 65012,MOB: POKE 64453,SCRO
355 POKE 65072, BEG
360 POKE 64750,VL: POKE 64772,VL: POKE 64827,VL: POKE 64810,NL:POKE 64866,NL
365 GO SUB 40
370 PRINT AT 6,0; "
    ЕСЛИ ВСЕ ДАННЫЕ ВЫ ВВЕЛИ КАК ПО-
    ЛОЖЕНО, НАЖМИТЕ <ENTER> И НАЧ-
    НЕТСЯ ГЕНЕРАЦИЯ СИСТЕМНОГО КОДА."
372 PRINT INK 2;AT 13,4; "ИНАЧЕ ВВЕДИТЕ <S> - STOP"
373 INPUT LINE A$
374 IF A$="S" THEN GO TO 9999
500 GO SUB 40
502 PRINT INK 2; BRIGHT 1; FLASH 1;AT 8,12; "ЖДИТЕ...";AT 10,5; " ИДЕТ ГЕНЕРАЦИЯ ДАННЫХ."
505 RESTORE 1000
506 DIM V$(V,VL)

```

```

508 FOR Y=1 TO V
510 READ V$(Y)
515 FOR X=1 TO VL
520 POKE (((DAV-VPP)+(Y*VP)+X)), CODE V$(Y,X)
525 IF CODE V$(Y,X)=32 THEN POKE (((DAV-VPP)+(Y*VP)+X)), 0
530 NEXT X
535 READ Z
540 POKE ((DAV-1)+(Y*VP)), Z
545 NEXT Y
550 POKE (DAV+(V*VP)), 199
600 DIM O$(MOB), L)
602 RESTORE 2000
603 DIM N$(MOB+NON+21), NL)
604 IF MCOB=0 THEN GO TO 670
605 FOR Y=1 TO MCOB
610 READ N$(Y)
615 FOR X=1 TO ML
620 POKE (((DAN-MPP)+(Y*NP)+X)), CODE N$(Y,X)
625 IF CODE N$(Y,X)=32 THEN POKE (((DAN-MPP)+(Y*NP)+X)), 0
630 NEXT X
635 READ Z
640 POKE ((DAN-1)+(Y*NP)), Z
645 READ B$
650 LET O$(Y)=B$
655 READ B$
660 LET O$(Y+MOB-MCOB)=B$
661 NEXT Y
662 FOR X=0 TO ((MCOB*NP)+1)
663 POKE (DAN+X)+((MOB-MCOB)*NP), PEEK (DAN+X)
665 NEXT X
670 FOR Y=(MCOB+1) TO (MOB-MCOB)
675 READ N$(Y)
680 FOR X=1 TO NL
685 POKE (((DAN-MPP)+(Y*NP)+X)), CODE N$(Y,X)
690 IF CODE N$(Y,X)=32 THEN POKE (((DAN-MPP)+(Y*NP)+X)), 0
695 NEXT X
700 READ Z
705 POKE ((DAN-1)+(Y*NP)), Z
710 READ B$
715 LET O$(Y)=B$
720 NEXT Y
730 FOR Y=(MOB+1) TO (MOB+NON+21)
735 READ N$(Y)
740 FOR X=1 TO NL
745 POKE (((DAN-MPP)+(Y*NP)+X)), CODE N$(Y,X)
750 IF CODE N$(Y,X)=32 THEN POKE (((DAN-MPP)+(Y*NP)+X)), 0
755 NEXT X
760 READ Z
765 POKE ((DAN-1)+(Y*NP)), Z
770 NEXT Y
775 POKE (DAN+(N*NP)), 199
780 RESTORE 3000
785 LET DMOV=DAN-(NOR*7)-5
787 LET DMOVL=DMOV-256*INT (DMOV/256): LET DMOVM=INT(DMOV/256)
788 POKE 65009, DMOVL: POKE 65010, DMOVM
790 FOR Y=DMOV TO (DMOV+(NOR*7)-1)
795 READ A: POKE Y, A: NEXT Y
800 RESTORE 4000
801 FOR X=65078 TO (65078+((MOB-1)*4)) STEP 4
802 POKE X, 35: POKE X+1, 62: POKE X+3, 119
803 READ A: POKE X+2, A
804 NEXT X
805 POKE 65078+(MOB*4), 201
810 LET PBS=65078+(MOB*4)+5
820 LET PBSL=FBS-256*INT (PBS/256): LET PBSM=INT (PBS/256)

```

```

825 POKE 64931,PBSL: POKE 64932,PBSM
826 POKE 64941,PBSL: POKE 64942,PBSM
827 POKE 64969,PBSL: POKE 64970,PBSM
828 POKE 64979,PBSL: POKE 64980,PBSM
860 FOR X=1 TO 11: BEEP .1,X: NEXT X
865 FOR X=10 TO 1 STEP -1: BEEP .1,X: NEXT X
867 REM USE BY BASIC =====
868 LET SAVE=DMOV-10: LET LEN=65368-SAVE
869 LET SAVED=SAVE-256*INT (SAVE/256): LET SAVED=INT (SAVE/256)
870 LET LENL=LEN-256*INT (LEN/256): LET LENM=INT (LEN/256)
871 RESTORE 874
872 FOR X=0 TO 9
873 READ Z: POKE (64116+X),Z: NEXT X
874 DATA SAVED, SAVED, MAX, MOV, LENL, LENM, PBSL, PBSM, DMOVL, DMOVM
875 GO SUB 40
885 PRINT AT 8,0;"
      ПОДГОТОВЬТЕСЬ К ВЫГРУЗКЕ НА
      ЛЕНТУ. НАЖМИТЕ <ENTER>, КОГДА
      БУДЕТЕ ГОТОВЫ."
886 PRINT INK 2; AT 12,0;"
      ЕСЛИ НЕ УВЕРЕНЫ, ЧТО ВСЕ ПРА-
      ВИЛЬНО, ТО ВВЕДИТЕ <S> - STOP."
887 INPUT A$
890 IF A$="S" THEN GO TO 9999
900 GO SUB 40
910 PRINT AT 8,11;"ЗАПИСЬ..."
915 SAVE "SYSTEM"CODE SAVE,LEN
920 SAVE "Objects" DATA O$( )
925 PRINT AT 8,0;"
      ПРЕЖДЕ, ЧЕМ БЕЙСИК-ПРОГРАММА
      И КОДЫ БУДУТ ЗАГРУЖЕНЫ В ПАМЯТЬ,
      ДОЛЖНО БЫТЬ ВЫПОЛНЕНО:" INK 2:TAB 10;"CLEAR ";SAVE-1
927 BEEP .5,20: PAUSE 50
930 PRINT PAPER 2; INK 7; BRIGHT 1;AT 15,0;"
      ОТМОТАЙТЕ ЛЕНТУ
      И ВЫПОЛНИТЕ ВЕРИФИКАЦИЮ
932 BEEP .5,20: PAUSE 50
935 PRINT AT 18,0;"
      НАЖМИТЕ ЛЮБУЮ КЛАВИШУ, КОГДА БУДЕТЕ ГОТОВЫ."
940 PAUSE 0
945 GO SUB 40
950 PRINT AT 8,9; "ВЕРИФИКАЦИЯ..."
955 VERIFY "system"CODE SAVE,LEN
960 VERIFY "objects" DATA O$( )
965 CLS
970 GO TO 9999
999 REM ГЛАГОЛЫ DATA 0-8 =====
1000 DATA "ИДИТИ",0,"СЕВЕР",0,"С",0,"ЮГ",0,"Ю",0,"ВОСТОК",0,"В",0,"НАПРАВО",0,"ЗАПАД",0,"З",
      0,"НАЛЕВО",0,"ВВЕРХ",0,"Х",0,"ВНИЗ",0,"Н",0,"ВЗЯТЬ",1,"БРАТЬ",1,"ПОЛОЖИТЬ",2,
      "БРОСИТЬ",2,"ПРОВЕРИТЬ",3,"ОСМОТРЕТЬ",3,"СМОТРЕТЬ",4,"ИНВЕНТАРЬ",5,"И",5,"СТОП",6,
      "КОНЕЦ",6,"СОХРАНИТЬ",7,"ЗАГРУЗИТЬ",8
1002 REM ДАННЫЕ ДЛЯ ПРОЧИХ ГЛАГОЛОВ (КОМАНД) =====
1999 REM ОБЪЕКТЫ, КОТОРЫЕ ИЗМЕНЯЮТ СВОЕ ОПИСАНИЕ В ХОДЕ ИГРЫ==
2019 REM ПРОЧИЕ ОБЪЕКТЫ =====
2189 REM ПРОЧИЕ СЛОВА. РАСПОЗНАВАЕМЫЕ ПРОГРАММОЙ =====
2190 DATA "СЕВЕР",MOV+1,"С",MOV+1,"ЮГ",MOV+2,"Ю",MOV+2,"ВОСТОК",MOV+3,"В",MOV+3,"НАПРАВО",
      MOV+3,"ЗАПАД",MOV+4,"З",MOV+4,"НАЛЕВО",MOV+4,"ВВЕРХ",MOV+5,"Х",MOV+5,"ВНИЗ",MOV+6,
      "Н",MOV+6,"СМОТРЕТЬ",99,"ИНВЕНТАРЬ",99,"И",99,"СТОП",99,"КОНЕЦ",99,"СОХРАНИТЬ",99,
      "ЗАГРУЗИТЬ",99
2999 REM ДАННЫЕ О ВОЗМОЖНЫХ ПЕРЕМЕЩЕНИЯХ =====
3999 REM ИСХОДНОЕ РАЗМЕЩЕНИЕ ОБЪЕКТОВ =====
9990 GO SUB 40: SAVE "ABS" LINE 2
9991 REM
9993 PRINT AT 8,0;"ОТМОТАЙТЕ ЛЕНТУ ДЛЯ ВЕРИФИКАЦИИ."
9994 BEEP .1,20: PAUSE 50

```

```

9995 PRINT AT 12,0;"НАЖМИТЕ ЛЮБУЮ КЛАВИШУ, КОГДА БУДЕТЕ ГОТОВЫ.": PAUSE 0
9996 GO SUB 40
9997 PRINT AT 8,7; "ВЕРИФИКАЦИЯ....."
9998 VERIFY "ABS"
9999 BORDER 7: PAPER 7: INK 0

```

Теперь дополните текст Мастер-копии строками данных для демонстрационной программы из Листинга\_2 или наберите их отдельно (и сохраните на ленте). В последней случае загрузите мастер-копию, объедините ее при помощи MERGE с данными для демонстрационной игры из Листинга\_2 и сохраните полученную программу командой RUN 9990. Эту выгрузку надо делать на рабочую ленту (назовем ее лента\_A).

#### Листинг\_2. Данные для демонстрационной игры. (Объединить с файлом "ABS")

```

1003 DATA "ОТКРЫТЬ",9,"ОТПЕРЕТЬ",10,"КОПАТЬ",11,"ВЫКОПАТЬ",11,"ЗАЖЕЧЬ",12,"ГАСИТЬ",13,
      "ПОГАСИТЬ",13,"ПОДНЯТЬСЯ",14,"ЗАЛЕЗТЬ",14,"ЛЕЗТЬ",14,"СПУСТИТЬСЯ",15,"СЛЕЗТЬ",15
1005 DATA "ЗАКРЫТЬ",16,"ШВЫРНУТЬ",17,"ЗАКОПАТЬ",18,"ЗАПЕРЕТЬ",19,"ОДЕТЬ",20,"НАДЕТЬ",20,
      "СНЯТЬ",21
1007 DATA "ПРЫГАТЬ",22,"СПРЫГНУТЬ",22,"ЗАПРЫГНУТЬ",22,"ИСПОЛЬЗОВАТЬ",23
2000 DATA "ПЛЕД",1,"ШЕРСТЯНОЙ ПЛЕД",1,"ПЛЕД, НАДЕТЫЙ НА ПЛЕЧИ"
2001 DATA "ФАКЕЛ",2,"МАЛЕНЬКИЙ ФАКЕЛ",2,"ГОРЯЩИЙ ФАКЕЛ"
2020 DATA "ШКАТУЛКА",3,"ДЕРЕВЯННАЯ ШКАТУЛКА"
2021 DATA "ЛОПАТА",4,"КОРОТКАЯ ЛОПАТА"
2022 DATA "КЛЮЧ",5,"БРОНЗОВЫЙ КЛЮЧ"
2023 DATA "ЩЕПКИ",6,"ДЕРЕВЯННЫЕ ЩЕПКИ"
2024 DATA "АЛМАЗ",7,"КРУПНЫЙ АЛМАЗ"
2191 DATA "ДВЕРЬ",16,"БУФЕТ",17,"ЛУЖАЙКА",18,"СТУПЕНИ",19,"ЛЕСТНИЦА",20
2192 DATA "ВОРОТА",21,"КОПАТЬ",22,"ЯМУ",22,"САД",23,"ПОЛЕ",24
2194 DATA "СЕЙФ",25,"ПЯТНО",26,"ПРЫГАТЬ",27
3001 DATA 1,0,3,0,0,0,0
3002 DATA 2,0,0,0,0,0,6
3003 DATA 3,1,7,0,0,0,0
3004 DATA 4,0,9,5,0,0,0
3005 DATA 5,0,10,0,4,0,0
3006 DATA 6,0,0,7,0,2,11
3007 DATA 7,3,13,8,6,0,0
3008 DATA 8,0,0,9,7,0,0
3009 DATA 9,4,0,10,8,0,0
3010 DATA 10,5,15,0,9,0,0
3011 DATA 11,0,0,0,0,6,0
3012 DATA 12,0,0,13,0,0,0
3013 DATA 13,7,0,14,12,0,0
3014 DATA 14,0,0,15,13,0,0
3015 DATA 15,10,0,0,14,0,0
4000 DATA 4,10,0,1,0,0,0,0

```

Теперь наберите загрузчик будущей демонстрационной игры - это Листинг\_3 и сохраните его командой RUN 200. Эту выгрузку делаем на новую ленту (Ленту\_B). Ленту назад не отматывайте.

#### Листинг\_3. Загрузчик демонстрационной игры. Файл "INTRO".

```

10 CLEAR 49999: LOAD "chr"CODE 50000
20 BORDER 7: PAPER 7: INK 0: CLS : POKE 23606,80: POKE 23607,194
30 PRINT INK 1; BRIGHT 1; AT 2,6;"ДЕМОНСТРАЦИОННАЯ ИГРА"
40 PRINT AT 4,11;"СОСТАВЛЕНА"" " С ИСПОЛЬЗОВАНИЕМ ПРОГРАММЫ"
50 PRINT PAPER 0; INK 6; BRIGHT 1; AT 7,3;" ADVENTURE BUILDER SYSTEM "
60 PRINT AT 9,5; "T.D. FROST + ""ИНФОРКОМ""
70 PRINT PAPER 0; INK 5; BRIGHT 1; AT 12,9;" ЗАГРУЗКА... "

```



```

80 PRINT INK 1;AT 15,0;"
      ЦЕЛЬ ЭТОЙ КОРОТКОЙ ПРОГРАММЫ
      ОЧЕНЬ ПРОСТА. ВАМ НАДО РАЗЫСКАТЬ
      КРУПНЫЙ АЛМАЗ И ЗАВЛАДЕТЬ ИМ."
90 PRINT AT 19,0; INK 7;
100 LOAD "adventure"
200 SAVE "INTRO" LINE 10

```

Сейчас наберите сам текст демонстрационной игры - Листинг\_4. Точнее сказать, это не вся программа, а только первая ее часть. Вторую - мы приведем в следующем выпуске "ZX-РЕВЮ" (то будет Листинг\_5). Но для упрощения отладки мы рекомендуем Вам и набирать две части программы по отдельности. Когда первая часть заработает, можно продолжить набор или догрузить вторую часть при помощи MERGE.

#### Листинг\_4 Текст первой части демонстрационной игры.

##### Файл "adventure".

```

45 LET R$(1)="НЕ ПОЛУЧАЕТСЯ."
50 FOR X=1 TO 4
52 IF R$(X,1) = "a" THEN LET X=5: INK 6: GO TO 195
54 RANDOMIZE USR SS: PRINT INK 5;R$(X): LET R$(X)="*"
56 NEXT X
58 GO TO 195
100 BORDER 0: PAPER 0: INK 6: BRIGHT 1: CLS
103 GO SUB 7000+PEEK F*10
104 IF PEEK F=11 AND PEEK (0+9)<>99 THEN GO TO 195
105 RANDOMIZE USR 64930
109 PRINT TAB 3; PAPER 1;"A"                                G"
110 PRINT TAB 3; PAPER 1; "A"; INK 7;" В ПОЛЕ ВАШЕГО ЗРЕНИЯ: ";INK 6;"G"
115 LET HE=0
120 FOR X=0 TO (MOB-1)
122 IF PEEK (PBS+X)=0 THEN LET X=MOB: GO TO 128
124 RANDOMIZE USR SS: PRINT TAB 3; PAPER 1; INK 6;"A"; INK 7; " ";0$(PEEK (PBS+X)); INK
    6;"G": LET HE=HE+1
126 NEXT X
128 IF HE=0 THEN PRINT TAB 3;PAPER 1; INK 6;"A"; INK 7;"НЕТ НИЧЕГО ИНТЕРЕСНОГО"; INK 6;"G"
129 PRINT TAB 3; PAPER 1;"CBBBBBBBBBBBBBBBBBBBBBBBBBBF": POKE 64453,PEEK 23689-1
195 RANDOMIZE USR SS: PRINT INK 4;"ВАШИ ДЕЙСТВИЯ?": BEEP .1,20
196 INK 6
200 RANDOMIZE USR 64459: PRINT
202 LET VB=PEEK 64114
204 LET NO=PEEK 64115
206 REM ANY SPECIALS? =====
208 IF PEEK F=11 AND PEEK (0+9)<>99 AND VB<>0 AND (VB<>12 AND VB<>13 AND VB<>14 AND VB<>6
    AND VB<>5 AND VB<>7 AND VB<>6) THEN LET R$(1)="ЗДЕСЬ НЕ ОБОЙТИСЬ БЕЗ СВЕТА!": GO TO
    50
228 IF VB=200 OR NO=201 THEN GO TO 45
235 GO TO 1000+VB*100
1000 IF NO>MOB+6 THEN GO TO 100
1002 LET NO=NO-MOB: POKE 64115,NO
1005 IF ((PEEK F=8 AND NO=3) OR (PEEK F=9 AND NO=4)) AND PEEK (F+4)=0 THEN LET R$(1)="ДВЕРЬ
    ЗАКРЫТА.": GO TO 50
1010 IF ((PEEK F=14 AND NO=3) OR (PEEK F=15 AND NO=4)) AND PEEK (F+7)=0 THEN LET
    R$(1)="ВОРОТА ЗАКРЫТЫ.": GO TO 50
1020 IF (PEEK F=7 AND PEEK (F+5)=1) OR (PEEK F=15 AND PEEK (F+6)=1) THEN CLS : PRINT AT
    10,16;FLASH 1;"*": GO SUB 8800: CLS : PRINT AT 2,1;
    "ВЫ НЕОЖИДАННО СВАЛИЛИСЬ В ЯМУ.":AT 3,1;
    "КОТОРУЮ САМИ ЖЕ ЗДЕСЬ ВЫКОПАЛИ"
    ;AT 4,1;"И СЛОМАЛИ НОГУ.":
    BEEP 1.2,28: PAUSE 100: BEEP .5,15: GO TO 1602
1090 RANDOMIZE USR 65005
1092 IF PEEK 64115=0 THEN LET R$(1)="ТУДА ВАМ НЕ ПРОЙТИ.": GO TO 50
1093 POKE F,PEEK 64115
1094 GO TO 100

```

```

1800 LET R$(1)="В СЛЕДУЮЩЕМ ВЫПУСКЕ ""ZX-РЕВЮ""": GO TO 50
1900 IF (PEEK F=8 OR PEEK F=9) AND NO=16 AND PEEK (F+4)=0 THEN POKE (F+4),1: LET R$(1) =
    "О.К. ТЕПЕРЬ ДВЕРЬ ОТКРЫТА.": GO TO 50
1902 IF PEEK F=14 AND NO=21 AND PEEK (F+7)=0 THEN LET R$(1)="НЕ ОТКРЫВАЮТСЯ, ЗАКРЫТЫ
    ИЗНУТРИ.": GO TO 50
1905 IF PEEK F=15 AND NO=21 AND PEEK (F+7)=0 THEN POKE (F+7),1: LET R$(1)="О.К. ТЕПЕРЬ
    ВОРОТА ОТКРЫТЫ.": GO TO 50
1910 IF (((PEEK F=8 OR PEEK F=9) AND PEEK (F+4)=1 AND NO=16) OR (((PEEK F=14 OR PEEK F=15)
    AND PEEK (F+7)=1) AND NO=21)) THEN LET R$(1)="ДА ВЕДЬ И ТАК УЖЕ ОТКРЫТО!": GO TO 50
3315 LET R$(1)="В СЛЕДУЮЩЕМ ВЫПУСКЕ ""ZX-РЕВЮ""": GO TO 50
7010 PRINT T$
7011 PRINT "
    А ВЫ НАХОДИТЕСЬ В ЧИСТОМ И АККУ-Г
    АРАТНОМ АМБАРЕ. ФЕРМЕР, КОТОРО-Г
    АМУ ОН ПРИНАДЛЕЖИТ, ПО-ВИДИМОМУ Г
    АЛЮБИТ СВОЕ ДЕЛО. G"
7012 PRINT B$
7019 RETURN
7020 PRINT T$
7021 PRINT "
    А ЗАБРАВШИСЬ ПО ЛЕСТНИЦЕ НАВЕРХ, Г
    А ВЫ ПОПАЛИ КАК БЫ НА ОТКРЫТЫЙ Г
    АЧЕРДАК. СКОРЕЕ ЕГО МОЖНО ОПИ-Г
    АСАТЬ, КАК БОЛЬШОЕ ХРАНИЛИЩЕ Г
    АНАД КОМНАТОЙ. G"
7022 PRINT B$
7029 RETURN
7030 PRINT T$
7031 PRINT "
    А ВЫ ВЫШЛИ НА ДВОР ФЕРМЫ. НИКА-Г
    АКИХ ЖИВОТНЫХ НЕ ВИДНО. НА СЕ-Г
    АБЕРЕ НАХОДИТСЯ АМБАР, НА ЮГЕ -Г
    А-ПОЛЕ. G"
7032 PRINT B$
7039 RETURN
7040 PRINT T$
7041 PRINT "
    А УЮТНАЯ ГОСТИННАЯ КРЕСТЬЯНСКОГО Г
    АДОМА. ВЫ ЧУВСТВУЕТЕ НА СЕБЕ Г
    АЧЕЙ-ТО ВЗГЛЯД. G"
7042 PRINT B$
7049 RETURN
7050 PRINT T$
7051 PRINT "
    А КАЖЕТСЯ, ЧТО ЭТОЙ СТОЛОВОЙ Г
    АКОМНАТОЙ ДАВНО НИКТО НЕ ПОЛЬ-Г
    АЗОВАЛСЯ. НАВЕРНОЕ, ЗДЕСЬ НИКТО Г
    АНЕ ЖИВЕТ. G"
7052 PRINT B$
7059 RETURN
7060 PRINT T$
7061 PRINT "
    А КОЮШНЯ СОВЕРШЕННО ПУСТА. НЕ Г
    АДАЖЕ ХАРАКТЕРНОГО ЗАПАХА ЖИ-Г
    АВОТНЫХ. ЛЕСТНИЦА ВЕДЕТ НА ЧЕР-Г
    АДАК, ВИДНЫ ТАКЖЕ СТУПЕНИ, Г
    АСКРЫВАЮЩИЕСЯ ГДЕ-ТО ВНИЗУ, В Г
    АТЕМНОТЕ. G"
7062 PRINT B$
7069 RETURN
7070 PRINT T$
7071 PRINT "
    А ВЫ ВЫШЛИ В ОТКРЫТОЕ ПОЛЕ. ОНО Г
    АБЫЛО НЕДАВНО ВСПАХАНО, НО НИ-Г

```

АЧЕГО ПОСАЖЕНО НЕ БЫЛО. МОЖНО  
 АИДИ В ЛЮБОМ НАПРАВЛЕНИИ. G“

7072 PRINT B\$  
 7079 RETURN  
 7080 PRINT T\$  
 7081 PRINT “  
 АВЫ НАХОДИТЕСЬ У ДВЕРИ ФЕРМЕР-Г  
 АСКОГО ДОМА. ДОМ СДЕЛАН ОЧЕНЬГ  
 АДОБРОТНО И КРЕПКО, КАК БУДТОГ  
 АХОЗЯЕВАМ БЫЛО ОТ ЧЕГО СКРЫ-Г  
 АВАТЬСЯ. G“

7082 PRINT B\$  
 7089 RETURN  
 7090 PRINT T\$  
 7091 PRINT “  
 АПЕРЕД ВАМИ НЕБОЛЬШАЯ ПРИХОЖАЯ.Г  
 АВ УГЛУ СТОИТ БУФЕТ И БОЛЬШЕГ  
 АНИЧЕГО ЗДЕСЬ НЕТ. НА ВОСТОКЕГ  
 АКУХНЯ, НА СЕВЕРЕ - ГОСТИННАЯ. G”

7092 PRINT B\$  
 7099 RETURN  
 7100 PRINT T\$  
 7101 PRINT “  
 АКУХНЯ СВЕРКАЕТ БЕЛИЗНОЙ И ЧИ-Г  
 АСТОТОЙ. В ВОЗДУХЕ ОПРЕДЕЛЕННОГ  
 АЧУВСТВУЕТСЯ ЗАПАХ ЧЕГО-ТО ПРИ-Г  
 АЯТНОГО. G”

7108 PRINT B\$  
 7109 RETURN  
 7110 IF PEEK (0+9)<>99 THEN PRINT TAB 5; PAPER 7; INK 0;”ЗДЕСЬ НИЧЕГО НЕ ВИДНО! “: RETURN  
 7111 PRINT T\$  
 7112 PRINT “  
 АВ ОТЛИЧИЕ ОТ БОЛЬШИНСТВА ОБЫЧ-Г  
 АНЫХ ПОГРЕБОВ, ЭТОТ - ЯВНО НЕГ  
 АИСПОЛЬЗУЕТСЯ ДЛЯ ХРАНЕНИЯ ВСЯ-Г  
 АКИХ ЗАПАСОВ. А В УГЛУ МОЖНОГ  
 АРАЗГЛЯДЕТЬ БОЛЬШОЙ СТАРОМОДНЫЙГ  
 АСЕЙФ СО ВСТРОЕННЫМ В НЕГОГ  
 АБРОНЗОВЫМ ЗАМКОВ. G”

7113 PRINT B\$  
 7119 RETURN  
 7120 PRINT T\$  
 7121 PRINT “  
 АЭТО ТУПИК. МОЖНО БЫЛО СЮДА ВО-Г  
 АОБЩЕ НЕ ПРИХОДИТЬ. G”

7122 PRINT B\$  
 7129 RETURN  
 7130 PRINT T\$  
 7131 PRINT “  
 АЭТА ОЧЕНЬ УЗКАЯ ДОРОГА ВЕДЕТ СГ  
 АВОСТОКА НА ЗАПАД. НО, КАЖЕТСЯ,Г  
 АМОЖНО ПРОЙТИ И НА СЕВЕР. G”

7132 PRINT B\$  
 7139 RETURN  
 7140 PRINT T\$  
 7141 PRINT “  
 АДорога УПИРАЕТСЯ В ВОРОТА, ЗАГ  
 АНИМИ РАСПОЛОЖЕН ПРИУСАДЕБНЫЙГ  
 АСАД. G“

7142 PRINT B\$  
 7149 RETURN  
 7150 PRINT T\$  
 7151 PRINT “  
 АЭТО ДОВОЛЬНО ПУСТЫННЫЙ САД,Г  
 АКоторый СКОРЕЕ ПОХОЖ НА ЛУЖАЙ-Г

```

АКУ С НЕСКОЛЬКИМИ ЦВЕТЧНЫМИГ
АКЛУМБАМИ. ЗДЕСЬ БЫЛО БЫ СОВСЕМО
АКРАСИВО, ЕСЛИ БЫ НЕ БЕСЦВЕТНОЕО
АВЫТОПТАННОЕ ПЯТНО ПЕРЕД ВОРО-О
АТАМИ, ВЕДУЩИМИ НА ЗАПАД.      О“
7152 PRINT B$
7159 RETURN
8800 FOR X=-5 TO 15
8802 BEEP .02,X: NEXT X
8804 FOR X=14 TO -5 STEP -1
8806 BEEP .02,X: NEXT X
8808 RETURN
9940 REM СТАРТ ИГРЫ =====
9950 LOAD "system".CODE
9955 LOAD "objects" DATA O$( )
9956 LET F=64280: LET O=64379: LET SS=64440
9957 LET MAX=PEEK 64118: LET MOB=PEEK 64119
9958 LET PBS=256*PEEK 64123+PEEK 64122
9959 LET SAVE=256*PEEK 64117+PEEK 64116: LET LEN=256*PEEK 64121+PEEK 64120
9960 LET DMOV=256*PEEK 64125+PEEK 64124
9961 POKE 64130,O
9965 DIM R$(4,32)
9970 FOR X=1 TO 4: LET R$(X,1)="*": NEXT X
9975 BORDER 0: PAPER 0: INK 6: BRIGHT 1: CLS
9976 PRINT AT 8,10; "ПОДГОТОВКА..."
9980 RANDOMIZE USR 65058
9984 RESTORE 9987
9985 POKE 23675,88: POKE 23676,255: FOR X=USR "A" TO USR "A"+63
9986 READ Z: POKE X,Z: BEEP .005,20: NEXT X
9987 DATA 128,128,128,128,128,128,128,128,128,0,0,0,0,0,0,255,128,128,128,128,128,128,128,255,
        255,128,128,128,128,128,128,128,128,255,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,255,1,1,1,1,1,1,1,1,
        255,0,0,0,0,0,0,0,0
9988 LET T$="ДННННННННННННННННННННННННННННННННННННЕ": LET V$="CBVVVVVVVVVVVVVVVVVVVVVVVVVVVVVBF":
    LET VB=1
9989 GO TO 100
9990 SAVE "adventure" LINE 9940
9991 CLS
9993 PRINT BRIGHT 1:AT 8,0;"ОТМОТАЙТЕ ЛЕНТУ ДЛЯ ВЕРИФИКАЦИИ."
9994 BEEP .1,20: PAUSE 50
9995 PRINT AT 12,0;"    НАЖМИТЕ ЛЮБУЮ КЛАВИШУ, КОГДА БУДЕТЕ ГОТОВЫ.": PAUSE 0
9996 VERIFY "adventure"
9999 BORDER 7:PAPER 7:INK 0

```

Набрав первую часть демонстрационной программы, сохраните ее командой RUN 9990 на ленту В (сразу же за Листингом 3), и ленту назад не отматывайте.

Пока с вводом программ достаточно, а сейчас некоторые пояснения к листингам 1-4.

## Работа с листингами.

Загрузите программу с ленты\_A, запустите ее и в ответ на запросы программы введите: 23,4,9,4,22,2,13,2,15,0,7 (немного ниже мы дадим подробные комментарии к этим числам). Так закончится подготовка необходимых данных, программа сама укажет Вам, когда надо произвести выгрузку блоков кодов и данных - выгружаются 2 блока. Эту отгрузку надо произвести на ленту В, (следом за листингами 3 и 4).

Теперь мы будем работать с Лентой\_В. Отмотайте ее в начало и загрузите. После окончания загрузки основной Бейсик-программы демонстрационной игры (Лист\_4). Она стартует со строки 9940, загружая блок кодов и данные. После этого начнется подготовка данных, о чем свидетельствует табличка на экране, а затем игра начинается.

Вы уже можете начать исследование игровой ситуации. Можете пройти по всем локациям, все посмотреть, на Вас никто не будет нападать, правда Вы не сможете делать никаких действий, кроме "ОТКРЫТЬ", но этого должно быть достаточно для того, чтобы отладить БЕЙСИК-программу (листинг 4) и удалить из нее все ошибки, допущенные при

наборе.

### **Головная программа ABS.**

После старта головная программа ABS выдает серию экранных запросов, по которым надо ввести необходимые данные. Проследите по демонстрации, как и в каком формате вводятся данные по этим запросам, а мы сейчас их подробно рассмотрим. После каждого заголовка раздела стоит число в скобках, обозначающее, что должно вводиться на этот запрос в демонстрации.

1. Количество прочих глаголов. (23). Эти глаголы заносятся в строки DATA 1003...1999 (Листинг\_2) Например:

```
DATA "ОТКРЫТЬ", 9, "ОТПЕРЕТЬ", 10
```

Формат: глагол (полностью) + присвоенный ему номер. Поскольку номера с 0...по 8 уже присвоены стандартным командам, то первая пользовательская команда (глагол) начинается с номера 9. Слова синонимы, такие как ГАСИТЬ, ПОГАСИТЬ, имевшие одинаковое действие, но разное написание, вводятся порознь, но им присваивается один и тот же номер. Тем не менее, в ответ на запрос "Количество глаголов?" их надо считать как отдельные слова.

В системе ABS применяется следующий формат ввода команды: первое слово считается глаголом, последнее - существительным или направлением (в случае перемещения). Можно в отдельных случаях ввод осуществлять и одним словом (буквой), но для реализации этого надо, чтобы это слово (буква) было занесено и в список глаголов и в список существительных (мы еще вернемся к этому моменту позднее).

Основные глаголы заданы в строке 1000 Мастер-копии программы ABS (Листинг\_1). Основные существительные - в строке 2190 этого же листинга.

Такие действия, как ИДТИ НА ВОСТОК, можно вводить также и ИДТИ НА В, или ИДТИ В, или ВОСТОК, или только буквой В, а также словами ИДТИ НАПРАВО или просто НАПРАВО. При помощи одного слова можно также проверить ИНВЕНТАРЬ (или просто И), ввести СТОП и другие - те, которые одновременно заданы и в строке 1000 и в строке 2190. А вот глаголы взять или ПОЛОЖИТЬ не могут использоваться отдельно, без существительного и, поэтому, они не заданы в строке 2190.

На тот случай, если Вы захотите изменить основные глаголы, заданные в Листинге\_1, в строке 1000, обращаем Ваше внимание на то, что общее количество глаголов в строке 1000 отражено также в строке 126 этого листинга. Это число - 28. Если будете изменять данные в строке 1000, проконтролируйте и число в строке 126. Аналогично и для строки 2190 - полное число слов в этой строке отражено числовыми параметрами в строках 603 и 730. Это число - 21.

2. Символы идентификации команды (4). Для того, чтобы пользователю не приходилось полностью набирать длинные слова (и не дай Бог при этой сделать ошибку), программа может идентифицировать команду не по всему слову, а по нескольким первым символам. Например, вместо команды прыгать можно дать команду ПРЫГ (если установлено число 4) или ПРЫ (если установлено 3). Это число может быть задано в диапазоне от 3 до 7. Обычное число - 4 или 5. Для детей - 4, для взрослых программы посложнее, можно задать 5, так как при большом числе слов возможны ситуации, когда первые 4 буквы разных слов совпадают. Для числа 5 - это происходит значительно реже. Число 3 еще как-то приемлемо для несложных английских программ, но в русском языке существуют приставки, которые могут быть одинаковыми у разных слов и очень возможны ошибки.

3. Количество объектов (9). Это количество объектов в программе, которые могут быть инвентарем, т.е. которые пользователь может брать, переносить с места на место или как-то использовать. Подробности, как заносить список этих объектов в листинги, смотрите в комментариях к пункту 6.

4. Символа идентификации объекта (4). Это то же, что и п.2, но для второй части команды - для существительных или направления. Эти числа могут и различаться, но лучше, чтобы сокращения для глаголов и существительных делались по единым правилам, пользователь к ним проще привыкает и чаще пользуется.

5. Максимальная длина описания объекта (22). Это количество символов, которое может занимать самое длинное описание объекта (например, ПЛЕД, НАДЕТЫЙ НА ПЛЕЧИ).

6. Количество объектов, способных изменять свое описание по ходу игры (2). Этот пункт имеет отношение к объектам типа "ФАКЕЛ" - "ГОРЯЩИЙ ФАКЕЛ" и т.п. Данные на эти объекты заносятся в строки 2000 ... 2018 (см. Листинг\_2 демонстрационной программы), например:

```
DATA "ПЛЕД", 1, "ШЕРСТЯНОЙ ПЛЕД", "ПЛЕД, НАДЕТЫЙ НА ПЛЕЧИ"  
DATA "ФАКЕЛ", 2, "МАЛЕНЬКИЙ ФАКЕЛ", "ГОРЯЩИЙ ФАКЕЛ"
```

Это были объекты 1 и 2, а их измененные версии будут потом названы объектами 8 и 9. Поэтому при ответе на запрос о количестве предметов, являющихся инвентарем, надо задавать число 9, хотя в строках 2000-2024 заданы только 7 предметов. Кроме строк 2000 и 2001 (объекты, которые могут изменяться), остальной инвентарь, который не может изменять свое состояние, задан в строках 2020-2188 в формате:

```
DATA "ШКАТУЛКА", 3, "ДЕРЕВЯННАЯ ШКАТУЛКА"
```

7. Другие слова, распознаваемые программой (13). Эти слова заданы в строках с 2191. Первый присваиваемый им номер рассчитывается по формуле:

(количество объектов) + 7

Поэтому в демонстрационной программе первое слово дверь имеет номер 16.

8. Максимальное число объектов, которые одновременно можно взять и иметь при себе (2).

9. Общее число локаций (15).

10. Фактор скроллинга (0). Если Вы хотите, чтобы определенное количество верхних строк не скроллировалось при вводе команд, то можете до запросу задать это число. В демонстрационном примере это не используется, поэтому введен 0.

11. Стартовая локация (7). Это номер локации, в которой начинается игра.

### **Возможные перемещения.**

Со строки 3000 (до строки 3999) в Листинге\_2 находится таблица расположения локаций, она же определяет и возможность взаимных перемещений между локациями. Первое число в строке DATA - это номер локации. Всего их 15 (в данной демонстрационной программе). Их расположение и наименование для демонстрационной программы показано на плане (рис.1).

Формат задания допустимых перемещений следующий. Первая цифра в строке DATA - это номер локации. Далее идут номера других локаций, в которые можно попасть из данной, если идти в разных направлениях. Всего 6 направлений, по-порядку соответственно СЕВЕР, ЮГ, ВОСТОК, ЗАПАД, ВВЕРХ и ВНИЗ. Так, например, для 1 локации возможен только переход в 3 локацию при движении на ЮГ:

```
DATA 1, 0, 3, 0, 0, 0, 0
```

а для 7 локации - допустимы 4 направления:

```
DATA 7, 3, 13, 8, 6, 0, 0
```

в 3 локацию, если идти на СЕВЕР,

В 13 -//- -//- ЮГ.

В 8 -//- -//- ВОСТОК.

В 6 -//- -//- ЗАПАД

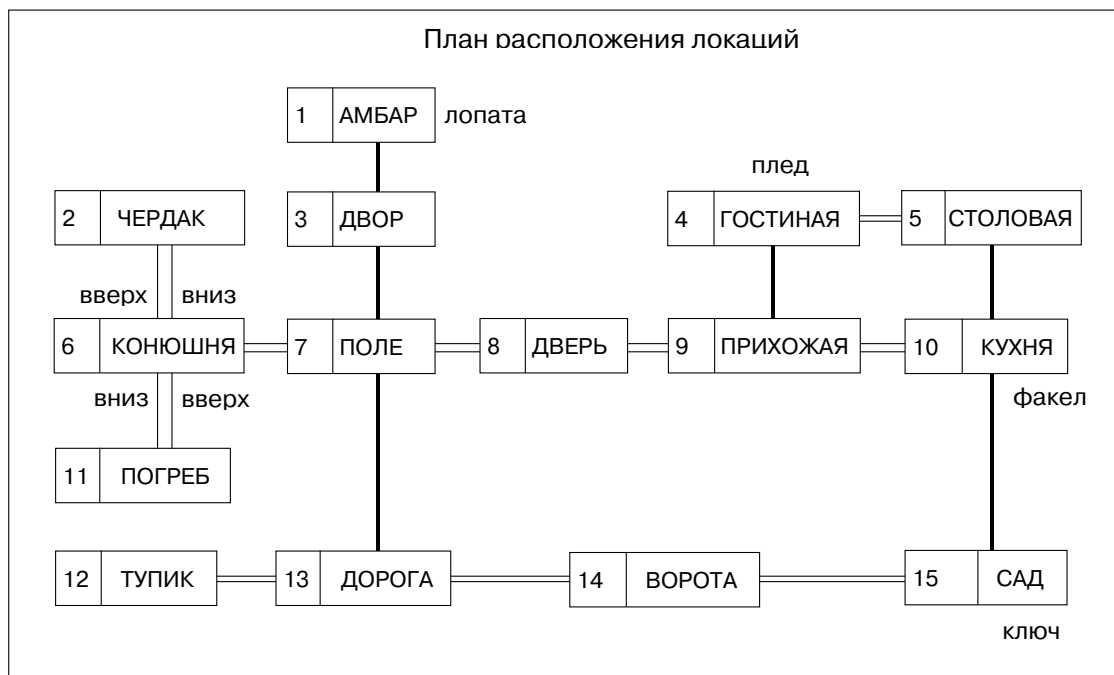


Рис. 1

### Расположение объектов.

И, наконец, в строке 4000 задается исходное расположение объектов, являющихся инвентарем. Формат записи такой. По порядку, начиная с объекта номер 1 (ПЛЕД), задаются номера локаций, в которых этот предмет находится. Если предмет в данный момент не виден, то заносится 0. Всего в строке DATA будет столько цифр, сколько задано объектов, являющихся инвентарем (для нашей демонстрационной программы - 9).

DATA 4, 10, 0, 1, 0, 0, 0, 0, 0

Это означает, что ПЛЕД расположен в 4 локации, ФАКЕЛ - в 10 локации, ШКАТУЛКА - пока не видна и т.д., для всех 9 предметов, являющихся инвентарем (включая и те, которые изменяют свое состояние в процессе игры). Если Вы захотите, чтобы вначале игры какой-нибудь объект находился у Вас в руках, то надо для него задать номер 99 - это будет соответствовать "объект взят".

### Загрузчик.

Листинг\_3 является загрузчиком. Его функции - переустановка RAMTOP, загрузка русского символьного набора и его активизация, а также загрузка основной Бейсик-программы демонстрационной игры (Листинг\_4).

### Демонстрационная программа.

Автостарт программы происходит со строки 9940, где после загрузки кодового блока и блока данных, начинается задание параметров, сохраненных в кодовом блоке, задаются массивы для переменных и т.д. Затем задаются символы UDG-графики А-Н. Последние служат для изображения рамок, оконтуривающих текст. Они встречаются в тексте программы в кавычках. Весь текст за редким исключением (символы "Y" или "N") набирается по-русски, а встречающиеся символы с "А" по "Н" (например, в строке 9988, строки с 7000 и др.) - это символы UDG-графики, изображающие рамки.

Подготовка исходных данных заканчивается строкой 9989, где управление передается на начало исполнения программы - строку 100. В процессе отладки, когда Вы будете останавливать программу, вновь стартовать ее можно, подав команду GO TO 100, но не RUN 100.

(Окончание - в следующем номере)

# Компьютерная новелла

(С) Матвеев Ю.А., 1993 г

## СТРАТЕГИЯ КАПИТАНА КРЕНОНА

(по мотивам программы REBEL STAR)

### Часть 2. Столкновение

Они смотрели на белый, ослепительно яркий диск Венеры, который занимал добрую половину обзорного экрана. У самой кромки диска, в фиолетовой дымке, гигантским алмазом сверкал в солнечных лучах орбитальный комплекс "Венера-Центр", окруженный целой флотилией грузовых и пассажирских кораблей разнообразных типов и марок.

Полет близился к концу. Через час космический лайнер "Викинг" должен был достичь границы действия гравитационного поля комплекса. Кренон сидел в пол-оборота к экрану и часто поглядывал на внука, наблюдая за его реакцией на великолепный космический пейзаж, созданный природой и человеком.

- Здорово! - не отрывая глаз от экрана с восхищением сказал юноша. - Это не то, что на наших тренажерах в школе.

Кренон похлопал внука по плечу:

- Ничего, учись пока, скоро сам летать будешь.

- Скорей бы, - вздохнул будущий пилот сверхскоростных звездных кораблей. Он развернулся в кресле и хитро прищурился. - Дед, мы уже скоро прилетим, а ты еще не рассказал мне про вашу встречу с инопланетянами.

Кренон посмотрел на часы. Через два часа они будут на Венере, в гостях у его друга детства, Дона Кейлони, и тогда ему трудно будет найти свободную минуту, чтобы спокойно рассказать внуку о той трагической встрече с коварными звездными монстрами.

Это случилось во времена великих завоеваний человечества. Расширялись границы обитания, открывались новые миры, строились огромные звездные лайнеры, способные переносить все большее количество землян на дальние космические колонии. Человеческая цивилизация заняла достойное место во Вселенной и была одной из самых молодых и быстрорастущих. Космическое пространство на десятки парсеков от Солнца было досконально изучено и тщательно исследовано, белых пятен оставалось все меньше, однако...

\* \* \*

Эта первобытная планета ему сразу не понравилась. Вполне подходящая для жизни, она по каким-то не вполне понятным причинам, еще долгое время оставалась неизученной. Странно, что Земля, испытывающая постоянную нехватку пригодных для жизни планет, так вяло проводила исследования на Зеро. В название планеты первооткрыватели вложили двойной смысл: с одной стороны этот мир некоторое время действительно был перевалочной базой для крупных межзвездных экспедиций, своего рода "точкой отсчета" новых завоеваний. С другой стороны, не внушающие оптимизма скудная растительность и серенький животный мир, еле умещавшиеся на единственной материке, который скорее походил на остров, омываемый со всех сторон холодным океаном. Планета не оправдала надежд космобиологов и серьезные исследователи с разочарованием умывали руки.

Один единственный город, построенный колонистами, располагался на юге материка. Там же находился и космопорт. Север, из-за своей труднодоступности, был малоизучен и здесь оставались обширные земли, где еще не ступала нога человека.

Периодически из числа колонистов формировались экспедиции для изучения природы материка, которые с каждым разом уходили все дальше и дальше на Север. Жизнь на Зеро текла своим чередом и, казалось, ничто не могло поколебать ее спокойствие и размеренность. Однако вскоре произошло непредвиденное.



В один ясный, холодный день прервалась связь с экспедицией профессора Джонсона, который был ведущим специалистом по флоре и фауне дикого Севера планеты. Спасательный отряд, собранный наспех из добровольцев и в срочном порядке отправленный на поиски, бесследно исчез на следующие сутки. Тогда руководство колонии и обратилось за помощью в Зональный Центр Космической Безопасности.

Капитан Кренон возглавил десантный отряд из шестнадцати человек. В его подчинении были неплохо подготовленные десантники. Многие из них уже прошли хорошую закалилку в поисковых и спасательных операциях.

Получив задание, отряд капитана Кренона немедленно приступил к его завершению. Город выделил десанникам легкий винтолет, на котором сразу из космопорта их доставили в район поиска пропавшей экспедиции.

Надо сказать, что северная часть материка буквально утопала в смердящих болотах, а небольшие участки суши покрывали непроходимые заросли ползучих лиан и болотного тростника. Иногда здесь можно было встретить не очень высокие пальмообразные деревья с толстыми сочными листьями ядовито-фиолетового цвета. Живности здесь было немного, если не считать болотных крыс размером с корову, да местных крокодилов с длинными электрическими щупальцами. И с теми и с другими встреча не сулила ничего хорошего: острыми, словно лезвие бритвы, клыками, болотные крысы могли, пожалуй, перегрызть стальной канат в руку толщиной, а об электрических крокодилах, которых колонисты называли свамперами, и говорить не приходилось.

Отряд шел цепью. Вот уже вторые сутки, используя светосабли, они продирались сквозь заросли лиан. От сырого воздуха, наполненного зловониями болотных испарений, мутило, а ноги деревенели от постоянного сопротивления вязкой хлюпающей жижи.

В два часа пополудни Кренон связался с Центром и доложил о результатах первых суток поисков. Получив необходимые инструкции он решил дать отдохнуть уставшим десанникам.

Отряд расположился на небольшой поляне, окруженной со всех сторон густым кустарником. На привале они обменялись мнениями. Рыжеволосый Зира Кама скептически относился к приказу центра продолжить поиски. Он протирал забрызганное грязью лазерное ружье и угрюмо бубнил что-то про свамперов и крыс. В той или иной степени его поддерживали еще трое: Джекки Доннелли, Лазарь Шарки и дон Кейлони.

- Им там легко рассуждать, - ворчал Шарки. - Пойдите туда, пойдите сюда... Пусть сами подышат местным ароматом. Этого профессора наверняка давно сожрали свамперы, а мы тут ищем сами не знаем что.

- Да ладно тебе, - одернул его Дон Кейлони, уловив мрачный взгляд Кренона. - Скоро вернемся, не вечно же нам здесь блуждать. Правда, капитан?

Кренон многозначительно посмотрел на Дона и, расстегивая крючки комбинезона, повернулся к Диане Вудлей - врачу отряда:

- Больные есть?

- Все здоровы, - ответила Диана.

- Тогда какого дьявола мне здесь закатывают истерику? - вскипел капитан, поднимаясь с надувного походного кресла, давая тем самым понять, что любые споры закончены. - Значит так. Здесь разбиваем постоянный лагерь. Со мной продолжат поиски десять человек. Остальные, включая врача, остаются в лагере до нашего возвращения. Вопросы?..

- Кто остается? - спросил Люк Соло.

Что? Я разве не сказал? - Кренон задумался. - В лагере останутся капрал Хикс, Дэниел Дар, Хадсон, Диана Вудлей и Лазарь Шарки. Займетесь благоустройством лагеря. Ну и ужин хороший приготовьте. Мы к вечеру вернемся.

- Смотрите только сами себя свамперам не скормите, - добавил любимец публики Хенли, разряжая обстановку.

Кренон оставлял в лагере самых молодых десантников, рассчитывая лишь на опытных партнеров.

Спустя полчаса группа из одиннадцати человек продолжила поиски. Тогда они еще не

знали, что в лагерь им не суждено будет вернуться, а некоторые из них и вовсе навсегда останутся в этих проклятых Богом болотах.

Когда в воздухе прогремели выстрелы, разрывая привычную тишину, Кренон остановился и прислушался. Шедший рядом с ним Дон Кейлони ловким движением перевел лазерное ружье в боевое положение и вопросительно посмотрел на него. Командир десантников включил рацию:

- Кто стрелял? Прием...

Какое-то время в эфире царила мертвая тишина. Все замерли в ожидании ответа, затем на связь вышел Сью Шоу. Его хриплый, вечно простуженный голос трудно было с кем-нибудь спутать:

- Это я. По-моему, мы забрели в гости к болотным крысам. Одна меня только что чуть не слопала. Вы там смотрите в оба. Не нравится мне все это.

- Слушайте мой приказ,- решительно сказал Кренон. - Держитесь ближе друг к другу. Оружие привести в боевую готовность.

Он посмотрел налево и увидел как к ним с Доном Кейлони идут Хенли и Курт Левин. Справа приблизились Люк Соло, Джим Дигриз и Эйра Кама. Кренон подал знак рукой, давая понять, что дистанция стала оптимальной, и они продолжили путь.

Вскоре им вновь пришлось преодолевать непроходимые заросли ползучих лиан. Дон обнажил светосаблю. Неожиданно справа зашелестели упругие стебли и прямо перед ними, обжигая горячим дыханием, появилась огромная серая морда болотной крысы. С черной, покрытой жесткой щетиной клыкастой пасти стекала желтая пена. Готовясь к прыжку, крыса попятилась и угрожающе приподнялась на задние когтистые лапы. Все произошло настолько быстро, что Дон даже не успел ударить хищника светосаблей. Кренон, шедший следом, отпрыгнул в сторону и два раза нажал на пусковую кнопку фотонного излучателя. После первого выстрела крыса с дымящимся брюхом по инерции еще двигалась к Дону, но когда в раскрытую пасть ударил второй луч, агрессор, заливая траву малиновой кровью, в предсмертных конвульсиях завалился на бок. Длинный, похожий на канат, облезлый хвост крысы несколько секунд судорожно метался по траве, пока не замер.

- Спасибо, друг, - Дон благодарно посмотрел на капитана.

У Кренона дрожали руки, но он старался не подавать виду:

- Идем...

Местное светило стояло почти в зените, однако тепла практически не ощущалось. Толиман - так называлась звезда, вокруг которой вращалась планета. Очень похожая на Солнце, эта звезда, однако, не могла согреть Зеро, из-за обилия космической пыли, плотным облаком прикрывавшей планету от света. Вероятно в далеком будущем из этого скопления пыли образуется спутник Зеро и тогда света станет несомненно больше и жизнь на планете начнет развиваться бурными темпами. А пока, скорее похожая на царство теней, планета оставалась холодным шариком, населенным опасными хищниками, выжившими в жестокой схватке с природой. С этими мыслями Кренон пересек небольшое болотце и вышел на обширное сухое пространство, покрытое низкой травой и мхом. То, что он увидел, никак не вписывалось в рамки его представлений об этой планете. Прямо по центру поляны, образуя правильную окружность, стояли несколько тщательно отполированных булыжников высотой с человеческий рост. В центре окружности располагался гладкий плоский камень красного цвета таких размеров, что на нем могли без особого труда уместиться пара взрослых болотных крыс. Пространство вокруг этого странного сооружения было усеяно костями. Картинка из древних фильмов про кровожадных туземцев. Кренон застыл в нерешительности. Он не заметил, как сзади тихо подошли Дон Кейлони и Курт Левин.

- Что скажешь, командир? - спросил Дон, ловким движением убирая светосаблю в ножны.

Кренон не ответил. Дон хотел еще что-то спросить, но его слова потонули в треске близких выстрелов. Стреляли сразу из нескольких стволов. Потом они слышали чьи-то выкрики и ругательства. Курт Левин, не говоря ни слова, выхватил из кобуры лазерный пистолет и помчался через всю поляну на звук выстрелов. Кренон уже успел понять, что

отряд столкнулся с целым полчищем болотных крыс. Когда все стихло на связь вышел Леон Троцкий:

- У нас все в порядке. Завалили парочку крысенышей. Никто не пострадал... Капитан, мы на берегу ручья, переходить?

- Подождите,- Кренон смотрел на Джима Дигриза, который стоял вдалеке, на противоположном конце поляны и подавал какие-то знаки. - Джим, вижу тебя. Что случилось?

- Там кто-то есть,- Дигриз показывал рукой на север, в сторону ручья. - На крыс вроде не похоже.

- Я тоже заметил какое-то движение. - Это был голос Сью Шоу, который замыкал цепь десантников. Элмер Кренон внимательно посмотрел вокруг. Странное сооружение из камней и, замеченное многими движение на другом берегу ручья, заставляли задуматься. В голове вихрем пронеслись кровавые сцены расправы загадочных аборигенов с мирной исследовательской экспедицией профессора Джонсона и с колонистами, ушедшими на ее поиски.

Удивительным было то, что ни в одном справочнике по этой планете не было ни слова о какой-либо, пусть даже примитивной, форме разумной жизни. Видимо, недаром Джонсон уделял такое пристальное внимание северной части материка. Перспектива оказаться на религиозном обряде аборигенов в качестве жертвы, приносимой в угоду местному божеству, Кренона конечно же не устраивала. Но и отступать ему совсем не хотелось. Он успокаивал себя тем, что вооруженным и хорошо обученным солдатам не составит особого труда разобраться с доисторическими туземцами, стоявшим на несомненно более низком уровне развития. Ну разве могут спорить с лазерным оружием каменные топоры?

Кренон хотел было уже отдать приказ собратиться всем на поляне как в воздухе рядом с ним со свистом пронеслась стрела и глубоко воткнулась в ствол одиноко стоящей фиолетовой пальмы в пяти шагах от него. Невидимый лучник прятался в небольшой роще ниже по ручью.

- Быстрее сюда!- Это кричал Дон Кейлони, который вовремя сориентировался и уже успел спрятаться за ближайшими кустами. Возможно, его окрик спас Кренону жизнь: вторая стрела просвистела, рассекая воздух, прямо над тем клочком земли, на котором только что стоял капитан. Эфир заполнили голоса и крики десантников. Стрелы летели с противоположного берега ручья сразу из нескольких мест. Похоже, что лучников было много, может быть десятков, а то и два.

- Командир, говорит Джеки Доннелли. У нас тут ранили Сью Шоу. Я осмотрел стрелу. Облита какой-то гадостью. Похоже на яд...

- Я в порядке,- скрипя зубами перебил напарника Сью. - Задета только кисть руки.

- Вы как хотите, а я открываю огонь,- решительно прозвучал в эфире женский голос. Это была Васкез - молодая горячая девушка, только недавно закончившая Школу Космических Десантников, но по уровню боевой подготовки не уступавшая многим сослуживцам - мужчинам.

- Васкез, успокойся, - сказал Кренон, внимательно разглядывая противоположный берег. - Аборигенов надо отвлечь. Леон Троцкий возглавит группу, которая попытается обойти их с тыла. С ним пойдут Дон, Курт, Люк Соло и Эйра Кама. Пройдете направо вверх по ручью, перейдете на тот берег и атакуете туземцев с тыла. Джим, ты будешь прикрывать группу Леона с этого берега. Выбери удобную позицию. Я, Зира Кама и Хенли пойдём следом за ними и, в случае чего, поддержим огнем. Васкез, Сью и Джеки Доннелли - оставайтесь пока в укрытии и смотрите в оба, чтобы нас самих не обошли. Задача ясна?

Вопросов не было. Кренон хлопнул по плечу притаившегося рядом Кейлони:

- С Богом!

Дон короткими перебежками от куста к кусту стал пробираться к Леону Троцкому. За спиной, не сказав ни слова, прошмыгнул и Курт Левин.

Кренон дождался пока к нему подойдут Зира Кама и Хенли, а затем быстро перебежал ближе к ручью, который мирно журчал по ту сторону поляны. За ним последовали остальные. Выбрав удачную позицию за полуразвалившимся валуном, утопавшим в

болотной жиже, Кренон осмотрел противоположный берег ручья, основательно заросший высокой травой и кустарником. Пальмовая роща, из которой вылетела первая стрела, находилась левее, ниже по ручью. Ничего подозрительного там не наблюдалось. Видимо, туземцы были хорошими воинами и удачно использовали естественную среду для маскировки.

- Леон, отзовись, - Кренон не отрывал глаз от пальмовой рощи.

- Все в порядке, - доложил Троцкий, - Только что видели свампера. Быстро плавает, ничего не скажешь. Правда, мы его все равно обогнали. Так что, когда пойдете через ручей, будьте осторожны - дедушка крокодил остался голодным.

- Учтем обязательно, - ответил Кренон.

Тишина уже действовала на нервы. Казалось, что не было никакой атаки туземцев, не свистели ядовитые стрелы. Но внутреннее чутье подсказывало, что враг рядом и пристально следит за каждым их движением. Кренон посмотрел налево и увидел, как из плотной стены болотного тростника выскочил Джекки Доннелли и, чуть пригнувшись, побежал к берегу ручья. В этот момент капитан десантников и заметил аборигена. Сначала как-то подозрительно зашевелился развесистый куст на противоположном берегу прямо напротив пересекавшего поляну Джекки, а затем, словно из-под земли, появилось ОНО. Кренон находился в удобной позиции, под углом к существу, и ему удалось хорошо рассмотреть его. Высокого роста, передвигавшийся на двух мускулистых ногах, покрытых серой чешуей, монстр скорее походил на динозавра из-за своей продолговатой, с хорошо развитой челюстью, головы. В любой другой ситуации Кренон никогда бы не поверил, что перед ним разумное существо, но массивный лук в мускулистых руках и кожаный колчан, туго набитый отравленными стрелами, говорили сами за себя.

Абориген собирался стрелять по бегущему десантнику и Кренону требовалось во что бы то ни стало опередить его. Он прицелился в голову и в тот момент, когда монстр уже натягивал тетиву, выстрелил. Неприятель дернулся, словно от удара током, однако на ногах устоял и попытался спрятаться за кустом, но второй луч насквозь прожег покрытый чешуей череп. Кренон был неприятно удивлен увидев, что монстр не свалился после первого выстрела, а даже попытался скрыться. Такой степенью защиты не обладали даже бронежилеты землян. Кренон помахал рукой в ответ на благодарственное приветствие Доннелли, который успешно добрался до зарослей тростника у самого берега.

- Командир, - вышел на связь Троцкий, - я видел, как ты его уложил, но похоже это только начало.

Над болотами пронеслось эхо выстрелов. Кренон на слух определил, что стреляли из фотонного излучателя и из лазерного пистолета. Видимо, группа Троцкого столкнулась с аборигенами.

Под прикрытием Джекки Доннелли Кренон, Хенли и Зира Кана в мгновение ока перебрались на крутой берег.

Впереди за деревьями они увидели двух аборигенов, отступавших под мощным натиском десантников. Кренон и его партнеры находились в очень выгодной позиции и им не составило труда отправить незадачливых лучников к праотцам.

Когда все стихло капитан вышел в эфир:

- Леон, не стреляйте. Мы идем к вам.

Из-за деревьев появились фигуры десантников. Кренон направился к ним. На Леона Троцкого он сразу обратил внимание. Тот был явно чем-то озабочен и, когда капитан подошел к нему, Троцкий молча показал в сторону голубой полосы небольшой реки, которая отделяла их от... Впрочем, Кренон не сразу понял, что он там увидел. В редких просветах пальмового леса, начинавшегося сразу за рекой, были видны какие-то постройки и сооружения.

- Похоже, это их логово, - сказал Троцкий - Что будем делать?

- Вообще-то это уже не по нашей линии, - заметил Зира Кама. - Нет никакого смысла туда соваться.

- Надо идти. - Кренон был непоколебим. - И мы должны выяснить все, чтобы те, кто придет после нас, могли здесь спокойно работать.

Он включив передатчик:

- Васкез, как у вас?

- Тут пришли Диана, Хикс и Хадсон. Они услышали стрельбу и поэтому поспешили к нам.

- Командир, - вмешался капрал Хикс, - Мы можем чем-нибудь помочь?

- Вызывайте из лагеря остальных. Мы нашли логово этих папуасов. Все идем туда. Ваша группа пойдет по левому флангу через устье реки. Направление точно на север. Заодно прикроете нас. Старший группы капрал Хикс. Все, кто еще не перешел ручей, присоединяются к вам. Понятно?

Капрал Хикс не ответил. Вместо этого по лесу прокатилось эхо выстрелов.

- Я думаю, они разберутся, - пробормотал Кренон. - Пошли.

Где-то совсем рядом просвистела стрела. Он услышал сдавленный крик и обернулся. На траве без движений лежал окровавленный десантник. Это был Люк Соло. Стрела угодила ему в висок.

- Убит, - выдохнул Зара Кама, склонившись над телом.

Едва он успел произнести это слово, как над его головой, сшибая ветки кустов, пролетели пули. Зира плюхнулся на землю и замер. Вся группа затаилась в ожидании. На время все стихло. Кренон прятался за толстым стволом старой пальмы. Это было что-то новенькое: туземцы, как оказалось имели огнестрельное оружие. "Вот тебе и каменные топоры," - подумал командир десантников, прислушиваясь к каждому шороху.

- Одного я вижу, - прошептал стоящий рядом Дон Кейлони. - Попробую...

Он не договорил. На их группу снова обрушился град пуль и стрел. Аборигены засели у самого берега реки и, видимо, никого не собирались пропускать на другую сторону. Но Дону удалось использовать секундную паузу: его лазерное ружье выстрелило в тот момент, когда один из аборигенов попытался проскочить между деревьями. Выстрел оказался точным. Массивное тело монстра покачнулось и, под хруст ломаемых веток, завалилось на землю. Вдохновленные удачей Кейлони, десантники открыли огонь по зарослям тростника, за которым прятались туземцы. Под хорошим прикрытием лазерных лучей Курт Левин добрался до трупа аборигена, убитого Доном и там подобрал смертоносное оружие внеземного происхождения.

Когда десантники прекратили обстрел вражеских позиций, Курт показал оружие инопланетян Кренону.

- Так это же терминатор! - воскликнул капитан, взяв в руки ружье.

Курт утвердительно кивнул. Всем опытным десанникам было известно это новое оружие, которое по своим характеристикам могло сравниться, пожалуй, даже с фотонным излучателем. Терминатор стрелял разрывными пулями, наполненными кислотой. В магазин помещалось до двухсот пятидесяти патронов, что являлось неоспоримым преимуществом перед многими другими видами оружия. Помимо этого ружье имело оптический прицел с встроенным прибором ночного видения и еще массу всякого рода достоинств. Впервые терминаторы обнаружили на одной давно вымершей планете в системе Сириуса. Там был найден целый арсенал всякого рода вооружений. Цивилизация, оставившая это наследство, перестала существовать задолго до появления первого человека на Земле и, вообще-то, не достигла больших высот в освоении Вселенной из-за глобальной термоядерной войны, практически уничтожившей планету. Кто дал оружию название терминатор никто уже не помнил, но оно было во всех учебниках по вооружениям.

- Боюсь, что мы их не за тех приняли, - вслух подумал Кренон, отдавая инопланетное оружие Курту Левину.

- Насколько мне известно, в системе Сириуса цивилизация была гуманоидной, а эти монстры ничего общего с ними не имеют, - Курт повесил терминатор на плечо.

- А это уж пусть ученые решают, как терминаторы попали к этим... Кренон не нашел подходящего названия для существ, с которыми они столкнулись.

В продолжение этого разговора они двигались к реке, за которой находилось логово монстров. Несколько раз им пришлось вступать в перестрелку с туземцами, вооруженными терминаторами и луками. С лучниками все было достаточно просто, а вот с обладателями

оружия из системы Сириуса приходилось повозиться.

Когда они добрались до огромного каменного строения, изумлению их не было предела. Неправильной формы, с многочисленными окнами, походившими скорее на бойницы старинных крепостей, строение действительно напоминало приземистую крепость, каких на Земле, впрочем, никогда не строили. То, что издалека им казалось многочисленными постройками, на самом деле оказалось одним единственным домом, имевшим несколько прилегавших друг к другу флигелей. Ухоженная площадь перед логовом монстров была пустынной. Интересным было еще и то, что у этого дома не было дверей, в привычном понимании. Вместо них в стенах чернели проходы, похожие скорее на норы диких животных.

Группа капитана Кренона приблизилась к строению с правой стороны и, под прикрытием каменных стен, направилась ко входу. Остановившись возле норы, ведущей внутрь, Кренон вышел на связь с группой капрала Хикса.

- Нас окружили, - без предисловий сообщил Хикс. - Мне удалось прорваться и я сейчас иду к вам.

- Диана, - позвал Кренон. - Отзовись!

- Погибла Диана, - прохрипел Сью Шоу. Свамперы убили. Она была ранена и не успела перейти ручей... Васкез, прикрой меня.

Прогремели залпы далеких выстрелов. Всем было ясно, что группе капрала Хикса пришлось туго. Но лишь благодаря им Кренон и его партнеры получили возможность добраться до логова монстров.

Капитан дождался пока затихнут выстрелы, а затем снова вышел в эфир:

- Все кто меня слышат - отзовитесь.

- Дэниел Дар на связи. Мы с Шарки вызвали спасательный корабль. Он уже вылетел, сейчас идем к вам по вашему следу.

- Хорошо.

- Капитан, это Джим. Мы с Васкез и с Хадсоном пытаемся вырваться из окружения, как слышно? Прием.

- Слышу вас, Сью. Ты где?.. Прием.

Сью Шоу на связь не вышел. Напрасно Кренон вызывал его. Видимо ему так и не пришлось перейти злосчастный ручей. Никто больше не видел Сью с того рокового момента, когда он, под прикрытием Васкез, попытался прорваться сквозь окружение.

- Говорит Джекки Доннелли. У меня все в порядке, я подхожу к реке.

- Мы идем в гости к монстрам, - сообщил для всех Кренон и, повернувшись к десантникам, которые стояли рядом, добавил: - Ну что ж, посмотрим, как они нас встретят.

В небе сгущались сумерки, когда они вошли в дом. Прямо у входа их встречали два монстра с трезубцами в сильных руках. Впрочем, для десантников не составило особого труда быстро с ними расправиться. За спинами монстров прятались детеныши этих загадочных существ. Когда Кренон вошел в просторное помещение с фотонным излучателем наперевес, детеныши бросились врассыпную.

- Курт и Хенли налево, остальные за мной, - скомандовал Кренон.

Каменный пол и стены небольшого зала, в котором они оказались, были расписаны причудливыми узорами. Справа от них, у стены прямо сквозь пол рос небольшой куст какого-то растения с широкими плотными листьями, а на противоположной стороне они увидели два огромных яйца, из которых по-видимому и вылуплялись аборигены. Каждое яйцо лежало отдельно в небольшом углублении с подстилкой из сухой травы.

Осмотр зала занял всего несколько секунд, но как только десантники приблизились к своеобразному инкубатору, произошло то, чего так опасался Кренон. В широком проходе в соседнее помещение появилась громадная тварь, только отдаленно напоминавшая монстров, с которыми они воевали. Скрюченное туловище существа, покрытое серым панцирем, венчала гигантская голова с чешуйчатой мордой. Из продолговатого рыбьего рта торчали клыки, а большие выпуклые глаза малинового цвета зло смотрели из под свисавших мешками век. Уродливые тонкие руки были сложены на покато груди. Тварь передвигалась на двух жирных коротких ногах, похожих на широкие тумбочки. Вне всяких сомнений это

была matka монстров. Десантники открыли огонь. Лазерные лучи шипя вгрызались в прочный панцирь, оставляя на сером теле черные выжженные пятна. Но, похоже, мамаше этих тварей, лучи лазерных ружей и пистолетов не доставляли особых неудобств. Под лавиной огня matka ухитрилась несколько раз выплюнуть кислотной слюной, едва не зацепив кого-то из десантников. Каменная стена, в которую попала тварь, в считанные секунды превратилась в пыль. Их спасало только то, что расстояние, на которое долетала кислота, было небольшим и, выдерживая безопасную дистанцию, они достаточно спокойно вели обстрел. Положение осложнялось, однако, тем, что детеныши монстров не остались в стороне во время этой схватки, а бросились на помощь своей мамаше. Они были без оружия, но их крепкие зубы без особого труда могли перегрызть горло любому человеку. От клыков одного детеныша пострадал Хенли. Он перезаряжал пистолет, когда к нему сзади подкрался детеныш и, запрыгнув на спину десаннику, вцепился клыками в шею. От неминуемой смерти Хенли спас Леон Троцкий, одним выстрелом уложивший монстра. Битва закончилась неожиданно. После очередного залпа matka, обстрелянная со всех сторон, издала предсмертный крик и опрокинулась навзничь, звонко ударившись панцирем о каменный пол. Разобраться с отдельными защитниками королевы племени было уже нетрудно. Группа Кренона прошла вглубь опустевшего логова и с радостью встретила там до зубов вооруженных инопланетным оружием Хикса, Васкез, Хадсона и Джима Дигриза.

- А где Джекки Доннелли? - поинтересовался Кренон у Васкез.

- На корабле.

- За нами уже прилетели? - он выглянул в окно и действительно увидел стальную громадину корабля, удачно посаженного прямо у выхода из жилища монстров.

Кренон связался с Дэниелом, который вместе с Шарки в эти минуты уже пересек поляну перед домом и сейчас направлялся к кораблю. Дэниел ответил после недолгого молчания:

- Нам надо поторапливаться. Мы только что видели целый отряд туземцев, который пришел с юга. Они видимо вернулись с охоты и сейчас направляются сюда.

Как в подтверждение этих слов, со стороны реки прогремели выстрелы.

- Хватит нам бессмысленных смертей, всем быстро на корабль! - выкрикнул Кренон, подталкивая десантников к выходу.

Они пересекли просторный коридор и оказались в помещении расстрелянной королевы аборигенов. Здесь они снова увидели полуметровые яйца, в которых развивались зародыши монстров.

- Наверное их нужно прихватить с собой, - высказал мнение Дон Кейлони.

- Конечно, пусть ученые поломают головы, - поддержал его Троцкий.

Кренон с ними согласился и приказал взять с собой несколько яиц. Повесив ружья за спину, Дон, Хадсон и Зира Кама подняли по одному яйцу и последовали за всеми к выходу. А у корабля уже шла перестрелка и всем стало ясно, что выйти без боя из дома им не удастся.

У трапа корабля лежал сраженный пулями Лазарь Шарки. Раненый Дэниел Дар с перекошенным от боли лицом прятался за углом дома.

- Мы прикроем вас, - сказал Кренон застывшим в нерешительности десанникам, которые несли тяжелые яйца, подобранные в комнате матки.

Это были самые трудные минуты в его жизни. Один за другим десантники перебежали опасную зону, разделявшую выход и трап корабля.

На его глазах, не добежав до трапа двух шагов, замертво рухнул на землю Зира Кама, настигнутый пулей. Яйцо, выпавшее из его рук, осталось лежать на траве, мешая проходу другим. В эти последние минуты был ранен и сам Кренон. Выбросив бесполезный фотонный излучатель с опустевшим магазином, Элмер Кренон рванул к трапу. Пули зацепили его в тот момент, когда капитан бежал вверх по ступенькам. Все уже были на борту, и как только Кренон, тяжело дыша ввалился в люк, корабль стал медленно отрываться от земли.

Долго еще он не мог забыть эту высадку на Зеро. Иногда по ночам ему снились кошмарные сны, в которых он опять вел свой отряд в логово монстров, а из зарослей болотного тростника погибшие Зира Кама, Сью, Люк Соло и Лазарь Шарки укоризненно смотрели ему в спину и молчали. После этой истории Кренон ушел из отряда и подался на

небольшой частный корабль, капитан которого был дальним родственником Кренона. И долгие годы он провел, занимаясь межпланетной торговлей. Многих из своего отряда он больше так и не увидел. Кто-то погиб на далеких планетах, а некоторые, как и он, оставили службу и поселились в разных концах Галактики.

Впервые за последние годы Кренон выбрался, чтобы навестить своего друга Дона Кейлони, который к этому времени уже возглавлял Центр Космической Безопасности Солнечной Системы. Впервые он так отчетливо вспомнил события на Зеро и рассказал о них внуку. Но он не смог удовлетворить любопытства юноши, поскольку не знал всех подробностей расследования гибели профессора Джонсона.

- Дед, ну откуда тогда у этих монстров взялось инопланетное оружие?

- Одно время ходили слухи, что эти существа все-таки были дальними родственниками гуманоидов с Сириуса, но по каким-то причинам они пошли по другому пути развития. Возможно, главную роль здесь сыграла термоядерная война, разразившаяся на их родной планете.

- А как они тогда оказались на Зеро?

Кренон встал из кресла и, застегивая костюм, устало посмотрел на внука:

- Ты знаешь, спроси это лучше у дяди Дона, когда мы прилетим.

Он посмотрел на экран. Их корабль уже медленно входил в шлюзовые ворота стыковочного отсека станции "Венера - Центр"...



## ПОСЛЕДНИЕ ИЗВЕСТИЯ

В эти октябрьские дни "ИНФОРКОМУ" исполнилось четыре года. В связи с этим мы предприняли ряд важных шагов, для которых, как нам кажется, настало время.

\* \* \*

На днях принято решение (и соответствующие меры), чтобы "ZX-РЕВЮ" стал в будущем году полноценным изданием с полиграфической точки зрения. У него будет стандартный книжный формат, высококачественная печать, красочная обложка, большее количество страниц и значительно увеличится тираж. В отличие от прошлых лет мы не будем искусственно сдерживать темп его распространения. Стил и содержание останутся теми же. В соответствии с пожеланиями читателей больше внимания будет уделяться вопросам музыки, графики и работы в дисковых системах. К концу года планируется начать освещение работы "Спектрума" в сетях.

Благодаря качественному полиграфическому исполнению каждого номера мы откажемся от традиционной практики выпуска в конце года годового сборника. И так, годовой сборник "ZX-РЕВЮ-93" станет последним.

\* \* \*

Объявленное в прошлом выпуске акционерное общество "ИНФОРКОМ-ПРЕСС" прошло государственную регистрацию и уже работает. Заканчивается подготовка первого номера электронного журнала "РС-РЕВЮ", посвященного чисто игровой тематике для РС-совместимых компьютеров. Объем, содержание и оформление этого журнала пока являются беспрецедентными для нашей страны.

Редакция нового журнала будет представлена читателям в первом выпуске, который пойдет в распространение 01.01.94г.

Журнал будет распространяться по каналам собственной сети региональных дистрибуторов фирмы без защиты от копирования. Подробности см. в "ZX- РЕВЮ" N 7-8 за этот год.

Подготовлены условия для продажи ограниченного количества экземпляров этого издания в подарочном исполнении (в красочном альбоме) через специализированные компьютерные салоны г. Москвы по престижно высоким ценам (порядка 7-12 \$ за номер). Эта мера принята для поддержки региональных дистрибуторов. Наличие в Москве престижно высокого ценового уровня будет способствовать успеху их деятельности в регионах.

\* \* \*

Вышел из печати и поступил в рассылку владельцам именных сертификатов второй том "Графики Спектрума".

В ближайшие дни выходит из печати книга "30 часов БЕЙСИКА".

\* \* \*

Заключено соглашение между фирмами "СЛОТ" и "ИНФОРКОМ" об эксклюзивном освещении в "ZX-РЕВЮ-94" дисковой операционной системы для "Спектрума" IskraDOS.

Разработан план совместных мероприятий по продвижению этой системы на рынки программных средств СНГ. Эти шаги согласованы с владельцем авторских прав на систему, компанией IskraSOFT, г. С.-Петербург и получили поддержку. (Фирма "СЛОТ", г.Москва, является Генеральным дистрибутором компании IskraSOFT).

Подготовлена и передана в печать книга, посвященная работе с этой системой.

Систему IS-DOS, пакеты прикладных программ, разработанные под нее и сопутствующую литературу можно будет приобретать как у фирм IskraSOFT и "СЛОТ", так и по каналам "ИНФОРКОМа".



"ИНФОРКОМ" 121019, Москва, Г-19, а/я 16

### Вниманию читателей!

Мы продолжаем публиковать адреса пунктов, в которых можно приобрести наши материалы.

г. МОСКВА, ул. Новый Арбат, д. 2. 19-е отделение связи, 1-ый этаж операционного зала.

г. МОСКВА, радиорынок в Митино, проезд до ст. метро Тушинская или поездом до пл. Трикотажная (Рижское направл.). Суббота, воскр. 9 - 14. Киоск N J-37.

г. МОСКВА, радиорынок в Царицыно, проезд до ст. метро Царицыно. Суббота, воскр. 9 - 14.

г. БЕЛГОРОД, магазин "РАДИОТОВАРЫ", ул. Ленина, 32.

г. БЕЛГОРОД, Октябрьская 84, кв. 103, "СТУДИЯ КОМПЬЮТЕР".

г. ВЛАДИВОСТОК, Океанский проспект, 140, магазин "ПАНОРАМА". Проезд трамваем до ост. Некрасовская.

г. ВОРОНЕЖ, студия компьютерных игр SAN-SAN, магазин-салон "ЭЛЕКТРОНИКА", тел. 14-00-73.

г. ДНЕПРОПЕТРОВСК, ул. Шевченко, 34. фирма "ЭКОС".

г. ЕКАТЕРИНБУРГ, магазин "СПЕКТРУМ", Главный проспект, 99.

г. ИЖЕВСК, радиорынок "Козий парк".

г. КЕМЕРОВО, магазин "ТЕХНИЧЕСКАЯ КНИГА", ул. Весенняя, 24.

г. КЕМЕРОВО, магазин "ОРБИТА", пр. Ленина, 133.

г. КИРОВ, "Дом науки и техн.", магазин-салон "МАРС", ул. Производственная, Д.27.

г. КРАСНОЯРСК, радиорынок, проезд до ост. "Затон", субб, вскр.

г. НАБЕРЕЖНЫЕ ЧЕЛНЫ, Татарстан, Новый Город, салон-магазин "ПРИНТЕР".

г. НИЖ. НОВГОРОД, ИМА "Ф-ПЛЮС". Магазин "ФОТОЛЮБИТЕЛЬ", ул. Горького, 146.

г. ОРЕНБУРГ, магазин "ВОЕННАЯ КНИГА", ул. Советская.

г. ПРОКОПЬЕВСК, Кемеровской об. Тырган, магазин "ОДЕЖДА", фирма "РОНЭТ".

г. РЫБИНСК, ул. Гоголя, 5. ТТЦ "ГНОМ".

г. ХАБАРОВСК, ул. Запарина 65, магазин "ФИЛАТЕЛИЯ".

г. ЧЕБОКСАРЫ, маг. "ЭКСПРЕСС". НПК фирма "НОВА", ул. Привокзальная, Д.6.

г. ЯРОСЛАВЛЬ, магазин "РАДИОСПОРТ-ТУРИЗМ", Ленинградский пр-т.

## СПЕКТРУМ В ШКОЛЕ

В последнее время наши статьи из рубрики "Спектрум в школе" предназначались в основном для старшеклассников. Но ведь если в Вашей школе есть кабинет информатики, то, наверное, и малышам было бы интересно в него заглянуть.

Предельно простая программа, которую мы приводим сегодня, послужит для развлечения и приобщения к компьютеру тех, кто только-только учится считать до десяти.

Суть программы состоит в том, что на экране на некоторое время изображаются несколько фигурок и малыш должен сосчитать, сколько их и нажать соответствующую клавишу на компьютере. В качестве фигурок мы приняли любимый детьми "черепашек-ниндзя", изобразив их с помощью четырех символов UDG-графики. Несмотря на свою очевидную простоту, эта программа может иметь вполне реальный успех, если ей сделать приличный художественный дизайн, ввести в нее элементы мультипликации и красиво озвучить. Так что держайте, все в Ваших руках.

Конструкция одного "героя" сделана из четырех символов UDG и имеет размер 16X16 пикселей. Символы располагаются следующим образом:

[A] [B]

[C] [D]

Для того, чтобы отличить символы UDG в программе от обычных символов, мы взяли их в квадратные скобки. Вам, конечно, никаких скобок набирать при наборе программы не

надо (надо только перейти в графический режим - CAPS SHIFT + 9 - курсор "G").

```
10 REM **СОСЧИТАЙКА**
20 LET wait = 310
25 LET povtor=140
30 REM Ввод символов графики пользователя.
40 FOR n=0 TO 31
50 READ a
60 POKE USR "a"+n,a
70 NEXT n
80 DATA 1,1,3,15
81 DATA 31,59,63,15
82 DATA 128,129,134,248
83 DATA 252,198,194,124
84 DATA 55,70,134,12
85 DATA 12,12,24,16
86 DATA 224,112,48,48
87 DATA 96,96,112,0
90 REM Запустите набранные строки командой RUN, после чего можете продолжать набор
    программы.
100 LET a$="[AB]"
110 LET b$="[CD]"
120 DIM x(10): DIM y(10)
130 BORDER 1: PAPER 7: INK 2: BRIGHT 1
140 REM povtor
150 CLS
160 LET count=1+INT(9*RND)
170 REM Расположение черепашек
180 LET x(1)=2*INT(16*RND)
190 LET y(1)=2*INT(10*RND)
200 FOR n=2 TO count
210 LET x(n)=2*INT(16*RND)
220 LET y(n)=2*INT(10*RND)
230 FOR k=1 TO n-1
240 IF x(n)=x(k) AND y(n)=y(k) THEN LET n=n-1: REM Если у нас случайно совпали координаты
    позиции печати двух черепашек, то играющий на экране увидит только одну и потому этот
    вариант надо исключать.
250 NEXT k: NEXT n
260 REM Печать "черепашек"
270 FOR n=1 TO count
280 PRINT AT y(n),x(n); a$
282 LET y(n) = y(n)+1
285 PRINT AT y(n),x(n); b$
290 NEXT n
300 PRINT INK 9; AT 20,10;"?????????????"
310 REM wait
320 IF INKEY$="" THEN GO TO wait: REM здесь между кавычками не должно быть пробела
330 IF INKEY$ <> STR$ count THEN BEEP .5,0: BEEP 1,-24: GO TO wait
340 REM Правильный ответ
350 FLASH 1: CLS
360 FOR n=1 TO 10
370 BEEP .33,12
380 BEEP .33,24
390 NEXT n
400 FLASH 0
410 GO TO povtor
```

# SINCLAIR LOGO

(Окончание)

Начало см. на с. 69-74, 90-96, 134 - 138, 180 - 186.

## ГЛАВА 9. ВВОД И ВЫВОД ИНФОРМАЦИИ

До сих пор мы рассматривали только один метод ввода информации в компьютер - READLIST и (если не учитывать черепашую графику) только один метод вывода информации - PRINT. В этой главе мы рассмотрим некоторые методы, которые дадут программисту новые возможности для ввода/вывода, хотя и ценой более сложного программирования.

READCHAR ждет, когда пользователь нажмет какую-либо клавишу, принимает один символ с клавиатуры и выдает его в качестве выходного результата. После нажатия клавиши не надо нажимать ENTER, принятый символ не изображается на экране, ввод символа не сопровождается изображением "приглашения". Попробуйте:

```
MAKE "CHARACTER READCHAR
```

Нажмите ENTER - система ожидает нажатия любой клавиши. Нажмите любую клавишу. На экране появится приглашение "?" - это означает, что ЛОГО-система вернулась в командный режим.

Вы можете проверить, что за символ был введен:

```
PRINT :CHARACTER
```

Это обычный прием, когда надо узнать, что за клавиша была нажата. Вот, например, программа, с помощью которой Вы можете управлять движением черепашки по экрану.

```
TO DRAW
  MAKE "CHARACTER READCHAR
  IF :CHARACTER = "F [FORWARD 5]
  IF :CHARACTER = "B [BACK 5]
  IF :CHARACTER = "R [RIGHT 10]
  IF :CHARACTER = "L [LEFT 10]
  IF :CHARACTER = "S [STOP]
DRAW
END
```

Функция READCHAR бывает очень полезной, когда вместо символов Вы работаете с их кодами. Коды символов с 0 по 127 называют кодами ASCII (American Standard Code for Information Interchange). Коды с 0 по 31 используются компьютером в качестве управляющих при операциях ввода/вывода (например для перемещения курсора) и не являются печатными, т.е. их нельзя изобразить на экране, хотя они и влияют на то, как изображаются другие (печатные) коды. Коды с 32-го по 127 - печатные символы (буквы, цифры, знаки препинания). Дополнительно к стандарту ASCII "Спектрум" имеет еще ряд кодов для символов.

Коды 123 - 143 представляют символы блочной графики, а символы 144 - 164 - это символы графики пользователя.

Вы можете узнать, какому символу какой код соответствует с помощью следующей несложной программы.

```
TO SHOWASCII
  MAKE "CHARACTER READCHAR
  PRINT ASCII :CHARACTER
SHOWASCII
END
```

Создав такую процедуру, Вы можете дать команду SHOWASCII и, нажав любую клавишу, увидеть на экране код этого символа.

Противоположное действие имеет команда CHAR. Она переводит код от 32 до 143 в соответствующий графический символ.

Так, PRINT CHAR 65 даст:

A

Можно использовать команду CHAR совместно с командой ASCII. Нижеприведенная

процедура будет шифровать ту информацию, которую Вы введете с клавиатуры.

```
TO CONFUSE
  MAKE "CODE ASCII READCHAR
  IF :CODE = 13 [STOP]
  TYPE CHAR (:CODE + 1)
CONFUSE
END
```

Процедура прекратит свою работу, когда будет нажата клавиша ENTER (ее код = 13).

Если Вам не надо ожидать нажатия клавиши, то Вы можете использовать функцию KEYP. Она проверяет факт нажатия какой-либо клавиши и выдает TRUE или FALSE в зависимости от того, имело или нет место такое событие.

Вот еще одна программа для черепашки, отличающаяся тем, что между нажатиями клавиш черепашка продолжает свое движение.

```
TO DRAW2
  IF KEYP [DOIT]
  FORWARD 5 DRAW2
END
TO DOIT
  MAKE "CHARACTER READCHAR
  IF :CHARACTER ="M [RIGHT 90]
  IF :CHARACTER ="Z [LEFT 90]
END
```

Клавишами M и Z Вы теперь можете управлять движением черепашки по экрану.

Символы от 144 до 164 включительно могут задаваться непосредственно пользователем. О том, как это сделать, рассказано в инструкции к компьютеру (и в книгах ИНФОРКОМа). Но для того, чтобы задать эти символы, в определенные адреса памяти компьютера надо внести свои значения. В языке ЛОГО для этой цели служит команда .DEPOSIT. Но будьте осторожны при работе с этой командой. Она напрямую меняет содержимое ячеек памяти компьютера. ЛОГО-система не в состоянии проверить правильность Ваших действий и, если Вы станете использовать эту команду без четкого понимания того, что и зачем Вы делаете, есть большая вероятность того, что Вы можете разрушить созданные Вами и хранящиеся в памяти компьютера процедуры.

Пара нижеприведенных процедур демонстрирует задание нового символа графики пользователя путем помещения в память компьютера (начиная с адреса 64215) списка из восьми чисел, которые и определяют конструкцию нового символа.

```
TO DEFCHAR :CH :ALIST
  IF OR NOT NUMBERP :CH NOT LISTP :ALIST [STOP]
  IF OR :CH<144 :CH>164 [STOP]
  DODEF :ALIST 0
END
TO DODEF :ALIST :COUNT
  IF EMPTY P :ALIST [STOP]
  .DEPOSIT 64216 + :CH*8 + :COUNT FIRST :ALIST
  DODEF BUTFIRST :ALIST :COUNT +1
END
```

Для того, чтобы определить, какими должны быть эти восемь чисел, задающие конструкцию символа, Вам придется почитать соответствующую литературу или поэкспериментировать. Так, например, следующая группа чисел задаст символ 144 в виде стрелки, направленной вниз:

```
DEFCHAR 144 [16 16 16 16 146 84 56 16]
```

Для управления выводом текста на экран компьютера тоже существует ряд полезных процедур и мы последовательно их рассмотрим.

SETCURSOR - принимает в качестве входных параметров список из двух чисел и помещает курсор в соответствующую позицию экрана. С этой позиции и начнется в последующем печать выводимого текста. Первое число списка указывает на номер экранного столбца (от 0 до 31), а второе число - на номер экранной строки (от 0 до 21). Левому верхнему углу соответствуют координаты (0,0).

SETTC (SET Text Color) тоже принимает на входе два числа. Первое число задает цвет

фона (аналог оператора БЕЙСИКА PAPER), а второе число задает цвет символов (аналог INK). Обычная ситуация для режима печати черным по белому - [7,0]. Вы можете поэкспериментировать со следующей процедурой:

```
TO TESTCOL
  TEXTSCREEN
  SETCUR [12 10]
  SETTC [1 6]
  PRINT [HELLO THERE]
  MAKE "G READCHAR
  SETTC [7 0]
END
```

В последних строках организовано ожидание нажатия клавиши для восстановления нормальных цветов. Параметр яркости фона, на котором печатаются символы, может задаваться командой BRIGHT (как и в БЕЙСИКе). Команда BRIGHT должна иметь при себе один параметр, который может быть равен 0 или 1. BRIGHT 1 - повышенная яркость, BRIGHT 0 - обычный режим.

Команда INVERSE включает режим печати символов в инвертированном виде, т.е. цвет символов становится цветом фона и наоборот.

Команда FLASH включает режим мигания символов, когда попеременно цвета символов и фона меняются местами.

Команда NORMAL отбивает эффект команд BRIGHT, INVERSE, FLASH и восстанавливает исходный режим печати текста.

Для тех, кто хочет посмотреть эти команды в работе, предлагается следующий пример:

```
TO FANCYNAME
  PRINT [WHAT IS YOUR NAME?]
  MAKE "NAME READLIST
  TEXTSCREEN
  BRIGHT 1
  SHOWNAME
  SETTC [7 0]
  SETCUR [0 21]
END
```

```
TO SHOWNAME
  IF KEYP [STOP]
  MAKE "X RANDOM 31
  MAKE "Y RANDOM 21
  MAKE "BG RANDOM 8
  MAKE "FG RNDCOL
  SETTC SENTENCE :BG :FG
  SETCUR SENTENCE :X :Y
  TYPE :NAME
  SHOWNAME
END
```

```
TO RNDCOL
  MAKE "FG RANDOM 8
  IF :FG=:BG [OUTPUT RNDCOL] [OUTPUT :FG]
END
```

Здесь процедура RNDCOL не только генерирует случайное число для установки цвета печатаемого сообщения, но еще и проверяет, не совпадает ли выбранный цвет символов с ранее установленным цветом фона и, если это так, то вызов RNDCOL повторяется до тех пор, пока эти цвета не станут разными.

А вот еще один пример для демонстрации приведенной выше техники. Группа процедур обрабатывает результаты матчей небольшой футбольной лиги. Главная процедура организует списки и печатает в цвете заголовки будущей таблицы, затем вызывается процедура HEADCOL, которая печатает заголовки столбцов, а потом процедура DOTEAM, обрабатывающая результаты для каждой команды.

```
TO LEAGUE
```

```

MAKE "TEAMS [SPARTAK DINAMO AVANGARD PROGRESS]
MAKE "COLS [P W D L PTS]
TEXTSCREEN
SETCURSOR [9 0]
SETTC [5 0]
TYPE [LEAGUE TABLE]
SETCURSOR [0 2]
SETTC [7 3]
TYPE "TEAM
HEADCOL 1
SETTC [7 0]
DOTEAM 1
PRINT "
END

```

Процедура HEADCOL выбирает данные из списка заголовков колонок и печатает эти заголовки в нужной позиции экрана с помощью команды SETCURSOR. Работа продолжается, пока все пять заголовков не будут распечатаны.

```

TO HEADCOL :COL
IF :COL>5 STOP
SETCURSOR SENTENCE (3*:COL + 11] 2
TYPE ITEM :COL :COLS
HEADCOL :COL + 1
END

```

Процедура DOTEAM определяет номер строки, в которой печатаются результаты для данной команды (он зависит от номера команды в списке команд). Она принимает из списка название команды, а затем принимает от пользователя с клавиатуры количество сыгранных игр (P), количество побед (W) и ничьих (D). При этом она использует еще одну процедуру - GETNUM.

Первыми двумя входными параметрами GETNUM являются координаты позиции экрана, в которой должна состояться печать. Третий входной параметр поначалу - пустое слово, оно будет содержать число, введенное пользователем.

Количество проигрышей команды (L) и набранные ею очки (PTS) вводить не надо, они могут быть рассчитаны самой программой. Предполагается, что победа оценивается двумя очками, а ничья - одним очком.

```

TO DOTEAM :TEAMNO
IF :TEAMNO COUNT :TEAMS [STOP]
MAKE "LINE 2*:TEAMNO + 2
SETCURSOR SENTENCE 0 :LINE
TYPE ITEM :TEAMNO :TEAMS
MAKE "PLAYED GETNUM 14 :LINE "
MAKE "WON GETNUM 17 :LINE "
MAKE "DRAWN GETNUM 20 :LINE "
MAKE "LOST :PLAYED - :WON - :DRAWN
SETCURSOR SENTENCE 23 :LINE
TYPE :LOST
MAKE "POINTS 2*:WON + :DRAWN
SETCURSOR SENTENCE 25 :LINE
TYPE :POINTS
DOTEAM :TEAMNO + 1
END

```

Прежде, чем рассмотреть процедуру GETNUM, мы рассмотрим еще одну связанную с ней процедуру GETDIG. Она принимает цифру с клавиатуры, а возвращает и эту цифру и ее код ASCII. Она передает также коды ENTER (13) и DELETE (12), если соответствующие клавиши нажаты. Если же нажата любая другая клавиша, процедура ее игнорирует и вызов повторяется. Цифрам от нуля до 9 соответствуют коды ASCII от 48 до 57.

```

TO GETDIG
MAKE "DIG READCHAR
MAKE "CODE ASCII :DIG
IF OR :CODE = 13 :CODE = 12 [STOP]
IF OR :CODE<48 :CODE>57 [GETDIG]
END

```

Теперь рассмотрим процедуру GETNUM. Начинается она довольно странно. Сначала устанавливается курсор в позицию, соответствующую ее первым двум входным параметрам, после чего печатается ее третий параметр, хоть он и пустой при первом вызове. Это нужно потому, что процедура работает рекуррентно и в процессе ее работы третий параметр будет изменяться.

В качестве курсора для печати используем мигающий вопросительный знак. Это легко реализуется следующими строками:

```
FLASH  
TYPE *?  
NORMAL
```

Цифра, введенная пользователем, поставляется из процедуры GETDIG. Оттуда же могут поступить и коды ENTER или DELETE. При появлении этих кодов следует убрать вопросительный знак из текущей позиции, в противном случае он продолжает мигать. Удаление вопросительного знака выполняется печатью на его месте пробела (символ 32). Могут быть проблемы с точным определением местоположения вопросительного знака. Самый лучший способ - пойти к началу вводимого числа (мы знаем где это начало) и снова напечатать число, а за ним уже пробел - на месте вопросительного знака.

```
SETCURSOR SENTENCE :X :Y  
TYPE NO  
TYPE CHAR 32
```

Процедура GETDIG может выдать три возможных результата: была нажата цифра с кодом от 48 до 57, либо клавиша ENTER (код 13) либо DELETE (код 12).

Если это цифра, то ее просто надо напечатать, добавив к ранее напечатанным цифрам вводимого числа. В нижеследующем примере предполагается, что числа имеют не более двух знаков и при попытке ввести третью цифру она игнорируется.

```
IF AND :CODE>13 (COUNT :NO)<2 MAKE "NO WORD :NO :DIG
```

Если нажата клавиша ENTER и если число не пустое, то мы можем напечатать это число.

Если нажата клавиша DELETE, то мы должны удалить последний символ числа, используя BUTLAST. Но и в этом случае сначала надо удостовериться, что число не является пустым словом. И если Вы рассмотрите нижеприведенную процедуру, то увидите в ней несколько необычную строку:

```
IF EMPTY :NO [MAKE "NO"]
```

Кажется, что в ней нет необходимости, но это не так. Дело в том, что когда мы применяем BUTLAST к слову, в котором есть только один символ, то рассчитываем получить пустое слово, но вместо него получаем пустой список, с которым команда WORD, помещающая очередную цифру в число не сможет работать. Эта странная операция конвертирует пустой список в пустое слово,

```
TO GETNUM :X :Y :NO  
  SETCURSOR SENTENCE :X :Y  
  TYPE :NO  
  FLASH  
  TYPE ""?  
  NORMAL  
  GETDIG  
  SETCURSOR SENTENCE :X :Y  
  TYPE :NO  
  TYPE CHAR 32  
  IF AND :CODE = 13 (NOT EMPTY :NO) [OUTPUT :NO]  
  IF AND :CODE = 12 (NOT EMPTY :NO) [MAKE "NO BUTLAST :NO]  
  IF EMPTY :NO [MAKE "NO"] IF AND :CODE>13 (COUNT :NO)<2 MAKE "NO WORD :NO :DIG  
  OUTPUT GETNUM :X :Y :NO  
END
```

Программу можно улучшать и дальше. Можно предусмотреть, чтобы пользователь не смог ввести число побед больше, чем было сыграно игр, в результате чего число поражений станет отрицательным. Совершенствованию нет предела, но сама техника Вам теперь должна быть понятна.



## ЗВУК.

"Спектрум" может выдавать звуковые сигналы. Это делается с помощью команды SOUND, после которой идет список из двух чисел. Первое число выражает длительность звучания ноты в секундах и должно быть в диапазоне от 0 до 10.5. Второе число - высота ноты, измеренная в полутонах относительно ноты "до" первой октавы. Это число должно быть в диапазоне от -60 до +69. Следующая процедура превратит клавиатуру Вашего компьютера в электроорган, но играть на нем весьма сложно.

```
TO KEYBOARD
  MAKE "KEY (ASCII READCHAR)-65
  IF AND :KEY>-59: KEY<70
    [SOUND SENTENCE 0.5 :KEY]
  KEYBOARD
END
```

Сыграть список нот равной длительности Вы можете с помощью процедуры:

```
TO PLAY :ALIST
  IF EMPTY :ALIST [STOP]
  SOUND SENTENCE 1 FIRST :ALIST
  PLAY BUTFRST :ALIST
END
```

Полутон - это звуковой интервал между двумя соседними (белыми или черными) клавишами на фортепиано. Если Вы не знаете, как складывается музыкальный ряд из полутонов, то попробуйте сыграть следующую гамму "до мажор".

```
PLAY [0 2 4 5 7 9 11 12]
```

К сожалению, способ задания звуков с помощью команды SOUND очень далек от общепринятой нотной нотации. Впрочем, можно написать процедуру, которая сделает за нас необходимое преобразование. Для этого служит следующая приведенная нами процедура. Первый параметр, который она ожидает, задает темп музыкальной мелодии. Это продолжительность звучания (в секундах) ноты, длительность которой принята нами за единицу. Затем процедура принимает список нот и размещает их в предварительно созданном списке SLIST, после чего исполняется мелодия.

```
TO MUSIC
  MAKE "NOTELIST [DO DO# RE RE# MI FA FA# SO SO# LA LA# SI DO']
  MAKE "SLIST[]
  PRINT [TEMP?]
  PRINT [ВВЕДИТЕ НОТЫ ПО ОДНОЙ, СНАЧАЛА ДЛИТЕЛЬНОСТЬ]
  GETMUS
  PLAY :SLIST
END
```

Процедура GETMUS принимает с клавиатуры список из двух чисел.

Первое число - длительность ноты. Предполагается, что самая короткая из них имеет длину 1, а длительность прочих - 1, 2, 3,... и т.д. Реальная же длительность (в секундах) будет зависеть от введенного Вами параметра TEMP.

Высоту ноты будет определять ее название - Do, Re, Mi и т.д.

Процедуре GETMUS нужна еще одна вспомогательная процедура FIND, которая найдет объект в списке. Первые два входных параметра для процедуры FIND - это имена объекта и списка, а третий параметр - стартовый номер, с которого начинается отсчет в списке. Если мы хотим, чтобы первой нотой была "ДО" первой октавы, то положим этот параметр равным нулю.

```
TO GETMUS
  MAKE "NIOTE READLIST
  IF EMPTY :NOTE [STOP]
  MAKE "DUR :TEMP*FIRST :NOTE
  MAKE "PITCH FIND LAST :NOTE :NOTELIST 0
  MAKE "SLIST LPUT LIST :DUR :PITCH :SLIST
  GETMUS
END
```

```
TO FIND :OBJECT :ALIST :N
  IF EMPTY :ALIST [OUTPUT 0]
```

```

IF :OBJECT = FIRST :ALIST
  [OUTPUT :N]
  OUTPUT FIND :OBJECT BUTFIRST :ALIST :N + 1
END

```

И, наконец, само исполнение музыки реализует процедура PLAY. Она отличается от ранее приведенной процедуры с тем же именем PLAY только тем, что полагает каждый элемент своего входного списка тоже списком, состоящим из двух элементов, которые затем передаются команде SOUND).

```

TO PLAY :ALIST
  IF EMPTY? :ALIST [STOP]
  SOUND FIRST :ALIST
  PLAY BUTFIRST :ALIST
END

```

Введя все процедуры, попробуйте программу в деле:

```

TEMP?
?0.3
  ВВЕДИТЕ НОТЫ ПО ОДНОЙ, СНАЧАЛА ДЛИТЕЛЬНОСТЬ
?2S0
?4D0'
?2SI
?2LA
?4S0
?2LA
?1SI
?1D0'
?2MI
?2MI
?2FA
?2RE
?6D0

```

Когда закончите, нажмите ENTER и после короткой паузы музыка заиграет. Один раз введя мелодию, Вы сможете воспроизвести ее всякий раз, как захотите с помощью команды:

```
PLAY :SLIST
```

## ГЛАВА 10. ЕСЛИ ЧТО-ТО НЕ ПОЛУЧАЕТСЯ

Мы начнем с того, что скажем несколько успокоительных слов тем, кто только что понял, что написанная им процедура не работает. Главное, что вам нужно знать о программировании и о компьютерах - это то, что неработоспособность - это естественное состояние любой компьютерной программы.

Неопытного программиста легко отличить от опытного. Новичок неприятно удивляется, когда только что написанная им программа не желает работать с первого раза. Опытные программист всегда удивляется, когда она с первого раза работает как надо и начинает искать в ней скрытые подвохи.

Можно сказать, что при программировании на языке ЛОГО Вас ждут два разных типа ошибок. Если Вы сделали ошибку при написания команд и ЛОГО не понимает, что Вы хотите сделать, Вы получите сообщение об ошибке. В других случаях ЛОГО понимает, что Вы хотели сделать и выполняет Ваши команды, но результат Вы получаете совсем не тот, который Вам необходим.

Методика отыскания и исправления ошибок и в том и в другом случае очень похожа. Хотя первый тип ошибок проще поддается исправлению, поскольку ЛОГО-система сама уже сказала Вам в своем сообщении что ей не нравится и в какой процедуре она наткнулась на препятствие. В этом смысле ЛОГО удобнее многих других языков программирования, поскольку его сообщения как правило более осмысленны и содержательны, чем в других языках. Тем не менее, все-таки эти сообщения были написаны создателями языка задолго до того, как Вы начали писать свою программу. Им как бы пришлось "предугадать" Ваши возможные ошибки. Это не всегда можно сделать точно. Потому не удивляйтесь, если эти

сообщения будут не абсолютно точны в определении Ваших ошибок.

Делу может помочь краткий обзор некоторых наиболее часто встречающихся ошибок и сообщений. Например:

1. \*You don't say what to do with ABCD\*

(Вы не указали, что делать с ABCD)

Такое сообщение легко объяснить. Помните, что всякая операция, выдающая какой-то результат, слева должна иметь имя команды, которая примет этот результат. В нашем примере операция произвела слово ABCD и либо отсутствует команда, которая примет этот результат в качестве входного параметра, либо она есть, но это не тот входной параметр, который для нее приемлем.

Рассмотрим пример. Предположим, что некоторая процедура SAYABCD выдает в качестве результата ABCD и Вы запишете такую строку:

```
PRINT "RESULT SAYABCD
```

В результате Вы получите приведенное выше сообщение об ошибке. Дело в том, что команда PRINT ожидает только один параметр в качестве входного и он здесь есть - "RESULT. Никакого указания, что делать с ABCD, здесь нет. Чтобы исправить ошибку, очевидно надо использовать команду SENTENCE, дабы команда PRINT могла распечатывать несколько данных.

2. \*Not enough inputs to ABCD\*

(недостает входных параметров для ABCD)

Если Вы рассмотрите определение процедуры ABCD, то должны обнаружить, что она ожидает большое число входных параметров, чем Вы ей дали. Может быть, это будет выглядеть и неочевидным, но метод анализа строк ЛОГО, который мы привели в Главе 3 Вам должен помочь.

3. \*XYZ doesn't output to ABCD\*

(XYZ не выдает данные для ABCD)

Эта ошибка возникает тогда, когда операция ABCD применяется к некоторой команде XYZ (не являющейся операцией), которая не выдает выходного результата, необходимого для успешной работы ABCD.

4. \*Too many inside parenthesis\*

(слишком много скобок).

Эта ошибка происходит, когда Ваш список имеет слишком много внутренних круглых скобок. В большинстве случаев присутствие такой ошибки может так напугать программиста, что он вообще будет стараться обходиться без таких скобок, хотя это тоже излишняя крайность.

5. "XYZ doesn't like ABCDE as input\*

(XYZ не принимает ABCDE в качестве входного параметра).

Эта ошибка может происходить в двух случаях. Либо результат ABCDE неприемлем для XYZ, т.к. он имеет не тот тип (это, например, число, а должен быть список):

```
SETCUR 10 15
```

Либо результат имеет правильный тип, но не находится в допустимых пределах:

```
SETCUR [10 45]
```

При появлении такой ошибки Вам следует внимательно вручную проверить что именно Ваша процедура ожидает в качестве входного параметра.

Сообщение об этой ошибке может иметь весьма обескураживающий вид, например:

\*XYZ doesn't like as input\*

Здесь выпало слово ABCDE - это потому, что результат ABCDE является в данном случае пустым словом.

6. \*Not enough space to proceed\*

(Не хватает рабочего пространства).

Это сообщение о том, что наступила нехватка оперативной памяти компьютера для продолжения дальнейших вычислений. В такой ситуации можно попробовать принять

предохранительные меры. Один из приемов - запустить процедуру RECYCLE. Эта команда выполнит то, что на компьютерном жаргоне называют "уборкой мусора". Она разыщет области памяти, выделенные в свое время для каких-то операций, в которых более нет необходимости и освободит дополнительный объем памяти. Может быть, эта мера и не приведет к какому-то положительному результату, поскольку время от времени система сама производит "уборку мусора" автоматически.

Другой, более радикальный прием - удалить из памяти те процедуры, которые Вам более не нужны.

Но команду RECYCLE можно использовать и для повышения быстродействия Ваших программ. Поскольку автоматическая "уборка мусора" может занимать некоторое время, то работа программы может быть задержана. В разных частях программы, в разных процедурах у Вас могут быть разные требования по скорости. Поэтому имеет смысл самому поставить RECYCLE в тех местах, где Вы можете позволить себе ожидание, а не дожидаться, пока автоматика начнет заниматься этим делом в тех местах, где для Вас это не удобно.

Теперь мы рассмотрим ряд приемов, с помощью которых отыскивают и устраняют ошибки в программах. В качестве примера мы вернемся к одной из ранее рассмотренных процедур - это процедура GETNUM из Главы 9. Процедура предназначена для ввода двузначных чисел и изображения мигающего курсора в экранной позиции с координатами x и y. Процедура GETNUM использует в работе процедуру GETDIG, рассмотренную в предыдущей главе.

```
TO GETDIG
  MAKE "DIG READCHAR
  MAKE "CODE ASCII :DIG
  IF OR :CODE=13 :CODE=12[STOP]
  IF OR :CODE<48 :CODE>57[GETDIG]
END
```

Испытание процедуры GETDIG прошло нормально. При этом нажимались различные цифровые клавиши и процедура правильно распечатывала значения DIG и CODE.

После этого испытания была набрана (так, как показано ниже) процедура GETNUM.

```
TO GETNUM :X :Y :NO
  SETCUR SE :X :Y
  TYPE :NO
  FLASH
  TYPE ""
  NORMAL
  GETDIG
  IF AND :CODE=13 NOT EMPTYP :NO
    [OUTPUT :NO]
  IF AND :CODE=12 NOT EMPTYP :NO
    [MAKE "NO BUTLAST :NO]
  IF AND :CODE>13 COUNT :NO<2
    [MAKE "NO WORD :NO :DIG]
  OUTPUT GETNUM :X :Y :NO
END
```

Может быть Вам будет интересно набрать эту процедуру и проследить этапы отладки.

Первое тестирование процедуры выполнялось командой:

```
PRINT GETNUM 10 10 "
```

Все прошло нормально и мигающий вопросительный знак был напечатан в позиции с координатами [10 10]. Но при попытке нажать клавишу 3 появилось сообщение об ошибке:

```
5 is not TRUE or FALSE in GETNUM
```

(5 не имеет значения "ИСТИНА" или "ЛОЖЬ" в процедуре GETNUM).

Единственно, где в процедуре GETNUM могут участвовать логические значения TRUE или FALSE - это три строки IF... Неясно только, какая же из трех работает неправильно.

Для выяснения этого вопроса в процедуру GETNUM были встроены три отладочные строки печати после каждой из строк IF.

```
PRINT "I1
PRINT "I2
PRINT "I3
```

После этого вновь была нажата клавиша "3" и был получен следующий результат:

```
I1
I2
5 is not TRUE or FALSE in GETNUM
```

Теперь стало ясно, что первые две строки IF проходят нормально и неприятности происходят в третьей строке. Три вспомогательные отладочные строки PRINT были удалены, но перед последним IF были поставлены две новые контрольные строки:

```
PRINT :CODE
PRINT COUNT :NO
```

Процедура была запущена еще раз и вновь нажата клавиша "3". Результат был таким:

```
51
0
5 is not TRUE or FALSE in GETNUM
```

Итак, 51 - это код символа "3", а 0 - это количество символов в переменной NO (в этот момент времени). Так откуда же появилось число 5? Для проверки были опять изменены две строки контрольной печати.

```
PRINT :CODE>13
PRINT COUNT :NO<2
```

Вторая строка дала неожиданный результат:

```
< does not like as input
```

Обратим внимание на то, что единственный случай, когда команда "<" может получить пустое слово в качестве входного параметра, может быть только тогда, когда вместо COUNT :NO будет стоять просто :NO. Итак, может быть все дело просто в порядке исполнения операций в этой строке?

Тогда решение будет в использовании круглых скобок так, чтобы сначала выполнялась функция COUNT :NO, а потом ее результат сравнивался бы с числом 2.

```
IF AND :CODE>13 (COUNT :NO)<2
[MAKE "NO WORD :NO :DIG]
```

Итак, что же произошло? Сначала почему-то при сравнении пустого слова :NO и числа 2 прошел результат FALSE (видимо, это некорректное сравнение), а потом COUNT FALSE дал число 5 (по количеству символов в слове FALSE), которое конечно не может служить корректным вводом для последующей операции AND.

Теперь процедура работает нормально? Нажав клавиши "3" и "2", мы получим на экране "32" и мигающий знак вопроса. После нажатия ENTER на экране появится число 32, как результат работы команды PRINT. Прочие тесты показывают, что прочие клавиши игнорируются и более двух цифр ввести в число не удастся.

Но нам надо еще проверить работу DELETE. Запустим процедуру еще раз. Нажмем клавишу "2". На экране появится:

```
2?
```

Теперь нажмем DELETE и на экране появится:

```
??
```

Внимательное исследование показало, что левый знак вопроса появился после стирания цифры 2 на ее месте, там ему и положено быть, а правый знак вопроса - это тот, который остался там с прошлого раза - его не должно бы быть. Прежде чем стирать цифру 2 этот вопросительный знак должен быть погашен печатью поверх него пробела. Об этом мы говорили в предыдущей главе.

После исправления и этой ошибки нажатие на цифру "2" дает:

```
2?
```

А стирание - DELETE дает:

```
?
```

Отлично! Теперь для проверки нажмем на цифру "3" ... и получим новое сообщение об ошибке:

```
Word doesn't like [] as input in GETNUM
```

(команда WORD не принимает [] в качестве входного параметра в процедуре GETNUM)

Локализовать эту ошибку несложно, поскольку в процедуре GETNUM команда WORD встречается только один раз. Но сообщение вполне определенно указывает, что в качестве параметра команды WORD оказалось не слово, а пустой список []. Более того, мы

обнаружили, что проявляется эта ошибка только после использования DELETE и потому может быть не строка, в которой стоит WORD является причиной ошибки, а та строка, которая реализует операцию DELETE.

Теперь можно поэкспериментировать:

```
MAKE "NO 2
```

```
MAKE "NO BUTLAST :NO
```

И попробуйте дать команду

```
PRINT :NO
```

В качестве результата Вы получите пустую строку, что и следовало ожидать. А теперь попробуйте так:

```
MAKE "NO WORD :NO 3
```

и Вы получите сообщение об ошибке такое же, как в предыдущей процедуре. Теперь надо вспомнить об еще одной полезной команде вывода, о которой мы до сих пор не говорили. Мы рассказывали только о PRINT и TYPE, а есть еще команда SHOW. Она отличается тем, что показывает списки вместе с наружными скобками - очень полезная вещь для проверок.

```
SHOW :NO
```

дает:

```
[ ]
```

Вот и открыта причина необычного поведения процедуры. После применения BUTLAST к слову, содержащему только один символ, мы получили не пустое слово, а пустой список. Это фактически системная ошибка. Не исключено, что в той версии языка, с которой работаете Вы, эта ошибка уже ликвидирована, но для нас она послужила наглядным примером технологии отладки программ.

Способ обхода этой ошибки был нами продемонстрирован в предыдущей главе и здесь мы не будем повторяться.

Можно сделать первые выводы. Во-первых, отлаживайте свои процедуры порознь. ЛОГО любезно сообщает вам, в какой процедуре произошла ошибка, но первопричина может быть в другой процедуре. Например, причиной ошибки может быть неправильно переданный параметр, полученный из другой процедуры. Поэтому прежде чем подключать одну процедуру к другой, постарайтесь выполнить в ней все возможные проверки. Может быть, Вам потребуется написать специальные тестирующие процедуры для тестирования отдельных фрагментов будущей сложной структуры.

Например, если Вы пишете программу для карточной игры, то прежде чем подключать к комплексу процедуру, которая будет сдавать карты из колоды в случайном порядке, полезно написать тестирующую процедуру, которая будет просто распечатывать значения карт из колоды и убедиться, что они действительно идут в случайном порядке.

Если Вы убедитесь, что в этой процедуре все в порядке, то столкнувшись впоследствии с проблемами в вышестоящей процедуре, Вы сможете сказать себе "Здесь я все проверил, здесь ошибку можно не искать, будем искать ее во вновь добавленных фрагментах". Хотя, конечно полностью исключить вероятность ошибки удастся не всегда.

Другой метод поиска ошибок - трассирование (трейсинг). Он состоит в прослеживании шаг за шагом того, что делает программа. Этот метод является наиболее важным в тех случаях, когда программа "зависает", т.е. она работает, но не выдает никаких результатов на экран. Возникает впечатление, что она "не работает".

Трейсинг можно делать двумя разными способами. Самый простой - сделать так, чтобы в начале каждой процедуры стоял оператор печати имени этой процедуры. Тогда даже при "зависании" программы Вы сможете увидеть, в какой процедуре или в каких процедурах это "зависание" произошло. Подобный трейсинг делать очень просто, а эффект он дает большой. Когда программа отлажена, то лишние строки печати можно убрать.

Другой прием трейсинга - состоит в прослеживании не процедур, а переменных. Это следующая линия атаки, когда Вы в принципе уже установили, в каких процедурах происходит сбой. В этом случае Вы в нескольких местах этих процедур ставите печать важных переменных (переменной) и по характеру их (ее) изменения выясняете конкретно точку, вызывающую зависание.

Если Вы столкнетесь с тем, что трассирующие операторы ведут печать на экране так быстро, что не удастся визуально отследить и понять, что делает программа, то в паре с операторами печати надо поставить и замедляющие отладочные операторы. Удобным таким оператором является READCHAR, который приостановит работу до тех пор, пока Вы не нажмете клавишу.

Вы можете написать и свою трассирующую процедуру, назвав ее например TRACE. При создании такой процедуры Вы можете использовать три новых примитива.

**COPYDEF** - выполняет новую копию процедуры

**TEXT** - конвертирует процедуру в текстовый список ее операторов. С этим списком может работать ЛОГО, как со списком, а не как с исполняемой процедурой.

**DEFINE** - выполняет обратное действие - конвертирует список инструкций в исполняемую процедуру.

Теперь мы можем создать трассирующую процедуру TRACE, которая будет перед тем, как исполнять очередную строку отлаживаемой процедуры, распечатывать ее содержимое на экране. Вызывается она так:

```
TRACE "PROC
```

**A** отключается

```
UNTRACE "PROC
```

**PROC** - имя отлаживаемой процедуры. Отладочные строки будут печататься только тогда, когда выполняется эта процедура PROC.

```
TO TRACE :APROC
```

```
IF DEFINEDP WORD ". :APROC
```

```
  [PRINT SENTENCE :APROC [ALREADY TRACED] STOP]
```

```
IF NOT DEFINEDP :APROC
```

```
  [PRINT SENTENCE :APROC [NOT A PROCEDURE] STOP]
```

```
COPYDEF WORD ". :APROC :APROC
```

```
MAKE :APROC TEXT :APROC
```

```
MAKE ".P LIST FIRST THING :APROC SENTENCE "PRINT WORD CHAR 34 :APROC
```

```
TREST BUTFIRST THING :APROC
```

```
DEFFINE :APROC :.P
```

```
END
```

```
TO TREST :ALIST
```

```
IF EMPTY :ALIST [STOP] MAKE ".P LPUT LIST "PRINT
```

```
FIRST :ALIST :.P
```

```
MAKE ".P LPUT FIRST :ALIST :.P
```

```
MAKE ".P LPUT MAKE ". READCHAR :.P
```

```
TREST BUTFIRST :ALIST END TO UNTRACE :APROC
```

```
IF NO DEFINEDP WORD ". :AFROC [STOP]
```

```
COPYDEF :APROC WORD ". :APROC
```

```
ERASE WORD ". :APROC
```

```
END
```

Может быть Вы, если захотите, расширите эти отладочные процедуры, чтобы они давали еще больше информации. Придется поэкспериментировать. Может, правда, случиться так, что из-за слишком высокоразвитых отладочных процедур у Вас возникнут проблемы с нехваткой оперативной памяти компьютера. Но, к счастью, необходимость в таких отладочных процедурах почти не возникает. В реальной практике вполне достаточно тех общих принципов отладки, на которые мы уже указали.

Стюарт Николс.

# ПРИМЕНЕНИЕ АСSEMBЛЕРА ДЛЯ СОЗДАНИЯ БЫСТРОРАБОТАЮЩИХ ПРОГРАММ

Перевод с английского Пашорина В.И.  
Редактирование текста "ИНФОРКОМ".

Окончание.

Начало см. с. 9-12, 48-61, 97-103, 139-144, 187-189.

## 10. ПЕРЕВОД БЕЙСИК-ПРОГРАММЫ В МАШИННЫЕ КОДЫ

Если Вы дошли до этой главы с ясным пониманием всего прочитанного, то примите поздравления, теперь у Вас не должно быть проблем с программированием в машинных кодах.

В этой же главе мы разберем способ создания игровой программы полностью в машинных кодах, используя в качестве примера программу на БЕЙСИКе. Мы полагаем, что, Вы уже умеете программировать на этом языке.

Весьма важно перед созданием любой программы отработать ее алгоритм в виде структурной схемы. Это облегчит выполнение перевода BASIC-программы в машинные коды. Мы займемся переводом в машинные коды игровой БЕЙСИК-программы "CROSS", что позволит приобрести навыки для перевода прочих собственных программ.

Идея игровой программы "CROSS" состоит в том, чтобы управляя перемещением человечка вернуть его домой целым и невредимым, несмотря на обилие всевозможных препятствий (Лист\_1).

Главная программа включает в себя инициализацию программных переменных, вывод на экран инструкции пользователю и цикл перемещения человечка по экрану с определением его текущего положения. Существует две точки выхода из этого цикла:

1. Если человек сталкивается с препятствием, то осуществляется переход к процедуре HIT и проводится проверка оставшегося числа жизней. Если это число равно нулю, то игра заканчивается, в противном случае осуществляется переход к основному циклу главной программы.

2. Выход к процедуре HOME. Здесь производится проверка числа занятых домов, а затем увеличивается значение скорости перемещения и добавляется еще один паук-людоед. Из процедуры HOME возврат осуществляется к основному циклу главной программы.

БЕЙСИК-программа довольно проста, хотя для задания UDG, цвета и звука, а также попиксельного перемещения экрана используются фрагменты в машинных кодах, так как на БЕЙСИКе это выполнялось бы крайне медленно. Машинные коды вводятся в программу через оператор POKE. Хорошо разберитесь с работой БЕЙСИК-программы, потому что мы переходим к построчному переводу ее в машинные коды.

Листинг\_1.

Программа CROSS на БЕЙСИКе (с сокращениями).

Внимание: Символы графики пользователя [UDG] в тексте программы взяты в квадратные скобки [...]. Они набираются в графическом режиме.

```
20 CLEAR 32243: GO TO 40
25 BEEP .01,b-a
30 PRINT OVER 1; PAPER 8; INK 8; AT a,y2; "[G]": RETURN
40 PRINT AT 11,5; "Please wait"
50...140 - запись кодов процедуры ROLL через цикл FOR...NEXT с помощью READ...DATA и POKE.
150 LET a=PEEK 23675 + 256*PEEK 23676
```



```

160 FOR b=a TO a+167: READ c: POKE b,c: NEXT c
180 DATA.... Данные по конструкции
..... символов UDГ-графики
270 DATA.... (168 байтов)
280 PRINT AT 11,3; "Do you want instructions?";AT 13,11; "(Y)es"; AT 15,11;"(n)o"
290 LET a$=INKEY$: IF a$="" THEN GO TO 290
295 IF a$<>"y" THEN GO TO 400
300 CLS: PRINT AT 0,11;"OBJECT": PRINT: PRINT "To guide a [G] across a road and a river,
      avoiding [AB] [CDE] [F] [GRRS] [TU]":
310 PRINT: PRINT "A [OP] patrols the central island.":
320 PRINT: PRINT "There are 4 HOMES to be filled. i.e. gaps in fence [NNN] [NNN]"
330 PRINT: PRINT "Once all 4 HOMES are filled the speed will increase, an extra spider is
      added and the HOMES will empty"
340 PRINT AT 18,9; "Press any Key"
350 PAUSE 0
370 CLS: PRINT AT 7,11; "CONTROLS": PRINT
375 PRINT "      !      <O>"; FLASH 1
380 PRINT AT 11,6; "1"; FLASH 0;" 2 3 4 5 6 7 8 ";FLASH 1; "9"; FLASH 0; " "; FLASH 1; "0";
      FLASH 0
385 PRINT AT 18,5; "Press any Key to PLAY"
390 PAUSE 0
400 BRIGHT 1: PAPER 5: BORDER 5: CLS
410 LET hi=0
420 PRINT PAPER 4; AT 10,0;"      [OP]      "
430 LET lives=9: LET score=0: LET home=0: POKE 32425,201: POKE 32450,201: POKE 32469,201
450 PRINT AT 0,0; PAPER 4; "[NNNN]"; PAPER 7; " ";
      PAPER 4; "[NNNNNN]"; PAPER 7;" ";
      PAPER 4; "[NNNNNNNN]"; PAPER 7;" ";
      PAPER 4; "[NNNNNNNN]"; PAPER 7;" ";
      PAPER 4; "[NNNN]"
455 IF home<>0 THEN GO TO 660
460 PRINT PAPER 4; INK 5;
470 PRINT
480 PRINT INK 7;
490 PRINT INK 2;
500 PRINT INK 7;
510 PRINT INK 1;
520 PRINT INK 7;
530 PRINT
540 PRINT PAPER 4;
550 PRINT PAPER 0; INK 7 AT 11,0;
560 PRINT PAPER 0; INK 3
570 PRINT PAPER 0; INK 7
580 PRINT PAPER 0; INK 5
590 PRINT PAPER 0; INK 7
600 PRINT PAPER 0; INK 4
610 PRINT PAPER 0; INK 7
620 PRINT PAPER 0; INK 6
630 PRINT PAPER 4;
640 PRINT PAPER 4;
650 PRINT PAPER 1; INK 7 " SCORE ";AT 21,11;" MEN "; PAPER 5; INK 0; " LIVES "; PAPER 1; INK
      7; " HI SCORE "
660 LET X1=20: LET y1 = 16: LET x2=x1: LET y2=y1
670 PRINT PAPER 8; INK 8; AT x1,y1; " "
680 RANDOMIZE USR 32295
690 IF SCREEN$ (x2,y2)=" " THEN GO TO 880
700 LET a=x2: FOR b=25 TO 35: GO SUB 25: GO SUB 25: NEXT b
730 FOR a=x2 TO 20 STEP 2: GO SUB 25: GO SUB 25: NEXT a
740 LET lives=lives-1: PRINT AT 21,16; lives
750 LET x2 = 20
760 IF lives <> 0 THEN GO TO 680
770 IF hi >score THEN GO TO 790
780 LET hi = score: PRINT AT 21,27; hi
790 PRINT FLASH 1; PAPER 7; AT 12,0; "      GAME OVER      "
800 PRINT AT 14,0; "Another game?      (y)es      (n)o      "

```

```

840 IF INKEY$ = "n" THEN RANDOMIZE USR 0
850 IF INKEY$ <> "y" THEN GO TO 840
860 PRINT PAPER 5; AT 21,7;" " : GO TO 480
880 IF x2<>0 THEN GO TO 1050
890 PRINT PAPER 8; INK 8; AT x1,y1;" " : PRINT AT x2,y2;"[G]"
900 RESTORE 920: FOR a=1 TO 8: READ b,c
910 BEEP b,c: NEXT a
920 DATA .1,11,.1,11,.8,16,.05,11,.05,16,.05,11,.05,16,1,20
930 LET home=home+1: LET score=score+50: PRINT AT 21,7; score
950 IF home/4<>INT(home/4) THEN GO TO 660
960 IF home=4 THEN POKE 32425,0
970 IF home=8 THEN POKE 32450,0
980 IF home=12 THEN POKE 32469,0
985 IF home>36 THEN GO TO 450
990 LET a=RND*30
1000 LET a=a+1
1005 IF a>31 THEN LET a=0
1010 IF SCREEN$(10,a)="" THEN GO TO 1000
1020 IF SCREEN$(10,a+1) = "" THEN GO TO 1000
1030 PRINT PAPER 4; AT 10,a; "[OP]"
1035 RESTORE 920: FOR a=1 TO 8: READ b,c: BEEP b,c: NEXT a
1040 GO TO 450
1050 PRINT PAPER 8; INK 8; AT x2,y2; "[G]"
1060 LET x1=x2: LET y1=y2
1070 IF INKEY$ <> "1" THEN GO TO 1100
1080 BEEP .001,33
1090 LET x2=x2-2: LET score=score+5: PRINT AT 21,7; score
1100 LET y2=y2+(INKEY$="0" AND y2<>31)-(INKEY$="9" AND y2<>0)
1110 GO TO 670

```

**СТРОКА 20.** Эта строка не нужна в окончательной версии, так как она устанавливает верхнюю границу RAMTOP для БЕЙСИК-программы.

**СТРОКИ 25 и 30.** Это две строки подпрограммы, вызываемой в строках 700 и 730. Размещение ее в начальных строках необходимо для сокращения времени поиска при вызове этой подпрограммы, так как всегда поиск нужных подпрограмм начинается с начальных строк БЕЙСИКа. Машинные коды, соответствующие этим двум строкам, размещаются в участке памяти, где будет записана процедура HIT.

**СТРОКА 40.** Для того, чтобы ввести фрагменты программы в кодах с помощью POKE требуется время. Поэтому в БЕЙСИК-программе и введена эта строка с сообщением о необходимости немного подождать. В программе, написанной полностью в кодах, давать такое сообщение, конечно, нет необходимости.

**СТРОКИ 50...140.** Это строки для ввода через POKE процедуры "ROLL", обеспечивающей попиксельное смещение изображения на экране. Понятно, что для программы в машинных кодах эту операцию выполнять также не нужно, так как эти коды будут записаны в соответствующие адреса при загрузке программы в память (см. листинг процедуры ROLL в конце статьи).

**СТРОКИ 150...270.** Это первые строки, которые нуждаются в преобразовании для перехода к программе в машинных кодах. Здесь осуществляется запись в область UDG всех графических элементов пользователя, так как этот участок памяти занимает 158 байтов, то и в строках DATA записаны 168 чисел, определяющих форму UDG.

В дальнейшем, конечно, для задания UDG мы будем непосредственно записывать в участок памяти UDG значение этих 168 байтов. Сначала мы разместим данные, характеризующих форму UDG, начиная с адреса 24001, а затем перешлем их в участок памяти UDG, используя команду LDIR. При этом в регистровую пару DE записывается начальный адрес участка UDG из системной переменной 5C7B (83675), а в регистровую пару HL начальный адрес данных, характеризующих форму UDG. Процедура пересылки, написанная в машинных кодах, размещается сразу же за данными 168 байтами. Вызов этой процедуры поэтому производится по команде RANDOMIZE USR 24169.

### UDG-графика.

5DC1H

```
DEFB 0F 12 22 7F FF FF 28 10
DEFB 80 40 20 FE FE FF 28 10
DEFB 7F 7F 7F 7F 7F FF 15 08
DEFB FF FE FE FE FF FF 40 80
DEFB 00 F8 C4 C4 FE FE 28 10
DEFB 18 18 24 7E 3C 5A A5 42
DEFB 38 28 92 72 38 38 28 6C
DEFB 01 02 04 7F 7F FF 14 08
DEFB F0 48 44 FE FF FF 14 08
DEFB 00 1F 23 23 7F 7F 14 08
DEFB 7F 7F 7F 7F FF FF 02 01
DEFB FE FE FE FE FE FF A8 10
DEFB 10 29 C7 00 26 00 00 00
DEFB 00 44 FF 44 44 FF 44 00
DEFB 00 22 55 8F 97 A3 A0 00
DEFB 00 44 AA F1 E9 C5 05 00
DEFB 10 10 10 FE 3F IF 0F 07
DEFB 00 00 00 00 1E FF FF FF
DEFB 60 7C 54 78 7F FF FE 7C
DEFB 00 00 03 02 0F 3F FF 00
DEFB 05 0C 96 F0 E0 55 FF 00
```

5E69

```
24169 LD DE, (5C7B)
24173 LD HL, 5DC1
24175 LD BC, 00AB
24179 LDIR
24181 JR, 5EA2
```

**СТРОКА 280.** Сообщение, выводимое на экран по команде PRINT в этой строке, можно выводить на экран, используя процедуру ОЗУ, находящуюся по адресу 3252 (203CH). Машинные коды этой процедуры записаны в участке 5EA2... 5EAF, а сам текст сообщения - в участке 5E77...5EA1 (Листинг-4).

### Печать сообщения.

5E77H

```
DEFB 16 0B 03 44 5F 20 79 6F
DEFB 75 20 77 61 6E 74 20 69
DEFB 6E 73 74 72 75 63 74 69
DEFB 6F 6E 73 3F 16 0D 0B 28
DEFB 79 29 65 73 16 0F 0B 28
DEFB 6E 29 6F
```

5EA2

```
24225 LD A, 02
24228 CALL 1601H
24231 LD DE, 5E77H
24234 LD BC, 002BH
24237 CALL 203CH
```

**СТРОКИ 290 и 295.** Эти строки необходимы, чтобы приостановить выполнение программы до тех пор, пока не будет нажата клавиша. Если будет нажата клавиша "у", то на экране появится инструкция по правилам игры. Если же будет нажата любая другая клавиша, то произойдет переход к адресу 608AH. Машинные коды, соответствующие этим строкам, записаны по адресу с 5EB0H по 5EBEH (Лист.5)

### Проверка нажатия клавиши "у".

```
24240 HALT
24241 BIT 5, (IY+01)
24245 JR Z, -07
```

```
24247 LD A, (5C08H)
24250 CP 79
24252 JP NZ, 608AH
```

**СТРОКА 300:1 CLS** - Перевод этой команды в машинные коды мы уже разбирали (Листинг-6).

#### Очистка экрана.

```
5EBFH
24255 LD A, 02
24257 CALL 1601H
24260 CALL 0D6BH
24263 JP 5F73H
```

**СТРОКИ 300:2...350.** Для вывода сообщений на экран снова будем использовать процедуру вывода на экран последовательности символов ПЗУ, коды которых записаны в участок памяти с именем DATA. Коды символов первого сообщения находятся в памяти с 5ECAH по 5F72H, а второго сообщения 5F84H...5FF9H. Процедуры же вывода PRINT DATA находятся соответственно 5F73H...5F83H и 5FFAH... 6002H (Листинг\_7).

В ячейках с 6003H по 600CH находятся машинные коды, реализующие команду PAUSE 0. Не забывайте, что бит 5 переменной 23611 (Y+1) должен быть сброшен (ожидание нажатия клавиши).

#### Печать сообщений, пауза.

```
5ECAH
DEFB 16 00 0B 4F 42 4A 45 43
DEFB 54 0D 0D 54 6F 20 67 75
DEFB 69 64 65 20 61 20 96 20
DEFB 61 63 72 6F 73 73 20 61
DEFB 20 72 6F 61 64 20 61 6E
DEFB 64 20 61 72 69 76 65 72
DEFB 2C 61 76 6F 69 64 69 6E
DEFB 67 20 90 91 20 92 93 94
DEFB 20 95 20 A0 A1 A1 A2 20
DEFB A3 A4 0D 0D 41 20 9E 9F
DEFB 20 70 61 74 72 6F 6C 73
DEFB 20 74 68 65 20 63 65 6E
DEFB 74 72 61 6C 20 69 73 6C
DEFB 61 6E 64 EE 0D 0D 54 68
DEFB 65 72 65 20 61 72 65 20
DEFB 34 20 48 4F 4D 45 53 20
DEFB 74 6F 20 62 65 20 66 69
DEFB 6C 6C 65 64 2E 20 69 2E
DEFB 65 2E 20 67 61 70 73 20
DEFB 69 6E 20 65 65 6E 63 65
DEFB 20 9D 9D 9D 20 9D 9D 9D
DEFB 0D
```

```
5F73H
24435 LD A, 02
24437 CALL 1601H
24440 LD DE, 5ECAH
24443 LD BC, 00A9H
24446 CALL 203CH
24449 JP 5FFAH
```

```
5FB4H
DEFB 0D 4F 6E 63 65 20 61 6C
DEFB 6C 20 34 20 48 4F 4D 45
DEFB 53 20 61 72 65 20 66 69
DEFB 6C 6C 65 64 20 74 68 65
DEFB 20 73 70 65 65 64 20 77
```

```

DEFB 59 6C 6C 20 69 6E 63 72
DEFB 65 61 73 65 2C 20 61 6E
DEFB 20 65 78 74 72 61 20 73
DEFB 70 69 64 65 72 20 59 73
DEFB 20 61 64 64 65 64 20 61
DEFB 6E 64 20 74 68 65 20 48
DEFB 4F 4D 45 53 20 77 69 6C
DEFB 6C 65 6D 70 74 79 16 12
DEFB 09 50 72 65 73 73 20 61
DEFB 6E 79 20 6B 65 79

```

```

5FFAH
24570 LD DE,5F84H
24573 LD BC,0076H
24576 CALL 203CH
24579 RES 5,(IY+01)
24583 HALT
24584 BIT 5,(IY+01)
24568 JR Z,-07

```

**СТРОКА 370: 1 CLS.** В участке памяти с 600E по 6012 записаны машинные коды процедуры очистки экрана. Здесь применяется другой способ, основанный на применении процедуры ПЗУ 0E44H, которая очищает столько строк экрана (считая снизу), сколько установлено в регистре B(Листинг\_8).

#### Очистка экрана.

```

600EH
24590 LD B,18H
2459E CALL 0E44H
24595 JP 6076H

```

**СТРОКИ 370:2...390.** Коды символов, выводимых на экран, записаны в участке 6016H... 6075H, а сама же процедура печати находится в адресах - 6076H... 607EH. Здесь опять же имеются машинные коды, реализующие команду PAUSE 0 (Листинг\_9).

#### Печать сообщений, пауза.

```

6015H
DEFB 16 07 0B 43 4F 4E 54 52
DEFB 4F 4C 53 0D 0D 20 20 20
DEFB 20 20 20 5E 20 20 20 20
DEFB 20 20 20 20 20 20 20 20
DEFB 20 20 20 3C 30 3E 12 01
DEFB 16 0B 06 31 12 00 20 32
DEFB 20 33 20 34 20 35 20 36
DEFB 20 37 20 38 30 12 01 39
DEFB 12 00 20 12 01 30 12 00
DEFB 16 12 05 50 72 65 73 73
DEFB 20 61 6E 79 20 6B 65 79
DEFB 20 74 6F 20 50 4C 41 59

```

```

6076H
24694 LD DE,6016H
24897 LD BC,0060H
24700 CALL 203CH
24703 RES 5,(IY+01)
24707 HALT
24708 BIT 5,(IY+01)
24712 JR Z,-07

```

**Строка 400.** В первой главе обсуждался порядок перевода всех операторов этой строки в машинные коды. Вызов процедуры CLS здесь необходим для установки на экране соответствующих атрибутов, как и в БЕЙСИК-программе. Машинные коды записаны в

участке 608АН...609ВН.

#### Установка атрибутов

```
608АН
24714 LD A, 05
24716 CALL 229ВН
24719 LD A, 68Н
24721 LD (5С8D), A
24724 LD A, 02
24725 CALL 1601Н
```

**СТРОКА 410.** В этой строке переменной hi, являющейся показателем достигнутого уровня, присваивается нулевое начальное значение. Полагая, что значение этой переменной не может быть больше 65535, выделим в буфере принтера 2 байта для хранения значения этой переменной. Соответствующие этому оператору машинные коды записаны в участке памяти 609СН ...60А1Н.

#### Инициализация переменной hi.

```
609СН
24732 LD HL, 0000
24735 LD (5В00Н), HL
24738 JR +25Н
```

**СТРОКА 420.** Коды символов сообщения, выводимого на экран, записаны в адресах 60А4Н...60С8Н. Процедура печати - 60С9Н...60D6Н.

#### Печать строки.

```
60А4Н
DEFB 11 04 16 0А 00 20 20 20
DEFB 20 20 20 20 20 20 20 20
DEFB 20 20 20 20 9Е 9F 20 20
DEFB 20 20 20 20 20 20 20 20
DEFB 20 20 20 20 20
```

```
60С9Н
24777 LD A, 02
24779 CALL 1601Н
24782 LD DE, 60А4Н
24785 LD BC, 0025Н
24788 CALL 203СН
```

**СТРОКА 430.** В этой строке определяются исходные значения количества "жизней", занятых домов и достигнутого уровня. Эти значения также хранятся в буфере принтера по следующим адресам:

количество "жизней" - 5В02  
достигнутый уровень - 5В03/4  
количество домов - 5В05

Машинные коды, соответствующие этой строке, см. 60D7Н...60Е5Н.

#### Инициализация переменных.

```
60D7Н
24791 LD A, 09
24793 LD(5В02Н), A
24796 LD HL, 0000
24799 LD (5В03Н), HL
24802 XOR A
24803 LD (5В05), A
```

**СТРОКА 440.** Операторы этой строки обнуляют скорость перемещения и устанавливают команды RET в соответствующих ячейках подпрограммы LINE ROLL.

Машинные коды см. 60E6H...60F3H.

#### Коррекция процедуры ROLL.

```
60E6H
24806 LD A, C9H
24808 LD (7EA9H), A
24811 LD (7EC2H), A
24814 LD (7ED5H), A
24817 JP 6129H
```

СТРОКА 450. Коды символов этой строки и процедура печати:

#### Печать строки.

```
60F4H
DEFB 16 00 00 11 04 9D 9D 9D
DEFB 9D 11 07 20 11 04 9D 9D
DEFB 9D 9D 9D 9D 11 07 20 11
DEFB 04 9D 9D 9D 9D 9D 9D 9D
DEFB 11 07 20 11 04 9D 9D 9D
DEFB 9D 9D 9D 9D 11 07 20 11
DEFB 04 9D 9D 9D 9D
```

```
6129H
24873 LD A,
24875 CALL 1601H
24878 LD DE, 60F4H
24881 LD BC, 0035
24884 CALL 203CH
```

СТРОКА 455. В этой строке проверяется количество домов и, если оно равно нулю, то осуществляется перевод к строке 660. Для реализации этой строки в машинных кодах необходимо извлечь содержимое ячейки 5B05H, где записано количество домов, записать это значение в регистр A и использовать для проверки на нулевое значение команду AND A. Если флаг нуля (ZERO) после этой операции не будет установлен, то осуществляется переход к процедуре PRINT DISPLAY. Если же этот флаг не включен, то эта процедура пропускается и начинается игра. Машинные коды, соответствующие этой строке, записаны в ячейках 6137H...6140H.

#### Проверка переменной home.

```
6137H
24687 LD A, (5B05H)
24890 AND A
24891 JP NZ, 641EH
24894 JP 63D2H
```

СТРОКИ 460...640. Машинные коды, соответствующие этим строкам, занимают большой объем памяти. Коды всех символов сообщений этих строк записаны в участке памяти 6141H... 63D1H, а процедура печати - 63D2H...63DFH.

#### Инициализация экрана.

```
6141H
DEFB 11 04 10 05 8C 8C 8C 8C  DEFB 9C 9C 20 20 20 20 20 9C  DEFB 9C 9C 20 20 20 20 10 01
DEFB 8C 8C 8C 8C 8C 8C 8C 8C  DEFB 9C 9C 20 20 20 20 20 20  DEFB A1 A2 20 20 20 20 20 00
DEFB 8C 8C 8C 8C 8C 8C 8C 8C  DEFB 9C 10 02 20 20 A3 A4 20  DEFB A1 A1 A2 20 20 20 20 20
DEFB 8C 8C 8C 8C 11 05 10 00  DEFB 20 20 20 20 A3 A4 20 20  DEFB 20 A0 A1 A1 A1 A1 AE 20
DEFB 20 A0 A1 A1 A2 20 20 20  DEFB 20 20 20 A3 A4 20 20 20  DEFB 20 20 20 20 A0 A1 A1 A1
DEFB A0 A1 A1 A1 A2 20 20 20  DEFB 20 A3 A4 20 20 20 20 20  DEFB 10 07 20 20 20 9C 9C 20
DEFB 20 A0 A1 A1 A3 20 20 20  DEFB 20 A3 A4 20 10 07 9C 9C  DEFB 20 9C 9C 9C 9C 9C 20 20
DEFB A0 A1 A1 A2 20 20 20 20  DEFB 20 20 20 9C 20 20 20 20  DEFB 20 9C 9C 20 E0 20 20 20
DEFB 10 07 20 20 20 9C 9C 9C  DEFB 20 20 9C 9C 9C 9C 9C 20  DEFB 20 9C 9C 9C 20 20 20 80
DEFB 20 20 20 20 20 9C 9C 9C  DEFB 20 20 20 20 20 20 20 9C  DEFB 20 20 10 00 A4 20 20 20
```

|                                 |                              |                                 |
|---------------------------------|------------------------------|---------------------------------|
| DEFB 20 20 20 20 20 20 A3 A4    | DEFB 07 2D 2D 2D 2D 2D 3D 2D | DEFB 07 2D 2D 2D 2D 2D 2D 2D    |
| DEFB 20 20 20 20 20 20 A3 A4 20 | DEFB 2D 2D 2D 2D 2D 2D 2D 2D | DEFB 2D 2D 2D 2D 2D 2D 2D 2D    |
| DEFB 20 20 20 20 20 20 A3 A4 20 | DEFB 2D 2D 2D 2D 2D 2D 2D 2D | DEFB 2D 2D 2D 2D 2D 2D ED 2D 2D |
| DEFB 20 20 20 20 A3 11 04 9D 9D | DEFB 2D 2D 2D 2D 2D 2D 2D 2D | DEFB 2D 2D 2D 2D 2D 2D 2D 2D    |
| DEFB 9D 9D 9D 9D 9D 9D 9D 9D    | DEFB 2D 10 05 20 90 92 20 20 | DEFB 2D 10 06 9A 9B 20 20 20    |
| DEFB 9D 9D 9D 9D 9D 9D 9D 9D    | DEFB 20 20 90 91 20 20 20 E0 | DEFB 97 98 20 20 97 98 20 95    |
| DEFB 9D 9D 9D 9D 9D 9D 9D 9D    | DEFB 90 91 20 20 90 91 20 20 | DEFB 20 20 95 20 20 99 9A 9B    |
| DEFB 9D 9D 9D 9D 9D 9D 11 00    | DEFB 20 20 20 20 20 20 90 91 | DEFB 9A 9B 20 20 20 95 20 20    |
| DEFB 10 07 16 0B 00 9D 9D 9D    | DEFB 20 20 20 10 07 3D 3D 3D | DEFB 20 20 99 11 04 10 00 9D    |
| DEFB 9D 9D 9D 9D 9D 9D 9D 9D    | DEFB 3D 3D 3D 3D 3D 3D 3D 3D | DEFB 9D 9D 9D 9D 9D 9D 9D 9D    |
| DEFB 9D 9D 9D 9D 9D 9D 9D 9D    | DEFB 3D 3D 3D 3D 3D 3D 3D 3D | DEFB 9D 9D 9D 9D 9D 9D 9D 9D    |
| DEFB 9D 9D 9D 9D 9D 9D 9D 9D    | DEFB 3D 3D 3D 3D 3D 3D 3D 3D | DEFB 9D 9D 9D 9D 9D 9D 9D 9D    |
| DEFB 9D 9D 9D 9D 9D 10 03 80    | DEFB 3D 3D 3D 3D 3D 10 04 98 | DEFB 9D 9D 9D 9D 9D 9D 9D 11    |
| DEFB 90 91 20 95 20 20 90 91    | DEFB 20 20 20 20 20 97 98 20 | DEFB 04 20 20 20 20 20 20 20    |
| DEFB 20 20 92 93 92 93 94 20    | DEFB 20 20 97 98 20 20 20 20 | DEFB 20 20 20 20 20 20 20 20    |
| DEFB 95 20 95 20 20 20 92 93    | DEFB 20 20 20 97 98 20 20 20 | DEFB 20 20 20 20 20 20 20 20    |
| DEFB 94 20 20 20 20 20 20 10    | DEFB 20 20 20 20 20 97 10    | DEFB 20 20 20 20 20 20 20 20    |

```

63D2H
25554 LD A, 02
25556 CALL 1601H
25559 LD DE, 6141H
25562 LD BC, 0291H
25565 CALL 203CH

```

**СТРОКА 650.** Для реализации этой строки в машинных кодах необходимо предварительно преобразовать число, записанное в ячейке 5B02H. Для этого оно извлекается из этой ячейки, суммируется с числом 48 и результат записывается в ячейку 6402. Машинные коды этой операции 63E0H...63E9H. Коды символов сообщения для вывода на экран записаны в 63EАН...640FH, а процедура печати - 6410H...641EH.

#### Инициализация строки результата

```

63E0H
25568 LD A, (5B02H)
25571 ADD A, 30
25573 LD (6401H), A
25576 JR +26

```

```

63EАН
DEFB 11 01 10 07 20 53 43 4F
DEFB 52 45 20 16 15 0B 20 4D
DEFB 45 4E 20 11 05 10 00 39
DEFB 11 01 10 07 20 48 49 2D
DEFB 53 43 4F 52 45 20

```

```

6410H
25616 LD A, 02
25618 CALL 1601H
25621 LD DE, 63E1H
25624 LD BC, 0026H
25627 CALL 203CH

```

**СТРОКА 660.** В этой строке определяются исходные координаты положения человечка на экране x1 и y1, а также координаты его нового положения x2 и y2. Для программы в машинных кодах эти координаты хранятся также в буфере принтера:

```

x1 - 5B06H x2 - 5B08H
y1 - 5B07H y2 - 5B09H

```

Для записи в буфер принтера координат лучше использовать шестнадцатиричную систему счисления, тогда процедура записи будет иметь следующий вид:

Инициализация координат героя.



```
641FH
25630 LD HL, 1014H
25633 LD (5B06H), HL
25636 LD (5B08H), HL
```

Из этого примера понятно, что использовать десятичную систему счисления для записи координат в регистровую пару HL не удобно, т.к. трудно будет представить положение объекта на экране.

**СТРОКА 670.** Поскольку здесь необходимо выводить на экран только один символ, то лучше использовать команду RST 10H.

#### Печать героя.

```
6427H
25639 LD A, 02
25641 CALL 1601H
25644 LD A, 11H
25646 RST 10H
25647 LD A, 08
25649 RST 10H
25650 LD A, 10H
25652 RST 10H
25653 LD A, 08
25655 RST 10H
25656 LD A, 16H
25658 RST 10H
25659 LD A, (5B06H)
25662 RST 10H
25663 LD A, (5B07H)
25666 RST 10H
25667 LD A, 20H
25669 RST 10H
```

**СТРОКА 680.** В этой строке производится вызов процедуры ROLL, записанной в машинных кодах с адреса 7E27H. Код, соответствующий этой строке:

#### Вызов процедуры ROLL.

```
6446H
25670 CALL 7E27H
```

**СТРОКА 690.** в этой строке проверяется состояние символьной ячейки с координатами x2, y2. Для реализации этой строки в машинных кодах будем использовать процедуру SCREEN\$, описанную в главе 8, передающую значение параметров исследуемой ячейки на стек калькулятора и затем в регистры A,B,C,D,E. Для определения состояния ячейки используем:

```
LD A, (DE)
CP 20H
```

Помните, что для продолжения программы, после использования калькуляторного стека, необходимо очистить рабочий участок. Если в исследуемой ячейке не пробел, то осуществляется переход к процедуре HIT, которая начинается с адреса 6548H. В БЕЙСИК-программе соответственно переход к строке 880. Машинные коды, соответствующие этой строке записаны в участке 6449H...645DH.

#### Проверка местоположения героя.

```
6449H
25673 LD BC, (5B08H)
25677 CALL 2538H
25680 CALL 2BF1H
25683 LD A, (DE)
25684 PUSH AF
25685 CALL 16BFH
```

```
25688 POP AF
25689 CP 20H
25691 JP NZ, 6548H
```

**СТРОКА 880.** В этой строке проверяется: достиг ли человек вершины экрана, т.е. дошел ли он до дома. В машинных кодах для этого записываем значение координаты x2 в регистр A:

```
LD A, (5B08)
```

и сравниваем это значение с нулем. Если это значение равно нулю, то выполняется переход к процедуре HOME, которая начинается с адреса 66A6H. В Бейсик-программе соответственно осуществляется переход к строке 1050. Машинные коды, соответствующие этой строке 645EH...6465H.

#### Проверка достижения "дома"

```
645EH
25694 LD A, (5B08H)
25697 CP 00
25699 JP Z, 66A6H
```

**СТРОКА 1050.** Здесь также для вывода на экран одного символа в позицию с координатами x2, y2 лучше использовать команду RST 10H, как и в строке 670. Машинные коды - 6466H... 6484H.

#### Печать героя в новой координате

```
6466H
25702 LD A, 02
25704 CALL 1601H
25707 LD A, 11
25709 RST 10H
25710 LD A, 08
25712 RST 10H
25713 LD A, 10H
25715 RST 10H
25716 LD A, 08
25718 RST 10H
25719 LD A, 16H
25721 RST 10H
25722 LD A, (5B08H)
25725 RST 10H
25726 LD A, (5B09H)
25729 RST 10H
25730 LD A, 96H
25733 RST 10H
```

**СТРОКА 1060.** Для реализации этой строки в машинных кодах достаточно переписать значение ячеек 5B08 и 5B09 в ячейки 5B06 и 5B07 соответственно.

#### Запоминание новой координаты.

```
6485H
25733 LD HL, (5B08H)
25736 LD (5B06H), HL
```

**СТРОКА 1070.** Чтобы улучшить работу програна в машинных кодах для проверки состояния клавиатуры и контроля нажатия клавиши лучше использовать не системную переменную LAST KEY, а непосредственно команду IN A,(C). Причем, чтобы проверить нажата ли клавиша "1", достаточно проверить состояние только клавиш "1"... "5" так, как это описано в главе 5. Машинные коды - 648BH...6493H.

#### Проверка нажатия клавиши "1".

```
648BH
```

```
25739 LD BC, F77EH
25742 IN A, (C)
25744 BIT 0, A
25746 JR NZ, +1C
```

**СТРОКА 1080.** Поскольку длительность и тональность звукового сигнала принципиального значения не имеют, то для реализации этой строки в машинных кодах достаточно только вызвать процедуру BEEP из ПЗУ - CALL 03B5H.

Генерация звукового сигнала.

```
6494H
25748 LD HL, 0032H
25751 LD DE, 0005H
25754 CALL 03B5H
```

**СТРОКА 1090.** Для записи этой строки в машинных кодах необходимо извлечь содержимое ячейки 5B08 и вычесть из этого значения число 2, а результат вновь записать в эту же ячейку. Значение текущего счета, записанное в ячейках 5B03/4, увеличивается на 5 и вновь записывается в эти ячейки. Машинные коды этих операций - 649DH...64AFH. Для вывода на экран текущего значения счета используются процедуры STACK BC и PRINT VALUE ON STACK, но первоначально определяются параметры PRINT AT. Машинные коды - 64B1H...64C8H.

Коррекция счета (score).

```
649D2H
25757 LD A, (5B08H)
25760 DEC A
25761 DEC A
25763 LD (5B08H), A
25765 LD HL, (5B03H)
25768 INC HL
25769 INC HL
25770 INC HL
25771 INC HL
25772 INC HL
25773 LD (5B03H), HL
25776 NOP
25777 LD A, 02
25779 CALL 1601H
25782 LD A, 16H
25785 LD A, 15H
25787 RST 10H
25788 LD A, 07
25790 RST 10H
25791 LD BC, (5B03H)
25795 CALL 2D2BH
25798 CALL 2DE3H
```

**СТРОКА 1100.** Опять же, для чтения состояния клавиатуры и установки значения у2 используем команду IN A,(C). В программе в машинных кодах добавлена процедура, которой нет в БЕЙСИК программе. В участке памяти 64F3H...64F4H записаны машинные коды, которые составляют процедуру проверки нажатия клавиши "6". Если эта клавиша нажата, то осуществляется выход из программы. На этом главный цикл программы заканчивается.

Если теперь Вы запустите эту программу, то перемещение человечка по экрану будет настолько быстрым, что Вы не будете его замечать. Поэтому необходимо в главный цикл программы в машинных кодах включить замедляющую процедуру.

Этот же эффект можно получить, если в главный цикл включить дополнительно процедуру музыкального сопровождения перемещения. Машинные коды этой процедуры 64F5H...650BH. Завершающей командой главного цикла является команда JP 6427,

обеспечивающая переход на начало главного цикла. В БЕЙСИК-программе это выполняется переходом к строке 670.

#### Опрос клавиатуры, музыка.

```
64C9H
25801 LD BC, EFFE H
25804 IN A, (C)
25806 BIT 0, A
25808 JR NZ, +0EH
25810 LD A, (5B09H)
25813 CP 1F
25815 JR Z, +01
25817 INC A
25818 LD (5B09H), A
25831 JP 64F5H
25834 BIT 1, A
25836 JR NZ, 02
25838 LD A, (5B09H)
25831 CP 00
25833 JR Z, +01
25835 DEC A
25836 LD (5B09H), A
25839 JP 64F5H
```

```
64F3H
25843 BIT 4, A
25844 RET Z
```

```
64F5H
25845 LD B, 0A
25847 LD DE, 0078H
25850 LD HL, 0001H
25853 PUSH HL
25854 PUSH DE
25855 PUSH BC
25856 CALL 03B5
25859 POP BC
25860 POP DE
25861 POP HL
25883 INC A
25864 LD L, A
25865 NOP
25866 DJNZ -0F
25868 JP 6427
```

#### Процедура HIT

**СТРОКА 700.** В этой строке вводится новая переменная  $a = x2$ . Для того, чтобы сохранить значение этой переменной, будем использовать свободную ячейку 5CB0H из области системных переменных. Переменная  $b$  цикла FOR/NEXT записывается в регистр B:

```
LD B, 19H (25D)
```

и в ячейку 5CB1H для использования в следующей процедуре. Машинные коды, соответствующие этой строке - 6548H...6561H. Вызов процедуры, соответствующей БЕЙСИК-оператору GO SUB 35, выполняется командой CALL 650FH.

Непосредственно связаны с этой строкой строки 25 и 30 и мы рассмотрим их здесь совместно в едином блоке.

**СТРОКА 25.** Для перевода этой строки в машинные коды необходимо использовать калькуляторный стек, куда записывается число 0.1 и результат операции  $b-a$ . Вычитание проще выполнить, используя последовательность команд:

```
LD HL, (5CB0H)
LD A, H
SUB L
```

Затем, используя процедуру 2D28H переписываем значение со стека в регистр A и

вызываем процедуру BEEP - CALL 03F8H. Машинные коды, соответствующие этой строке, 650F...6521.

СТРОКА 30. Для вывода символа на экран соответственно в позицию а,72 используем команды LD A,n и RST 10H. Машинные коды - 6522H...6547H.

Расчет b-a, генерация звука.

```
650FH
25871  RST 28H ; Калькулятор.
DEFB  34 EA 23 D7 0A 3D 38
25679  LD HL, (5CB0H)
25882  LD A, H
25883  SUB L
25884  CALL 2D2BH
25887  CALL 03F8H
```

Печать "героя"

```
6522H
25890  LD A, 02
25892  CALL 1601H
25895  LD A, 15H
25897  RST 10H
25898  LD A, 01
25900  RST 10H
25901  LD A, 11H
25903  RST 10H
25904  LD A, 08
25906  RST 10H
25907  LD A, 10H
25909  RST 10H
25910  LD A, 68
25912  RST 10H
25913  LD A, 16H
25915  RST 10H
25916  LD A, (5CB0H)
25919  RST 10H
25920  LD A, (5B09H)
25923  RST 10H
25924  LD A, 96H
25925  RST 10H
25927  RET
```

Создание переменной "а".

```
6548H
25928  LD A, (5B08H)
25931  LD (5CB0H), A
25934  LD B, 19H
25936  LD A, B
25937  PUSH BC
25938  LD (5CB1), A
25941  CALL 650FH
25944  CALL 650FH
25947  POP BC
25048  INC B
25949  LD A, B
25950  CP 23
25952  JR NZ, -11
```

СТРОКА 730. Поскольку переменная цикла этой строки у нас хранится в ячейке 5CB0H, то организовать цикл в машинных кодах не представляется особенно трудным. Для этого необходимо извлекать это значение из этой ячейки, увеличивать его на 2 при каждом проходе и сравнивать с числом 14H (20D). Машинные коды - 6562H...6573H

#### Цикл по "а".

```
6562H
25954  CALL 650FH
25957  CALL 650FH
25960  LD A, (5CB0H)
25963  INC A
25964  INC A
25965  LD (5CB0H), A
25968  CP 14H
25970  JR NZ, -12H
```

**СТРОКА 740.** В этой строке производится подсчет числа "жизней". Для реализации этой строки в машинных кодах необходимо извлекать значение из ячейки 5B02H, уменьшать его на 1 и результат опять записывать в эту ячейку. Для вывода на экран этого значения необходимо первоначально установить необходимые параметры PRINT AT, а затем к числу, записанному в ячейке 5B02H добавить 30H(48D) для перевода его в код ASCII. Машинные коды 6574H...658EH.

#### Коррекция числа "жизней".

```
6574H
25972  LD A, (5B02H)
25975  DEC A
25976  LD (5B02H), A
25979  LD A, 02
25981  CALL 1601H
25984  LD A, 16H
25986  RST 10H
25987  LD A, 16H
25989  RST 10H
25990  LD A, 10H
25992  RST 10H
25993  LD A, (5B02H)
25996  ADD 30H
25998  RST 10H
```

**СТРОКА 750.** В этой строке переписывается значение переменной x2 так, чтобы x2 = 14H (20D). Машинные коды 658FH...6593H.

#### Коррекция переменной x2.

```
658FH
25999  LD A, 14H
26001  LD (5B02H), A
```

**СТРОКА 760.** В этой строке выполняется условный переход. Если число "жизней" не равно 0, то выполняется возврат к началу главного цикла - строке 680. Для программы в машинных кодах это переход по адресу 6446. Машинные коды, соответствующие этой строке - 6594H...659AH.

#### Проверка на конец "жизней".

```
6594H
26004  LD A, (5B02H)
25007  AND A
26008  JP NZ, 5446H
```

### Процедура END GAME

**СТРОКА 770.** В этой строке сравнивается достигнутый счет в этой игре с лучшим результатом прошлых игр. Если новый результат выше, то осуществляется переход к строке 790. Эту операцию легко выполнить для программы в машинных кодах, используя команду SBC HL, DE

и проверяя значение флага переноса (CARRY). Если флаг будет включен после

команды SBC, то выполняется переход по адресу 6611H. Машинные коды для этой строки - 659BH...65A6H.

#### Проверка на новый рекорд.

```
659BH
26011  XOR A
26012  LD HL, (5B03H)
26015  LD DE, (5D00H)
26019  SBC HL, DE
25021  JR C, +6AH
```

**СТРОКА 780.** В этой строке устанавливается новое значение лучшего результата и это значение выводится на экран. В машинных кодах для выполнения этой операции достаточно переписать значения из ячеек 5B03/4 в ячейки 5B00/1. Для вывода на экран используем процедуры STACK VALUE IN BC и PRINT TOP VALUE ON CALCULATOR STACK, предварительно установив параметры PRINT AT. Машинные коды 65A7H...66C2H.

#### Коррекция рекорда.

```
65A7H
26023  LD HL, (5B03H)
25026  LD (5B00H), HL
26029  PUSH HL
26030  POP BC
26031  CALL 2D2BH
26034  LD A, 02
26036  CALL 1601H
25039  LD A, 16H
26041  RST 10H
26042  LD A, 15H
26044  RST 10H
26045  LD A, 1BH
25047  RST 10H
26046  CALL 2DE3H
26051  JR +4C
```

**СТРОКИ 790 и 800.** В этих строках печатаются сообщения. Коды символов находятся в 65C5H...6610H, а сама процедура печати - 6611H...6619H.

#### Печать сообщений.

```
65C5H
DEFB 12 01 11 07 16 0C 00 20
DEFB 20 20 20 20 20 20 20 20
DEFB 20 20 47 41 4D 45 20 20
DEFB 4F 56 45 52 20 20 20 20
DEFB 20 20 20 20 20 20 20 16
DEFB 02 00 12 00 20 20 41 6E
DEFB 6F 74 68 65 72 20 67 61
DEFB 6D 65 20 3F 20 28 79 29
DEFB 65 73 20 20 28 6E 29 6F
DEFB 20 20 20 20
```

```
6611H
26129  LD DE, 65C5H
26132  LD BC, 004CH
26135  CALL 203CH
```

**СТРОКИ 840...850.** В машинных кодах, соответствующих этим строкам, для проверки нажатой клавиши используется системная переменная LAST KEY, а после проверки осуществляется соответствующий переход. Если ни клавиша "y", ни клавиша "n" не будут нажаты, то осуществляется возврат к циклу ожидания нажатия клавиш.

#### Проверка нажатой клавиши.

```
661AH
26138  HALT
26139  BIT 5, (IY+01)
26143  JR Z, -07
36145  RES 5, (IY+01)
26149  LD A, (5C0BH)
26152  CP 6EH
26154  JR NZ, +03
26156  CALL 0000
26159  CP 79H
26161  JR NZ, -19
```

**СТРОКА 860.** Перед тем, как выполнится переход к началу главного цикла по команде JP 60C9H, переписывается значение лучшего результата (переменная hi) с помощью команд LD A,n : RST 10H.

#### Печать нового рекорда.

```
661AH
26163  LD A, 02
26165  CALL 1601H
26168  LD A, 11H
26170  RST 10H
26171  LD A, 05
26173  RST 10H
26174  LD A, 16H
26176  RST 10H
26177  LD A, 15H
26179  RST 10H
26180  LD A, 07
26182  RST 10H
26183  LD A, 20H
26185  RST 10H
26186  LD A, 20H
26188  RST 10H
26189  LD A, 20H
26191  RST 10H
26192  LD A, 20H
26194  RST 10H
26195  JP 60C9H
```

### ПРОЦЕДУРА НОМЕ

**СТРОКА 890.** Для вывода на экран здесь также используется процедура для печати символьной строки, но первоначально в данных этой строки переписываются значения x1,y1,x2 и y2. Коды символов сообщения при этом записаны в 669AH...66A5H, перенос значений переменных x1,y1,x2 и y2 - в 66A6H..66B1H, а процедура печати - В 66B2H...66BFH.

#### Печать "героя".

```
669AH
DEFB 11 08 10 08 08 16 02 04
DEFB 15 00 04 96
```

```
66A6H
26278  LD HL, (5B06H)
26281  LD (669F), HL
26284  LD HL, (5B03H)
26287  LD (66A3H), HL
26290  LD A, 02
26292  CALL 1601H
26295  LD DE, 669AH
26298  LD BC, 000C
```



26301 CALL 203CH

**СТРОКА 900.** Эта строка в программе в машинных кодах не нужна, т.к. расположенные здесь данные для музыки будут загружены непосредственно в память при загрузке машинного кода.

**СТРОКИ 910...920.** Здесь через CALL 566AH вызывается процедура для проигрыша торжественной мелодии. Сама мелодия расположена в адресах 6656H...6669H, процедура - в 666AH...6699H, а вызов выполняется из точки 66C0H.

#### Музыка.

66C0H

26304 CALL 666AH

6656H

DEFB 00 00 00 00 00 00 ED 0B

DEFB ED 0B F0 10 EC 0B EC 10

DEFB EC 0B EC 10

666AH

26218 LD B, 07

26220 LD HL, 665CH

26223 LD A, (HL)

26224 LD(6678H), A

26227 INC HL

26226 PUSH BC

26229 PUSH HL

26230 RST 28H ; Калькулятор.

DEFB 34 EC 4C CC CC CC 38

26238 POP HL

26239 LD A, (HL)

26240 PUSH HL

26241 CALL 2D28H

26244 CALL 03F8H

26247 POP HL

26248 POP BC

26249 INC HL

26250 DJNZ -1DH

26252 LD A, 01

26254 CALL 3D28H

26257 LD A, 14H

26259 CALL 2D28H

26262 CALL 03F8H

26265 RET

**СТРОКА 930.** Изменение значения переменных home и score в машинных кодах см. 66C3H...66D7H, а вывод значения score на экран выполняется с использованием печати числа, находящегося на вершине стека калькулятора. Машинные коды 66D4H...66E9H.

#### Коррекция переменных.

66C3H

26307 LD A, (5B05H)

26310 INC A

26311 LD (5B05H), A

26314 LD HL, (5B03H)

26317 LD DE, 0032H

26320 ADD HL, DE

26321 LD (5B03H), HL

#### Печать новых значений.

66D4H

26324 PUSH HL

25325 LD A, 02

```

25327 CALL 1601H
26330 LD A, 16H
26332 RST 10H
26333 LD A, 15H
26335 RST 10H
26336 LD A, 07
26338 RST 10H
26339 POP BC
26340 CALL 2D2BH
26343 CALL 2DE3H

```

**СТРОКА 950.** В этой строке выполняется проверка: все ли 4 дома уже заняты. Если нет, то осуществляется переход к главному циклу. Программа в машинных кодах, выполняющая эту операцию, проще, чем строка BASIC-программы. Для этого достаточно записать в регистр A значение из ячейки 5B05H и с помощью команды AND 4 маскировать все биты, кроме второго. Если в результате операции флаг нуля (ZERO) не будет установлен, то выполняется переход к главному циклу с помощью команды JP 641E.

Проверка 4-х домов.

```

66EАН
26346 LD A, (5B05H)
26349 AND 04
26351 JP NZ, 641EH

```

**СТРОКИ 960...980.** Эти строки позволяют увеличить скорость игры, когда количество занятых домов достигает 4,8 или 12. Это достигается путем исключения команд RET из процедуры ROLL.

**СТРОКА 985.** В этой строке проверяется, выше ли число занятых домов, чем 36. Если это так, то пропускается процедура для добавления еще одного паука и осуществляется переход к главному циклу. Для реализации этого в машинных кодах достаточно вычесть от значения переменной "home" число 37. Если при этом флаг переноса (CARRY) не будет включен, то значит число занятых домов  $\geq 37$ , и выполняется переход по команде JP NC 6129. Машинный код 6714H...6718H.

Изменение скорости игры.

```

66F2H
26354 XOR A
26355 LD A, (5B05H)
26358 CP 04
26360 JR NZ, +06
26362 XOR A
26363 LD (7EA9H), A
26366 JR +19
26368 CP 08
26370 JR NZ, +06
26372 XOR A
26373 LD (7EC2H), A
26376 JR +0F
26378 CP 0C
26380 JR NZ, +06
26382 XOR A
26383 LD (7ED5H), A
26386 JR +5
26368 SUB 25H
26390 JP NC, 6129H

```

**СТРОКА 990.** Работу со случайными числами мы уже обсуждали в четвертой главе, поэтому объяснять порядок преобразования этой строки БЕЙСИК-программы в машинные коды мы здесь не будем. Необходимо только отметить, что для получения случайного числа в этой программе вначале извлекается значение системной переменной SEED, а затем оно

изменяется по следующей программе:

```
LD A, C
AND 1F
SUB 2
```

Все операции выполняются с использованием кодов калькулятора. Машинные коды - 6719H...6740H.

Генерация случайных чисел.

```
6718H
26393 LD BC, (5C76H)
26397 CALL 2D2BH
26400 RST 28; калькулятор.
DEFB A1 0F 34 37 16 04 34 80
DEFB 41 00 00 80 33 02 A1 03
DEFB 31 38
26419 CALL 2DA2H
26422 LD (5C76P), BC
26426 LD A, C
26427 AND 1F
26429 SUB 02
26431 JR C, +05
```

СТРОКА 1000. Все, что необходимо сделать, это увеличить на 1 значение переменной "а". Машинный код - 6741H.

СТРОКА 1005. В этой строке проверяется, не превышает ли значение переменной "а" величины 31. Если это так, то значение этой переменной обнуляется.

```
6741H
26433 INC A
26434 CP 20H
26436 JR NZ, +01
26438 XOR A
```

СТРОКА 1010. Для того, чтобы проверить свободна ли позиция экрана с координатами (10,а), используется процедура SCREEN\$. Для этого первоначально в регистр В записывается значение переменной "а", а в регистр С - значение 0AH (10D), и вызываются процедуры STACK VALUE IN BC и SCREEN\$, а затем параметры со стека передаются в регистры A,B,C,D,E. Теперь выполняется проверка, SCREEN\$(10,а) = 20H. Если это так, то выполняется переход для проверки следующей позиции экрана по команде JR 6741. Машинные коды, соответствующие этой строке, 6747H...675EH.

Поиск свободного знакомого.

```
6747H
26439 PUSH AF
26440 LD B, A
26441 LD C, 0A
26443 CALL 2538H
26446 CALL 2BF1H
26449 LD A, (DE)
26450 PUSH AF
26451 CALL 16BFH
26454 POP AF
26455 CP 20
26457 JR Z, +03
26459 POP AF
26460 JR -1D
26462 POP AF
```

СТРОКА 1020. Эта строка повторяет предыдущую, за исключением того, что значение переменной "а" увеличивается на 1 перед вызовом процедуры SCREEN\$. Обратите внимание, что значение переменной "а" сохраняется перед вызовом процедуры SCREEN\$

(PUSH AF), чтобы его можно было использовать в следующей процедуре. Машинные коды 675FH...6776H.

#### Поиск свободного знакоместа.

```
675FH
26463  INC A
26464  PUSH AF
26465  LD B,A
26466  LD C,0A
26468  CALL 2538H
26471  CALL 2BF1H
26474  LD A,(DE)
26475  PUSH AF
26476  CALL 16BFH
26479  POP AF
26480  CP 20
26483  JR Z,+03
26484  POP AF
26485  JR -36H
```

СТРОКА 1030. Значение переменной "а" извлекается со стека и записывается в регистр А, а затем размещается в участке памяти с именем DATA для следующего вывода на экран процедурой PRINT DATA. Машинные коды 6777H... 6792H.

#### Печать добавочного паука.

```
6777H
26487  POP AF
26468  DEC A
26489  LD (6782H),A
26492  JR +07
DEFB 11 04 16 0A 19 9E 9F
26501  LD A,02
26503  CALL 1601H
26506  LD DE,677EH
26509  LD BC,0007
26512  CALL 203CH
```

СТРОКА 1039. Здесь повторно вызывается процедура, воспроизводящая фанфары по команде CALL 666A.

СТРОКА 1040. Это строка безусловного перехода в начало цикла. В машинных кодах это выполняется по команде JP 6129H.

#### Музыка и возврат.

```
6793H
26515  CALL 666AH
26518  JP 6129H
```

Как видите, выполнять перевод БЕЙСИК-программы в машинные коды не так уж сложно. Лучше использовать при написании программ в машинных кодах шестнадцатиричную систему счисления, т.к. она более наглядна, если выполняется запись чисел сразу же в регистровую пару.

Нам осталось только привести машинный код для скроллинга экрана вправо-влево. Это код процедуры ROLL.

# Машинный код процедуры ROLL.

|       |               |               |    |       |              |
|-------|---------------|---------------|----|-------|--------------|
| ORG   | 32244         |               |    | 32357 | LD HL, 18496 |
| RIGHT | 32244         | LD C, 8       |    | 32360 | CALL RIGHT   |
| L1    | 32246         | PUSH HL       |    | 32363 | LD HL, 18496 |
|       | 32247         | LD DE, 31     |    | 32366 | CALL RIGHT   |
|       | 32250         | ADD HL, DE    |    | 32369 | JR L6        |
|       | 32251         | LD A, (HL)    | L5 | 32371 | LD HL, 18527 |
|       | 32252         | SBC HL, DE    |    | 32374 | CALL LEFT    |
|       | 32254         | RRA           |    | 32377 | LD HL, 18527 |
|       | 32255         | LD B, 32      |    | 32380 | CALL LEFT    |
| L2    | 32257         | LD A, (HL)    |    | 32383 | LD HL, 18527 |
|       | 32258         | RRA           |    | 32386 | CALL LEFT    |
|       | 32259         | LD (HL), A    | L6 | 32389 | LD HL, 18560 |
|       | 32260         | INC HL        |    | 32392 | CALL RIGHT   |
|       | 32261         | DJNZ L2       |    | 32395 | LD HL, 18560 |
|       | 32263         | POP HL        |    | 32398 | CALL RIGHT   |
|       | 32264         | INC H         |    | 32401 | LD HL, 18560 |
|       | 32265         | DEC C         |    | 32404 | CALL RIGHT   |
|       | 32266         | JR NZ, L1     |    | 32407 | LD HL, 18624 |
|       | 32268         | RET           |    | 32410 | CALL RIGHT   |
|       | 32281         | LD B, 32      |    | 32413 | LD HL, 20511 |
| L4    | 32283         | LD A, (HL)    |    | 32415 | CALL LEFT    |
|       | 32284         | RLA           |    | 32419 | LD HL, 20575 |
|       | 32285         | LD (HL), A    |    | 32422 | CALL LEFT    |
|       | 32286         | DEC HL        |    | 32425 | RET          |
|       | 32287         | DJNZ L4       |    | 32426 | LD HL, 18560 |
|       | 32289         | POP HL        |    | 32429 | CALL RIGHT   |
|       | 32290         | INC H         |    | 32432 | LD HL, 18624 |
|       | 32291         | DEC C         |    | 32435 | CALL RIGHT   |
|       | 32292         | JR NZ, L3     |    | 32438 | LD HL, 20511 |
|       | 32294         | RET           |    | 32441 | CALL LEFT    |
|       | 32295         | LD HL, 16479  |    | 32444 | LD HL, 20575 |
|       | 32298         | CALL LEFT     |    | 32447 | CALL LEFT    |
|       | 32301         | LD HL, 16512  |    | 32450 | RET          |
|       | 32304         | CALL RIGHT    |    | 32431 | LD HL, 16479 |
|       | 32307         | LD HL, 16512  |    | 32454 | CALL LEFT    |
|       | 32310         | CALL RIGHT    |    | 32457 | LD HL, 16512 |
|       | 32313         | LD HL, 16607  |    | 32450 | CALL RIGHT   |
|       | 32316         | CALL LEFT     |    | 32463 | LD HL, 18432 |
|       | 32319         | LD HL, 18432  |    | 32466 | CALL RIGHT   |
|       | 32322         | CALL RIGHT    |    | 32469 | RET          |
|       | 32325         | LD HL, 18432  |    | 32470 | LD HL, 16479 |
|       | 32328         | CALL RIGHT    |    | 32473 | CALL LEFT    |
|       | 32331         | LD HL, 18432  |    | 32475 | LD HL, 16607 |
|       | 32334         | CALL RIGHT    |    | 32479 | CALL LEFT    |
|       | 32337         | LD A, (23673) |    | 32482 | LD HL, 18560 |
|       | 32340...32346 | - NOP         |    | 32485 | CALL RIGHT   |
|       | 32347         | AND 02        |    | 32486 | LD HL, 18624 |
|       | 32349         | JR Z, L5      |    | 32491 | CALL RIGHT   |
|       | 32351         | LD HL, 18496  |    | 32494 | RET          |
|       | 32354         | CALL RIGHT    |    |       |              |

# ЧИТАТЕЛЬ - ЧИТАТЕЛЮ



Вместо вступления.

Дорогие друзья! Эту объемную статью не хочется предварять утомительным вступлением, но если Вас абсолютно не интересует музыка и, тем более, у Вас нет идей по работе с программой WHAM, мы все же просим Вас внимательно просмотреть статью. Дело в том, что здесь представлена такая гирлянда идей, полезных советов и маленьких хитростей, которая украсит многие программы, даже если они и не имеют отношения к музыке.

"ИНФОРКОМ".

(С) Алексеев А.Г., 1993г.

## WHAM! The Music Box. НОВЫЕ ВОЗМОЖНОСТИ

Этот музыкальный редактор наверняка уже знаком читателям "ZX-РЕВЮ". Он помогает не только освоить азы музыкальной грамотности, но и придать особый блеск Вашим собственным программам, тот завершающий штрих, который придает им неповторимый облик.

"WHAM" достаточно широко распространен на рынке программного обеспечения и читатель наверняка имеет ту или иную версию в своем арсенале. Существуют и подробные описания к программе. Более широко развить эту тему вынуждают некоторые тонкости, связанные с ее работой. В частности, неграмотная адаптация под БЕТА-ДИСК многих версий, имеющих хождение на рынке программного обеспечения. В результате этого они неправильно выполняют запись на диск скомпилированного кодового музыкального фрагмента. Предлагаемый материал вносит ясность в этот вопрос. Остановлюсь и на некоторых других особенностях работы этой программы, и главное - довольно существенном ее усовершенствовании.

Надо сказать, что чаще при изложении материала применяется следующий подход: какое-то явление или процедура и как ее можно использовать. Этот материал находится несколько в иной плоскости: конкретная программа и как к ней применить сведения из разных областей, так или иначе имеющие к ней отношение.

Итак, "WHAM". Вот типовой набор файлов, входящих в программу.

```
"WHAM!"      BASIC
"SCREEN$"    CODE 16384,6913
"WHAM!1"     CODE 65300,235
"WHAM!2"     CODE 30000,15000
"WHAM!3"     CODE 50000,7000
"M.BOX"      BASIC
```

Первый Бейсик-файл является загрузчиком. Во-первых, он переустанавливает значение RAMTOP=29999. Таким образом, для блоков кодов отводится место с адреса 30000. Во-вторых, именно в этом Бейсик-загрузчике находится команда, переключающая символьный набор на загружаемый утолщенный, стилизованный. Новое значение

системной переменной CHARS задается при помощи POKE:

POKE 23606,43: POKE 23607,217

CHARS=42+256\*217=55594

Переключение CHARS происходит в первой же строке Бейсик-загрузчика и является, кроме того, методом защиты. Это приводит к тому, что при нажатии BREAK ничего не видно: сам символьный набор еще не загружен. Это отпугивает начинающих. На самом деле все просто: переключитесь на символьный набор ПЗУ, для этого подайте "вслепую" прямую команду POKE 23606,0: POKE 33607,60 и все встанет на свое место.

Символьный набор расположен с адреса, на 256 большего, чем CHARS: 55594+256=55650. Если у Вас появятся желание заменить его или русифицировать всю программу, то Вы сможете это сделать, расположив новый символьный набор с адреса 55650. При русификации надо также учесть, что значительная доля текстовых сообщений содержится в машинно-кодовой части программы.

Большая часть программы выполняется в машинных кодах, но программа обращается и к Бейсику в определенных случаях. Это загрузка-выгрузка мелодии, компиляция и запись скомпилированного блока кодов. Одной из первоначальных проблем является то, как остановить программу и выйти в Бейсик. Для этого можно в титульном меню нажать "1" или "2" (загрузка или запись). Когда в нижней части экрана программа запросит ввод устройства - это уже работает Бейсик. Нажимайте BREAK для остановки программы. Запустить опять ее можно командой RUN.

Забегая вперед, скажу, что, когда появилась идея дополнить программу некоторыми новыми функциями, встал вопрос, как уместить все эти дополнения в рамках отведенной памяти. Чудес ведь не бывает. Для набивки новых строк необходимо отодвигать RAMTOP в сторону более старших адресов. Поэтому помимо обычных средств по экономии памяти (о которых будет рассказано ниже), пришлось пожертвовать частью буфера для хранения мелодий.

В первоначальном варианте "WHAM" может хранить в памяти 6 мелодий. Новый, усовершенствованный вариант - на одну меньше. Но такая жертва, по-моему, все же оправдана теми дополнительными возможностями, о которых будет рассказано ниже. Поэтому в новом загрузчике в строке 30 устанавливается значение RAMTOP, равное 31999 а не 29999, как в исходном варианте.

Изначальный текст Бейсик-файла "M.BOX" примерно на 80% был переработан. Во избежание неоднозначности, ниже приведен полностью его новый вариант: "m.box". Если Вы захотите воспроизвести программу у себя, внося изменения в старый файл "M.BOX", проверьте, пожалуйста, аккуратно все строки. Обратите внимание на их номера. Во многих случаях это существенно. Поэтому на начальном этапе без необходимости старайтесь ничего не изменять. Когда новая программа у Вас заработает, тогда можете вносить любые дальнейшие изменения.

Изменениям подверглись и кодовые блоки программы для поддержки ее новых возможностей. Подробно все изменения в кодовых блоках приведены в конце статьи. Условимся, что новые файлы, входящие в "WHAM" будут называться:

"WHAM+" BASIC

"wham \$" CODE

"Wham 1" CODE

"wham 2" CODE

"wham 3" CODE

"m.box" BASIC

Это нужно для того, чтобы различать новые и старые версии файлов.

Теперь некоторые вопросы, связанные с адаптацией под БЭТА-диск.

### **Универсальный загрузчик.**

Адаптация загрузчика под БЭТА-диск тривиальна. Заменяем все LOAD"" на RANDOMIZE USR 15619:REM LOAD "ИМЯ".

Но сейчас я хочу предложить прием, позволяющий реализовать универсальный загрузчик, одинаково работающий и с магнитофоном и с БЭТА-дискетом. Такой прием можно

использовать, в принципе, для любой программы, но для многих - это ненужная избыточность. Однако для таких программ, как "WHAM", которые после адаптации под диск остаются двухрежимными (загрузка и запись сохраняются для ленты и диска) эффективнее выглядит предлагаемый вариант. Он придает большую гибкость программе.

Универсальность основана на том факте, что при использовании БЭТА-диска системная переменная PROG (начало расположения Бейсик-программы) отличается от магнитофонного варианта компьютера на 112 байтов для дополнительных системных переменных. При проверке, если зафиксировано значение PROG=23755, то значит интерфейс Бета-диска отсутствует или неинициализирован. В этом случае программа выполняет загрузку с ленты, Если значение PROG больше, чем 23755, программа переходит на строки загрузки с диска.

Универсальный двухрежимный загрузчик для "WHAM" выглядит так.

```
10 REM "WHAM!" Universal Tape-Disk loader.
20 BORDER 0: PAPER 0: INK 0: POKE 23624,0: CLEAR 31999
25 IF (PEEK 23635+256*PEEK 23636)=23755 THEN POKE 23739,111:LOAD ""SCREEN$: LOAD ""CODE :
    LOAD ""CODE : LOAD ""CODE : GO SUB 100: LOAD ""
30 RANDOMIZE USR 15619: REM : LOAD "wham $"CODE 16384
40 RANDOMIZE USR 15619: REM : LOAD "wham 1"CODE
50 RANDOMIZE USR 15619: REM : LOAD "wham 22"CODE
60 RANDOMIZE USR 15619: REM : LOAD "Wham 3"CODE
70 GO SUB 100
80 RANDOMIZE USR 15619: REM : LOAD "m.box"
100 POKE 23675,88: POKE 23676,255: POKE 23606,42: POKE 23607,217
110 RETURN 9999 SAVE "WHAM+" LINE 10
```

Подпрограмма GO SUB 100 кроме переключения символьного набора выполняет также переключение символов UDG-графики, восстанавливая исходное значение, которое могло быть изменено дисковым загрузчиком "boot".

### Основной файл "m. box".

Как уже говорилось, новый вариант потребовал принятия мер по экономии памяти. Некоторые из них уже реализованы в файле "M.BOX". Это, например, задание вначале программы в 10 строке:

```
LET O=NOT PI: LET I=SGN PI: LET N7=VAL "7" ...
```

Буква О похожа на ноль, а буква I - на единицу, так что такая замена практически не сказывается на "читаемости" программы. Кроме того, при помощи переменной N7 задано число 7. Оно часто встречается в программе (например, BORDER N7: PAPER N7:). Замена часто встречающихся чисел переменными позволяет существенно сократить объем программы. Но для БЭТА-диска и всех нововведений этого все же недостаточно. Пришлось бороться буквально за каждый байт. Потребовалось заменить все числа на выражения VAL "число" во всей программе. Только теперь можно быть уверенным, что не произойдет сбоя программы из-за нехватки памяти при работе.

Листинг файла "m.box".

```
1 REM (C) 1985 MARK TIME LTD (C) 1993 ALANSOFT
2 GO TO VAL "10"
5 GO SUB VAL "90": STOP
6 RANDOMIZE USR VAL "15616"
7 STOP
10 CLS : LET O=NOT PI: LET I=SGN PI: LET N7=VAL "7": BORDER N7: RANDOMIZE USR VAL "50000":
    INPUT "": LET T$=""
15 LET STR=VAL "50000": LET LEN=VAL "2000"
20 LET KP=PEEK VAL "23559"-VAL"48"
25 IF KP<0 OR KP>N7 THEN RUN
30 PRINT AT VAL "5"+KP,VAL "9" ; OVER I; INVERSE I; "
35 POKE VAL "23658",VAL "6"
40 IF KP<INT PI THEN GO TO VAL "3000"
45 IF KP=VAL "4" THEN GO TO VAL "8000"
```



```

50 CLS : BORDER VAL "5": PRINT AT VAL "8",0;" 1 -Print HELP-PAGE."" 2 -Select MUSIC-KEY
   --->"" 3 -Routine RECOMPILER."" 4 -Tune TRANSPONER."
55 RANDOMIZE USR (PEEK VAL "51253"+VAL "56576"): PAUSE 0
60 IF INKEY$="2" THEN GO SUB VAL "600": GO TO VAL "55"
65 IF INKEY$="3" THEN GO TO VAL "9000"
70 IF INKEY$="4" THEN GO TO VAL "5000"
75 IF INKEY$="1" THEN CLS : PRINT : RANDOMIZE USR VAL "55420": PAUSE 0
80 RUN
90 IF PEEK VAL "23635"+VAL "256"*PEEK VAL "23636"=VAL "23755" THEN SAVE "m.box" LINE VAL
   "2": RETURN
92 RANDOMIZE USR VAL "15619": REM : ERASE "m.box"
94 RANDOMIZE USR VAL "15619": REM : SAVE "m.bOX" LINE 2
99 RETURN
100 IF T$="M" THEN GO SUB VAL "4000": IF KP=I THEN RANDOMIZE USR VAL "54000": GO TO VAL
   "140"
105 GO SUB VAL "6000": LET ERR=0
110 IF KP=VAL "2" THEN GO TO VAL "200"
115 IF T$="D" THEN LET ERR=USR VAL "15619": REM : LOAD F$CODE STR,LEN
117 IF ERR<>0 THEN GO TO VAL "300"
120 IF T$="T" THEN CLS : PRINT "START TAPE""SEARCHING FOR :";F$: IF F$<>" " THEN
   LOAD F$CODE STR
130 IF T$="T" AND F$=" " THEN PRINT "Loading any tune - no name given": LOAD ""CODE
   STR,LEN
140 IF PEEK STR>VAL "220" THEN POKE VAL "52860",PEEK STR
150 RUN
200 POKE STB,PEEK VAL "52860"
210 IF T$="D" THEN LET ERR=USR VAL "15619": REM : SAVE F$CODE STR,LEN
215 IF ERR<>0 THEN GO TO VAL "300"
220 IF T$="T" THEN SAVE F$CODE STR,LEN
225 IF T$="M" THEN RANDOMIZE USR VAL "54006": FOR A=I TO VAL "10": POKE VAL
   "65289"+A+(TN*VAL"10"),CODE F$(A): NEXT A: RUN
250 PRINT #0;"VERIFY ? (Y/N)": BEEP VAL ",1",VAL "20": GO SUB VAL "500": IF NOT F THEN RUN
260 LET ERR=0: PRINT AT VAL "19",0;" Verifying..."
270 IF T$="T" THEN VERIFY F$CODE STR,LEN
280 IF T$="D" THEN LET ERR=USR VAL "15619": REM : VERIFY F$CODE STR,LEN
290 IF ERR=0 THEN PRINT #0;TAB VAL "12"; INVERSE I;" O.K. ": BEEP VAL ".1",VAL "20": PAUSE
   VAL "50": RUN
300 PRINT #0; FLASH I;" ERROR ";ERR;" ": BEEP I,0: PAUSE 0: RUN
400 POKE VAL "23658",VAL "8": INPUT I: RETURN
500 POKE VAL "23658",VAL "8": PAUSE 0: INPUT ;: IF INKEY$ ="Y" THEN LET F=I: RETURN
510 LET F=0: RETURN
600 IF PEEK VAL "50908"=VAL "159" THEN POKE VAL "50908",VAL "191": POKE VAL "51253",VAL
   "178": RETURN
610 IF PEEK VAL "50908"=VAL "191" THEN POKE VAL "50908",VAL "159": POKE VAL "51253",VAL
   "170": PRINT AT VAL "13",VAL "27";" ";AT VAL "14",VAL "27";" ": RETURN
700 INPUT ;: PRINT #0; FLASH I;" Please, wait... ": RETURN
800 GO SUB VAL "400": PRINT #0;" Select device for "; "LOAD" AND KP=VAL "7"; "SAVE" AND
   KP=VAL "4";" ":" Press: T -TAPE; D -DISK"
805 PAUSE 0: INPUT ;: LET T$=INKEY$: IF T$<>"T" AND T$<>"D" THEN GO TO VAL "800"
810 RETURN
3000 PRINT #0;TAB VAL "4"; "SELECT PERIPHERAL DEVICE"
3010 PRINT #0""TAPE MEMORY DISK (SPACE = EXIT) "
3020 PAUSE 0: LET T$=INKEY$: IF INKEY$=" " THEN RUN
3030 IF T$<>"M" AND T$<>"T" AND T$<>"D" THEN GO TO VAL "3020"
3040 GO TO VAL "100"
4000 CLS : PRINT ""TUNES STORED IN MEMORY"
4010 LET D=VAL "65300": PRINT ""NO. TITLE:";
4020 FOR T=I TO VAL "5": PRINT '
4030 FOR D=D TO D+VAL "9": PRINT CHR$ PEEK D;: NEXT D: NEXT T
4040 PRINT #0;"TUNE NUMBER? L"
4050 LET N$=INKEY$: IF N$<"1" OR N$>"5" THEN GO TO VAL "4050"
4060 LET TN=VAL N$: POKE VAL "23681",TN+I: RETURN
5000 CLS : PRINT AT VAL "10", VAL "5"; "Tune Transponierung"
5010 GO SUB VAL "400": PRINT #0; "Press: U -for UP; D -for DOWN"
5020 PAUSE 0: IF INKEY$<>"U" AND INKEY$<>"D" THEN GO TO VAL "5020"

```

```

5030 LET R=(INKEY$="U")-(INKEY$="D")
5040 INPUT ;: PRINT #0; " 1 - 1 Channel 2 - 2 Channel 3 - 1 and 2 Channel"
5050 PAUSE 0: IF INKEY$<"1" OR INKEY$>"3" THEN GO TO VAL "5050"
5060 LET C=VAL INKEY$
5070 INPUT "Tune change (12=octave): ";S
5100 LET V=0: GO SUB VAL "700"
5110 IF C<>VAL "2" THEN LET A=VAL "60001": GO SUB VAL "5200"
5120 IF C<>I THEN LET A=VAL "61001": GO SUB VAL "5200"
5130 IF V=0 THEN LET V=I: GO TO VAL "5110"
5140 INPUT ;: PRINT AT VAL "10",VAL "5"; "Transponierung - O.K.":BEEP VAL ".1",VAL "20":
    PAUSE 0: RUN
5200 IF PEEK A=VAL "64" OR A=VAL "61000" OR A=VAL "62000" THEN RETURN
5210 IF PEEK A>VAL "40" AND PEEK A<VAL "244" THEN LET A=A+I: GO TO VAL "5200"
5220 LET T=S*R+PEEK A
5230 IF T>VAL "255" THEN LET T=T-VAL "256"
5240 IF T<0 THEN LET T=T+VAL "256"
5250 IF V=0 THEN IF T>VAL "40" AND T<VAL "244" THEN GO TO VAL "5300"
5260 IF V=I THEN POKE A,T
5270 LET A=A+I: GO TO VAL "5200"
5300 INPUT ;: PRINT AT VAL "15",0;" Tune ";("1" AND A<VAL "61000");("2" AND A>VAL
    "61000");" Channel is wery ";("high." AND R=I);("low." AND R=-I)
5310 PRINT "Transponierung is not available.": BEEP I,0: PAUSE 0: RUN
6000 DIM F$(VAL "10"): INPUT "Press ENTER for CATALOGUE DISK" AND T$="D" FILENAME: ";F$
6010 IF F$=" " AND T$="D" THEN GO TO VAL "6100"
6020 RETURN
6100 RANDOMIZE USR VAL "15619": REM : CAT
6110 GO TO VAL "6000"
7000 LET CHN1=VAL "57266": IF WHIND=VAL "3000" THEN LET CHN1=VAL "57464"
7010 LET CHN2=CHN1+LEN1+I
7020 LET HL=VAL "60001": LET DE=CHN1: LET BC=LEN1: GO SUB VAL "7500": LET HL=VAL "61001":
    LET DE=CHN2: LET BC=LEN2: GO SUB VAL "7500"
7030 LET TLENG=CHN2+LEN2-VAL "57000"
7031 LET CHN1=CHN1-VAL "57000"+ASEM
7032 LET CHN2 = CHN2-VAL "57000"+ASEM
7040 PRINT AT VAL "10",VAL "6";"CODE LENGTH : ";TLENG
7050 POKE VAL "57030",INT (CHN1/VAL "256"): POKE VAL "57029",CHN1-VAL "256"*PEEK VAL
    "57030"
7055 POKE VAL "57002",INT ((CHN1-I)/VAL "256"): POKE VAL "57001",CHN1-I-VAL "256"*PEEK VAL
    "57002"
7060 POKE VAL "57034",INT (CHN2/VAL "256"): POKE VAL "57033",CHN2-VAL "256"*PEEK VAL "57034"
7065 POKE VAL "57008", INT ((CHN2-I)/VAL "256"): POKE VAL "57007",CHN2-I-VAL "256"*PEEK VAL
    "57008"
7080 RETURN
7500 LET Z=VAL "256": LET H=INT(HL/Z): LET L=HL-H*Z: LET D=INT(DE/Z): LET E=DE-D*Z: LET
    B=INT(BC/Z): LET C=BC-B*Z: RESTORE VAL "7700": FOR Z=VAL "16416" TO VAL "16428": READ
    S: POKE Z,S: NEXT Z: RANDOMIZE USR VAL "16416":RETURN
7700 DATA VAL "33",L,H,VAL "17",E,D,I,C,B,VAL "237",VAL "176",VAL "201",PI
7701 PRINT #0;"RETURN OPTION 1,2 OR 3"
7702 PRINT AT VAL "12",0;"1,KEYPRESS""2,ALWAYS""3,TUNE END"
7703 IF INKEY$="1" THEN RETURN
7704 IF INKEY$="2" THEN PRINT AT VAL "4",VAL "20";"ALWAYS ": POKE VAL "57020",VAL "62":
    RETURN
7706 IF INKEY$="3" THEN POKE VAL "57063",VAL "225": POKE VAL "57064",VAL "225": POKE VAL
    "57065",VAL "251": POKE VAL "57066",VAL "201": PRINT AT VAL "4",VAL "20";"TUNE END":
    LET A$=" N/A ": PRINT AT VAL "7",VAL "20";A$:AT VAL "9",VAL "20";A$: RETURN
7707 GO TO VAL "7702"
8000 PAPER N7: INK I: BORDER VAL "5": CLS : PRINT INK N7;AT I,0;" "; INK I:
    RANDOMIZE USR VAL "55640"
8110 POKE VAL "23681",I: LET LEN1=USR VAL "54032"-VAL "60000"
8115 PRINT AT VAL "6",VAL "20";LEN1
8116 IF LEN1=VAL "999" THEN PRINT AT VAL "6",VAL "20";"NOT PRESENT"
8120 POKE VAL "23681",VAL "2": LET LEN2=USR VAL "54032"-VAL "61000"
8125 PRINT AT VAL "8",VAL "20";LEN2
8126 IF LEN2=VAL "999" THEN PRINT AT VAL "8",VAL "20";"NOT PRESENT"
8130 LET WHIND=USR VAL "54056"

```

```

8133 IF VHIND=VAL "3000" THEN PRINT AT VAL "5",VAL "20";"*PRESENT*"
8140 IF LEN1=VAL "999" OR LEN2=VAL "999" THEN GO TO VAL "8800"
8141 LET SM=LEN1-I: IF LEN2<LEN1 THEN LET SM=LEN2-I
8142 IF (LEN1-I)/SM<>INT ((LEN1-I)/SM) OR (LEN2-I)/SM<>INT ((LEN2-I)/SM) THEN GO TO VAL
    "8600"
8150 INPUT "ASSEMBLY ADDRESS: ";ASEM
8160 IF ASEM<VAL "16384" THEN GO TO VAL "8150"
8161 LET TLENG=LEN1+LEN2+VAL "267"+(VAL "198"*(WHIND=VAL "3000")): IF ASEM+TLENG>VAL "65536"
    THEN PRINT #0;"Error :-no room for tune.": BEEP I,0: LET ASEM=VAL "85536"-TLENG:
    PRINT AT VAL "21",0; "    MAX. ADDRESS: ";ASEM: PAUSE 0: GO TO VAL "8150"
8162 PRINT AT VAL "3",VAL "20";ASEM
8164 POKE VAL "55703",INT (ASEM/VAL "256");POKE VAL "55702",ASEM-VAL "256"*PEEK VAL "55703":
    RANDOMIZE USR VAL "55700"
8165 IF LEN1=LEN2 THEN GO SUB VAL "7700": INPUT "":PRINT AT VAL "12",0"": GO TO VAL "8170"
8166 PRINT #0;"CHANGE RETURN OPTION TO ALWAYS ?": GO SUB VAL "500": IF F THEN POKE VAL
    "57020",VAL "62": PRINT AT VAL "4",VAL "20";"ALWAYS "
8170 GO SUB VAL "700": GO SUB VAL "7000": INPUT :
8176 INK 0: PRINT AT VAL "12",0;"ROUTINE COMPILATION COMPLETED OK",,
8177 PRINT "ADJUSTMENT POKES :-",
8178 PRINT "REPLAY SPEED :";ASEM+VAL "35";", (230 TO 255)" "BORDER COLOUR:";JASEM+VAL
    "26";", (0 TO 7)"
8179 PRINT "TO RUN - RANDOMIZE USR ";ASEM
8180 POKE VAL "57035",PEEK VAL "52860"-VAL "4": POKE VAL "57026",PEEK VAL "52851"
8205 POKE VAL "56814",INT (ASEM/VAL "256"): POKE VAL "56813",ASEM-VAL "256"*PEEK VAL "56814"
8207 POKE VAL "56812",INT (TLENG/VAL "256"): POKE VAL "55811",TLENG-VAL "256"*PEEK VAL
    "56812"
8208 POKE VAL "56718",PEEK VAL "56811": POKE VAL "56719",PEEK VAL "56812"
8210 GO SUB VAL "6000": PRINT AT VAL "21",0,, : PRINT AT VAL "2",VAL "8"; INK I;"TUNE NAME :
    ";F$
8220 FOR A=I TO VAL "10": POKE VAL "56800"+A,CODE (F$(A)): NEXT A
8230 GO SUB VAL "800"
8240 IF T$="T" THEN PRINT #0;"Start tape, then press any key.":PAUSE 0: BORDER N7: INPUT ;:
    RANDOMIZE USR VAL "58700": GO TO VAL "8300"
8250 LET ERR=USR VAL "15619": REM : SAVE F$CODE 57000,TLENG
8260 IF ERR<>0 THEN GO TO VAL "300"
8270 RANDOMIZE USR VAL "56728"
8300 LET STR=VAL "57000": LET LEN=TLENG: GO TO VAL "250"
8600 PRINT AT VAL "12",VAL "8";"WARNING""End marker mismatch.""may cause distorted
    tune.""Continue ? (y/n)"
8610 GO SUB VAL "500": IF F THEN GO TO VAL "8145"
8630 RUN
8810 PRINT AT VAL "12",0;"Tune compilation abandoned."
8811 PRINT "No end markers defined."
8813 PRINT "Use the w key to place the end marker for the current channel."
8820 PAUSE 0: RUN
9000 CLS : PRINT AT VAL "10", VAL "6"; "Routine Recompiling": GO SUB VAL "800"
9010 IF T$="D" THEN GO SUB VAL "6090": LET ERR=USR VAL "15619":REM : LOAD F$ CODE 57000
9020 IF T$="D" THEN LET STR=PEEK VAL "23782"+VAL "256"*PEEK VAL "23783": IF ERR<> THEN GO TO
    VAL "300"
9030 IF T$="T" THEN LOAD ""CODE VAL "57000": LET STR=PEEK (PEEK 23649+256*PEEK
    23650+30)+256*PEEK (PEEK 23649+256*PEEK 23650+31)
9040 PRINT AT 0,0; PAPER VAL "7"; INK VAL "7"; "    ": GO SUB VAL "700"
9050 RESTORE VAL "9000": FOR A=VAL "16384" TO VAL "16397": READ B: POKE A,B: NEXT A:
    RANDOMIZE USR VAL "16384"
9060 DATA VAL "33",VAL "96",VAL "234",VAL "54",VAL "41",VAL "17",VAL "97",VAL "234",SGN
    PI,VAL "207",VAL "7",VAL "237",VAL "176",VAL "201"
9070 IF PEEK VAL "57035">VAL "250" THEN POKE VAL "60000",VAL "255"
9080 IF PEEK VAL "57035"<=250 THEN POKE VAL "60000",PEEK VAL "57035"+VAL "4"
9090 POKE VAL "52860",PEEK VAL "60000"
9100 LET A1=VAL "57000"-STR+(PEEK VAL "57001"+VAL "256"*PEEK VAL "57002")+I: LET A2=VAL
    "60001"
9110 GO SUB VAL "9200": LET M1=A2
9120 LET A1=A1+VAL "2": LET A2=VAL "61001"
9130 GO SUB VAL "9200": LET M2=A
9140 CLS : PRINT AT VAL "10",VAL "7": "Recompiling - O.K.": BEEP VAL ".1",VAL "20"

```

```

9150 GO SUB VAL "400": PRINT #0;"Delete markers""Tune end""? (Y/N)": GO SUB VAL "500"
9160 IF F THEN POKE M1,VAL "41": POKE M2,VAL "41"
9170 RUN
9200 POKE A2,PEEK A1
9210 IF PEEK A1<>VAL "64" THEN LET M=A1+I: LET A2=A2+I: GO TO VAL "9200"
9220 RETURN

```

После загрузки основного Бейсик файла "m.box", происходит старт программы с начальной строки. Несколько начальных строк выполняют вспомогательные функции. RUN5 5 - самозапись файла на ленту или диск (в зависимости от его наличия проверка в строке 90 по PROG выполняется аналогично тому, как это сделано в универсальном загрузчике). Переключение символьного набора - GO SUB 8. Обратное переключение его GO SUB 9.

Фактическое начало программы - строка 10. Титульная заставка прорисовывается при помощи подпрограммы в машинных кодах, расположенной по адресу 50000. Последняя устроена так, что при первом старте после загрузки сначала воспроизводится демонстрационная мелодия, прекращающаяся после нажатия на любую клавишу. Эта мелодия представляет собой скомпилированный фрагмент, расположенный и стартуемый с адреса 42500. При последующих запусках программы командой RUN, мелодия при прорисовке заставки не воспроизводится.

Когда подпрограмма в кодах RANDOMIZE USR 50000 возвращается в Бейсик, в ячейке 23559 сохраняется код нажатой клавиши. Строка 20 возвращает номер нажатой цифровой клавиши в переменную KP и дальнейшее распределение происходит в зависимости от ее значения. Так, строка 40 переводит на подпрограмму записи-считывания, строка 45 - на компиляцию. В том случае, если были нажаты клавиши "5" - "6", возврата в Бейсик вообще не происходит. А вот если нажата клавиша "7", то вместо "HELP PAGE" вызывается дополнительное системное меню: "SYST. MENU" - (соответствующее изменение текста титульного меню произведено в кодовом блоке). В "SYST.MENU" заложены все дополнительные возможности, но об этом чуть позже.

А сейчас более подробно об адаптации файла "m.box" под БЕТА-ДИСК. Изменения будут касаться процедур загрузки и выгрузки. Это пункты "1" и "2" в титульном меню программы. Эти процедуры имеют много общих строк, так что технология адаптации упрощается. В исходном варианте "WHAM" после нажатия клавишей "1" или "2" строка 40 переводит на блок строк с 3000, выполняющий задание устройства ввода-вывода. Внизу экрана появляется новое меню, предлагающее выбрать устройство:

```
TAPE, MEMORY, DRIVE (SPACE=EXIT)
```

Так как микродрайвы в нашей стране не слишком распространены, удобно было DRIVE заменить на DISK с изменением всех Бейсик-строк, относящихся к микродрайву. Заменена прежде всего строка, выводящая это меню - 3010.

Строка 3040 завершает выбор устройства и переводит на строку 100. Здесь выполняются дальнейшие действия, связанные с загрузкой-сохранением.

Строка 100 - выполнение подпрограммы записи-считывания мелодии при работе с буфером памяти: GO SUB 4000.

Подпрограмма GO SUB 6000 (строка 105) - это ввод имени файла. Удобно будет при выборе диска предложить пользователю вывести его каталог. Поэтому подпрограмма устроена так, что в случае дисководов, если не вводя имени нажать ENTER, будет выдан каталог диска. Такое решение удобнее, чем дополнительное меню, запрашивающее вывод каталога, так как освобождает оператора от лишних действий. Это, казалось бы, незначительная мелочь, но как раз из таких мелочей складывается впечатление о той или иной программе. Приятно иметь дело с программой, где все продумано до мелочей. Я не пытаюсь представить эту программу как эталон, усовершенствованию нет предела, но Вы можете в своих разработках использовать те моменты и приемы, которые уже хорошо зарекомендовали себя.

В строке 105 кроме ввода имени, задается нулевое начальное значение параметра ERR - кода ошибки TR-DOS. Далее, в строке 10 происходит разделение на процедуры

записи и загрузки. Если запись, то переход на строку 200. Если загрузка, то - на строку 115.

Как Вы знаете, в случае ошибки загрузки с магнитофона, программа остановится с сообщением "Tape loadind error". В случае ошибки при загрузке с диска, мы ничего не заметим. Поэтому необходимо предусмотреть специальные меры по перехвату ошибок TR-DOS. Для этого код ошибки возвращается в переменную ERR для контроля выполнения команды.

Строка 300 должна сигнализировать о невыполненной команде TR-DOS. Здесь применяется упрощенный прием, когда не расшифровывается подробно причина неудачи, а просто выводится код ошибки. Как правило, бывает достаточно даже просто одного сигнала об ошибке, чтобы вспомнить, что, например, файл с таким именем уже есть на дискете. Но лучше, все же, вывести код ошибки ERR. Это не отнимет много памяти, как подробные комментарии на все случаи жизни. Для дальнейшей оценки кода ошибки можно воспользоваться руководством к TR-DOS, но на практике, как правило, в этом не бывает необходимости.

После завершения записи программа предлагает выполнить верификацию. Если в ответ на запрос ввести любую клавишу, кроме "?", то от нее можно отказаться. Верификация реализуется в строках с 250.

Строки загрузки с магнитной ленты и записи на ленту остались без изменения.

Изменения, представленные выше, предназначены для загрузки и выгрузки на диск исходных текстов мелодий. Рассмотрим теперь проблемы, возникающие при компиляции мелодии и записи готового блока кодов, но сначала немного о компиляции. Программа устроена так, что можно скомпилировать музыкальный фрагмент в произвольно заданный адрес. Причем, наблюдательный пользователь наверняка заметил, что выгрузка кодового блока на магнитную ленту происходит необычно на слух: нет привычной паузы между заголовком и самим блоком кодов. Это наводит на мысль, что выгрузка выполняется нестандартной процедурой. Для чего это понадобилось? Выгрузка скомпилированного блока кодов происходит всё время из одного и того же места памяти: из адреса 57000. Хитрость заключается в том, что заголовок файла подменяется другим, в который заносится адрес, заданный Вами при компиляции, а сам массив кодов выгружаются всегда из 57000 с уже изменённой адресацией под заданный Вами адрес. Запись скомпилированного музыкального фрагмента выполняется в строке 8210 при помощи процедуры в кодах 56700. Вот эта программа.

```
56700: LD IX, 56800
. LD DE, 00017
. SUB A
. SCF
. CALL 1222
. NOP
. LD IX, 57000
. LD DE, Длина
. LD A, 255
. SCF
. CALL 1222
. EI
56727: RET
```

Как видим, запись производится в два этапа. Сначала записывается блок кодов, имитирующий заголовок кодового файла. Этот блок расположен с адреса 56800 и имеет стандартную для заголовка длину 17 байтов. При компиляции туда заносятся заданный стартовый адрес и получившаяся длина кодового блока. Затем выполняется вторая часть программы - запись самого блока кодов, как уже говорилось, из адреса 57000 (длина тоже заносится при компиляции).

Самый простой приём при адаптации под диск может быть следующий. Запись выполнить при помощи Бейсика, обычной командой TR-DOS, из адреса 57000. Это выполняют строки 8230 ... 8260.

Естественно, в этом случае в заголовках блоков, скомпилированных в разные адреса, будет всегда стартовый адрес 57000. Загружать записанный таким образом

скомпилированный блок надо, обязательно указывая адрес загрузки, тот который Вы задали при компиляции. Поэтому, если Вы завтра забудете адрес, для которого сегодня скомпилировали мелодию, то, скорее всего, Вы уже никогда не сумеете воспользоваться этим блоком кодов.

Поэтому, для устранения этого недостатка, процедура записи дополняется небольшой программой в машинных кодах, которая, будучи выполненной сразу же после записи файла, "подменяет" заголовок на диске на новый, с заданным Вами стартовым адресом.

Расположена эта программа следом за процедурой записи, с адреса 56728 (#DD96):

|       |        |      |             |
|-------|--------|------|-------------|
| #DD98 | 0E0A   | LD   | C, #0A      |
| #DD9A | CD133D | CALL | #3D13       |
| #DD9D | 79     | LD   | A, C        |
| #DD9E | 2AEDDD | LD   | HL, (#DDED) |
| #DDA1 | 22E65C | LD   | (#5CE6), HL |
| #DDA4 | 0E09   | LD   | C, #09      |
| #DDA6 | CD133D | CALL | #3D13       |
| #DDA9 | C9     | RET  |             |

Для того, чтобы подробнее разобраться, как она действует, откройте ZX-РЕВЮ № 1-2 за этот год на стр. 25 и вспомните, как выполняются программы TR-DOS из машинного кода: в регистр С заносится код команды и вызывается подпрограмма 15636 (3D13H).

Программа работает так. Вначале в регистр С заносится код 10, что соответствует процедуре поиска файла в каталоге диска. Параметры для поиска - спецификация файла, то есть его имя и тип. Эти данные уже заданы - они остались в системных ячейках после записи файла. После завершения поиска, результат будет в регистре С. Следующей будет выполнена процедура записи 16 байтов заголовка с изменённым параметром START на место старого. В регистре А задаётся номер заголовка (из регистра С), согласно требованиям к последующей команде при С=9 - записи заголовка файла. Теперь надо то значение адреса, которое было задано при компиляции, занести (в двухбайтном виде) в область спецификации файла в таблице системных переменных TR-DOS, а именно, в ячейки 23782, 23783 (5CE6H, 5CE7H). Для этого вспомним, что стартовый адрес уже занесён при компиляции в 17-байтный заголовок магнитофонного файла. Два его байта находятся по адресу 56813 (DDEDH). Поэтому берём его оттуда и при помощи регистра HL переписываем в системные переменные TR-DOS. Далее осталось только выполнять подпрограмму 15635 (3D13H) с параметром С=9. Теперь заголовок переписан с тем значением адреса, которое было задано при компиляции и проблем с адресом запуска больше не будет. Практически, при записи такой процедурой, процесс записи внешне никак не отличается от традиционного.

Рассмотренная процедура выполняется в строке 8270. Строка 8300 задаёт параметры: стартовый адрес и длину для верификации. Последняя осуществляется блоком строк, начиная с 250.

Надо сказать, что показанный приём может применяться и в других Ваших программах. Блок кодов, переписывающий заголовок, может использоваться, например, при желании наоборот, засекретить истинное значение стартового адреса или с иными целями. Сама записывающая процедура может работать в любом месте памяти.

### **Как "бороться" с COPY.**

От некоторых пользователей, в основном, начинающих, можно услышать претензии по поводу того, что после нажатия "7" в титульном меню, нормально выводится на экран страничка - подсказка ("HELP - PAGE"), но после этого "программа зависает". HELP - PAGE выводится при помощи блока в кодах, значит в нём причина "зависания"? Тогда устранить её будет сложно! Но на самом деле всё очень просто.

"Зависание" происходит на команде COPY (расположенной в старом варианте "WHAM" в строке 50 Бейсик - программы) на тех компьютерах, где имеется интерфейс для принтера LPRINT - 3 с собственным теневым ПЗУ. Оно активизируется, когда компьютер пытается выполнить команду COPY. Но он не может её выполнить, так как предварительно должен быть задан режим для получения графической копии экрана командами:

```
LPRINT CHR$ 0: PRINT CHR$ 1
```

```
LPRINT CHR$ 0: PRINT CHR$ 6
```

Выходов из этого положения два: сложный и простой. Сначала - сложный (но он пока не реализован в Бейсик - файле "m. box"). Для этого надо организовать команду, инициализирующую интерфейс принтера в одной из начальных строк, чтобы она выполнялась при старте программы. Причём команда эта по возможности должна сама определять, инициализирован интерфейс принтера или нет. Вот так это можно выполнить для LPRINT-3:

```
IF PEEK VAL "23296"<>VAL "95" THEN LPRINT: LPRINT CHR$ NOT PI; CHR$ VAL "5"  
GO TO VAL "10"
```

Контроль (упрощённый) того, что интерфейс инициализирован, выполняется по наличию в буфере принтера (с адреса 23296) драйвера печати, перебрасываемого туда из теневого ПЗУ при инициализации интерфейса LPRINT-3.

Но надо сказать, что начальная инициализация поможет правильно распечатать HELP - PAGE на принтере, но не поможет "бороться с COPY", если у Вас выключен или отсоединён принтер. Ну действительно, зачем всё время выполнять графическую копию экрана, может Вы хотите просто посмотреть страничку - подсказку и ничего больше. Можно, конечно, организовать запрос, например, "НАЖМИТЕ [P] ДЛЯ ПЕЧАТИ НА ПРИНТЕРЕ". Но гораздо изящнее всё это выглядит, когда выполняется автоматически. Помните о мелочах!

Попробуйте так:

```
75 IF INKEY$="1" THEN CLS: PRINT: RANDOMIZE USR VAL "55420"  
76 IF kp=N7 THEN BORDER N7: PAPER N7: INK 0: CLS: PRINT: RANDOMIZE USR VAL "55420"  
77 IF IN 123<>255 THEN COPY  
78 PAUSE 0: RUN
```

Команда RANDOMIZE USR 55420 в строке 75 (в старом файле "M. BOX" это происходит в строке 50) выводит на экране HELP - PAGE, после чего происходит контроль, включён ли принтер и готов ли он к печати. Если готовность принтера есть, тогда IN 123 даст значение 127 (выключится 7 бит) и будет выполнена распечатка графической копии экрана. Таким образом, если Ваш принтер выключен, то строка 77 никак не будет влиять на работу программы, но как только Вам понадобится "твёрдая копия", достаточно будет всего лишь включить принтер.

Но даже такой "мудрый" подход к команде COPY проблему решает только отчасти, так как невозможно предусмотреть заранее все многообразие интерфейсов принтеров, имеющих хождение в стране. Не у всех же LPRINT-3. Да и вряд ли имеет смысл тратить много сил и компьютерной памяти для решения проблемы в рамках этой программы. Ну в самом деле? Это же не текстовый редактор, где печать является определяющим параметром. Поэтому, предложу теперь второй выход - простой, но подходящий на все случаи жизни: вообще исключить команду COPY из программы. Она (программа) не сильно пострадает от этого. Кстати, COPY встречается в программе ещё в одном месте: в старом варианте "WHAM" это строка 8210. Там по замыслу авторов должны выдаваться на печать параметры скомпилированного музыкального фрагмента. COPY надо удалить и в этой строке, ограничившись выводом данных только на экран.

### Компиляция.

Теперь поговорим подробнее о самой компиляции. В принципе, при компиляции возможно задание любого адреса. Но определённые ограничения вносит строка 8160. В старом варианте "WHAM" она запрещает задавать адрес меньше 32768. но если Вы захотите скомпилировать мелодию, скажем, для размещения её в нулевой Бейсик-строке или в дисплейном файле, то Вы не сможете это сделать. Для чего авторами программы внесено такое ограничение, наверное, уже никогда не узнать. Но Вы можете устранить этот дефект простой заменой числа в строке 8160.

Кроме того, наверное ограничение накладывает строка 8161 старого "M. BOX". Если мелодия расположена близко к концу памяти (65535), то заданный адрес может оказаться

недоступен, хотя при полученной реальной длине Вы можете рассчитать, что скомпилированный блок кодов поместился бы до конца памяти. Для устранения этого надо также изменить числовые параметры в строке 8161 (см. листинг "m. box").

Теперь будет возможно разместить мелодию в любом месте ОЗУ, и только адреса ПЗУ для компиляции будут недоступны. Причём, если задать заведомо большой адрес, например 65500, то программа после вывода сообщения о недопустимой длине, сама рассчитает и предложит Вам такое предельное значение адреса, которое позволит Вам расположить скомпилированный блок вплотную к концу ОЗУ (это опять, казалось бы, мелочь, но как приятно работать с такой дружелюбной программой!).

Ещё один момент, связанный с компиляцией, на который хочется обратить внимание пользователя. В строке 7500 старого варианта "WHAM" есть такой фрагмент:

```
... FOR Z=VAL "23296" TO VAL "23308": READ S: POKE Z, S: NEXT Z: RANDOMIZE USR VAL "23296" ...
```

Здесь создаётся вспомогательный кодовый блок, необходимый для компиляции.

С одной стороны мы говорили о корректной работе интерфейса принтера, а с другой - в драйвер печати, находящийся в буфере принтера, вносятся изменения, разрушающие его... Если Вы решите убрать команду COPY из программы и отказаться от вывода на печать, то не произойдёт ничего плохого. Но если Вы всё же сохраните печать (как это было показано выше для интерфейса LPRINT-3), то приведённый фрагмент будет систематически нарушать процесс печати. Вспомним, что печать (COPY) используется при компиляции, как раз после того, как содержимое буфера принтера испорчено приведённым выше фрагментом.

Хотя в предлагаемом варианте "WHAM" команда COPY исключена, всё же зарезервирована возможность для дальнейшего усовершенствования, пока не реализованного - распечатки полного текста на принтере. Поэтому для вспомогательного кодового фрагмента найдено другое место. Так как его работа временная, то есть возможность расположить его в экранной памяти, прямо в дисплейном файле, на том месте, где нет никакой информации. А для того, чтобы не "портить" изображение, надо предварительно "замаскировать" его, напечатав на этом месте пробелы с одинаковым белым цветом INK и PAPER, включив эту печать, например, в строку 8000. Здесь печатается 13 пробелов для "маскировки" блока кодов, который будет создан на этом месте строкой 7500. В последней - соответствующим образом изменены значения адресов.

Теперь вспомогательный блок кодов создаётся прямо на экране, но для пользователя это происходит совершенно незаметно.

Другим недостатком процесса компиляции является такой момент. После завершения её, на экран выводятся комментарии по поводу ячеек, в которых можно задать темп мелодии и цвет бордюра. При запуске скомпилированного блока кодов Вы увидите, что и темп и цвет бордюра не совпадает с тем, что были заданы Вами в редакторе перед компиляцией. Поэтому придётся ещё неоднократно вносить изменения в кодовый блок, добиваясь требуемого темпа воспроизведения.

Этот недостаток можно легко устранить, если процесс компиляции дополнить специальной Бейсик - строкой. Параметры, темп и цвет бордюра хранятся в памяти:

темп - в ячейке 52860,

а цвет бордюра - в 52851

и достаточно всего лишь перенести их в скомпилированный блок кодов. Но при этом надо учесть один момент. Перед занесением параметра, определяющего темп, надо внести определённые коррективы из-за разницы во времени работы воспроизводящих процедур. Экспериментальным путём подобрана величина смещения - заносится значение на 4 меньше, чем в ячейке 52850 (строка 8080).

Теперь после завершения компиляции Вам не потребуется вносить изменения - блок сразу же готов к работе.



### **Кодировка мелодии.**

Исходный (нескомпилированный) текст мелодии, которая в данный момент редактируется, располагается с адреса 60000. Это - рабочая область. Общая длина массива - 2000 байтов. При сохранении мелодии в памяти этот массив перебрасывается в буфер, находящийся в старом "WHAM" с адреса 30000 длиной 12 Кб, а в новом - с 32000, длиной 10 Кб. Рабочая область с 60000 поделена на две равные части по 1Кб. С адреса 60000 расположены коды первого канала, с адреса 61000 - второго канала. Изначально область, где расположен текст мелодии, заполнена кодом 41. Этот код для текста мелодии означает отсутствие ноты или "пауза" - то есть то же, что "пробел" для обычного текста.

В ячейке 60000 находится код, определяющий скорость воспроизведения мелодии. Коды вводимых нот будут располагаться начиная с адреса 60001. Для второго канала - с 61001.

Для обозначения нот используются коды (в порядке повышения тона) с 244 по 255 и далее с 0 по 40. Самая низкая нота - "до" 1-ой октавы имеет код 244. Следующая за ней "до - диез" 1-ой октавы - код 245 и т. д. Код 255 соответствует ноте "си" 1-ой октавы. Следующая нота - "до" 2-ой октавы имеет код 0. А самая высокая нота - "ми" 5 октавы (клавиша SYMB. SH. в режиме 4 октавы) имеет код 40.

Коды с 42 по 239 для нот не используются, они соответствуют тем звуковым эффектам, которые можно реализовать в программе: имитируют звуки барабана, ударника, и т.д. Вставляя шумовые эффекты в мелодию, надо учитывать, что при этом в память заносятся два числа: в области памяти для 1 и для 2 каналов. Пара этих чисел определяет звучание эффекта. Этим объясняется их многообразие. Кстати, здесь кроется причина того, что поставив музыкальный эффект в каком-то месте, пользователь с удивлением обнаруживает, что удалить его он не может. Для нот удаление выполняется клавишей ENTER (при этом в память заносится код 41). Для звуковых эффектов это не проходит, так как эффект определяется не одним числом, как нота, а двумя. Поэтому для того, чтобы удалить эффект, надо нажать ENTER в одном канале, затем вернуться на 1 шаг назад (клавиша "0"), переключить канал клавишей "T" и еще раз нажать ENTER, только тогда звуковой эффект будет стерт.

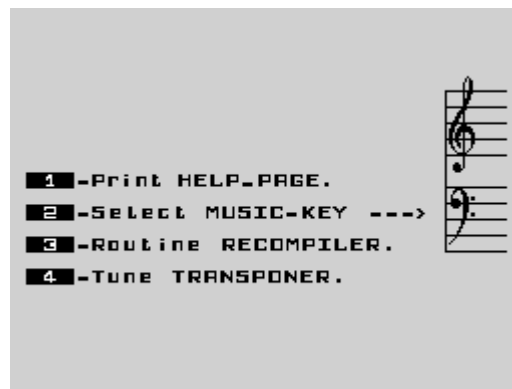
При завершении мелодии, перед тем, как ее скомпилировать, необходимо "зациклить" ее, установив в обоих каналах маркер "конец мелодии". Кстати, он может быть различным для каждого канала, например, 1-ый канал является солирующим и имеет максимальную длину, а 2-ой канал, сопровождая мелодию, все время повторяет одну и ту же короткую последовательность нот, с периодом в несколько тактов.

"Концу мелодии" соответствует код 64. Он ставится клавишей "W". Не все знают о том, что маркер "конец мелодии" легко можно удалить. Это выполняется так. Надо подойти к концу и остановиться на самом последнем такте мелодии, затем нажать ENTER. При этом в память будет занесена "пауза" на то место, где был маркер "конец мелодии". Код 64 просто заменится кодом 41. Это выполняется в каждом канале отдельно.

Теперь мы подошли к наиболее интересной части, о которой говорилось выше.

### **Дополнительные функции "WHAM".**

Они реализованы через дополнительное системное меню "SYST.MENU", вход в которое происходит теперь из титульного меню при нажатии клавиши "7" (в старом "WHAM" - это соответствовало "HELP PAGE"). При нажатии "7" программа попадает на строку 50. Вид системного меню представлен на рис 1.



При нажатии "1" выводится страничка-подсказка, аналогично тому, как это выполнялось в старом "WHAM". При нажатии "2" происходит смена нотного ключа на нижнем нотном стане. Этот момент требует комментария.

### Нотный стан.

Динамический диапазон "WHAM" охватывает четыре полных октавы с первой по четвертую (и частично захватывает пятую). Эти октавы относятся к скрипичным, басовые октавы в "WHAM" не используются.

Тем не менее, верхний нотный стан - верхние пять линеек - предназначен для обозначения нот в скрипичном ключе, а нижний - в басовом. Поэтому Вы можете заметить, что, например, нота "соль" 3-ей октавы, находящаяся на предпоследней линии верхнего нотного стана, не соответствует таким образом ноте "соль", для 1 октавы, находящейся на последней линейке на низшем нотном стане. Но учитывая то, что нижний нотный стан представлен в басовом ключе, все выглядит верно.

Это может оказаться неудобным если, например, надо набрать с нот мелодию, написанную так, что и верхний и нижний нотные станы находятся в скрипичном ключе. Поэтому добавлена новая функция - переключение нотного ключа на нижнем нотном стане. Теперь при переходе к редактированию мелодии (например, "6" в титульном меню), на экран выводится изображение нотных ключей. Нижний нотный стан формируется теперь тоже в соответствии с нотным ключом: в зависимости от последнего, с разницей в одну линейку. Это поясняет рисунок 2.



Переключение выполняется при помощи подпрограммы GO SUB 600. Формирование изображения нотных ключей производится подпрограммой в машинных кодах, выполняемой в строке 55.

В соответствии с типом нижнего нотного ключа, вход в подпрограмму печати ключей должен происходить с разных точек входа: для басового ключа - 56746 (DDAAH), для скрипичного - 56754 (DDB2H).

Для изображения ключей применен специальный символьный набор, а точнее, 23 дополнительных символа, соответствующие кодам с 33 по 55 включительно (символы с "!" по "7"). Их образы расположены на свободном месте с адреса 56816 по 56999 включительно. Изображение скрипичного ключа, например, складывается из 6 строк по 2 знакоместа. "Текст", изображающий нотные ключи, состоит из трех фрагментов, расположенных по адресам 65184 (FEA0H), 65218 (FEC2H) и 65264 (FEF0H), то есть по адрес 65299 и таким образом в новом варианте включен в файл "Wham 1"CODE.

Подпрограмма печати нотных ключей переключает CHARS на этот новый символьный набор, производит печать нотных ключей, выводя на экран строку текста, "вырисовывающую" нотный ключ, а затем возвращает прежнее значение CHARS.

Эта процедура расположена с адреса 56746 (DDAАН), следом за приведенной выше процедурой записи на диск заголовка файла.

```
DDAA 11A0FE LD DE, #FEA0
DDAD 012200 LD BC, #0022
DDB0 1806 JR #DDB8
DDB2 11C2FE LD DE, $FEC2
DDB5 012E00 LD BC, #002E
DDB8 2A365C LD HL, (#5C36)
DDBB E5 PUSH HL
DDBC 21E8DC LD HL, #DCE8
DDBF 22365C LD (#5C36), HL
DDC2 CD3C20 CALL #203C
DDC5 11F0FE LD DE, #FEF0
DDC8 012400 LD BC, #0024
DDCB CD3C20 CALL #203C
DDCE E1 POP HL
DDCF 22365C LD (#5C36), HL
DDD2 C9 RET
```

Подпрограмма на Бейсике GO SUB 600 в зависимости от типа ключа вносит изменения в формирователь нижнего нотного стана: для скрипичного ключа нотные линейки должны быть сдвинуты на 1 линию вниз по сравнению с басовым. Кодовый блок программы корректируется при помощи POKE.

### Рекомпиляции.

Это очень мощная новая возможность программы "WHAM". Но сначала некоторые дополнительные сведения о компиляции.

При компиляции мелодии происходит следующее. Сначала подготавливается воспроизводящая процедура. Надо сказать, что ее длина непостоянна и зависит от исходного текста мелодии, наличия в ней шумовых эффектов. Звучание нот в обоих каналах определяется "нотной" процедурой, а шумового эффекта - "шумовой" процедурой. Если шумовые эффекты отсутствуют, длина воспроизводящей только "нотной" процедуры будет равна 267 байт. Если в мелодии используется хоть один шумовой эффект, то длина воспроизводящего блока увеличивается на 198 байт - столько занимает "шумовая" процедура.

Затем к воспроизводящей процедуре пристыковывается нотная последовательность - берется фактическая длина исходного текста мелодии, отмеченная в рабочей области маркером "конец мелодии" для каждого голоса, формируется скомпилированный блок, как Вы уже знаете, с адреса 57000. В полученном блоке кодов изменяется адресация машиннокодовых команд переходов с абсолютной адресацией. Так выполняется "привязка" к тому адресу, который Вы задали при компиляции. Получается фрагмент кодов, состоящий из двух частей: "воспроизводящая процедура" и "нотная последовательность". Первая же команда воспроизводящей процедуры имеет вид:

```
LD HL, ADDR
```

где ADDR - адрес, с которого начинается в памяти нотная последовательность.

Исходя из этих данных, сделана подпрограмма, которая выполняет операцию, обратную компиляции, то есть выделяет из скомпилированного блока кодов исходный текст мелодии. Это очень часто нужно на практике. Например, Вы озвучили свою программу мелодией собственного сочинения, А потом захотели внести какие-то изменения в эту мелодию. Если Вы не сохранили исходный текст мелодии, то для выполнения этой задачи Вам пришлось бы набирать всю мелодию заново. Наличие подпрограммы рекомпиляции позволит Вам не сохранять все исходные тексты мелодий, а в любой момент получить из скомпилированного блока кодов полноценный исходный текст, доступный для дальнейшего

редактирования.

Вызов подпрограммы рекомпиляции происходит при нажатии "3" в системном меню. При этом программа переходит на строку 9000. После выбора устройства (магнитофон или дисковод) происходит загрузка скомпилированного блока кодов. Она выполняется в адрес 57000, то есть туда, где как раз готовился к выгрузке скомпилированный блок.

Теперь необходимо определить стартовый адрес STR, для которого блок был скомпилирован. В случае диска, эти данные можно взять из системных переменных TR-DOS, где они остались после загрузки (строка 9020). В случае загрузки с магнитофона, стартовый адрес можно прочесть в заголовке загружаемого файла. Он (заголовок) считывается во временную рабочую область памяти. Сразу же после завершения загрузки, пока временная рабочая область еще не уничтожена, его можно прочесть при помощи РЕЕК. Это выполняет строка 9030. В этой строке не используется прием по сокращению памяти типа VAL "число", так как расчет значения числа приводит к изменениям во временной рабочей области и преждевременному разрушению информации о считанном заголовке.

Строки 9040... 9050 при помощи вспомогательного блока кодов, формируемого в дисплейном файле, производят очистку области памяти с 60000 длиной 2000 - где располагается текущая мелодия. Сюда будет выполняться рекомпиляция. Строки 9070-9090 выполняют восстановление темпа воспроизведения. Правда, надо опять, как при компиляции, учитывать разницу во время работы машиннокодовых процедур, воспроизводящих мелодию.

Строка 9100 рассчитывает адрес, с которого располагается нотная последовательность. Коды, расположенные с этого адреса, надо переписать в рабочую область с адреса 60001. Это выполняется при помощи подпрограммы GO SUB 9200. Если встречается код 64, соответствующий маркеру "конец мелодии", подпрограмма 9200 заканчивает работу. Адрес расположения маркера "конец мелодии" запоминается в переменной M1 (строка 9110). Строки 9120-9130 выполняют аналогичные действия для второго канала. Адрес маркера конца - в переменной M2.

После звукового сигнала, свидетельствующего о завершении рекомпиляции (строка 9140), в строке 9150 следует запрос: надо ли уничтожить маркеры "конец мелодии". Это удобно, если Вы собираетесь продолжить мелодию, сделать ее длиннее, чем она была. В этом случае нажмите клавишу "Y", иначе - любую другую (если Вам надо всего лишь скорректировать некоторые ноты, то маркеры удалять незачем).

На этом подпрограмма рекомпиляции заканчивает свою работу и происходит возврат к титульному меню. Теперь восстановленный исходный текст мелодии находится в рабочей области. Его можно обрабатывать дальше, как обычно: редактировать, прослушать, сохранить и т.д.

### **Транспонирование мелодии.**

В эту подпрограмму можно попасть при нажатии "4" в системном меню. Это очень удобный режим, если надо изменить тональность или перезадать мелодию для другой октавы.

Программа транспонирования расположена со строки 5000. Прежде всего необходимо ответить на запрос, в какую сторону требуется изменить тональность: вниз или вверх. Это реализуется в строках 5010...5030.

Следующий запрос (строки 5040-5060) - для какого канала выполнить транспонирование: 1 - только для первого; 2 - только для второго; 3 - для обоих каналов.

В строке 5070 идет последний запрос: на какую величину произвести смещение. Минимальная величина - 1 тон. Если надо изменить тональность на 1 октаву, то величина смещения должна быть равна 12.

Работа начинается в строке 5100 и выполняется в два этапа: вначале - проверка, затем - собственно замена кодов. В строке 5100 обнуляется флаг режима - переменная V. Если V=0, то проверка, если V=1, то замена кодов. При V=0 происходит предварительный просмотр всего текста мелодии от начала до конца - возможно ли вообще осуществить транспонирование - не выйдет ли результирующая мелодия за пределы динамического

диапазона "WHAM". Это выполняется подпрограммой GO SUB 5200 для двух каналов - соответственно в областях памяти с адреса 60001 и с 61001 в зависимости от заданных каналов (строки 5110-5180). Если в результате проверки выясняется, что должен быть получен тон, недоступный для "WHAM", слишком высокий или слишком низкий, то процесс проверки прерывается с выдачей сообщения о невозможности транспонирования. Это выполняется переходом на строку 5300. После чего - возврат к титульному меню. А если проверка доведена до конца, то в строке 5130 устанавливается флаг V=1, что означает "транспонирование возможно". Можно переходить к замене кодов. Это выполняется путем возврата на строку 5110. Здесь процесс вызова подпрограммы GO SUB 5200 повторяется вторым циклом, но уже со значением флага V=1. В этом случае строка 5260 производит замену кодов. Выход из подпрограммы 5200 по достижении маркера "конец мелодии" или, при его отсутствии, по достижении последнего адреса рабочей области.

Строка 5140 подводит итог транспонирования, после чего - возврат в титульное меню. Дальше - обычная работа с полученной мелодией.

Надо сказать, что работа подпрограмм рекомпиляции и транспонирования, организованная в Бейсике, не отличается быстродействием, так что, выполняя их, наберитесь терпения. "Неторопливость" подпрограмм усугубляется внесением элементов экономии памяти: VAL "число" и заменой чисел переменными. Но учитывая, что этими возможностями не каждую минуту придется пользоваться, с этим можно смириться. Лучше было бы, конечно, построить такие процедуры в машинных кодах, тем более, что это не так уж сложно и вполне доступно начинающим. Попробуйте, если хотите, заменить Бейсик машинным кодом. Я же - ограничусь идеей.

### **Сокращение программы.**

Как известно, после загрузки программы мы имеем в памяти шесть демонстрационных мелодий. Кроме того, непосредственно в памяти может находиться седьмая. Это удобно, когда Вы в первый раз загружаете программу - можно послушать их, посмотреть, как задаются ноты и эффекты. Но когда Вы освоитесь и будете заниматься самостоятельным набором мелодий для оформления своих собственных программ, демонстрационные мелодии не понадобятся. Вы все равно будете удалять их, записывая в память на их место свои заготовки. Каждая мелодия занимает в памяти по 2 Кб. Исключив их из загрузки, можно сократить программу на 12 Кб. А если еще "выбросить" цветную заставку (около 7 Кб) то получится ощутимая экономия.

Владельцы магнитофонных версий, представьте, во что выльется сокращение времени при каждой загрузке! Здесь речь идет не об урезании программы вообще, а о "персональной копии" - специальной версии, предназначенной для быстрой загрузки. И не об удалении демонстрационных мелодий вообще, их надо сохранить, но отдельно, на ленте (или дискете), чтобы всегда можно было при желании загрузить опять. Для этого надо каждую мелодию сначала загрузить из буферной памяти в рабочую область редактора, а затем сохранить на ленте (или диске). После того, как демонстрационные мелодии надежно сохранены, можно приступить к модификации кодового блока "WHAM!2" CODE. Начиная с адреса 30000, длиной по 2 Кб, здесь расположено 6 мелодий. Итого - по адрес 41999 включительно. Можно сделать так:

```
LOAD "WHAM!2"CODE 30000,15000
SAVE "wham 2"CODE 42000,3000
```

Осталось только дополнить загрузчик небольшой подпрограммой, которая будет инициализировать область с 32000 (в старом варианте это надо было делать с 30000), заполняя ее кодом 41 - "пауза". Вот эта подпрограммка:

```
LD    HL, 32000
LD    (HL), 041
LD    DE, 32001
LD    BC, 09999
LDIR
RET
```

Для ее формирования и запуска универсальный Бейсик-загрузчик "WHAM", приведенный в начале статьи, надо дополнить следующими строками:

```

102 FOR a=20480 TO 20493: READ b: POKE a,b: NEXT a
104 RANDOMIZE USR 20480
106 DATA 33,0,125,54,41,17,1,125,1,15,39,237,176,201

```

Кроме того, надо очистить (а точнее, заполнить пробелами или, например, словом: "EMPTY") ту область, где хранятся названия демонстрационных мелодий. Это область с адреса 65300, по 10 байтов на каждое название. Эти исправления надо сделать в файле "WHAM!3"CODE.

Для устранения загрузки картинки, из загрузчика можно удалять строку 30 и из строки 25 исключить:

```
... LOAD ""SCREEN $: ...
```

и перекомпоновать заново файлы для магнитофонного варианта.

Теперь, когда описание программы и всех ее новых функций завершено, переходим к тем изменениям, которые надо выполнить в кодовых блоках.

### Изменение кодовых блоков.

Ниже приводится программа на Бейсике, выполняющая все необходимые изменения в кодовых блоках "WHAM!1"CODE и "WHAM!2"CODE. По завершении она выдает новые файлы "wham 1"CODE и "wham 3"CODE.

Программа предназначена для работы с магнитофоном. Владельцы "Спектрумов" с БЭТА-диском, наверняка сами сумеют внести необходимые коррективы.

```

10 CLEAR 31999: POKE 23675,88: POKE 23676,255
20 LOAD "WHAM!1"CODE
30 LOAD "WHAM!3"CODE
40 POKE 50908,159: POKE 50909,79: POKE 50921,223: POKE 50922,71
50 POKE 51248,0: POKE 51253,170: POKE 51254,221
60 FOR A=51374 TO 51406: READ B: POKE A,B: NEXT A
70 DATA 022,017,001,127,049,057,056,053,032,077,065,082,075,032,084,073,077,069,043,127,049,
    057,057,051,032,065,076,065,078,083,079,070,064
80 FOR A=51495 TO 51527: READ B: POKE A,B: NEXT A
90 DATA 083,089,083,084,046,077,069,078,085,019,000,022,015,007,032,080,082,069,083,083,032,
    067,079,078,084,082,079,076,032,075,069,089,032
100 FOR A=56728 TO 56786: READ B: POKE A,B: NEXT A
110 DATA 014,010,205,019,061,121,042,237,221,034,230,092,014,009,205,019,061,201,017,160,
    254,001,034,000,024,006,017,194,254,001,046,000
120 DATA 042,054,092,229,033,232,220,034,054,092,205,060,032,017,240,254,001,036,000,205,060,
    ,032,225,034,054,092,201
130 FOR A=56816 TO 56999: READ B: POKE A,B: NEXT A
140 DATA 000,000,000,000,000,000,000,255,016,040,044,076,076,076,076,255,128,128,128,128,
    128,128,128,255,092,088,088,120,112,112,096,255
150 DATA 128,129,131,131,135,134,142,255,224,192,192,064,064,064,064,255,156,153,187,187,
    183,183,182,255,224,248,252,252,078,070,070,255
160 DATA 182,179,147,153,136,132,131,255,070,070,068,068,072,080,224,255,128,128,128,134,
    143,143,142,135,064,054,064,064,064,064,128,000
170 DATA 000,000,000,006,015,015,014,007,128,128,128,128,128,128,128,135,140,136,144,
    144,158,159,255,128,192,224,096,118,118,112,255
180 DATA 159,142,128,128,128,128,128,255,112,112,112,112,102,102,096,255,128,128,128,129,
    129,131,130,255,192,192,128,128,000,000,000,255
190 DATA 132,136,144,160,128,128,128,255,000,060,064,124,006,056,060,000,000,126,002,004,
    008,016,016,000
200 FOR A=USR "A" TO USR "B"+7: READ B: POKE A,B: NEXT A
210 DATA 008,120,248,248,248,112,000,000,000,000,000,009,014,008,008,008
220 FOR A=USR "E" TO USR "F"+7: READ B: POKE A,B: NEXT A
230 DATA 248,112,000,000,000,000,000,000,009,014,008,008,008,120,248,248
240 FOR A=65184 TO 65299: READ B: POKE A,B: NEXT A
250 DATA 016,000,017,007,022,008,027,035,033,033,022,009,027,047,048,033,022,010,027,049,
    050,033,022,011,027,051,052,033,022,012,027,053
260 DATA 033,033,016,000,017,007,022,008,027,046,128,128,022,009,027,035,034,033,022,010,
    027,035,036,033,022,011,027,037,038,033,022,012

```

```

270 DATA 027,039,040,033,022,013,027,041,042,033,022,014,027,045,044,128,022,002,027,033,
    034,033,022,003,027,035,036,033,022,004,027,037
280 DATA 038,033,022,005,027,039,040,033,022,006,027,041,042,033,022,007,027,043,044,128
290 FOR N=1 TO 6: RESTORE 300: FOR M=1 TO 10: READ B: POKE (65300+10*(N-1)+M-1),B: NEXT M:
    NEXT N
300 DATA 045,045,069,077,080,084,089,045,045,032
400 SAVE"wham 1"CODE 65184,352
410 SAVE"wham 3"CODE 50000,7000

```

Цифры в строках DATA напечатаны одно под другим только для упрощения "читаемости" программы. Вы можете набирать их так, как Вам удобно (например, вместо "002" разумеется, можно набрать "2").

На этом мы заканчиваем изложение некоторых вопросов, связанных с музыкальным редактором "WHAM", но ни в коем случае не заканчиваем музыкальную тематику. Наоборот, планируем всячески ее развивать в дальнейшем. В первом номере "ZX-РЕВЮ" будущего года мы продолжим рассказ о музыкальных редакторах. Будет рассмотрен другой вариант редактора "WHAM" - рассчитанный на работу с трехканальным музыкальным сопроцессором AY-8910 (8912). А затем, в дальнейшем, программы "SOUND-TRACER" и "SONGCOMPILER", также рассчитанные на музыкальный сопроцессор. С их помощью можно создавать звуковое сопровождение такого высокого качества и широкой палитры звучания, что оно становится практически соизмеримо с музыкой, которую мы слышим по радио.

# ВОЗВРАЩАЯСЬ К НАПЕЧАТАННОМУ

(С) Комиссаров П., 1993г.

(С) "ИНФОРКОМ", 1993г.

## НЕСТАНДАРТНАЯ ЗАГРУЗКА

В "ZX-РЕВЮ" №1-2 за этот год на стр. 31 мы опубликовали процедуру загрузки с изменением цвета бордюрных полос, взятую из игры "BARBARIAN-3", присланную Комиссаровым Павлом из пос. Видово Мурманской обл. В этот раз мы продолжаем рассказ об этой процедуре, о иной нестандартной загрузке, а также связанных с этим некоторых приёмах защиты, пользуясь дополнительным материалом, любезно предоставленным нам нашими корреспондентом.

Павел прислал дополнения к загрузчику, которые позволяют изменить его свойства.

Правда, следует заметить, что, тестируя присланные им листинги, мы получали иногда несколько иные результаты, чем те, о которых пишет Павел. Поэтому публикуемый материал является коллективным творчеством: в основе - то, что прислал наш корреспондент, но с нашими изменениями и дополнениями. Итак, модернизация загрузчика блоков кодов без заголовка.

1. Загрузка кодов, начиная со старших адресов к младшим. Для этого надо сделать такую замену: FE55H - 2B. Это приведёт к тому, что по адресу FE54H вместо команды INC IX получится команда DEC IX. Здесь надо отметить два момента. Во-первых, в вызывающей программе в регистре IX указывается не начальный, а конечный адрес загрузки (начальный адрес плюс длина DE), так как загрузка идёт "наоборот". Другая особенность заключается в том, что, задав, скажем, для загрузки экрана, длину 6912 байтов. Вы обнаружите, что последний байт блока кодов (он будет первым байтом дисплейного файла) не грузиться. Поэтому, при подготовке блока к загрузке, следует задать длину на 1 байт большую, то есть 6913 байтов. Для загрузки экранного файла, например, вызывающая программа, может выглядеть так.

```
LD    IX,#5B00
LD    DE,#1B01
LD    A,#FF
SCF
CALL  #FDE8
RET
```

При этом загружаемую "картинку" необходимо сначала "перевернуть" и увеличить её длину на 1 байт, например такой программой на Бейсике:

```
CLS: LOAD "" SCREEN$
FOR a=0 TO 6912: POKE (46912-a), PEEK(16384+a): NEXT a
SAVE "name" CODE 40000, 6913
```

При выполнении записи на ленту в строке 3 пропустите заголовок, а только затем включайте магнитофон, то есть запишите только сам кодовый блок.

При загрузке со старших адресов к младшим, будет слышно, как вначале грузятся атрибуты, а затем - графика.

К сказанному можно добавить такой факт. В своё время нам попалась игра "ROLLER COASTER" (ELITE), в которой загрузка велась тоже "наоборот". Причём запись была отвратительного качества и едва читалась. Эта программа была переписана с магнитофона на магнитофон, так как ни один копировщик не мог её скопировать из-за большой длины. Даже копировщики с компрессированием памяти (типа "TF-COPY"). При более детальном



рассмотрении выяснилось, что загрузка идёт там тоже со старших адресов в младшие, причём начинается с экрана (с атрибутов), затем пробегает область всего ПЗУ, затем, после достижения нулевых значений в счётчике адреса и его переполнения, продолжается с конца памяти и заканчивается буфером принтера. То есть, загружается (точнее говоря, делают вид, что загружаются) все 64К памяти! При этом переписывается и машинный стек, обеспечивая после окончания загрузки и возврата по RET автостарт программы. Сам факт существования ещё "не взломанной" программы уже говорит о высоком качестве защиты.

Однако, такой метод защиты имеет смысл, если Ваша программа большая - использована практически вся память. Иначе сколько же лишнего времени будет занимать загрузка! Это очень "действует на нервы" и резко снижает симпатию к программе. Это тоже надо учитывать.

Однако, возвращаемся к загрузчику от BARBARIANA.

2. При загрузке пилоттоны хаотично изменяют свой цвет, переливаясь всеми цветами радуги. Для получения этого эффекта в загрузчике от BARBARIANA нужно изменить всего одну ячейку: FE90H - 00. Можете также попробовать другой вариант: FE8FH - ED, FE90H - 5F.

3. При загрузке по бордюру проскакивают тонкие короткие разноцветные штрихи. Этот эффект получается, если изменить загрузчик следующим образом:

|      |           |     |          |
|------|-----------|-----|----------|
| FE8F | FD5F      | LD  | A, R     |
| FE93 | D3FE      | OUT | (#FE), A |
| FE95 | E600      | AND | #00      |
|      | FE97 F608 | OR  | #06      |
| FE99 | D3FE      | OUT | (#FE), A |
| FE9B | 37        | SCF |          |
| FE9C | C9        | RET |          |

4. Бордюр чёрный, индикация считывания (бордюрные полосы) производится в заданных позициях (знакоместах) экрана. Надо внести следующие изменения:

|      |        |     |            |
|------|--------|-----|------------|
| FE32 | EE00   | XOR | #00        |
| FE79 | 3E0B   | LD  | A, #0B     |
| FE8F | 3E00   | LD  | A, #00     |
| FE91 | 2F     | CPL |            |
| FE92 | 3290FE | LD  | (#FE90), A |
| FE95 | E606   | AND | #06        |
| FE97 | EE04   | XOR | #04        |
| FE99 | F642   | OR  | #42        |
| FE9B | ...    |     |            |

Дальнейшая часть программы зависит от того, в каком месте экрана Вы хотите осуществить вывод бордюрных полос. Например, пусть это будут два знакоместа экрана: левое верхнее и соседнее с ним справа. Адреса ячеек в файле атрибутов, соответствующие этим знакоместам будут: 5800H и 5801H и продолжение программы будет таким:

|      |        |     |            |
|------|--------|-----|------------|
| FE9B | 320058 | LD  | (#5800), A |
| FE9E | 320158 | LD  | (#5801), A |
| FEA1 | 37     | SCF |            |
| FEA2 | C9     | RET |            |

При этом в выбранных знакоместах должен находиться какой-нибудь графический образ. Тогда те пикселы, которые выключены (имеют цвет PAPER) будут иметь чёрный цвет и меняться не будут, а включенные пикселы (имеют цвет INK) будут переливаться красно-жёлтыми полосами, воспроизводя такие же полосы, какие мы видим по бордюру экрана при загрузке. Довольно-таки оригинальный эффект получается: одно и то же знакоместо имеет чёрный фон и, в то же время, на включенных пикселах наблюдаем красные и жёлтые полосы. Вы спросите, как же такое вообще может быть? Ведь "Спектрум" устроен так, что в принципе одно знакоместо не может иметь больше двух цветов: INK и PAPER? Дело здесь в том, что на получающуюся картину накладывается ещё одно явление - строчная развёртка экрана.

Поэтому быстрая смена цвета INK приводит к тому, что во время развёртки кадра одна строка имеет, красный цвет, а другая строка развёртки уже имеет жёлтый цвет, что и создаёт необычный эффект.

Ещё на один оригинальный способ загрузки, применённый в программе "KRAKOUT-II", обратил внимание Павел. Смысл её состоит в том, что блок кодов с одним пилот-тоном может вести загрузку в разные адреса. Например, картинка там загружается так: верхняя треть экрана, затем атрибуты верхней трети, нижняя треть, атрибуты нижней трети, средняя треть, атрибуты средней трети. Программа при этом выглядит так:

```
LD    IX,#4000
LD    DE,#0800
LD    A,#FF
SCF
DI
INC   D
EX    AF,A'F'
DEC   D
CALL  #05EA
LD    IX,#5800
LD    DE,#0100
CALL  #05A9
RET
```

Это тоже можно несложно использовать, нужно только предварительно переделать загружаемый блок под этот загрузчик (как при загрузке экрана "наоборот"). Например, для того, чтобы загружать экран так: верхняя треть, затем её атрибуты, средняя треть и её атрибуты, нижняя треть и её атрибуты, надо вначале подготовить блок кодов к загрузке при помощи такой программы на Бейсике.

```
1 CLS: LOAD "" CODE 16384
2 FOR a=0 TO 2048: POKE (40000+a),PEEK (16384+a): NEXT a
3 FOR a=0 TO 256: POKE (42048+a),PEEK (22528+a): NEXT a
4 FOR a=0 TO 2048: POKE (42304+a),PEEK (18432+a): NEXT a
5 FOR a=0 TO 256: POKE (44352+a),PEEK (22784+a): NEXT a
6 FOR a=0 TO 2048: POKE (44608+a),PEEK (20480+a): NEXT a
7 FOR a=0 TO 256: POKE (46656+a),PEEK (23040+a): NEXT a
8 SAVE "name" CODE 40000,6912
```

Для загрузки этого блока кодов (без заголовка) может быть использована следующая программа в кодах.

|      |          |      |          |
|------|----------|------|----------|
| AB00 | DD210040 | LD   | IX,#4000 |
| AB04 | 110008   | LD   | DE,#0800 |
| AB07 | 3EFF     | LD   | A,#FF    |
| AB09 | 37       | SCF  |          |
| AB0A | F3       | DI   |          |
| AB0B | 14       | INC  | D        |
| AB0C | 08       | EX   | AF,A'F'  |
| AB0D | 15       | DEC  | D        |
| AB0E | CD6A05   | CALL | #056A    |
| AB11 | DD210058 | LD   | IX,#5800 |
| AB15 | 110001   | LD   | DE,#0100 |
| AB18 | CDA905   | CALL | #05A9    |
| AB1B | DD210048 | LD   | IX,#4800 |
| AB1F | 110008   | LD   | DE,#0800 |
| AB22 | CDA905   | CALL | #05A9    |
| AB25 | DD210059 | LD   | IX,#5900 |
| AB29 | 110001   | LD   | DE,#0100 |
| AB2C | CDA905   | CALL | #05A9    |
| AB2F | DD210050 | LD   | IX,#5000 |
| AB33 | 110008   | LD   | DE,#0800 |

|      |          |      |           |
|------|----------|------|-----------|
| AB36 | CDAA905  | CALL | #05A9     |
| AB39 | DD21005A | LD   | IX, #5A00 |
| AB3D | 110001   | LD   | DE, #0100 |
| AB40 | CDA905   | CALL | #05A9     |
| AB43 | FB       | EI   |           |
| AB44 | D8       | RET  | C         |
| AB45 | CF       | RST  | 8         |
| AB46 | 1A       | DEFB | #1A       |

Для примера, этот блок кодов расположен с адреса AB00H, но он может находиться в любом другом месте.

Если контроль качества считывания не нужен, то замените команду по адресу AB44H на RET (C9H).

Изложенное явление можно использовать не только для загрузки картинки, но и для любого другого массива кодов, причём количество таких дописываемых по отдельности блоков может быть достаточно большим, создавая много хлопот взломщикам.

А теперь предлагаем ещё один способ нестандартной загрузки, присланной Комиссаровым Павлом. Он "вытащил" его из программы "FREDDY-2". Этот загрузчик загружает экранный файл длиной 6912 байт по четвертям экрана в шахматном порядке. Приблизительно это выглядит так:

|   |   |
|---|---|
| 1 | 3 |
| 4 | 2 |

Причём загрузка изображения в каждую четверть идёт так: Сначала - графика, но не как обычно - линиями с шагом по 8 пикселей, да ещё с делением по третям экрана, а пиксельные линии идут подряд, одна за другой, сверху донизу, картинка разворачивается плавно и непрерывно, пока не заполнится вся четверть экрана. Затем - загрузка атрибутов, но опять, не как обычно - по строкам сверху вниз - а по столбцам, то есть изображение окрашивается слева направо. И так повторяется для каждой четверти.

Для того, чтобы загрузить таким образом картинку, надо её сначала подготовить. Павел предлагает программу на Бейсике, которая это выполняет аналогично тому, как это делалось при загрузке "наоборот".

```

1 CLEAR 39999: LOAD "" CODE 40000, 6912
10 LET na=50000
20 FOR q=1 TO 4: FOR s=0 TO 11: READ adr: PRINT s; CHR$ 143;: FOR k=0 TO 7: FOR a=0 TO 15
30 POKE na, PEEK adr: LET adr=adr+1: LET na=na+1: NEXT a: LET adr=adr+240: NEXT k: NEXT s:
  PRINT: PRINT
40 READ adr: LET ka=adr: FOR a=0 TO 15: FOR b=0 TO 11: POKE na, PEEK adr: LET adr=adr+32:
  LET na=na+1: NEXT b
50 LET adr=ka+1: LET ka=ka+1: NEXT a: NEXT q
60 BEEP .1, 32: SAVE "name" CODE 50000, 6912: STOP
70 DATA 40000,40032,40064,40096,40128,40160,40192,40224,42048,42080,42112,42144,46144
80 DATA 42192,42224,42256,42288,44112,44144,44176,44208,44240,44272,44304,44336,46544
90 DATA 40016,40048,40080,40112,40144,40176,40208,40240,42064,42096,42128,42160,46160
100 DATA 42176,42208,42240,42272,44096,44128,44160,44192,44224,44256,44288,44320,46528

```

После запуска программа ждёт загрузки нормальной картинки, а после загрузки начинается перекодировка. Пока она выполняется, на экран выводятся цифры. По ним можно определить, много ли осталось ждать до завершения работы. Программа должна

вывести четыре строки чисел от 0 до 11.

После этого программа предложит записать полученный блок кодов. При записи пропустите заголовок, и только после него включайте магнитофон.

Загрузка полученного блока кодов без заголовка выполняется при помощи следующего загрузчика:

|      |          |      |             |      |        |      |           |
|------|----------|------|-------------|------|--------|------|-----------|
| EA60 | ED7360EB | LD   | (#EB60), SP | EAD2 | DDE5   | PUSH | IX        |
| EA64 | 37       | SCF  |             | EAD4 | 1208   | LD   | E, #08    |
| EA65 | 3E00     | LD   | A, #00      | EAD6 | DDE5   | PUSH | IX        |
| EA67 | FE01     | CP   | #01         | EAD8 | 0610   | LD   | B, #10    |
| EA69 | 08       | EX   | AF, AF'     | EADA | C5     | PUSH | BC        |
| EA6A | F3       | DI   |             | EADB | CD4EEB | CALL | #EB4E     |
| EA6B | 3E0F     | LD   | A, #0F      | EADE | 3068   | JR   | NC, #EB48 |
| EA6D | D3FE     | OUT  | (#FE), A    | EAE0 | C1     | POP  | BC        |
| EA6F | DBFE     | IN   | A, (#FE)    | EAE1 | DD7500 | LD   | (IX+0), L |
| EA71 | 1F       | RRA  |             | EAE4 | DD23   | INC  | IX        |
| EA72 | E620     | AND  | #20         | EAE6 | 10F2   | DJNZ | #EADA     |
| EA74 | F602     | OR   | #02         | EAE8 | DDE1   | POP  | IX        |
| EA76 | 4F       | LD   | C, A        | EAEA | DD24   | INC  | I         |
| EA77 | BF       | CP   | A           | EAEC | 1D     | DEC  | E         |
| EA78 | C248EB   | JP   | NZ, #EB48   | EAED | 20E7   | JR   | NZ, #EAD6 |
| EA7B | CDE705   | CALL | #05E7       | EAEF | DDE1   | POP  | IX        |
| EA7E | 30F8     | JR   | NC, #EA78   | EAFF | DD7C   | LD   | A, I      |
| EA80 | 211504   | LD   | HL, #0415   | EAF1 | E618   | AND  | #18       |
| EA83 | 10FE     | DJNZ | #EA83       | EAF3 | 47     | LD   | B, A      |
| EA85 | 2B       | DEC  | HL          | EAF5 | DD7D   | LD   | A, X      |
| EA86 | 7C       | LD   | A, H        | EAF6 | E6E0   | AND  | #E0       |
| EA87 | H5       | OR   | L           | EAF8 | 07     | RLCA |           |
| EA88 | 20F9     | JR   | NZ, #EA83   | EAFB | 07     | RLCA |           |
| EA8A | CDE305   | CALL | #05E3       | EAFD | 07     | RLCA |           |
| EA8D | 30E9     | JR   | NC, #EA78   | EAFE | BC     | OR   | B         |
| EA8F | 069C     | LD   | B, #9C      | EAFD | 3C     | INC  | A         |
| EA91 | CDE305   | CALL | #05E3       | EAFD | 47     | LD   | B, A      |
| EA94 | 30E2     | JR   | NC, #EA78   | EAFD | E618   | AND  | #18       |
| EA96 | 3EC6     | LD   | A, #C6      | EAFD | F640   | OR   | #40       |
| EA98 | B8       | CP   | B           | EAFD | DD67   | LD   | I, A      |
| EA99 | 30E0     | JR   | NC, #EA7B   | EAFD | 78     | LD   | A, B      |
| EA9B | 24       | INC  | H           | EAFD | E607   | AND  | #07       |
| EA9C | 20F1     | JR   | NZ, #EA8F   | EAFD | 0F     | RRCA |           |
| EA9E | 06C9     | LD   | B, #C9      | EAFD | 0F     | RRCA |           |
| EA9F | CDE705   | CALL | #05E7       | EAFD | 0F     | RRCA |           |
| EAA0 | 30D3     | JR   | NC, #EA78   | EAFD | 47     | LD   | B, A      |
| EAA3 | 78       | LD   | A, B        | EAFD | DD7D   | LD   | A, X      |
| EAA5 | FED4     | CP   | #D4         | EAFD | E61F   | AND  | #1F       |
| EAA8 | 30F4     | JR   | NC, #EA9E   | EAFD | B0     | OR   | B         |
| EAAA | CDE705   | CALL | #05E7       | EAFD | DD6F   | LD   | X, A      |
| EAAD | D248EB   | JP   | NC, #EB48   | EAFD | 15     | DEC  | D         |
| EAB0 | 79       | LD   | A, C        | EAFD | 20BB   | JR   | NZ, #EAD2 |
| EAB1 | EE03     | XOR  | #03         | EAFD | DDE1   | POP  | IX        |
| EAB3 | 4F       | LD   | C, A        | EAFD | DD7C   | LD   | A, I      |
| EAB4 | 2600     | LD   | H, #00      | EAFD | E618   | AND  | #18       |
| EAB6 | 06B0     | LD   | B, #B0      | EAFD | 0F     | RRCA |           |
| EAB8 | CD4EEB   | CALL | #EB4E       | EAFD | 0F     | RRCA |           |
| EABB | 118048   | LD   | DE, #4880   | EAFD | 0F     | RRCA |           |
| EABE | D5       | PUSH | DE          | EAFD | F658   | OR   | #58       |
| EABF | 111040   | LD   | DE, #4010   | EAFD | DD67   | LD   | I, A      |
| EAC2 | D5       | PUSH | DE          | EAFD | 0610   | LD   | B, #10    |
| EAC3 | 119048   | LD   | DE, #4890   | EAFD | C5     | PUSH | BC        |
| EAC6 | D5       | PUSH | DE          | EAFD | 060C   | LD   | B, #0C    |
| EAC7 | DD210040 | LD   | IX, #4000   | EAFD | DDE5   | PUSH | IX        |
| EACB | 0604     | LD   | B, #04      | EAFD | C5     | PUSH | BC        |
| EACD | C5       | PUSH | BC          | EAFD | CD4EEB | CALL | #EB4E     |
| EACE | DDE5     | PUSH | IX          | EAFD | 3017   | JR   | NC, #EB48 |
| EAD0 | 160C     | LD   | D, #0C      | EAFD | DD7500 | LD   | (IX+0), L |

|      |          |      |             |      |        |      |           |
|------|----------|------|-------------|------|--------|------|-----------|
| EB34 | 012000   | LD   | BC, #0020   | EB4C | FB     | EI   |           |
| EB37 | DD09     | ADD  | IX, BC      | EB4D | C9     | RET  |           |
| EB39 | C1       | POP  | BC          | EB4E | 06B4   | LD   | B, #B4    |
| EB3A | 10EF     | DJNZ | #EB2B       | EB50 | 2E01   | LD   | L, #01    |
| EB3C | DDE1     | POP  | IX          | EB52 | CDE305 | CALL | #05E3     |
| EB3E | DD23     | INC  | IX          | EB55 | D0     | RET  | NC        |
| EB40 | C1       | POP  | BC          | EB56 | 3ECB   | LD   | A, #CB    |
| EB41 | 10E3     | DJNZ | #EB26       | EB58 | B8     | CP   | B         |
| EB43 | C1       | POP  | BC          | EB59 | CB15   | RL   | L         |
| EB44 | DDE1     | POP  | IX          | EB5B | 06B0   | LD   | B, #B0    |
| EB46 | 1085     | DJNZ | #EACD       | EB5D | 30F3   | JR   | NC, #EB52 |
| EB48 | ED7B60EB | LD   | SP, (#EB60) | EB5F | C9     | RET  |           |

## ИСПРАВЛЕННЫЕ ОШИБКИ ПЗУ.

В последних двух номерах прошлого года мы опубликовали обзор зарубежной прессы, обобщив те сведения, которые имеются по ошибкам в стандартном системном ПЗУ "Спектрума".

Работа получила благожелательные отзывы от наших читателей. По-видимому, им важно всё, что связано с ПЗУ, а тем более с его ошибками.

Вместе с тем встал интересный вопрос: "А как обстоит дело в ПЗУ ТУРБО-90?". Результатами своих исследований по этому поводу делятся наши корреспонденты Роман Купцов и Мальтов Денис из г. Новгорода. Тема не закрыта, мы опубликуем и другие взгляды, освещающие этот вопрос, тем более, что интересно бы рассмотреть и свои "собственные" ошибки Турбо-ПЗУ.

**КОРР:** Мы являемся постоянными читателями "ZX-РЕВЮ" с 1992 года. Ваш журнал выше всяких похвал! Сами мы занимаемся программированием три года и Вы нам в этом очень помогли, особенно в том, что связано с программированием в машинных кодах. У Вас постоянно свежий и хорошо поданный материал, а Ваши статьи по секретам ПЗУ в 1991 г. - просто класс! Неоценимая помощь любому программисту.

**ИФК:** Спасибо, конечно, друзья на добром слове, но позвольте нам остаться при своём мнении. Статьи эти о ПЗУ в 1991 году были, возможно, полезны, но мы их рассматриваем как **СОВЕРШЕННО НЕУДАЧНЫЙ** опыт и **РЕШИТЕЛЬНО** ими **НЕДОВОЛЬНЫ**. Знаете ведь: "Первый блин... и т.д.". В общем, случился с нами приступ внутренней аллергии. Так бывает, когда хочешь сделать что-то хорошее, а чувствуешь, что получается не то.

Потому мы и прекратили их давать. Мы тогда только начинали и опыта общения с читателями было маловато. Сейчас мы всё сделали бы по-другому. И будьте уверены, сделаем! В наших планах на 2-ой квартал 1994 года стоит книга о системном ПЗУ, которая действительно класс! Мы готовим удобный и понятный способ подачи материала и уверены, что книга будет отличной. Так что позвольте Ваши комплименты принять авансом, а мы постараемся этот аванс отработать по-честному.

.....  
.....

**КОРР:** Мы хотим поделиться с Вами и со всеми читателями некоторыми находками из ПЗУ ТУРБО-90. Немного покопавшись в нём, мы обнаружили (не без помощи "ZX-РЕВЮ"), что некоторые из ошибок системного ПЗУ там исправлены.

### Исправлено:

1. Процедура обработки немаскированного прерывания INT, начиная с адреса 0066H, т. е.

```
PUSH AF
PUSH HL
LD HL, (5CB0)
LD A, H
OR L
JR Z, 0070H; ошибка исправлена, в фирменном ПЗУ здесь стоит JR NZ, 0070H
```

### 2. Ошибка деления, т. е.:

```
IF 1/2 <> 0.5 THEN PRINT "Ku-Ku".
```

В Турбо-ПЗУ здесь "Ku-Ku" не печатается.

3. Ошибка "-65536"; т.е. PRINT IN "-65536" даёт - 65536.

4. CHR\$ 9 работает.

5. CHR\$ 8 работает правильно.

6. STR\$ работает правильно, т. е.

```
PRINT "Ku-Ku" + STR$ 0.5
```

даёт:

Ku-Ku0.5

7. Ошибка SCREEN\$ исправлена, т.е.

```
IF "X"=SCREEN$(0,0) THEN PRINT "Ku-Ku"
```

Ku-Ku не печатается, когда в текущем знакоместе (0, 0) нет "X"

8. Исправлена ошибка курсора текущей строки: т.е.

```
9000 PRINT\9001\EDIT,
```

где значок "\" означает ENTER, выдаёт на редактирование строку 9000, а значок ">" в нижней части экрана не появляется.

9. Исправлена ошибка ведущего пробела.

Неисправленные ошибки:

1. Особенность регистровой пары H'L' (альтернативной).
2. Ошибка оператора PLOT.
3. Ошибка CLOSE\$.
4. Ошибка Scroll?
5. Ошибка K-режима.
6. Ошибка проверки синтаксиса.

Непроверенные ошибки.

1. Ограничение по использованию регистровой пары Y.
2. Особенности пользовательской функции FN.
3. Ошибки кодов управления цветом.
4. Ошибка MOD\_DIV (кальк-р).
5. Ошибка E\_TO\_FP (кальк-р).
6. Ошибка INKEY\$ #0.

# Adventure Games

## "Adventure Building system"

Окончание. Начало см. в предыдущем выпуске.

Чтобы завершить демонстрационную программу, начатую нами в предыдущем номере "ZX-РЕВЮ", загрузите программу с ленты В (Листинг\_4) и добавьте строки: с 1100 по 3315 из Листинга\_5. Сохраните готовую демонстрационную игру командой RUN 9990. Скомпонуйте файлы готовой демонстрационной программы в следующем порядке:

Файл "INTRO" (Лист.\_3)

Файл "adventure" (Лист.\_4+5)

Файл "system"CODE

Файл "objects"DATA

Последние два файла сгенерированы при помощи головной программы ABS.

### Листинг\_5.

Текст 2-ой части демонстрационной игры. Продолжение файла "adventure".

```
1100 IF NO>MOV THEN LET R$(1)="ЭТО НЕВОЗМОЖНО!": GO TO 50
1105 IF PEEK (0+NO)=99 THEN LET R$(1)="ПРОВЕРЬТЕ СВОЙ ИНВЕНТАРЬ!": GO TO 50
1110 IF PEEK (0+NO)<>PEEK F THEN LET R$(1)="НАВЕРНОЕ, ЭТО ВАМ ПРИСНИЛОСЬ!": GO TO 50
1115 IF PEEK (F+99)>=MAX THEN LET R$(1)="У ВАС ВЕДЬ ТОЛЬКО ДВЕ РУКИ,": LET R$(2)="ВАМ ВСЕ
    ЭТО НЕ УНЕСТИ.": GO TO 50
1120 IF NO=7 THEN CLS : GO SUB 8800: PRINT AT 2,10;"ПОЗДРАВЛЯЕМ": GO SUB 8800: PRINT AT
    5,3;"ВЫ ЗАКОНЧИЛИ СВОЮ МИССИЮ.": GO SUB 8800: GO TO 1602
1125 POKE (0+NO),99: POKE (F+99),PEEK (F+99)+1: LET R$(1)="О.К." : GO TO 50
1200 IF NO>MOV THEN LET R$(1)="НЕ БУДЕМ ДЕЛАТЬ ГЛУПОСТЕЙ!": GO TO 50
1205 IF PEEK (0+NO)<>99 THEN LET R$(1)="ПРОВЕРЬТЕ СВОЙ ИНВЕНТАРЬ!": GO TO 50
1210 IF NO=9 AND PEEK F<>11 THEN POKE (0+9),0: POKE (0+2),PEEK F: POKE (F+2),0: POKE
    (F+99),PEEK (F+99)-1: LET R$(1)="О.К. ОН СРАЗУ ЖЕ ПОГАС.": GO TO 50
1215 IF NO=9 THEN POKE (0+9),0:POKE (0+2),PEEK F: POKE (F+2),0: POKE (F+99),PEEK (F+99)-1:
    RANDOMIZE USR SS: PRINT INK 5;"О.К. НО ОН ТУТ ЖЕ ГАСНЕТ!!!": GO SUB 8800: GO TO 100
1220 IF NO=8 THEN LET R$(1)="НО ВЕДЬ ОН НАДЕТ НА ВАС.": GO TO 50
1230 POKE (0+NO),PEEK F: POKE (F+99),PEEK (F+99)-1: LET R$(1)="О.К.": GO TO 50
1300 IF NO<10 THEN GO TO 1350
1302 IF NO<>16 THEN GO TO 1315
1305 IF (PEEK F=8 OR PEEK F=9) THEN LET R$(1)="ДВЕРЬ СДЕЛАНА ИЗ ДОБРОТНОГО": LET
    R$(2)="ДЕРЕВА И В ДАННЫЙ МОМЕНТ": LET R$(3)="ЗАКРЫТА, НО НЕ ЗАПЕРТА."
1310 IF PEEK (F+4)=1 THEN LET R$(3)="ОСТАЕТСЯ ШИРОКО ОТКРЫТОЙ."
1313 GO TO 50
1315 IF NO<>21 THEN GO TO 1335
1317 IF (PEEK F=14 OR PEEK F=15) THEN LET R$(1)="ВОРОТА СДЕЛАНЫ ИЗ НЕВАЖНОГО": LET
    R$(2)="ДЕРЕВА И В ДАННЫЙ МОМЕНТ": LET R$(3)="ЗАКРЫТЫ ИЗНУТРИ НА ЗАСОВ."
1320 IF PEEK (F+7)=1 THEN LET R$(3)="ОСТАЮТСЯ ОТКРЫТЫМИ."
1322 GO TO 50
1325 IF (PEEK F=15 AND (NO=23 OR NO=18) AND PEEK (F+6)=1) OR (PEEK F=7 AND PEEK (F+5)=1 AND
    NO=24) THEN LET R$(1)="ЗДЕСЬ ЯМА, БУДЬТЕ ОСТОРОЖНЫ!": GO TO 50
1330 IF ((PEEK F=15 AND (NO=23 OR NO=18)) OR (PEEK F=7 AND NO=24)) THEN LET R$(1) =
    "ЧУВСТВУЕТСЯ ЗАБОТЛИВЫЙ УХОД,": LET R$(2)="НО НЕДАВНИЙ ДОЖДЬ ОСТАВИЛ": LET
    R$(3)="РАСКИСШУЮ ПОЧВУ!": GO TO 50
1335 IF NO=22 AND ((PEEK F=7 AND PEEK (F+5)=1) OR (PEEK F=15 AND PEEK (F+6)=1)) THEN LET
    R$(1)="ЭТО ДОВОЛЬНО ГЛУБОКАЯ ЯМА,": LET R$(2)="И ЕСЛИ ТУДА СВАЛИТЬСЯ, ТО...!": GO TO
    50
1340 IF PEEK F=15 AND NO=26 AND PEEK (F+6)=0 THEN LET R$(1)="ВЫГЛЯДИТ ТАК, КАК БУДТО ЗДЕСЬ":
    LET R$(2)="КТО-ТО КОПАЛ.": GO TO 50
1345 GO TO 1375
1350 IF (PEEK (0+NO)<>PEEK F AND PEEK (0+NO)<>99) THEN LET R$(1) ="НО ВЕДЬ ЕГО РЯДОЙ С ВАМИ
    НЕТ!": GO TO 50
```



[illegible]

```

1925 IF NO=3 AND (PEEK (0+3)=99 OR PEEK (0+3)=PEEK F) THEN LET R$(1)= "НЕСМОТРЯ НИ НА КАКИЕ
      УСИЛИЯ": LET R$(2)="ОТКРЫТЬ ЕЕ НЕ УДАЕТСЯ!": GO TO 50
1930 IF PEEK F=11 AND NO=25 AND PEEK (F+8)=0 THEN LET R$(1)="НЕ ОТКРЫВАЕТСЯ! ЗАПЕРТ НА
      КЛЮЧ.": GO TO 50
1935 IF PEEK F=11 AND NO=25 AND PEEK (F+8)=1 THEN LET R$(1)="ОН УЖЕ ОТКРЫТ!": GO TO 50
1940 GO TO 45
2000 IF ((PEEK F=8 OR PEEK F=9) AND NO=16) OR ((PEEK F=14 OR PEEK F=15) AND NO=21) THEN LET
      R$(1)="ОКАЗЫВАЕТСЯ, ЗДЕСЬ НЕ ЗАПЕРТО!": GO TO 50
2005 IF ((PEEK F=8 OR PEEK F=9) AND NO=16 AND PEEK (F+4)=1) OR ((PEEK F=14 OR PEEK F=15) AND
      NO=21 AND PEEK (F+7)=1) THEN LET R$(1)="НО ВЕДЬ УЖЕ ОТКРЫТО НАСТЕЖЬ!": GO TO 50
2010 IF PEEK F=11 AND NO=25 AND PEEK (F+8)=0 AND PEEK (0+5)=99 AND PEEK (F+9)=0 THEN POKE
      (F+8), 1: POKE (F+9), 1: POKE (0+3), 11: LET R$(1)="СЕЙФ ОТКРЫВАЕТСЯ, И ВНУТРИ ВЫ": LET
      R$(2)="ВИДИТЕ ДЕРЕВЯННУЮ ШКАТУЛКУ.": GO TO 50
2015 IF PEEK F=11 AND NO=25 AND PEEK (F+8)=1 THEN LET R$(1)="А ОН ТЕПЕРЬ НЕ ЗАПЕРТ!": GO TO
      50
2020 IF PEEK F=11 AND NO=25 AND PEEK (F+8)=0 AND PEEK (0+5)<>99 THEN LET R$(1)="ДЛЯ ЭТОГО
      НУЖЕН КЛЮЧ!": GO TO 50
2030 IF NO=3 AND (PEEK (0+3)-PEEK F OR PEEK (0+3)=99) AND PEEK (0+5)=99 THEN LET R$(1)="КЛЮЧ
      К ШКАТУЛКЕ НЕ ПОДХОДИТ.": GO TO 50
2035 GO TO 45
2100 IF (PEEK F<>7 AND PEEK F<>15) THEN LET R$(1)="ЗДЕСЬ НЕЛЬЗЯ КОПАТЬ": GO TO 50
2105 IF (((PEEK F=15 AND (NO=23 OR NO=18 OR NO=22 OR NO=26)) OR (PEEK F=7 AND (NO=24 OR
      NO=22))) AND PEEK (0+4)<>99) THEN LET R$(1)="И КАК ЖЕ ЭТО СДЕЛАТЬ?": LET R$(2)="ДЛЯ
      ЭТОГО НУЖЕН ПОДХОДЯЩИЙ": LET R$(3)="ИНСТРУМЕНТ.": GO TO 50
2110 IF PEEK F=15 AND PEEK (F+10)=0 AND (NO=18 OR NO=22 OR NO=23 OR NO=26) THEN POKE
      (F+10), 1: POKE (0+5), 15: POKE (F+6), 1: LET R$(1)="ВЫ ИСПОРТИЛИ КРАСИВУЮ ЛУЖАЙКУ": LET
      R$(2)="ГЛУБОКОЙ ЯМОЙ. НО ВОТ ЧТО-ТО": LET R$(3)="БЛЕСТНУЛО... ЭТО КЛЮЧ!": GO TO 50
2115 IF ((PEEK F=15 AND PEEK (F+5)=1) OR (PEEK F=7 AND PEEK (F+5)=1)) THEN LET R$(1)="НЕТ
      СПАСИБО, БОЛЬШЕ НЕ НАДО!": GO TO 50
2120 IF PEEK F=15 AND PEEK (F+6)=0 AND PEEK (0+4)=99 AND (NO=23 OR NO=22 OR NO=18 OR NO=26)
      THEN POKE (F+6), 1: LET R$(1)="О.К. ВЫ ВЫКОПАЛИ ЯМУ В САДУ.": GO TO 50
2125 IF PEEK F=7 AND PEEK (F+5)=0 AND PEEK (0+4)=99 AND (NO=24 OR NO=22) THEN POKE (F+5), 1:
      LET R$(1)="О.К. ВЫ В ПОЛЕ ВЫКОПАЛИ ЯМУ.": GO TO 50
2130 GO TO 45
2200 IF PEEK (0+9)=99 AND NO=9 THEN LET R$(1)="НО ОН И ТАК ГОРИТ!": GO TO 50
2205 IF NO=2 AND PEEK (0+2)=99 AND PEEK F<>11 THEN POKE (0+2), 0: POKE (0+9), 99: POKE
      (F+2), 1: LET R$(1)="О.К. ТЕПЕРЬ ФАКЕЛ ГОРИТ.": GO TO 50
2210 IF NO=2 AND PEEK (0+2)=99 THEN POKE (0+2), 0: POKE (0+9), 99 : POKE (F+2), 1: RANDOMIZE
      USR SS: PRINT INK 5; "О.К. ТЕПЕРЬ ФАКЕЛ ЯРКО СВЕТИТ!": GO SUB 8800: GO TO 100
2215 GO TO 45
2300 IF PEEK (0+2)=99 AND NO=9 THEN LET R$(1)="НО ВЕДЬ ОН НЕ ГОРИТ!": GO TO 50
2305 IF NO=9 AND PEEK (0+9)=99 AND PEEK F<>1 THEN POKE (0+9), 0: POKE (0+2), 99: POKE (F+2), 0:
      LET R$(1) = "О.К. ФАКЕЛ ПОГАС.": GO TO 50
2310 IF NO=9 AND PEEK (0+9)=99 THEN POKE (0+9), 0: POKE (0+2), 99: POKE (F+2), 0: RANDOMIZE USR
      SS: PRINT INK 5; "О.К. ФАКЕЛ БОЛЬШЕ НЕ СВЕТИТ.": GO SUB 8800: GO TO 100
2315 GO TO 45
2400 IF PEEK F=11 AND (NO=19 OR NO=14) THEN POKE F, 6: GO TO 100
2405 IF PEEK F=6 AND (NO=20 OR NO=14) THEN POKE F, 2: GO TO 100
2410 IF PEEK F=2 AND NO=15 THEN POKE F, 6: GO TO 100
2412 IF PEEK F=6 AND NO=15 THEN POKE F, 11: GO TO 100
2415 IF (PEEK F=14 OR PEEK F=15) AND PEEK (F+7)=0 THEN LET R$(1)="ЭТО НЕСЕРЬЕЗНО, ЛУЧШЕ
      ОТКРОЙТЕ": LET R$(2)="ВОРОТА ИЗ САДА!": GO TO 50
2420 IF (PEEK F=14 OR PEEK F=15) AND PEEK (F+7)=1 THEN LET R$(1)="И ЗАЧЕМ ВАМ ЭТО ДЕЛАТЬ?":
      LET R$(2)="ВЕДЬ ВОРОТА ОТКРЫТЫ.": GO TO 50
2425 GO TO 45
2500 IF PEEK F=2 AND (NO=20 OR NO=15) THEN POKE F, 6: GO TO 100
2505 IF PEEK F=6 AND (NO=19 OR NO=15) THEN POKE 64280, 11: GO TO 100
2510 GO TO 45
2600 IF (PEEK F=8 OR PEEK F=9) AND NO=16 AND PEEK (F+4)=1 THEN POKE (F+4), 0: LET R$(1)="О.К.
      ТЕПЕРЬ ДВЕРЬ ЗАКРЫТА.": GO TO 50
2605 IF (PEEK F=14 OR PEEK F=15) AND NO=21 AND PEEK (F+7)=1 THEN POKE (F+7), 0: LET
      R$(1)="О.К. ТЕПЕРЬ ВОРОТА ЗАКРЫТЫ.": GO TO 50
2610 IF PEEK F=9 AND NO=17 AND PEEK (F+3)=1 THEN POKE (F+3), 0: LET R$(1)="О.К. ТЕПЕРЬ БУДЕТ
      ЗАКРЫТ.": GO TO 50

```

```

2615 IF PEEK F=11 AND NO=25 AND PEEK (F+8)-1 THEN LET R$(1)="КАЖЕТСЯ, ДВЕРЦУ ЗАКЛИНИЛО!": GO
    TO 50
2620 GO TO 45
2700 IF PEEK F=2 AND NO=3 AND PEEK (0+3)=99 AND PEEK (F+11)=1 THEN POKE (0+7),6: POKE
    (0+6), 6: POKE (0+3),0: POKE (F+99),PEEK (F+99)-1: LET R$ (1) ="ВОТ ЭТО БРОСОК!!!":
    LET R$(2) = "ОТ УДАРА ШКАТУЛКА РАЗБИЛАСЬ.": LET R$(3)="ЧТО-ТО ИЗ НЕЕ ВЫПАЛО И
    БЛЕСНУВ,": LET R$(4) = "ПОКАТИЛОСЬ ПО ПОЛУ КОНЮШНИ.": GO TO 50
2705 IF PEEK F=2 AND NO=3 AND PEEK (0+3)=99 THEN LET R$(1)="Я БЫ ЭТОГО НЕ ДЕЛАЛ.": LET
    R$(2)="НАДО ЕЩЕ КОЕ-ЧЕГО": LET R$(3)="ДОБИТЬСЯ ОТ БУФЕТА!": GO TO 50
2710 GO TO 1200
2800 IF ((PEEK F=15 AND PEEK (F+6)=1) OR (PEEK F=7 AND PEEK (F+5)=1)) AND PEEK (0+4)<>99 AND
    NO=22 THEN LET R$(1)="И КАК ЖЕ ВЫ СОБИРАЕТЕСЬ ЭТО": LET R$(2)="СДЕЛАТЬ БЕЗ НУЖНОГО
    ИНСТРУМЕНТА!": GO TO 50
2802 IF (PEEK F=15 AND PEEK (F+6)=0 AND (NO=22 OR NO=26)) OR (PEEK F=7 AND PEEK (F+5)=0 AND
    NO=22) THEN LET R$ (1)="ШИКАРНО ПОЛУЧИЛОСЬ!": GO TO 50
2805 IF PEEK F=7 AND NO=22 AND PEEK (0+4)=99 AND PEEK (F+5)=1 THEN POKE (F+5),0: LET
    R$(1)="О.К. ВЫ ЗАКОПАЛИ ЯМУ В ПОЛЕ.": GO TO 50
2810 IF PEEK F=15 AND NO=22 AND PEEK (0+5)=15 THEN LET R$(1)="ОСТОРОЖНО! ЗАКОПАЕТЕ КЛЮЧ!":
    GO TO 50
2812 IF PEEK F=15 AND NO=22 AND PEEK (0+4)=99 AND PEEK (F+6)=1 THEN POKE (F+6),0: LET
    R$(1)="О.К. ВЫ ЗАКОПАЛИ ЯМУ": LET R$(2)="И ОСТАВИЛИ НА ЛУЖАЙКЕ": LET
    R$(3)="НЕКРАСИВОЕ ПЯТНО!": GO TO 50
2815 GO TO 45
2900 IF PEEK F=11 AND NO=25 AND
2902 (F+8)=1 THEN LET R$(1)="НЕ ПОЛУЧАЕТСЯ, ДВЕРЬ ЗАЕЛО!": GO TO 50
2905 GO TO 45
3000 IF NO=1 AND PEEK(0+1)=99 THEN POKE (0+1),0: POKE (0+8),99: POKE (F+1),1: POKE
    (F+99),PEEK (F+99)-1: LET R$(1)="ОТЛИЧНО! ОСВОБОДИЛАСЬ РУКА.": LET R$(2)="ДА И ПЛЕД
    ВАМ ОЧЕНЬ ИДЕТ.": GO TO 50
3005 IF NO=1 AND PEEK (0+8)=99 THEN LET R$(1)="ОН УЖЕ ОДЕТ НА ВАС!": GO TO 50
3008 IF NO=1 THEN LET R$(1)="НЕ ПОЛУЧАЕТСЯ, СНАЧАЛА НАДО": LET R$(2)="ВЗЯТЬ ЕГО В РУКИ.": GO
    TO 50
3010 GO TO 45
3100 IF NO=8 AND PEEK (F+99)=MAX THEN LET R$(1)="НЕ ВЫХОДИТ! ОБЕ РУКИ ЗАНЯТЫ.": GO TO 50
3102 IF NO=8 THEN POKE (0+1),99 : POKE (0+8),0: POKE (F+1),0: POKE (F+99),PEEK (F+99)+1: LET
    R$(1)="О.К. НО ПОЧЕМУ?": LET R$(2)="РАЗВЕ ВЫ К НЕМУ ЕЩЕ": LET R$(3)="НЕ ПРИШЛИ?": GO
    TO 50
3105 IF NO=1 THEN LET R$(1)="НО ЕГО НЕТ НА ВАС!": GO TO 50
3110 GO TO 45
3200 IF PEEK F=2 AND (NO=15 OR NO=27) THEN POKE F,6: FOR X=-5 TO 15: BEEP .04,K: NEXT X: FOR
    X=14 TO -5 STEP -1: BEEP .08,X: NEXT X: BEEP 1,-6: GO TO 100
3205 IF (NO=27 OR NO=5 OR NO=5) THEN LET R$(1)="О.К. НО НИЧЕГО НЕ СЛУЧИЛОСЬ!": GO TO 50
3210 GO TO 45
3300 IF NO<10 AND PEEK (0+NO)<>99 THEN LET R$(2)="ПРОВЕРЬТЕ СВОЙ ИНВЕНТАРЬ!": GO TO 50
3305 IF NO=5 AND PEEK (F+8)=0 THEN LET R$(1)="И КАК ЖЕ НАДО ИСПОЛЬЗОВАТЬ": LET R$(2)="ЭТОТ
    КЛЮЧ?": GO TO 50
3310 IF NO=4 THEN LET R$(1)="ПОДУМАЙТЕ... - НАВЕРНОЕ,": LET R$(2)="ВЫ ЗНАЕТЕ, ЧТО МОЖНО
    ДЕЛАТЬ": LET R$(3)="С ПОМОЩЬЮ ЛОПАТЫ!":GO TO 50
3315 LET R$(1)="ЧТО ЗНАЧИТ **ИСПОЛЬЗОВАТЬ**!": LET R$(2)="НЕЛЬЗЯ ЛИ УТОЧНИТЬ?": GO TO 50

```

Теперь пришло время расшифровки некоторых ключевых моментов, которые помогут Вам отслеживать логику работы программы.

### **Управление адвентюрной игрой.**

БЕЙСИК-программа управляет игрой с помощью следующих данных:

- номер глагола (или команды) (VB);
- номер существительного (или направления) (NO);
- таблицы текущего расположения объектов;
- таблицы состояния флагов.

В прошлом номере "ZX-РЕВЮ" мы рассмотрели параметры, задаваемые при помощи строк DATA (Лист. 2). Теперь, подводя итог сказанному, мы приводим полный перечень слов (Листинги 1 и 2), распознаваемых программой, в виде двух таблиц, отдельно для глаголов и существительных (и прочих слов). Для каждого слова указан его номер.

## Анализатор текста.

Система, анализирующая текст, позволяет вводить до 30 символов, среди них могут быть русские и латинские заглавные буквы, цифры и символы. Русские буквы набираются непосредственно буквенными клавишами, за исключением Ч, Э, Ш, Щ, Ю. Последние набираются одновременно с нажатием SYMB. SH. клавишами А, S, F, G, X. Для "забоя" последнего введенного символа используется клавиша "DELETE" или "КУРСОР ВЛЕВО". Приглашение процедуры ввода изображается символом ">", а курсор - в виде символа "\*".

```

ВЫ НАХОДИТЕСЬ У ДВЕРИ ФЕРМЕР-
СКОГО ДОМА. ДОМ СДЕЛАН ОЧЕНЬ
ДОБОРОТНО И КРЕПКО, КАК БУДТО
ХОЗЯЕВАМ БЫЛО ОТ ЧЕГО СКРЫ-
ВАТЬСЯ.

В ПОЛЕ ВАШЕГО ЗРЕНИЯ:
НЕТУ СОВСЕМ НИЧЕГО

ВАШИ ДЕЙСТВИЯ?
>ИДИ НА СЕВЕР
ТУДА НЕ ПРОЙТИ
ВАШИ ДЕЙСТВИЯ?
>ИДИ НА ВОСТОК
ДВЕРЬ ЗАКРЫТА!
ВАШИ ДЕЙСТВИЯ?
>ОТКРЫТЬ ДВЕРЬ
О.К. ТЕПЕРЬ ДВЕРЬ ОТКРЫТА.
ВАШИ ДЕЙСТВИЯ?
>ИДИ НА ВОСТОК *
  
```

Как уже говорилось, процедура построена таким образом, что анализирует пробелы между словами, выделяя первое и последнее слово. Первое слово считается глаголом, а последнее - существительным или направлением перемещения. Поэтому команды "БРОСИТЬ ЭТУ ЛОПАТУ", "БРОСИТЬ КОРОТКУЮ ЛОПАТУ" дают тот же эффект, что и просто "БРОСИТЬ ЛОПАТУ".

Анализатор просматривает введенный текст команды и возвращается в БЕЙСИК с числом VB, соответствующим номеру использованного глагола и с числом NO, соответствующим номеру использованного существительного. Любое нераспознанное слово выдаёт в БЕЙСИК число 200 для глагола и число 201 для существительного.

Таблица 1.

| ГЛАГОЛЫ       |                   |
|---------------|-------------------|
| ОСНОВНЫЕ      | ДОПОЛНИТЕЛЬНЫЕ    |
| ИДИТИ.....0   | ОТКРЫТЬ.....9     |
| СЕВЕР.....0   | ОТПЕРЕТЬ.....10   |
| С.....0       | КОПАТЬ.....11     |
| ЮГ.....0      | ВЫКОПАТЬ.....11   |
| Ю.....0       | ЗАЖЕЧЬ.....12     |
| ВОСТОК.....0  | ГАСИТЬ.....13     |
| В.....0       | ПОГАСИТЬ.....13   |
| НАПРАВО.....0 | ПОДНЯТЬСЯ.....14  |
| ЗАПАД.....0   | ЗАЛЕЗТЬ.....14    |
| З.....0       | ЛЕЗТЬ.....14      |
| НАЛЕВО.....0  | СПУСТИТЬСЯ.....15 |
| ВВЕРХ.....0   | СЛЕЗТЬ.....15     |
| Х.....0       | ЗАКРЫТЬ.....16    |
| ВНИЗ.....0    | ШВЫРНУТЬ.....17   |
| Н.....0       | ЗАКОПАТЬ.....18   |
| ВЗЯТЬ.....1   | ЗАПЕРЕТЬ.....19   |
| БРАТЬ.....1   | ОДЕТЬ.....20      |
| ПОЛОЖИТЬ...2  | НАДЕТЬ.....20     |
| БРОСИТЬ....2  | СНЯТЬ.....21      |
| ПРОВЕРИТЬ..3  | ПРЫГАТЬ.....22    |
| ОСМОТРЕТЬ..3  | СПРЫГНУТЬ.....22  |
| СМОТРЕТЬ...4  | ЗАПРЫГНУТЬ.....22 |
| ИНВЕНТАРЬ..5  | ИСПОЛЬЗОВАТЬ..23  |
| И.....5       |                   |
| СТОП.....6    |                   |
| КОНЕЦ.....6   |                   |
| СОХРАНИТЬ..7  |                   |
| ЗАГРУЗИТЬ..8  |                   |

Эта информация может обыгрываться в БЕЙСИКЕ как угодно, но в нашей демонстрационной программе применяется самое простое решение, - в этом случае пользователь всегда получает ответ "НЕ ПОЛУЧАЕТСЯ".

Если же Вы даёте команду, состоящую из одного слова, например, "СМОТРЕТЬ", то за глагол Вы получите VB=4, а за отсутствие существительного - должны были бы получить NO=201, что вызовет реакцию "НЕ ПОЛУЧАЕТСЯ", хотя нам такая реакция не нужна. Для того, чтобы избежать подобных ситуаций, мы включаем глагол "СМОТРЕТЬ" и в список существительных тоже, при этом присваиваем ему порядковый номер 99 (см. Листинг\_1, строка 2190 и Таблицу\_2).

Со всеми другими глаголами (командами), которые могут использоваться без существительного, надо поступить тем же образом.

Возможен и другой вариант, когда такому глаголу Вы присваиваете номер не 99, а соответствующий тому существительному, которое с этим глаголом обычно связано. В демонстрационной программе мы так поступили со словом КОПАТЬ - ему присвоен номер 22 (см. Листинг\_2, строка 2192 и Таблицу\_2), потому что КОПАТЬ и КОПАТЬ ЯМУ - понятия равнозначные. Поэтому пусть читатель, бросивший беглый взгляд на таблицы глаголов и существительных без чтения текста, не упрекает нас в незнании основ русской грамматики.

Таблица\_2

| СУЩЕСТВИТЕЛЬНЫЕ И ДРУГИЕ СЛОВА |    |
|--------------------------------|----|
| ИНВЕНТАРЬ                      |    |
| ПЛЕД.....                      | 1  |
| ШЕРСТЯНОЙ ПЛЕД.....            | 1  |
| ФАКЕЛ.....                     | 2  |
| МАЛЕНЬКИЙ ФАКЕЛ.....           | 2  |
| ШКАТУЛКА.....                  | 3  |
| ДЕРЕВЯННАЯ ШКАТУЛКА.....       | 3  |
| ЛОПАТА.....                    | 4  |
| КОРОТКАЯ ЛОПАТА.....           | 4  |
| КЛЮЧ.....                      | 5  |
| БРОНЗОВЫЙ КЛЮЧ.....            | 5  |
| ЩЕПКИ.....                     | 6  |
| ДЕРЕВЯННЫЕ ЩЕПКИ.....          | 6  |
| АЛМАЗ.....                     | 7  |
| КРУПНЫЙ АЛМАЗ.....             | 7  |
| ПЛЕД, НАДЕТЫЙ НА ПЛЕЧИ....     | 8  |
| ГОРЯЩИЙ ФАКЕЛ.....             | 9  |
| НАПРАВЛЕНИЕ                    |    |
| СЕВЕР.....С.....               | 10 |
| ЮГ.....Ю.....                  | 11 |
| ВОСТОК.....В...НАПРАВО....     | 12 |
| ЗАПАД.....З...НАЛЕВО....       | 13 |
| ВВЕРХ.....Х.....               | 14 |
| ВНИЗ.....Н.....                | 15 |

| ПРОЧИЕ СЛОВА   |    |
|----------------|----|
| ДВЕРЬ.....     | 16 |
| БУФЕТ.....     | 17 |
| ЛУЖАЙКА.....   | 18 |
| СТУПЕНИ.....   | 19 |
| ЛЕСТНИЦА.....  | 20 |
| ВОРОТА.....    | 21 |
| КОПАТЬ.....    | 22 |
| ЯМУ.....       | 22 |
| САД.....       | 23 |
| ПОЛЕ.....      | 24 |
| СЕЙФ.....      | 25 |
| ПЯТНО.....     | 26 |
| ПРЫГАТЬ.....   | 27 |
| КОМАНДЫ        |    |
| СМОТРЕТЬ.....  | 99 |
| ИНВЕНТАРЬ..... | 99 |
| И.....         | 99 |
| СТОП.....      | 99 |
| КОНЕЦ.....     | 99 |
| СОХРАНИТЬ..... | 99 |
| ЗАГРУЗИТЬ..... | 99 |

Обратите внимание, что для тех существительных, которые имеют длину, меньше заданной для идентификации (4 символа) - это, например, существительные ЯМА, САД - задавать их надо в родительном падеже, то есть: ЯМУ (а САД, ПОЛЕ, СЕЙФ и т.д. и в родительном и в именительном падеже совпадают). Ведь подаваемая команда может быть: ВЫКОПАТЬ ЯМУ или ОСМОТРЕТЬ ЯМУ или ПРЫГНУТЬ В ЯМУ и т. д. Если же допускать такие команды, которые потребуют так же и слова ЯМА, например, ПРОВЕРИТЬ, КАК ВЫКОПАНА ЯМА, то существительные надо задавать дважды: ЯМА и ЯМУ, но присваивать им одинаковые номера.

Обратите также внимание на разницу в использовании некоторых глаголов. Это, например, СМОТРЕТЬ и ОСМОТРЕТЬ. Первый не относится к какому-либо предмету

конкретно и подразумевает, как бы оглядеться по сторонам, при этом перерисовывается экран и заново сканируется и отображается табло "В ПОЛЕ ВАШЕГО ЗРЕНИЯ". Глагол СМОТРЕТЬ может использоваться самостоятельно, без существительного. Другой глагол - ОСМОТРЕТЬ - самостоятельно использоваться не может и служит для получения каких-то дополнительных сведений о конкретном предмете, например, о ДВЕРИ, ПОЛЕ, ШКАТУЛКЕ, ЩЕПКАХ от неё, когда она будет разбита и т. д.

Различаются такие глаголы ОТКРЫТЬ и ОТПЕРЕТЬ. Учитывайте, что, например, ДВЕРЬ может быть закрыта, но не заперта на замок, а СЕЙФ закрыт, да ещё и заперт.

### **Таблица расположения объектов.**

Переменная О определяет начало этой таблицы. В ней находятся данные о текущем расположении 50 объектов. Для каждого объекта в любой момент времени здесь содержится число, определяющее номер локации, в которой в данный момент находится объект.

Например, для объекта с номером 1 (ПЛЕД) Вы можете узнать его расположение командой РЕЕК (О+1), а изменить его состояние - командой РОКЕ (О+1), N. Здесь "N" - новое место размещения объекта.

Если РЕЕК (О+Х)=99, то это означает, что объект находится у Вас в руках (или на Вас).

Если РЕЕК (О+Х)=0, это значит, что объект в данный момент не виден.

По ходу игры Бейсик-программа вносит изменения в таблицу расположения объектов.

Пример расшифровки логической конструкции:

```
IF РЕЕК (О+9)<>99 THEN ...
```

означает следующее:

ЕСЛИ У ВАС ПРИ СЕБЕ НЕТ ГОРЯЩЕГО ФАКЕЛА, ТО ...

### **Таблица состояния флагов.**

Некоторые объекты имеют одно и то же имя, например ФАКЕЛ. В то время, когда это просто ФАКЕЛ или МАЛЕНЬКИЙ ФАКЕЛ, то это объект номер 2. В то же время, если это ГОРЯЩИЙ ФАКЕЛ, то это уже объект номер 9. Кроме того, по ходу игры могут происходить разные события с предметами, не входящими в инвентарь. Это, например, отпирание и открывание дверей, фиксация того, что интересующее событие произошло и т.д. Все эти моменты отражаются в таблице флагов.

Переменная F определяет адрес ячейки, с которой начинается эта таблица. В ней возможно расположение данных о 100 объектах, причём некоторые флаги зарезервированы для особых случаев.

Определяется состояние флага предмета X командой РЕЕК (F+X), а изменяется командой РОКЕ (F+X), Z, где Z - новое состояние предмета.

По ходу игры Бейсик-программа вносит изменения в таблицу состояния флагов.

Флаг (F+0) содержит номер текущей локации - то есть место, где Вы сейчас находитесь. Расшифровка логических конструкций в программе будет выглядеть так. Например:

```
IF РЕЕК F=7 THEN ...
```

означает следующее:

ЕСЛИ ДЕЙСТВИЕ ПРОИСХОДИТ В ПОЛЕ ТО ...

Другой пример:

```
... РОКЕ F, 6 ...
```

означает:

... МЫ ОКАЗЫВАЕМСЯ В КОНЮШНЕ ...

Флаг (F+99) служит для отражения количества предметов (ИНВЕНТАРЯ), которые герой имеет при себе в данный момент.

Пример логической конструкции:

```
РОКЕ (F+99), РЕЕК (F+99)+1
```

означает ни что иное, как то, что Вы берёте объект в руки. Другой пример:

```
IF РЕЕК (F+99)=MAX THEN ...
```

означает:

ЕСЛИ У ВАС В РУКАХ МАКСИМАЛЬНОЕ КОЛИЧЕСТВО ПРЕДМЕТОВ, ТО ...

Когда игра начинается, все флаги, за исключением (F+0) имеют нулевое значение. В процессе игры, когда герой надевает ПЛЕД, флагу (F+1) присваивается единица, а когда он зажигает ФАКЕЛ, единица присваивается флагу (F+2) и т. д.

Ниже приводится таблица флагов, использующихся в игре. Значение 1 присваивается соответствующему флагу, если выполняется действие, показанное в таблице.

| ТАБЛИЦА СОСТОЯНИЯ ФЛАГОВ                       |
|------------------------------------------------|
| F+0=NO - НОМЕР ТЕКУЩЕЙ ЛОКАЦИИ                 |
| F+1=1, если ПЛЕД НАДЕТ НА ПЛЕЧИ                |
| F+2=1, если ФАКЕЛ ЗАЖЖЁН                       |
| F+3=1, если БУФЕТ ОТКРЫТ                       |
| F+4=1, если ДВЕРЬ ОТКРЫТА                      |
| F+5=1, если ВЫКОПАНА ЯМА В ПОЛЕ                |
| F+6=1, если ВЫКОПАНА ЯМА В САДУ                |
| F+7=1, если ВОРОТА ОТКРЫТЫ                     |
| F+8=1, если СЕЙФ НЕ ЗАПЕРТ                     |
| F+9=1, если ШКАТУЛКА НАЙДЕНА                   |
| F+10=1, если КЛЮЧ НАЙДЕН                       |
| F+11=1, если БУФЕТ ОТКРЫВАЛСЯ<br>ХОТЯ БЫ 1 РАЗ |
| F+99=NO - ЧИСЛО ВЗЯТЫХ ОБЪЕКТОВ                |

Если с этим объектом происходит противоположное действие (дверь опять закрывается), то флаг должен обнуляться. Однако могут быть ситуации, когда состояние одного объекта оценивается несколькими флагами. Например, если ОТКРЫТЬ БУФЕТ в первый раз, то установится в единицу два флага: F+3 и F+11. Если теперь ЗАКРЫТЬ БУФЕТ, то сбросится флаг F+3, а флаг F+11 так и останется в состоянии 1.

Теперь можно проследить, как в игре формируются более сложные действия. Ситуация, когда зажигается факел, например, моделируется следующим образом:

POKE (F+2), 1: POKE (0+2)=0: POKE (0+9), 99

То есть, во-первых, устанавливается флаг зажжённого факела, во-вторых, исчезает объект под названием МАЛЕНЬКИЙ ФАКЕЛ, в третьих - у Вас в руках появляется новый объект: ГОРЯЩИЙ ФАКЕЛ.

Ситуация, когда Вы кладёте горящий факел, например, в саду (при этом он гаснет), будет выглядеть так:

POKE (0+9), 0: POKE (F+2), 0: POKE (0+2), 15:

Сначала исчезает объект ГОРЯЩИЙ ФАКЕЛ и обнуляется его флаг, а затем в САДУ появляется объект МАЛЕНЬКИЙ ФАКЕЛ.

Теперь, когда Вы имеете представление о том, как моделируются те или иные логические конструкции, можно проследить действие программы сначала.

### Демонстрационная программа.

В строке 100 начинается прорисовка игрового экрана. На нём изображён только текст, так задумано в концепции отладочной системы ABS. Однако ничего не мешает Вам изменить 7000-ные строки так, чтобы, например, запускался декомпрессор, выводя в верхней трети (или двух третях) экрана рисунок локации. Эти экраны можно подготовить, например, при помощи компрессора, опубликованного в "ZX-РЕВЮ" №3-4, стр. 61-63.

Строка 102 выполняет подпрограмму из большого блока строк, который начинается со строки 7000. Описание локации вводится обычным оператором PRINT в строках 7000 + номер локации, умноженный на 10. Это происходит в зависимости от номера локации РЕЕК F. Для АМБАРА (1 ЛОКАЦИЯ) будет GO SUB 7000, для ЧЕРДАКА (2 ЛОКАЦИЯ) - GO SUB 7010 и т. д. Длина описания каждой локации - не более 10 БЕЙСИК-строк. Каждое описание должно заканчиваться командой RETURN.

Строка 104 (вместе со строкой 7110) обеспечивают реакцию программы на событие, когда Вы попадаете в погреб без ГОРЯЩЕГО ФАКЕЛА.

Строки 105-129 обеспечивают проверку того, что можно увидеть в этой локации. Если

что-то есть, то эта информация также выводится на экран в виде табло: "В ПОЛЕ ВАШЕГО ЗРЕНИЯ". Блок в машинных кодах сам выполняет проверку того, какие объекты следует объявить в этих ситуациях, а начать выполнять процедуры Бейсика. Вместо слов можно вводить изображение соответствующих предметов при помощи символов UDG-графики, печатая их оператором PRINT прямо поверх картинки.

Процедура RANDOMIZE USR SS (строка 195) проверяет необходимость и осуществляет скроллинг экрана и должна выполняться всякий раз, после того, как происходит перерисовка экрана.

Далее, в строке 200, запускается большая процедура в кодах, обеспечивающая сканирование клавиатуры и печать на экране вводимых команд, а также их первичный анализ - "Анализатор текста", о котором уже говорилось выше.

Результатом работы этой процедуры являются значения двух ячеек памяти: 64114 и 64115. В первой из них содержится номер использованного глагола, во второй - существительного (или направления перемещения). Они передаются в Бейсик-переменные в строках 202 и 204. Номер глагола будет теперь VB, а номер существительного - NO.

Строка 208 разрешает при отсутствии ГОРЯЩЕГО ФАКЕЛА в ПОГРЕБЕ только следующие действия: ИДТИ, ЗАЖЕЧЬ, ПОГАСИТЬ, ПОДНЯТЬСЯ, СТОП, ИНВЕНТАРЬ, СОХРАНИТЬ и ЗАГРУЗИТЬ.

Если в результате работы процедуры ввода команды не зафиксирован ни один из допустимых глаголов, то значение ячейки 64114 будет равно 200. Аналогично, если в команде не зафиксировано ни одно из заданных существительных (или направления перемещения), то значение ячейки 64115 будет 201. Как поступить в этих случаях - решать Вам. В демонстрационной программе в этом случае строка 228 адресует на строку 45, где выводиться сообщение "НЕ ПОЛУЧАЕТСЯ".

Если глагол и существительное идентифицированы, то дальнейшее распределение производится в строке 235, в зависимости от значения глагола: происходит переход на подпрограммы обработки глаголов, расположенные в большом блоке строк, который начинается со строки 1000.

Поскольку наиболее вероятно, что Вы будете оформлять программу с разделённым экраном, нужно было бы вводить проверку на необходимость скроллинга перед тем, как печатать сообщение от программы на экране. Чтобы избежать такой проверки, мы в демонстрационной программе оформляем реакцию программы в виде символьных переменных R\$(1), R\$(2), R\$(3), R\$(4), которые печатаются с помощью главной печатающей процедуры (строка 50).

### **Обработка глаголов.**

Каждая подпрограмма обработки глаголов должна иметь не более 100 Бейсик-строк. Номер первой строки определяется по номеру глагола: 1000 + номер глагола, умноженный на 100. Так обеспечивается переход, например в случае ИДТИ - на строку 1000, в случае ВЗЯТЬ - на строку 1100 и т. д.

Вот полный перечень строк для соответствующих глаголов.

|                 |      |
|-----------------|------|
| ИДТИ.....       | 1000 |
| ВЗЯТЬ.....      | 1100 |
| ПОЛОЖИТЬ.....   | 1200 |
| ОСМОТРЕТЬ.....  | 1300 |
| СМОТРЕТЬ.....   | 1400 |
| ИНВЕНТАРЬ.....  | 1500 |
| СТОП.....       | 1600 |
| СОХРАНИТЬ.....  | 1700 |
| ЗАГРУЗИТЬ.....  | 1800 |
| ОТКРЫТЬ.....    | 1900 |
| ОТПЕРЕТЬ.....   | 2000 |
| КОПАТЬ.....     | 2100 |
| ЗАЖЕЧЬ.....     | 2200 |
| ПОГАСИТЬ.....   | 2300 |
| ЛЕЗТЬ.....      | 2400 |
| СПУСТИТЬСЯ..... | 2500 |



|                 |      |
|-----------------|------|
| ЗАКРЫТЬ.....    | 2600 |
| ШВЫРНУТЬ.....   | 2700 |
| ЗАКОПАТЬ.....   | 2800 |
| ЗАПЕРЕТЬ.....   | 2900 |
| НАДЕТЬ.....     | 3000 |
| СНЯТЬ.....      | 3100 |
| ПРЫГАТЬ.....    | 3200 |
| ИСПОЛЬЗОВАТЬ... | 3300 |

Специфичной является подпрограмма перемещения, расположенная со строки 1000. Мы попадаем на неё независимо от направления, во всех случаях предусмотренного перемещения. В строке 1002 происходит изменение номера слова, определяющего переменные. Так, для СЕВЕР вместо 10 получается значение 1, для ЮГ - вместо 11 - значение 2 и т. д. Это необходимо для корректного дальнейшего выполнения программы в кодах.

В строке 1005 выполняется следующая проверка. Если Вы находитесь перед дверью и идёте на восток или находитесь в прихожей и идёте на запад и в обоих этих случаях дверь закрыта, то вывод текста "ДВЕРЬ ЗАКРЫТА".

Строка 1010. Аналогичная проверка для ворот.

Строка 1020. Если Вы пытаетесь идти в поле, в котором выкопана яма или в саду, в котором выкопана яма, то Вы в неё падаете. В этом случае игра для Вас заканчивается.

В строке 1090 выполняется подпрограмма в кодах, которая выполняет перемещение в другую локацию согласно таблице допустимых перемещений. Входным параметром для этой процедуры является значение содержимого ячейки 64115 (оно было занесено туда в строке 1002). Другим параметром для этой процедуры является значение флага (F+0), то есть текущая локация. Результат работы этой процедуры возвращается также в ячейку 64115, теперь там значение, соответствующее новой локации. Если же там ноль, то это значит, что в направлении, которое Вы задали, перехода нет. Этот контроль производится в строке 1092. В строке 1093 происходит изменение флага текущей локации - Вашего положения. Строка 1094 возвращает на начало программы с новыми условиями - в новой локации.

Обратите внимание, что заикливание происходит тремя способами. Первый - если команда не воспринимается - на строку 45 с выводом сообщения "НЕ ПОЛУЧАЕТСЯ". Второй - на строку 50 - без изменения верхней части экрана - то есть без изменения локации и отображения предметов, находящихся в поле Вашего зрения. Третий способ - переход на строку 100 (как при старте игры) - это перерисовка всего экрана (переход в новую локацию или команда СМОТРЕТЬ). При этом также изменяется и табло "В ПОЛЕ ВАШЕГО ЗРЕНИЯ:".

### **Примеры расшифровки подпрограмм.**

Теперь, для примера, мы дадим расшифровку реакции программы на некоторые Ваши действия. Потом Вы должны будете продолжать эти рассуждения для всех строк программы, если хотите в деталях разобраться в том, что происходит в разных ситуациях.

Итак, например, строка 1100 - сюда мы попадаем в случае обработки глагола ВЗЯТЬ.

Строка 1100 накладывает ограничение на попытку взять объект, не являющийся инвентарем (то есть имеющий больший номер).

Строка 1105 недоумевает по поводу попытки взять объект, который уже находится у Вас в руках.

Строка 1110 удивляется, если Вы хотите взять инвентарь, который в принципе существует, но его сейчас нет в этой локации.

Строка 1115 возмущается, если Вы хотите поднять больше объектов, чем это разрешено программой.

Строка 1120 радуется, если Вы берёте в руки АЛМАЗ - это конечная цель миссии.

Строка 1125 соглашается с Вами, если Вы берёте остальные объекты инвентаря, которые не были зафиксированы в предыдущих строках.

Крупным блоком является блок, обрабатывающий глагол ОСМОТРЕТЬ (ПРОВЕРИТЬ). Здесь проверка разветвляется на несколько ветвей для ускорения анализа.

Строка 1300 разделяет проверку на относящуюся к инвентарю и прочим объектам.

Строка 1305 отражает закрытое, а строка 1310 - открытое (в случае выключенного флага двери) состояние двери. Строка 1312 закидывает программу.

Строка 1315 аналогично двери перехватывает проверку ворот.

Строки 1317 и 1320 аналогично двери отражают открытое или закрытое состояние ворот (в зависимости от флага ворот). После чего строка 1322 закидывает программу.

Строка 1325 предупреждает о выкопанной яме, если Вы находитесь в саду и осматриваете САД или ЛУЖАЙКУ и при этом яма выкопана (флаг ямы в саду - установлен) или Вы находитесь в поле с выкопанной ямой и осматриваете ПОЛЕ.

Строка 1330 отражает результаты проверки, если Вы находитесь в саду и осматриваете сад или лужайку, но яма не выкопана (этот случай был перехвачен предыдущей строкой) или в поле, в котором тоже нет ямы.

Строка 1335 показывает результат проверки ЯМЫ, если Вы находитесь в поле, в котором выкопана яма или в саду, в котором выкопана яма.

Строка 1340 - результат осмотра ПЯТНА, если Вы находитесь в саду и яма закопана.

Строка 1345 - это результат всех остальных проверок.

Строка 1350 перехватывает проверки тех предметов, которых нет в этой локации и нет у Вас.

Строка 1355 показывает результат осмотра ПЛЕДА или ПЛЕДА НАДЕТОГО НА ПЛЕЧИ.

Строка 1360 - результат осмотра ФАКЕЛА или ГОРЯЩЕГО ФАКЕЛА.

Строка 1365 - осмотр ШКАТУЛКИ.

Строка 1370 - осмотр АЛМАЗА.

Строка 1372 - осмотр ЩЕПОК.

Строка 1375 - все прочие случаи, не предусмотренные программой.

Подробную расшифровку некоторых строк мы дали только для примера. С остальными Вы теперь сможете разобраться самостоятельно.

### **Кодовый блок программы.**

Эта информация представляет определённый интерес для "хакеров". Речь идёт о блоке кодов "system" CODE. В том случае, если Вы не изменяли данные, заданные в Листингах 1 и 2, адрес загрузки блока кодов - переменная SAVE - будет равен 63400. Этот блок занимает пространство до 65368 (начало символов UDG - графики, стандартно расположенных здесь). Если Вы увеличили число слов, распознаваемых программой, то адрес загрузки Вашего блока отодвинется в сторону младших адресов. Здесь надо проконтролировать, чтобы он не перекрывался с загружаемым символьным набором, который сейчас расположен с адреса 50000 и имеет длину 768 байт (см. Листинг\_3) - может этот адрес и изменить.

С адреса загрузки SAVE блока "system" CODE расположены те данные, которые были заданы в строках DATA программы ABS (вместе с Листингом\_2). Здесь расположена таблица переходов между локациями, как она была задана в 3000-х строках, группы четвёрок символов, заданных для идентификации слов и т.д.

Начиная с адреса 64000, блоки, расположенные здесь, имеют постоянные адреса и не изменяют своего положения.

Ячейки 64114 и 64115 являются рабочими ячейками, в которых хранятся выходные параметры анализатора текста и подпрограмм перемещения.

С адреса 64116 расположена таблица данных, передаваемых через эту область из программы ABS в демонстрационную программу (см. строки 9957-9960 Листинга\_4).

С адреса 64130 зарезервирована область размером 150 байт. Она служит для временного сохранения текущего состояния игры, которое определяется таблицей состояния флагов (100 байтов) и таблицей текущего расположения объектов (50 байтов). Сами эти таблицы расположены с адреса 64230 (этому числу равна переменная F, а переменная O на 99 байтов больше, то есть 64379). Ячейка 64130 - первая ячейка сохраняемой области - соответствует значению флага (F+0), то есть текущей локации. Это число не может быть равно нулю (не бывает нулевой локации). При старте демонстрационной программы, в эту ячейку заносится ноль (строка 9961 Листинга\_4),

предохраняя, таким образом, от сбоя при попытке загрузить состояние, если оно не было прежде сохранено. Эту проверку выполняет строка 1814 Листинга\_5.

С адреса 64430 расположены процедуры в кодах.

Процедура по адресу 64440 проверяет необходимость и выполняет скроллинг экрана, обеспечивая в заданных пределах сохранение на экране Ваших команд и ответов программы на них.

Процедура по адресу 64495 (строка 200) - анализатор текста.

Процедура по адресу 64930 сканирует таблицу расположения объектов, выявляя те предметы инвентаря, которые находятся в текущей локации и видны (строка 105 Листинг\_4).

Процедура по адресу 64968 (строка 1502) выполняет аналогичную операцию для того инвентаря, который находится у Вас в руках (или на Вас).

Процедура по адресу 65005 (строка 1090) - процедура перемещения.

Процедура по адресу 65058 (строка 9980 Листинг\_4) выполняет инициализирующие действия при старте программы, в частности, обнуляет 150-байтовую область с адреса 64280 - таблицу флагов, заносит значение стартовой локации в таблицу флагов (F+0) а также исходное расположение объектов в таблице текущего расположения.

### **Создание собственной игры.**

Когда Вы отладите программу и демонстрация заработает, перед Вами встанет следующий вопрос: "А как мне написать свою игру?"

Во-первых, прежде чем приступить к созданию собственной игры, прогоните демонстрацию и внимательно изучите все детали. Во-вторых, даже и не подходите к компьютеру, пока тщательно не продумаете сценарий своей игры на бумаге. Вы должны, по крайней мере, разработать карту и разместить на ней все участвующие в игре объекты с указанием, к какому типу они относятся.

Сначала спланируйте весь сценарий просто на бумаге, после чего можете приступить к работе с головной программой (Листинги 1 и 2, объединённые при помощи MERGE, иными словами, то, что теперь записано на ленте \_A). Ещё раз отметим, что самый простой способ понять, что и как надо делать - это максимально подробно проследить логику работы демонстрационной программы.

### **С чего практически начать?**

Посмотрите на Листинг\_4 - кроме блока строк с 7000 - это почти готовый дебют для новой игры. Надо лишь добавить к нему блоки строк с 1700 и с 1800 - СОХРАНЕНИЕ и ЗАГРУЗКА, да ещё со строки 1600 - КОНЕЦ и получится прекрасная заготовка для Вашей собственной игры. Осталось только добавить (или изменить имеющиеся) строки, отвечающие за закрытые или запертые двери, освещение, монстров и разные ловушки, которые будут в Вашей игре. Эти изменения надо ввести в блок перемещения - со строки 1000.

### **Скорость работы и объём памяти.**

При создании собственной игры размер занимаемой памяти может стать критичным. Место, отведённое для Бейсик - программы можно расширить, отодвинув символьный набор в сторону старших адресов, но так, чтобы он не наложился на блок кодов "system" CODE, об этом уже говорилось несколько выше. Практически для этого надо изменить параметры оператора CLEAR и изменить адрес загрузки символьного набора "chr" CODE. Надо также в соответствии с новым адресом загрузки символьного набора изменить значение CHARS, задаваемое в строке 20 Листинг\_3.

Другим решением бывает замена часто встречающихся чисел переменными. Например, 0 и 1 постоянно встречаются в листинге. Поэтому, например, если присвоить: LET M=0: LET J=1 и по всему тексту программы заменить 0 и 1 на M и J Вы получите существенную экономию памяти. Числа 0 и 1 могут заменяться иначе. 0 - NOT PI, а 1 - SGN PI. В то же время, если этот подход довести до неразумных пределов, он может сказаться на скорости работы программы.

### **Структура подпрограмм.**

Подпрограмма "ПРОВЕРИТЬ" или "ОСМОТРЕТЬ" по-видимому, у Вас будет одной из крупнейших подпрограмм, но можно ускорить время её работы, если разбить её на несколько блоков. Так, например, это делает строка 1300 в демонстрационной программе, разветвляя исполнение в зависимости от номера объекта по разным ветвям. Но и дальнейшая разбивка этих ветвей тоже возможна.

# FORUM

## Проблемы ELITE

Внимание, друзья! Мы представляем Вам нашего корреспондента из Альметьевска - Абдрахманова Руслана. Впервые мы приводим свидетельство пилота, который лично видел РАККСЛУ.

А начиналось все с "ZX-РЕВЮ". Руслан начал выписывать его только в 93-м году и журнал стал как бы "ключом" к тайнам "Spessy", позволившим заглянуть дальше, чем LOAD и Press any key.

КОРР: ... До недавнего времени программа ELITE меня не интересовала: паутинная графика, никакого звука, непонятный сюжет. Но после публикации "The Dark Wheel" все переменялось. Я стал отчаянно торговать, воевать, искать приключения. После рейтинга Competent дело застопорилось. Но тут вмешался случай. По воле провидения я нашел планету RAXXLA!!! И вот, как это произошло.

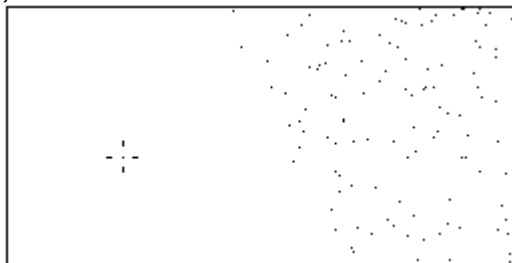
... На последние деньги я купил галактический гиперпривод и стартовал с планеты LAVE. На экране пошли концентрические круги и вдруг в блоке питания что-то щелкнуло, но программа не сбросилась, переход продолжался. Очутился я у планеты URBA. Правление - диктатура. Желая посмотреть карту, я нажал "O" и тут увидел, что нахожусь в галактике без номера, а около меня нет ни одной планеты, кроме RAXXLA! Представьте себе мои чувства?!

МФК: Это представить невозможно, это надо пережить. Кстати, уважаемый Руслан, в пылу переживаний Вы могли не заметить, что кодов для планеты URBA тоже нет в знакогенераторе игры, как и для Ракксла. Это является косвенным подтверждением того, что могут существовать планеты с именами, состоящими не только из стандартных слогов. Очень хорошо, что Вы запомнили название этой планеты,

КОРР: ... Я сразу понял, что надо записать результат, но для этого пришлось пробиваться на станцию планеты URBA. У нее был высокий технологический уровень -10, но, тем не менее, космос кишел пиратами. Меня бесконечно атаковали "Фер-де-Лэнсы", хотя статус у меня был чистый. Кроме них там были "Кобры" и астероиды.

С трудом пробившись на станцию, я обнаружил, что к продаже там имелись только золото и радиоактивные материалы, причем все это по очень высоким ценам. Я конечно записал свое состояние на ленту, но, как потом выяснилось, безрезультатно. Этот блок оказался длиной всего 99 байтов и никуда не пошел.

А на Раккслу меня не пустили, выдав сообщение "Hyperspace, Range?", хотя топлива было на 7 световых лет. Других планет рядом не было. На карте галактики я обнаружил, что она заполнена всего наполовину, вот так:



Крестиком отмечено место, где был я.

Остается добавить, что я пользуюсь версией M128, а гипердрайв был у меня установлен перед отлетом на планету DIZO.

Я не утверждаю, что всем надо взламывать свои блоки питания, но может быть мой случай кому-нибудь поможет.

ИФК: Спасибо, Руслан за интересный отчет. Нам кажется, что Вы грамотно и четко держали себя в этой экстремальной ситуации и сделали все, что могли. Наконец-то у нас появилось хоть косвенное, но свидетельство. Единственное, о чем стоит пожалеть, так это о том, что Вы не прислали распечатку отгруженного блока. Хотя в нем всего лишь 99 байтов, но он, тем не менее, мог бы быть весьма интересен. По-крайней мере, по содержащимся в

нем кодам можно было бы попробовать восстановить эту неизвестную галактику и попробовать по ней полетать.

Будет очень обидно, если у Вас этот блок не сохранился.

КОРР: У меня есть просьба к экспертам. Я был бы очень рад прочитать что-то о программах "DRILLER", "TAI-PAN", "READ HEAD", "NORTH & SOUTH". Эти игры многоуровневые и разобраться с ними начинающему пока не под силу.

ИФК: Очень многие, желающие попробовать свои силы, спрашивают нас, какие игры мы могли бы порекомендовать для исследования. Просьба Руслана - достойный ответ на такие письма.

\* \* \*

Поклонник незабвенной ELITE, пилот Дм. Егоров из Уфы (DANGEROUS) подсказывает тем, кто имеет статус FUGITIVE и не может законным образом совершить посадку на станцию. Став невидимым с помощью CLOAKING DEVICE, Вы сможете обмануть офицеров службы безопасности.

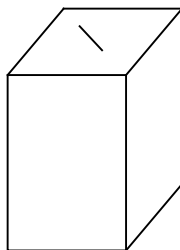
\* \* \*

Нам пишет Владимир Бакаев из г. Челябинска.

Здравствуйте!

Вчера впервые прочитал выпуски "ZX-РЕВЮ" 1991 и 1992 г. и открыл для себя, что "Элитой" увлекся не я один. Мой рейтинг "Competent", стаж - 5 месяцев. Прочитав о том, что Вы просите писать обо всем, что встречается в этой игре очень редко, спешу сообщить об одном происшествии.

Решив как-то начать с нуля, я загрузил свой корабль на 100 Cr и с LAVE отправился на ZAONCE (извините, эту планету пишу по памяти). После J-прыжка поймал сигнал станции и стал к ней приближаться. Вскоре она появилась на экране. Но вид был необычным. Обычно это сначала точка, которая, по мере приближения становится вращающимся додекаэдром. А это была вертикально стоящая черточка. Я начал думать, что это корабль поколений, т.к. размеры его огромны. Но сигнал "S" шел от него. Подлетев поближе, я увидел, что "это" похоже на брусок.



На верхней грани по диагонали - линия. Подлетев очень близко, я решил облететь вокруг. Но не тут-то было. Корабль мой был разрушен (никаких сообщений).

Опять вылетев с LAVE и подлетев к "ZAONCE", увидел ту же картину. Подлетев близко со стороны верхней грани и не рискуя опять взорваться, в последний момент решил влететь в черточку посреди нее. (Была надежда, что это вход в станцию). И это мне (я надеюсь) удалось. Но так как маневр не был подготовлен, то, задевая стенки тоннеля, корабль, истощив энергоблоки, был разрушен.

Больше эта станция мне не встречалась. Буду рад, если кто-нибудь встречался с ней и знает, что это такое.

\* \* \*

В "ZX-РЕВЮ" N5-6 за этот год наш читатель из Москвы Т.М. Канразе попросил осветить недавно появившуюся программу ELITE-2.

Ему не удалось установить в ней существенных отличий и она очень напоминает

"версию-3" по классификации (N3-4,1993). Для прояснения ситуации мы адресовали этот вопрос к нашим читателям. Ответ последовал незамедлительно, тоже из Москвы, причем кажется, что из невинного вопроса нашего читателя рождается новое и интересное направление исследований. Вот, что пишет Алексей Лапшин (AZURE soft.)

KOPP: Недавно, копаясь в новых демонстрационных программах, я обнаружил несколько выпусков новосибирского электронного журнала TELESHOW. В первом же номере я нашел нечто сенсационное. Мне удалось вытащить эту статью с диска в текстовый редактор, не трогая содержания. Я лишь убрал опечатки и характерные ошибки. На авторские права я, конечно же не претендую, но не послать ее Вам было невозможно. Привожу ее целиком.

ИФК: Мы свидетельствуем свое глубокое уважение редакции и создателям TELESHOW и с радостью бы вошли в контакт с целью установления взаимовыгодного сотрудничества, но, не зная адреса редакции, обречены на ожидание возможного контакта. Вдруг до них дойдет это наше обращение.

Приведенная ниже информация является фактически рекламой программы ELITE-2, причем не просто рекламой, а лучшей ее формой - пропагандой. Фирма, подготовившая ее, отлично знает свое дело, жаль только нет координат, к кому следует обратиться за приобретением новой игры.

Мы окажем всемерную помощь в рекламе и распространении игры тем авторам, которые подготовили этот интересный материал. Обращайтесь, Вы не будете разочарованы. У нас созданы дистрибуторские сети и они могут Вам помочь.

#### \*\*\* ELITE 2 (48 K) \*\*\*

Вы можете продолжить свою боевую и деловую карьеру, загрузив запись состояния, полученную в ELITE, но можете начать все сначала, с выдачи пилотской лицензии на планете LAVE в галактике 1. В любом случае первоначально у Вас имеется тот же корабль Кобра-МК3. Если управление Вам незнакомо, воспользуйтесь электронным справочником. В этом рекламном листе приводятся вкратце лишь отличия новой версии.

Действие игры ELITE-2 происходит в 3192 году. Галактическая Федерация по-прежнему использует высоконадежные орбитальные станции типа "Coriolis", но теперь они обязательно оборудованы усовершенствованными средствами наблюдения и опознавания. это означает, что в виду базы можно беспрепятственно использовать любые виды оружия для отражения атаки врагов Федерации: Таргонов, пиратов, нарушителей. На базах размещаются любые корабли, кроме Таргонов и пиратов и, зависнув недалеко от входа, можно наблюдать, как кто-нибудь покидает станцию и уходит в гиперпространство. Кроме того, "Coriolis" обязательно несет флот полицейских кораблей типа Viper, которые в любой момент готовы обрушиться на голову агрессоров, атаковавших станцию или мирный корабль.

В ELITE-2 получено значительно большее приближение к фантастической реальности этой игры. В отличие от ELITE, в новой версии возможно воочию наблюдать, как другие корабли взаимо- и противо- действуют, выполняя сложные маневры в пространстве, используя лазеры, ракеты и прочее оборудование, подобно тому, как это делаете Вы сами. На первых порах, особенно при полетах на неблагоприятные планеты, анархические и Феодалные особенно, имеет смысл искать в пространстве союзников, и ни в коем случае не отказываться от полицейского эскорта. Следует также с умом использовать разногласия, возникающие между различными пиратскими кланами, и жадность охотников, легко отвлекающихся на собирательство обломков и контейнеров - для того, чтобы оторваться от погони.

Применение устройства невидимости теперь не является панацеей - так же, как это делаете Вы сами, другие пилоты пытаются поразить невидимого противника, используя локатор и электронную систему наведения ракет, правда, вероятность попадания по

невидимке значительно ниже.

В пространстве Вы можете встретить все старые корабли: Asp, Fer de Lance, Cobra, Pithon, Viper, Sidewinder, Adder, Krait, Thargoid. Кроме того, начали выпускаться новые корабли классов Gecko и Mamba, а Таргоны приняли на вооружение новую разновидность корабля вторжения - с шестью гранями, получивший среди охотников условное название Targoid-6.

Станции во время технического обслуживания в обязательном порядке снабжает все корабли новым оборудованием. Все корабли класса Cobra, в том числе и Ваш, получает новый локатор (включается по CAPS+L), и модифицированный компас (CAPS+C). Новый локатор наиболее полезен в ближнем бою, и прекрасно дополняет старый. В межзвездном пространстве новый компас автоматически переключается в режим гирокомпаса и показывает стороны света N-S, W-E, соответствующие северу-югу, западу-востоку в карте галактики. Гирокомпас позволяет ориентироваться в межзвездном пространстве и совершать перелеты от одной планетной системы к другой без гиперпереходов, на J-двигателе.

В пределах планетной системы новый компас в режиме 1 работает, как и прежде, показывая направление на планету, а в режиме 2 начинает попеременно показывать направление на Солнце и на планету. В частности, это очень удобно и полезно для быстрого поиска пояса астероидов, который обычно начинается на достаточно большом удалении от Солнца.

Технологический прогресс неостановим и на планетах уровня-15 Вы уже можете купить Super-Laser (33000 Cr). Кроме того, на экспериментальных двойных станциях, расположенных вблизи солнц в планетарных системах, лежащих на главной диагонали, возможно приобрести новый корабль "Кобра МК4", но об этом будет сказано ниже, в разделе, посвященном новым миссиям.

Произошли кое-какие изменения во внутренней и в экономической политике. Ввиду готовящегося крупномасштабного вторжения Таргонов, Федерация решила легализовать торговлю оружием - Firearms перестали быть запрещенным товаром.

Полицейские корабли - неприкосновенны. Уничтожение полицейского равно как и уничтожение орбитальной станции, принадлежащей Федерации, превращает Вас в нарушителя номер 1. В галактиках 2-8 таких нарушителей встречает вблизи планеты полицейский кордон.

Как и прежде Вам могут быть выданы специальные миссии: по спасению беженцев со станций, солнце которых превращается в новую, по уничтожению корабля-невидимки и по уничтожению базы, захваченной Таргонами. Но, кроме этих трех миссий (которые в новой игре могут повторяться неоднократно), добавлено несколько новых.

Миссия-4 - "Mineral Life". В планетных системах на главной диагонали галактики в поясе астероидов Вы можете, разрабатывая астероиды, подобрать необычные объекты - "минеральную жизнь". Каждая тонна породы, содержащей "живые кристаллы", может быть продана за 1000 Cr. Однако, существуют особые условия для сохранения жизнеспособности этих привередливых космических овец. Их должно быть не менее 3-х тонн и не более 63-х тонн, потому их нельзя продать в количестве 1 или 2 тонны одному покупателю.

В момент выполнения гиперперехода они питаются драгоценными камешками Вашего сейфа, съедая по 1 грамму за каждую тонну живого груза. И если скольких-то каратов драгоценного сена им не хватит, в Ваших отсеках окажется обычная дешевая порода. А в случае гибели этой минеральной жизни (по любой причине) отсеки вашего корабля окажутся зараженными кристаллическим лишаем, который уже не даст Вам возможность вновь превратить свой корабль в летающую ферму, пока отсеки не будут очищены при выполнении миссии по спасению беженцев или при доставке Desent'a.

Миссия 5 - "Alien Computer".

Уничтожив корабль Таргонов, можно подобрать контейнер, содержащий это устройство, позволяющее запускать имеющиеся на борту Alien Items вместо ракет. Для запуска используются клавиши "D+F", но этот вид оружия эффективен только против самих же Таргонов.



Следует добавить, что Alien Computer обеспечивает еще и автоматический возврат запущенных Alien Items на борт корабля после уничтожения цели.

Миссия 6 - "Кобра МК4". В планетных системах на главной диагонали галактики вблизи Солнц расположены спаренные станции "Кориолис" - это экспериментальные лаборатории Федерации. При подлете к ним, Вам предлагают купить корабль класса "Кобра МК-4". Однако, добраться сюда не просто. Многочисленные корабли, а в основном это Asp'ы и Gecko ведут здесь ожесточенные сражения и без эскорта Alien Items делать там нечего.

От сопровождения следует вовремя избавиться, чтобы "двойные базы" не приняли Вас за враждебную флотилию. Для покупки нового корабля требуется 250000 Cr, но он стоит этого! Имея Кобру МК4, Вы можете купить до 4-х устройств сразу. Причем покупка четырех Large Cargo Bays увеличивает грузоподъемность до 80 тонн, а покупка нескольких Energy Units дает соответствующий рост скорости подзарядки энергетических отсеков. На Кобре МК4 возможен перелет в любую галактику - достаточно нажать номер галактики от 1 до 8, пока идет предстартовый отсчет. Лазеры на новой Кобре установлены на поворотном лафете, на "горбу" корабля. В любой момент Вы можете повернуть лафет по часовой стрелке (CAPS + A) и поставить на нужный экран нужный Вам лазер.

При нажатии клавиши "J" даже и при наличии поблизости больших масс достигается значительное увеличение скорости и маневренности Кобры МК4.

Миссия-7 - "Space War". В течение более, чем столетия Таргоны готовились к войне с цивилизованными галактиками. Их эскадры бороздили межзвездное пространство, и лишь изредка их корабли появлялись на границах планетных систем. Совсем недавно армада кораблей вторжения вновь пыталась захватить орбитальную станцию федерации, но благодаря отважному командеру база была взорвана и угроза нашествия приостановлена.

Но вот наступил час Икс и Таргоны начинают крупномасштабное вторжение в одну из галактик. Каждый раз, когда Вы получаете сообщение "MAYDAY! Galactic N Invaded!", область, захваченная Таргонами, увеличивается. Планетарные системы, находящиеся в состоянии активных боевых действий, обозначаются как RESISTING. Системы, в которых сопротивление подавлено, а орбитальные станции захвачены Таргонами, отмечены, как OCCUPIED. На оккупированных планетах флот Таргонов организует кордон при полёте к планетам и гиперперелёты между охваченными огнём планетами становятся опасными. Межзвёздное пространство заполнено эскадрами Таргонов, перехватывающими корабли федерации.

Вы можете остаться посторонним наблюдателем, но если Вы настоящий патриот Федерации, то непременно пожелаете вступить в ряды её добровольных защитников.

Сделать это можно на любой воюющей, но ещё не оккупированной планете. Вам присваивается звание - лейтенант, и выдаётся первая миссия - эвакуация гражданских лиц, раненых и беженцев из воюющих (RESISTING) систем. После того, как командование решит, что Вы справляетесь с этим заданием нормально, оно присваивает Вам звание - капитан и даёт задание по доставке оружия и продовольствия на воюющие планеты. Исполняя эти первые две миссии, Вы совершаете полёты между воюющими и невоюющими системами. Вам достаточно часто будут встречаться как одиночные Таргоны, так и целые флотилии перехватчиков. Это обеспечит достаточную практику для выполнения последующих миссий.

Когда, наконец, Вы получите звание - майор и задание по доставке десанта на оккупированные территории, Вы должны быть уже достаточно опытным бойцом, чтобы преодолеть кордон из кораблей Таргонов и прорваться к захваченной ими базе. Взяв на борт десантный корабль типа Sidewinder, Вы должны будете добраться до базы Таргонов и высадить десант. При этом Вы не сможете использовать Alien Computer, т.к. грузовой отсек занят десантным кораблём. Не сможете использовать и ракеты, вместо них запускается десантный корабль.

Запуск десанта следует выполнять с как можно более близкого расстояния к базе, чтобы до телепортации через силовое поле станции его не успели расстрелять Таргоны, обороняющие базу. Сразу после успешной высадки десанта Вы можете сами телепортироваться на станцию, включив Docking Computer - база уже контролируется

силами федерации. Чем глубже в тыл Таргонов Вам удастся высадить десант, тем большее количество оккупированных планет выйдут из-под контроля Таргонов.

Последняя, предлагаемая Вам вместе со званием полковник, миссия состоит в поиске командной базы Таргонов, расположенной где-то в глубинах космоса. Штаб сможет Вам сообщить лишь приблизительное название планетной системы, на расстоянии в несколько световых лет от которой находится эта база. Такое же задание даётся всем пилотам, поступившим на службу добровольно. В принципе, исход войны уже предрешён, цивилизованные галактики снова в безопасности.

Следует отметить, что добровольцы воюют на своих собственных кораблях, которые вооружаются на собственные сбережения. Но при этом они в значительной степени остаются самостоятельными в выборе маршрутов движения, действиях и т.д.

\* \* \*

КОРР: И, в завершение, несколько комментариев от новосибирцев:

Во-первых, программе требуется дисковод. Это связано с тем, что программа сильно защищена ..... (строки письма корреспондента с соображениями о практическом подходе к копированию программы опущены по известным причинам..).

Во-вторых, в программе существует ещё множество тонкостей, которые Вы сами сразу же найдёте. Есть и ещё некоторые "вещи", о которых Вы узнаете, поиграв приличное время.

В-третьих, программе требуется очень узкий список аппаратных и программных средств. Так, авторами сообщается о необходимости работы в TR-DOS 5.03. Испытания показали, что и на TR-DOS 5.04s программа тоже почему-то работает.

В-четвёртых, в программе корабли больше соответствуют фирменному описанию. Те счастливцы, которые имели доступ к фирменному описанию, знают, что увидев "Кобру МКЗ" или "Питон" следует выключить лазер и прикинуться "бедной овечкой" ...

ИФК: Здесь есть какой-то элемент непонимания. Дело в том, что к "счастливым" следует отнести всех читателей "ZX-РЕВЮ", т.к. фирменное описание у нас есть и то описание, которое мы давали в "ZX-РЕВЮ" №2, 1991г. - это и был наш перевод фирменного описания, правда чуть-чуть сокращённый ради экономии места. При этом, к сожалению, выпали чертежи кораблей, но и их мы потом тоже дали вдогонку в "ZX-РЕВЮ" №10, 1991г.

КОРР: ...

В-пятых... Но пожалуй уже хватит. Приобретите себе программу и узнаете обо всём сами".

В конце от себя добавлю, что по состоянию на середину октября 1993 года на московских радиорынках настоящей ELITE 2 замечено не было (а у новосибирцев, между прочим, она есть!). То, что уже длительное время продаётся в Москве под этим названием - Версия 3 по Вашей классификации.

ИФК: Ну, вот, конец с концами и сошлись. Ясно теперь, почему Камразе Т.М. не нашёл в ней ничего особенно нового.

КОРР: И ещё любопытная информация по этому поводу. На компьютере AMIGA (500, 500+, 600, 1000, 1200 . . и т.д.) к августу 93-го года тоже, как ни странно, существовало три версии игры. Это ELITE (классика), ELITE Plus (соответствующая "версии 3"). И новая (!) ELITE 2.0, которая наверное и похожа на вышеописанную "спектрумовскую" ELITE 2.

Кстати, каталоги игр для АМИГИ подозрительно напоминают спектрумовские. Один только сериал DIZZY (1-8!) чего стоит, но об этом можно было бы писать ещё долго.

ИФК: Спасибо, Алексей. Наши почитатели DIZZY теперь кстати будут знать, что их ждёт в ближайшие годы.

В заключение Алексей очень тепло отзывается о своей машине ATM-turbo 512++ v4.50 (память как у АМИГИ, возможности как у СПЕКТРУМА). Второй знак "плюс" означает доработку до почти полной совместимости - классная машина! Алексей очень рекомендует

также текстовый редактор ZX-WORD v1.01 из Харькова.

Ещё раз спасибо, Алексей, мнение эксперта очень важно нашим читателям.

#### Вопросы совместимости.

Алекс Вебер из посёлка Краснозёрское Новосибирской области готов помочь тем, у кого есть проблемы с запуском программы Silent Service. Суть проблемы в том, что после загрузки БЕЙСИК - блока, программа стартует, появляется надпись S. Service, выдаётся короткий звуковой сигнал и ... программа сбрасывается.

Вскрытие показало, что загрузчик в машинных кодах встроен в БЕЙСИК после REM и там сразу за выдачей звукового сигнала стоит опрос порта 31 (Кемпстон-джойстик). Если на джойстике - 0, то программа сбрасывается.

Итак, нужно позаботиться, чтобы во-первых Кемпстон-джойстик был подключён, а во-вторых, чтобы на нем был не ноль, например, нажать кнопку джойстика во время опроса. После нормального старта загрузки кнопку можно и отпустить.

Другой подход - убрать этот опрос из загрузчика.

Непонятно, какие цели преследовал тот, кто таким образом переделал загрузчик.

И ещё: в программах, взломанных INX'S SOFT и MICROPOL, а также Б. Гильбертом после 90-го г. уже встроены POKES, но стоят за оператором REM. Поэтому если Вы хотите ими воспользоваться, то удалите REM в БЕЙСИК - загрузчике и запустите его снова, а затем грузите программу дальше.

Спасибо за "Программирование в машинных кодах". По Вашей книге я научился разбираться с кодом фирменных программ.

ИФК: Спасибо и Вам, Алекс. Ваши слова - для нас отличная реклама. Мы знаем, что эта книга пользуется особой популярностью и надеемся на то, что наши новые книги, стоящие в плане на 1994 год тоже получат Ваше признание.

\* \* \*

Масленников В.Г. из г. Чебоксары пишет о том, что почти на всех отечественных моделях не идут программы SHORT CIRQUIT и TOP GUN. Причина - в отсутствии порта FFH.

Опрашивание порта в программе SHORT CIRQUIT выглядит примерно так:

```
LOOP LD A, 28H
      IN A, (FFH)
      INC A
      JR Z, LOOP
      RET
```

Подпрограмма зациклена до тех пор, пока с порта FFH поступает число FFH. Сделано это для того, чтобы спрайты на экране не мерцали.

Вместо этого фрагмента можно вставить в программу другой, имеющий ту же длину и не портящий никаких регистров, кроме аккумулятора:

```
PUSH HL
LOOP LD A, (HL)
      INC HL
      INC A
      JR NZ, LOOP
      POP HL
      RET
```

Для тех же, кому важнее конечный результат, можно порекомендовать загрузить блок длиной 41986 в копировщик COPY - COPY под адрес 23296 и дать:

```
POKE 33621,229; 126; 35; 60; 32; 251; 225; 201
```

Аналогично ремонтируется и TOP GUN. Для этого блок 40100 загружается в COPY - COPY под адрес 24625 и:

```
POKE 26642,251; 118; 118; 0; 0; 0; 0; 0
POKE 56160,229; 126; 35; 60; 32; 251; 225
```

Значение такого совета трудно переоценить и мы сердечно благодарим Владислава Геннадьевича от имени всех читателей, которым помогли его советы. Вполне возможно, что с такой технологией удастся "отремонтировать" ещё не одну программу. Ждём сообщений.

### Советы и секреты.

Нам пишет THE CAT из г. Харькова.

Привет, ИНФОРКОМ! Посылаю выполненное мной в жанре "компьютерная новелла" описание игры SAMURAI WARRIOR (к сожалению, фирму - изготовителя указать не могу - известно только, что взломана она была командой S. S. CAPTAIN & BAMSEE) в 1989 году.

Жанр игры определить довольно трудно - я бы назвал это ADVENTURE/ACTION. Игру отличает динамичность, прекрасно выполненная графика и интересный сюжет. Напечатайте, если можно...

ИФК: Печатаем, причем с огромным удовольствием. Пробным камнем для "новеллы", как Вы знаете, является вопрос ЗАХОЧЕТСЯ ЛИЛИ НЕТ ЧИТАТЕЛЮ СЫГРАТЬ В ЭТУ ИГРУ?! Попробовали Вашу новеллу на себе и так захотелось сыграть... В общем, ещё десяток подобных новелл и у нас некому будет выпускать "ZX - РЕВЮ", все будут играть, играть и ещё сто раз играть.

Единственное, что успокаивает, так это то, что написать так, как это сделали Вы, очень непросто и вряд ли на нас в ближайшее время обрушится ЦЕЛЫЙ ДЕСЯТОК.

Мы надеемся, что Вы простите небольшие комментарии от нас, вставленные после Вашей новеллы. Сами ведь понимаете, нам тоже хочется прислониться к хорошему делу, а то ещё уволят по сокращению штатов...(шутка).

КОРР: И ещё в раздел "ФОРУМ". В игре "ИМПАКТ" можно получить вечную жизнь, если в строке 30 загрузчика убрать команду REM перед стоящим там POKE (У меня версия игры, в которой после загрузки первого блока, остановки и нажатия "BREAK" на экране появляется надпись "Disked by Billy").

Вот и всё. С большим уважением. THE CAT.

P.S. Кто бы помог (если можно) достать программы "LORD OF THE RINGS" и "SHADOWS OF MORDOR"?

310142 Харьков а/я 1968, THE CAT.

\* \* \*

Нам пишет Шалыминов С.П. из Нижнего Новгорода.

КОРР: Хочется дополнить план игры к программе Aliens, опубликованный в № 5-6 за этот год. Можно видеть, что комнаты пронумерованы от 1 до 255, но нигде на плане нет комнат с номерами 249...252. Попасть в них можно только из комнаты 248, предварительно расстреляв там все коконы. План принимает вид:

```

      XXX
      255
      254
      205
.... 205 ....
      253
250 249 248 251 252
      247
      ...
```

ИФК: Спасибо за ценное дополнение. Сейчас уже не установить, то ли в письмах авторов эти комнаты были не разысканы (черновики не сохранились), то ли это мы при наборе плана их пропустили. Скорее второе, чем первое. Тем не менее, мы рады Вашему письму и если вина лежит на нас, то это к счастью, ведь благодаря этой ошибке мы имели честь познакомиться с Вами.

КОРР: ...Я это узнал только методом проб и ошибок. Долгое время я играл, держа всю карту в памяти, пока не устал. Тогда составил свой план и обнаружил этот парадокс с

"пропавшими" комнатами. Начал их искать, и вот ... результат перед Вами.

Я тоже очень люблю составлять карты к играм и составил их к Robin Good, Dan Dare, Equinox, Gunfright, Death Wish 3, Resque, на подходе карта к Strike Force Cobra. Если кто-то нуждается в них, могу прислать.

603163, Н. Новгород, Ул. Бринского, д.5, к.1, кв.80 Шалыминову С.П.

Ещё я люблю разыскивать POKES или проверять взятые из других источников. Вот несколько, найденных или лично проверенных POKES:

|                    |                            |
|--------------------|----------------------------|
| Stop the Express - | 34467,0; 34929,0; 35260,0  |
| Curse              | 63033,0; 64613,0           |
| Renegade           | 342204,182; 42789,182      |
| Nebulus            | 32913,0                    |
| Zybex              | 45277,0; 45368,0           |
| L. Ninja           | 236578,0                   |
| Robot Escape       | 54697,0; 54698,50; 54699,3 |
| Bionic             | 34247,0; 34274,0           |

Мне очень нравятся Ваши публикации по ELITE. Сам я пользуюсь версией M128, переделав её под дисковод. Вот только никак не могу сделать отгрузку/подгрузку для диска. Не могу понять, как там это организовано.

ИФК: Может быть кто-то даст четкие советы по этому поводу нашему уважаемому корреспонденту и другим читателям?

KOPP: ...Я тоже раскрутил словарь в программе и тоже знаю, что кода "XX" в нем нет, но я точно знаю, что есть названия планет, которые состоят не только из тех слогов, что есть в словаре, так что рано предполагать, что "Ракксла" не существует.

ИФК: Это интересно! Особенно если учесть информацию, полученную от Абдрахманова Руслана (см. выше).

KOPP: В этой версии почему-то не проходит безтопливный перелёт со станции на станцию, а вот в версии Be-Be Soft у меня это получалось.

В целом я отношу версию M128 К сложным версиям. Пираты здесь, как правило, ASP MK II, KRAIT, SEWINDER, реже - PYTON, совсем редко - COBRA MK III. Ни разу не встретил ADDER. Fer-De-Lance можно встретить в двух случаях: если летишь к звезде и статус - Clean, он не стреляет первым. Если же у Вас статус Fugitive - он нападает сразу, даже если появится в переднем экране.

ИФК: Типичное поведение "Охотника за призами", занимающегося боевыми действиями ради спортивного интереса. Можете быть уверены, его рейтинг - ELITE или близкий к нему.

KOPP: ... Полиция (VIPER) весьма маневренна. На неё лучше не нападать. А станция - вообще святыня! Стоит один раз по ней выстрелить и по одному начнут вылетать полицейские корабли, до 10 штук. Как правило, это смертельно, если не уйти в гиперпространство.

Очень часто встречаются астероиды с отшельниками, причём по внешнему виду их не отличить от обычных. Если по нему выстрелить, начинает маневрировать и выпускать корабли - KRAIT и SEWINDER.

Вообще, кажется, что это самая интересная версия игры из тех, что я видел.

ИФК: Мы надеемся, что наши пилоты в поисках острых ощущений учтут это мнение. Спасибо за Ваше письмо.

\* \* \*

Гаврилов Сергей Александрович из г. Костромы приводит проверенный им POKE для программы REX-1

23624, 0; 39402, 0; 40057, 0

\* \* \*

Свои POKES, найденные лично, Вам предлагает читатель из Красноярска Татаренко

A.A.

- |                          |   |                                                                        |
|--------------------------|---|------------------------------------------------------------------------|
| 1. SPELLBOUND            | - | 27871, 0; 36133, 0; - энергия или<br>35101, 195; 35102, 53; 35103, 206 |
| 2. DIZZY 2               | - | 29289, 201 - неуязвимость                                              |
| 3. DIZZY 3               | - | 42481, X - кол-во жизней<br>63001, 0 - беск. жизни                     |
| 4. DIZZY 4               | - | 29623, 0; 29624, 195 - беск. жизни                                     |
| 5. DIZZY 5               | - | 51291, 0 - беск. жизни                                                 |
| 6. STAR BOWLS            | - | 47806, X - кол-во жизней<br>46278, 0 - беск. жизни                     |
| 7. SCUMBALL              | - | 49098, X - жизни                                                       |
| 8. COCORAMA              | - | 43855, 0; 48658, 0 - беск. жизни                                       |
| 9. TOMCAT                | - | 37174, 0 - беск. жизни<br>37171, 201 - неуязвимость                    |
| 10. TRACER               | - | 50273, 0 - время<br>50613, 0 - энергия<br>50649, 0 - беск. жизни       |
| 11. TWISTER              | - | 42412, 0; 42490, 0 - энергия                                           |
| 12. DARK FUSION          | - | 50407, 0; 50408, 0 - беск. жизни                                       |
| 13. XENON                | - | 25148, 201 - жизни                                                     |
| 14. CONQUEST (ERBE SOFT) | - | 62370, 0 - останавливает противников                                   |

\* \* \*

А эти POKES пришли к нам из Белоруссии, их прислал наш читатель из г. Гомеля Владимир Хронов.

- |                                            |                                       |
|--------------------------------------------|---------------------------------------|
| Gomel                                      | Technosoft'92                         |
| Все POKES даны для игр в развёрнутом виде. |                                       |
| 1. FROST BYTE                              | 33989, 0; 36560, 0                    |
| 2. JOE BLADE-3                             | 43059, 195                            |
| 3. KRION                                   | 45004, 0                              |
| 4. PANAMA JOE                              | 38633, 60                             |
| 5. GOODY                                   | 46187, 0                              |
| 6. CAPTAIN TRUENO                          | 38310, 24; 38363, 201                 |
| 7. DRAKONUS                                | 64215, 0                              |
| 8. SAVAGE 1                                | 39446, 0                              |
| 9. ELVEN WARRIOR                           | 36767, 0                              |
| 10. CROM                                   | 56190, 0 - Energy<br>56220, 0 - Lives |
| 11. YETI                                   | 47894, 0 - Lives                      |

\* \* \*

Семёнов Алексей из г. Балаково лично проверил следующие POKES:

- |                |                                                                                        |
|----------------|----------------------------------------------------------------------------------------|
| ELEPHANT ANTIC | 35354, 255; 35363, 255                                                                 |
| BALL BREAKER   | 35840, 0                                                                               |
| BALL BREAKER 2 | 38874, 0; 35938, 0; 35729, 99                                                          |
| KING KONG      | 30033, 0 - бессмертие<br>30038, 0 грязь к ногам не липнет;<br>30053, 24 огонь не жжёт; |

|                |                                                           |
|----------------|-----------------------------------------------------------|
|                | 30063, 24 бочки не сбивают.                               |
|                | 30043, 24                                                 |
|                | 30048, 24                                                 |
| ALPIN GAMES    | остановить программу кнопкой BREAK, удалить строку 5620 и |
| дать CONTINUE. |                                                           |
| PHOTO ALBUM    | 56203, 0                                                  |
|                | 56364, 0 время стоит                                      |
|                | 56249, 0                                                  |
|                | 51765, 0 неогр. ошибки                                    |
| IMPACT         | пароли:                                                   |
|                | EGGS, CHIP, LEAD, TICK, CASE, FACE, J.D.                  |
| BREAKNECK      | 46090, 0 - жизни                                          |
|                | 46013, X - (0<X<200) - номер комнаты.                     |

\*       \*       \*

Наш читатель Шабалин М.М. из г. Ухты вскрыл "жучок" в программе "RENEGADE" и готов им поделиться.

Этот "жучок" относится к тем, кто не может пройти четвёртый уровень игры (босса с пистолетом) и заключается в том, что когда в Вас стрельнет "босс", сразу нажимайте кнопку "PAUSE" (если Вы выбирали клавиатуру, то должны были её назначить). После этого летящая пуля исчезает и это облегчает прохождение уровня.

\*       \*       \*

У Алекса Кучкина из г. Петропавловска - Камчатского для наших читателей заготовлено несколько POKES и есть один вопрос.

|                |                        |
|----------------|------------------------|
| TOTAL ECLIPSE  | 47793, 0 - вода,       |
|                | 47690, 0 - здоровье.   |
| ROBOCOP        | 39577, 24 - энергия.   |
| N. O. M. A. D. | 40703, 0 - бессмертие. |
| NINJA COMMANDO | 51763, 0 - бессм.      |
| TURTLES        | 53744, 0 - энерг.      |
|                | 47503, 0 - жизни.      |
| MASTER BLASTER | 49988, 0 - жизни.      |

К сожалению, прочие POKES мы дать не смогли по причине неразборчивости письма.

Теперь вопрос. Алекс очень любит авиаимитаторы, особенно F-16 фирмы Digital Interceptor. К сожалению, после поражения программа высвечивает надпись PLAY ... и чего-то ждёт. Есть подозрение, что в ней не все блоки. Может кто-то дать раскладку блоков этой программы?

\*       \*       \*

Изящный букет секретов прислал наш юный читатель из г. Находка Приморского Края Максим Кобяков. Сначала несколько POKES:

|                   |   |                             |
|-------------------|---|-----------------------------|
| 1. AFTER BURNER   | - | 37934, 0; 37935, 9; 37936,0 |
| 2. FRIGHMARE      | - | 44051, 0; 48455,0           |
| 3. LAST DUEL      | - | 37605,0; 40063,0 - 1 игрок  |
| 4. LED STORM      | - | 37337, 201                  |
| 5. RING WARS      | - | 39417, 0; 39534, 0          |
| 6. STREET FIGHTER | - | 42147, 195                  |

7. ZYBEX - 45277, 0; 45368, 0  
8. DIZZY 5 - 51291, 0; 41815, 0

Все POKES лично проверены Максимом.

Кроме того, вниманию читателей предлагаются пароли для программы MEGANOVA:

26719 - 2-ая миссия;

16640 - 3-я миссия.

Полезный "жучок" из программы NEWYORK WARRIORS:

Если начать игру вдвоём и в первом блоке убить первого игрока, чтобы он больше не появлялся, а вторым игроком пройти этот блок до конца, то во втором блоке первый игрок появиться снова, но уже с 15-ю жизнями.

Этот способ применим в любом блоке.

\* \* \*

Просьба о помощи.

Наш постоянный читатель из г. Талнах Красноярского края Фёдор Юхнович обращается к экспертам - профессионалам. Он считает, что проработка программ - имитаторов им удаётся лучше всего, пример тому работы Хоминича и Жарова по программе ACADEMY.

ИФК: Дорогие друзья! Мы ежемесячно получаем тысячи писем, в которых нам выражается благодарность за то, что мы делаем. Мы ценим Ваше участие и рады этому, но наконец-то к нам пришло письмо с оценкой труда наших корреспондентов. Наверное, не очень справедливо оставлять за кадром тех, кто бескорыстно делится с Вами всем, что знает и умеет.

ПРИМЕЧАНИЕ: статьи наших корреспондентов мы помечаем символом (С) ("копирайт"), свидетельствуя тем самым своё уважение к их авторскому праву. Статьи, не помеченные этим символом и без указания автора, принадлежат "ИНФОРКОМу" и выполнены его сотрудниками в порядке служебного задания.

КОРР: ... Помогите разобраться с программой-имитатором космического полёта - программой ENTERPRI (аналог ELITE).

ИФК: Мы с этой программой не знакомы и тоже с интересом выслушали бы отзывы о ней.



# СОВЕТЫ ЭКСПЕРТОВ

## THE DOUBLE

J.S. Ltd.

Эксперт Матвеев Ю.А. г. Москва.

Вы когда-нибудь представляли себя в роли менеджера футбольного клуба? Если нет, то попробуйте сыграть в неплохую игру "THE DOUBLE" ("ДУБЛЬ") фирмы J.S.Ltd.

Программа "THE DOUBLE" является достойным представителем игр жанра BUSINESS/MANAGEMENT. Любители футбольных баталий, имеющие в своем распоряжении "Спектрум" и эту игру, получают возможность испытать свои способности в роли руководителя английского футбольного клуба.

Ваша задача - привести свой клуб к золотым медалям чемпионата и выиграть Кубок, то есть сделать "дубль", добившись двойного успеха.

Поначалу Вам достается команда среднего уровня, имеющая в своем составе несколько неплохих игроков, на которых, собственно говоря, держится все. Но по мере того как Вы, используя выделенные Вам деньги, покупаете новых, более талантливых футболистов и проводите мудрую политику в подборе состава на очередной матч, Ваша команда начинает расти на глазах. В начале чемпионата Вы присматриваетесь к футболистам, экспериментируете с составом. В конце концов Вы приходите к выводу, что с некоторыми игроками Вам не по пути и с ними придется расстаться. Первые попытки продать слабых футболистов в другие клубы будут неудачными, и это понятно. Кому нужен лишний балласт? Ведь каждому игроку приходится платить немалую зарплату из бюджета клуба. Многие менеджеры придерживаются принципа: "Мы не настолько богаты, чтобы покупать дешевых игроков". Возьмите и Вы этот принцип на вооружение. Не отчаивайтесь, если не удастся быстро продать футболиста. Вполне возможно, что его купят позже, когда из-за травм ведущих игроков некоторые менеджеры будут вынуждены снизить свою требовательность.

После загрузки программы Вы сразу попадете в главное меню:

MAIN MENU

CLUB REPORT (отчет по клубу) LEAGUE DETAILS (информация о ходе чемпионата)

FA FIXTURES (матчи на Кубок) CONTROL MENU (меню управления игрой)

QUIT GAME (прекратить игру)

CONTINUE (продолжить игру)



В начале игры установлены следующие управляющие клавиши:

5 - влево      7 - вверх

6 - вниз      9 - вправо

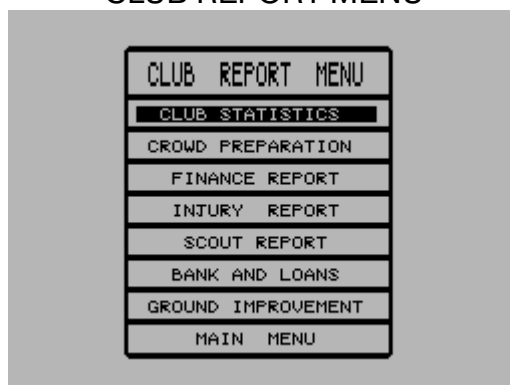
ENTER - выбор

Вы можете использовать джойстик, если в опции CONTROL главного меню, произведете соответствующую застройку. Там же находятся опции сохранения (SAVE GAME) и загрузки (LOAD GAME) отложенной игры.

После выбора управления Вы переходите непосредственно к игре.

Итак, Вам предлагается возглавить футбольный клуб LEICESTER CITY, который выступает в Первом Дивизионе. Отношение к Вам со стороны дирекции клуба достаточно хорошее (об этом можно судить по строке MANAGEMENT CREDIBILITY <кредит доверия> над главным меню). Руководство клуба надеется, что Вы оправдаете высокое доверие, которое Вам оказано, и приведете команду к золотым медалям.

### CLUB REPORT MENU



Здесь есть следующие разделы:

CLUB STATISTICS (статистика клуба).

CROWD PREPARATION (планирование посещаемости матчей).

FINANCE REPORT (финансовый отчет).

INJURE REPORT (отчет по травмам игроков).

SCOUT REPORT (отчет наблюдателя).

BANK AND LOANS (расчеты с банком).

GROUND IMPROVEMENT (развитие стадиона).

MAIN MENU (возврат в главное меню).

### Раздел CLUB STATISTICS:

| AVERAGE GATE<br>14530     | LEICESTER CITY |        |       |       | GROUND CAPACITY<br>31,000 |
|---------------------------|----------------|--------|-------|-------|---------------------------|
| PLAYER                    | POS            | PLAYED | GOALS | WAGES |                           |
| ANDREWS                   | 1              | 0      | 0     | £430  |                           |
| FEELEY                    | 2              | 0      | 0     | £430  |                           |
| VENUS                     | 3              | 0      | 0     | £430  |                           |
| OSMAN                     | 4              | 0      | 0     | £495  |                           |
| MORGAN                    | 4              | 0      | 0     | £550  |                           |
| O'NEILL                   | 5              | 0      | 0     | £470  |                           |
| WALSH                     | 5              | 0      | 0     | £430  |                           |
| McALLISTER                | 6              | 0      | 0     | £495  |                           |
| LYNEX                     | 7              | 0      | 0     | £495  |                           |
| MAUCHLEN                  | 7              | 0      | 0     | £405  |                           |
| MORAN                     | 8              | 0      | 0     | £575  |                           |
| SMITH                     | 9              | 0      | 0     | £520  |                           |
| RAMSEY                    | 10             | 0      | 0     | £430  |                           |
| SEALY                     | 10             | 0      | 0     | £380  |                           |
| WILSON                    | 11             | 0      | 0     | £430  |                           |
| KELLY                     | 11             | 0      | 0     | £380  |                           |
| SELL PLAYER PRINTOUT MENU |                |        |       |       |                           |

Выбрав этот раздел, Вы сможете узнать о средней посещаемости Вашего стадиона (AVERAGE GATE), О количестве мест на стадионе (GROUND CAPACITY), а также такие данные, как номер Дивизиона, в котором играет ваша команда (LEAGUE DIVISION), место, занимаемое командой в турнирной таблице (LEAGUE POSITION) и количество зрителей, побывавших на последней домашней игре (LAST HOME GATE). Тут же Вам предлагается еще одно подменю:

SQUAD DETAILS (Информация по команде).

STAFF DETAILS (Информация по персоналу).

MENU (Возврат в предыдущее меню).

После выбора опции SQUAD DETAILS, Вам представится возможность познакомиться с составом команды, посмотреть статистику на каждого игрока. Здесь приведены данные и о еженедельной зарплате игроков. В низу экрана еще одно меню:

SELL PLAYER (продать игрока).

PRINTOUT (вывести данные на принтер).

MENU (возврат в предыдущее меню). При желании Вы можете продать любого футболиста. Установите курсорную строку на фамилию игрока и выберите опцию SELL PLAYER.

При наличии заявок на покупку футболиста на экране появится таблица с названиями клубов, заинтересовавшихся Вашим предложением, и суммами, которые они предлагают за игрока. Чтобы отказаться от сделки надо выйти из этого подменю. Для оформления сделки, нужно выбрать курсорной строкой устраивающую Вас сумму и нажать FIRE. С этого момента футболист уже не играет в Вашей команде, а деньги за него поступили на Ваш счет в банке.

Вы можете оценить силу каждого игрока по зарплате (WAGE), которая ему выплачивается. Но будьте осторожны! Сумма еженедельных выплат не всегда отражает уровень игры футболиста. Это, возможно, связано с привилегированным положением некоторых игроков. Звездный час у них уже давно в прошлом. Однако, за свои прежние заслуги они еще имеют хорошую зарплату, которая может быть гораздо выше, чем у молодых. Впрочем, бывают и другие ситуации, молодым, подающим надежду футболистам, руководство клуба иногда также завышает зарплату, как бы авансируя их будущие успехи. Вы, кстати, не занимаетесь этими вопросами и все решения о зарплате лежат на совести директора клуба.

Чтобы верно оценить уровень игры каждого отдельного игрока, Вы нанимаете независимый экспертов - наблюдателей (SCOUT), которые могут описать достоинства и отличительные характеристики любого интересующего Вас футболиста.

Для начала войдите в подменю STAFF DETAILS. У Вас имеется один эксперт, который трудится под вашим чутким руководством с окладом в 450 фунтов стерлингов в неделю. Вы можете нанять второго эксперта. Это будет стоить еще 450' в неделю. В этом подменю предлагается за те же деньги нанять и физиотерапевта (PHYSIOTHERAPIST), который будет необходим, как воздух, когда игроки начнут получать травмы в матчах. Впрочем, Вам не обязательно нанимать терапевта и второго эксперта. В ваших силах даже уволить и первого эксперта, дабы сэкономить средства и побыстрее набрать круглую сумму в банке, но делать этого не рекомендуется из-за проблем, которые могут впоследствии возникнуть.

#### Раздел CROWD PREPARATION:

Здесь Вам сообщают о месте проведения следующей встречи. В том случае, если Ваша команда играет дома, Вам предлагают ввести Ваш прогноз о количестве зрителей, которые придут на матч. Эти цифры будут использованы при подготовке полиции к матчу, которая будет обеспечивать безопасность на стадионе. Если Вы ошибетесь в своем прогнозе и назовете заниженные цифры, то полиция, не готовая к большому количеству зрителей, не сможет поддержать порядок на стадионе и болельщики могут устроить беспорядки. На центральных матчах тура всегда присутствуют эксперты из Федерации футбола и в случае возникших беспорядков Ваш клуб может быть оштрафован на круглую сумму, если в Федерации решат, что это произошло по вашей вине. Крайне нежелательно так ошибаться несколько раз подряд, Вы можете даже потерять свое кресло, поэтому будьте осмотрительны и перед каждой домашней встречей уделяйте внимание безопасности стадиона. Завышенный прогноз хоть и не влечет за собой таких мрачных последствий, но все же влияет на финансовое положение клуба, а значит и на кредит доверия к Вам со стороны руководства.

#### Раздел FINANCE REPORT:

Здесь Вы получите полную раскладку всех доходов и расходов клуба.

GATE RECIEPTS (плата за входные билеты).

TRANSFERS (плата за продажу/покупку игроков).

PLAYERS WAGES (Зарплата игроков).

STAFF WAGES (зарплата персонала).

TRAVEL AND HOTEL (расходы на переезды и гостиницы).

FIRES AND DAMAGES (штрафы и повреждения).

POLICE EXPENCES (оплата полиции).

GROUND RENT (арендная плата за землю).

GROUND IMPRONEMENT (расходы на расширение стадиона).

#### Раздел INJURY REPORT:

В этом разделе сообщается о травмах, полученных футболистами. Если в Вашей команде все благополучно, то выводится сообщение: NO INJURIES (травм нет). Если есть травмированные, то сообщается фамилия футболиста, вид травмы и количество недель, на которое игрок выбывает из основного состава.

Вам предлагается решить: освободить травмированного футболиста от тренировок полностью или нет.

В разделе есть подменю, в котором определяется вид тренировки для травмированного футболиста:

NO WORKOUT (без тренировки).

LIGHT WORKOUT (легкая нагрузка).

MEDIUM WORKOUT (средняя нагрузка).

HARD WORKOUT (тяжелая нагрузка).

Игроки могут получить следующие виды травм:

TORN LIGAMENTS (порваны связки).

BROKEN ARM (перелом руки).

ANCELE INJURE (повреждена лодыжка).

BROKEN LEG (перелом ноги).

HARMSTRING - растяжение сухожилий.

BROKEN NOSE - перелом носа.

Существуют и другие виды травм, если Вы достаточно глубоко погрузитесь в эту игру, то вам придется иметь с ними дело.

При легких видах травм игроки выбывают на небольшой срок, а при тяжелых травмах этот срок может достигать полугода.

#### Раздел SCOUT REPORT:

Этот раздел позволяет ознакомиться с отчетами экспертов-наблюдателей, которых Вы посылаете на очередные матчи с целью изучения уровня игры футболистов, выставленных на продажу другими клубами. Вы, конечно, можете рассчитывать только на свои силы и обойтись без услуг экспертов, но в этом случае высока вероятность купить "кота в мешке". Для того, чтобы отправить эксперта на матч, выберите с помощью курсорной строки опцию SEND ON MISSION, затем задайте дивизион, в который направляется эксперт, а потом выберите команду и футболиста, которого следует посмотреть. Эксперт прощается с Вами и со словами: "Увидимся после матча" отправляется выполнять поручение.

Прочитать последнее сообщение эксперта можно, если выбрать опцию READ LAST REPORT. Аналогичные действия проводятся и со вторым экспертом (если он есть).

Рекомендуется в начале сезона, пока не начались переходы футболистов из клуба в клуб, оценить с помощью экспертов игроков своей команды.

После очередного тура чемпионата эксперт возвращается с отчетом на футболиста, которым Вы заинтересовались. Отчет состоит из пяти характеристик плюс оценка приблизительной цены за игрока. Эксперт описывает футболиста, используя спортивный сленг. Словарь эксперта приведен в конце настоящего описания.

#### Раздел BANK AND LOANS:

В этом разделе приводятся данные по Вашему счету в банке.

WEEKLY INTEREST (процент по вкладу в неделю).

WEEKLY REPAYMENTS (процент по долгам в неделю).

CURRENT BALANCE (сумма на счете).

OUTSTANDING LOANS (Внешний долг).

Здесь же Вы можете провести с банком финансовые операции:

APPLY FOR LOAN (попросить кредит)

## MAKE PART REPAYMENTS (выплатить часть долга)

### Раздел GROUND IMPROVEMENT:

Вы можете расширить стадион своего клуба, увеличив количество зрительских мест, что в принципе может повысить доход от проданных билетов, если команда будет играть успешно и посещаемость матчей будет расти. Таблица данных:

PRESENT SPECTATOR CAPACITY - (вместимость стадиона).

MAXIMUM SPECTATOR CAPACITY - (максимально возможное количество мест).

COST/1000 EXTRA SPECTATORS - (плата за каждую тысячу дополнительных мест) и подменю:

INCREASE SPECTATOR CAPACITY -(увеличить число зрительских мест).

CLUB REPORT MENU (выход).

Расширять стадион Вам придется не скоро, ибо это влечет за собой слишком больше расходы. Да и посещаемость матчей с участием Вашей команды в первое время невелика.

### LEAGUE DETAILS

Здесь Вам предлагается вся справочная информация по чемпионату. При выборе этой опции Вы оказываетесь в подменю: LEAGUE FIXTURES (предстоящие матчи).

LEAGUE RESULTS (результаты матчей).

LEAGUE TABLES (турнирные таблицы)

CLUB STATISTICS (статистика по клубам).

MAIN MENU (возврат в вышестоящее меню).

### Раздел FA FIXTURES:

Здесь указывается пары команд, которые встречаются в играх на Кубок. Первые встречи начинаются на двадцать второй неделе чемпионата. Жеребьевка клубов проводится за несколько недель до старта игр на Кубок.

### CONTINUE.

Эта опция включается, когда все необходимые приготовления закончены и Вы переходите к расстановке игроков из поле на предстоящий матч. После того, как Вы расставите игроков и нажмете FIRE, Вас спросят нужно ли выводить результаты матчей на принтер. После Вашего ответа на экране появится таблица с результатами всех матчей тура. Помимо счета в таблице указываются составы команд, а напротив фамилий игроков, забивавших в этой игре, пишется число забитых мячей.

Если Вы неправильно спрогнозировали число зрителей, и силы полиции не смогли обеспечить порядок на стадионе, то в этом случае после статистики игры с участием Вашей команды сообщается о имевших место беспорядках. Здесь же приводится заявление полиции о том, что они были неверно информированы о количестве зрителей. Далее следует сообщение, что на матче присутствовал инспектор от Федерации футбола.

По окончании тура вопрос о беспорядках на стадионе будет рассмотрен в Федерации футбола и Вал клуб могут оштрафовать. Иногда, правда, Федерация приходит к выводу, что в этом нет Вашей вины и оставляет клуб в покое.

К десятой неделе чемпионата начинаются торги на бирже игроков. До этого приходят заявки на продажу того или иного игрока. Вот здесь Вам и потребуется помощь Ваших экспертов. Просмотрев такую заявку и выбрав подходящего игрока, Вы передаете все данные о нем своему эксперту. Эксперт, посмотрев этого футболиста, сообщает Вам его отличительные особенности и цену, за которую его можно будет приобрести. Помните, однако, что эксперт тоже не Бог и он никогда не сможет назвать Вам точные цифры. Цена игрока определяется на аукционе. Футболист переходит в тот клуб, который объявит за него максимально цену. Иногда бывает и так, что игрок остается в своей команде под давлением клуба или из-за травмы. Вы, кстати, тоже не можете продать травмированного игрока.

Не хотелось бы раскрывать всех секретов этой замечательной игры, но об одном сказать можно. В Вашей карьере менеджера будут взлеты и падения. Пока Вы не наберете

необходимый опыт Вас ждут неприятные разборки в кабинете директора клуба, проклятия болельщиков и недовольство команды. Возможно, Вы будете с позором уволены с работы и Вас приютит какая-нибудь другая, более бедная и слабая команда. Зато потом, когда раскроется во всей красе Ваш талант, ждите приглашений от прославленных богатых клубов, известных всему миру. Тогда Вы, попивая кофе и раскуривая толстую сигару, будете сами выбирать клуб, который достоин того, чтобы в него пришел работать такой известный и талантливый менеджер, как Вы.

### **Приложение. СЛОВАРЬ ЭКСПЕРТОВ**

Характеристика каждого игрока складывается из пяти отдельных, независимых друг от друга высказываний. Для удобства пользования словарей, фразы разбита на пять групп. Первую фразу эксперта надо искать в первой группе, вторую во второй и т.д. Для характеристики вратарей все фразы, относящиеся к ним, выделены в отдельный раздел, состоящий из пяти групп.

Внимательный читатель обратит внимание на то, что абсолютное большинство заявлений экспертов об игроках имеют положительную тональность. Никто никого не ругает и не порочит. В этом, наверное что-то есть, но все же Вам надо как-то суметь выделить отдельные нюансы этих высказываний. Держайте сами, для этого нужен только опыт.

#### **1. ВРАТАРИ**

##### **A**

GOOD HANDLING (хорошо обращается с мячом).  
NOT MACH GETS AWAY FROM HIM (мимо него и муха не пролетит).  
COMFORTABLE ON THE BALL (спокойно действует с мячом).  
THE SAFEST PAIR OF HANDS IN THE GAME (это надежная пара рук).

##### **B**

GETS RID OF THE BALL WELL (хорошо выбивает мяч в поле).  
GOOD AT CLEARING HIS LINES (хорошо действует на выходах).  
OK ON CLEARANCES (неплохо забирает мяч).  
ACCURATE WITH HIS KICKING (точно выбивает мяч).

##### **C**

QUICK TO REACT (быстро реагирует)  
SUPERB REFLEXES (превосходная реакция).  
VERY ALERT (всегда начеку).  
MOVES WELL (хорошо перемещается).

##### **D**

AWAKE OF SITUATIONS (хорошо чувствует ситуацию).  
UNCANNY POSITION SENSE (невероятное чувство позиции).  
OCCASIONALLY GETS CAUGHT IN TWO MINDS (иногда берет мяч интуитивно).  
READS THE GAME SUPERBLY (превосходно чувствует игру).

##### **E**

HE COULD BE A SHIP (возможно, это "верняк").  
HE WON'T COME CHEAP - HE'S GOOD (этого парня дешево не возьмешь).  
WORTH LOOKING AT BOSS (босс, к нему стоит присмотреться).

#### **2. ПОЛЕВЫЕ ИГРОКИ**

##### **A**

DOESN'T DWELL ON THE BALL (не мешкает с мячом).  
LOOKS COMFORTABLE ON THE BALL (на первый взгляд, спокойно действует с мячом).  
NICE CONTROL (хороший контроль мяча).  
VERY GOOD CONTROL AROUND THE BOX (отлично контролирует мяч на любом участке поля).  
DOESN'T FLAP UNDER PRESSURE (отлично действует в сложной ситуации).  
IMPRESSIVELY COOL ON THE BALL (хладнокровен в обработке мяча).  
SUPERB BALANCE AND CONTROL (превосходно удерживает и контролирует мяч).  
NOT AFRAID TO TAKE ON OPPONENTS (смело идет на плотную игру).

## В

NOT FAINT HEARTED (смелый игрок).

COMPETITIVE (упорный).

HUSTERS DEFENDERS WELL (хорошо обводит защитников).

HE'S DETERMINED IN THE TACKLE (решителен в схватках).

DISHES OUT A LOT OF STICK (Пресекает проходы соперников).

OPPONENTS HAVE A BATTLE WITH THIS LAD (этот парень доставляет соперникам много хлопот).

GETS WELL AND TRULY STUCK IN (удачно и быстро подключается к атаке).

CHALLENGES EVERYTHING IN THE BOX (умеет все на поде).

## С

UNSELFISH (не передерживает мяч).

DISTRIBUTES THE BALL WELL (хорошо распределяет мячи).

SPRAYS HIS PASSES WELL (делает неплохие передачи).

USES THE BALL WELL (ХОРОШО владеет мячом).

GREAT VISION - HIS PASSES ARE CLASS (отлично видит поле - его пасы великолепны).

HIS PASSING IS PURE MAGIC (его передача - просто волшебство).

HIS PASSING IS REAL QUALITY (надежно играет в пас).

SETS UP HIS FELLOW FORWARDS WELL (хорошо питает мячами линию атаки).

## Д

THIS LAD'S HUNGARY FOR GOALS (остро нацелен на ворота).

THIS LAD WILL SCORE A FEW (этот парень еще себя покажет).

LOVES TO SHOOT GIVEN HALF A CHANCE (бьет при любой возможности).

HIS FINISHING IS LETHAL (его удар смертелен).

HAS THE OCCASIONAL POT AT GOAL (забивает время от времени).

NIGHT SCORE A FEW (из него может быть толк).

ALWAYS LOOKING TO GET ON SCORE SHEET (он рвется забивать).

NOT AFRAID TO SHOOT AT GOAL (бьет из любых положений).

## Е

NICE TEMPERAMENT WORTH WATCHING (прекрасный темперамент, за ним стоит понаблюдать).

KEEP AN EYE ON THIS LAD BOSS (босс, присмотритесь к этому парню).

COULD BE THE BARGAIN YOU'RE LOOKING FOR (возможно, это то, что Вы ищете).

HIS GOALS WILL REPAY HIS FEE I'M SURE (я уверен, он стоит своих денег).

A REAL ASSET - NO DOUBT ABOUT IT (отличное приобретение, можно не сомневаться).

A NATURAL, QUICK, SHARP. - (одаренный, быстрый, точный игрок).

## URBAN UPSTART



Эксперт Дурченков Алексей г. Челябинск

Программа "URBAN UPSTART" не является такой же логической головоломкой, как, скажем, "SHERLOCK", однако если Вы решите заняться ее прохождением, она доставит Вам немало приятных часов.

Если у Вас нет пока опыта в общении с приключенческими программами, но Вы ищете, с чего бы начать - начните с программы "URBAN UPSTART" - и Вы не пожалеете о своем выборе. Течение игры очень спокойное. Ваш герой погибнуть не может, самое страшное, что может Вас ожидать - временный арест в КПЗ или же принудительное лечение в муниципальном госпитале. Итак, если Вы решились - начнем.

Городок Скартхопрс - забытое Богом местечко, где даже дети держат за пазухой кнопочные ножи. Здесь все изменения сводятся к тому, что с годами на домах появляется новый слой краски. Люди годами ждут свободного рабочего места и зачастую умирают, так и не дождавшись его. Немногие люди приезжают сюда. Сможете ли Вы спастись из этой глуши, сбежать к цивилизации?

Церковные колокола пробили 3 раза. Попробуйте спастись и, может, Вам повезет.

Внимание!

ИНФОРКОМ ПРЕДУПРЕЖДАЕТ:

Чтение этой статьи может быть опасным для тех, кто предпочитает проходить игры без подсказок.

Приведенные сведения в значительной степени раскрывают технику прохождения игры.

Для тех, кто захочет использовать подсказки только при острой необходимости, мы разбили текст статьи на разделы, привязанные к конкретным эпизодам. Надеемся, что автор простит нам такое вмешательство, это сделано в интересах читателей.

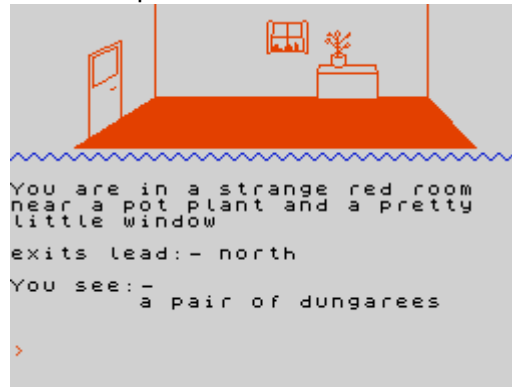
Вот эти разделы (ознакомьтесь с ними до чтения статьи и пользуйтесь именно тем, который Вам нужен в данный момент):

1. Дебют.
2. Книжный магазин.
3. Автобусная остановка.
4. Телефонная будка.
5. Банк.
6. Городской парк.
7. Заброшенный дом.
8. Городская ратуша.
9. Дорога в аэропорт.
10. В аэропорту.
- \* 11. Полезные советы.
- \* 12. Нерешенные проблемы.
- \* / - чтение раздела безопасно.



1.

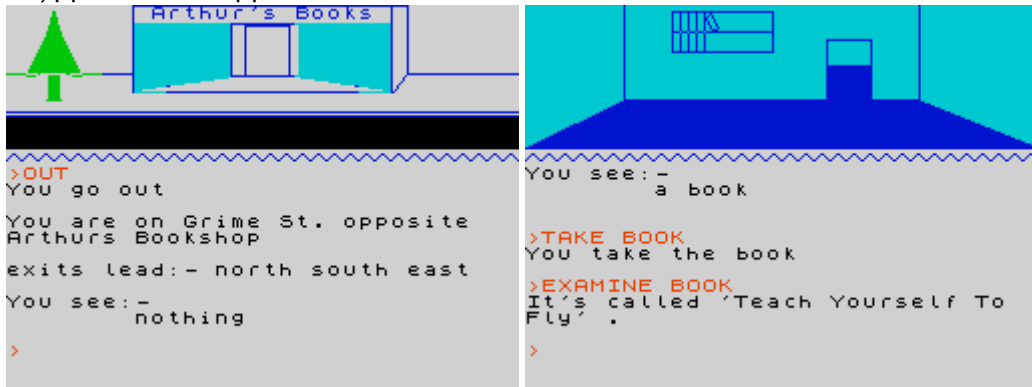
Игру Вы начинаете в маленькой уютной комнате с цветком на окне. Рядом с Вами лежит пара брюк (DUNGAREES), которые не мешало бы взять (TAKE), а потом и одеть (WEAR), учитывая в особенности то, что если Вы будете ходить раздетый по улицам городка, то это вряд ли понравится местной полиции.



Спустившись на первый этаж, Вы окажетесь перед огромной дверью. Можете попытаться открыть ее и выйти наружу, но из этого вряд ли что получится. Выход один - искать ключ. Найдя ключ (A LARGE KEY), не поленитесь осмотреть и остальные комнаты дома - в них Вы можете обнаружить еще пару полезных предметов - ножницы (A PAIR OF SCISSORS) и банку пива (A CAN OF LAGER), стоящую в холодильнике (FRIDGE). Прихватите все с собой - и в путь. Откройте дверь (UNLOCK DOOR, OPEN DOOR) и смело выходите на улицу (OUT).

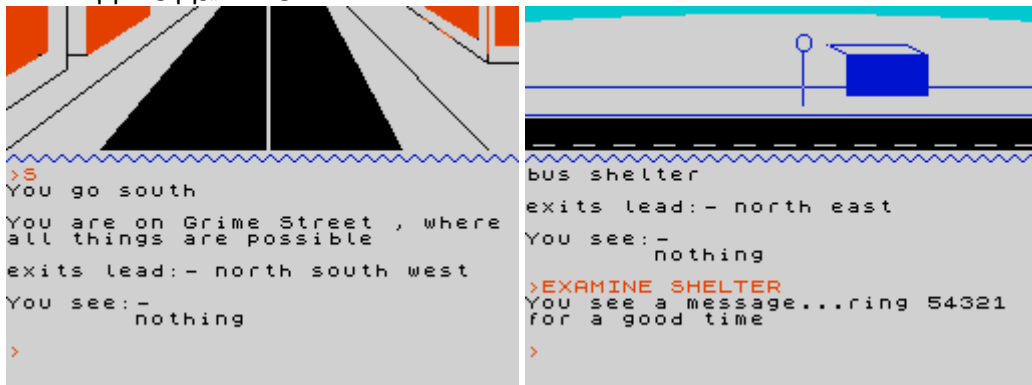
2.

Вы окажетесь на мрачной улице перед уныло стоящим здесь книжным магазином. Зайдя в него (IN), Вы не встретите там покупателей, да и ассортимент имеющейся литературы не балует - на полках стоит всего одна книга (BOOK). Взяв ее и обследовав (EXAMINE), Вы узнаете, что это книга по искусству пилотирования. Решив, что все-таки это лучше, чем ничего, двигайтесь дальше.



3.

К автобусной остановке, находящейся на мрачной улице, вряд ли когда-либо подходит автобус. Зато там лежит забытый одним из потенциальных пассажиров зонтик (AN UMBRELLA). Надпись на стенке гласит - "Желаете приятно провести время - звоните 54321". Прихватите зонтик и идите дальше.



На углу Вашего дома стоят два мусорных бачка, покопавшись в которых Вы можете найти интересное письмо (A LETTER), изучив которое, Вы узнаете, что это уведомление из банка, присланное вместе с кредитной карточкой (A CHEQUE CARD)) и напоминанием, что для того, чтобы узнать Ваш личный номер банковского счета, Вам необходимо позвонить по номеру 77722. Возьмите с собой карточку и хорошо запомните номер телефона банка. Ненужные более предметы - письмо и большой ключ от входной двери можно бросить здесь (DROP) не опасаясь полиции - в конце концов на то она и свалка, чтоб на ней мусорить.

4.

Дойдя до телефонной будки (PHONE BOX), смело заходите внутрь и звоните в банк (DIAL). Приятный голос поблагодарит Вас за звонок и вам сообщат номер банковского счета. Позвоните 54321, если желаете развлечься (хотя напомним - Ваша цель сбежать из города, тратить время на развлечения просто неразумно). Вам тут же сообщат, что время 3 часа 45 минут и 30 секунд. Удивлены? А каких еще развлечений Вы ждали в этом Богом забытом местечке?

5.

Ну, а теперь в банк. На вид банк кажется закрытым, но не пугайтесь - смело заходите в него (WEST). Вставьте кредитную карточку в машину (INSERT CARD). На дисплее высветилась надпись: "Пожалуйста, наберите номер Вашего банковского счета". После набора номера машина выплюнет Вам назад кредитную карточку и Ваши деньги, хотя их всего лишь - пятерка. Карточку можете оставить здесь - она Вам больше не понадобится и, прихватив пятерку (TAKE FIVER), двигайтесь дальше.

6.

Пройдя по проселочной дороге мимо мертвых деревьев, Вы окажетесь около Скартхопрского парка. Обойдя окрестности парка, Вы можете полюбоваться на его достопримечательности - могильный камень на могиле Джона Смита, разрушенную церквушку.

Мрачновато... Но, еще раз укрепившись в своем решении бежать из этого города, прихватите с собой еду (SOME FOOD) и кусочек сыра (SOME CHEESE) - и прочь из парка выполнять свой план побега. Пройдя по аллее Крикунов (названной так, вероятно потому, что ее облюбовали для своих сборищ фанаты местного футбольного клуба) до стадиона, Вы встретите там фаната местных футболистов. Успехами игроки Скартхопрской команды своих болельщиков не балуют и из-за вчерашнего разгромного поражения от "Ливерпуля" и теперь уже твердого последнего места в турнирной таблице настроение у встретившегося Вам фаната преотвратное. Желая на ком-нибудь сорвать зло, он готов убить Вас и вместо побега из города Вам придется валяться в муниципальном госпитале. Поэтому, не теряя времени, отдайте ему прихваченное из дома пиво (GIVE FAN LAGER), прихватите валяющуюся у его ног мышеловку (A RAT TRAP) и уходите прочь из этого квартала, дав себе слово никогда больше здесь не появляться.

Отойдя немного подальше, остановитесь и осмотрите все, что есть у Вас с собой (INVENTORY). В результате Ваших походов у Вас есть книга по искусству пилотирования самолетов, мышеловка, пятерка денег, ножницы, одетые на Вас брюки, зонтик, кусочек сыра и немного еды. Не густо... Особенно удручает безденежье. Начинать новую жизнь с пятеркой в кармане не очень-то веселая перспектива. Волей-неволей вспоминаются рассказы старожил о несметных сокровищах, спрятанных в подвале заброшенного дома, которые охраняются полчищами голодных крыс.

7.

Немногие смельчаки пробовали пробраться туда, но никому это пока не удалось. Оценив свою мышеловку и отметив, заодно, что у Вас есть отличная приманка - сыр, Вы решаете рискнуть. Зайдя по дороге в магазин, реклама которого призывно обещает рыбу и чипсы. Вы ничего из обещанного там не увидите, разве что красную селедку (A RED

HERRING), стоит которую только взять, как на Вас набросятся полчища голодных котов. Кстати, второй перевод слова RED HERRING с английского языка - "ложная наводка". Так что решайте сами - брать или же нет. Да к тому же хозяин магазина - скучающий меломан - выпустит Вас из магазина не раньше, чем Вы послушаете по радио старый номер Френка Синатры (LISTEN).

Идя по улице Дождя без зонтика можно простудиться и попасть в больницу, поэтому как только выйдете на эту улицу, поторопитесь открыть зонтик (OPEN UMBRELLA).

Пробежав по мостику через канал (CROSS BRIDGE), Вы окажетесь у цели. Заржавевшая дверь преграждает Вам дорогу. Открыть ее удастся только после того, как Вы подкрепите свои силы едой (EAT FOOD). Спустившись внутрь по шаткой приставной лестнице, Вы моментально попадаете под атаку дюжины облезлых голодных крыс. Не раздумывая снаряжайте мышеловку! (PUT CHEESE IN TRAP, DROP TRAP). Крысы, завидя, как мышеловка сломала хребет самой здоровой из них, решают убраться в свою нору.

В подвале им сможете найти картонный ящик (A CARDBOARD BOX). Вот оно сокровище! Такой, наверное, будет и Ваша мысль. Что же, берите его и откройте ножницами (OPEN BOX WITH SCISSORS). Внутри Вы увидите только пару ботинок, которые, несмотря на прошедшие века, отлично сохранились и выглядят как новые. Что ж, возьмите их и выбирайтесь из дома.

Остановившись подышать свежим воздухом у канала, Вы найдете маленький ключ (A SMALL KEY). Точно такой же Вы видели у своего однокурсника - летчика. Ну конечно же! Из Скартхопрса надо не убегать, отсюда надо улетать! Но вот проблема - на аэродром пустят только при наличии официального пропуска, а он хранится в ратуше, находящейся на Гражданской улице.

## 8.

Чиновник, служащий в ратуше, не захочет Вас пустить внутрь. Придется отправиться на поиски дальше. Дойдя до западного конца Больничной улицы, Вы обнаружите там Красную папку (SOME RED TAPE). Взяв ее, Вы сможете беспрепятственно проникнуть в ратушу - видимо, Красная капка не только у нас в стране является атрибутом власти. Зайдя в ратушу, возьмите там пропуск на аэродром (SOME OFFICIAL PAPERS).

## 9.

Теперь Ваш путь лежит в аэропорт. Свернув с Навозной аллеи на улицу Дождя и попав оттуда на стройку, оденьте ботинки, иначе Ваши ноги увязнут в грязи и санитары тут же уложат Вас в больницу.

Внимательно осмотрите пустые трубы, валяющиеся на стройке - в них Вы найдете летный костюм

(EXAMINE PIPES, FLYING SUIT).

Пройдя по аварийному переулку до входа в аэропорт, смело входите внутрь.

## 10.

Отдайте дежурному офицеру пропуск (GIVE OFFICER PAPERS). Он потребует с Вас пятерку за вход. Ничего не поделаешь, придется отдать ему последние деньги, только тогда он пропустит Вас к самолету (GIVE OFFICER FIVER) и вот наконец Вы у самолета. Смело забирайтесь внутрь (GO IN PLANE), вставляйте ключ в замок зажигания (INSERT SMALL KEY IS PANEL). Моторы самолета ожили, можно было бы и лететь, но не спешите - управлять самолетом Вы не умеете и Ваш полет закончится весьма плачевно, если Вы не прочтаете книгу по искусству пилотирования (READ BOOK). Прочитали? Ну а теперь в путь. Взлетайте - и Вы победили (TAKE OFF).

## 11.

И в заключение несколько советов:

Не ходите по улицам раздетым, иначе полиция арестует Вас за оскорбление общественной нравственности.

Не изучайте брошенные на улицах машины - Вас арестует за бесцельное времяпрепровождение.

Не бросайте ненужные предметы на улице, иначе будете арестованы за загрязнение окружающей среды.

Не берите красный шарф - футбольные фанаты сильно изобьют Вас за это и санитары отправят в больницу (по-видимому, это атрибут болельщиков команды соперников).

Для того, чтобы выбраться из больницы, найдите белый халат и оденьте его, иначе лечащий врач задержит Вас прямо у входа и вернет на больничную койку.

Для того, чтобы выбраться из полицейского участка, подойдите к сержанту и подавайте команду WAIT до тех пор, пока не зазвонит телефон и Вы не будете автоматически выброшены из участка.

## 12.

Теперь о проблемах.

В программе имеются предметы, которые не удалось нигде использовать:

Около полицейского участка лежит старая шляпа (AN OLD HAT).

Западнее улицы Дождя, на стоянке грузовиков можно взять молоко (SOME MILK). Его можно выпить (DRINK MILK), но что это дает, установить не удалось.

Севернее выхода из госпиталя находится здание, называемое "большой оффис" (A LARGE OFFICE BLOCK). Попасть в него так и не удалось.

Программа реагирует на команду SCORE. Наивысший результат, которого удавалось достичь - 18 из 22 возможных. Где взять еще 4 недостающих до полной победы балла - не ясно.

Ну вот, наверное, и все.

# Компьютерная новелла

(C) THE CAT, 1993 г., Харьков.

## Битва Усаги.

(По игре "SAMURAI WARRIOR")

Усаги было всего 15, когда его отец Акира Тойимбо покончил с собой, сделав "сэппуку"<sup>(1)</sup>. День назад, когда ничто не предвещало печального конца, молодой Усаги стоял вместе с ним во внутреннем дворике отцовского дома, почтительно склонившись перед гостем, только что приехавшим из Киото<sup>(2)</sup>. На куртке гостя была эмблема дома Минамото<sup>(3)</sup>, сам же Усаги имел честь носить герб дома Ходзе. Гость низко поклонился отцу, Усаги же удостоился легкого кивка головой. В один голос с отцом он произнес: "Добро пожаловать!", - и отец жестом пригласил гостя во внутренние покои.

Говорили они недолго и через час только пыль на Западной дороге напоминала о том, что сегодня у них кто-нибудь был. С востока надвигалась черная туча...

Была уже почти ночь, когда старый Акира Тойимбо позвал сына к себе. Когда Усаги вошел, отец сидел на циновке в белом кимоно. Его седая голова была низко опущена. Первые слова, которые он произнес, были: "На все воля неба, сын мой..." Выдержав паузу, он сказал: "Вчера самураи Тайра напали на храм, где твоя мать укрылась от бренности этого мира. Она умерла вместе с другими монахинями с улыбкой под мечом этих собак...". Усаги вскочил. Но Акира повелительно вскинул руку, приказывая ему сесть. Его глаза на миг блеснули. Усаги медленно опустился на циновки. Отец тихим голосом продолжал: "Я стар, сын мой... - К тому же я не могу нарушить клятву, данную мною Будде, больше не убивать ни одного человека. Но я был и остаюсь самураем и поэтому сегодня покину этот мир. Ты - мой единственный сын, единственная надежда сохранить честь семьи. Отомсти за свою мать. Мое завещание ты найдешь в этой шкатулке", - отец указал в левые угол комнаты. "Мужайся, сын мой, и пусть не дрогнет твоя рука ни сейчас, ни после", - и Акира обнажил свой короткий меч для сэппуку. Усаги, встав по левую сторону от отца, медленно вынул из ножен и занес свою катану<sup>(4)</sup>, готовясь облегчить страдания отца. Перед тем, как вонзить короткий меч себе в живот, акира в последний раз взглянул на сына. Глаза Усаги затуманились, его собственный меч сверкнул в этом тумане, как молния, обрывая жизнь самого дорогого для него человека...

Всю ночь Усаги не сомкнул глаз, обливаясь слезами. "Прощальная песня"<sup>(5)</sup> его отца гласила:

В ясном небе  
Растворяется  
Ясной молнии блеск...

Ниже Рукой Акиры было написано:

"Сын мой!

Сейчас, когда мы с твоей матерью снова вместе, мое сердце печется лишь о тебе. Выполни свой долг. Свято чти кодекс чести "Бусидо". Не обижай и защищай слабых. Придет время и тирания Тайра падет. Наш гость сообщил мне, что светлейший князь Еритомо Минамото вскоре выступит против Тайра. Я хочу, чтобы ты, после того, как отомстишь за мать, встал под его знамена. И последнее: убийцу твоей матери зовут Кавасаки Тайра.

Да пребудет Будда с тобой!"

Усаги вышел из дома рано утром. В его кошельке было всего три ре<sup>(6)</sup>. Первым, кого он встретил, был седой крестьянин, идущий с поля. Усаги был далек от тех столичных щеголей, которые при виде крестьян в лучшем случае брезгливо морщились, а в худшем проверяли на них остроту фамильного меча. Отец учил Усаги уважать труд. "У каждого своя карма"<sup>(7)</sup>, -

говорил он. Усаги слегка склонил голову. Крестьянин низко поклонился. Усаги протянул ему один ре. "Спасибо, добрый самурай", - сказал крестьянин и снова низко поклонился. Усаги пошел дальше по пыльной дороге. Где-то там, за лесом, за двумя городами, в горах была усадьба Кавасаки Тайра...

Вдруг что-то заставило Усаги напрячься. Его руки мгновенно извлекли из ножен катану. Меч и человек слились в одно целое. В следующую секунду с дерева с протяжным криком, взмахнув мечом, спрыгнул человек в черной маске. Но у самой земли его встретил молниеносный росчерк меча Усаги.

Усаги медленно стер с меча кровь и осторожно вложил его в ножны. Он и раньше кое-что слышал о людях, закрывающих свои лица. Их звали "ниндзя". Суеверные крестьяне верили, что они знают с нечистой силой, могут внезапно появиться, исчезать и сражаться как звери, неистово махая мечом и метая звездочки - "сюрикены". Крестьянин, которого Усаги вскоре встретил, в ответ на вопрос о ниндзя посмотрел на него с ужасом: "...Да, да, господин, мы слышали с них и раньше. Люди поговаривают, что если свернуть с дороги налево по тропинке перед ручьем, то можно забрести в пещеру, где у них логово... люди оттуда не возвращаются. Не ходите туда, благородный господин. Если уж вам нужно попасть в город, идите прямо, через лес..."

Встреченный Усаги самурай в шлеме, похожем на волчью голову, после того, как Усаги приветствовал его поклоном, сообщил: "Ах, ниндзя? Вон там в кустах лежат трупы двоих. Эти бестии прикинулись крестьянами и напали на меня сзади..."

В городе Усаги решил пополнить свои силы и свернул на постоялый двор. У входа его встретил плутоватый крестьянин и предложил азартную игру. Усаги из любопытства протянул ему монету. Сначала он выиграл, а потом проиграл. Молодой самурай решил не искушать судьбу, так как нужно было еще заплатить хозяйке за еду.

У выхода его поджидал какой-то самурай, который явно хватил лишнего: "Поединок до первой крови, самурай!" - заявил он. Усаги пытался отстранить незнакомца, но тот назвал его трусом и Усаги пришлось все-таки извлечь меч из ножен. Но он не хотел убивать соперника и лишь слегка ранил его. Чужой самурай сразу же протрезвел, и, пробормотав извинения, спешно удалился...

Наконец, после долгого пути, он был у ворот усадьбы Кавасаки Тайра. Стражники узнали герб на куртке Усаги и кинулись на него. Усаги вспомнил об отце и матери и это удесятило его ярость...

Что было дальше, он помнит плохо... кажется, сражаясь против самого Кавасаки Тайра, он видел призрак своего отца из-за спин нападающих...

\* \* \*

Примечание автора:

(1) - Сэппуку (в народе называемое харакири) - обряд ритуального самоубийства самураев.

(2) Тайра, Минамото, Ходзе - самурайские дома в феодальной Японии. В XII веке вспыхнула затяжная кровопролитная война за власть между домами Тайра и Минамото. Дом Ходзе выступил на стороне Минаното.

(3) Киото - резиденция японских императоров.

(4) Катана - длинный меч, используемый самураями в бою, в отличие от короткого, используемого, в основном, для сэппуку.

(5) Прощальная песня - стихотворение (хокку) на тему природы, слагаемое самураями перед смертью. В прощальной песне отца Усаги обыгрывается его имя - Акира ("ясный").

(6) Еримото Минамото - глава дома Минамото, ставший впоследствии сегуном - военным правителем Японии.

(7) Ре - золотая монета.

(8) Карма - закон причинно-следственных связей, по которому человек обязательно получает воздаяние за добрые или злые дела. Ключевое понятие в буддийской философии.

### Примечание "ИНФОРКОМа":

Так получилось, что эта новелла пришла от уважаемого автора в то время, когда мы готовили первый выпуск нового электронного журнала "PC-РЕВЮ" и работали над статьей "Средневековое оружие в программе MIGHT & MAGIC III", исследуя сопутствующую литературу. Обзор этой литературы позволяет сделать пару небольших добавлений.

1. Автор не вполне справедливо называет катану длинным мечом. Конечно же он длиннее, чем упомянутый ритуальный меч для сэппуку, который немного больше кинжала, но все же это не длинный меч, его принято считать коротким.

Дело в том, что в стандартный боевой комплект богатого самурая входило два меча - вакидзаши и катана. Вакидзаши - длинный меч, которым можно действовать как одной, так и двумя руками, а катана - более короткий, вспомогательный меч для левой руки. Конечно, при отсутствии вакидзаши катана становится основным мечом и может считаться относительно длинным. Кстати, именно не очень большие размеры катаны способствовали тому, что это оружие позаимствовали у самураев бойцы ниндзя, т.к. его можно было переносить тайно, укрывая в складках одежды, а при ношении за спиной оно не стесняло быстрых и ловких движений ночных диверсантов.

2. События, описанные в новелле, имеют реальные исторические корни. Они относятся к периоду феодальных войн 12 века, когда власть императоров была заменена военным сегунатом. К сожалению, князя Еритомо Минамото (1147 - 1199) не принято считать светлой личностью. Дело в том, что власть дома Тайра сверг Есицунэ Минамото (1159 - 1189), а Еритомо пришел к власти, коварно загубив своего младшего брата.

Впрочем, согласно легенде, он получил по заслугам. Еритомо погиб, упав с лошади, когда ему привиделся призрак его убитого брата. Предполагают, что роль "призрака" сыграл один из хорошо подготовленных ниндзя, отомстивший правителю за подлость.

Для справки: программа MIGHT & MAGIC III, известная также под названием ISLES OF TERRA выпущена компанией New World Computing и относится к жанру RPG - Role Playing Games - ролевые игры. К сожалению, для "Спектрума" этот жанр малодоступен. Пожалуй, именно в этом жанре, а также в имитаторах IBM имеет преимущество над "Спектрумом". Во всех прочих играх наши читатели могут чувствовать себя на равных с владельцами IBM-совместимых машин, а во многих случаях даже имеют ощутимое преимущество за счет большей интеллектуальности "синклеровских" игр. Впрочем, те, кто заинтересуются этим вопросом подробнее, смогут в ближайшее время стать читателями "PC-РЕВЮ", если хотя бы на время найдут подход к IBM-совместимой машине, а сейчас такая возможность есть уже и в самых отдаленных пунктах.

Мы рассчитываем, что наша стратегия распространения "PC-РЕВЮ" сможет доводить выпуски этого журнала до каждой IBM-совместимой машины в СНГ в течение 5-6 месяцев после выхода очередного номера.

## ИНФОРКОМ-ПРЕСС ПРЕДСТАВЛЯЕТ

Дорогие друзья!

Мы продолжаем рекламную кампанию по продвижению нового электронного журнала "РС-РЕВЮ". К сожалению, у нас не получается, как было объявлено, осветить в этой выпуске итоги ТЕНДЕРА, объявленного в NN 7-8. Дело в том, что этот выпуск ZX-РЕВЮ передается в типографию 10 ноября 1993 Г., а итоги ТЕНДЕРА будут подводиться только 15 декабря. Такая необычная для нас поспешность с выпуском данного номера связана с тем, что мы кардинально улучшаем качество и формат издания ZX-РЕВЮ-94 (каждый номер - 96 страниц формата А5, цветная обложка, выполненная одним из известнейших российских художников и пр.) и потому должны иметь солидный резерв времени на подготовку и печать первого номера будущего года. Кстати, многие наши читатели сейчас не выписывают РЕВЮ-94, полагая, что в конце года будет выпущен годовой сборник. Это ошибка. При новом оформлении и объеме ZX-РЕВЮ-94 мы не сможем издать годовой сборник при всем своем желании.

Тем не менее, все участники ТЕНДЕРА получают от нас персональное уведомление об его итогах, публично же его результаты мы обнародуем в первой выпуске будущего года.

Пока же мы заканчиваем подготовку первого выпуска и уже можем оценить его себестоимость. Исходя из нее сейчас можно предположить, что по всей видимости цена отсечки будет не ниже 25 тысяч рублей. Таким образом, стоимость лицензии на право распространения одного номера РС-РЕВЮ составит не менее 25 тысяч рублей как для юридических, так и для физических лиц при приобретении лицензии сразу на несколько номеров, например на все 12 выпусков будущего года, конечно будут действовать скидки.

У читателей ZX-РЕВЮ есть несколько возможностей приобретения РС-РЕВЮ (для личного пользования или для участия в его распространении) от очень дорогих до практически бесплатных.

1. Вы хотите читать РС-РЕВЮ, но не хотите участвовать в его распространении.

1.1. Вы сможете обратиться в своем городе к нашему дистрибутору и скопировать у него очередной номер за назначенную им плату. Мы не можем определять ту цену, которую наши дистрибуторы назначат за копирование журнала, но можем предположить, что рынок выведет их на уровень от 2-х до 7 процентов (в зависимости от количества клиентов) от той суммы, которую они сами заплатили за лицензию. Плюс, конечно, стоимость дискеты (дискет).

Это доступный и оперативный метод получения свежих выпусков журнала.

1.2. Если в Вашем городе нет нашего дистрибутора, Вы можете обратиться в любой компьютерный салон или в любую фирму, занимающуюся торговлей компьютерами или программными продуктами в своем городе и предложить им связаться с нами. Может быть, им придется предварительно изучить потенциальный спрос. Если заявок будет хотя бы несколько десятков, они наверное согласятся включиться в нашу дистрибуторскую сеть, им это будет интересно, необременительно, а главное - перспективно. Это дело, которое будет развиваться и кто начнет его раньше, тот пойдет дальше.

1.3. Вы можете вообще ничего не делать, а терпеливо ждать, когда номер РС-РЕВЮ сам придет к Вам, в руки через друзей или по компьютерным сетям поскольку выпуски будут распространяться без защиты от копирования, этот момент наступит через несколько месяцев (ориентировочно 5-6). Это практически бесплатный, но и самый длинный путь.

1.4. Ограниченное количество дискет каждого номера мы будем продавать в



подарочном (коллекционном) исполнении в красочных альбомах через дорогие компьютерные салоны, книжные магазины и музыкальные магазины г. Москвы по престижно высоким ценам. Поступать туда эти номера будут не ранее, чем через полтора месяца после рассылки мастеркопий дистрибуторам.

Очевидно, для Вас это самый неудобный, дорогой и неоперативный способ получения журнала, но он необходим для поддержки маркетинговых усилий наших дистрибуторов в их регионах.

## 2. Вы готовы подключиться к распространению РС-РЕВЮ.

2.1. В этом случае для обладания свежим выпуском Вам придется предварительно перечислить нам установленную цену за мастердиск и за лицензию на копирование.

2.2. Далее Вы можете делать все, что хотите, Вы можете копировать журнал для всех желающих на дискету заказчика за назначенную Вами плату. Ваши координаты войдут в список официально зарегистрированных дистрибуторов и к Вам будут обращаться читатели.

Вы можете распространять этот журнал и на своих дискетах, используя для этого торговые возможности других фирм.

Вы можете заниматься этим в порядке предпринимательской деятельности без образования юридического лица, что предусмотрено действующим законодательством.

Вы можете специально для этого зарегистрировать фирму или использовать уже имеющуюся и заключить с нами договор.

2.3. Не исключено, что Вы захотите стать Генеральным дистрибутором в своем городе (области, республике) и заключить с нами эксклюзивный договор, согласно которому Вы будете единственным дистрибутором в своем регионе.

Это будет возможно, но не всегда. Мы пойдем на такой договор только в том случае, если Вы в своем регионе будете первым. Разорвать уже сложившиеся связи с партнерами мы конечно же не сможем. И, разумеется, такая генеральная лицензия будет стоить дороже. Это будет зависеть от размера региона. Генеральная лицензия на Эстонию будет стоить дороже, чем на один город Нарву.

Для Москвы и Московской области Генеральная лицензия на эксклюзивное распространение не продается, но обычные лицензии продаются обычным порядком.

Мы с Вами находимся у истоков огромного дела. В настоящее время в России и СНГ установлено свыше трех миллионов IBM-совместимых компьютеров, но специалисты считают, что пока это капля в море. И до каждого компьютера Вы должны добраться. Не исключено, что Вы находитесь у истоков дела, в которое в ближайшие годы будут вовлечены сотни миллионов рублей и девяносто процентов из них - Ваши, так что решайтесь! Да, первое время может быть трудно, может быть придется побегать, покрутиться, кого-то убедить, кого-то организовать, но успех приходит только к подготовленным людям. Если у Вас есть вера в свои силы, Вы пойдете дальше вместе с нами.

Со всеми вопросами, предложениями, пожеланиями, обращайтесь по адресу:

121019, Москва, Г-19, а/я 15, А/О "ИНФОРКОМ-ПРЕСС"

ВНИМАНИЮ ЛЮБИТЕЛЕЙ ПЕРСОНАЛЬНОГО  
КОМПЬЮТЕРА "ZX-SPECTRUM"  
И СОВМЕСТИМЫХ

В 1993 году "ИНФОРКОМ" выпустил следующие книги:

1. "ZX-РЕВЮ-91", выпуски 1-12, 264 с. Полный годовой комплект "ZX-РЕВЮ", сведенный в единый том.
2. "ZX-РЕВЮ-92", выпуски 1 12, 264 с. - Полный годовой комплект "ZX-РЕВЮ", сведенный в единый том.
3. "ZX-РЕВЮ-93", выпуски 1-12, 272 с. - Полный годовой комплект "ZX-РЕВЮ", сведенный в единый том.
4. "Персональный компьютер "ZX-Spectrum". Программирование в машинных кодах и на языке АССЕМБЛЕРА, 273 с. Второе издание известной нашей книги, ранее распространявшейся в трех томах.
5. "Персональный компьютер "ZX-Spectrum". Элементарная графика, 208 с. Первый том четырехтомного сериала, посвященного реализации графических возможностей компьютера.
6. "Персональный компьютер "ZX-Spectrum". Прикладная графика, 208 с. Второй том четырехтомного сериала, посвященного реализации графических возможностей компьютера.
7. Клайв Пригмор, Пол Шрив. "30 часов БЕЙСИКа для начинающих", - пер. с англ., редакция, адаптация "ИНФОРКОМа".

В 1994 году "ИНФОРКОМ" выпускает следующие книги:

1. "Персональный компьютер "ZX-Spectrum". Динамическая графика, 208 с. Третий том четырехтомного сериала, посвященного реализации графических возможностей компьютера.
2. "Персональный компьютер "ZX-Spectrum". Дизайн Ваших программ, 208 с. Заключительный том четырехтомного сериала, посвященного реализации графических возможностей компьютера.
3. "Персональный компьютер "ZX-Spectrum". Языки программирования", 208 с. Описание "синклеровских" версий языков программирования БЕТА-БЕЙСИК, МЕГА-БЕЙСИК, ЛАЗЕР-БЕЙСИК, ФОРТ, ПАСКАЛЬ, СИНКЛЕР-ЛОГО.
4. "Периферия своими руками", 192 с. - сборник схем и описаний для доработки и расширения возможностей Вашего компьютера.
5. "БЕЙСИК для опытного пользователя", 208 с. - книга рассчитана на читателей, закончивших работу с пособием "30 часов БЕЙСИКа для начинающих".
6. "Работа с ПЗУ", 208 с. - на конкретных примерах рассмотрены возможности использования процедур ПЗУ при написании собственных программ в машинных кодах.
7. "Игровые программы своими руками", 208 с. - сборник увлекательных игровых программ для самостоятельного набора и отладки.
8. "Игровые программы своими руками - 2", 208 с.
9. "Игровые программы своими руками - 3", 208 с.

По всем вопросам приобретения указанной литературы по почте, а также по вопросам оптовых поставок обращаться письменно с вложением заполненного конверта с обратным адресом или лично на наш опорный пункт в г. Москве.

121019, Москва, Г-19, а/я 10, "ИНФОРКОМ"

ВНИМАНИЮ ЛЮБИТЕЛЕЙ ПЕРСОНАЛЬНОГО  
КОМПЬЮТЕРА "ZX-SPECTRUM"  
И СОВМЕСТИМЫХ

Вы можете приобрести литературу, выпускаемую "ИНФОРКОМом" у наших дистрибьюторов в следующих опорных пунктах:

г. Москва, ул. Новый Арбат, д.2. 19-е отделение связи. 1-ый этаж операционного зала. Здесь Вы также можете подписаться на "ZX-РЕВЮ-94".

г. Москва, радиорынок в Митино, проезд до ст. метро Тушинская или поездом до платформы Трикотажная (Рижское направление). Суббота, воскресенье 9 - 14. Киоск N J-37.

г. БЕЛГОРОД, магазин "РАДИОТОВАРЫ", ул. Ленина, 32.

г. БЕЛГОРОД, Октябрьская 84, кв. 103, студия "КОМПЬЮТЕР".

г. ВЛАДИВОСТОК, Океанский пр-кт, 140, магазин "ПАНОРАМА". Проезд трамваем до ост. Некрасовская.

г. ВОРОНЕЖ, студия компьютерных игр SAN-SAN. Магазин - салон "ЭЛЕКТРОНИКА", тел. 14-00-73,

г. ДНЕПРОПЕТРОВСК, ул. Шевченко, 34, фирма "ЭКОС".

г. ЕКАТЕРИНБУРГ, магазин "СПЕКТРУМ", Главный проспект, 99.

г. ИЖЕВСК, радиорынок "Козий Парк".

г. КЕМЕРОВО, магазин "ТЕХНИЧЕСКАЯ КНИГА", ул. Весенняя, 24.

г. КЕМЕРОВО, магазин "ОРБИТА", пр. Ленина, 133.

г. КИРОВ, "Дом Науки и Техники", магазин-салон "МАРС", ул. Производственная, 27.

г. КРАСНОЯРСК, радиорынок, проезд до ост. Затон, субб., воскр.

г. НАБЕРЕЖНЫЕ ЧЕЛНЫ, Татарстан. Новый Город, салон-магазин "ПРИНТЕР".

г. НИЖНИЙ НОВГОРОД, ИМА "Ф-ПЛЮС", магазин "ФОТОЛЮБИТЕЛЬ", ул. Горького,

16.

г. ОРЕНБУРГ, магазин "ВОЕННАЯ КНИГА", ул. Советская.

г. ПРОКОПЬЕВСК, Кемеровской обл, Тырган, магазин "ОДЕЖДА", фирма "РОНЭТ".

г. РЫБИНСК, ул. Гоголя, 1, ТТЦ "ГНОМ".

г. ХАБАРОВСК, ул. Запарина, 65, магазин "ФИЛАТЕЛИЯ".

г. ЧЕБОКСАРЫ, магазин "ЭКСПРЕСС", НПК фирма "НОВА", ул. Привокзальная, д. 6.

г. ЯРОСЛАВЛЬ, магазин "РАДИО-СПОРТ-ТУРИЗМ", Ленинградский проспект.

ВНИМАНИЮ ЛЮБИТЕЛЕЙ ПЕРСОНАЛЬНОГО  
КОМПЬЮТЕРА "ZX-SPECTRUM"  
И СОВМЕСТИМЫХ

ИНФОРКОМ ПРЕДЛАГАЕТ

Комплекс программ и описание новейшей операционной системы IS DOS.  
Поставка производится по договору с фирмой "СЛОТ" на дискетах 360 К.

1. Описание системы IS DOS. - Книга содержит полное описание работы с базовый комплектом и прикладными программами. Книга издается фирмой "СЛОТ" при техническом и финансовом содействии "ИНФОРКОМа",

2. Базовый комплект IS DOS :

- Операционная система, файловая оболочка;
- текстовый редактор;
- программы и утилиты DOS;
- работа с дисками TR DOS и MS DOS;
- упаковщики;
- программы печати текстовых и экранных файлов;
- резидентные задачи;
- драйверы клавиатуры, монитора, принтера, виртуального диска.

3,4. МУЗЫКАЛЬНЫЙ РЕДАКТОР - программа предназначена для работы с AY-8910/2 и операционной системой IS DOS. С ее помощью Вы приобретете стерео/моно звуковое сопровождение при работе всех IS DOS-совместимых программ.

ПРОГРАММА СОЗДАНИЯ РЕКЛАМНЫХ РОЛИКОВ (SHOW) - пакет программ служит для создания и показа на мониторе рекламы, а также другой информации в непрерывном режиме демонстрации. Обе программы поставляются на одной дискете.

5. ЗАПИСНАЯ КНИЖКА - пакет аналогичен записной книжке с разбивкой информации по буквам и по числам в каждом месяце. Предусмотрены широкие возможности поиска.

6. ДЕЛОВОЙ КАЛЕНДАРЬ - программа необходима для тех, кто хочет планировать свои ежедневные дела или вести дневниковые записи в объеме до 615 знаков в течение нескольких лет.

7. ФИНАНСЫ БЕЗ ПРОБЛЕМ - пакет автоматизирует бухгалтерский учет, сводя его к вводу и редактированию совершаемых операций в естественной, понятной форме. Все бухгалтерские проводки выполняются автоматически. Всегда готов оборотный баланс

8. МАТЕРИАЛЬНЫЕ ЦЕННОСТИ - пакет предназначен для учета мат.ценностей. В процессе работы ведутся автоматически акты учета и списания по месяцам.

9. СКЛАД - пакет предназначен для учета поступающих товаров, деталей и т.п.

10. МАГАЗИН - все виды учета и отчетности магазина.

11. КАРТОТЕКА - пакет предназначен для создания и ведения небольших картотек с широкими возможностями поиска, редактирования и просмотра.

ПРИМЕЧАНИЕ: Готовится к изданию книга "IS-DOS, руководство программиста". Выход из печати ожидается в феврале-марте 1994 г.

ВНИМАНИЮ ЛЮБИТЕЛЕЙ КОМПЬЮТЕРНЫХ ИГР  
для IBM-совместимых компьютеров

ОБЩЕСТВО ЗАКРЫТОГО ТИПА "ИНФОРКОМ-ПРЕСС"

начиная с 01.01.94г. приступило к выпуску ежемесячного электронного журнала "PC-REVIEW".

К 01.02.94 г. подготовлено и выпущено два выпуска объемом по 2.5 Мегабайт каждый.

В состав выпусков входят:

- авторские обзоры игровых программ с иллюстрациями;
- аналитические статьи, посвященные исследованию известных и широкораспространенных программ
- переписка с читателями: ответы на вопросы, помощь и консультации, обмен мнениями и
- обзоры зарубежных полиграфических и электронных журналов, посвященных тематике игрового программного обеспечения;
- интервью с видными производителями программного обеспечения как в России, так и за рубежом; обсуждение технологических и маркетинговых проблем;
- авторские компьютерные новеллы, написанные по мотивам известных игровых программ;
- игровые этюды (отгруженные модули состояния известных игровых программ), предоставляющие читателям возможность увлекательного исследования в поисках решения;

В качестве задач ближайшего времени "PC-REVIEW" видит помощь отечественным производителям программного обеспечения в налаживании эффективного маркетинга их продукции и становлении в России цивилизованного рынка игровых программ.

"ИНФОРКОМ-ПРЕСС" приглашает к участию над выпуском PC-REVIEW самодеятельных авторов. Принимаются авторские материалы по любой тематике, никаких требований к оформлению статей и обзоров не существует. Если Ваша статья написана грамотно и интересно, наши редакторы оформят и подготовят ее к изданию сами. Опубликованные авторские статьи оплачиваются в размере до 1.0 \$ US за 2 килобайта исходного текста (почтовым переводом в рублях по курсу ММВБ). Свои предложения и проекты направляйте по следующим каналам:

почта - 121019, Москва, Г-19, а/я 16.

факс - (095) 956-16-31

E-mail - [postmaster@ircpress.su](mailto:postmaster@ircpress.su)

Контактный телефон ДЛЯ АВТОРОВ - (090) 956-16-31.

Единственный способ распространения PC-REVIEW - через региональных дистрибьюторов. "ИНФОРКОМ-ПРЕСС" розничную продажу выпусков журнала НЕ ПРОИЗВОДИТ. Стать дистрибьюторами, получить незащищенную копию журнала и лицензию на неограниченное тиражирование одного выпуска могут все желающие, оплатившие стоимость данной лицензии. Оплата принимается в любой форме. Дистрибуция журнала предельно проста и выполняется путем неограниченного копирования на одну Дискету 5.25 HD или на три дискеты 5.25 DD.

Заявку о желании стать дистрибьютором Вы можете направить по почте, факсу, электронной почте или сообщить по телефону.

Контактный телефон для дистрибьюторов - (095) 452-31-03 (по вторникам с 10 до 17).

PC-REVIEW - это Ваше издание!