Scan, OCR & spellcheck by NUK, km, Lexys (Cybrex)
TXT version by Lexys (Cybrex)

ZX - РЕВЮ
N3,N4 1992

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

Уважаемые читатели!

Мы обращаем Ваше внимание на изменение нашего почтового адреса. Все почтовые отправления просим направлять по адресу 121019, Москва, Г-19, а/я 16, Те, кто связан с нами не первый год, знают, что с этим адресом мы когда-то начинали и сейчас возвращаемся к нему опять.

Это временная мера. Мы подготовим новый постоянный почтовый адрес и в следующем выпуске Вас оповестим.

В связи с этим мы вынуждены прекратить прием от Вас предоплаты за наши разработки. Все, что Вы сочтете нужным заказать, будет Вам высылаться наложенным платежом без предоплаты. Полностью также прекращается прием новых подписчиков. Если к концу года останутся нераспределенные экземпляры, предложим их наложенным платежом.

Мы также приносим Вам извинения за то, что Вы получите этот номер со значительной задержкой, вызванной сложностью с привлечением доступных нам по затратам полиграфических мощностей. Мы надеемся, что Вы простите нам эту задержку. Вместе с тем, мы еще раз просим Вас не беспокоиться, все 12 выпусков в 1992 Вы получите без каких-либо доплат, переподписок и т.п. Мы не исключаем возможности задержек, но все взятые обязательства выполним невзирая на известные экономические трудности страны.

СПЕКТРУМ В ШКОЛЕ

К УРОКУ ИСТОРИИ.

Эту программу можно использовать на уроке истории для проверки знаний. Вы, конечно, сами сообразите, как её можно адаптировать для других учебных дисциплин. Правильный ответ выбирается путём нажатия на клавишу от 1 до 4.

Учитель может увеличить количество вопросов по своему желанию, добавляя строки DATA (500,501,502,....9999). Следует помнить, что в последней строке DATA должно стоять "eof". (End of File - конец файла).

Мы использовали для диалога с пользователем русский шрифт, полагая, что Ваш компьютер русифицирован. Если он не имеет русифицированного ПЗУ, русифицируйте его программным путем, о чем мы неоднократно писали, например в работе "Большие возможности Вашего "Спектрума" ".

Программа будет также полезна начинающим для самостоятельного разбора, она достаточно проста для этой цели. Укажем только на небольшую особенность в строках 410 и 420.

Здесь Вам предлагается повторить игру нажать клавишу "Y". Поскольку неизвестно заранее, что именно нажмет играющий "y" или "Y", в строке 420 проверяется код символа, закрепленного за нажатой клавишей. Код "Y" равен 89, а код "y" равен 121. И в том и в другом случае выполняется переход к строке another (строка 110), после чего тест повторяется.

Вы можете усложнить эту программу по своему вкусу, например рандомизировав (сделав случайным) порядок следования вопросов. Можете ввести свою систему очков за правильный ответ с первой попытки. За правильный ответ со второй попытки и штраф за

неправильный ответ и т. п.

```
10 REM УРОК ИСТОРИИ.
20 LET data = 480:
      LET finish = 360:
      LET nextquest= 130:
      LET another=110
30 REM оформление экрана
40 BORDER 2: PAPER 7: INK 9: BRIGHT 1: CLS
50 PRINT PAPER 1; FLASH 1; AT 9,9 ;"УРОК ИСТОРИИ"
60 PAUSE 500
70 CLS
80 PRINT AT 4,2; "Эта программа служит для проверки Ваших знаний по истории. После
      каждого вопроса Вам предлагаются четыре возможных варианта ответа, только один из
      которых является верным. Выберите правильный ответ нажатием клавиши 1...4."
90 PRINT FLASH 1; AT 19,5; "Нажми любую клавишу"
100 PAUSE 0
110 REM another
120 RESTORE data: DIM r(3)
130 REM nextquest
140 CLS
150 READ a$
160 IF a$="eof" THEN GO TO finish
170 READ b$,c$,d$,e$,f$
180 PRINT AT 6,2; a$ ' '
190 PRINT TAB 4; "1. "; b$
200 PRINT TAB 4; "2. "; c$
210 PRINT TAB 4; "3. "; d$
220 PRINT TAB 4; "4. "; e$
230 FOR n=1 TO 2
240 PRINT AT 2,2; "Попытка ";n
250 PAUSE 0
260 IF INKEY$=f$ THEN PRINT PAPER 1; AT 19,13; FLASH 1; "ВЕРНО":
      LET r(n)=r(n) + 1: PAUSE 150: GO TO nextquest:
      REM: Правильный ответ
270 NEXT n
280 REM Неверный ответ после двух попыток
290 LET r(3)=r(3)+1
300 IF f$="1" THEN PRINT AT 18,7; INK 2; FLASH 1; b$
310 IF f$="2" THEN PRINT AT 18,7; INK 2; FLASH 1; c$
320 IF f$="3" THEN PRINT AT 18,7; INK 2; FLASH 1; d$
320 IF f$="4" THEN PRINT AT 18,7; INK 2; FLASH 1; e$
340 PAUSE 300
350 GO TO nextquest
360 REM finish
370 CLS
380 PRINT AT 6, 2: "Правильных ответов с первой попытки "; r(1)
390 PRINT AT 9, 2; "Правильных ответов со второй попытки "; r(2)
400 PRINT AT 15,2; "Неправильных ответов "; r(3)
410 INPUT "Попробуем еще раз?",y$
420 IF CODE y$ =89 OR CODE y$=121 THEN GO TO another
430 STOP
440 REM  ДАННЫЕ
450 REM  Введите столько дополнительных вопросов, сколько хотите в строки 503 и далее.
      За каждым вопросом должны следовать четыре альтернативных ответа и цифра,
      показывающая какой же из них является правильным. Список вопросов и ответов
      должен заканчиваться строкой DATA, к которой стоит запись "eof", как показано в
      нашем примере.
500 DATA
      "В каком году в России было отменено крепостное право ?", "1825",
      "1855","1861","1917","3"
501 DATA
      "Кто из русских царей одержал победу в Полтавской битве?",
      "Иван Грозный", "Петр Первый","Павел 1", "Николай 1","2"
```

```
502 DATA
      "Кто из русских царей был кавалером Мальтийского Ордена",
      "Иван Грозный", "Петр Первый", "Павел 1", "Николай 1" "3"
9999 DATA "eof"
```

POKES

NETHER EARTH

Наберите приведенный листинг и запустите его (RUN) перед загрузкой программы. Вы получите неограниченные ресурсы для создания боевых роботов. Будет неплохо, если Вы перед запуском этого блока отгрузите его на ленту, чтобы избежать необходимости повторного набора, если досадная опечатка испортит Вам всю работу.

```
1 BORDER 0: PAPER 0: INK 7
5 CLEAR 65535
10 PRINT AT 10,3; "START 'NETHER EARTH' TAPE"
20 LOAD "" CODE 64730
25 POKE 64753,254
30 FOR f=65024 TO 65036
40 READ a: POKE f, a: NEXT f
50 DATA 62, 18, 50 ,111, 173
60 DATA 62, 33, 50 ,71, 174
70 DATA 195, 0, 166
80 RANDOMIZE USR 64730
```

* * *

HEAD OVER HEELS

Очень рекомендуем Вам сыграть в эту замечательную программу. Ниже приведенный листинг позволит вам стать непобедимым и нормально исследовать все ее лабиринты. "ИНФОРКОМ" с удовольствием напечатает грамотно и художественно написанный отчет о Вашем путешествии.

Программа снабжена счетчиком контрольной суммы и сможет Вас предупредить, если при наборе Вы где-либо ошибетесь, но копию все же лучше сделать.

```
1 CLEAR 64500
2 LET t = 0: LET W=1
5 FOR f=32000 TO 32170
10 READ a: POKE f,a
15 LET t=t+a*w: LET w = w+1
20 NEXT f
15 IF t<>1764297 THEN PRINT "DATA ERROR": STOP
30 PRINT AT 10,1; "START 'HEAD OVER HEELS' TAPE"
50 RANDOMIZE USR 32000
100 DATA 221,33,203,92,17,234
110 DATA 6,62,255,55,205,86,5
120 DATA 48,241,243,237,94,33
130 DATA 44,125,229,33,173,98
140 DATA 229,51,51,17,163,252
150 DATA 1,22,3,33,253,94,62
160 DATA 202,237,79,195,173,98
170 DATA 33,70,125,229,33,199
180 DATA 252,229,51,51,17,209
190 DATA 252,1,232,2,33,209,252
200 DATA 62,196,237,79,195,199
210 DATA 252,33,209,252,17,209
220 DATA 138,1,92,0,237,176,33
230 DATA 228,138,34,233,138,34
240 DATA 237,138,33,218,138,34
250 DATA 245,138,33,255,138,34
260 DATA 9,139,62,195,50,29,139
270 DATA 33,116,125,34,30,139
```

```
280 DATA 195,209,138,175,50
290 DATA 166,255,62,195,50,99
300 DATA 255,33,250,250,34,100
310 DATA 255,33,145,125,17,250
320 DATA 250,1,50,0,237,176,195
330 DATA 55,255,33,0,0,34,113
340 DATA 163,33,34,25,34,115
350 DATA 163,62,33,50,120,163
360 DATA 50,123,163,49,255,255
370 DATA 195,48,112
```

* * *

INTO THE EAGLE'S NEST

Набрав этот блок Вы не только станете непобедимым, но еще и получите неограниченное количество боеприпасов и ключей от дверей замка. Запрос в строке 200 относится к неограниченному боекомплекту. Можете при желании от него и отказаться.

Строка 220 - выбор или отказ от непобедимости.

Строка 240 - выбор или отказ от неограниченного количества ключей.

```
1 CLEAR 25599
5 LET t=0: LET w=0
10 FOR f=64000 TO 64037
20 READ a: POKE f,a
30 LET t=t+a*w: LET w=w+1
40 NEXT f
50 IF t<>82517 THEN PRINT "ERROR IN DATA": STOP
100 DATA 33,14,250,17,0,91,1,50
110 DATA 0,237,176,195,0,91
120 DATA 33,255,227,17,255,255,1,0
130 DATA 128,237,184,175,58
140 DATA 32,143,58,176,160,58
150 DATA 64,158,195,0,128
190 POKE 23658,8
200 INPUT "INFINITE AMMO (Y/N)?"; a$
210 IF a$="Y" THEN POKE 64026,50
220 INPUT "INVINCIBLE (Y/N)? ":a$
230 IF a$="Y" THEN POKE 64029,50
240 INPUT "INFINITE KEYS (Y/N)?";a$
250 IF a$="Y" THEN POKE 64032,50
300 PRINT AT 10,0; "START 'INTO THE EAGLE'S NEST' TAPE"
310 LOAD "" CODE
400 POKE 58380,26
410 POKE 58392,250
420 POKE 58695,100
450 RANDOMIZE USR 58368
```

* * *

URIDIUM

Эта программа позволит Вам настроить игру на свой вкус.

Строка 110 - запрос на то, желаете ли Вы иметь бесконечное количество попыток.

120 - свободный пролет сквозь стены и т. п.

130 - игра без противников.

```
1 LET t=0: LET w=0
5 FOR f=64983 TO 65066
10 READ a: POKE f,a
11 LET t=t+a*w: LET w=w+1
12 NEXT f
15 IF t<>397017 THEN PRINT "ERROR IN DATA"
20 DATA 221,33,39,244,17
```

```
25 DATA 125,2,62,255,55
30 DATA 205,86,5,210,215
35 DATA 253,62,48,50,48
40 DATA 245,33,195,0,62
45 DATA 254,34,186,245,50
50 DATA 188,245,33,0,0
55 DATA 34,62,145,195,0
60 DATA 245,33,14,254,17
65 DATA 0,64,1,40,0,237
70 DATA 176,195,0,64,33
75 DATA 255,239,17,255,255
80 DATA 1,0,165,237,184
85 DATA 62,34,58,75,138
90 DATA 62,201,58,86,152
95 DATA 62,195,58,99,138
100 DATA 195,80,253
105 POKE 23658,8
110 INPUT "INFINITE LIVES(Y/N)?"; a$
115 IF a$="Y" THEN POKE 65051,50
120 INPUT "PASS OVER WALLS etc (Y/N)?"; a$
125 IF a$="Y" THEN POKE 65056,50
130 INPUT "NO ALIENS, LAND NOW PRINTED STRAIGHT AWAY (Y/N)?";a$
135 IF a$="Y" THEN POKE 65061,51
150 PRINT AT 10,,3;"START 'URIDIUM' GAME TAPE"
200 RANDOMIZE USR 64983
```

* * *

AMAUROTE

```
1 CLEAR 26590
2 POKE 23658,8
10 PRINT AT 10,5;"START 'AMAUROTE' TAPE"
15 LOAD ""SCREEN$: LOAD ""CODE
20 INPUT "INFINITE MONEY (Y/N)?";a$
25 IF a$="Y" THEN POKE 46381,20
30 INPUT "INFINITE BOMBS (Y/N)?"; a$
35 IF a$="Y" THEN POKE 40615,0
40 INPUT "INVINCIBLE (Y/N)?";a$
45 IF a$="Y" THEN POKE 46312,0
50 RANDOMIZE USR 26600
```

* * *

GAUNTLET

```
1 CLEAR 64999
5 LET t=0: LET w= 0
10 FOR f=65000 TO 65032
15 READ a: POKE f,a
20 LET t=t+a*w: LET w=w+1
25 NEXT f
26 IF t<>64702 THEN PRINT "ERROR IN DATA": STOP
30 DATA 221,33,218,254,17
40 DATA 81,1,62,255,55,205
50 DATA 86,5,48,241,33,1,254
60 DATA 34,57,255,243,195
70 DATA 0,255,62,201,50,82
75 DATA 184, 195, 0, 132
80 PRINT AT 10,5;"START 'GAUNTLET' TAPE"
90 RANDOMIZE USR 65000
```

BETA BASIC

Продолжение. (Начало см. на стр. 3).

Ознакомившись с общими чертами языка программирования БЕТА-БЕЙСИК 3.0, мы теперь можем перейти к подробному рассмотрению его команд и функций.

РАЗДЕЛ 2. КОМАНДЫ

Команды приведены в алфавитном порядке.

1. ALTER <атрибуты> TO атрибуты

Ключевое слово расположено на клавише "A". Команда ALTER позволяет выполнять значительные манипуляции с атрибутами экрана (INK, PAPER, BRIGHT и FLASH) для каждого знакоместа. В своей простейшей форме команда ALTER может изменять атрибуты по всему экрану, не очищая его.

Пример.

```
100 PRINT AT 10,10: "TEST": PAUSE 50: ALTER TO PAPER 1
```

Эта строка изменит цвет PAPER всех символов на экране и сделает его синим. Можно провести по всему экрану установку и какой-либо комбинации атрибутов.

```
ALTER TO PAPER 2,INK 7, FLASH 1
```

Можно производить и выборочную смену атрибутов. Для этого надо перед TO указать какие атрибуты Вы хотите поменять и на какие:

```
ALTER INK 7 TO INK 0
```

В этом примере все, что написано на экране белым цветом INK изменится на черный. Перед вами открывается возможность создавать оригинальные видеоэффекты. Например, Вы можете изобразить что-либо одинаковым цветом INK и PAPER и тогда изображение на экране будет не видно. Теперь командой ALTER Вы меняете INK или PAPER и изображение проявляется перед вами.

Попробуйте сделать вот такую головоломную комбинацию - она тоже будет работать:

```
ALTER INK 3, BRIGHT 1, PAPER 7 TO INK 5, FLASH 1
```

- но затронет только те знакоместа, в которых установлены INK=3,BRIGHT=1 и PAPER =7 одновременно.

Следующая программа продемонстрирует некоторые из эффектов техники применения ALTER. Попробуйте поэкспериментировать с этим оператором. Скорость мигания полей на экране Вы можете изменять в строках 170 и 220.

```
100 LET a=2,b=4
110 FOR I=1 TO 5
120 FOR n=1 TO 16
130 PRINT INK a; PAPER b; "XXXX"; PAPER a; INK b; "0000";
140 NEXT n
150 LET c=a, a=b, b=c
160 NEXT I
170 LET t=30
180 ALTER INK a TO INK b: PAUSE t
190 ALTER PAPER a TO INK a: PAUSE t
200 ALTER INK a TO PAPER b: PAUSE t
210 ALTER INK b TO PAPER a: PAUSE t
220 LET t=t-t/10+1
230 GO TO 180
```

2. ALTER <ссылка> TO ссылка

Ключевое слово расположено на клавише "A". Это принципиально иная разновидность команды ALTER. По этой команде программа ищет появление первого значения и заменяет его на второе. Ссылкой здесь может быть имя переменной, число или

строка символов. Например:

```
ALTER a$ TO b$
```

- заменит переменную a$ на b$;

```
ALTER count TO c
```

- заменит переменную count на переменную c, но не затронет переменные accounts, counter и т. п.

```
ALTER 1 TO 23
```

- заменит число 1 на число 23. При этом, что очень важно, изменится также и "невидимое" пятибайтное представление числа, т.е. не только на экране вместо единицы будет изображено число 23, но и в расчетах тоже будет участвовать число 23.

Однако:

```
ALTER 1 TO "23"
```

- заменит число 1 и его пятибайтное представление на пару символов "23" и расчеты в программе будут производиться неправильно, хотя то, что Вы увидите на экране в измененных строках, будет записано вроде бы правильно.

Для всех вышеприведенных примеров по команде ALTER производится поиск по программе, при этом то, что стоит в кавычках, поиску не подлежит (предполагается, что Вы шлете и заменяете название процедуры, имя переменной или число). Но, в то же время, символьная строка будет найдена в программе где угодно, даже внутри кавычек, например:

```
ALTER "break to stop" TO "any key to stop"
```

Сами же кавычки при этом не разыскиваются. Если же вам надо вместо одной из символьных строк использовать имя переменной, то оно должно быть заключено в скобки для того, чтобы команда ALTER понимала, что Вы желаете изменить не имя переменной, а ее содержимое. Например:

```
LET s$ = "execute": ALTER (s$) TO "execute"
```

Если же Вам надо использовать ALTER с символьными строками, содержащими ключевые слова, впечатайте их, воспользовавшись SYMBOL SHIFT/ENTER для того, чтобы принудительно включить курсор "K".

Вы можете использовать ALTER для быстрого удаления чего-либо из программы путем замены на пустую строку, например:

```
ALTER "word" TO ""
```

Вы, наверное знаете (а в "ZX-РЕВЮ-91" мы этот вопрос обсуждали) , что числа требуют большого расхода памяти в БЕЙСИКЕ по сравнению с переменными и выражениями. Это происходит вследствие того, что после символьного выражения числа следует еще его пятибайтный аналог. И нередко в программах производят подмену, например вместо 1 употребляют SGN PI (экономятся 5 байтов) или, скажем вместо любого числа применяют VAL "число" (экономятся 3 байта). Это можно легко сделать с помощью ALTER. Например, для чисел от 1 до 100.

```
1 LET free = MEM()
2 FOR  n=1 TO 100
3 ALTER (n) TO "VAL" + CHR$ 34 + STR$ n + CHR$ 34
4 NEXT  n
5 PRINT "saved "; MEM()+ 28 - free;" bytes"
```

Программа напечатает Вам сколько байтов ей удалось сэкономить. Переменная n в строке 5 стоит в скобках для того, чтобы по команде ALTER изменялось не имя переменной "n", а то число, которое она хранит. Номера строк взяты минимальными - от 1 до 5 и это сделано специально. Дело в том, что при такой работе ALTER возможны сокращения расхода памяти в БЕЙСИК-области и все строки могут "поехать" вниз в адресах памяти, а это недопустимо для циклов FOR... NEXT. Так что нельзя допускать, чтобы до этого цикла в программе могли бы быть какие то строки, способные измениться по ALTER.

Число "28" в строке 5 введено для того, чтобы компенсировать те затраты памяти, которые потребовались на создание нужных в этой процедуре переменных free и n, т.е. чтобы эксперимент был чистым.

И последнее замечание. Будьте очень осторожны при использовании команды ALTER в данной форме. Неаккуратной работой Вы можете легко внести в программу неисправимые изменения. Например, если Вы измените все переменные "apple" на "a", а потом

сообразите, что "a" уже использовалась в программе, то будет поздно. Вы не сможете сделать обратный ход и снова поменять "a" на "apple", поскольку при этом изменятся и те переменные "a", которые и должны быть "a". Можно прежде, чем делать такие замены, убедиться, что "a" в программе не использовалась - например командами REF или LIST REF (см. далее).

3. AUTO <номер строки> <,шаг>

Клавиша - "6".

AUTO - включает режим автоматической нумерации строк, что делает более удобным написание программ. Если значение шага не указано, предполагается, что шаг равен 10. Если при этом не указан и номер строки, с которого начнется автонумерация, то предполагается номер текущей строки (строки, в которой установлен программный курсор) плюс десять.

Режим AUTO отключается, когда номер строки менее 10 или более 9983 или при выдаче любого системного сообщения. Обычно принято выходить из AUTO нажатием и удержанием BREAK продолжительностью более секунды.

Если, находясь в режиме AUTO, Вы хотите выпустить некоторый блок строк, сотрите предложенный программой номер строки и впишите вместо него свой собственный. Тогда следующим номером, предложенным Вам, будет тот, что Вы ввели, плюс величина шага.

Примеры.

AUTO - от текущей строки + 10 с шагом через 10;

AUTO 100 - от строки с номером 100 и с шагом через 10;

AUTO 100,5 - от строки с номером 100 и с шагом через 5.

4. BREAK

Клавиши - CAPS SHIFT + SPACE в неграфическом режиме.

BREAK - не ключевое слово. Обычная команда BREAK из стандартного БЕЙСИКа вполне подходит для большинства приложений, но те, кто программируют в машинных кодах, знают, как часто программа входит в замкнутый цикл и не прерывается нажатием BREAK.

БЕТА-БЕЙСИК имеет резервную BREAK-систему. Если Вы нажмете и удержите SHIFT + SPACE более, чем в течение 1 секунды, эта система поймет, что Вы "зависли" и исполнит BREAK даже если обычная система не работает. Этим методом можно выйти из затруднений, если Вы неосторожно отключили "STOP in INPUT" или "BREAK into program" посредством команды ON ERROR.

Этот же прием выведет Вас из INPUT LINE, EDIT и AUTO.

Примечания:

1) Если Вы "вывалились" в исходное "Синклеровское" сообщение, этот метод Вам не поможет.

2) Программистам в машинных кодах, желающим воспользоваться работой резервной BREAK-системы, не следует отключать прерывания.

Система работает на прерываниях 2-го рода и они должны быть включены постоянно.

5. CLEAR число.

Эта команда должна представлять большой интерес для тех, кто работает с машинным кодом. CLEAR с числом, меньшим чем 767 перемещает указатель RAMTOP вниз на заданное этим числом количество байтов.

Почему 767? В этом числе всего три знака, поэтому Вы его никак не спутает с обычной командой CLEAR, после которой, как известно должен идти пятиразрядный адрес. Кроме того, это число легко реализуется на Ассемблере.

На экран, переменные и на стеки GOSUB, DO-LOOP и PROC эта команда не влияет.

Клавиши определенные пользователем, а также область определения окон (расположенная непосредственно выше RAMTOP) перемещаются вместе с RAMTOP. За этими определениями образуется свободное пространство, простирающееся до начала кода БЕТА-БЕЙСИКа 47070. Обратите внимание: это пространство не заполняется нулями.

Та же команда CLEAR с отрицательным числом передвинет RAMTOP и области определения клавиш пользователя и окон - вверх на заданной число байтов.

Никаких проверок на перекрытие машинного кода БЕТА-БЕЙСИКа не производится, поэтому будьте осторожны и вообще не надо пользоваться этой командой, если Вы ранее не перемещали RAMTOP вниз.

Маленький нюанс. Когда Вы опускаете RAMTOP вниз на сколько-то байтов, образуется свободная область размером столько же байтов. Она образуется непосредственно под машинным кодом БЕТА-БЕЙСИКа и вовсе не обязательно над RAMTOP, поскольку RAMTOP могла указывать в совсем иное место.

Пример позволяет Вам поэкспериментировать с этой командой.

```
100 LET ramtop = 23730
110 PRINT DPEEK (ramtop)
120 CLEAR 100
130 PRINT DPEEK (ramtop)
140 CLEAR -50
150 PRINT DPEEK (ramtop)
```

6. CLOCK число или строка

Клавиша: "C"

Этот оператор позволяет управлять часами, показания которых могут быть показаны в правом верхнем углу экрана. Показания содержат часы, минуты и секунды. Часы - в 24-часовом формате. Может быть задействован режим таймера (будильника). В этом случае в заданное время может быть выдан звуковой сигнал, а может быть включена процедура GOSUB.

Часы работают от прерываний и потому счетчик работает и тогда когда Вы пишете программу и тогда, когда Вы ее запускаете. Единственное, когда часы не работают - тогда, когда идет загрузка/выгрузка, исполняется BEEP или работает теневая периферия типа INTERFACE I.

Параметром, определяющим в каком режиме работают часы, является число, стоящее после оператора. Это число может быть в диапазоне от 0 до 7 и имеет следующее содержание:

Число	Переход GOSUB	Звуковой сигнал	Экран
0	НЕТ	ВЫКЛ	ВЫКЛ
1	НЕТ	ВЫКЛ	ВКЛ
2	НЕТ	ВКЛ	ВЫКЛ
3	НЕТ	ВКЛ	ВКЛ
4	ДА	ВЫКЛ	ВЫКЛ
5	ДА	ВЫКЛ	ВКЛ
6	ДА	ВКЛ	ВЫКЛ
7	ДА	ВКЛ	ВКЛ

Часы начнут работать сразу после загрузки БЕТА-БЕЙСИКа. Начальная установка "00:00:00". Выдать показание на экран можно командой CLOCK 1. Установка конечно не будет соответствовав истинному времени, но поправить дело можно командой:

CLOCK строка , - например:

```
CLOCK "09:29:55"
```

Здесь разряды отделены двоеточием, хотя в этом нет никакой необходимости. При вводе времени для установки, компьютер воспринимает только шесть цифр, а все символы-разделители (кроме символа "a") игнорирует. Если в Вашем вводе будут только пять цифр или меньше, он просто заменит недостающие нулем, например:

```
CLOCK "xyz10"
```

Здесь XYZ будут восприняты как символы - разделители и проигнорированы, а "10" - как установка первых двух разрядов. В результате получится: "10:00:00". Символ "a", о котором мы упоминали, как об исключении, служит для установки таймера: "A06:20" -

установит будильник на 6 часов 20 минут. По достижении этого времени включится звуковой сигнал, если его режим был включен командой CLOCK 2,3,6 или 7.

Можно сделать и так, что в данное время программа выполнит переход GOSUB к назначенной подпрограмме если режим был включен командой CLOCK 4,5,6 или 7. Правда такой переход возможен только в состоянии работающей программы. Если Вы в этот момент выполняете ввод или редактирование. Вашу работу прерывать компьютер не будет.

Подпрограмма, которая вызывается, может быть любой сложности и размера. Это может быть: 10 GO TO 10 - а может быть текстовой редактор или игра. По достижении заданного времени компьютер закончит ту строку, с которой он работает, а потом только сделает переход. Завершение строки может занять определенное время, особенно если это INPUT или PAUSE.

Выбор и назначение подпрограммы, к которой выполняется переход GOSUB делается тоже оператором CLOCK число.

Здесь "число" - это номер строки, к которой делается переход. Это число должно быть в интервале от 8 до 9999. Нижний предел 8 назначен, чтобы отличать это назначение от выбора режима работы, в котором параметр может быть от 0 до 7.

Другой метод ввода подпрограммы для перехода по таймеру - прямой:

```
CLOCK: оператор: оператор:... : RETURN
```

В подпрограмме обработки перехода по таймеру нельзя использовать те же имена переменных, что и в главной программе. Если Вы не хотите, чтобы их содержимое изменялось, Вы можете оформить эту подпрограмму как процедуру и назначить используемые в ней параметры как LOCAL. Если Вы хотите, чтобы эта подпрограмма выполняла сохранение данных, а в главной программе возможно использование RUN или CLEAR, то Вам надо сохранять эти данные посредством POKE в соответствующих областях памяти, неповреждаемых по RUN или CLEAR.

Возможные применения подпрограммы перехода по таймеру могут быть такими:

- Проигрывание музыкальной мелодии в заданное время (разновидность будильника);
- Перестроение экрана;
- Бой часов (чтобы узнать при этом сколько сейчас времени, т.е. сколько раз должны пробить ваши часы можно воспользоваться функцией TIME$ - см. далее)

Вы можете даже изменять масштаб времени с помощью внутренней переменной БЕТА-БЕЙСИКа, размещённой в адресе 56866. Исходно там установлен режим хода 1/50 секунды на каждый ход.

POKE 56866,58 сделает Вам режим 100 секунд в минуте, а POKE 56866,54 - вернет к нормальному режиму.

Специалисты в электронике могут организовать регулярный (скажем, каждый час или каждую минуту) прием данных от внешних устройств, например:

```
8999 STOP
9000 PRINT "Процедура включена"
9010 LET pointer=DPEEK (USR "a"): POKE pointer,IN 127
9020 LET pointer = pointer +1: IF pointer>65535 THEN LET pointer = USR "a" + 2
9030 DPOKE USR "a", pointer: LET z$ = TIME()
9040 LET hours = VAL z$ (1 TO 2), mins = VAL z$ (4 TO 5)
9050 LET mins = mins +1: IF mins = 60 THEN LET hours=hours+1, mins = 0
9060 CLOCK "a" USING$ ("OO",hours) + USING$ ("OO",mins): RETURN
```

Не запускайте эту программу через RUN, а вместо этого дайте прямую команду:

```
DPOKE USR  "a", USR "a" +2 : CLOCK 9000: CLOCK  5
```

Эта прямая команда проинициализирует указатель (pointer), расположенный в адресах USR "a" и USR "a" + 1. Строка 9000 назначена как строка перехода по таймеру. Команда CLOCK 5 делает этот переход возможным.

Для проверки задайте время срабатывания таймера: CLOCK "AXXXX" - XXXX - установка времени.

Теперь запускайте некоторую программу через RUN. Подпрограмма CLOCK будет активироваться каждую минуту и считывать данные из внешнего порта 127.

Прочитанные данные сохраняются в области графики пользователя UDG в адресах, на которые указывает указатель (pointer). Указатель постоянно наращивается, а при

достижении верхнего предела физической памяти 65535 возвращается в исходное положение. Если у Вас нет внешних устройств, подключенных к внешним портам, то можете организовать работу так, чтобы какое-нибудь другое полезное дело с помощью этой подпрограммы выполнялось через регулярные интервалы времени.

Строки 9040 - 9060 переустанавливают таймер на время, отстоящее на 1 минуту от текущего времени, потом выполняется возврат в главную программу.

7. CLS <номер окна>

Просто команда CLS без указания параметра выполняет очистку текущего окна. (Более подробно см. WINDOW). CLS с параметром, выполняет очистку окна с номером, равным параметру (если конечно это окно было задано). Если же Вы забыли определить это окно, то получите сообщение об ошибке:

`"Invalid I/O device". (Неверно задано устройство ввода/вывода)`

Команда CLS 0 - это то же самое, что и команда CLS стандартного БЕЙСИКа, т.е. она выполняет очистку экрана вне зависимости от того, какое окно в настоящий момент является активным.

8. CONTROL CODES (управляющие коды)

Управляющие коды - это специальные символы, употребляемые в командах печати PRINT и PLOT, но сами по себе они не печатаются, а служат для того, чтобы указывать где что печатать. Так что в отличие от обычных символов их нельзя увидеть - они не имеют графического изображения.

В БЕТА БЕЙСИКе есть две группы управляющих кодов. Первая служит для управления курсором и позицией печати по командам PRINT и PLOT, а вторая применяется при управлении специальными блоками экрана, обрабатываемыми по команде GET (см. ниже).

Коды управления курсором.

Коды	Наименование	Область действия
CHR$ 2	Курсор влево	экран
CHR$ 3	Курсор вправо	экран
CHR$ 4	Курсор вниз	экран
CHR$ 5	Курсор вверх	экран
CHR$ 8	Курсор влево	окно
CHR$ 9	Курсор вправо	окно
CHR$ 10	Курсор вниз	окно
CHR$ 11	Курсор вверх	окно
CHR$ 12	DELETE	окно
CHR$ 15	Добавочный ENTER	окно

Разница между кодами с номерами 2...5 и 8...11 в том, что коды, предназначенные для работы в окне, не могут вывести курсор (позицию печати) за пределы текущего окна. Это бывает очень удобным во многих случаях, например при создании текстового редактора.

Коды 2...5 не имеют этого ограничения и применяются, как правило, с командой PLOT. Фактически же PLOT сама конвертирует коды 8...11 в коды 2...5 во время своей работы.

В стандартном БЕЙСИКЕ обработка кода CHR$ 8 имеет ошибку. Так, невозможно перемещением курсора влево поднять его с нижележащих строк на вышележащие, а если он находится на самой верхней строке, то таким перемещением можно его вообще вывести за пределы экрана и войти в распечатку программы.

Здесь эта ошибка исправлена.

Код CHR$ 9 в стандартном БЕЙСИКе тоже обрабатывается с ошибкой и здесь она тоже исправлена.

Стандартный БЕЙСИК распечатывает коды 10, 11, 12 в виде вопросительного знака,

что не очень полезно. Здесь они работают так, как это должно бы быть.

В качестве примера рассмотрим, как управляющие коды, включенные в текстовые строки, позволяют выводить на экран сложные формы через PRINT или PLOT.

```
10 LET a$ = "1235" + CHR$ 8 + CHR$ 10 + "5" + CHR$ 8 + CHR$ 10 +"678" + CHR$ 8 + CHR$ 11
      + "9"
20 PRINT AT 10,10; a$
30 PAUSE  100: CLS
40 FOR n= 32 TO 255
50 PLOT n, n/2: a$:  NEXT n
```

Особенно полезным это может быть при работе с графикой пользователя - эксперименты проведите сами.

CHR$ 15 работает, как ENTER в том смысле, что позволяет прервать строку в любом месте и продолжить ее набор в новой экранной строке. Обычный ENTER послал бы строку при этом в листинг программы и работа бы с ней закончилась.

Кроме того, CHR$ 15 можно вставлять в текстовые строки, чтобы организовывать печать так, как Вам это нравится. Вводится CHR 15 одновременным нажатием CAPS SHIFT + ENTER.

Коды управления экранными блоками.

Этих кодов два: CHR$ 0, CHR$ 1.

Код CHR$ 0 говорит о том, что за ним следуют восемь байтов, определяющие графический образ элемента экрана.

Код CHR$ 1 говорит о том, что за ним следует байт атрибутов и далее - восемь байтов, определяющие графический образ элемента экрана. Эти управляющие коды совместно с кодами CHR$ 8 и 10 используются командой GET для сохранения блока экрана в качестве строковой переменной. Может быть Вам и не нужна нижеследующая информация, но кому-то она покажется интересной.

Когда команда PLOT встречает символ CHR$ 0, она понимает, что очередные 8 байтов не являются печатными символами, а задают рисунок знакоместа размером 8X8 пикселей, который и должен быть воспроизведен на экране. Это же относится и к команде PRINT, если задан CSIZE, отличный от нуля. (Для нормального экрана следует задавать CSIZE равным восьми.) Кодируется изображение знакоместа по пиксельным строкам абсолютно также, как это делается для символов графики пользователя. Изображается построенный шаблон в текущих установленных цветах INK и PAPER. Все, как с графикой пользователя UDG. Более того, Вы можете создать массив из 9-ти символьных строк, начинающихся с CHR$ 0 и использовать каждый элемент массива, как свой UDG-символ. Понятно, что при этом Вы можете иметь огромные символьные наборы.

CHR$ 1 слегка отличается, т.к. за этим управляющим кодом идет еще один байт, определяющий атрибуты. Поэтому при печати такого шаблона принимаются не текущие цветовые установки, а то, что содержится в этом байте.

Команда GET во время своей работы снимает изображение заданной области экрана, режет его на знакоместа, каждое знакоместо кодирует 8-ью или 9-ью байтами, добавляет перед каждой серией байтов CHR$ 0 или CHR$ 1, вставляет между сериями символы управления курсором и, тем самым, представляет область экрана в виде длинной строки символов, которую и запоминает в какой-то переменной.

Если хотите поэкспериментировать с управляющими кодами, создайте строку и напечатайте ее сразу всю целиком, а не по одному символу.

```
10 CSIZE 8
20 LET a$ = CHR$ 0 + CHR$ 255 + CHR$ 129 + CHR$ 129 + CHR$ 129 +CHR$ 129 + CHR$ 129 +
      CHR$ 129 + CHR$ 255
30 PRINT a$: PRINT CSIZE 16; a$: PLOT 128,88; a$
```

9. COPY строка COPY массив

Команда имеет очень близкое отношение к команде JOIN. Подробности см. в команде JOIN.

10. CSIZE ширина <, высота>

Клавиша: SHIFT + 8.

CSIZE управляет размером символов при использовании операторов PLOT, PRINT и LIST. Аналогично INK и PAPER эта команда имеет глобальный характер, когда используется в качестве самостоятельного оператора и распространяется только на один оператор, если стоит в качестве элемента в списке PRINT.

Ширина и высота задаются в единицах пикселей. Если Вы не задаете параметр высоты, то по умолчанию высота принимается равной ширине. Нижеприведенный пример показывает символы, имеющие размеры в четыре раза больше, чем стандартные (CSIZE 32). Но Вы можете сделать и символы во весь экран - CSIZE 255,176. Правда, для очень больших символов придется нажимать BREAK, чтобы остановить автоматический скроллинг экрана.

```
10 FOR n=8 TO 32 STEP 8
20 CSIZE n
30 CLS
40 PRINT "CSIZE  "; n
50 LIST
60 NEXT n
70 CSIZE 0
```

Символы больших размеров выполняются пропорциональным увеличением стандартных "Спектрумовских" символов. CSIZE 16 - увеличение в 2 раза, CSIZE 24 - в три раза и т. д. Небольшие изменения в CSIZE могут повлиять на расстояние между символами при печати, не влияя на размеры самих символов.

Попробуйте в приведенном примере в строке 10 удалить оператор STEP 8 и посмотрите, что получится.

CSIZE с параметрами 8,9,10,11 работает с шрифтом 8X8. CSIZE с параметрами 12...19 использует уже шрифт 16X16 и т.д.

Для некоторых параметров возможно, что при печати поля очередного символа будут перекрывать (затирать) рядом лежащий символ. Неплохой идеей борьбы с этим является использование при печати режима OVER 1.

Для вышеприведенного примера, можно получить интересные эффекты, если перед командой LIST дать прямые команды:

```
INVERSE 1: CSIZE 9
```

или

```
CSIZE 32,8
```

или

```
CSIZE 8,32
```

Итак, мы рассмотрели работу крупными символами, но можно поработать и с мелкими. CSIZE 3,8 позволит Вам иметь 85 символов строке, но это не очень зрелищно, разве что если Вы работаете только с малыми буквами и имеете хороший монитор.

CSIZE 4,8 дает 64 символа строке. CSIZE 6,8 и CSIZE 7,1 используют стандартные символы экономят на пробелах между ними. Если Вам надо иметь 40 символов строке, делайте так:

```
OVER 1: CSIZE 6,8
```

После этого, с помощью команды WINDOW уменьшите размер экрана по ширине до 240 пикселей и у Вас будут точно 40 символов в строке.

Подпрограмма, выполняющая печать, отличается большой ухищрённостью. Она может печатать в любой точке экрана (с координацией позиции печати до пикселя), в любом размере, согласует печать с активным в данный момент окном.

Поэтому она работает конечно медленнее, чем стандартная PRINT процедура. Для возврата к стандартной процедуре Вы можете использовать специальную команду CSIZE 0. При этом в качестве активного окна выбирается WINDOW 0 (весь экран).

После загрузки БЕТА БЕЙСИКа исходной является установка CSIZE 0.

Все прочие коды управления печатью такие как TAB, AT, запятая, PRINT, коды управления курсором - работают в соответствии с установленным значением CSIZE. Например, для режима CSIZE 4,8 Вы сможете дать такую команду, как TAB 63. Внутри

операторов PRINT и PLOT, CSIZE используется для создания временных эффектов.

```
10 PRINT CSIZE 8,16:"двойная высота"; CSIZE 8; "нормальная высота"; CSIZE 4,8; "малая
     высота"
20 PLOT CSIZE 32,16;100,100; "AS"
```

При печати символов графики пользователя есть небольшие особенности. Если ширина задана меньше, чем 6 пикселей, печатается только правая часть символа. Вы, конечно, можете подготовить свой набор символов, учитывающий этот факт.

Символы блочной графики обрабатываются как обычные символы - растягиваются, сжимаются и пр. Есть ограничения на печать блоков экрана, снятых по GET. Этот блок, как мы говорили выше, представляет собой строковую переменную, определяющую конструкцию шаблона 8X8 и взаимосвязь между соседними шаблонами.

Поэтому при печати таких блоков желательно использовать задания CSIZE, кратные восьми, в крайнем случае - 4, т.е. это 4,8,16,24 и т.д.

При использовании параметра 4 могут исчезать отдельные пиксели, но иногда для регулярных рисунков результат получается неплохим.

10. DEFAULT переменная = значение <,переменная = значение>...

Клавиша: SHIFT + 2.

DEFAULT в принципе работает, как и LET, но в отличие от него не изменяет значение переменной, если она уже существует.

```
10 LET a=10
20 DEFAULT a=20
30 DEFAULT b=30
40 PRINT a,b
```

Строка 40 напечатает Вам, что a=10, поскольку DEFAULT в строке 20 будет проигнорирован, а b=30, т.к. DEFAULT в строке 30 сработает.

DEFAULT следует понимать как выражение принять значение, если иное не задано. Как и за LET, в БЕТА-БЕЙСИКе за DEFAULT могут идти множественные определения, причем обрабатываются они независимо, одно за другим.

```
100 DEFAULT a$="abedef",zx=123,q=0
```

Этот оператор можно использовать в программе где угодно, но наиболее широкое применение он имеет в определениях процедур для того, чтобы пользователь мог опускать те или иные параметры при вызове процедуры.

11. DEFAULT = устройство

Эта разновидность оператора DEFAULT позволяет задавать по умолчанию устройство ввода/вывода. Она рассчитана на работу с ИНТЕРФЕЙСОМ-1.

Команда позволяет использовать обычные команды SAVE/LOAD/MERGE/VERIFY при работе с микродрайвом, локальной сетью, глобальной сетью через порт RS232.

То есть, благодаря ей, упрощается синтаксис команд в работе с этими видами устройств.

В состоянии поставки БЕТА-БЕЙСИК имеет установку "t", то есть на магнитофон. Примеры прочих установок:

DEFAULT = m	Микродрайв 1
DEFAULT = M1	Микродрайв 1
DEFAULT = M2	Микродрайв 2
DEFAULT = n5	Локальная сеть, станция N5
DEFAULT = B	Порт RS232, канал "B"
DEFAULT = t	Магнитофон

В качестве примера покажем, какие команды могут быть использованы после того, как была проведена установка DEFAULT = m. Все они будут работать с микродрайвом 1.

```
SAVE "test"
SAVE 1; "test"
SAVE "m"; 1; "test"
SAVE 10 TO 100; "test"
LOAD "test"
```

```
VERIFY "test"
MERGE "test"
ERASE "test"
CAT
```

При этом, чтобы использовать микродрайв 2, можно либо задать DEFAULT = m2, а можно и не задавать, а использовать:

```
SAVE 2, "test"
LOAD 2, "test"
ERASE 2, "test"
CAT 2
```

Вы можете вместо номера использовать переменную, например:

```
LET x=2: DEFAULT mx
```

Но нельзя использовать для этой цели строковую переменную.

Даже если в качестве устройства ввода/вывода назначен магнитофон, такие команды как SAVE 1; "abc" или LOAD n,"prog" будут работать с микродрайвом, поскольку заданный параметр номера однозначно указывает на то, что речь не идет о магнитофоне. Текущие параметры установки устройства будут выгружены вместе с кодом БЕТА-БЕЙСИКа, если Вы захотите выгрузить свою программу и БЕТА-БЕЙСИК вместе с ней.

Примечание: БЕТА-БЕЙСИК разрешает в командах, связанных с микродрайвом использовать запятую "," вместо точки с запятой ";".

12. DEF KEY односимвольная строка; строка

или

DEF KEY односимвольная строка; оператор: оператор: …

Клавиша: 1 (та же, что и DEF FN).

См.также LIST DEF KEY

БЕТА-БЕЙСИК позволяет присвоить любой цифровой или буквенной клавише значение символьной строки или программных строк. При этом символы могут вводиться в компьютер, а могут оставаться в нижней части экрана (в области редактирования) до нажатия ENTER.

Последний случай реализуется следующим образом: надо сделать так, чтобы последним символом Вашей строки стояло двоеточие ":" или чтобы последним оператором программной строки стоял оператор ":". Попробуйте:

```
DEF KEY  "1"; "HELLO: "
```

Теперь нажмите совместно SYMBOL SHIFT и SPACE. Курсор изменится и станет мигающей звездочкой. Если Вы теперь нажмете клавишу "1", в нижней части экрана появится надпись HELLO. Поскольку никакие другие клавиши не были переопределены, при их нажатии Вы получите их нормальные значения.

```
DEF KEY "a":PRINT "Goodbye"
```

В этом примере часть программной строки, следующая после DEF KEY "a", присваивается клавише "a" (или "A", что одно и то же). Эта строка не исполняется после ее набора. После же того, как Вы нажмете SYMB SHIFT + SPACE и затем нажмете "a", она будет введена и исполнена, поскольку она не заканчивается двоеточием.

```
DEF KEY "a"; "10 REM hello"
```

Такая строка после нажатия на клавишу "a" будет реально введена в листинг программы после того, как пройдет проверку на синтаксис.

Клавише можно сделать переприсвоение в любое время. Старое значение при этом будет переписано. Если присвоить клавише пустую строку или если после задания клавиши нет ни одного оператора, клавиша не будет иметь определения. Оператор DEF KEY ERASE уничтожит все сделанные определения, в том числе и те, которые размещены выше RAMTOP и, тем самым, защищены даже от NEW. Процедура SAVE в загрузчике БЕТА БЕЙСИКа выгрузит все определения клавиш вместе с машиннокодовой частью БЕТА-БЕЙСИКа (она производит выгрузку кода от RAMTOP до конца БЕТА-БЕЙСИКа).

Чтобы просмотреть все определения клавиш, пользуйтесь командой LIST DEF KEY. Если желаете подредактировать определение, присвоенное клавише, то можете выполнить

следующий прием. Наберите номер строки, затем нажмите номер желаемой клавиши и Вы получите редактируемую строку, которую Вы сможете оформить в оператор DEF KEY. RAMTOP автоматически понижается, когда Вы задаете клавиши. Если Вы воспользуетесь обычным CLEAR для изменения RAMTOP, то вполне может быть, что Ваши определения клавиш пропадут. С другой стороны, БЕТА-БЕЙСИК имеет другой вариант CLEAR (см. выше), с помощью которого можно без риска для определений клавиш выделять пространство для своего машинного кода выше уровня RAMTOP.

13. DEF PROC имя процедуры <параметр><,REF параметр>...

или

DEF PROC имя процедуры <DATA>

Клавиша: 1 (та же, что и DEF FN).

См. также PROC, END PROC, LOCAL, DEFAULT, LIST PROC, REF,ITEM.

Оператор DEF PROC открывает определение процедуры. Он должен быть первым оператором в программной строке (разрешаются только ведущие пробелы или предшествующие управляющие цветовые коды). Имя процедуры должно обязательно начинаться с буквы, а заканчиваться может пробелом, двоеточием, REF, ENTER или DATA. Имя процедуры может быть записано почти любыми символами, за исключением пробела, но желательно, чтобы Вы использовали буквы, цифры, знак подчеркивания "_", и, пожалуй символы "#" и "$". Буквы верхнего и нижнего регистров - эквивалентны. Процедура может иметь то же имя, что и переменная и путаница не произойдет.

За именем процедуры может идти список параметров или ключевое слово DATA. Параметры, заданные в определении процедуры, называются формальными параметрами. Это имена переменных. Когда процедура будет вызвана, то все имеющиеся в ней переменные с именами, совпадающими с формальными параметрами, будут защищены и только потом им будут присвоены значения фактических параметров, присутствующих в вызове. Когда же перед именем формального параметра стоит REF, все происходит почти так же, но в добавок по окончании работы процедуры значение формального параметра будет присвоено соответствующему фактическому параметру. В процедуре могут участвовать и массивы, но они обязательно должны быть оформлены как ссылки, то есть перед именем массива в определении процедуры обязательно должно стоять REF.

Когда вместо списка параметров стоит ключевое слово DATA, никаких переприсвоений не делается, а список фактических параметров при вызове процедуры может быть принят с использованием оператора READ и функции ITEM.

14. DELETE <номер строки> TO <номер строки>.

Клавиша 7 (та же, что и ERASE).

Команда удаляет все строки в указанном блоке программы. Если номер начальной строки опушен, предполагается минимальный, отличный от нуля. Если опущен номер конечной строки удаления, принимается номер последней строки программы.

Примеры.

DELETE TO 100 - удаляет все строки после нулевой и до строки 100 включительно;

DELETE 100 TO - удаляет строку 100 и все последующие;

DELETE 100 TO 100 - удаляет только строку 100;

DELETE 0 TO 0 - удаляет только нулевую строку;

DELETE TO - удаляет всю программу, за исключением нулевой строки.

Последний пример отличается от NEW тем, что не вычищает программные переменные. Все строки, явно указанные в операторе, должны реально существовать, иначе Вы получите сообщение об ошибке:

```
U "No such line" (нет такой строки).
```

DELETE можно включать в текст программы, но с некоторыми предосторожностями. Когда вышележащие строки программы удаляются оператором DELETE, расположенным в подпрограмме, процедуре, в цикле FOR...NEXT или DO...LOOP, программа обычно прекращает нормальную работу, поскольку запомненный адрес возврата уже не

соответствует реальным новым адресам строк. Если же оператор DELETE применяется для того, чтобы удалить самого себя из программы, он должен быть последним оператором в строке.

Возможное применение DELETE в программе - это удаление строк DATA после того, как они уже были прочитаны для того, чтобы освободить память для программных переменных, поскольку числа в строках DATA очень сильно расходуют память - по крайней мере по 8 байтов на каждое число. Программа может также удалять часть строк для того, чтобы подзагрузкой выполнить MERGE нового блока на освободившееся место.

15. DELETE имя массива <пределы>

или

DELETE строка <пределы>

Клавиша 7 (та же, что и ERASE)

Кроме удаления программных строк, DELETE может удалять массивы полностью или частично и символьные строки.

```
10 LET a$ = "123456789"
20 DELETE a$ (4 TO 7)
30 PRINT a$: REM Prints "12389"
40 DELETE a$
50 PRINT a$: REM переменная не найдена
```

Когда удаляется строковая переменная, то она перестает существовать полностью, а не просто становится пустой строкой. Команда работает одинаково для строк и для массивов. Создайте массив для эксперимента и попробуйте:

```
10 DIM a$ (10,4)
20 FOR n=1 TO 10
30 LET a$(n)=CHR$(64+n)+"xxx"
40 NEXT n
50 FOR n=1 TO length (1,"a$")
60 PRINT a$ (n)
70 NEXT n
```

В строке 50 использована функция LENGTH вместо числа 10 потому, что количество элементов в массиве будет изменяться. Теперь добавьте дополнительные строки:

```
45 DELETE  a$ (3)
45 DELETE  a$ (3 TO)
45 DELETE  a$ (TO 4)
45 DELETE  a$
```

Опять запустите программу командой RUN и посмотрите, какие элементы будут удаляться. В числовом массиве команда DELETE a(6) удалит шестой элемент, если массив одномерный или целую строку элементов, если массив двумерный.

DELETE a(3 TO 8) удалит "вырезку" из элементов одномерного массива или группу строк из двумерного. (В числовом двумерном массиве количество рядов задаете первым числом.)

Продолжение следует.

МИХАИЛЕНКО В.С.

ЗАЩИТА ПРОГРАММ

Сегодня мы продолжаем разговор о приемах защиты программ от просмотра. Начало статьи см. в "ZX-РЕВЮ" 1-2, 1992 г., стр. 9-16.

2.3.4. Управляющие коды, используемые для защиты от просмотра.

Естественно, что того описания которое приведено в разделе 2.3.3, недостаточно для полноценного использования управляющих кодов. Здесь мы более подробно рассмотрим их применение для защиты от листинга, изучая программы, взломанные известными хаккерами, а также просматривая наиболее удачные программные приемы фирменных программ.

Итак по порядку.

Код N7 EDIT не используется для целей защиты. Это управляющий код, который создает программа ввода с клавиатуры при нажатии клавиш "CAPS SHIFT" и "1" для вызова строки в область редактирования. Эта область изменяется в соответствии с размерами строки.

N8 BACKSPACE. Этот код, означающий в переводе с английского "Курсор назад" достаточно широко используется для скрытия некоторых частей строки Бейсика, для накладывания частей друг на друга с целью дезинформации и для создания необычных эффектов, один из которых был нами рассмотрен в примерах (раздел 2.3.2).

Естественно, можно использовать этот код сразу для нескольких целей одновременно. Рассмотрим теоретические способы использования BACKSPACE для скрытия информации и исполнения накладок (с практическими приемами Вы будете ознакомлены в следующей статье).

Предположим, первой строкой в Вашей программе идет строка, выполняющая реальные действия, например осуществляющая запуск программы в машинных кодах, встроенной в Бейсик (о том как сделать такую программу читайте в Главе 1. Эта программа осуществляет реальную загрузку и только она может правильно загрузить и запустить блок кодов с магнитной ленты.

Чтобы ввести пользователя в заблуждение, текст этой строки скрывают, а в следующих за ней строках ставят ловушки (естественно эти строки не скрывают - пусть все смотрят).

Ловушки могут быть самые разные от продолжения программы на Бейсике, создающей видимость загрузки до запуска программы в кодах в таком месте, где либо коды умышленно путаются с целью зависания компьютера, либо с отправкой на команду АССЕМБЛЕРа JP 0000, аналогичной RANDOMIZE USR 0, что обеспечивает перезапуск компьютерной системы). Таким образом, вся сложность подобной системы защиты сводится к тому, чтобы освоить создание невидимой строки.

Делается это следующим образом: пишется нормальная строка на Бейсике, после этого она зануляется, чтобы её невозможно было вызвать для редактирования, а потом "сжимается" до нуля влево с помощью символов BACKSPACE.

* * *

Примечание: ещё больше затруднить поиск точного адреса старта Вашей программы в кодах можно используя измененный вариант числа управляющим кодом 14, который описан ниже, а кроме этого рекомендуется искажать подлинный смысл операторов Бейсика с использованием методов, описанных в разделе "Новейшие достижения защиты".

* * *

Более подробно как это сделано показано на рис. 1:

```
 MM NN|<-|<-|<-|<-|<-|<-|<-|<-|
|<-|<-|<-|<-|<-|<-|<-|<-|<-|<-|
|<-|<-|<-|  RANDOMIZE USR ABCDE
```

РИС. 1

где MM - номер строки NN - длина строки ABCDE - номер ячейки памяти, с которой осуществляется запуск программы в кодах.

После Бейсик строки мы размещаем ровно столько символов BACKSPACE чтобы текст строки был скрыт из виду т.е. каждому символу строки соответствует символ BACKSPACE.

Сокрытие лишь части строки из виду является разновидностью данного метода и используется в аналогичных целях. В частности, в широко распространенной программе COMMANDO (фирма ELITE) - оно используется следующим образом:

```
10 CLEAR  40000
20 LOAD  ""  CODE
30 RANDOMIZE USR (неверный, т.е. ложный шаг, с которого якобы начинается программа в
      кодах.)
```

Достаточно хитро спрятан действительный адрес, по которому происходит запуск программы в кодах. Он размещен в строке 20 после команды загрузки LOAD "" CODE и "сделан невидимым" с помощью символов BACKSPACE. Для тех, кто интересуется более подробно, как это сделано, рекомендую составить программу для непосредственного просмотра ячеек памяти Бейсик-программы (дампинга памяти), разместив ее в конце программы:

```
9997 FOR I=23755 TO 30000
9998 PRINT PEEK I; " ";CHR PEEK I; " "; I
9999 NEXT I
```

Запускается эта программа командой GO TO 9997, а в случае остановки ее из-за невозможности распечатать тот или иной символ или по другим причинам, необходимо набрать с клавиатуры NEXT I и дампинг продолжится. Этот метод рекомендуется применять для просмотра содержимого любых программ, т.к. он позволяет получить истинную картину Бейсика.

Коды N9, N10 , N11 - RIGHTSPACE, DOWNSPACE и UPSPACE для защиты программ не используются, они выделены в отдельную группу т.к. генерируются программой ввода с клавиатуры для передвижения курсора в заданном направлении:

"CAPS SHIFT" +8 - -> RIGHTSPACE
"CAPS SHIFT" +6 - -> DOWNSPACE
"CAPS SHIFT" +7 - -> UPSPACE

Коды N12, N13 - DELETE и ENTER для защиты программ не используются. N14: - этот управляющий код предшествует числам в программах. Как известно, в "СПЕКТРУМЕ" при программировании на Бейсике, обычные числа являются наибольшими расточителями памяти. Это происходит потому, что фактически числа записываются в память дважды - первый раз записано число в том виде, в каком оно печатается на экране, а второй раз записано истинное значение числа, т.е. то, которое обрабатывается интерпретатором. Свидетельством того, что началась запись числа для интерпретатора и является управляющий код 14.

Введем, например, строку:

```
10 PLOT 10,9
```

и посмотрим, каким образом она запишется в память. Выглядит она так, как показано на рис. 2:

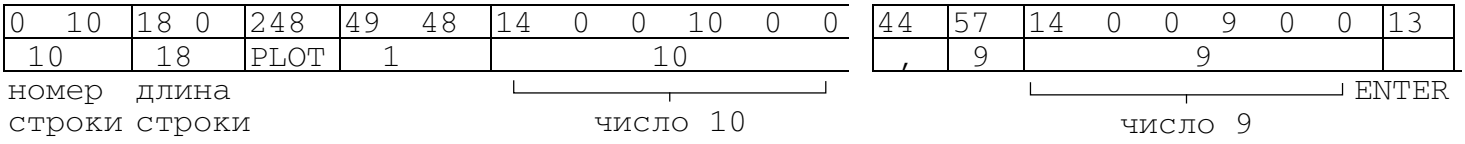


Рис 2.

Как видите, текст, который хранится в памяти, отличается от того, что изображается на экране. После последней цифры каждого числа, выступающего в тексте строки как параметр функции PLOT, интерпретатор выделил 6 байтов памяти и поместил там символ с кодом 14, а за ним - 5 байтов, в которых записано значение этого числа (число записано способом, понятным интерпретатору. Для желающих более подробно изучить способы представления чисел в "СПЕКТРУМЕ" рекомендую обратиться к методической разработке "ИНФОРКОМа" "Первые шаги в машинных кодах". Это ускоряет в определенной мере выполнение программ на Бейсике, т.к. во время выполнения интерпретатор не должен каждый раз переводить числа из алфавитно-цифрового представления (просто последовательность цифр) в 5-ти байтовое представление, пригодное для вычислений на встроенном в ПЗУ калькуляторе "СПЕКТРУМА". Готовое значение выбирается из памяти после управляющего кода 14.

Рассмотрим небольшой пример конкретной защиты программ. Во многих программах загрузчиках присутствует такая строка:

```
0 RANDOMIZE USR 0: REM...
```

На первый взгляд эта строка после запуска должна перезапустить компьютер и, соответственно, очистить всю память "СПЕКТРУМА", но этого не происходит. После более внимательного рассмотрения (с помощью PEEK-программы, которая была предложена ранее) оказывается, что после USR 0 и символа 14 нет 5 нулей, а именно так записалось бы число 0 в пятибайтной форме. Эти значения были умышленно изменены, чтобы сбить Вас с толку.

Но после управляющего символа 14 мы видим не нулевую комбинацию, а определенную последовательность, например: 0,0,218,92,0, что равнозначно числу 23770, т.е. практически функция RANDOMIZE USR не осуществит переход по адресу 0, а делает его именно на адрес 23770, а это как раз адрес байта, находящегося сразу после инструкции REM, где размещена программа в машинных кодах (о том, как разместить её там, мы писали в Главе 1).

N 15 - Этот код "СПЕКТРУМОМ" в стандартном БЕЙСИКе не используется. Использование его в других языках оговаривается в их описаниях, например в БЕТА-БЕЙСИКе 3.0.

N 16 - код управления цветом символов - INK CTRL. После этого символа обязательно наличие одного байта, уточняющего о каком цвете идет речь. Цифровая шкала цветов совпадает со стандартной шкалой цветов СПЕКТРУМА:

0-черный
1-синий
2-красный
3-пурпурный
4-зеленый
5-голубой
6-желтый
7-белый

(В разделе "Секреты ПЗУ" в прошлом году на стр. 148 "ИНФОРКОМ" указывал, что существуют также установки цвета с параметром 8 ("прозрачный") и 9 ("контрастный").

Если после этого управляющего кода будет находиться байт, содержащий значение, не совпадающее со значением одного из цветов, то это вызовет остановку компьютера с выдачей сообщения об ошибке INVALID COLOR (неверный цвет).

Создание системы команд, вызывающих остановку компьютера с выдачей сообщения об ошибке очень часто используется при защите Бейсик-программ от просмотра. В

частности, если мы используем этот метод совместно с методом зануления всех строк то это обеспечит полную защиту листинга, а кроме этого мы сможем оставлять свои сообщения в программах и закрывать доступ к ним (естественно, делать это стоит только на программах, созданных Вами лично).

Рассмотрим более подробно организацию строки Бейсика при использовании управляющего кода INK CONTROL. Как Вам уже вероятно известно, печать всех сообщений на экране "СПЕКТРУМА" осуществляет специальная процедура, встроенная в ПЗУ. Когда эта процедура встречает управляющий код 16, она анализирует следующий за ним байт и принимает его за числовое значение цвета INK, конечно если оно лежит в допустимых пределах.

После того, как код принят, все последующие символы на экране печатаются с цветом INK, соответствующим значению, стоящему после управляющего кода INK CONTROL. В строках Бейсика применение управляющих кодов позволяет экономить память, что достаточно важно в программах и приобретает особую актуальность в загрузчиках.

Рассмотрим, откуда берется эта экономия. Традиционная форма записи Бейсик-строки: INK 0.

Другой вариант - введение в строку управляющего кода INK CONTROL и за ним символа "0". Правда, набрать на клавиатуре эту комбинацию не так просто. О том, как это осуществить практически, смотрите ниже.

И в том и в другом варианте расходуется "приблизительно" одинаковое количество байтов памяти за исключением того, что интерпретатор Бейсика переводит число 0 в специальную 5-ти байтную форму, с которой он и оперирует. То есть фактически при подаче команды INK 0, расходуется не 2 байта памяти а 8:

```
INK 0 14 0 0 0 0 0 ,
```

т.е. в 4 раза больше. Комбинация же:

```
INK CONTROL 0
```

расходует всего 2 байта, поскольку здесь число 0 представлено в своем нормальном виде. Даже метод, использующий комбинацию: INK NOT PI, (что фактически аналогично INK 0) расходует 3 байта памяти, т.е. еще раз убеждаемся, что использование управляющих кодов является самым экономичным из рассмотренных методов.

Еще более существенная экономия места в памяти достигается в том случае, когда управляющие коды используются в строке оператора Бейсика PRINT вместо вставного INK. Это объясняется тем, что в этом случае приходится учитывать символ ";" (точка с запятой), которым необходимо "окаймлять" с двух сторон команду INK.

```
10 PRINT;INK 0; "ZX SPECTRUM"
```

В данном случае под вставку команды INK 0 расходуется 10 байтов:

;	INK	0	14	0	0	0	0	0	;
1	2	3	4	5	6	7	8	9	10

В случае же употребления управляющего кода INK CONTROL истратим лишь 2 байта. Фактически это применение может использоваться не только для защиты программ, но и для элементарной экономии памяти, в частности, очень существенной эта экономия оказалась в программе "MONOPOLY" (аналог известной во всем мире настольной игры), где управляющие коды используются при распечатки сообщений в операторе PRINT.

Практически используя этот и другие приемы, программистам удалось создать великолепный экземпляр игры, в которую можно играть как в одиночку, так и в паре с соперником (под игрой в одиночку я понимаю поединок со "СПЕКТРУМом", когда компьютер выступает вместо второго игрока). Этот код дает Вам возможности не только экономить память. Но и создавать эффекты, достигнуть которые иными путями будет просто невозможно. В частности, хотите ли вы, чтобы при загрузке Вашей программы после ключевого слова PROGRAM текст названия Вашей программы печатается иным (т.е. таким, который Вы заранее зададите) цветом? Если да, то и об этом поговорим на страницах данных статей.

17 управляющий код PAPER CONTROL используется точно так же, как и предыдущий

код INK CONTROL, только в данном случае числовое значение следующее после этого кода изменяет цвет не символа, а фона. В большинстве случаев управляющие коды INK CONTROL и PAPER CONTROL используются вместе для достижения каких либо эффектов. Именно совместное их использование создает наибольшую выразительность и улучшает читаемость информации, а также позволяет достигнуть наилучших цветовых эффектов. Рассмотрим более подробно теоретические аспекты совместного использования управляющих кодов INK CONTROL и PAPER CONTROL на базе ПРИМЕРА 1. (см. раздел 2.3.2).

Вкратце напомню, что тогда мы просто изучали эффекты, возникающие при использовании некоторых управляющих кодов без какой-либо предварительной теоретической проработки. Применение этих символов аналогично часто применяемым операторам изменения цвета INK и PAPER на базе оператора PRINT. Имеются, правда, существенные отличия. Первое - это существенная экономия оперативной памяти (в некоторых случаях этот объем удается сократить в пять раз). И второе отличие состоит в том, что встроенный в PRINT оператор INK или PAPER изменит цвет символов или фона лишь у текста, который должен распечатать PRINT, в то время как соответствующие управляющие символы INK CONTROL и PAPER CONTROL изменяет цвет всего следующего за ними текста. В частности, для эксперимента Вы можете вставить эти управляющие коды в один из фрагментов листинга программы способом, который будет описан в следующей статье (Раздел 2.3.5. "Практическое применение управляющих кодов для защиты). В целях защиты это может применяться, например в тех случаях, когда Вам нужно сделать текст программы невидимым), т.е. действие управляющих кодов INK CONTROL и PAPER CONTROL аналогично действию постоянных операторов INK и PAPER.

Управляющий код 18 задает или отменяет мигание - FLASH CONTROL. После этого управляющего символа следует байт параметра. Кодировка аналогична стандартной кодировке, т.е. она может иметь значение 0,1 или 8. FLASH CONTROL вызывает мерцание всех следующих после этого управляющего кода символов. FLASH CONTROL 8 оставляет в позициях символов прежние ранее установленные значения FLASH.

FLASH CONTROL 0 уничтожает действие предыдущих операторов FLASH 1 и FLASH 8, поэтому все индицируемые после этого управляющего кода символы не мерцают. Учтите, что FLASH CONTROL 1 заставляет мерцать позицию всего символа (8x8), даже в том случае, если высвечивается только одна точка.

Использование этого управляющего кода Вы могли наблюдать при загрузке программы-копировщика "COPY-COPY" в тот момент, когда на экране появляется заголовок "PROGRAM", а после него следуют два слова программы, попеременно мерцающие. Рассмотрим теоретические аспекты применения управляющего кода FLASH CONTROL. Его применение, как и других аналогичных управляющих символов, сопровождается значительной экономией памяти. В некоторых случаях за счет использования FLASH CONTROL в сравнении с FLASH удается "выиграть" до 8 байтов памяти, что при достаточно частом использовании приносит ощутимую экономию.

Для того, чтобы заставить мерцать все символы. Вам необходимо лишь установить управляющий код перед первым из данных символов:

```
FLASH CONTROL 1 <текст программы>.
```

Если же Вы желаете, чтобы мерцал лишь небольшой отрезок текста (именно это необходимо в большинстве случаев), то Вам необходимо перед этим отрезком включить мерцание, а после него - выключить:

```
<текст программы> FLASH CONTROL 1 <отрезок текста> FLASH CONTROL 0 <текст программы>.
```

После ввода вышеприведенных кодов будет мерцать лишь <отрезок текста>.

Ничем не поддержанное использование управляющего кода FLASH CONTROL не принесет эффективности в защиту Вашей программы от несанкционированного доступа. Но применение этого кода может озадачить малоопытного "хаккера", который столкнется с новым для себя приемом программирования.

Управляющий код N19 - BRIGHT CONTROL - индицирует следующие за ним символы более ярко. Это достаточно часто используется в игровых программах, но имеет очень слабую перспективу в применении для защиты программ.

После управляющего кода BRIGHT CONTROL следует байт параметра, который

соответственно придает или отменяет более яркую окраску в зависимости от своего значения:

BRIGHT CONTROL 1 дает более яркую окраску символам, выводимым после этого оператора.

BRIGHT CONTROL 8 указывает, что яркость символов в данном знакоместе должна остаться такой, какой она в нем была и ранее.

BRIGHT CONTROL 0 устраняет действие BRIGHT 1 и BRIGHT 8, и далее символы индицируются с нормальной яркостью.

Теоретические аспекты применения BRIGHT CONTROL полностью аналогичны использованию управляющего кода FLASH CONTROL.

Управляющий код N20 - INVERSE CONTROL меняет местами цвета в позиции символа, т.е. цвет INK становится цветом PAPER и наоборот. После управляющего кода INVERSE CONTROL следует байт, определяющий применение данного управляющего символа. INVERSE CONTROL 1 меняет местами цвета INK и PAPER у всех последующих символов, индицируемых на экране после этого кода.

INVERSE CONTROL 0,соответственно, возвращает первоначальные установки.

Этот управляющий символ также как и BRIGHT CONTROL почти не применяется для защиты программ. Теоретические аспекты применения INVERSE CONTROL аналогичны применению FLASH CONTROL (см. выше).

Управляющий код N21 OVER CONTROL используется тогда, когда необходимо индицировать символ, не уничтожая символа, который уже находился в этой позиции. В зависимости от следующего после OVER CONTROL байта, т.е. того значения, которое находится там, наполнение либо осуществляется, либо нет:

OVER CONTROL 0, который действует по умолчанию, уничтожает в индицируемой позиции ранее индицированный символ.

OVER CONTROL 1 индицирует символ, не уничтожая то, что находится в данном, и, таким образом, получается комбинация символов.

Использование этого управляющего кода может применяться для защиты листинга программ, в частности, совместное его использование с управляющими кодами

BACK SPACE, AT CONTROL, TAB CONTROL, позволит осуществить наложение символов таким образом, что текст программы станет полностью нечитаемым.

Для примера рассмотрим теоретические аспекты совместного применения управляющих символов OVER CONTROL и BACKSPACE для создания готического шрифта. Если на алфавитные символы накладывать какой-либо простой символ, например "/", (наклонная черта), то мы можем получить подобие готического шрифта. Для того, чтобы осуществить наложение, нам необходимо напечатать сам символ, включить режим совмещения, сдвинуть курсор на одну позицию влево управляющим кодом BACKSPACE и после этого напечатать поверх этого символа наклонную черту.

Последовательность действий:

```
B BACKSPACE OVER CNTR 1 /
```

Для пояснения приведем алгоритм на Бейсике, аналогичный вышеприведенной строке. Введем строку:

```
PRINT "B"; CHR 8;OVER 1; "/"
```

после чего мы увидим, как наклонная черта перечеркивает "B".

Здесь был рассмотрен пример, позволяющий украсить шрифт. Однако, нетрудно видеть, что можно умышленно накладывать некоторые символы один на другой так, чтобы исключить возможность прочтения листинга Вашей программы.

Код N22 (AT CONTROL) служит для печати сообщений в заданном знакоместе экрана. Компьютер получает информацию о том, в каком месте осуществлять печать символов после анализа двух байтов, следующих за рассматриваемым управляющим кодом.

Первый байт может быть в пределах от 0 до 21 и указывает номер строки, в которой будут индицироваться данные. Второй байт может лежать в пределах от 0 до 31 и указывает номер колонки, начиная с которой будут индицироваться данные. Неправильное задание данных значений вызовет останов компьютера по ошибке. (Под неправильным применением

в данном случае подразумевается выход значений за допустимые пределы). Управляющий код AT CONTROL является одним из наиболее употребительных и применяется для защиты не реже, чем INK CONTROL и PAPER CONTROL. Возможности его применения очень широки и здесь все будет зависеть от Вашей фантазии.

Рассмотрим теоретические аспекты на конкретных примерах, а к конкретному их применению вернемся позже. Например, Вы хотите, чтобы после остановки Вашей программы по команде BREAK и попытке листинга на экране появлялось заранее заданное Вами сообщение. Причем появляться оно будет в той позиции экрана, в которой Вы желаете. Если хотите, то можете создать комбинацию, которая не будет выводить никаких сопроводительных сообщений - т.е. на экране не будет номера строки и других Бейсиковских атрибутов.

Для достижения подобного эффекта нам необходимо:

Во-первых, уничтожить Бейсиковские атрибуты, используя управляющий код BACKSPACE.

Во-вторых, задать место печати сообщения в необходимом нам месте экрана, используя управляющий код AT CONTROL.

В-третьих изменить цвет INK и PAPER таким образом, чтобы дальнейший листинг программы не был бы виден.

Схема Бейсик-строки, позволяющей достигнуть этого:

```
REM |<-|<-|<-|<-|<-|<-|AT CNTR    |
    |row | col |<текст сообщения>|
    | INK CNTR |0|PAPER CNTR | 0 |
```

Здесь:

<- - упр. код BACKSPACE;

row - номер строки, в которой должно печататься Ваше сообщение;

col - номер колонки, с которой начнется Ваше сообщение хаккеру, взламывающему Вашу программу;

После того, как нам удастся создать эту строку практически, желаемый эффект будет достигнут. Подобное использование AT CNTR практически аналогично использованию AT совместно с PRINT. Но несмотря на это, применение управляющего кода позволяет получать эффекты, достижение которых иными методами было бы невозможно. Вышеприведенный пример является этому подтверждением.

Рассмотрим еще одно любопытное применение управляющих кодов. Это касается использования их в заголовке программ. Многие из Вас, вероятно наблюдали в некоторых программах появление названия программы (следующего после ключевого слова PROGRAM), высвечивающееся не в обычно принятом для этих целей месте, т.е. после слова PROGRAM, а в других местах, в частности, в центре экрана. Кроме этого, в некоторых случаях можно наблюдать появление заголовка вообще без слова PROGRAM (в этом случае название, как правило, печатается с начала строки). Все это достигается с помощью использования управляющего символа AT CONTROL.

Как Вам известно, хэдер (английское название заголовка) состоит из 17 байт. Когда мы загружаем программу, то сначала идет короткий хэдер, а после него непосредственно загружается программа. Хэдер содержит информацию о типе программы(BASIC, CODE и т.д.), о длине программы, о месте размещения программы в памяти компьютера и другие подробности, различные для каждого типа загружающихся программ). Под название программы отводится 10 байтов. Этим объясняется тот факт, что мы не можем записать на ленту программу с количеством символов в названии большим, чем 10.

А теперь давайте представим ситуацию, когда мы вместо первых трех байтов в хэдере введем управляющий код AT CONTROL (возможно введение и любого другого управляющего символа) с соответствующими значениями столбца и строки, а остальные семь байтов у нас будет занимать название программы. В результате этой операции мы получим возможность распечатывать название программы в любом месте экрана. Если же мы будем использовать другой управляющий код, то в зависимости от того, какими

функциями он управляет, будем получать соответствующие изменения. Модель данной строки:

AT CNTR	row	col							
1	2	3	4	5	6	7	8	9	10

Первые три байта занимает управляющий символ AT CONTROL со своими параметрами управлений номером строки и столбца, а байты 4-10 отводятся под название текста программы. Как видим, использование AT CONTROL несколько ограничивает наши возможности - в заголовке остается лишь семь свободных байтов для записи названия, но зато это создает неповторимый эффект. Кроме того, при удачном размещении в оставшихся 7 байтах хэдера ключевых слов Бейсика, Вам может удаться эффект создания полноценного названия, а в некоторых случаях общий объем печатаемого текста будет даже превышать 10 знакомест.

Код N23 (TAB CONTROL) используется для задания позиции данного, либо следующего столбца, начиная с которого в текущей строке печатаются указанные данные. После TAB CONTROL следует один байт значение которого должно лежать в пределах от 0 до 31.

Основные аспекты использования TAB CONTROL для защиты аналогичны подобным моментам для AT, с той разницей, что TAB CONTROL со своим параметром занимает всего 2 байта, в то время как AT CONTROL занимает 3 байта.

Это обстоятельство определяет, преимущественное отношение к TAB CONTROL в тех случаях, когда особенно важна экономия памяти. В частности, использование этого кода в хэдере дает выигрыш в 1 байт, правда несколько оскуднеет эффект применения.

* * *

Мы рассмотрели использование всех, кроме одного, управляющих кодов. Не рассмотренный код COMMA CONTROL оставлен на самый конец повествования, ввиду того, что понять его работу лучше, если знаешь уже теоретические основы применения AT CNTR и TAB CNTR.

Итак:

Код 6 - управляющий код COMMA CONTROL (или управляющая запятая оператора PRINT) используется для индицирования элементов, следующих после этого управляющего кода на смещённых на пол экрана.

Более подробно алгоритм действия COMMA CONTROL можно понять на примере использования запятой в Бейсиковском операторе PRINT. Если в операторе PRINT элементы данных разделены запятыми, то они начинают индицироваться или с начала экрана или с его середины.

Наиболее любопытным аспектом при использовании COMMA CONTROL является тот факт, что это не простое табулирование 16 символов, т.к. при этом очищаются все 16 знакомест, по которым осуществлялась табуляция. Это значит, что COMMA CONTROL можно использовать для стирания некоторых надписей на экране, особенно в тех случаях когда необходима особая экономия памяти. Одним из этих случаев является применение COMMA CONTROL в хэдере. Именно с использованием этого управляющего символа удается уничтожить ключевое слово PROGRAM, которое должно появляться при загрузке Бейсик-блока. Достигается это с помощью полного "стирания" данного слова с помощью COMMA CONTROL.

В самом деле, если в 10 байтах хэдера, отведенных под заголовок, первой пойдет комбинация управляющих символов AT CONTROL выставляющая позицию печати в начале ключевого слова PROGRAM. После этого это слово сотрется, с использованием COMMA CONTROL. Далее позиция печати переносится в любое удобное для нас место экрана, например, в его середину опять-таки с использованием AT CONTROL и уже начиная с этого

места осуществляется печать названия программы. Схема распределения данных 10 байтов хэдера приведена на рисунке:

AT CNTR	row	col		AT CNTR	row	col			
1	2	3	4	5	6	7	8	9	10

Байты 1-3 занимает управляющий символ AT CONTROL со своими атрибутами. Байт 4 занимает символ COMMA CONTROL осуществляющий табуляцию в середину строки. Байт 5-7 занимает AT CONTROL с атрибутами. И всего лишь, 3 байта 8-10 остается под название программы.

* * *

Примечание "ИНФОРКОМА" даже и в этих 3 байтах можно разместить довольно длинные названия, используя токены ключевых слов.

Например:
RETURN TO RUN
LAND MOVE (L + AND + MOVE)
DRAW CAT !
STEP OVER BORDER
TANK (TAN + BACKSPACE + K)
RUN TO POINT
GO TO NEW POINT
TABOR (TAB * BACKSPACE + OR)
OUTRUN
OVERLIST
и т.д. и т.п.

Несмотря на простоту, подобное использование управляющих кодов достаточно неэкономично, но я надеюсь, что читатель в качестве эксперимента, укрепляющего свои познания в этой области, сумеет найти более рациональные решения, сохраняющие требуемый эффект.

2.3.5. Практическое использование управляющих кодов для защиты.

Мы рассмотрели теоретические основы применения управляющих кодов. Я постарался дать исчерпывающую информацию обо всех. О том, как это у меня получилось, судить Вам, читатель. В этом разделе перед нами стоит задача освоить предложенные теоретические идеи на практике. Если Вы хорошо поняли предыдущий материал, то и этот освоите без труда. Итак, все по порядку.

Управляющие коды "СПЕКТРУМа" можно получить несколькими способами. Первый - самый легкий, понятный и доступный - использование встроенной функции CHR.

Как Вам уже вероятно известно, все существующие на клавиатуре ключевые слова и символы, а также определенные потребителем графические символы совместно с управляющими кодами образуют полный набор символов "ZX SPECTRUM". Используя CHR и номер кода символа, можно получить строчное изображение каждого символа. Такое правило будет соблюдаться для всех символов, кроме управляющих кодов. Если вместе с CHR мы наберем управляющий код, то никакой новый символ на экране не появится, а будет выполнено действие, соответсвующее данному управляющему коду.

Рассмотрим применение CHR с управляющими кодами на базе оператора PRINT. Существует несколько вариантов использования CHR. После CHR и значения указанного кода может следовать точка с запятой, например:

```
PRINT  "A" ; CHR 6; "B"
```

Этот оператор выдает индикацию на экране:

```
A       B
```

т.к. используется управляющий код COMMA CONTROL. Использовать управляющие коды CHR можно и иначе - выстраивая из них составную строку. Так, оператор:

```
PRINT "A" + CHR 6 + "B"
```
даст тот же эффект, что и приведенный ранее пример.

Как мы уже знаем, коды от 16 до 23 управляют цветами и позицией печати.

Каждый из них может использоваться в составной строке. После CHR 16 (INK CONTROL) и CHR 17 (PAPER CONTROL) должны следовать CHR с указанным параметром цвета, а CHR 18... CHR 21 (FLASH, BRIGHT, INVERSE и OVER CONTROL) должны применяться вместе с CHR 0, CHR 1 или CHR 8.

Так, команда PRINT CHR 16 + CHR 2 + CHR 17 + CHR 6 + CHR 18 + CHR 1+ "SINCLAIR" изобразит на экране надпись "SINCLAIR" мерцающую красным и желтым цветами. Как и в случае, рассмотренном ранее, каждый знак плюс может быт заменен точкой с запятой.

После CHR 22 (AT CONTROL), как мы уже знаем, должны следовать два ключевых слова CHR с параметрами, указывающими номера строки и столбца позиции печати.

Так команда
```
PRINT CHR 22 + CHR 11 + CHR 16 + "W"
```
отпечатает в середине экрана символ "W".

После CHR 23 (TAB CONTROL) могут следовать 2 символа - это дает любопытный эффект: первое указывает позицию TAB, а второе обычно равно 0.

Так команда
```
PRINT CHR 23 + CHR 16+ CHR 0+"S"
```
отпечатает в середине экрана символ "S".

Как видите, такое применение данных кодов достаточно примитивно, а техника программирования, как Вы понимаете, не стоит на месте. Был создан более эффективный способ внедрения управляющих кодов, позволяющий обойтись без CHR.

Я не располагаю информацией о том, кто и когда впервые начал использовать управляющие коды и кто впервые использовал их для защиты. Но мне известно одно, а именно: ни одна современная программа к "ZX SPECTRUM" не обходится без применения управляющих кодов в своей защите. В связи с этим, возникают два вопроса:

1. Как научиться ставить защиту использующую управляющие коды, аналогичную применяемым в фирменных программах?

2. Как научиться быстро и просто снимать эту защиту с фирменных программ?

Рассмотрим теперь практическое применение управляющих кодов. Итак по-порядку:

BACKSPACE

Незаменим, когда вам необходимо скрыть какую-либо информацию Бейсика, например номер строки или некоторые другие атрибуты. За счет обратной табуляции Вы сможете также сокрыть от посторонних глаз особо ценную информацию.

Рассмотрим конкретный пример. Предположим у Вас имеется строка вида:
```
1 REM IT'S A FANTASTIC TO USE BACKSPACE
```
а мы бы желали, чтобы во время листинга на экране не было видно ни номера строки, ни Бейсик-оператора REM. Для этого вам необходимо изменить текст исходной строки, а именно: ввести между оператором REM и исходной надписью 6 символов (не имеет значение каких), чтобы потом заменить их на управляющий код BACKSPACE. Причем между REM и вставленными символами, а также между вставленными символами не должно быть пробелов.

Теперь нам необходимо символьную комбинацию, которую мы добавили между REM и текстом заменить управляющим кодом BACKSPACE. Для этого вычислим адрес, по которому расположен первый символ данной комбинации.

Для этого используем известный нам факт, что область Бейсика начинается с адреса 23755 (это условие верно только в том случае, когда к компьютеру не подключен INTERFACE 1 с микродрайвом и некоторая другая периферия). В случае сомнений найдите этот адрес для своей системы сами, используя системную переменную PROG (23635, 2 байта):
```
PRINT PEEK 23635 + 256*PEEK 23636
```
Количество символов, которое нам необходимо пропустить, равно пяти (4 символа это 2 байта номера строки и 2 байта длина строки + код оператора REM). Найдем адрес

ячейки, в которой хранится первый символ нашей комбинации:

23755+5=23760

Проверим это, подав команду:

```
PRINT CHR PEEK 23760
```

После этого заменим зарезервированную комбинацию управляющим кодом BACKSPACE. Пододадим команды с клавиатуры:

```
FOR i=23760 TO 23765:
POKE i,8:
NEXT i
```

После чего содержимое нашей команды исчезнет с экрана, но оно останется в памяти и будет обрабатываться интерпретатором.

Для тех, кто желает оставить "надпись-памятку" для "хаккеров", рекомендую вернуться к примерам раздела 2.3.2. (самый первый шаг).

Практическое применение управляющих кодов INK CONTROL и PAPER CONTROL лучше всего описывать вместе, поскольку именно такое их использование приносит наилучший эффект. Итак, по-порядку.

В защите применение INK CNTR и PAPER CNTR наиболее эффективно, когда нам необходимо скрыть текст листинга. Достигается это с помощью подачи команд, задающих одинаковый цвет INK и PAPER. Рассмотрим более подробно, как это сделать.

Предположим, Вы желаете сделать так, чтобы после подачи команды LIST экран оставался пустым. Для этого нам необходимо первой строкой программы поставить такую строку, которая бы "уничтожала" номер строки и изменяла цвета INK и PAPER управляющими кодами INK CONTROL и PAPER CONTROL. Чтобы стереть номер строки на экране нам понадобится использовать управляющий код BACKSPACE.

Итак, для начала зарезервируем место, используя оператор REM, чтобы потом спокойно заполнять его управляющими кодами. Наберем

```
1 REM HHHHHHHHH
      9символов
```

Нам необходимо зарезервировать место именно для 9 символов т.к. первые 5 заменит BACKSPACE, а остальные 2+2 мы используем для INK и PAPER CONTROL с соответствующими параметрами таким образом, чтобы сделать одинаковыми цвета символов и фона.

Для ввода первых пяти символов BACKSPACE пододадим команду с клавиатуры:

```
FOR i=23760 TO 23764:
POKE i,8:
NEXT i.
```

Для задания черного цвета INK пододадим команды

```
POKE 23765,16: POKE 23765,0
```

Для задания черного цвета PAPER пододадим команды

```
POKE 23767,17: POKE 23768,0
```

Теперь следующие строки вводимые после этого операторы не должны быть видны. Введем первый оператор, создав новую Бейсик-строку и убедимся, что это соответствует действительности.

Теперь рассмотрим одно из наиболее любопытных применений управляющего кода AT CONTROL. Многим из Вас, наверняка доводилось встречать программы, при загрузке которых на экране появлялось лишь одно название программы (имеется ввиду, что отсутствовало слово PROGRAM), либо название программы появлялось в необычном месте экрана, например в середине. Достигалось это использованием кода AT CONTROL, а в первом случае с применением COMMA CONTROL. Итак, по-порядку.

Для того, чтобы сделать такой заголовок, нам необходимо записать на ленту хэдер, содержащий управляющие коды.

Поскольку, сначала необходимо создать такой хэдер, то возникает, вопрос: "Как это сделать?", - ввиду того, что набор управляющих кодов с клавиатуры проблематичен.

Я предлагаю сделать следующее: после того, как Вы записали свою программу на

ленту с любым названием, содержащим 10 символов, загрузить Ваш хэдер в копировщик COPY - COPY, найти там адрес ячеек, содержащих название, используя команду LIST и поменять их командой POKE, следуя нижеизложенным рекомендациям.

Для тех, кто не имеет копировщика COPY - COPY, возможен другой вариант. Одной из строк своей программы Вы делаете строку

```
nn SAVE "имя программы" LINE m
```

После этого Вам необходимо получить дампинг памяти, используя одну из предложенных в этой главе программ. Запомнив адреса ячеек памяти, в которых находятся байты с названием Вашей программы, необходимо изменить их, следуя нижеизложенным рекомендациям, после чего обратиться к этой строке командой GO TO nn и записать файл в магнитофон. Неудобство второго метода заключается в том, что созданная Вами строка nn SAVE "имя программы" LINE m, должна будет остаться в программе, т.к. при записи хэдера в общую длину программы вошла также и длина этой строки.

Итак, какие надо делать изменения, чтобы название печаталось в произвольном месте экрана? Необходимо лишь задать данную позицию, используя управляющий код AT CONTROL. Из отводящихся под название 10 байтов, 3 будет занимать AT CONTROL со своими параметрами. Ввиду этого под само название остается лишь 7 байтов.

Для того, чтобы распечатать название NEITRON в позициях AT 10,12, Вам необходимо, чтобы в 10 байтах хэдера, отведенных под название, содержалась следующая комбинация:

22	10	12	78	69	73	84	82	79	78
AT CNTR	r	c	N	E	I	T	R	O	N
1	2	3	4	5	6	7	8	9	10

Если же мы хотим создать систему команд, уничтожающую слово PROGRAM, то необходимо действовать в следующей последовательности:

1) Установить курсор в позицию AT 1,0 используя код AT CONTROL, чтобы следующей командой стереть это слово.

2) Стереть слово PROGRAM, используя управляющий код COMMA CONTROL.

3) Установить курсор в то место экрана, с которого мы желали бы распечатать текст названия программы с помощью AT CONTROL.

4) Распечатать название программы.

Таким образом, как видим, в предложенном варианте под текст названия программы остается всего 3 знакоместа. Для тех, кому этого окажется мало, можно не делать пункт 3 данного плана, что позволяет сэкономить еще 3 байта. Итак, чтобы распечатать название программы MVS в позиции экрана 8, 8, уничтожив перед этим слово PROGRAM необходимо, чтобы 10 байтов заголовка имели вид:

22	1	0	6	22	8	8	77	86	83
AT CNTR	r	c	COM CNTR	AT CNTR	r	c	M	V	S
1	2	3	4	5	6	7	8	9	10

Возможно, что экспериментируя над управляющими кодами, читателю удастся обнаружить нечто новое в этой интересной области.

Теперь некоторые советы. Выше было описано использование управляющих кодов в операторе PRINT через CHR. Я полагаю, что Вам будет удобней экспериментировать именно используя непосредственно управляющие коды.

И последнее. Запомните, пожалуйста, что байт, стоящий после управлявшего кода, содержит именно значение данной цифры, а не её ASCII код. Это правило касается абсолютно всех управляющих кодов.

ПРОГРАММА ДЛЯ СНЯТИЯ ЗАЩИТ

Предназначена для программ, которые используют управляющие коды INK CONTROL и PAPER CONTROL.

```
9990 REM Программист МИХАЙЛЕНКО ВАДИМ МРТИ 010207,1991
9991 PAPER 7: INK 0: BORDER 7: CLS
```

```
9992 FOR I=23758 TO 65000
9993 IF PEEK I =13 THEN IF PEEK (I+1)=39 AND PEEK (I+2)=6 THEN STOP
9994 IF PEEK I=13 THEN LET I=I+4
9995 IF PEEK I=16 THEN POKE (I+1),0: LET I=I+2
9996 IF PEEK I=17 THEN POKE (I+1),7: LET I=I+2
9997 NEXT I
```

Краткое описание.

Задав необходимые значения цветов INK, PAPER и BORDER, программа в цикле начинает анализировать содержимое каждой ячейки памяти и, если встречается код INK CONTROL то цвет символов принудительно задаётся чёрным, а для PAPER CONTROL - белым.

Строка 9993 следит за тем, чтобы как только программа дойдёт до самой себя (до строки 9906), произошла остановка, поэтому наличие этой строки - обязательно.

Глава 3. Методы защиты от MERGE

Среди приемов, наиболее часто применяемых для взлома программ, одним из распространенных является использование MERGE вместо команды LOAD"". Это позволяет загрузить программу в память компьютера без ее автостарта.

Таким образом, в случае наличия в программе POKES, которые защищают от BREAK, либо делают листинг программы нечитаемым, их удается разблокировать.

Автостарт должен был бы запустить программу, что позволило бы организовать защиту посредством POKES, но MERGE загружает программу без автостарта.

Вот почему одним из первоочерёдных условий защиты является создание системы команд, способных защитить Вашу программу от использования MERGE"".

Для того, чтобы уяснить себе, как действуют данные методы защиты, необходимо проанализировать выполнение функции MERGE. Итак, рассмотрим, как работает этот раздел интерпретатора Бейсика. Как Вам уже вероятно известно, встроенные в ПЗУ программы обработки LOAD, SAVE, VERIFY и MERGE работают вместе, используя многие общие процедуры. Команда MERGE очень близка по своей сути к команде LOAD, естественно с некоторыми специфическими отличиями. В частности, MERGE загружает файл Бейсика в специальный буфер. После загрузки интерпретатор начинает анализировать последовательно каждую строку загруженной программы. Если все проанализированное с точки зрения интерпретатора верно, то происходит непосредственное выполнение команды MERGE, заключающееся в соединении двух программ в одну, причем новая программа затирает строки с теми же номерами, а также переменные с теми же именами.

Но это происходит лишь в том случае, если до загрузки командой MERGE в памяти уже находилась какая-либо программа. Если же мы загружаем посредством MERGE новую программу в очищенную память, то, естественно, никакого слияния не происходит, а программа просто загружается в память, анализируется интерпретатором и после этого компьютер останавливает свою работу, позволяя нам просмотреть листинг данной программы. Так использование MERGE вместо LOAD позволяет загрузить программу без автостарта.

Быть может, теперь у нетерпеливого читателя возникнет желание просмотреть все фирменные программы, используя эту команду. Что ж, попробуйте, но необходимо заметить, что в большинстве из них стоит защита от MERGE. Необходимость такой защиты очевидна из-за описанных выше возможностей для хаккеров просматривать программы. Поэтому профессионалы достаточно широко используют данный метод защиты.

Рассмотрим теоретические аспекты работы защит от MERGE.

Как вам уже известно, работа оператора MERGE заключается в том, что он загружает программу в специальный буфер и там её анализирует. Если анализ проходит успешно, то никаких сбоев не будет, но если он проходит не "очень гладко", то компьютер просто-напросто зависает. То есть, как видим, задача программиста, разрабатывающего защиту от MERGE, состоит в том, чтобы его программа имела "встроенный дефект", который бы после

загрузки с помощью MERGE проявлял себя. В то же время необходимо отметить, что этот "дефект" не должен никак проявляться после загрузки программы командой LOAD"".

Одним из методов, позволяющих осуществить подобную защиту, является задание фальшивого номера строки параллельно с фальшивой длиной строки. Фальшивый номер и длина строки сразу же проявятся, как только интерпретатор начнет анализировать программу во время выполнения команды MERGE.

Но существует один нюанс, который тоже необходимо учитывать. Интерпретатор анализирует и ту строку программы, которую он выполняет во время работы программы. А это значит, что если после загрузки программы интерпретатору Бейсика попадается созданная нами строка, то работа компьютера будет прервана по ошибке:

```
"Ошибка в Бейсике (NONSENSE IN BASIC)"
```

Следовательно, нам необходимо создать данную строку таким образом, чтобы она никогда не обрабатывалась интерпретатором во время работы программы. Одним из методов, позволяющим сделать это, является размещение данной строки в самом конце программы.

Рассмотрим, как это осуществить на практике.

Предположим, Вами создана программа, причем одним из необходимых условий является то, чтобы номера последних шагов программы не превышали 9980, поскольку в этой области памяти мы разместим специальную программу, которая создаст вам требуемую строку защиты от MERGE.

```
9985 FOR I=23758 TO 65000
9986 IF PEEK I=13 THEN IF PEEK (I+1)=39 AND PEEK (I+2)=6 THEN POKE (I+1),255:
      POKE(I+2),255: POKE(I+3),255: POKE(I+4),255: PRINT "THE END": STOP
9987 NEXT I
9990 REM защита от MERGE
9994 REM ПРОГРАММИСТ МИХАИЛЕНКО ВАДИМ, МЕНСК, МРТИ, ГР 010207.
```

Данная программа осуществляет принудительное изменение одного из номеров содержащихся в ней же строк. Номер этой строки - 9990, так что наличие ее в программе - обязательно. Если читатель хочет оставить памятку - сообщение для взломщиков, то он может написать в этой строке после REM произвольный текст, достаточно убедительно показывающий непосвященному, что программа защищена именно Вами. Подобные комментарии постоянно оставляет после себя один из наиболее известных в нашей стране "хаккеров" BILL GILBERT, причем он использует именно этот метод защиты от MERGE"".

Программа действует следующим образом. В цикле анализируется содержимое текущей ячейки памяти. Если оно равно 13, то, следовательно, это код ENTER, а это, в свою очередь, не что иное, как окончание данной строки. А раз данная строка окончена, то, зная структуру Бейсик-программы, мы можем утверждать, что после этого кода следуют 4 байта, ответственные за номер следующей строки и за ее длину. Раз так, тогда анализируем номер строки. Если его кодовое представление равно 9990, - именно этот номер нам необходим, то тогда принудительно засылаем в ячейки Бейсика, ответственные за номер, значение 255, чтобы умышленно изменить содержащиеся там правильные значения. После того, как эта операция закончится, на экране печатается "THE END" и программа останавливается.

После того, как Вы создали "защитную" строку Вы уже не увидите ее на экране. Но она будет последней строкой и Вам необходимо строго соблюдать условие, чтобы интерпретатор Бейсика данную строку не анализировал.

После того, как предложенная программа выполнила возложенную на нее миссию - создала строку защиты от MERGE, её необходимо уничтожить, подав команды:

```
9984 ENTER
9985 ENTER
9986 ENTER  и т.д.
```

Заканчивая рассмотрение системы защиты от MERGE, хотелось бы подчеркнуть тот факт, что здесь был описан лишь один из многочисленных приемов, применяющихся для создания защитной строки на Бейсике, препятствующих применению MERGE. Но то, что эта программа производит автоматически, можно осуществлять и "вручную", если использовать специальную программу для получения "дампа" памяти, описанную в предыдущей Главе.

Существуют методы защиты, достаточно кардинально отличающиеся от описанного выше.

Мы уже говорили о том, что в БЕЙСИК-строке можно разместить программу в машинных кодах. Это используется с помощью применения оператора Бейсика REM и описано в ГЛАВЕ I. Там же указано, что для нормальной работы интерпретатора необходимо, чтобы программа в машинных кодах не содержала кода перевода строки - 13. Это объясняется тем, что интерпретатор, анализируя данную строку Бейсика, определяет ее окончание именно по коду 13 и, если он его находит, то следующие за ним 4 байта он рассматривает, как номер и длину следующей, БЕЙСИК-строки. А эти байты, естественно, не могут правильно характеризовать БЕЙСИК-строку и, следовательно, должен произойти сбой в работе интерпретатора. Это очень напоминает ситуацию, когда мы пытались сделать фальшивыми номер и длину первой идущей в программе строки. То же самое происходит и в нашей программе, если мы используем в ней машинные коды, в которых есть код 13.

В этом есть некое рациональное зерно. Преимущества налицо, если у Вас есть программа, в которой на Бейсике сформирована строка машинных кодов, где присутствует код 13 (если он отсутствует, то его можно внедрить в программу, используя специальную директиву АССЕМБЛЕРа DEFB). В этом случае Вам уже не понадобится формировать специальную строку, ответственную за защиту от MERGE, что способствует экономии оперативной памяти, а это очень немаловажно для Бейсика, особенно если это загрузчик.

В то же время, следует учитывать, что этот метод может оказаться недействительным, если программа в кодах будет стоять в первой строке БЕЙСИКа. В этом и заключена одна из сложностей применения данного метода, поскольку интерпретатор начинает анализировать программу в машинных кодах, как несколько строк БЕЙСИК-программы, разделенных кодом 13. В этом случае он может остановить свою работу и выдать код сообщения об ошибке:

```
NONSENSE IN BASIC
```

Избавиться от проблемы можно, если сделать, чтобы машинные коды шли последней строкой программы. Осуществляется это достаточно просто: необходимо при создании своей программы лишь соблюсти определенную последовательность операций, которая и приведет к желаемому результату:

- Сначала формируем строку с машинными кодами, но теперь при формировании даем ей такой номер, чтобы все строки нашей исходной программы размещались до нее.

Причем необходимо, чтобы до начала следующего пункта нашего плана эти строки были сформированы. Здесь необходимо отметить тот факт, что даже в команде запуска программы в машинных кодах (RANDOMIZE USR NN) должны присутствовать значения цифр и именно такие, чтобы они приблизительно указывали на место расположения Вашей программы в кодах. Это необходимо для того, чтобы ваша программа имела точно фиксированное место расположения в оперативной памяти компьютера. После того, как мы в следующем пункте найдем точку старта подпрограммы в кодах, нам останется лишь заменить адрес в команде RANDOMIZE USR на правильный, но количество символов, соответствующих номеру старта не должно поменяться, т.к. уменьшение или увеличение числа цифр в номере приведет к смещению на один байт нашей программы в кодах. В общем случае после команды RANDOMIZE USR должно следовать пятизначное число, указывающее на какой-либо адрес, а после обнаружения правильного значения, это число соответственно откорректируется.

Для того, чтобы обнаружить истинное место в оперативной памяти нашей программы, можно использовать несколько методов:

1 - В частности, поручить это делать компьютеру, составив соответствующую программу.

2 - Сделать просмотр памяти "вручную".

В любом из этих случаев нам понадобится вводить дополнительные строки в Бейсик, а делать это необходимо таким образом, чтобы они размещались после строки, в которой содержится подпрограмма в кодах (используя такой прием, мы не изменим местоположение этой строки с подпрограммой в оперативной памяти).

В первом из рассмотренных случаев нам понадобится составить программу, чтобы

она самостоятельно в цикле анализировала бы область Бейсика с тем, чтобы обнаружить там номер строки в кодах. При обнаружение этой строки она должна распечатать номер с учетом 4-х байтов, отводящихся под номер и длину, и с учетом пятого байта, отведенного под REM. Выше было приведено достаточное количество подобных программ анализаторов, так что я надеюсь, что читатель сам справится с этой задачей.

Второй случай аналогичен первому с той разницей, что вам придется использовать программу получения дампинга, которая также была приведена в предыдущей главе.

Естественно, что после того, как эта программа выполнит свою задачу, ее строки необходимо будет уничтожить.

(Продолжение следует).

40 ЛУЧШИХ ПРОЦЕДУР

Мы продолжаем печать книгу Дж. Хардмана и Э. Хьюзона. Сегодня вашему вниманию предлагается подробный разбор еще 15 полезных процедур для самообразования.

Начало см. "ZX-РЕВЮ" N1-2, стр. 17-28.

5.8 Сдвиг вниз на один символ.

Длина: 73

Количество переменных: 0

Контрольная сумма: 7987

Назначение: Эта программа сдвигает содержимое дисплейного файла вниз на 8 пикселей.

Вызов подпрограммы:

```
RANDOMIZE USR адрес
```

Контроль ошибок: Нет

Комментарий: Нет

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА

            LD HL,22527        33    255   87
            LD DE,22495        17    223   87
SAVE        PUSH HL            229
            PUSH DE            213
            LD C,23            14    23
NEXT_L      LD B,32            6     32
COPY_B      LD A,(DE)          26
            LD (HL),A          119
            LD A,C             121
            AND 7              230   7
            CP 1               254   1
            JR NZ,NEXT_B       32    2
            SUB A              151
            LD (DE),A          18
NEXT_B      DEC HL             43
            DEC DE             27
            DJNZ COPY_B        16    241
            DEC C              13
            JR Z,REST          40    21
            LD A,C             121
            AND 7              230   7
            CP 0               254   0
            JR Z,N_BLOCK       40    24
            CP 7               254   7
            JR NZ,NEXT_L       32    225
            PUSH DE            213
            LD DE,1792         17    0     7
            AND A              167
            SBC HL,DE          237   82
            POP DE             209
            JR NEXT_L          24    215
REST        POP DE             209
            POP HL             225
            DEC D              21
            DEC H              37
            LD A,H             124
            CP 79              254   79
```

```
            RET Z               200
            JR SAVE             24      201
N_BLOCK     PUSH HL             229
            LD HL,1792          33      0       7
            EX DE,HL            235
            AND A               167
            SBC HL,DE           237     82
            EX DE,HL            235
            POP HL              225
            JR NEXT_L           24      193
```

Как она работает:

В пару регистров HL загружается адрес последнего байта дисплейного файла, а в DE загружается адрес байта, соответствующего изображению, отстоящему на восемь линий вверх. HL и DE сохраняются на стеке. В C-регистр загружается число, на 1 меньшее, чем число строк на экране. Затем в B-регистр загружается количество байтов на одной линии дисплея - он используется, как счетчик. В аккумулятор загружается байт с адресом DE и это значение пересылается в ячейку по адресу HL. В аккумулятор загружается (для проверки) содержимое регистра C и, если оно равно 1, 9 или 17, то в ячейку по адресу DE помещается 0. HL и DE уменьшаются на единицу, указывая на следующий байт дисплея. Счетчик байтов в регистре B уменьшается и, если и он не равен 0, происходит переход к COPY_B.

Далее уменьшается счетчик строк в регистре C. Если он равен 0,происходит переход к процедуре 'REST'. Если C содержит 8 или 16, то происходит переход к процедуре 'N_BLOCK'. Если C не содержит 7 или 15, подпрограмма переходит к 'NEXT_L'. Затем из HL вычитается 1792 - теперь HL указывает на следующую треть экрана и подпрограмма переходит к 'NEXT_L'.

В процедуре REST значения DE и HL берутся из стека и из них вычитается число 256. В итоге DE и HL указывают на строку, позиция которой выше, чем та, что была в предыдущем цикле. Если HL содержит 20479, подпрограмма возвращается в BASIC, иначе происходит переход к процедуре SAVE.

В процедуре N_BLOCK, 1792 вычитается из DE - т.е. после этого DE указывает на следующий блок экрана. Подпрограмма затем переходит к NEXT_L.

5.9 Сдвиг влево на один пиксел.

Длина: 17

Количество переменных: 0

Контрольная сумма: 1828

Назначение: Сдвиг содержимого дисплейного файла на один пиксел влево.

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок: Нет

Комментарии: Эта программа осуществляет более плавное перемещение, чем сдвиг влево на один символ, но требуется вызывать ее восемь раз, чтобы переместить дисплей на один полный символ.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР           ЧИСЛА ДЛЯ ВВОДА

            LD HL,22527         33      255     87
            LD C,192            14      192
NEXT_L      LD B,32             6       32
            OR A                183
NEXT_B      RL (HL)             203     22
            DEC HL              43
            DJNZ NEXT_B         16      251
            DEC C               13
            JR NZ,NEXT_L        32      245
```

```
        RET             201
```

Как она работает:

В пару регистров HL загружается адрес последнего байта дисплейного файла, а в регистр C загружается количество линий в дисплейном файле (он используется, как счетчик линии). В регистр "B" загружается количество байтов на одной линии (он используется, как счетчик байтов). Флаг переноса устанавливается в 0.

Байт, адресованный HL, сдвигается на один бит влево, бит переноса копируется в крайний правый бит, а крайний левый бит копируется во флаг переноса. Пара регистров HL уменьшается для указания на следующий байт, и счетчик (B-регистр) уменьшается. Если и он не равен 0, подпрограмма осуществляет переход к NEXT_B, иначе уменьшается количество обрабатываемых линий, и, если оно не равно 0, подпрограмма возвращается к процедуре NEXT_L.

По окончании работы программа возвращается в BASIC.

5.10 Сдвиг вправо на один пиксел.

Длина: 17

Количество переменных: 0

Контрольная сумма: 1550

Назначение: Сдвиг содержимого дисплейного файла на один пиксел вправо.

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок: Нет

Комментарий: Эта программа осуществляет более плавное перемещение, чем сдвиг вправо на один символ, но требуется восемь раз вызывать эту программу, чтобы переместить дисплей на один полный символ.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
METKA       ACCEMБЛЕР           ЧИСЛА ДЛЯ ВВОДА

            LD HL,16384         33    0     64
            LD C,192            14    192
NEXT_L      LD B,32             6     32
            OR A                183
NEXT_B      RR (HL)             203   30
            INC HL              35
            DJNZ NEXT_B         16    251
            DEC C               13
            JR NZ,NEXT_L        32    245
            RET                 201
```

Как она работает:

В пару регистров HL загружается адрес дисплейного файла, а в C-регистр загружается количество линий на экране (он используется, как счетчик линий). В регистр "B" загружается количество байтов на одной линии (он используется, как счетчик байтов). Флаг переноса устанавливается, в 0. Байт по адресу HL сдвигается на один бит вправо, бит переноса копируется в крайний левый бит, а крайний правый бит копируется во флаг переноса. Пара регистров HL увеличивается, указывая на следующий байт, и счетчик (B-регистр) уменьшается.

Если он не равен 0, подпрограмма осуществляет переход к NEXT_B, иначе уменьшается количество обрабатываемых линий, и, если оно не равно 0, подпрограмма возвращается к NEXT_L.

Закончив работу, процедура возвращается в BASIC.

5.11 Сдвиг вверх на один пиксел.

Длина: 91

Количество переменных: 0
Контрольная сумма: 9228
Назначение: Сдвиг содержимого дисплейного файла вверх на один пиксел.
Вызов программы: RANDOMIZE USR адрес
Контроль ошибок: Нет
Комментарий: Нет

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА

            LD HL,16384        33    0     64
            LD DE,16640        17    0     65
            LD C,192           14    192
NEXT_L      LD B,32            6     32
COPY_B      LD A,(DE)          26
            LD (HL),A          119
            LD A,C             121
            CP 2               254   2
            JR NZ,NEXT_B       32    2
            SUB A              151
            LD (DE),A          18
NEXT_B      INC DE             19
            INC HL             35
            DJNZ COPY_B        16    243
            PUSH DE            213
            LD DE,224          17    224   0
            ADD HL,DE          25
            EX (SP),HL         227
            ADD HL,DE          25
            EX DE,HL           235
            POP HL             225
            DEC C              13
            LD A,C             121
            AND 7              230   7
            CP 0               254   0
            JR NZ,SUBTR        32    10
            PUSH DE            213
            LD DE,2016         17    224   7
            AND A              167
            SBC HL,DE          237   82
            POP DE             209
            JR N_BLOCK         24    14
SUBTR       CP 1               254   1
            JR NZ,N_BLOCK      32    10
            PUSH HL            229
            EX DE,HL           235
            LD DE,2016         17    224   7
            AND A              167
            SBC HL,DE          237   82
            EX DE,HL           235
            POP HL             225
N_BLOCK     LD A,C             121
            AND 63             230   63
            CP 0               254   0
            JR NZ,ADD          32    6
            LD A,7             62    7
            ADD A,H            132
            LD H,A             103
            JR NEXT_L          24    187
ADD         CP 1               254   1
            JR NZ,NEXT_L       32    183
            LD A,7             62    7
```

```
            ADD A,7             130
            LD D,A              87
            LD A,C              121
            CP 1                254     1
            JR NZ,NEXT_L        32      174
            RET                 201
```

Как она работает:

В пару регистров HL загружается адрес дисплейного файла, а в DE - адрес первого байта второй строки дисплея. В регистр "C" загружается число линий на экране. В регистр "B" загружается количество байтов на одной строке он используется как счетчик.

В аккумулятор загружается байт с адресом в DE и это значение передается в ячейку по адресу HL. В аккумулятор заносится содержимое регистра "C". Если оно равно 2, то DE указывает на нижнюю строку экрана и по этому адресу записывается 0. HL и DE увеличиваются, чтобы указать на следующий байт. Счетчик в регистре "B" уменьшается и, если он не равен 0, происходит переход к COPY_B.

224 добавляется к регистровым парам HL и DE, чтобы они указывали на следующую строку экрана. Затем уменьшается регистр C, счётчик линий. Если содержимое регистра C не кратно 8, происходит переход к SUBTR иначе 2016 вычитается из HL и происходит переход к N_BLOCK. Теперь HL указывает на следующие 8 линий.

В процедуре SUBTR, если значение (C-1) не кратно 8, происходит переход к N_BLOCK, иначе 2016 вычитается из DE, т.е. теперь DE указывает на следующие 8 линий. В процедуре N_BLOCK, если значение C-регистра кратно 64, 1792 добавляется к HL и делается переход NEXT_LINE - теперь HL указывает на следующий блок из 64 линий. В процедуре ADD, если значение (C-1) не кратно 64,1792 добавляется к DE, чтобы пара DE указывала на следующий блок 64 линий. Если C-регистр не содержит 1, то программа возвращается к N_LINE иначе - возврат в BASIC.

5.12 Сдвиг вниз на один пиксел.

Длина: 90
Количество переменных: 0
Контрольная сумма: 9862
Назначение: Сдвиг содержимого дисплейного файла вниз на один пиксел.
Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок: Нет
Комментарий: Нет

ЛИСТИНГ МАШИННЫХ КОДОВ

```
METKA       ACCEMБЛЕР           ЧИСЛА ДЛЯ ВВОДА

            LD HL,22527         33      255     87
            LD DE,22271         17      255     86
            LD C,192            14      192
NEXT_L      LD B,32             6       32
COPY_B      LD A,(DE)           26
            LD (HL),A           119
            LD A,C              121
            CP 2                254     2
            JR NZ,NEXT_B        32      2
            SUB A               151
            LD (DE),A           18
NEXT_B      DEC DE              27
            DEC HL              43
            DJNZ COPY_B         16      243
            PUSH DE             213
            LD DE,224           17      224     0
            AND A               167
```

```
            SBC HL,DE          237   82
            EX (SP),HL         227
            AND A              1     67
            SBC HL,DE          237   82
            EX DE,HL           235
            POP HL             225
            DEC C              13
            LD A,C             121
            AND 7              230   7
            CP 0               254   0
            JR NZ,ADD          32    8
            PUSH DE            213
            LD DE,2016         17    224   7
            POP DE             209
            JR N_BLOCK         24    11
            ADD CP 1           254   4
            JR NZ,N_BLOCK      32    7
            PUSH HL            229
            LD HL,2016         33    224   7
            ADD HL,DE          25
            EX DE,HL           235
            POP HL             225
N_BLOCK     LD A,C             121
            CP 0               254   0
            JR NZ,SUBTR        32    6
            LD A,H             124
            SUB 7              214   7
            LD H,A             103
            JR NEXT_L          24    188
SUBTR       CP 1               254   1
            JR NZ,NEXT_L       32    184
            LD A,D             122
            SUB 7              214   7
            LD D,A             87
            LD A,C             121
            CP 1               254   1
            JR NZ,NEXT_L       32    175
            RET                201
```

Как она работает:

В пару регистров HL загружается адрес последнего байта дисплейного файла, а в пару регистров DE загружается адрес байта линии, которая находится непосредственно над этим байтом. В регистр "C" загружается число линий экрана. В регистр "B" загружается количество байтов на одной строке дисплея - он используется, как счетчик.

В аккумулятор загружается байт из ячейки с адресом в DE, и это значение загружается в ячейку по адресу в HL. В аккумулятор загружается содержимое регистра "C" - если оно равно 2, то DE указывает на верхнюю строку экрана, в адрес которой заносится 0. HL и DE уменьшаются, чтобы указать на следующие байты. Счетчик (B-регистр) уменьшается и, если он не равен 0, происходит переход к COPY_B.

224 вычитается из регистровых пар HL и DE - теперь они указывают на следующую строку экрана. Регистр C (счетчик строк) уменьшается. Если содержимое регистра C кратно 8, т.е. выполнено 8 циклов для одной строки, происходит переход к процедуре ADD, иначе 2016 добавляется к HL и происходит переход к N_BLOCK - HL теперь указывает на следующие 8 линий.

В процедуре ADD, если значение (C-1) не делится без остатка на 8, происходит переход к N_BLOCK, в противном случае 2016 добавляется к DE, чтобы пара регистров DE указывала на следующие 8 линий. В процедуре N_BLOCK, если значение C - регистра делится без остатка на 64, из HL вычитается 1792 - HL теперь указывает на следующий блок из 64 линий, и происходит переход к NEXT_L. В процедуре SUBTR, если значение (C-1) не кратно 64, из DE вычитается 1792 - в итоге DE указывает на следующий блок из 64 линий.

Если С-регистр не содержит 1, то подпрограмма возвращается к NEXT_LINE, иначе - в BASIC.

6.ДИСПЛЕЙНЫЕ ПРОГРАММЫ

6.1 Слияние картинок

Длина: 21

Количество переменных: 1

Контрольная сумма: 1709

Назначение: Эта программа объединяет картинку, хранящуюся в ОЗУ, с текущим экраном дисплея. Атрибуты не изменяются.

Переменные:

Имя - screen_store

Длина - 2

Адрес - 23296

Комментарий: содержит адрес картинки в ОЗУ.

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок: Нет

Комментарии: Для объединения картинок должна быть использована приведенная на листинге программа, однако интересные результаты могут быть также получены заменой OR (HL) на XOR (HL) или AND (HL).

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА
            LD HL,16384        33    0     64
            LD DE,(23296)      237   91    0     91
            LD BC,6144         1     0     24
NEXT_B      LD A,(DE)          26
            OR (HL)            182
            LD (HL),A          119
            INC HL             35
            INC DE             19
            DEC BC             11
            LD A,B             120
            OR C               177
            JR NZ,NEXT_B       32    246
            RET                201
```

Как она работает.

В пару регистров HL загружается начальный адрес дисплейного файла, а в пару регистров DE - его длина. Регистровая пара BC используется в качестве счетчика.

В аккумулятор загружается байт с адресом в DE и выполняется логическое 'ИЛИ' ('OR') этого значения с байтом дисплейного файла. Результат затем помещается в область дисплея.

HL и DE перемещаются на следующую позицию, счетчик уменьшается. Если счетчик не равен 0, то подпрограмма возвращается назад для повторения процесса со следующим байтом.

6.2. Инвертирование экрана.

Длина: 18

Количество переменных: 0

Контрольная сумма: 1613

Назначение: Инвертирует все в дисплейном файле: если пиксел включен - он сбрасывается (OFF), если пиксел выключен, он устанавливается.

Вызов программы:

RANDOMIZE USR адрес

Контроль ошибок: Нет

Комментарий: Эта программа может быть использована для получения эффекта вспышки. Этот эффект усиливается, если делается вызов несколько раз и добавляется звук.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА
            LD HL,16384        33    0     64
            LD BC,6144         1     0     24
            LD D,255           22    255
NEXT_B      LD A,D             122
            SUB (HL)           150
            LD (HL),A          119
            INC HL             35
            DEC BC             11
            LD A,B             120
            OR C               177
            JR NZ,NEXT_B       32    247
            RET                201
```

Как она работает:

В пару регистров HL загружается адрес дисплейного файла, а в BC загружается его длина. В D-регистр помещается значение 255. Всякий раз, когда подпрограмма возвращается к NEXT_B, в аккумулятор загружается значение из D регистра. Этот метод предпочтительнее, чем инструкция LD A,255 т.к. LD A,D выполняется приблизительно в 2 раза быстрее, чем инструкция LD A,255. Значение байта, хранящегося в ячейке по адресу, указанному в HL, вычитается из аккумулятора, а результат загружается в тот же самый байт. Таким образом, делается инвертирование.

HL увеличивается, указывая на следующий байт, а счетчик BC, уменьшается. Если счетчик не равен 0, программа возвращается к NEXT_B. Если счетчик равен 0, программа возвращается в BASIC.

6.3 Инвертирование символа вертикально.

Длина: 20

Количество переменных: 1

Контрольная сумма: 1757

Назначение: Эта программа инвертирует символ вертикально. Например, стрелка, направленная вверх, должна стать стрелкой, направленной вниз, и наоборот.

Переменные:

Имя - chr. start

Длина - 2

Адрес - 23296

Комментарий: адрес символа в ОЗУ (ПЗУ).

Вызов программы:

RANDOMIZE USR адрес

Контроль ошибок: Нет

Комментарий: Эта программа полезна в играх, т.к. можно изменять отдельные символы, не затрагивая при этом соседние области изображения.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА

            LD HL,(23296)      42    0     91
            LD D,H             84
            LD E,L             93
```

```
            LD B,8              68
NEXT_B      LD A,(HL)           126
            INC HL              35
            PUSH AF             245
            DJNZ NEXT_B         16     251
            LD B,8              68
REPL        POP AF              241
            LD (DE),A           18
            INC DE              19
            DJNZ REPL           16     251
            RET                 201
```

Как она работает:

В пару регистров HL загружается адрес данных символа. Этот же адрес затем копируется в DE. В регистр B загружается значение 8 для использования регистра в качестве счетчика. Для каждого байта в аккумулятор загружается имеющееся в настоящий момент значение. HL увеличивается, указывая на следующий байт, а содержимое аккумулятора помещается на стек. Счетчик уменьшается, и, если он не равен 0, подпрограмма возвращается, чтобы повторить процесс для следующего байта. В регистр "B" повторно загружается значение 8, чтобы снова использовать его в качестве счетчика. Изображение символа сохранено на стеке.

Процедура REPL возвращает данные со стека на то же знакоместо, но уже в обратном порядке.

Байт за байтом берутся со стека и через аккумулятор помещаются по адресу, содержащемуся в DE. DE увеличивается, чтобы указать на следующий байт, а счетчик уменьшается. Если он не равен 0, программа возвращается к REPL. В противном случае она возвращается в BASIC.

6.4 Инвертирование символа горизонтально.

Длина: 19

Количество переменных: 1

Контрольная сумма: 1621

Назначение: Эта программа инвертирует символ горизонтально - например, стрелка, направленная влево, становится стрелкой, направленной вправо.

Переменные:

Имя - chr_start

Длина - 2

Адрес - 23296

Комментарий - адрес данных символа.

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок: Нет

Комментарий: Нет

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР           ЧИСЛА ДЛЯ ВВОДА

            LD HL,(23296)       42     0      91
            LD A,8              62     8
NEXT_B      LD B,8              6      8
NEXT_P      RR (HL)             203    30
            RL C                203    17
            DJNZ NEXT_P         16     250
            LD (HL),C           113
            INC HL              35
            DEC A               61
            JR NZ,NEXT_B        32 243
```

```
            RET             201
```

Как она работает: В пару регистров HL загружается адрес данных символа, а в аккумулятор загружается количество байтов, которые должны быть инвертированы. В регистр B загружается число битов в каждом байте он используется, как счетчик.

Байт с адресом в HL сдвигается вправо таким образом, что крайний правый бит копируется во флаг переноса. C-регистр сдвигается влево так, что флаг переноса копируется в крайний правый бит. Счетчик (B-регистр) уменьшается. Если счетчик не равен 0, происходит переход к NEXT_P для работы со следующим пикселом.

Инвертированный байт, который находится в регистре C, помещается в ячейку, из которой он был взят. HL увеличивается, указывая на следующий байт, а аккумулятор уменьшается. Если аккумулятор не равен 0, происходит переход к NEXT_BYTE, в противном случае - возврат в BASIC.

6.5 Вращение символа по часовой стрелке.

Длина: 42

Количество переменных: 1

Контрольная сумма: 3876

Назначение: Эта программа поворачивает символ на 90 градусов по часовой стрелке, например, стрелка, направленная вверх становятся направленной вправо.

Переменные:

Ими - chr_start

Адрес - 23296

Комментарий: адрес данных символа.

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок: Нет

Комментарий: Эта программа полезна в играх и для серьёзных целей, например в графике.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
METKA       АССЕМБЛЕР           ЧИСЛА ДЛЯ ВВОДА
            LD HL,(23296)       42    0     91
            LD E,128            30    128
N_BIT       PUSH HL             229
            LD C,0              14    0
            LD B,1              6     1
NEXT_B      LD A,E              123
            AND (HL)            166
            CP 0                254   0
            JR Z,NOT_S          40    3
            LD A,C              121
            ADD A,B             128
            LD C,A              79
NOT_S       SLA B               203   32
            INC HL              35
            JR NC,NEXT_B        48    242
            POP HL              225
            PUSH BC             197
            SRL E               203   59
            JR NC,N_BIT         48    231
            LD DE,7             17    7     0
            ADD HL,DE           25
            LD B,8              6     8
REPL        POP DE              209
            LD (HL),E           115
            DEC HL              43
            DJNZ REPL           16    251
```

```
RET                 201
```

Как она работает:

Каждый символ состоит из группы пикселей размера 8x8, каждый из которых может быть в состоянии ON (1) или OFF (0). Рассмотрим любой бит B2 байта B1 на Рис1. Данные хранятся в ячейке (B2,B1) в форме:

```
|N1    N3|
|N2    N4|
```

где: N1 = байт, в который пиксел (B2,B1) будет вставлен после вращения.
N2 = бит в N1, в который он будет вставлен.
N3 = значение, которое представляет текущее значение бита.
N4 = значение бита N2.

Каждый байт вращаемого символа будет сформирован добавлением значений всех битов N2, которые будут в новом байте.

1	128	2	64	3	32	4	16	5	8	6	4	7	2	8	1	1
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
1	128	2	64	3	32	4	16	5	8	6	4	7	2	8	1	2
1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	
1	128	2	64	3	32	4	16	5	8	6	4	7	2	8	1	3
2	4	2	4	2	4	2	4	2	4	2	4	2	4	2	4	
1	128	2	64	3	32	4	16	5	8	6	4	7	2	8	1	4
3	8	3	8	3	8	3	8	3	8	3	8	3	8	3	8	Байт (B1)
1	128	2	64	3	32	4	16	5	8	6	4	7	2	8	1	5
4	16	4	16	4	16	4	16	4	16	4	16	4	16	4	16	
1	128	2	64	3	32	4	16	5	8	6	4	7	2	8	1	6
5	32	5	32	5	32	5	32	5	32	5	32	5	32	5	32	
1	128	2	64	3	32	4	16	5	8	6	4	7	2	8	1	7
6	64	6	64	6	64	6	64	6	64	6	64	6	64	6	64	
1	128	2	64	3	32	4	16	5	8	6	4	7	2	8	1	8
7	128	7	128	7	128	7	128	7	128	7	128	7	128	7	128	
7		6		5		4		3		2		1		0		Бит (B2)

Рис. 1 Ключ к подпрограмме вращения символа.

В HL загружается адрес первого байта символа. В регистр E загружается значение байта, который имеет 7-й бит в состоянии ON и с 0 по 6 биты - в OFF т.е. 128. HL сохраняется в стеке. В регистр C засылается 0. Далее в него будут добавляться данные, давая новое значение для формируемого байта. В регистр B загружается значение байта, нулевой бит которого включен, а биты 1-7 - выключены т.е. это единица.

В аккумулятор загружается содержимое E-регистра (3). Это значение перемножается логически (AND) с байтом, адрес которого хранится в HL. Если результат равен 0, происходит переход к NOT_S, т.к. пиксель, адресованный регистром E и регистровой парой HL, сброшен (OFF). Если же он установлен (ON), в аккумулятор загружается имеющееся значение байта (N1). Регистр B (N4) добавляется к аккумулятору, и это значение загружается в регистр C. Регистр затем устанавливается, чтобы указать на следующий байт (B1). HL увеличивается, указывая на следующий байт (B1). Если байт N1 не завершен, подпрограмма возвращается к NEXT_B.

HL восстанавливается из стека, чтобы снова указать на первый байт символа, BC

сохраняется в стеке, чтобы запомнить значение последнего байта для завершения в C регистре. E-регистр настраивается на адрес следующего бита каждого байта. Если вращение не завершено происходит переход к N_BIT.

В DE загружается значение 7, и это значение добавляется к HL. Теперь HL указывает на последний байт данных. В регистр B загружается число байтов, которые должны быть взяты из стека. Для каждого байта новое значение копируется в E, и это значение помещается в ячейку с адресом в HL. HL уменьшается, чтобы указать на следующий байт и счетчик (B-регистр) уменьшается. Если счетчик не равен 0, происходит переход к REPL.

Программа возвращается в BASIC.

6.6 Изменение атрибута.

Длина: 21

Количество переменных:2

Контрольная сумма: 1952

Назначение: Эта программа изменяет значение атрибутов всех символов экрана на задаваемое - например, могут быть изменены цвета всех символов, или весь экран может замигать и т. д.

Переменные:

Имя - data saved

Длина - 1

Адрес - 23296

Комментарий: неизменяемые биты атрибута.

Имя - new_data

Длина - 1

Адрес - 23297

Комментарий: Новые биты, вводимые в байт атрибута.

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок: Нет

Комментарий: Отдельные биты атрибута каждого символа могут быть изменены с помощью инструкций AND и OR.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА
            LD HL,22528        33    0     88
            LD BC,768          1     0     3
            LD DE,(23296)      237   91    0     91
NEXT_B      LD A,(HL)          126
            AND E              163
            OR D               178
            LD (HL),A          119
            INC H              35
            DEC BC             11
            LD A,B             120
            OR C               177
            JK NZ,NEXT_B       32 246
            RET                201
```

Как она работает:

В пару регистров HL загружается адрес области атрибутов, а в пару регистров BC количество символов на экране, в регистр D загружается значение new_data, а в регистр E загружается значение data_saved.

В аккумулятор загружается байт с адресом, находящимся в HL, а биты устанавливаются соответственно значениям регистров D и E. Результат помещается обратно в ячейку с адресом, хранящимся в HL. HL увеличивается, указывая на следующий байт, а счетчик BC уменьшается. Если содержимое BC не равно 0, программа возвращается

к NEXT_B.

Программа возвращается в BASIC.

6.7 Смена атрибута.

Длина: 22

Количество переменных: 2

Контрольная сумма: 1825

Назначение: Эта программа ищет атрибуты с определенным значением и заменяет каждое найденное вхождение другим значением.

Переменные:

Имя - old_value

Длина - 1

Адрес - 23296

Комментарий: Значение байта, подлежащего замене.

Имя - new_value

Длина - 1

Адрес - 23297

Комментарий: Значение замещающего байта.

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок: Нет

Комментарий: Эта программа полезна для выделения областей текста и графических символов.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА
            LD HL,22528        33    0     88
            LD BC,768          1     0     3
            LD DE,(23296)      237   91    0     91
NEXT_B      LD A,(HL)          126
            CP E               187
            JR NZ,NO_CH        32    1
            LD (HL),D          114
NO_CH       INC HL             35
            DEC BC             11
            LD A,B             120
            OR C               177
            JR NZ,NEXT_B       32    245
            RET                201
```

Как она работает:

В пару регистров HL загружается адрес области атрибутов, а в BC загружается число символов на экране. В E-регистр загружается old_value, а в D-регистр - new_value. В аккумулятор загружается байт, адрес которого хранится в паре HL. Если аккумулятор хранит значение, которое эквивалентно содержимому E-регистра, то в байт с адресом, имеющимся в HL, помещается содержимое D-регистра. При этом HL увеличивается, указывая на следующий байт, а счетчик BC - уменьшается. Если BC не равно 0, происходит переход к NEXT_B, иначе программа возвращается в BASIC.

6.8 Закрашивание контура.

Длина: 263

Количество переменных: 2

Контрольная сумма: 26647

Назначение: Эта программа закрашивает область экрана, ограниченную линией пикселей.

Переменные:

Имя - X_coord
Длина - 1
Адрес - 23296
Комментарий: Координата X стартовой позиции.
Имя - Y_coord
Длина - 1
Адрес - 23297
Комментарий: Координата Y стартовой позиции.
Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок: Если координата Y>175 или POINT(X,Y)=1, то программа тотчас же возвращается в BASIC.

Комментарий: Эта программа - не перемещаемая, ее стартовый адрес - 31955. Когда закрашивается большая область сложной формы, нужно большое количество свободного пространства в ОЗУ. Если это невозможно, может произойти сбой.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
METKA       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА
            LD HL,(23296)      42    0     91
            LD A,H             124
            CP                 254   176
            RET NC             208
            CALL SUBR          205   143*125
            AND (HL)           166
            CP 0               254   0
            RET NZ             192
            LD BC,65535        1     255   255
            PUSH BC            197
RIGHT       LD HL,(23296)      42    0     91
            CALL SUBR          205   143*125
            AND (HL)           166
            CP 0               254   0
            JR NZ,LEFT         32    9
            LD HL,(23296)      42    0     91
            INC L              44
            LD (23296),HL      34    0     91
            JR NZ,RIGHT        32    236
LEFT        LD DE,0            17    0     0
            LD HL,(23296)      42    0     91
            DEC L              45
            LD (23296),HL      34    0     91
PLOT        LD HL,(23296)      42    0     91
            PUSH HL            229
            CALL SUBR          205   143*125
            OR (HL)            182
            LD (HL),A          119
            POP HL             225
            LD A,H             124
            CP 175             254   175
            JR Z,DOWN          40    44
            LD A,E             123
            CP 0               254   0
            JR NZ,RESET        32    16
            INC H              36
            CALL  SUBR         205   143*125
            AND (HL)           166
            CP 0               254   0
            JR NZ,RESET        32    7
            LD HL,(23296)      42    0     91
            INC H              36
            PUSH HL            229
```

```
          LD E,1             30    1
RESET     LD HL,(23296)      42    0     91
          LD A,E             123
          CP 1               254   1
          JR NX,DOWN         32    15
          INC H              36
          CALL SUBR          205   143*125
          AND (HL)           166
          CP 0               254   0
          JR Z,DOWN          40    6
          LD E,0             30    0
          JR DOWN            24    2
L_JUMP    JR RIGHT           24    167
DOWN      LD HL,(23296)      42    0     91
          LD A,H             124
          CP 0               254   0
          JR Z,NEXT_P        40    40
          LD A,D             122
          CP 0               254   0
          JR NZ,REST         32    16
          DEC H              37
          CALL SUBR          205   143*125
          AND (HL)           166
          CP 0               254   0
          JR NZ,REST         32    7
          LD HL,(23296)      42    0     91
          DEC H              37
          PUSH HL            229
          LD B,1             22    1
REST      LD A,D             122
          CP 1               254   1
          JR NZ,NEXT_P       32    14
          LD HL,(23296)      42    0     91
          DEC H              37
          CALL SUBR          205   143*125
          AND (HL)           166
          CP 0               254   0
          JR Z,NEXT_P        40    2
          LD D,0             22    0
NEXT_P    LD HL,(23296)      42    0     91
          LD A,L             125
          CP 0               254   0
          JR Z,RETR          40    12
          DEC L              45
          LD (23296),HL      34    0     91
          CALL SUBR          205   143*125
          AND (HL)           166
          CP 0               254   0
          JR Z,PLOT          40    129
RETR      POP HL             225
          LD (23296),HL      34    0     91
          LD A,255           62    255
          CP H               188
          JR NZ,L_JUMP       32    177
          CP 1               189
          JR NZ,L_JUMP       32    174
          RET                201
SUBR      PUSH BC            197
          PUSH DE            213
          LD A,175           62    175
          SUB H              148
          LD H,A             103
          PUSH HL            229
          AND 7              230   7
          ADD A,64           198   64
```

```
                LD C,A          79
                LD A,H          124
                RRA             203     31
                RRA             203     31
                RRA             203     31
                AND 31          230     31
                LD B,A          71
                AND 24          230     24
                LD D,A          87
                LD A,H          124
                AND 192         230     192
                LD E,A          95
                LD H,C          97
                LD A,L          125
                RRA             203     31
                RRA             203     31
                RRA             203     31
                AND 31          230     31
                LD L,A          111
                LD A,E          123
                ADD A,B         128
                SUB D           146
                LD E,A          95
                LD D,0          22      0
                PUSH HL         229
                PUSH DE         213
                POP HL          225
                ADD HL,HL       41
                ADD HL,HL       41
                ADD HL,HL       41
                ADD HL,HL       41
                ADD HL,HL       41
                POP DE          209
                ADD HL,DE       25
                POP DE          209
                LD A,E          123
                AND 7           230     7
                LD B,A          71
                LD A,8          62      8
                SUB B           144
                LD B,A          71
                LD A,1          62
ROTATE          ADD A,A         135
                DJNZ ROTATE     16      253
                RRA             230     31
                POP DE          209
                POP BC          193
                RET             201
```

Как она работает:

Эта программа вычерчивает горизонтальные линии из смежных пикселей. Назовем их строчками. Предел заполнения области строчками ограничен включенными (ON) пикселями. Каждая строчка запоминается с помощью занесения в стек координат крайнего правого пикселя этой строчки.

Запускаясь с определенных координат, программа делает заполнение в каждой строчке, отмечая позиции каждой из невыполненных строчек выше или ниже. По завершении одной строчки последнее значение отмеченных координат восстанавливается и для соответствующей строчки происходит заполнение. Этот процесс повторяется до тех пор, пока не останется незаполненных строчек.

Рис. 2 иллюстрирует технику процесса. Квадраты представляют пиксели, X обозначает стартовую позицию в пределах области штриховки, а * обозначает крайние правые пиксели строчек.

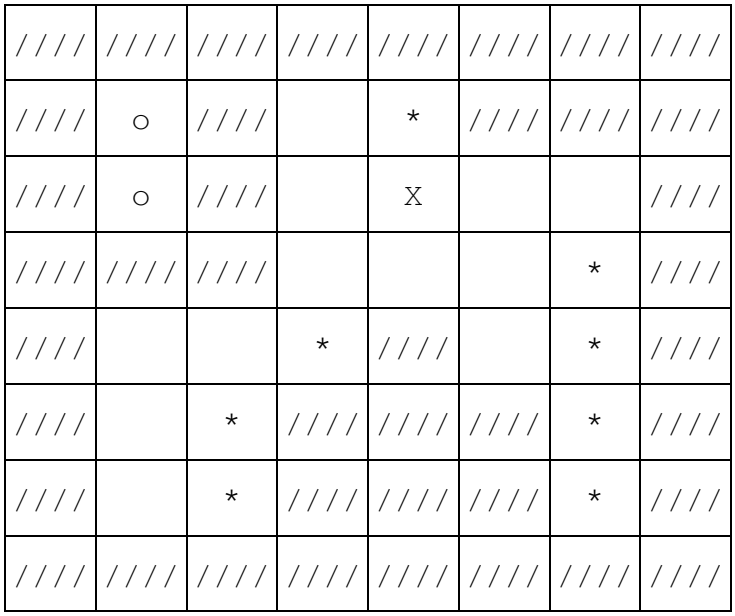


Рис. 2 Иллюстрация техники заполнения области. X - это стартовая позиция, * - начало строчек, o - оставшаяся незаштрихованная область.

Программа штрихует горизонтальную линию, содержащую стартовую позицию и сохраняет в стеке позицию начала строчки на линиях непосредственно выше и ниже. Далее она штрихует линию выше, а затем ниже, отмечая в последнем случае, что ещё 2 строчки запускаются на следующей нижней строке и т.д. Любая позиция в пределах области для штриховки может быть выбрана как стартовая позиция. Но заметим, что два пиксела, промаркированные нулями, нетронуты, т.к. они отделены от заштриховываемой области.

В регистр H загружается Y-координата, а в L-регистр - X-координата. Если значение Y больше, чем 175, программа возвращается в BASIC. Процедура SUBR вызывается, возвращая адрес бита (X,Y) в память. Если этот бит в состоянии 'ON' (включен), подпрограмма возвращается в BASIC.

Число 65535 помещается в стек, чтобы отметить первое сохраненное значение. Позднее, когда число восстанавливается из стека, оно интерпретируется, как пара координат. Однако, если число равно 65535, происходит возврат в BASIC, т.к. программа закончена.

В регистр H загружается Y-координата, а в регистр L - X координата. Процедура SUBR вызывается, возвращая в HL адрес бита (X,Y). Если этот бит установлен (ON), происходит переход к LEFT. Иначе X-координата увеличивается, и делается переход к RIGHT, если X не равен 256.

В процедуре LEFT, DE устанавливается в 0. Регистры D и E используются, как флаги: D - вниз (DOWN), E - вверх (UP). X-координата уменьшается. SUBR вызывается, и вычерчивается точка (X,Y). Если Y-координата равна 175, подпрограмма переходит к DOWN. Если флаг "ВВЕРХ" установлен, происходит переход к RESET. Если бит (X,Y+1) сброшен, значение X и Y+1 сохраняются в стеке и флаг "ВВЕРХ" включается.

В процедуре RESET, если флаг "ВВЕРХ" включен, происходит переход к DOWN. Если бит (X,Y+1) включен (ON). Флаг "ВВЕРХ" выключается. В процедуре DOWN, если Y-координата равна 0, происходит переход к NEXT_PIXEL. Если флаг "ВНИЗ" включен, происходит переход к REST. Если бит (X,Y-1) сброшен (OFF), то значения X и Y-1 сохраняются на стеке и флаг "ВНИЗ" включается.

В процедуре REST, если флаг "ВНИЗ" выключен, происходит переход к NEXT_P. Если бит (X,Y-1) установлен (ON), то флаг "ВНИЗ" выключается. В процедуре NEXT_P, если X-координата равна 0, подпрограмма переходит к RETR. X-координата уменьшается, и, если новый бит (X,Y) сброшен (OFF), происходит переход к PLOT. В процедуре RETR X и Y-координаты извлекаются из стека. Если X и Y равны 255, то - возврат в BASIC, т.к.

заполнение области завершено. Иначе подпрограмма возвращается к RIGHT.

Процедура SUBR должна подсчитать адрес бита (X,Y) в памяти. В BASIC этот адрес будет:

16384 + INT(Z/8) + 256 * (Z - 8 * INT(Z/8)) + 32 * (64 * INT(Z/64) + INT(Z/8) - 8 * INT(Z/64)), где Z=175-Y

Пары регистров BC и DE сохраняются на стеке. В аккумулятор засылается число 175 и из этого значения вычитается Y-координата. Результат копируется в H-регистр. Затем HL сохраняется на стеке. Пять левых битов аккумулятора устанавливаются в 0, а затем к ним прибавляется 64. Результат копируется в C-регистр. При умножении на 256 получаем:

16384 + 256 * (Z-8 * INT(Z/8)).

В аккумулятор загружается Z, это значение делится на 8, результат копируется в регистр B. Этот результат - INT(Z/8). Установка трех крайних правых битов в 0 при ротации дает значения 8 * INT(Z/64), которое загружается в D-регистр. В аккумулятор загружается Z и 6 крайних правых битов выключаются, что дает 64 * INT(Z/64). Это значение загружается в E-регистр. Значение из C-регистра копируется в H. В аккумулятор загружается X-координата, это значение делится на 8, а результат копируется в L.

В аккумулятор затем загружается значение E-регистра и к нему прибавляется содержимое B. Значение D-регистра вычитается и результат загружается в DE. Это значение умножается на 32. DE восстанавливается из стека и прибавляется к HL. Т.о. HL теперь хранит адрес бита (X,Y).

В аккумулятор загружается первоначальное значение X. Установка пяти левых битов в ноль дает значение X - 8 * INT(X/8). В B регистр затем загружается 8 минус значение аккумулятора, чтобы использовать его в качестве счетчика.

Аккумулятор устанавливается в 1, и это умножается на 2 (B-1) раз.

В этот момент в аккумуляторе необходимо установить бит, который соответствует биту (X,Y) с адресом в HL. DE и BC затем восстанавливаются из стека, и SUBR выполняет возврат в основную программу.

6.9 Построение шаблонов.

Длина: 196

Количество переменных:2

Контрольная сумма: 20278

Назначение: Эта программа чертит шаблон любого размера на экране. Под шаблоном понимается любая, ранее определенная фигура.

Переменные:

Имя - X_start

Длина -1

Адрес - 23296

Комментарий: X-координата первого пиксела.

Имя Y_start

Длина -1

Адрес - 23297

Комментарий: Y-координата первого пиксела.

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок: Если строковая переменная, хранящая информацию по шаблону A$, не существует, имеет нулевую длину или не содержит никакой информации, программа возвращается непосредственно в BASIC. Это происходит также в случае, если Y_start больве, чем 175.

Комментарий: Это полезная программа для хранения фигур в памяти и быстрого вычерчивания их на экране.

Использование этой программы:

(I) LET A$="информация по шаблону"

(II) POKE 23296, X-координата первого пиксела

(III)POKE 23297, Y-координата первого пиксела

(IV) RANDOMIZE USR адрес

Информация шаблона - это символов, который имеет следующий формат:
" 0 " поместить точку
" 5 " уменьшить X-координату
" 6 " уменьшить Y- координату
" 7 " увеличить X-координату
" 8 " увеличить Y координату
Любые другие символы игнорируются

Программа включает в себя возможность 'wrap-round'. Т.е. если X-координата выходит за левую часть экрана, шаблон появляется справа и т.п. Чтобы изменить программу для использования иной строковой переменной вместо A$, нужно изменить 65 * (код буквы A) на код иного символа.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
METKA       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА
            LD HL,(23627)      42    75    92
NEXT_V      LD A,(HL)          126
            CP 128             254   128
            RET Z              200
            BIT 7,A            203   127
            JR NZ,FORNXT       32    23
            CP 96              254   96
            JR NC,NUMBER       48    11
            CP 65              254   65*
            JR Z,FOUND         40    35
STRING      INC HL             35
            LD E,(HL)          94
            INC HL             35
            LD D,(HL)          86
ADD         ADD HL,DE          25
            JR INCR            24    5
NUMBER      INC HL             35
            INC HL             35
            INC HL             35
            INC HL             35
            INC HL             35
INCR        INC HL             35
            JR NEXT_V          24    225
FORNXT      CP 224             254   224
            JR C,N_BIT         56    5
            LD DE,18           17    18    0
            JR ADD             24    236
N_BIT       BIT   5,A          203   111
            JR Z,STRING        40    228
NEXT_B      INC HL             35
            BIT 7,(HL)         203   126
            JR Z,NEXT_B        40    251
            JR NUMBER          24    228
FOUND       INC HL             35
            LD C,(HL)          78
            INC HL             35
            LD B,(HL)          70
            INC HL             35
            EX DE,HL           235
            LD A,(23297)       58    1     91
            CP 176             254   176
            RET NC             208
AGAIN       LD HL,(23296)      42    0     91
            LD A,B             120
            OR C               177
```

```
            RET Z               200
            DEC BC              11
            LD A,(DE)           26
            INC DE              19
            CP 48               254   48
            JR NZ,NOT_PL        32    78
            PUSH BC             197
            PUSH DE             213
            LD A,175            62    175
            SUB H               148
            LD H,A              103
            PUSH  HL            229
            AND 7               230   7
            ADD A,64            198   64
            LD C,A              79
            LD A,H              124
            RRA                 203   31
            RRA                 203   31
            RRA                 203   31
            AND 31              230   31
            LD B,A              71
            AND 24              230   24
            LD D,A              87
            LD A,H              124
            AND 192             230   192
            LD E,A              95
            LD H,C              97
            LD A,L              125
            RRA                 203   31
            RRA                 203   31
            RRA                 203   31
            AND 31              230
            LD L,A              111
            LD A,E              123
            ADD A,B             128
            SUB D               146
            LD E,A              95
            LD D,0              22    0
            PUSH HL             229
            PUSH DE             213
            POP HL              225
            ADD HL,HL           41
            ADD HL,HL           41
            ADD HL,HL           41
            ADD HL,HL           41
            ADD HL,HL           41
            POP DE              209
            ADD HL,DE           25
            POP DE              209
            LD A,E              123
            AND 7               230   7
            LD B,A              71
            LD A,8              62    8
            SUB B               144
            LD B,A              71
            LD A,L              62    1
ROTATE      ADD A,A             135
            DJNZ ROTATE         16    253
            RRA                 203   31
            POP DE              209
            POP BC              193
            OR (HL)             182
            LD (HL),A           119
HERE        JR AGAIN            24    165
NOT_PL      CP 53               254   53
```

```
            JR NZ,DOWN          32    1
            DEC 1               45
DOWN        CP 54               254   54
            JR NZ,UP            32    8
            DEC H               37
            LD H,A              124
            CP 255              254   255
            JR NZ,SAVE          32    19
            LD H,175            38    175
UP          CP 55               254   55
            JR NZ,RIGHT         32    8
            INC H               36
            LD A,H              124
            CP 176              254   176
            JR NZ,SAVE          32    7
            LD H,0              38    0
RIGHT       CP 56               254   56
            JR NZ,SAVE          32    1
            INC L               44
SAVE        LD (23296),HL       34    0
            JR HERE             24    215
```

Как она работает:

Для нахождения адреса строковой переменной используется немного измененная первая часть программы 'Поиск подстроки'.

Длина строковой переменной загружается в BC, а адрес первого символа A$ загружается в DE. В аккумуляторе устанавливается начальное значение Y, и, если оно больше 175, подпрограмма возвращается в BASIC. В H регистр загружается Y-координата, а в L X-координата. Если значение пары регистров BC равно 0, подпрограмма возвращается в BASIC, т.к. достигнут конец строковой переменной. BC уменьшается, чтобы показать, что обрабатывается следующий символ. Следующий символ загружается в аккумулятор и DE увеличивается, указывая на следующий байт. Если аккумулятор не содержит код 48, происходит переход к NOT_PL. Точка (X,Y) вычерчивается используя процедуру SUBR из программы "Закрашивание контура". Затем программа возвращается назад к 'AGAIN'. В процедуре NOT_PL, если аккумулятор содержит число 53, X-координата уменьшается. В процедуре DOWN, если аккумулятор не содержит число 54, делается переход к UP. Y-координата уменьшается, и, если ее значение становится равным -1, Y-координата устанавливается на значение 175.

В процедуре UP, если аккумулятор не содержит 55, происходит переход к RIGHT. Y-координата увеличивается, и, если она равна 176, то Y-координата устанавливается в 0. В процедуре RIGHT, если аккумулятор содержит значение 56, X-координата увеличивается. В процедуре SAVE координаты X и Y помещаются в память, а программа делает переход к HERE.

6.10 Увеличение экрана и копирование.

Длина:335

Количество переменных:8

Контрольная сумма: 33663

Назначение: Эта программа копирует часть дисплея в другую область экрана, увеличивая копию по X или по Y.

Переменные: см. рис.3

Имя	Длина	Адрес	Комментарий
upper_Y_co-ord	1	23296	Y-координата верхнего ряда
lower_Y_co-ord	1	23297	Y-координата нижнего ряда
right_X_co-ord	1	23298	X-координата крайней правой колонки
left_X_co-ord	1	23299	X-координата крайней левой колонки
horizontal_scale	1	23300	Увеличение по X

vertical_scale	1	23301	Увеличение по Y
new_left_co-ord	1	23302	X-координата крайней левой колонки области, в которую делается копирование
nev_lower_co-ord	1	23303	Y-координата нижнего ряда области, в которую делается копирование

Вызов программы:

```
RANDOMIZE USR адрес
```

Контроль ошибок: Программа возвращается в BASIC, если одно из условий верно.

(I) horizontal_scale=0

(II) vertical_scale=0

(III) upper_Y_co-ord больше, чем 175

(IV) new_lower_co-ord больше, чем 175

(V) lower_Y_co-ord больше, чем upper_co-ord

(VI) new_lower_co-ord больше, чем right_X_co-ord

Однако, для краткости программы нет контроля, который проверял бы возможность размещения новой картинки на экране. Если этого не получается, может произойти сбой. Программа также требует большого объема свободной области ОЗУ, и, если это не доступно, может произойти сбой.

Комментарий: Эта программа - не перемещаемая из-за процедуры PLOT. Она размещается по адресу 65033 и, если скопированная область экрана имеет тот же самый размер, что и оригинал, масштаб должен быть установлен в 1, для двойного размера загружается масштаб 2:1, для тройного размера загружается масштаб 3:1 и т.д.

ЛИСТИНГ МАШИННЫХ КОДОВ

```
МЕТКА       АССЕМБЛЕР          ЧИСЛА ДЛЯ ВВОДА
            LD IX,23296        221   33    0     91
            LD A,175           62    175
            CP (IX+0)          221   190   0
            RET C              216
            CP (IX+7)          221   190   7
            RET C              216
            SUB A              151
            CP (IX+4)          221   190   4
            RET Z              200
            CP (IX+5)          221   190   5
            RET Z              200
            LD HL,(23296)      42    0     91
            LD B,L             69
            LD A,L             125
            SUB H              148
            RET C              216
            LD (23298),A       50    0     91
            LD E,A             95
            LD HL,(23298)      42    2     91
            LD C,L             77
            LD A,L             125
            SUB H              148
            RET C              216
            LD (23298),A       50    2     91
            PUSH BC            197
            LD L,A             111
            LD H,0             38    0
            INC HL             35
            PUSH HL            229
            POP BC             193
            INC E              28
ADD         DEC E              29
            JR Z,REMAIN        40    3
```

```
            ADD HL,BC          9
            JR ADD             24    250
REMAIN      LD A,L             125
            AND 15             230   15
            LD B,A             71
            POP HL             225
            LD C,L             77
            JR NZ,SAVE         32    2
FULL        LD B,16            6           16
SAVE        PUSH HL            229
            CALL SUBR          205   13    255
            AND (HL)           166
            JR Z,OFF           40    2
            LD A,1             62    1
OFF         POP HL             225
            RRA                203   31
            RL E               203   19
            RL D               203   18
            LD A,L             125
            CP (IX+3)          221   190   3
            JR Z,NEXT_R        40    6
            DEC L              45
N_BIT       DJNZ SAVE          16    231
            PUSH DE            213
            JR FULL            24    226
NEXT_R      LD L,C             105
            LD A,H             124
            CP (IX+1)          221   190   1
            JR Z,COPY          40    3
            DEC H              37
            JR N_BIT           24    241
COPY        PUSH DE            213
            LD B,0             60
            LD H,B             96
            LD L,B             104
RESET       LD (23306),HL      34    10    91
            LD A,B             120
            OR A               183
            JR NZ,RETR         32    3
            POP DE             209
            LD B,16            6     16
RETR        SUB A              151
            DEC B              5
            RR D               203   26
            RR E               203   27
            RL A               203   23
            PUSH DE            213
            PUSH BC            197
            PUSH AF            245
            LD H,1             38    1
LOOP        LD L,1             46    1
PRESET      LD (23304),HL      34    8     91
            LD A,(23307)       58    11    91
            LD HL,0            33    0     0
            LD DE,(23301)      237   91    5     91
            LD D,L             85
MULTIP      OR A               183
            JR Z,CALC          40    6
            ADD HL,DE          25
            DEC A              61
            JR MULTIP          24    249
L_JUMP      JR RESET           24    208
CALC        LD A,(23303)       58    7     91
            ADD A,L            133
            LD HL,(23304)      42    8     91
```

```
            ADD A,L             133
            DEC A               61
            PUSH AF             245
            LD A,(23306)        58    10    91
            LD HL,0             33    0     0
            LD DE,(23300)       237   91    4     91
            LD D,L              85
REPEAT      OR A                183
            JR Z,CONTIN         40    4
            ADD HL,DE           25
            DEC A               61
            JR REPEAT           24    249
CONTIN      LD A,(23302)        58    6     91
            ADD A,L             133
            LD HL,(23305)       42    9     91
            ADD H,L             133
            DEC A               61
            LD L,A              111
            POP AF              241
            LD H,A              103
            POP AF              241
            PUSH AF             245
            OR A                183
            JR NZ,PLOT          32    7
            CALL SUBR           205   13    255
            CPL                 47
            AND (HL)            166
            JR POKE             24    4
PLOT        CALL SUBR           205   13    255
            OR (HL)             182
POKE        LD (HL),A           119
            LD HL,(23304)       42    8     91
            INC L               44
            LD A,(23301)        58    5     91
            INC A               60
            CP L                189
            JR NZ,PRESER        32    165
            INC H               36
            LD A,(23300)        58    4     91
            INC A               60
            CP H                188
            JR NZ,LOOP          32    155
            POP AF              241
            POP BC              193
            POP DE              209
            LD HL,(23306)       42    10    91
            INC L               44
            LD A,(23298) 58     2     91
            INC A               60
            CP L                189
            JR NZ,L_JUMP        32    164
            LD L,0              46    0
            INC H               36
            LD A,(23296)        58    0     91
            INC A               60
            CP H                188
            JR NZ,L_JUMP        32    154
            RET                 201
SUBR        PUSH BC             197
            PUSH DE             213
            LD A,175            62    175
            SUB H               148
            LD H,A              103
            PUSH HL             229
            AND 7               230   7
```

```
            ADD A,64            198    64
            LD C,A              79
            LD A,H              124
            RRA                 203    31
            RRA                 203    31
            RRA                 203    31
            AND 31              230    31
            LD B,A              71
            AND 24              230    24
            LD D,A              87
            LD A,H              124
            AND 192             230    192
            LD E,A              95
            LD H,C              97
            LD A,L              125
            RRA                 203    31
            RRA                 203    31
            RRA                 203    31
            AND 31              230    31
            LD L,A              111
            LD A,E              123
            ADD A,B             128
            SUB D               146
            LD E,A              95
            LD D,0              22     0
            PUSH HL             229
            PUSH DE             213
            POP HL              225
            ADD HL,HL           41
            ADD HL,HL           41
            ADD HL,HL           41
            ADD HL,HL           41
            ADD HL,HL           41
            POP DE              209
            ADD HL,DE           25
            POP DE              209
            LD A,E              123
            AND 7               230    7
            LD B,A              71
            LD A,8              62     8
            SUB B               144
            LD B,A              71
            LD A,1              62     1
ROTATE      ADD A,A             135
            DJNZ ROTATE  16     253
            RRA                 203    31
            POP DE              209
            POP BC              193
            RET                 201
```

Как она работает:

В IX загружается адрес буфера принтера для использования его в качестве указателя переменных. Если верхняя Y-координата или новая нижняя координата больше 175, программа возвращается в BASIC. Если значение увеличения по горизонтали или вертикали равно 0, происходит возврат в BASIC.

В H-регистр загружается нижняя Y-координата, а в L-регистр - верхняя Y- координата. L-регистр копируется в B-регистр и в аккумулятор. H-регистр вычитается из аккумулятора и подпрограмма возвращается в BASIC, если результат отрицательный.

Значение аккумулятора затем помещается в ячейку 23298 для использования в качестве счетчика. Пара регистров BC затем сохраняется на стеке.

Регистр HL загружается значением аккумулятора, увеличивается и копируется в регистр BC. BC прибавляется к HL E раз, результирующее значение в HL является числом пикселей для копирования. В аккумулятор загружается значение L-регистра и 4 крайних

левых бита устанавливаются в 0. Результат копируется в B-регистр для использования его в качестве счетчика.

Пара регистров HL восстанавливается из стека и регистр L копируется в C-регистр. Если B-регистр содержит 0, в регистр загружается число 16 - это количество битов в регистровой паре. Затем вызывается процедура SUBR и в аккумулятор загружается значение POINT (L,H). Пара регистров DE сдвигается влево, а значение бита из аккумулятора загружается в крайний правый бит E- регистра.

Если L-регистр равен левой X-координате, подпрограмма переходит к NEXT_R (следующий ряд). Иначе уменьшается L-регистр, а затем - B-регистр. Если B-регистр не содержит 0, подпрограмма возвращается к SAVE для подачи следующего бита в пару регистров DE. Если B-регистр содержит 0, пара регистров DE помещается в стек и происходит переход к FULL.

В процедуре NEXT_R в L-регистр загружается правая X-координата, а в аккумулятор загружается значение H-регистра. Если значение аккумулятора равно нижней Y-координате, происходит переход к COPY, т.к. последний пиксель для копирования подан в DE. Иначе, H-регистр уменьшается, указывая на следующий ряд, и подпрограмма возвращается к N_BIT.

В процедуре COPY содержимое пары BE помещается в стек, а B, H и L-регистры устанавливаются в 0 для использования их в качестве счетчиков. Содержимое пары регистров HL помещается в ячейку с адресами 23306/7 - HL теперь может использоваться как счетчик последующих циклов без использования стека. Если B-регистр содержит 0, DE восстанавливается из стека, а регистр B повторно устанавливается на значение 16, определяя количество пикселей, хранимых в DE. B-регистр уменьшается показывая, что бит информации удален из DE. Крайний правый бит E-регистра загружается в аккумулятор, а пара регистров DE сдвигается вправо. DE,BC и AF помещаются в стек до тех пор, пока выполнятся определенные расчеты.

В регистры H и L загружается 1 для использования их в качестве счетчика, а HL помещается в ячейки с адресами 23304/5. В аккумулятор загружается значение байта по адресу 23307 - это один из счетчиков, сохраненных ранее. В пару регистров DE загружается масштаб по вертикали. Это значение затем умножается на значение аккумулятора, а результат подаётся в HL. Это значение прибавляется к новой нижней Y-координате в аккумуляторе. Байт по адресу 23304 затем добавляется к аккумулятору, а результат уменьшается.

Аккумулятор теперь хранит Y-координату для построения следующего пиксела. Это значение хранится в стеке до тех пор, пока подсчитывается X-координата похожим способом. После расчета X-координата загружается в L-регистр. Y-координата восстанавливается из стека и загружается в H-регистр. В аккумуляторе устанавливается последнее значение, хранящееся в стеке. Если оно равно 1, то должна быть построена точка (X,Y), иначе должна быть выполнена процедура UNPLOTTED. Вызывается SUBR и выполняются соответствующие действия.

Пара регистров HL загружается счетчиками цикла, хранящимися адресам 23304/5. Регистр L увеличивается, и, если он не содержит значение (1+vertical_scale), программа возвращается к PRESER. Регистр H увеличивается, и, ее он не содержит значение (1+horizontal_scale), происходит переход к LOOP.

Пары регистров AF,BC и DE восстанавливаются из стека, а в пару регистров HL загружается второе значение набора счетчиков цикла, которое хранится по адресам 23306/7. Регистр L увеличивается и происходит переход к RESET, если результат не равен (right_X_co-ord-left_X_co-ord + 1)

Регистр L устанавливается в 0 - это первоначальное значение счетчика цикла. H-регистр затем увеличивается и подпрограмма переходит к RESET, если результат не равен (upper_Y_co-ord -lower_Y_co-ord + 1). Программа возвращается в BASIC,

Процедура SUBR идентична той, что используется в программе "Закрашивание контура".

Продолжение следует

Алексеев А. Г.

МАСТЕРФАЙЛ-09 Полная русификация

Окончание.
Начало ZX-РЕВЮ-92',стр-29-32.

Текстовые сообщения, выводимые на экран в любой программе кроме печатаемых символов содержат также управляющие коды AT, TAB, INK, PAPER и т.д. Их без крайней необходимости лучше не изменять, так как это может нарушить работу программы, но в некоторых случаях придётся изменять и их. Для этого в программе мониторе предусмотрена строка 231. Если Вы уберёте REM из строки 231 и подставите REM в начало строки 230, то работа программы несколько изменится. Будет такой же вывод на экран, так же запрашивается информация, но теперь без кавычек. Это означает что ожидается ввод кода - числа от 0 до 255, которое после нажатия "ENTER" непосредственно будет записано в память. Режим ввода кодов будет нужен редко, но все-таки иногда пригодится. Для того, чтобы остановить программу в этом режиме, надо просто нажать "КУРСОР ВНИЗ" (CAPS SHIFT+6) или нажать "STOP" (SYMBOL SHIFT + A) и "ENTER".

Теперь подставьте REM в обе строки 230 и 231 и сделайте RUN. После ввода адреса Вам будет выдан дамп памяти. Этот режим Вы будете использовать для поиска текстовых сообщений в программе. С этого момента можно начинать работу, но прежде - несколько примеров.

Введите адрес 60350. Вы увидите на экране:

```
60350       237     GO SUB
60351       225     LLIST
60352       201     <>
60353       22      ?
60354       2       ?
60355       0       ?
60356       17      ?
60357       6       ?
60358       86      V
60359       69      E
60360       82      R
60361       84      T
60362       73      I
60363       67      C
60364       65      A
60365       76      L
60366       32
60367       76      L
60368       73      I
60369       78      N
60370       69      E
60371       255     COPY
```

В ячейках 60350 находится какая - то программа в машинных кодах, а вот с ячейки 60353 начинается строка символов. При этом в ячейках 60353...60357 находятся управляющие символы, а сам текст расположен в ячейках 60358...60370. Потом идет символ с кодом 255. который завершает строку текста. Далее опять начинается программа в машинных кодах.

Управляющие символы имеют следующее значение:

Код: 16 - упр. INK
Код: 17 - упр. PAPER
Код: 18 - упр. FLASH

Код: 19 - упр. BRIGHT
Код: 20 - упр. INVERSE
Код: 21 - упр. OVER
Код: 22 - упр. AT
Код: 23 - упр. TAB

Так что комбинация кодов в ячейках 60353...60358 эквивалентна фрагменту Бейсик-строки:

```
... AT 2,0; PAPER  6; ...
```

В данном случае нас интересует только текст, который должен быть заменен:

```
VERTICAL  LINE
ВЕРТИК.  ЛИНИЯ
```

(Можете, убрав REM из строки 230, запустить программу, задать адрес 60358 и попробовать заменить текст на русский.)

Другой пример. После старта программы-монитора задайте адрес 58425. В дампе памяти Вы узнаете текст основного меню MF 09:

```
AADD A RECORD ...
CCHOOSE A REPORT
DDISPLAY/PRINT ..
     . . .
```

Правда он при работе программы выводится несколько иначе:

```
ADD A  RECORD .......A
CHOOSE A REPORT .....C
DISPLAY/PRINT .......D
     . . .
```

В памяти сначала идет символ той клавиши, которая должна быть нажата, затем текст сообщения. Вдаваться в детали работы программы нет необходимости. У нас другая задача. Оставив без изменения символы тех клавишей, которые должны нажиматься (первые буквы, они набраны в режиме курсора [C], надо заменить текст на русский:

```
АНОВАЯ ЗАПИСЬ...
СОБЗОР ФОРМАТОВ.
ОДИСПЛЕЙ/ПЕЧАТЬ.
     . . .
```

Еще пример. После старта монитора задайте адрес 59759. На экране увидите:

```
59759   78   N
59760   79   O
59761   32
59762   84   T
59763   73   I
59764   84   T
59765   76   L
59766   197  OR
```

При работе программы эта надпись выводится так:

```
NO TITLE
```

Откуда же берется последняя буква <E> ?

Вообще надо сказать, что перед тем, как в машинных кодах подана команда вывода строки символов на экран, предварительно задан адрес начальной ячейки, где расположена строка символов и длина этой строки. Анализ программы MF 09 показывает, что здесь применяется еще и другой способ вывода. Длина выводимой строки не указывается, однако во время вывода анализируется код выводимого символа. Если это код с 0 по 127, то продолжается вывод на экран, а если код символа больше или равен 128, то это значит, что процедура вывода этим символом заканчивается, при этом на экран выводится символ, код которого на 128 меньше, чем содержащийся в памяти. В ячейке 59766 стоит OR. Его код (читаем на экране) равен 197. 197-128=69. По таблице кодов "Спектрума", а если ее нет под

рукой, то сделав PRINT CHR$ 69, выясняем, что этому коду соответствует буква "E". Вот откуда взялась последняя буква.

Этот текст можно заменить следующим образом:

```
NO TITLE
НЕ НАЗВ.
```

При этом буквы "НЕ НАЗВ" заменяем непосредственно русскими буквами, а вместо точки, код которой (выясняем: PRINT CODE ".") равен 46, надо ввести символ с кодом 46+128=174. По таблице кодов "Спектрума" или сделав PRINT CHR$ 174, определяем, что это VAL$. Можно ввести его в том же режиме, нажав "EXT. MODE", затем "SYMBOL SHIFT"+"j". Часто придется вводить в качестве последнего символа - "пробел" (его код 32). Это будет символ, код которого равен 32+128 =160. По таблице кодов находим, что это символ "Q" UDG-графики. Вводя его, нажмите "GRAPH" (CAPS SHIFT+9), затем "Q", затем еще раз "GRAPH" (пусть Вас не смущает, что напечатается что-то непонятное, так как на месте символов UDG-графики находятся коды программы MF 09). Однако, могут попадаться такие символы, которые с клавиатуры вводятся в режиме курсора [K]. Например, начиная с адреса 64522 находится текст:

```
64522   65      A
64523   82      R
64524   71      G
64525   32
64526   78      N
64527   79      O
64528   84      T
64529   32
64530   78      N
64531   85      U
64532   77      M
64533   69      E
64534   82      R
64535   73      I
64536   195     NOT
```

В ячейке 64536 "спрятана" буква, код которой 195-128=67. Это латинская буква "C". Заменяем эту строку на русский текст:

```
ARC NOT NUMERIC
АРГ.НЕ ЧИСЛОВОЙ
```

При этом вместо последней русской буквы "И", (ее код выясняем, сделав PRINT CODE "И" - при этом "И" набрана в режиме курсора [L], так как включен русско-латинский символьный набор; он равен 106) надо ввести символ с кодом 106+128=234. Это REM. Ввести этот символ, находясь в режиме курсора [L] или [C] никак не удастся. Для этого надо переключить курсор на [K] (командный режим). Как это сделать? Введите ключевое слово "THEN" (SYMBOL SHIFT+G). Теперь курсор стал [K] и Вы можете ввести "REM", нажав "E". Теперь нажмите "КУРСОР ВЛЕВО" и удалите "THEN" при помощи "DELETE" (CAPS SHIFT+0). Далее - "ENTER" - ввод кода в память.

Можно это сделать и иначе - просто остановить программу и "вручную" сделать POKE 64536,234.

Теперь Вы знаете, как располагаются текстовые сообщения в программе MF 09 и умеете работать с программой-монитором. Подставьте REM в обе строки 230 и 231 и начинайте работу. Запустив программу-монитор, задайте адрес 57328. (С адреса 56560 по 57327 расположен символьный набор, там делать нечего.) Сейчас задача - просмотреть все коды программы от начала до конца, найти все текстовые сообщения в программе и

записать их адреса и сами сообщения на бумагу, оставляя место для последующего перевода.

Далее, вооружившись англо-русским словарем, начинаем перевод. Не надо стремиться переводить текст дословно, так как мы слишком сильно сжаты рамками того места, которое отведено в программе под ту или иную фразу. Более подойдет литературный перевод, при этом, конечно, лучше, если Вы уже достаточно поработали с программой MF 09 и Вам понятен смысл переводимых сообщений. Это позволит Вам найти подходящую по смыслу замену, даже если она не является переводом английской фразы. Что касается программы MF 09, то сообщения, которые Вы там встретите и их перевод, реализованный в моем варианте "MF09 RUS", приведен ниже. Когда же Вы возьметесь за какую-нибудь другую программу, то Вам придется проделать самим всю эту работу. Это, пожалуй, самая большая по затратам времени, ответственная и творческая часть работы. Так что запаситесь терпением, может быть не на один день. Остальное - дело чисто механическое.

Текстовые сообщения программы MF 09 и их перевод

Вначале каждого сообщения указан адрес, с которого необходимо произвести замену текста. Далее - английский текст и под ним русский вариант перевода.

Подчеркивающая черта обозначает "пробел".

Символы, коды которых должны быть на 128 больше, отмечены "*" в конце строки, далее идет код символа и сам символ или ключевое слово.

```
57764:
ITEM_ALREADY_IN_RECORD  * 196 BIN
ЭТО_ПОЛЕ_УЖЕ_ВВЕДЕНО__  * 160
                        Q-GRAPH
57787:
AADD_ITEM........ RREPLACE_ITEM...
АНОВОЕ_ПОЛЕ...... RИЗМЕНЕНИЕ ПОЛЯ.

EERASE_ITEM...... HNEXT_ITEM......
ЕУДАЛЕНИЕ_ПОЛЯ... НВЫБОР_ПОЛЯ.....

DDISPLAY/PRINT... GGET_ITEM.......
ДДИСПЛЕИ/ПЕЧАТЬ.. GВЫЗОВ_ПОЛЯ.....

PPROMPT_ITEMS... =ANOTHER,RECORD.
РАВТОЗАПРОС..... =СЛЕДУЮЩАЯ ЗАП..

MMAIN_MENU......
МГЛАВНОЕ_МЕНЮ...

57940:
ENTER_TEXT_1-128_CHARS.
ВВОД_ТЕКСТА_1-128_СИМВ.

57988:
GIVE_DATA_REF * 198 AND
МЕТКА_ДАННЫХ_ * 160 Q-GRAPH

58089:
NOT_NAMED
НЕТ_ИМЕНИ

58130:
SAVE_PROG/FILE.
ЗАПИСЬ.ПР./ФАЙЛ

58243:
Y-TO_CONFIRM * 205 STEP
```

```
Y-ЕСЛИ_ДА___ * 160 Q-GRAPH

58425:
AADD_A_RECORD... CCHOOSE_A_REPORT
АНОВАЯ_ЗАПИСЬ... СОБЗОР_ФОРМАТОВ.

DDISPLAY/PRINT.. EEDIT_FORMAT_DEF
ДДИСПЛЕИ/ПЕЧАТЬ. ЕРЕДАКТ._ФОРМАТА

LLOAD_A_FILE.... NNAME_DATA_REF..
LЗАГРУЗКА_ФАЙЛА. НИМЕНА_ПОЛЕЙ_ДАН

SSEARCH_THE_FILEIIHVERT_SELECTN.
SПОИСК.......... ИНВЕРСИЯ_ВЫБОРА

RRESET_SELECTIONPPURGE_SEL_RECDS
RСБРОС_ВЫБОРА...РУДАЛЕН.ВЫБР.ЗАП

TTOTAL/AVERAGE.. VSAVE_PROG/FILE
ТИТОГ_ОБЩ./СРЕДНVЗАПИСЬ_ПР./ФАЙЛ

UEXEC_USER_BASIC
UВЫПОЛНИТЬ_BASIC

58903:
FILE_ONLY.......... F
ТОЛЬКО_ДАННЫЕ...... F

58998:
_RECS=00000_SEL=00000_SPA=00000_
__ЗАП=00000_ВЫБ=00000_СВБ=00000_

59326:        59759:
FILE_FULL     NO_TITLE  * 197 OR
НЕТ_МЕСТА     НЕ_НАЗВ.  * 174 VAL$

60065:        60077:
DELETE        COPY
УДАЛ._        КОП.

60082:
ALREADY_DEFINED  * 196 BIN
МЕТКА_ЗАНЯТА__   * 160 Q-GRAPH

60097:
NO_SUCH_FORMAT_DEFINED * 196 BIN
ТАКОИ_ФОРМАТ_НЕ_ЗАДАН_ * 160 Q-GRAPH

(Глюк в тексте)
601124:       60287:        60300:
REF_0         GENERAL       SEQU__:
МЕТ.0         ГЛАВНЫЙ       СОРТ._:

60312:        60325:
BORDER_       _INTVL 002
_БОРДЮР       _ИНТЕРВ002

60358:        60387:
VERTICAL_LINE LINE_ACROSS
ВЕРТИК._ЛИНИЯ ГОРИЗ.ЛИНИЯ

60435:        60495:        60574:
BOX           TEXT          DATA_REF__:
ПР.           ТЕКС          ПОЛЕ-ДАН._:
```

(Примечание: перевод слова TEXT как ТЕКС - не очень удачный вариант, но мы ограничены местом в памяти. То же касается перевода слова BOX - ПР. (Прямоугольник). Как с этим бороться, мы ещё рассмотрим ниже.)

```
60591:          60603:
WIDTH_000       DEPTH_000
ШИРИНА000       ВЫСОТА000

60615:          60645:
NULL:           PAPER_
НОЛЬ:           БУМАГА

60677:
GIVE_REPORT_REF * AND
МЕТКА_ФОРМАТА__ * Q-GRAPH

60793:          60801:          60809:
INV_N           BRI_Y           PAD_Y
ИНВ.N           ЯРК.Y           ФОН_Y

60817:          60828:          60872:
FLASH_N         MPRT=42         LINE_000
МИГАН.N         СИМВ=42         _СТР.000

60883:          60931:
COL_000         X_COORD_000__LENGTH_000
КОЛ.000         X-КООРД.000___ДЛИНА_000

60957:          61205:          61213:
Y_COORD_000     LENGTH=         WIDTH=
Y-КООРД.000     ДЛИНА__         ШИРИНА

61220:          61227:          61234:
DEPTH=          PAD             NULL_TEXT=
ВЫСОТА          ФОН             НОЛЬ_ТЕКСТ

61361:          61367:          61380:
LINE=           MICRO-PRT_      42_PITCH
_СТР.           МИКРОПЕЧ._      42_СИМВ.

61392:          61457:          61467:
COLUMN=         BRIGHT_         INVERSE_
КОЛОНКА         ЯРКОСТЬ         ИНВЕРСИЯ

61478:                          61484:
PAPER= * 189 ABS                FLASH_
БУМАГА * 225 LLIST              МИГАН.

61516:          61525:          61596:
X_COORD=        Y_COORD=        BORDER=
X-КООРД.        Y-КООРД.        БОРДЮР_

61603:                          61612:
SEQUENCE= * 189 ABS             INTERVAL=
СОРТИРОВ. * 174 VAL$            ИНТЕРВАЛ_

61639:
AADD_NEW_FORMAT.RREVIEW_FORMAT..
АНОВЫЙ.ФОРМАТ...RИЗМЕНЕНИЕ_ФОРМ.

MMAIN_MENU......
МГЛАВНОЕ_МЕНЮ...

61688:
```

```
AADD_NEW_ELEMENTRREPLACE_ELEMENT
АНОВЫЙ_ЭЛЕМЕНТ..RИЗМЕНЕНИЕ_ЭЛЕМ.

EERASE_ELEMENT..NNEXT_ELEMENT...
ЕУДАЛЕНИЕ__ЭЛЕМ.NВЫБОР_ЭЛЕМЕНТА.

CCOPY_FORMAT....XDELETE_FORMAT..
СКОПИРОВ. ФОРМ..ХУДАЛЕНИЕ_ФОРМ..

DDISPLAY/PRINT..MPREVIOUS_MENU..
ДДИСПЛЕИ/ПЕЧАТЬ.МПРЕДЫДУЩЕЕ_МЕНЮ

61817:
DDATA_FROM_RECD.LLITERAL_TEXT...
ДДАННЫЕ.........LЗАГОЛОВОК,ТЕКСТ

BBOX............HHORIZONTAL_LINE
ВПРЯМОУГОЛЬНИК..НГОРИЗОНТ._ЛИНИЯ

VERTICAL_LINE.. MPREVIOUS_MENU..
VВЕРТИК._ЛИНИЯ..МПРЕДЫДУЩЕЕ_МЕНЮ

62102:          62127:
TOTAL___=_      AVERAGE_=_
ОБЩЕЕ___=_      СРЕДНЕЕ_=_

62194:          62219:
GIVE_FILE_NAME  ___ERROR__
ИМЯ_ФАИЛА_____  __ОШИБКА__

63177:          63196:        63205:
REPORT_         MENU          .._MORE
ФОРМАТ_         МЕНЮ          .._ЕЩЁ_

63225:
f1  A  f0  LL_  f1  S  f0  INGLE_PAGE  * 197 OR
f1  A  f0  -ВСЕ,  f1   S  f0  -ЭКРАН__ * 160 Q-GRAPH
```

(здесь f1 и f0 - управляющие символы с кодами 18, 1 и 18, 0 - включение и выключение режима мигания)

```
63331:
NNEXT_PAGE......#ADVANCE_1-9_RCS
NСЛЕДУЮЩИЙ_ЭКРАН#ВПЕРЕД_НА_1-9..

OBACK_ONE RECORDBBACK_TO_1ST_REC
ОНАЗАД_НА_1.....ВВОЗВР._В_НАЧАЛО

PPRINT..........UUPDATE_TOP_REC.
РПЕЧАТЬ.........UИЗМЕН._ВЕРХ_ЗАП

EERASE_TOP_REC..OOMIT_TOP_RECORD
ЕУДАЛЕН.ВЕРХ_ЗАПОИСКЛЮЧ.ВЕРХ_ЗАП

CCOPY_TOP_RECORDSSEARCH_THE_FILE
СКОПИР. ВЕРХ_ЗАПSПОИСК..........

TTOTAL/AVERAGE..RSELECT_REPORT..
ТИТОГ_ОБЩ./СРЕДНRВЫБОР_ФОРМАТА..

MMAIN_MENU......QQUIT_THIS_MENU.
МГЛАВНОЕ_МЕНЮ...QУДАЛИТЬ_МЕНЮ...

64522:
ARG_NOT_NUMERIC  *  195 NOT
```

```
АРГ. НЕ_ЧИСЛОВОИ *  234 REM

64555:          64570:          64580:
ARGUMENT=       ALL_            SEL_
АРГУМЕНТ=       ВСЕ_            ВЫБР

64596:          64607:
CHAR_           NUM__
ТЕКС.           ЧИСЛ.

64623:
ASELECT_FROM_ALLLSELECT_FROM_SEL
АВЫБОР_ИЗ_ВСЕХ..LВЫБОР_ИЗ_ВЫБРАН

DDISPLAY/PRINT..MMAIN_MENU......
ДДИСПЛЕЙ/ПЕЧАТЬ.МГЛАВНОЕ_МЕНЮ...

64688:
CCHARACTER......NNUMBERIC.......
СТЕКСТОВЫИ......NЧИСЛОВОИ.......

MPREVIOUS_MENU..NПРЕДЫДУЩЕЕ.МЕНЮ

64737:
EEQUAL_TO.......UUNEQUAL_TO.....
ЕРАВНО_АРГУМЕНТУUНЕ_РАВНО_АРГ...

LLESS_THAN......GGREATER_THAN...
LМЕНЬШЕ.ЧЕМ_АРГ.GБОЛЬШЕ.ЧЕМ_АРГ.

SSTRING_SEARCH..MMAIH_MENU......
SСКАНИРОВАНИЕ...МГЛАВНОЕ_МЕНЮ...

65210:
NON-NUMERIC_DATA: f1  S  f0 KIP_
НЕ_ЧИСЛ.ДАНН. f1  S  f0 -ДАЛЬШЕ,

f1  U  f0  PDATE  * 197 OR
f1  U  f0  -СТОП  * 240 LIST

65320:
PPROGRAM+FILE...FFILE_ONLY......
РПРОГР.+_ФАЙЛ...FТОЛЬКО_ДАННЫЕ..

65353:
SAVE_NAME_?
ИМЯ_ФАЙЛА_?
```

После того, как перевод закончен на бумаге, загружаем программу-монитор. Убираем REM из строки 230 и начинаем замену английских текстов на русские в соответствии со своими записями. Не забывайте периодически выгружать результаты через RUN 5.

Попутно следует сказать о том, что не стоит экономить ленту, стирая старый вариант и записывая на него новый. Это позволит Вам вернуться назад в том случае, если в результате работы Вы случайно "запортите" программу, ошибочно изменив содержимое памяти где-нибудь в области машинных кодов.

Когда эта часть работы будет выполнена и записан последний вариант "русского" кодового куска, надо проверить, как все будет работать в новом виде. Запустите программу MF 09 и, после загрузки блоков, "MF LOADER" И "MF 09 LEER" остановите магнитофон и вставьте кассету с записью "русского" кодового куска. Загрузите его. После старта программы, если Вы все сделали аккуратно, Вы можете насладиться результатами своего труда. Но это потом. А сейчас надо внимательно посмотреть, нет ли где-нибудь

грамматических ошибок, которые практически неизбежны в результате такой большой работы, и вообще, все ли выполняется "ладно" и хорошо, нет ли где-нибудь "шероховатостей" перевода и т. д.

Если ошибок нет, то в общем можно считать работу выполненной, но присмотримся к работе программы повнимательнее. Ну например, вместо слова "BOX" - прямоугольник - мы ввели "ПР." (см. ячейку 60435). Это выглядит не очень удачно на экране, но, с другой стороны, что еще можно разместить в выделенных для этого трех байтах текста? Аналогично, вместо "TEXT" - мы ввели "ТЕКС" (ячейка 60495). Та же история - не хватает места. Что тут можно сделать?

Вспомним, как в машинных кодах осуществляется вывод на экран. Непосредственно перед выводом где-то должен быть задан адрес начала строки текста. Надо найти этот адрес и изменить его, расположив альтернативные текстовые сообщения на новом месте. Вот как это может выглядеть.

Текстовая строка "BOX" начинается фактически с адреса 60430:

```
60430   22      ?  - AT  2:0;
60431   2       ?
60432   0       ?
60433   17      ?  - PAPER  6;
60434   6       ?
60435   66      B  - т е к с т
60436   79      O
60437   88      X
60438   255     COPY    - "конец"
```

Вместо "ПР." желательно было бы разместить ну хотя бы:

```
AT 2 0 PAPER 6 ПРЯМОУГ. "конец"
```

Для такой строки надо 14 байтов памяти. Аналогично со строкой "TEXT" - фактическое начало - адрес 60490. Замена:

```
AT 2 0 PAPER 6 ТЕКСТ "конец"
```

Для этой строки надо 11 байтов памяти. Всего для двух строк нам нужны 25 байтов. Подумаем где бы их найти. В нашем случае есть свободное место перед символьным набором (расположенным с адреса 56560). Запустите программу-монитор (REM - в строке 321) и с адреса 56560-25=56535 осуществите ввод. Вначале пять пробелов (они нужны, чтобы потом на их место поставить управляющие коды AT и PAPER), затем текст "ПРЯМОУГ.", затем ещё пробел. И сразу же далее: еще пять пробелов, затем слово "ТЕКСТ" и еще один пробел. Ввод закончится ячейкой 56559. Теперь остановите программу, подставьте REM в строку 230 и удалите REM из строки 231. Запустите программу-монитор: GO TO 200 и снова с адреса 56535 введите следующие коды: 22, 2, 0, 17, 6, затем восемь раз просто нажмите "ENTER", ничего не вводя, затем введите: 255, 22, 2, 0, 17, 6, ещё 4 раза нажмите "ENTER", и, наконец, введите последнюю цифру 255. Ввод завершен на ячейке 56559.

Теперь надо найти в программе те места, где указаны адреса старых текстовых сообщений и изменить их на новые. Старый адрес для сообщения "BOX" - 60430. Для поиска подставим REM в строки 230 и 231 и введем новую строку:

```
210  IF (PEEK A+256*PEEK(A+1))<>60430 THEN GO TO 300
```

Запустите программу RUN, задайте адрес 56328 - это будет начало поиска - и можете откинуться на спинку кресла, пока компьютер не выдаст Вам необходимую информацию. На экране напечатается:

```
60405   14   ?
```

Не нажимайте "BREAK", подождите, пока не будет просмотрена вся программа, может быть на этот адрес есть ссылки и в других местах или это случайная комбинация чисел, никакого отношения не имеющая к процедуре вывода. Поиск закончится сообщением об ошибке: "B Integer out of range, 210:1". Это значит, что вся память просмотрена.

Место в программе, где указывается на искомый адрес, оказалось единственным. Тем проще для нас.

Теперь найдем адрес, где указано начало второго, интересующего нас сообщения - "TEXT" - это 60490. Изменим в строке 210 программы-монитора число 60430 на 60490 и GO TO 200. На экране появилось:

```
60458   74   J
```

Сообщение об ошибке. Поиск закончен, этот адрес тоже единственный. Теперь оба адреса надо заменить на новые. Для сообщения "ПРЯМОУГ. " это будет адрес 56535, а для "ТЕКСТ" - 56549. Младшие и старшие байты этих чисел находим, выполнив:

```
PRINT  56535-256*INT(56535/256)
PRINT  INT(65355/256)
PRINT  56549-INT(56549/256)
PRINT  INT(56549/256)
```

Получим, соответственно, числа 215, 220, 229, 220. Теперь выполним:

```
POKE 60405,215     POKE 60406,220
POKE 60458,229     POKE 60459,220
```

Осталось готовую программу записать на ленту, но прежде чем сделать RUN 5, вспомните, что наша программа стала длиннее на 25 байтов. Это надо учесть, производя изменения в строке 6 программы-монитора:

```
SAVE N$ CODE 56560,8976
```
надо заменить на
```
SAVE N$ CODE 56535,9001
```

Теперь можете сделать RUN 5. Надо также изменить программу-загрузчик "MF LOADER", заменив CLEAR 56559 на CLEAR 56534.

Попробуйте новый вариант программы. Вы согласны с тем, что она выиграла от такой замены?

Еще одна деталь. При запуске программы, когда появляется на экране главное меню, слева вверху на синем фоне появляется надпись

```
"ГИЛЕ:MF 09 LEER"
```

Вместо русско-латинского слова "ГИЛЕ" надо бы написать "ФАЙЛ", но вот беда: это слово мы не нашли в кодах программы MF 09. На самом деле все очень просто, Его там и нет. Надо искать в другом месте. Это слово находится внутри Бейсик-программы MF 09 LEER. Чтобы убедиться в этом, сделайте следующее. Остановите программу нажав в главном меню "L", затем "КУРСОР ВНИЗ" (CAPS SHIFT + 6). Теперь сделайте PRINT F$. Вы увидите надписи, выводимые на экран в режиме главного меню. Слово "ГИЛЕ" находится в файле данных Бейсик-переменной F$. Как поступить в этом случае?

Запустите программу MF 09. Находясь в главном меню, нажмите "V", а затем "F" и, указав имя, например "LEER", запишите пустой файл на магнитофон. (Это как раз и будет переменная F$). Дальше загрузите опять программу-монитор (и коды MF 09), остановите ее и загрузите переменную F$, подав прямую команду:

```
LOAD "LEER" DATA F$ ()
```

Теперь надо помнить о том, что нельзя подавать команду RUN только GO TO 1, так как по команде RUN Вы уничтожите переменную F$. Просматривая дамп памяти примерно с адреса 24400, вскоре Вы встретите фразы: "MASTERFILE BEP 09" и "FИЛЕ:LEER". Замените их на "MASTERFILE RUS 09" и "ФАЙЛ:LEER" обычным способом. Теперь остановите программу-монитор и запишите переменную F$ на магнитофон прямой командой:

```
SAVE "LEER" DATA F$ ()
```

Далее надо запустить программу MF 09 и, находясь в главном меню нажать "L". Задайте имя "LEER" и загрузите с магнитофона изменённый вариант переменной F$. После выхода в главное меню Вы увидите, правильно ли Вы все сделали. Теперь можете записать измененную Бейсик-программу MF 09, нажимая в главном меню "V" затем "P" и указав имя "MF 09 LEER".

Смотрим, что еще в программе не так. В режиме "дисплей" при просмотре записей в правом нижнем углу экрана на голубом фоне появляется надпись:

```
.._ЕЩЕ_ или No_ЕЩЕ_ ("_" - обозначает пробел)
```

При этом в памяти постоянно хранится только запись "_ЕЩЕ_", а "No" или ".." подставляются программой непосредственно в строку, подлежащую выводу, в зависимости от алгоритма работы. Таким образом, если мы хотим сделать перевод текста, то на место английского "No" не помещается даже русское "НЕТ". А русско-латинская фраза "No ЕЩЕ" выглядит довольно-таки неудачно. В этом случае можно попробовать использовать символы, например "+" и "-".

Анализируя работу программы MF 09 (кстати в этом мне очень помог трехтомник "Программирование в машинных кодах", написанный "ИНФОРКОМОМ"), я нашел, как мне кажется, приемлемый вариант замены. Для реализации этого варианта надо сделать:

```
POKE 62759, 230    POKE 63155, 32
POKE 63037, 230    POKE 63205, 32
POKE 63123, 230    POKE 62206, 32
POKE 63146, 230    POKE 62757, 45
POKE 63121, 43     POKE 63040, 45
```

Теперь, если при просмотре записей в режиме "дисплей" есть еще записи, то в левом нижнем углу будет: "_+_ЕЩЕ_", а если больше нет записей, то: "_-_ЕЩЕ_". По моему, это выглядят более гармонично.

Теперь еще об одной существенной детали. При вводе данных, для переноса строки в режиме "дисплей", применяется символ "вертикальная линия" - С.В.Л. (это в режиме "EXT. MODE" нажимаем SYMBOL SHIFT + S). Код этого символа равен 124. Но в кодах ASCII в символьном наборе КОИ-7 "НС", который применен в нашем варианте MF 09, коду 124 соответствует буква "Э". Поэтому, как только в тексте встретится буква "Э", она напечатана не будет, а произойдет перенос текста на новую строку.

Устранить можно и эту проблему, заменив символ, дающий команду переноса на какой-нибудь другой. Вопрос только: на какой? Ведь любой другой символ может встретиться в файлах данных, введенных на нерусифицированной программе MF 09.

Я считаю, что мне удалось найти удачное решение. Сделайте:

```
POKE 58237,0
POKE 63812,0
```

Теперь вместо С.В.Л. для переноса строки будет использоваться любой символ, с кодом больше 127. Это любое ключевое слово, например "STOP" или "AT" или любое другое, которое Вам удобнее набирать. При этом в файл данных будет занесен код 0 (независимо от того, какое ключевое слово Вы ввели). Символ с кодом 0 - это в кодах ASCII управляющий символ "NUL", он для печати не используется. Но в режиме "дисплей" теперь по этому

символу произойдет перенос на новую строку. Если же Вы при помощи переделанной программы будете просматривать файлы данных, введенные на непеределанной программе, то все символы переноса строки, находящиеся в тексте, станут видимыми.

Если при пользовании русифицированной программой MF 09 возникнут какие-либо неудобства, связанные с переносом строки, то напишите в ИНФОРКОМ, я готов придумать что можно сделать. (Например, можно написать несложную программу обработки уже готовых файлов данных, заменяющую все встречающиеся в тексте С.В.Л. на управляющие символы "NUL".)

Многие проблемы, существенные и несущественные, связанные с русификацией MF 09, мы решили. Но осталась еще одна очень важная проблема, пока еще нерешенная. Это сортировка по русскому алфавиту. Как производится сортировка по латинскому алфавиту? Очень просто. В кодах ASCII латинские буквы уже расположены в алфавитном порядке. Значит, располагая записи по возрастанию кодов, можно производить сортировку. С точки зрения программирования это довольно простая задача. С русским алфавитом сложнее. Здесь нет никакой связи между алфавитом и кодом символа. Поэтому сортировка по тому же принципу, что и латинского текста не годится. Причем проблема сортировки по русскому алфавиту гораздо шире, чем рамки программы MF 09. По-моему проблема вполне достойна того, чтобы вынести ее на общественный "Форум". Может быть, кому-нибудь из читателей удастся преодолеть ее? В таком случае заявите в ИНФОРКОМ. В свою очередь, я тоже пытаюсь найти приемлемый вариант решения этой задачи и когда закончу работу, сообщу об этом.

ЗАКЛЮЧЕНИЕ.

Мы закончили перевод программы MF 09 на русский язык. Выполняя эту работу, Вы познакомились с некоторыми методами, которые могут применяться для этого и немного "набили руку". Теперь Вы легко сможете применять "русский язык" при разработке своих программ.

Что же касается перевода фирменных программ, то тут следует отметить некоторые моменты. Приобретенные знания помогут Вам во многих случаях, однако будут встречаться разные "сюрпризы и хитрости", придуманные авторами (как было в MF 09, например, код последнего символа на 128 больше и др.). Процедуры вывода на экран могут быть самыми различными. Кроме того, трудности начнутся с загрузки программы, так как при нажатии "BREAK" программа "зависает" или происходит рестарт компьютера. Но даже если Вы удачно "взломаете" защиту загрузчиков программы, надо учесть, что очень многие коммерческие программы закодированы и поэтому, просматривая дамп памяти, мы ничего похожего на текстовые сообщения увидеть не можем. В этом случае надо, дизассемблируя программу, начиная со стартового адреса, найти процедуру декодирования, написать аналогичную процедуру в машинных кодах, раскодировать программу и только после этого заниматься переводом. Закодировать программу после перевода нет необходимости, проще изменить процедуру запуска, исключив элементы декодирования. Без знания машинных кодов Z-80 решить эту проблему, конечно невозможно.

* * *

?

"ИНФОРКОМ" получает массу писем от начинающих программистов с просьбами посоветовать им над чем бы стоило поработать, в какой области приложить свои усилия.

Мы считаем, что лучше всего в этой деле равняться на классиков. Поэтому пользуясь тем, что этот номер "ZX-РЕВЮ" является апрельским, мы поздравляем дорогих читателей с праздником 1-го апреля и рекомендуем обратить внимание на последние достижения всемирно известной фирмы PFA.

Мастеров "самого быстрого ксерокса" просим при перепечатке (или ином воспроизведении информации) не стесняться и смело указывать на первоисточник.

Проблем совместимости больше не существует.

Уникальная развлекательная программа разработана и выпущена фирмой PRESENT FROM AFRICA. Это первая версия имитатора "Super Real Bomb Simulator".

Программа отличается от всех известных аналогов тем, что ее не надо инициализировать, настраивать и вообще ее не надо загружать. Она не требует ни джойстика, ни клавиатуры.

Вам достаточно вставить кассету в магнитофон, нажать клавишу PLAY и спокойно ждать, когда она сработает.

Фирма особенно удовлетворена тем, что до сих пор от зарегистрированных пользователей не поступило ни одной жалобы (письма и заявления от родственников и друзей покойного фирма традиционно не рассматривает).

За особо дополнительную плату программа может поставляться на диске.

Решается вопрос о перспективах маркетинга программы в термоядерном исполнении (так называемый экспортный вариант). Это первая в мире программа, работающая на компьютере любой системы!

Проблем с продуктами больше нет!

Впервые в мире фирмой PRESENT FROM AFRICA выпущен имитатор продукта питания. Это завоевавшая всемирную известность программа "Shashlyik Basturma Simulator".

Программа состоит из двух частей. Первая часть - аркадная игра по нанизыванию кусочков баранины и лука на палочку.

Вторая часть требует специального блока питания (поставляется вместе с программой).

Подключив его и запустив программу, Вы уже через несколько минут сможете насладиться ароматом настоящего шашлыка, исходящим от Вашего компьютера.

Если запах немножко не тот, значит Вы забыли замочить компьютер в слабом растворе уксуса или в сухом вине. Сделать это лучше всего накануне вечером. Не забудьте добавить лука репчатого, перца молотого и посолить.

Суперновинка!

Развивая плодотворную идею программы "Shashlyik..." фирма PFA подготовила совершенно оригинальный программный продукт, объединяющий в себе абсолютно все известные жанры игровых, прикладных и вкусовых программ. Эта новинка поступила на прилавки под названием "Chuckie Egg F16".

Сначала Вы работаете в режиме "TOOLKIT", конструируя себе цыпленка-табака по вкусу. В этом режиме программа является также обучавшей по анатомии пернатых.

Увлекательный аркадный эффект возникает, если Вы ошибетесь и Ваш цыпленок начнет походить на утку или гуся - Вам придется отбиваться от группы сумасшедших яблок (APPLES), желающих в нем расположиться.

Вы можете выбрать своему цыпленку лапки, ножки, крылышки, грудку по вкусу, можете оснастить его хвостовым, оперением, подвесить дополнительные топливные баки, ракеты "воздух-воздух" и "воздух-земля", установить приборы ночного бомбометания и многое, многое другое.

Дальше программа может развиваться как STRATEGY/ACTION, а можно сразу перейти в кулинарный модуль с помощью описанного выше блока питания. Но в последнем случае способ предпусковой подготовки компьютера путем замачивания будет несколько отличаться, впрочем вместо инструкции к программе прилагается книга о вкусной и здоровой пище.

Проблем с ошибками больше нет!

Известно сколько неприятностей доставляют каждому работающему с компьютером сообщения об ошибках.

Наконец-то эта проблема решена всемирно известной фирмой PRESENT FROM AFRICA.

Покупайте новую программу "O.K. Simulator" (Имитатор O.K.)

Что бы Вы ни делали с компьютером, он всегда выдает радостное сообщение O.K.

Программа проста в работе. С ней успешно (и без ошибок) работают грудные дети. Группа академиков освоила работу с этой программой всего за полтора месяца во время учебного круиза по маршруту Одесса-Афины-Кейптаун-Сингапур-Токио и обратно. Теперь у них тоже все O.K.

По окончании круиза все академики высказали два пожелания и поставили один проблемный вопрос.

Первое пожелание - сократить количество утомительных нажатий клавиш хотя бы до двух (большее количество мешает круизу).

Второе пожелание - повторить круиз для окончательного решения вопроса о рекомендации программы ко всеобщему внедрению.

Проблемный вопрос: почему во всех городах мы побывали по два раза, а в Токио только один?

Сейчас фирма работает над новой версией, печатающей сообщение "O.K." в любых необходимых количествах автоматически, но, как сообщают, работа над проектом может затянуться из-за неразрешимости проблемного вопроса. Он стал известен в академических кругах под названием "Токийский синдром".

Ряд крупных академических институтов уже взялись за его отработку, по крайней мере на моделях. В модельных экспериментах функции Токио выполняли Лондон, Париж и Нью-Йорк, но проблема еще далека от разрешения. На очереди Сан-Франциско и Рио-де-Жанейро.

Происшествия!

В московском отделении фирмы PRESENT FROM AFRICA совершено крупное хищение. Украдены персональный компьютер, монитор, курсор и джойстик.

При обыске у подозреваемого обнаружены компьютер, джойстик и монитор.

Фирма гарантирует всем гражданам, которым известно что-либо о местоположении курсора, призовое вознаграждение своим программным обеспечением.

Без курсора фирма не может продолжать работы над замечательными проектами.

СОВЕТЫ ЭКСПЕРТОВ

PROFESSIONAL TENNIS

"Dinamic" 1990 г.

Эксперт М.Аксюта
г. Днепропетровск.

Эта игра относится к имитаторам спортивных игр. Вам предоставляется возможность хорошо потренироваться и сыграть на Уимблдонском турнире со звездами мирового тенниса. Но начнем по порядку. После загрузки игры перед Вами основное меню:

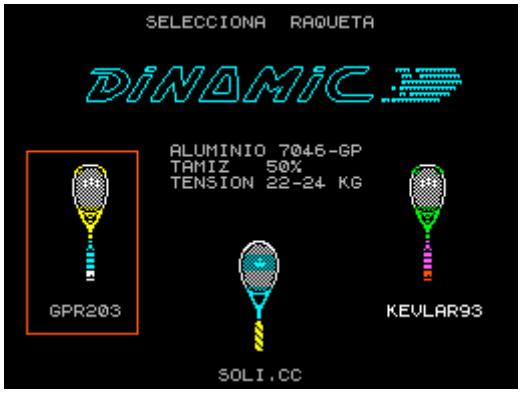
0. TORNEO (турнир)
1. CONTROLES (органы управления)
2. EQUIPAMIENTO (экипировка)
3. ENRENAMIENTO (тренировка)
4. CARACTERISTICAS (условия игры)

1. Controles

В этом режиме Вы можете ввести раскладку клавиатуры для управления двумя игроками: TECLADO_A и TECLADO_B

2. Equipamiento

Здесь Вы можете выбрать себе экипировку, а именно: - сначала кроссовки, а затем ракетку.



3. Entrenamiento

Вам предлагается потренироваться перед игрой на Уимблдоне. Вы можете выбрать тренировочный режим из следующего субменю:

1. PARTIDO
2. SAQVE
3. VOLEA

Выход из режима тренировки - нажатием клавиши "R".

Saqve - отработка подачи. Выбираете себе управляющий орган (например Кемпстон-джойстик) и проводите тренировку.

Нажмите кнопку "огонь" и переместите появившуюся на экране мишень в то место площадки, куда бы Вы хотели послать мяч. Подавать мяч нужно обязательно в накрест лежащий квадрат поля противника. Так, если Вы находитесь в правом нижнем квадрате, то подавать можно только в левый верхний. При ошибке или попадании мяча вне разрешенной зоны загорается сообщение и Вам дается вторая попытка, как в настоящей игре. Отработав подачу, можно нажатием "R" вернуться в субменю и перейти к отработке приема мяча.

Volea - отработка приема мяча. Ваш игрок может свободно перемещаться по площадке. Удар он выполняет нажатием кнопки "огонь". Если одновременно нажать "огонь" и "вверх", то удар получается более сильным, но пользоваться этим следует с осторожностью - когда Вы находитесь на задней линии или противник вышел близко к сетке, иначе мяч после Вашего удара может улететь за пределы площадки.

Если совместить удар с движением влево или вправо, то можно соответственно послать мяч влево или вправо, но это происходит не всегда и зависит от того, как Вы стоите относительно мяча. Так, если Вы стоите слева от мяча, он может полететь либо влево, либо прямо и наоборот, если Вы стоите справа от мяча во время приема.

Потренировавшись в приеме подачи и закончив тренировку клавишей "R", Вы можете сыграть пробную игру с компьютером, нажав клавишу "1" - Partido и выбрав для него силу игры "ORDENADOR".

Если Вас при этом не устраивают кроссовки или ракетка, вернитесь в главное меню и поменяйте их.

4. Caracteristicas

Вам предлагается следующее меню:

1. TIPO DE PISTA....
2. CAMBIO DE CAMPO....
3. NUMERO DE SET....

Tipo de pista - выбор покрытия корта, на котором Вы будете тренироваться. Вам будут предложены следующие варианты:

- TIERRA - грунтовое,
- HIERBA - травяное,
- RAPIDA - пластиковое.

Cambio de campo - определяет будет ли производиться смена сторон поля по ходу игры.

Если Вы хотите играть все время с одной стороны, выбирайте NO (НЕТ), если же

желаете проводить смены полей, выберите SI (ДА). Начинающим рекомендуется выбирать NO.

Numero de set - количество сетов (1-3-5), которое будет разыграно с компьютером при тренировке.

0. Torneo

Здесь разыгрывается турнирная игра. Сначала Вам предложат выбрать количество участников: 1-4 и ввести их имена, после чего Вы попадете в следующее субменю:

0 - CLASSIFICATION ATP
1 - INICIAR TEMPORADA
2 - CONTINUAR TORNEO
3 - TABLA DEL TORNEO

Classification ATP – таблица участников турнира, их номера. Вы всегда начинаете с самой низшей ступени.

Iniciar temporada - ввод имени игрока.

Continuar torneo - выбор управляющего органа и начало (продолжение) игры. Игра может быть прервана для перенастройки нажатием клавиши "R".

Tabla del torneo - таблица пар участников.

Итак, предварительно потренировавшись и выиграв у компьютера 3-4 сета, Вы можете принять участие в настоящем Уимблдонском турнире. Разве Вам не хочется победить Беккера или Лэндла?!

SNOOKER

Эксперт Ескевич А.А. г. Новосибирск.

"Снукер" - разновидность биллиарда.

Ныне эта игра - одна из популярнейших не только на туманном Альбионе, но и в более, чем тридцати других государствах. В Великобритании действует огромная сеть клубов, объединяющая более чем 6 млн. любителей снукера - это при том, что все население страны - 54 миллиона человек!

На биллиарде британцы играют уже свыше 100 лет. Таков же возраст и снукера. Происхождение этой игры точно не установлено. По одним сведениям снукер был придуман в Индии, а затем завезен в Англию, где и получил признание и широкое распространение. Другие знатоки утверждает, что в 1875 году желание как-то разнообразить обычный биллиард толкнуло одного из младших офицеров британской армии на то, чтобы включить в игру шары другого цвета.

Что же касается самого слова "снукер", то оно произошло от английского "snook", что в переводе означает "длинный нос". Снукерами же в британской армии называли юных кадетов только что начавших службу.

Так что же такое снукер?

Точный удар, уверенная срезка - и по зеленому сукну с мягким стуком раскатилось 22 разноцветных шара, останавливаясь в неожиданных положениях...

Это снукер - почти неизвестная в нашей стране игра на биллиарде. Изобретенная во второй половине прошлого века, игра оказалась сложнее, чем известные "Американка" и "Русская пирамида". Она скорее пробуждает интерес к сложным движениям шаров, чем к выигрышу.

Правила игры.

В снукер обычно играют вдвоем, но могут играть и несколько игроков.

Принцип очередности удара простой: если удар не принес очков, бьет следующий игрок.

Цветные шары оцениваются в зависимости от цвета:

- 15 красных шаров - по 1 очку;

- один желтый - 2 очка;
- один зеленый - 3 очка;
- один коричневый - 4;
- один синий - 5;
- один розовый - 6;
- один черный - 7 очков.

Белый шар - это биток. Только им можно бить по остальным шарам.

Разбивающий пирамиду может установить биток в любом месте зоны "дома" (площадь, ограниченная полукругом) - там, откуда удобнее бить.

Исходная расстановка шаров показана на рисунке. Вместо цвета шагов мы поставили их оценку в очках.

Первый ударом нужно сыграть только красный шар, а если биток заденет за какой-нибудь другой, игроку засчитывается ошибка и списываются очки в зависимости от ценности затронутого шара. Самый первым ударом важно не только удачно разбить пирамиду, но и отогнать биток как можно дальше от нее, создавая другому игроку позицию посложней.

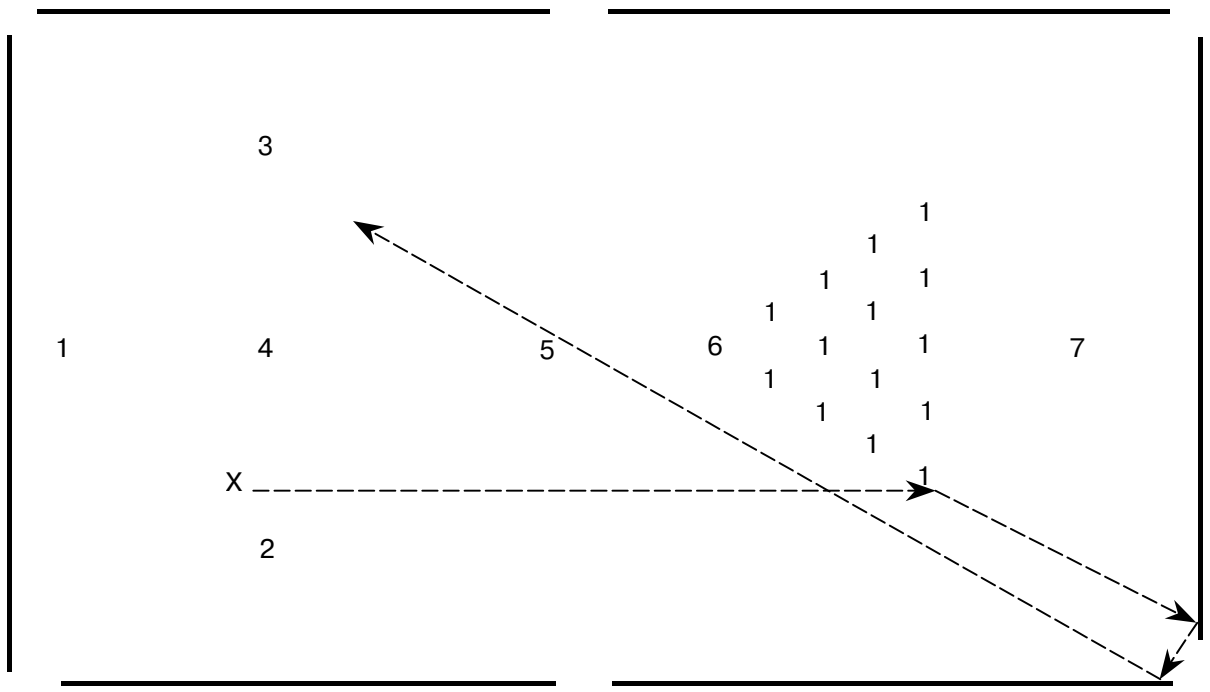
На рисунке классический начальный удар показан пунктиром.

Шары кладутся в любую из шести луз. За уложенный красный шар засчитывается одно очко. После этого биллиардист имеет право удара по любому цветному шару (цветными называются все шары, кроме красных и битка). Если начинающий игрок более уверен в каком-нибудь шаре, то он может бить и красный - один, другой, а потом, подогнав биток к цветному, уложить его в лузу.

Нужно твердо запомнить, что перед каждым цветным шаром должен быть забит красный!

Профессионалы снукера иногда усложняют игру тем, что после красного шара можно играть только цветной шар. Такая тактика всегда помогает набирать крупные серии. (Серия - сумма выигранных очков в течение одной очереди).

Например, вот какая серия может получиться, если было уложено в лузу несколько шаров: красный (1 очко), зеленый (3), снова красный (1 очко), розовый (6), красный (1), синий (5) и т.д.



Положенный в лузу красный шар выходит из игры. До тех пор, пока на столе есть хотя бы один красный шар, можно забивать цветные шары в любом порядке по своему выбору, но всякий раз перед цветным игрок должен уложить в лузу красный шар. Забитые цветные шары на этой стадии игры выставляются на те точки стола, где они стояли в начале игры.

Первая часть игры кончается, когда все красные шары забиты в лузы. Теперь правила по отношению к цветным шарам становятся более строгими: их надо класть в лузу в порядке их ценности, начиная с желтого (забив желтый, можно класть зеленый, затем коричневый, синий, розовый и черный. Уложенные таким образом цветные шары в игре уже больше не участвуют. Однако, если был сделан неправильный удар - биллиардист совершил ошибку при ударе, - цветной шар выставляется на свою основную позицию.

Игра подходит к завершению, когда на столе остается лишь 2 шара: "последний черный" и биток.

Бывает, что у партнеров примерно равное количество очков. Поэтому все решают эти семь очков за черный шар. Когда "последний черный" сыгран, или совершена ошибка - игра заканчивается. Редко, но может случиться и ничья - в этой случае для выявления победителя разыгрывается повторение "последнего черного". Черный шар выставляется на свое место, и по нему играют битком из "дома".

Когда знатоки биллиарда делают красивую игру, они не только укладывают шары в лузу, но при этом еще и маневрируют битком по всему полю. В результате маневра противнику может быть поставлен снукер - это кульминация игры, ее высший смысл, который и дал ей такое название. Снукер - это ловушка, из которой может выбраться только осторожный и хладнокровный игрок.

Снукер - это позиция, из которой биток не может достать тот шар, который по правилам должен быть сыгран: он либо "замазан", либо даже полностью закрыт, но шанс у Вас все-таки есть - выручит меткий бортовой удар.

А теперь ошибки, часто встречающиеся в снукере.

Случается, что игрок промахивается по шару, но это еще полбеды: этот шар, оттолкнувшись от борта, может задеть не тот, который должен быть сыгран. Если биток завалится в лузу, даже вслед за забитым шаром, эта ошибка наказывается штрафом в 4 очка. При назначении штрафа берется во внимание оценка шара, который следовало сыграть, или оценка шара, который сыгран неправильно. По правилам всегда берется наивысшее количество очков. Поэтому, если был правильно сыгран красный (или желтый, зеленый, коричневый) шар, но биток закатился в лузу, то списывается минимальное количество очков - 4; если же в таком положении окажется черный шар, штраф составит уже 7 очков (соответственно 6 - за розовый и 5 -за синий).

Когда нужно играть, например коричневый шар, и игрок делает правильный удар по нему, а закатывается какой-нибудь другой, то очки не списываются, а уложенный шар выставляется по правилам.

Подсчитано, что теоретический предел возможностей игрока в снукер - набрать за одну серию 155 очков (напомним, что серия - это сумма очков, выигранных за одну очередь).

Выдающийся мастер снукера англичанин Джо Дейвис установил в 1955 году рекорд мира, набрав серию в 147 очков. Может быть, Вы попробуете превзойти мастера?

Настройка программы.

После загрузки программа выдает следующее меню:

CONTROL OPTIONS	ОРГАН УПРАВЛЕНИЯ
1. KEYBOARD	1. КЛАВИАТУРА (O,P,Q,A,Enter)
2. KEMPSTON	2. КЕМПСТОН-ДЖОЙСТИК
3. CURSOR KEYS	3. КУРСОР (5,8,6,7,0)
4.INTERFACE 2	4. ИНТЕРФЕЙС 2 (Синклер-джойстик)
Press 5 to Continue	Нажмите "5" для продолжения.

Выбрав орган управления, нажмите "5" - появится следующее меню:

Game options:	Опции игры:
1. ONE PLAYER	- один игрок
2. TWO PLAYER	- два игрока
3. CURSOR SOUND ON	- звук курсора включен

4. CURSOR SOUND OFF - звук курсора выключен
5. LONG GAME - длинная игра
6. SHORT GAME - короткая игра
7.CURRAH SPEECH OFF - речь выключена
8. CURRAH SPEECH ON - речь включена
Press any other key to play Для начала игры нажните любую другую клавишу.

Сделав выбор, нажмите клавишу, и на экране появится биллиардный стол с расставленными шарами.

Структура экрана.

В левом верхнем углу показаны набранные Вами очки (POINTS) и сумма штрафных очков за ошибки (FOULS), в правом верхнем углу - номер серии (VISITS).

Внизу показаны сила удара (POWER), биток для выбора направления вращения шара (SPIN) и то, какой шар нужно сыграть в данный момент (например, "RED WANTED" - "Требуется красный").

После того, как Вы забьете красный шар, появится надпись "COLOUR WANTED" ("требуется цветной"). Выберите цветной шар, который Вы хотите забить и нажмите клавишу, соответствующую стоимости этого шара (2-желтый, 3-зеленый, 4-коричневый, 5-синий, 6-розовый, 7-черный). Например, Вы решили сыграть черный шар - нажмите "7" и надпись "COLOUR WANTED" сменится на "BLACK WANTED". После этого Вы обязательно должны бить по черному шару. Если промахнетесь, или биток заденет сначала за какой-либо другой шар, Вам начислят штрафные очки за удар не но правилам.

Начало игры.

Перед началом игры Вы должны установить биток в любой точке "дома". Для этого переместите курсор в выбранное место и нажмите "огонь" (эту операцию Вам придется проделывать всякий раз после того, как биток угодит в лузу).

Теперь Вы должны нанести удар. Это несложно, но требует тщательной подготовки:

1. Установите курсор в то место, куда хотите направить биток и нажмите "огонь".

2. Установите силу удара клавишами влево/вправо, пользуясь шкалой внизу экрана, и вновь нажмите "огонь".

3. Установите курсор на ту точку битка, куда Вы хотите "ударить кием" (проще всего бить в центр битка, но если Вы научитесь "срезать" шар, это будет весьма полезно) и вновь нажмите "огонь" - удар произведен.

Теперь, когда правила известны, остается лишь приобрести навык - и победа будет за Вами! Дерзайте, будущие мастера международной игры на биллиарде!

QUAZATRON

Сегодня мы представим Вам программу QUAZATRON, разработанную Стивом Тернером, с работами которого наши читатели уже знакомы по циклу статей "Профессиональный подход".

В мае 1986 года известный журнал "Sinclair User" присвоил этой программе свой знак:

Sinclair User
CLASSICS

Этот красный с золотом знак присваивается только тем программам, которые не просто являются лучшими в своем жанре, а открывают новые жанры или новые направления в своем жанре или открывают новые, ранее неизвестные приемы и методы. Для программиста получение такого знака может быть и не столь значительно, как получение премии "Оскар" для кинорежиссера, но не менее дорого.

Конечно, сейчас уже далеко не 1986 год, но игра эта еще во многом свежа. Возможно, что в силу определенной сложности, она долгое время выпадала из поля зрения наших пользователей и об этом приходится только сожалеть. Ведь программа настолько многопланова и разнообразна, что возвращаться к ней можно много и много раз. Она представляет довольно гармоничный баланс между играми аркадного и стратегического жанра.

Квазатрон - это огромный подземный технополис, населенный роботами. Расположен он на планете Квартех. Несколько попыток земных экспедиций по проникновению в этот комплекс закончились неудачно. Вам предлагается попробовать свои силы и захватить город с помощью специального робота MECHNOTECH KLP2.

KLP 2 был сконструирован в качестве разведывательного робота и провел специальную подготовку для участия в исследовательских экспедициях в составе разведывательных команд. К сожалению, его пришлось отстранить от групповых операций в связи с его патологической страстью разбирать другие устройства (обратите внимание на то, что KLP 2 читается по-английски как KLPTWO - очень похоже на КЛЕПТО - и вспомните, что непреодолимое желание тащить к себе все, что плохо лежит, называется клептоманией).

Отправленный для рутинной патрульно-таможенной службы на границы галактики, КЛЕПТО, тем не менее, завоевал известность и стал общественным героем, когда утилизировал левую ногу знаменитого межзвездного пирата, робота-андроида человекоподобного типа.

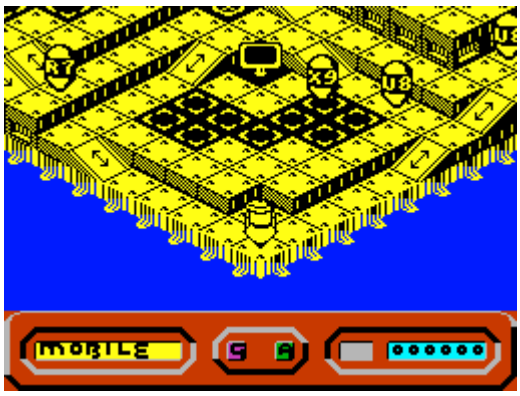
Вот почему о нем вспомнили, когда готовили миссию QUAZATRON.

Находясь под Вашим управлением, KLP 2 может работать в двух основных режимах:

- MOBILE - режим перемещения;
- GRAPPLE - режим захвата.

Когда он работает в режиме перемещения, игра развивается как обычная аркадная игра. Вы можете перемещаться по трехмерному уровню технополиса, можете периодически отстреливать враждебных роботов, можете перемещаться с уровня на уровень, можете подключаться к установленным в разных местах дисплеям или подпитываться энергией в специальных точках.

Подключение к дисплею позволяет войти в информационную сеть технополиса, через которую открывается доступ к самой разнообразной информации. Вы можете увидеть план уровня, схему расположения уровней в городе, можете получить справку о роботах-противниках, узнать их спецификацию.



Подзарядка энергией - тоже совершенно необходимое действие.

Как бы Вы ни работали, Ваш успех зависит от наличия энергии. Если ее мало, загорается надпись POWER, раздается неприятный звуковой сигнал и, если Вы сломя голову не броситесь к источнику энергии, можете считать, что миссия закончена. Самое интересное - это то, что для указания уровня энергии нет никаких счетчиков, никаких шкал, никаких указателей. Над головой KLP-2 есть небольшая "шапочка". Скорость, с которой она вращается, и зависит от уровня энергии.

Надо сказать, что обычное аркадное путешествие по этажам города закончится очень быстро. Дело в том, что среди населяющих его роботов есть такие, которые не дадут Вам ни малейшего шанса остаться в живых, если Вы попробуете перебежать им дорогу.

Вот тут-то аркадная игра и превращается в стратегическую (а если хотите, то в деловую, ведь менеджмент - тоже стратегия, хоть и отличается от военной). В этой фазе игры Вы работаете со своим роботом в режиме GRAPPLE. В этом режиме Вы можете войти в непосредственный контакт с противником, подключиться к его внутренним цепям и попытаться его перепрограммировать, если он не сделает то же с Вами. Здесь Ваш успех будет зависеть от быстроты реакции, от оснащенности Вашего робота и конечно от его энергетических возможностей.

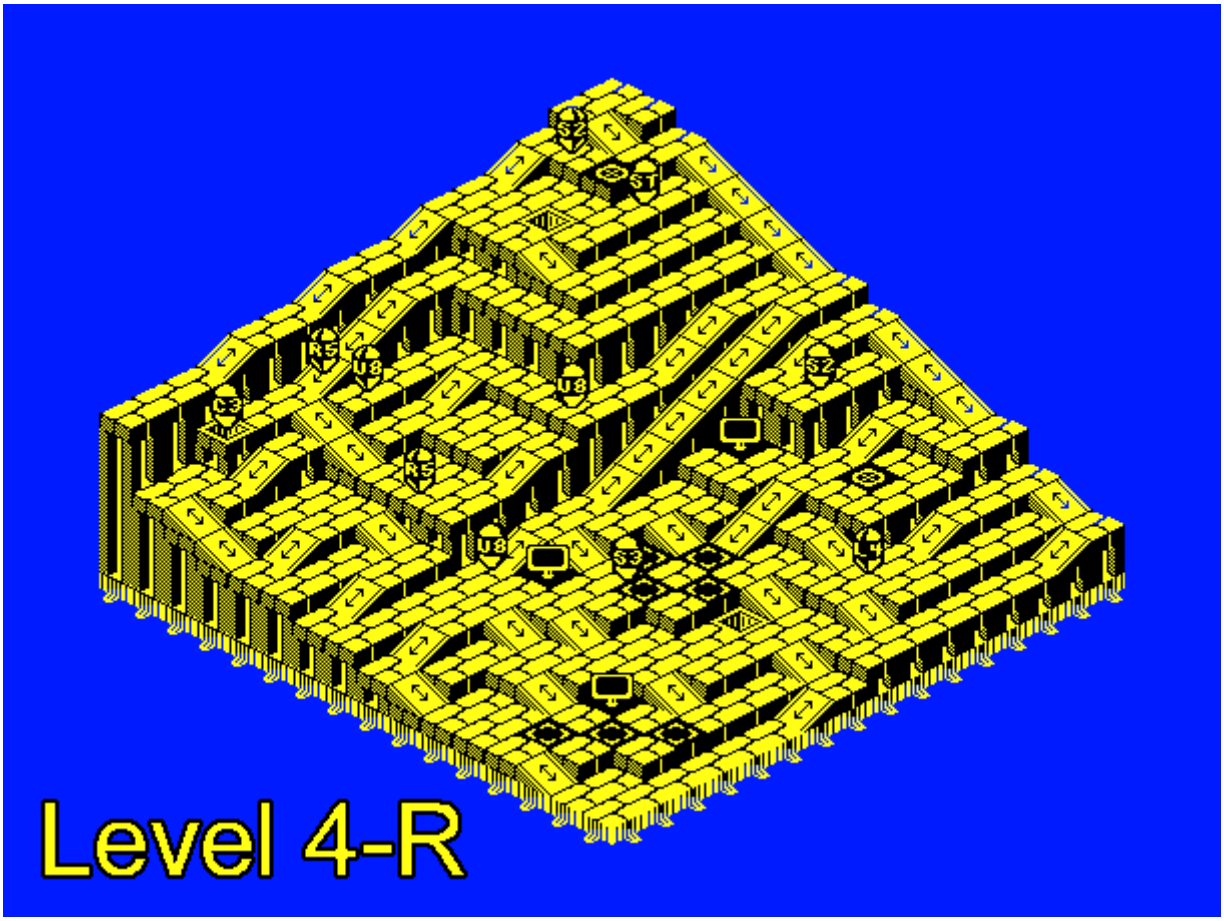
Схватка в режиме GRAPPLE представляет собой аркадную вставку в основную игру и развивается на особом экране, но мы о ней писать не будем, предоставив чуть ниже слово нашему эксперту Курбацкому Андрею Александровичу из Казани, накопившему определенный опыт в этом деле.

Если Вам удастся захватить хотя бы семь блоков из двенадцати в логических цепях Вашего противника, то схватка считается закончившейся в Вашу пользу, робот противника переходит в подчиненное состояние и Вы можете приступить к его тестированию. Тестирование покажет, какие модули остались неповрежденными в результате схватки и Вы сможете принять решение о том, что Вам можно использовать для себя. Так, постепенно каннибализируя павших противников, Вы можете улучшать оснащенность своего робота.

Вот где открывается поле для стратегии.

Во-первых, Вы должны сами принять решение о том, с каким роботом Вам имеет смысл входить в тесный контакт, а какого достаточно просто расстрелять. Решение Вы принимаете, зная модель робота по его бортовому номеру, его спецификацию и состав его модулей.

Противник не может быть намного мощнее Вас - это шаг к могиле, но и не должен быть слишком слабым, т.к. если с него нечего взять, то и нет смысла расходовать на него время и силы, разве что для тренировки.



Во-вторых, сама схватка должна проходить с Вашей стороны достаточно интеллектуально, ибо после иного боя Вы можете уже не найти у противника необходимых Вам компонентов в исправном состоянии. Попробуйте в начале игры перестрелять мощных боевых роботов противника и Вам впоследствии уже негде будет взять особо ценные модули, необходимые для захвата всего города.

В-третьих, утилизация для своих целей исправных блоков от захваченных роботов тоже должна проходить с умом. Возьмите очень мощный двигатель - и с Вашим скромным энергетическим отсеком Вы устанете бегать на подзарядку, а если один раз не добежите - игре конец. Возьмите самое изощренное орудие и система защиты и на Вашем слабеньком шасси KLP-2 будет еле ползать по поверхности, а падение с небольшой высоты станет приводить к значительным потерям энергии. И так далее и тому подобное.

Таким образом, игра представляет такой сплав стратегических решений, что ее серьезный анализ вполне может занять Вас не на одну неделю и доставит огромное удовольствие.

В заключение этого общего обзора программы прежде, чем предоставить слово нашему эксперту из Казани, мы приведем небольшое интервью со Стивом Тернером, и заглянем краем глаза на ту кухню, где готовятся такие вкусные блюда для Вашего компьютера.

* * *

В: Скажите пожалуйста, как долго шла работа над программой QUAZATRON?

О: Эта работа затянулась более, чем на шесть месяцев. Я знаю, конечно, что многие заявляют о том, что вполне пристойная аркадная игра может быть сделана и за два месяца, но у меня особый случай - ведь и музыку и графику я тоже делаю сам. Я вообще все делаю сам, хотя концепцию программы и наброски широко обсуждаю с коллегами.

В: Вам удается обыграть собственную программу?

О: Вы знаете, после полугода работы с машинным кодом, я месяца два вообще не мог смотреть на эту программу. Потом постепенно успокоился и нередко играю в нее. Но ответить на Ваш вопрос я не могу, ведь я не играю на выигрыш. У меня совсем другая цель - я ищу способы, которыми можно было бы "завалить" программу.

В: Когда Вы поняли, что эта программа Вам удалась?

О: Только в самый последний момент. Здесь надо знать мой метод работы над программами. Я не вижу ее до конца работы, все процедуры отрабатываются по отдельности или группами, но окончательная сборка производится в самый последний момент. Конечно, многие вопросы, особенно связанные с графикой, приходится обкатывать на моделях, ведь здесь важно быстродействие процедур с одной стороны и художественное впечатление с другой, но это совсем не то же, что работа с программой.

В: Что Вам не удалось в программе?

О: Отсутствие гладкого скроллинга экрана. У меня, конечно, были идеи как его обеспечить, но неминуемо приходилось бы жертвовать чем-то другим и сейчас, оглядываясь назад, я полагаю, что так получилось даже лучше.

В: Ваша программа чем-то напоминает PARADROID, выпущенный незадолго до нее для "Коммодора-64"? Она была в какой-то степени для Вас источником вдохновения?

О: Нет, все гораздо проще. Просто мы с Эндрю Брейбруком, автором "Парадроида", работаем в одной комнате. Естественно, по ходу работы идет постоянный обмен идеями. Кроме того, очень часто совместно обсуждаются вопросы дизайна. А если уж говорить об источнике вдохновения, то им явилась в некоторой степени программа MARBLE MADNESS и, в какой-то степени, ALIEN-8.

В: И последний вопрос. В самом начале программы звучит удивительно сложная музыка, если вспомнить, что у "Спектрума" только один канал...?

О: Да, конечно, каждый знает, что звуковая система "Спектрума" совершенно не соответствует прочим его возможностям. Но мне удалось программным путем, основываясь на системе прерываний, подготовить пакет процедур, имитирующих многоканальный звук. В конце концов, все это свелось только к манипуляциям фазой и частотой.

* * *

Теперь мы рассмотрим практические вопросы, связанные с боевыми действиями на планете Квартех. Слово имеет Курбацкий А.А., г.Казань.

Все роботы, с которыми Вы можете встретиться в технополисе, имеют бортовую маркировку, с помощью которой можно получить более подробную информацию об этом роботе.

Маркировка состоит из буквы и цифры. Буквенная маркировка указывает на класс робота, а цифровая - на положение робота в этом классе.

Классы роботов:

A (AUTOMATION)
B (BATTLE)
C (COMMAND)
L (LOGIC)
O (MEDIC)
R (REPAIR)
S (SECURITY)
U (UTILITY)
X (MENIAL)

Не входят в эту классификацию только два типа роботов. Это AB (ANDREWOID) И ST (PROGRAMMER).

Цифровая кодировка (от 0 до 9) указывает на статус робота в своем классе. Чем она ниже, тем выше статус робота, тем более совершенными системами он вооружен, тем больше доступный для него запас энергии.

Все роботы классов AB, ST, A и S относятся к высшей касте и защищены наиболее сильно. К средней касте можно отнести роботов L, O, R. К низшей касте относятся роботы U и X и защищены они довольно-таки слабо и в цифровой нумерации не опускаются ниже 7.

Нажав клавишу "огонь", Вы переходите в режим GRAPPLE и, коснувшись любого робота, входите в бой. Бой за перехват управления выглядит так (см. рис. 1).

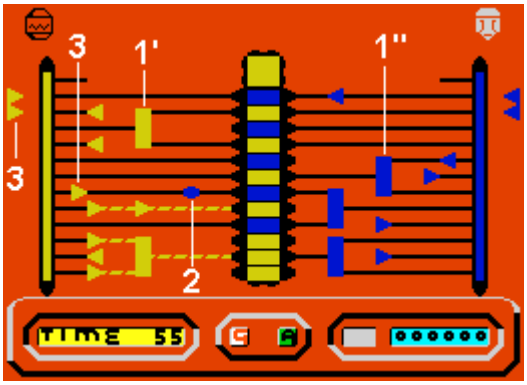


Рис.1

В начале игры у Вас всего три энергетические капсулы (3) которые Вы можете подсоединить к любому проводу, соединенному центральным блоком (см. рис.). Иногда на проводах стоят такие устройства как сумматоры (делители) (1), инверторы (2) и дополнительные энергетические капсулы (3). С помощью делителя (1') можно от одного провода запитать два и более отсека, сумматор же (1'') наоборот от несколько проводов запитывает один отсек. Инвертор (2) обращает Ваш сигнал и превращает в сигнал противника, так что Вам он мешает, а не помогает. Дополнительные энергетические капсулы могут помочь Вам увеличить время подпитки отсека, так как при израсходовании энергии в Вашей капсуле, подпитка блока прекращается.

Некоторые провода могут быть оборваны. Сбоку указано количество энергетических капсул, имеющихся в Вашем распоряжении.

Сначала Вам предоставляется право выбрать сторону, с какой Вы хотите вести бой. Это очень важный момент, от которого в значительной степени будет зависеть успех боя. После выбора стороны Вы можете с помощью клавиш "вверх", "вниз" подвести энергетическую капсулу к проводу, к которому Вы хотите подключиться, и нажать клавишу "огонь". Блок центрального отсека закрашивается. Конечная цель боя - захватить отсеков больше, чем Ваш противник. Всего на бой дается 60 секунд, желательно подключиться чуть позже, чем противник, так как это позволит при Вашем недостатке энергетических капсул продержаться как можно дольше и, в конечном итоге, захватить энергетический отсек. Если схватка закончится вничью, то Вам предоставляется другая энергетическая схема.

В случае победы противника, если это первый бой, Ваш робот разрушится, а если эта схватка происходила после того, как Вы уже кого-либо победили, то теряются все захваченные ранее блоки и оружие, а силовая установка остается почти без энергии. Тогда Вам после схватки нужно быстрее идти к энергетической клемме, расположенной на этаже, и подпитаться энергией. Маркировка последнего побежденного робота отображается в окне чуть правее центра пульта. Вот список устройств, которые Вы можете захватить, расположенный по степени приоритета.

Двигательная установка (DRIVE).
LINEAR MK 1
LINEAR MK 2
LINEAR MK 3
GRAVITRONIC MK 1
GRAVITRONIC MK 2
HEAVY DUTY
DUAL LINEAR
ULTRAGRAV

Оружие (WEAPONS)
PULSE LASER
DUAL LASER DESINTEGRATOR
AUTOCANNON

DISRUPTOR

Наибольшей разрушительной способностью обладает DISRUPTOR, который поражает всех роботов, видимых на экране. AUTOCANNON стоит на втором месте: пробивает любой корпус, но стреляет только в одном определенном направлении.

Имеет значительный вес. Все прочее оружие пробивает лишь определенные корпусы, да и то не всегда с первого раза.

Силовые установки (POWERS)
CHEMIFAX MK 1
ROBOTRONIC MK 1
ROBOTRONIC MK 2
ROBOTBONIC MK 3
TRIOBATIC
CYBONIC MK 1
CYBONIC MK 2

Чем выше класс силовой установки, тем более мощный двигатель и оружие в течение более долгого времени способна она поддерживать.

Шасси (CHASSIS)
DURALITE
TRIALIUM MK 1
TRIALIUM MK 2
CHROMITE
PLASTEEL
STRESSED PLASTEEL
CORALLOID MK 1
CORALLOID MK 2

Чем совершеннее шасси, тем легче Ваш робот перенесет падение с высоты.

Другие устройства (DEVICES).

Эти устройства являются дополнительными и потому они есть только у роботов высшего и среднего классов, у низших роботов они отсутствуют.

DISRUPTOR SHIELD - защита от воздействия DISRUPTOR-а. Она позволяет продержаться более долгое время при действии мощного оружия. Кстати, роботов, имеющих эту защиту, легче победить в борьбе за перехват управления, чем уничтожить оружием.

LASER SHIELD - защита от действия лазерного оружия, кстати неплохо защищает и от попыток пробить Ваш робот с помощью лобового тарана.

RAM THRUSTER - ускоритель. Позволяет увеличить скорость движения Вашего робота, способствует успеху тарана.

POWER BOOTS - обеспечивает уменьшение нагрузки на шасси за счет дополнительной тяги, что позволяет более стойко переносить падения с высоты.

DETECTOR - устройство, предназначенное для контроля за уровнем радиации. Сигнализирует о наличии опасного излучения.

OVERDRIVE - устройство для форсирования тяги двигательной установки.

Для тех, кто интересуется тем, какие устройства установлены на роботах, приводим список в таблице 1. Список не полный в связи с тем, что пока еще не все обнаружено. Желаем удачи в игре.

обозначение	класс	двигатель	силовая установка	оружие	шасси	дополнительное устройство
AB	альфа	ULTRAGRAV	CYBONIC mk 2	AUTOCANNON	CORRLOID mk 2	RAM TRUSTER
ST	бета	DUAL LINEAR	CYBONIC mk 1	DESINTEGRATOR	CORRLOID mk 1	DISRUPTOR SHIELD
		AUTOMATION				
A1	альфа	ULTRAGRAV	CYBONIC mk 2	AUTOCANNON	CORRLOID mk 2	LASER SHIELD
		BATTLE				
B2	бета	HEAVY DUTY	TRIOBATIC	DISRUPTOR	CORRLOID mk 1	DISRUPTOR SHIELD
B3	гамма	GRAVITRONIC mk 2	TRIOBATIC	DISRUPTOR	STR.PLASTEEL	DISRUPTOR SHIELD
B4	гамма	GRAVITRONIC mk 2	ROBOTRONIC mk 3	DISRUPTOR	STR.PLASTEEL	LASER SHIELD
B5	дельта	GRAVITRONIC mk 1	ROBOTRONIC mk 2	AUTOCANNON	STR.PLASTEEL	RAM TRUSTER
B6	дельта	GRAVITRONIC mk 1	ROBOTRONIC mk 2	DESINTEGRATOR	CORRLOID mk 2	LASER SHIELD
B7	дельта	GRAVITRONIC mk 1	ROBOTRONIC mk 2	DESINTEGRATOR	CORRLOID mk 1	RAM TRUSTER
		LOGIC				
L3	гамма	GRAVITRONIC mk 2	ROBOTRONIC mk 2	DUAL LASER	PLASTEEL	POWER BOOTS
L4	гамма	GRAVITRONIC mk 2	ROBOTRONIC mk 2	PULSE LASER	PLASTEEL	
L5	гамма	GRAVITRONIC mk 2	ROBOTRONIC mk 1	PULSE LASER	PLASTEEL	
L6	гамма	GRAVITRONIC mk 2	ROBOTRONIC mk 1	PULSE LASER	PLASTEEL	
		MEDIC				
00	гамма	GRAVITRONIC mk 2	ROBOTRONIC mk 1	PULSE LASER	PLASTEEL	POWER BOOTS
		REPAIR				
R5	дельта	LINEAR mk 3	CHEMIFAX mk 2	DUAL LASER	PLASTEEL	DISRUPTOR SHIELD
R6	дельта	LINEAR mk 3	CHEMIFAX mk 2	DUAL LASER	PLASTEEL	
R7	дельта	LINEAR mk 3	CHEMIFAX mk 2	DUAL LASER	TRIALIUM mk 2	
R8	дельта	LINEAR mk 3	CHEMIFAX mk 2	PULSE LASER	TRIALIUM mk 2	
		COMMAND				
C1	альфа	ULTRAGRAV	CYBONIC mk 1	DESINTEGRATOR	CORRALOID mk 1	POWER BOOTS
C2	альфа	DUAL LINEAR	CYBONIC mk 1	DESINTEGRATOR	CORRALOID mk 1	DETECTOR
C3	бета	DUAL LINEAR	CYBONIC mk 1	DESINTEGRATOR	CORRALOID mk 1	OVERDRIVE
C4	бета	DUAL LINEAR	CYBONIC mk 1	DESINTEGRATOR	STR.PLASTEEL	LASER SHIELD
C5	бета	HEAVY DUTY	TRIOBATIC	DESINTEGRATOR	STR.PLASTEEL	DISTRUPTOR
		SECURITY				
S2	бета	HEAVY DUTY	TRIOBATIC	AUTOCANNON	STR.PLASTEEL	DETECTOR
S3	бета	DUAL LINEAR	TRIOBATIC	AUTOCANNON	STR.PLASTEEL	DETECTOR
S4	бета	HEAVY DUTY	TRIOBATIC	DESINTEGRATOR	STR.PLASTEEL	DETECTOR
S5	бета	HEAVY DUTY	ROBOTRONIC mk 3	AUTOCANNON	STR.PLASTEEL	DETECTOR
S6	бета	GRAVITRONIC mk 2	TRIOBATIC	AUTOCANNON	STR.PLASTEEL	DETECTOR
		UTILITY				
U7	епсилон	LINEAR mk 2	CHEMIFAX mk 1	PULSE LASER	TRIALIUM mk 2	
U8	епсилон	LINEAR mk 1	CHEMIFAX mk 1	PULSE LASER	TRIALIUM mk 2	
U9	епсилон	LINEAR mk 1	CHEMIFAX mk 1		TRIALIUM mk 1	
		MENTAL				
X8	епсилон	GRAVITRONIC mk 2	CHEMIFAX mk 1		TRIALIUM mk 1	
X9	епсилон	LINEAR mk 1	CHEMIFAX mk 2		DURALITE	

CAPTAIN FIZZ

"Psyclapse" 1989 г.
Эксперт Троекуров В.И.
г. Киев

Программа "CAPTAIN FIZZ" принадлежит к аркадному жанру. Действие происходит на космическом корабле "ИКАРУС", который потерял управление и движется к Солнцу. Если он подойдет к нему слишком близко, то взорвется, разрывая галактику. Чтобы это предотвратить, необходимо телепортироваться на борт звездолета и разрушить компьютеры, управляющие его движением.

Палубы корабля представляют собой достаточно сложную систему лабиринтов и Ваша главная цель разобраться в этой системе, исследовать корабль и восстановить его управление.

Характерной особенностью игры является то, что она позволяет работать над главной задачей "кооперативно". Два игрока могут играть одновременно и помогать друг другу своими действиями.

Экран программы.

Экран разделен для двух игроков по горизонтали. Панель каждого игрока состоит из трех частей:

1. Информационное табло. Сюда выводится текущий результат игры (очки), шкала, показывающая температуру Вашего оружия (когда шкала исчезла от продолжительной стрельбы, необходимо отпустить кнопку и подождать, пока она восстановится).

ARMOUR - Ваша защита.

DAMAGE - Величина повреждений.

CHARGE - Заряд.

CREDIT - Количество "попыток".

2. Вторая часть - основной экран, на котором происходит действие игры. У каждого из игроков свой отдельный экран, что способствует независимости одного игрока от другого. Сверху над экраном расположены два слова:

HEALTH - состояние Вашего здоровья в 9999 энергетических единицах.

CARDS - карточки (пропуска), количество пропусков, их цвета (для каждого пропуска своя дверь, которую он может открыть).

3. Третья часть - индикатор состояния. Ломаная линия на нем - Ваша кардиограмма. По мере ухудшения Вашего самочувствия линия выпрямляется.

Здесь же представлены три дополнительных индикатора.

а) EXIT - "выход". Когда Вы ликвидировали все компьютеры на определенном уровне, то увидите, что загорится этот индикатор - значит можно телепортироваться на другой уровень. Открытый выход имеет овальную форму черного цвета с желтой окантовкой по краям. Закрытый выход имеет ту же форму но сверху на нем решетка желтого цвета.

б) SWITCHES - "переключатели". Здесь расположены четыре индикатора. Когда индикаторы горят, силовое поле отключено.

в) MINES - "мины". Индикатор предупредит Вас, что в помещении имеется мина и

лучше здесь не стрелять, т.к. при попадании в мину Вы потеряете при ее взрыве определенную часть энергии.

Программа имеет еще одну интересную особенность. После загрузки ее можно настроить на работу с одним из трех европейских языков:

1. Английский.
2. Немецкий.
3. Французский.

После ввода программы, нажмите клавишу "5" на клавиатуре (2 игрок) и "FIRE" на джойстике (1 игрок) и начинайте игру.

Итак, начинаем игру вдвоем. На экранах появятся два героя - один желтого цвета, другой - белого. Первый управляется от клавиатуры клавишами:

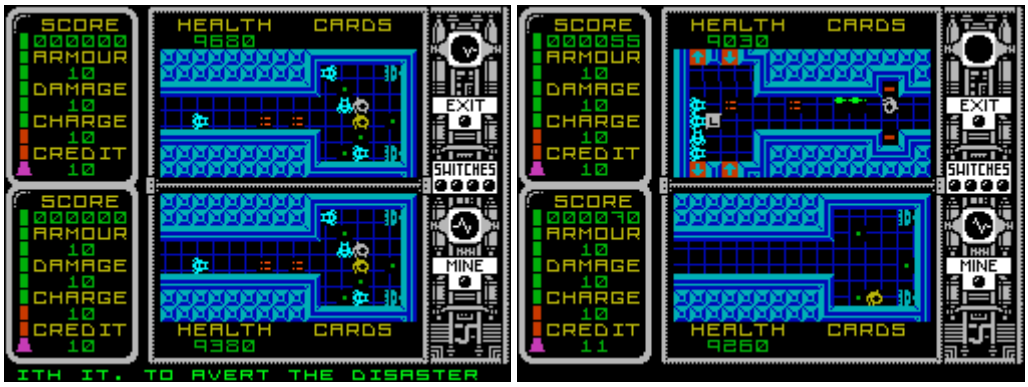
1 – влево 2 – вправо
3 – вниз 4 - вверх
5 - "огонь"

Второй герой может управляться как от джойстика, так и от клавиатуры:

6 – влево 7 – вправо
8 – вниз 9 - вверх
0 - "огонь"

Первое, что Вы должны сделать - это расстрелять компьютеры (они выглядят, как белые квадратики, маркированные буквой "L").

Сложность заключается в том, что их охраняют боевые роботы, с которыми предстоит провести перестрелку. Кроме того, есть небольшое силовое поле, способное какое-то время защищать компьютеры от поражения Вашим оружием.



Оба космонавта имеют по мощному бластеру с неограниченным боекомплектом. Но Вам следует внимательно следить за температурой своего оружия, так как оно не допускает перегрева и перестает стрелять. Кроме бластеров у космонавтов еще есть и супербомба, способная уничтожать враждебную боевую технику в пределах данного экрана, но не повреждающая компьютеры.

В исходном состоянии Ваша сила составляет 9999 энергетических единиц. Но здоровье можно поправить, когда Вы соберете энергетические аптечки (выглядят, как зеленые точки), но использовать их для своих целей можно только лишь подойдя к энергетической емкости. Чтобы понять, как выглядят энергетические емкости, надо обратить внимание на первую картинку игры. Рядом с двумя героями изображены зеленые точки (энергетические аптечки), а справа энергетические емкости.

Возьмите аптечку и подойдите к энергетической емкости. Вы увидите, что Ваша энергия возрастет до 9999 единиц - это предел и не тратьте лишние аптечки зря.

Карточки-пропуска понадобятся Вам, чтобы пройти в другие отсеки корабля. Пропуска выглядят как маленькие цветные прямоугольники. Они бывают 5 разных цветов: красного, желтого, синего, пурпурного и голубого. На дверях указан цвет пропуска, которым они открываются. Если Ваш пропуск не соответствует цвету на двери, то Вы не сможете ее открыть.

Есть двери, вырабатывающие защитное силовое поле, контактов с которым

необходимо избегать. В случае Вашей гибели задача партнера очень усложнится. Впрочем, если ему удастся пройти на следующий уровень, Вы появитесь снова, хотя уровень Вашей энергии при этом будет равен только 2000 единиц.

Гибель обоих игроков ведет к прекращению игры. После этого программа предложит Вам продолжить игру (CONTINUE). Утвердительный ответ (Y) позволяет вернуться на один-два уровня назад и продолжить игру. Отрицательный ответ (N) позволяет начать игру сначала.

Выключить силовое поле двери можно переключением четырех тумблеров. Их надо отыскать на данном уровне и определить правильную последовательность их включения. При правильном их положении под индикатором SWITCHES должны загореться четыре зеленых лампочки - поле отключено и можно идти. Тем не менее будьте осторожны - некоторые поля отключаются лишь на несколько секунд, что достаточно для входа, но для выхода придется переключать тумблеры еще раз.

Силовые поля, боевые роботы, мины - это еще не все неприятности, подстерегающие Вас. Наиболее серьезным препятствием является боевая механизированная установка, стреляющая самонаводящимися ракетами. Поразить ее без бомбы невозможно, но можно выстрелом из бластера изменить направление полета ракеты. Ракеты, не найдя нужной цели, самоликвидируются и Вы можете постараться как-то использовать это в своих целях.

На многих уровнях Вы встретите в красных квадратиках стрелки, показывающие определенное направление. Это транспортеры. Если Вы пойдете на стрелку, то пройдете сквозь нее, но назад здесь же вернуться не сможете, так что будьте внимательны.

Полезные советы.

1. Не играйте с партнером в перестрелку. Во-первых, это непроизводительные потери личной энергии, а во-вторых, при столкновении Ваших снарядов происходит блокировка этого места и Вы не сможете его пройти.

2. Как пройти вдвоем через дверь, если пропуск есть только у одного героя?

Делайте так: Герой, у которого есть пропуск, проходит через эту дверь и поджидает пока партнер не окажется напротив двери. Затем он подходит к двери, она открывается и пропускает партнера.

3. Главное правило: работу в отсеках космонавты выполняют по одиночке, а имущество собирают поровну. Если в комнате находятся два пропуска, то Вам надо взять один, а второй на всякий случай оставить партнеру. Если же пропуск всего один, значит кто-то из Вас должен его использовать, в то время, как второй космонавт должен находиться снаружи и страховать напарника, чтобы быть готовым в любой момент прийти к нему на помощь.

FORUM

Нам кажется, что программа ELITE столь же неисчерпаема, как и сам "ZX-Spectrum". Вот и сегодня пришло письмо, которое, возможно, откроет новые перспективы для тех, кто уже несколько раз прошел программу от начала до конца и ищет, чем бы теперь заняться.

Наш читатель из Киева Руслан Хоминич провел исследование трех разных версии игры, нашел в них существенные отличия и, если Вы хорошо освоили одну из версий, Вам может быть интересно попробовать свои силы и в другой.

Версия 1.

Программа вскрыта командой "JOYSTICK CLUB". После загрузки появляется надпись "Press SPACE Commander" и изображается корабль "ASP MK II".

Это самая легкая версия игры. Ваш корабль довольно активно атакуется, независимо от Вашего правового статуса, но погибнуть довольно сложно.

В этой версии сравнительно неэффективны ракеты - они уничтожают не более половины заряда энергетического отсека и, наверное, поэтому многие пилоты их не любят, считая дорогим и малоэффективным оружием.

В этой версии можно даже брать на таран корабли противника, причем статус FUGITIVE Вы за это не получите, даже если протараните мирный корабль.

Не встречается корабль ADDER.

Очень велики расстояния между планетами, топлива едва хватает для перелетов внутри круга, а на периферии круга расстояние уже слишком велико для перелета в один ход. Это же мешает использовать в полной мере клавишу F для перехвата таргоидов в гиперпространство, т.к. после возврата обычно не хватает топлива на обратный путь.

Атака на станцию не дает видимых результатов - станция просто не принимает Вас без каких либо объяснения. Вблизи станции корабли не встречаются.

Эту версию стоит рекомендовать начинающим пилотам.

Меняйло Е.В. из Калуги уточняет, что в этой версии самое короткое расстояние между соседними планетами составляет 3,2 св. года, подтверждает отсутствие корабля ADDER и добавляет, что никак не проходит заправка топливом от звезды.

Версия 2.

Программа вскрыта командой "TENARK SOFTWARE".

Главное отличие - отсутствуют корабли KRAIT. На заставочной картинке представлен корабль ADDER. В связи с этими обстоятельствами, в программе изменилась тактика пиратов.

В целом программа представляет средний уровень сложности.

Как и в предыдущей версии здесь ракеты малоэффективны. При попытке атаковать станцию, Вам выдают угрожающую надпись и Вы лишаетесь возможности стыковки. Станция оснащена системой ECM.

Эта версия позволяет стать миллионером в галактике 47 с помощью SAVE.

В обеих версиях имеют место захваты корабля пиратами при стыковке со станцией. Есть гипотеза, что Ваш корабль захватывается пиратами как бы "изнутри" в результате того, что Вы подбирали в космосе контейнеры с "рабами" и перегрузили свой корабль. Во всяком случае, не отмечалось захватов корабля, если в трюме не было "рабов".

Эта гипотеза имеет и то косвенное подтверждение, что были неоднократно отмечены захваты корабля в космическом пространстве "изнутри" после выхода из гиперперехода (об этом пишет, например, К. Дзреев из Ростова на Дону).

Версия 3

Версия имеет сообщение "M128".

Это наиболее сильная версия, она также наиболее соответствует фирменной инструкции.

Например, при попытке атаковать станцию, Вам навстречу вылетает стая полицейских кораблей и, хотя они не очень хорошо вооружены, их слишком много, чтобы устоять. Не спасает и перелет на другую планету, отныне в этой галактике Вы - изгнанник.

Значительно усложнена стыковка. Вокруг станции постоянно находятся различные корабли (преимущественно типа PYTHON) и, если Вы замешкаетесь, то рискуете с кем-нибудь столкнуться перед самой стыковкой. Часто это приводит к гибели.

Не менее опасно и резко тормозить после вылета со станции.

В этой версии большой разброс цен на некоторые запрещенные к перевозке товары, то есть Вас стимулируют к рискованным операциям. За Вами внимательно наблюдают. Если в глубинах космоса Вы совершите преступление, Вас могут не пустить потом на станцию. Здесь очень неприятно получать статус FUGITIVE. Не открывайте огонь первым, пока не увидите модель корабля визуально или пока он сам не начнет стрельбу.

Корабли типа KRAIT, THARGOID и SIDEWINDER здесь всегда пираты (возможно и ADDER, но пока он не встречался).

С кораблями типа PYTHON надо вести себя аккуратно. После первых выстрелов он выпускает KRAIT и SIDEWINDER - с ними можете воевать, это Ваша добыча, но головной корабль лучше оставить в покое и побыстрее покинуть поле боя.

FER-DE-LANCE - желательная добыча. Денег за него не дают, но рейтинг хорошо нарастает.

Нередко попадаются отшельники на астероидах. После обстрела они либо уходят, слабо маневрируя, либо отстреливаются, либо выпускают корабли KRAIT или SIDEWINDER. Из них можно "выжать" и контейнер с грузом, но поскольку отшельники защищены всегалактическим правом, Ваш статус может резко измениться.

В этой версии программы весьма эффективны ракеты. Попадание почти полностью выбивает защиту. Первым делом надо покупать систему ECM. Точно так же опасен и таран.

Сложность игры нарастает по мере повышения Вашего рейтинга. Сражения в галактике 1 намного проще, чем во второй галактике. Если в первой галактике Таргоиды поражались даже пульсирующим лазером, то во второй справиться с ними удалось далеко не сразу. Приходится иногда перестраивать тактику боя.

Фокус с 47-ой галактикой здесь не проходит.

Интересно, что отгрузочный блок имеет длину 104 байта. Видимо, это сделано для того, чтобы нельзя было использовать отгрузки с более легких версии.

Высокие качества этой версии подчеркивает также Сергей Дегтярев из Луганска. К сказанному выше мы можем добавить из его бортжурнала то, что при слишком высоком статусе FUGITIVE на станциях не открывается входное отверстие.

Кроме этих версии есть и другие, например версия, помеченная Jack O'Lantern. В ней стоит отметить отсутствие корабля KRAIT (сообщает Меняйло Е.В.). Условно назовем ее ВЕРСИЯ 4.

Лешинский А.М. (Н.Новгород) предполагает, что всего разных версии может быть порядка десятка, очень советует ввести какую-то систематизацию этих версий и упоминает о версии Родионова (назовем ее ВЕРСИЯ 5) и о версии Be-Be Soft (пароль 7Q) - назовем ее условно ВЕРСИЯ 6. К сожалению, он не приводит характерных особенностей данных версий.

"ИНФОРКОМ" горячо поддерживает предложение систематизировать версии и просит присылать свои предложения и идеи. В то же время, необходим какой-то универсальный параметр, который мог бы быть принят за основу классификации. Все читатели в письмах указывают на сообщения, сопровождающие или заканчивающие загрузку, но может быть целесообразно для каждой версии давать и раскладку по длинам входящих файлов?

* * *

У нас были вопросы от желающих получить дисковую версию программы ELITE - для них мы даем следующее сообщение:

Вышлю всем желающим дисковую версию ELITE ("Joystick Club") с записью состояния игры на диск.

169740, Коми, Усинский р-н, п. Приполярный, а/я 212, Сайфутдинов Е.В.

Сагдеев Р.Р. из г. Магнитогорска тоже работал с дисковой версией, вскрытой JOYSTICK CLUB, но в последнее время перешел на версию, в которой во время загрузки появляется надпись "M1 LOADING". Эта версия выглядит намного интереснее, отличается тем, что в ней есть очень сильные пираты и во много раз более серьезная полиция. Может быть, это что-то похожее на ВЕРСИЮ 3 по классификации Хоминича.

Перебросить ее на диск по блокам удается несложно с помощью копировщика PCOPIER, AMCOPY или каким-либо другим способом, загрузчик может быть таким:

```
10 BORDER NOT PI: PAPER NOT PI: CLS: CLEAR VAL "24751": RANDOMIZE USR VAL "15619": REM:
      LOAD "elite 1" CODE
20 RANDOMIZE USR VAL "15619": REM: LOAD "elite 2" CODE
30 RANDOMIZE USR VAL "24792": RANDOMIZE USR VAL "15619": REM: LOAD "elite 3" CODE VAL
      "16464"
40 REM POKE 46848,201
50 RANDOMIZE USR VAL "24795"
```

К сожалению, этого недостаточно, чтобы и отгрузка отложенного состояния игры и подгрузка тоже производились с дисковой поддержкой. Сагдееву Р.В. мешает это сделать отсутствие информации по управлению дисководом из машинного кода. Кстати, наш корреспондент отмечает, что адреса USR для старта блоков отличаются на 3. Это характерно для программ, сопровождающихся при загрузке надписью "M1 LOADING".

Перелеты к двойным звездам и невидимым звездам.

Мы уже писали о том, что в некоторых галактиках были обнаружены двойные звезды. Как уже было установлено, невидимые звезды - это тоже двойные звезды. К сожалению, не ко всем из них есть возможность слетать по желанию.

Меняйло Е.В. разработал схему перелета, основанную на безтопливном перелете со станции на станцию, о чем мы уже писали. (См. N11-12 "ZX-РЕВЮ-91", с. 253). В системе из пары звезд он называет одну основной, а вторую - подчиненной и предлагает следующий порядок действий.

1. Перелетите на основную планету любым способом.

2. Вызовите карту и с помощью поиска планеты по ее названию (клавиша "R") установите курсор на подчиненную планету. При помощи клавиши "P" это делать нельзя, т.к. информация имеется только об основной планете и при этом курсор будет всегда устанавливаться на основную планету.

3. Не нажимая клавишу "F" вылетите со станции и при помощи безтопливного перелета переместитесь на подчиненную планету. Оказавшись на ней, Вы можете просмотреть сводку цен, купить или продать товар, но при запросе информации Вам будут выданы данные только об основной планете.

Успешно посетил двойные звезды и наш читатель Владимир Кладов из Новосибирска. Он называет безтопливный перелет "перелетом-180", но пользоваться им не стал, а ввел в программу пару POKES.

POKE 60896,233

POKE 56272,233

Первый отключает поиск ближайшей планеты по команде "F", второй - по команде "H". Выдается информация (выполняется перелет) на планету, найденную предыдущей командой "O","R". Кстати, он приводит еще пару POKES.

POKE 56260,0 - "вечный межгалактический гипердвигатель".

POKE 28822,0 - "вечная энергетическая бомба".

Тайные возможности компьютера.

В эти апрельские дни, кажется решилась одна из важнейших проблем многочисленных любителей "Спектрума". Теперь каждый из Вас может, если захочет, в несколько минут конвертировать свой 48-ми килобайтный компьютер в полноценную 128-

килобайтную машину, причем сделает это абсолютно бесплатно.

Но, прежде чем мы перейдем к практике, несколько слов о сути.

Вы, конечно знаете, что Ваш Синклер-совместимый компьютер имеет 16 килобайтов памяти в ПЗУ и еще 48 килобайтов в ОЗУ, то есть всего 64K, а точнее 65535 байтов. Может он иметь больше? Наверное да, но ведь процессор Z-80 не всемогущ. Он работает с 16-разрядной адресной шиной и потому может обслуживать только 65535 байтов одновременно и не больше. Мы об этом писали в 1991 году на страницах "РЕВЮ", когда говорили о 128-килобайтных машинах. В них дополнительные 16-килобайтные блоки (называемые страницами) "впечатываются" на место страниц обычной памяти, подменяя их на время. Так что и в нем процессор работает всегда только с 64 килобайтами памяти.

А теперь рассмотрим такой вопрос. Вы, очевидно знаете, что Ваш компьютер в состоянии обслуживать еще и 65535 адресов внешних портов. Если Вы не работаете ни с какой периферией, то эта возможность просто никак не используется. А какая с точки зрения процессора разница - обслуживать внешние адреса или внутренние? Никакой. Вот если бы удалось так переключить процессор, чтобы он работал одновременно и с этими дополнительными адресами, как с оперативной памятью - Вы бы сразу имели 128 килобайтную машину без необходимости что-то паять и отлаживать.

Теперь мы Вас порадуем - такая возможность есть. Открыты два нигде не документированных регистра компьютера - раскрыта самая глубокая тайна К.Синклера, который уже в первых своих моделях предусмотрел их последующее расширение до 128K. Впрочем, нас интересует только первый регистр. Этот регистр называется первым Альтернативным Программным Регистром (сокращенно АПР.1). И за этим названием скрыта суть. Пожалуйста не путайте с альтернативным набором регистров процессора. Этот регистр к процессору не имеет никакого отношения и конфигурируется программным путем в результате серии сложнейших последовательностей команд IN и OUT.

Чтобы не забивать Вам голову, мы просто привели программу в машинных кодах, которая выполняет всю эту работу сама. К тому же мы еще сами не до конца разобрались, как же это все работает.

Наберите эту программу, запустите и наслаждайтесь грандиозным эффектом. До тех пор, пока компьютер не будет выключен, Вы можете чувствовать себя ничуть не хуже, чем любой владелец дорогой128-килобайтной машины, во всяком случае с этого момента жить Вам будет веселее.

Правда, после запуска этой программной последовательности, Вы не сможете получить того исходного меню, которое есть в стандартных 128-килобайтных "Спектрумах", но с этим ничего не поделаешь - ведь ПЗУ у Вас осталось от 48-килобайтной машины. Хотя, если подойти к задаче творчески, мы уверены, что кому-то из Вас удастся развить идею и получить меню.

Первая часть программы представляет обычный БЕЙСИК-загрузчик, который загружает машинный код, начиная с адреса 32768 и запускает его с адреса 32786. При загрузке проверяется также правильность Вашего набора кодов, которые хранятся в строках DATA.

После запуска Вам придется немного подождать (секунду-другую), пока программа "прощупает" архитектуру Вашей машины и выполнит необходимые операции.

Желаем успеха!

```
10 CLEAR 32767
20 LET x=32768
30 FOR i=0 TO 14
40 LET c=0
50 FOR j=1 TO 6
60 READ a
70 POKE x,a
80 LET x=x+1
90 LET c=c+a
100 NEXT j
110 READ a
120 IF a<>c THEN PRINT "ERROR IN LINE "; 10*i+160: STOP
```

```
130 NEXT i
140 RANDOMIZE USR 32786
150 STOP
160 DATA 22,0,0,17,7,16,62
170 DATA 3,65,80,82,73,76,379
180 DATA 32,49,45,83,84,32,325
190 DATA 62,2,205,1,22,1,293
200 DATA 7,0,17,0,128,205,357
210 DATA 60,32,6,64,197,1,360
220 DATA 11,0,17,7,128,205,368
230 DATA 60,32,193,16,243,17,563[1]
240 DATA 3,7,205,84,31,210,540
250 DATA 123,27,118,118,20,203,609
260 DATA 154,28,203,155,213,122,875
270 DATA 205,155,34,230,248,179,1051
280 DATA 33,0,88,17,1,88,227
290 DATA 1,255,2,119,237,176,790
300 DATA 209,24,219,0,0,0,452
```

Советы и секреты.

В ответ на просьбу о пароле к 8-му уровню игры IMPACT пришло немало писем от читателей. Так, Сельдемишев М.М. из г. Томска называет этот пароль - J.D. но ставит новый вопрос: - время от времени в этой программе в левом верхнем углу появляется буква "B" - что бы это значило? Рожков П.В. из Пензы не только дает этот пароль, но еще сообщает POKE 54500,183. Этот POKE уже есть в загрузчике программы, но стоит после оператора REM и потому не действует. Достаточно стереть REM и запустить программу. С другой стороны, у него есть просьба к обладателям игры TANTALUS (на кассетах ходит под названием TANTAL.KEY) - не может ли кто-нибудь помочь с POKES - игра хорошая, но пройти ее не удается. С помощью нашего трехтомника по программированию в машинных кодах ему удалось самостоятельно отыскать POKES в ряде программ и он рад ими поделиться:

FROST BYTE	33052,0 - жизнь
	33805,52 - время
DAN DARE 2	45891,0 - жизнь
SCEPTRE OF BAGDAD	58116,0 - жизнь
REBEL STAR 2	28599,N - кол-во ходов
PHANTOM CLUB	56486,0 - жизнь
TRUENO 1	25100,N - энергия
TRUENO 2	24785,0 - жизнь
	24984,N - энергия

Серию подсказок, достаточную для того, чтобы пройти всю игру SCEPTRE OF BAGDAD прислал наш читатель из Каракашев А.Г. из г. Грозный. Он очень интересуется адвентюрными играми, но, как и многие, не имеет возможности пополнять коллекцию в связи с тем, что они пока мало распространены.

1. Имея каску, можно пройти в комнату, где должна быть женщина, закрывающая ход в спальне. Теперь каску можно выбросить, больше никогда никто здесь не появится.

2. Флейта нужна, чтобы моток веревки при Вашем появлении в комнате поднимался вверх - тогда по ней можно подняться.

3. Имея крылья, идите к фонтану, ангел улетит на небеса, откроется подземный ход.

4. Огненного круг откроет люк (в фонтане).

5. Со связкой ключей идите в комнаты, расположенные за проходом, который раньше закрывала малоприятная личность, боящаяся каски. Дойдя до конца, прыгните в шкаф, ключи откроют потайной ход.

6. В фонтане трезубец можно обменять на жемчужину.

7. Имея сачок, идите в комнату с пчелой (пчелу предварительно надо выпустить из

[1] Скорее всего 561 (Прим. NUK)

гнезда). Поймайте пчелу и отнести ее в комнату с пауком. Паук исчезнет.

8. Шпагой можно отрубить веревку, на которой висел паук.

9. Жемчужиной, взятой в фонтане, зарядите рогатку.

10. Заряженной рогаткой сбейте кокосовый орех с пальмы, где сидит обезьяна.

11. Имея кокос можно идти в жаркую пустыню.

12. С книгой Али-Бабы можно открыть потайной код в подвале.

13. После этого можно в подвале наполнять кошелек золотом.

14. Имея кошелек, можно зайти в магазин и купить топор и башмаки.

15. Хлыстом усмирите быка и обменяйте его на ось.

16. Ось замените в подвале на сломанную.

17. Топором заточите кусок дерева.

18. С помощью зеркала убейте медузу.

19. Заточенным деревом убейте людоеда.

20. Веником усмирите волшебника и возьмите лампу, с которой можно смело идти за скипетром.

21. Одев башмаки, можно пройти через раскаленные угли.

22. Имея рыбу-шлем можно дышать под водой.

23. Веревкой натягивается лук.

24. Заряжается он стрелами.

25. С заряженный луком можно идти в комнату, расположенную за людоедом.

Далее уже совсем легко. Взяв жезл, возвращаетесь на балкон справа от исходной комнаты. Теперь мы видим счастливое лицо нашего героя, сжимающего в руках заветный скипетр.

Четыре предмета: песочные часы, тряпку, доспехи и перо применить нигде не удалось, но видимо они и не нужны, т.к. было набрано 100% очков.

Некоторые полезные советы:

1. Последовательность выполнения действия любая, удобная для игрока, правда, пройдя игру несколько раз, можно попробовать искать все более короткий путь. Конечно, при первом проходе трудно избежать бестолкового шатания по всему лабиринту туда-обратно.

2. Старайтесь никогда не делать себе безвыходных ситуаций, например, можно, забравшись по веревке вверх, умудриться забыть там флейту и спрыгнуть вниз без нее, или, например оставить в магазине мешок с золотом и выйти. Это может свести на нет все Ваши усилия.

3. Найдя бутылку, дающую OLD GAME не спешите тут же хватать ее, если у Вас еще достаточно жизней, а старайтесь сделать как можно больше действий с предметами, и взять бутылку, только когда жизней остается опасно мало.

ВНИМАНИЕ!

Дорогие друзья, сегодня мы можем помочь Вам стать непосредственными участниками новой, готовящейся в настоящее время телевизионной игры. Мы уполномочены на то, чтобы раскрыть перед Вами некоторые детали этого мероприятия и пригласить Вас к участию в небольшом конкурсе.

Ее устроитель, фирма "ФОНД", поставила перед собой интереснейшую задачу объяснить в форме популярной телевизионной игры что такое акции, что такое фондовая биржа, что такое дивиденд и как образуется курсовая разница. Зрители смогут наглядно увидеть и почувствовать, что такое фондовый рынок, узнают многое из того, что поможет им в будущем избежать возможных неприятностей, связанных с вложением денег в ценные бумаги.

"ИНФОРКОМ" участвует в подготовке этой телепрограммы на некоммерческой основе, обеспечивает ее компьютерную поддержку и, до некоторой степени, участвует в выработке ее концепции.

Начало регулярных трансляций телеигры в эфире запланировано на август-сентябрь месяц сего года. Названия телеигры мы Вам пока сообщить не можем, оно проходит регистрацию.

Теперь о сути.

Организация игры достаточно традиционна: пять игроков на сцене и несколько рядов зрителей в студии. Игроки получают начальный капитал (5000 рублей) и могут им распоряжаться, вкладывая деньги в акции тех или иных компаний. Акции можно не только покупать, но продавать если Вы чувствуете, что над компанией нависла угроза.

Всего в игре могут быть задействованы акции следующих двадцати компаний:

- судостроительная;
- лесоперерабатывающая;
- торговый дом;
- биржа;
- пищевая;
- авиационная;
- фармацевтическая;
- транспортная;
- топливно-энергетическая;
- газовая;
- пивоваренная;
- оборонная;
- банк;
- станкостроительная;
- золотодобывающая;
- прохладительных напитков;
- строительная;
- полиграфическая;
- автомобильная;
- химическая.

Финансовое положение этих компаний нестабильно и стоимость их акций непрерывно изменяется. Предугадать, как поведут себя акции той или иной компании можно только если очень внимательно читать газеты и делать при этом правильные выводы из прочитанных сообщений.

Перед каждым ходом игрокам дается какое-то газетное сообщение, которое может быть и совершенно шуточным и абсолютно серьезным, а уж они сами должны сообразить, как оно повлияет на стоимость акций тех или иных компаний.

Давайте рассмотрим, например вот такое сообщение:

"Как сообщают из высокопоставленных кругов, близких к правлению корпорации "Детский Мир", в течении двух-трех ближайших недель готовится десант Санта- Клаусов из штата Техас. В качестве ближней тактической задачи перед десантной группой ставится поздравление воспитанников московских детских домов и школ интернатов. Десант возглавляет президент компании "Шеврон". "

Такое сообщение может иметь, например следующие воздействия на рынок ценных бумаг:

Торговый Дом	+40%
Биржа	+20%
Фармацевтическая	+15%
Топливно-энергетическая	+22%
Газовая	+12%
Пивоваренная	-20%
Банк	+15%
Прохлад. Напитков	-25%
Автомобильная	-11%

Теперь рассмотрим, в чем здесь суть. В этом сообщении две ценных информации.

Во-первых, это привязка по дате. Очевидно, что речь идет о начале декабря месяца и приближается Рождество со всеми вытекающими отсюда последствиями.

Во-вторых (и это более сложно) те, кто внимательно следит за экономическими новостями, знают, что компания "Шеврон" подписала соглашение с Н. Назарбаевым о совместной эксплуатации нефтяных полей в Прикаспии. Эти нефтяные поля являются совместными для Казахстана и России и можно предположить, что гуманитарный визит президента этой компании в Москву может в будущем привести и к заключению аналогичных соглашений и с Россией.

Итак, акции Торгового Дома имеют всплеск, потому что в последний месяц перед Новым Годом совершается покупок больше, чем в последующие три месяца. Все хотят поздравить родных и близких. Кроме того, население, приученное к тому, что в новом году обязательно будут новые цены, спешит купить все, что запланировано, в последние недели.

Возрастание общего объема продаж влечет и повышение деловой активности на бирже и рост доходов биржи.

Временный подъем может иметь и фармацевтика, ведь в этот осенне-зимний период резко увеличивается количество вирусных заболеваний, а что делается в дни школьных каникул после многочисленных "Новогодних Елок" не надо объяснять.

Газовая промышленность имеет рост, поскольку приближаются зимние холода и требуется повышенный расход газа. То же относится и к топливно-энергетическому комплексу, но добавим еще ранее высказанные предположения о цели приезда президента компании "Шеврон".

Общий рост деловой активности несколько поднимает и акции банков.

Спад имеют автомобильная промышленность (спрос на автомобили в начале зимы традиционно меньше, чем в начале лета) и, конечно, сокращается потребление пива и прохладительных напитков.

* * *

Если хотите, можете попробовать свои силы в придумывании подобных сообщений. Присылайте то, что Вам удастся изобрести вместе со своими предположениями о реакции рынка на это сообщение.

Фирма "ФОНД" будет ежемесячно отбирать 7-8 лучших сообщений для включения их в последующие игры, а авторы этих сообщений получат приглашения в студию в качестве

зрителей. Иногородним фирма "ФОНД" обеспечит проживание и компенсацию затрат на проезд.

Кроме этого, среди приглашенных будет организован блиц-турнир. Его победитель получит право занять пятое игровое место на сцене и безвозмездно получит исходную сумму в 5000 рублей, необходимую для участия в игре. В ходе игры у него будет возможность ее увеличить или уменьшить.

После выхода в телеэфир презентационной передачи (август сентябрь) к участию в этом кон курсе подключатся миллионы телезрителей, но пока вы, наши уважаемые читатели, имеете неоспоримое преимущество.

Желаем Вам успеха!

Свои варианты присылайте пока на наш адрес, мы передадим их устроителям телепередачи - фирме "ФОНД". На конвертах делайте пометку "TV".

Уточняем наш адрес. 121019, Москва. Г-19, а/я 16.

Вниманию наших коммерческих представителей на местах, а также всем желающим подключиться к распространению программных продуктов "ИНФОРКОМа" для IBM-совместимых компьютеров мы представляем обучающий программный комплекс "БАРЬЕР".

I. НАЗНАЧЕНИЕ КОМПЛЕКСА

Комплекс предназначен для развития навыков скорочтения и, тем самым обеспечивает повышение интенсивности усвоения любых текстовых учебных материалов. Комплекс состоит из трех программ и имеет широкий спектр модификаций. Разработанный в период октябрь 1990 - март 1991г. , он реализуется нами на коммерческой основе уже более года. Количество продаж на сегодняшний день приближается к одной тысяче. В порядке подведения первых итогов скажем, что комплекс собирает самые благоприятные отзывы. Наиболее мощный эффект он имеет для детей любого возраста (умеющих читать).

Программы сами настраиваются на уровень подготовки пользователя, который с ними работает, не требуют никаких навыков по работе с ЭВМ, имеют ярко выраженный соревновательный характер и могут рассматриваться не только как тренирующие, но и как тестирующие и как игровые.

БАРЬЕР-1. Программа предназначена для развития навыков "фотографического" восприятия текстов и интенсивно тренирует фотографическую память.

БАРЬЕР-2. Программа предназначена для развития периферического зрения. С помощью тестовых таблиц производится диагностирование распределение концентрации внимания по зрительному полю.

БАРЬЕР-3. Тренирует блочное восприятие текстов при неполной информации (когда левая и правая часть поля страницы не охватываются периферическим зрением).

II. КОМПЛЕКТ ПОСТАВКИ

Комплекс поставляется на трех дискетах 5,25" (MS DOS, 360 K). Каждая программа на отдельной дискете.

III. МОДИФИКАЦИИ КОМПЛЕКСА

Программы комплекса "БАРЬЕР" дают проверенный эффект независимо от того, кто с ними работает, но тем не менее, мы подготовили ряд параллельных модификаций, рассчитанных на то, чтобы учебно-тренировочные тексты, используемые в программах, были по-возможности максимально приближены к конкретной сфере деятельности нашего заказчика.

Это следующие модификации:

БАРЬЕР-М - менеджмент и маркетинг

БАРЬЕР-В - вычислительная техника и программирование.

БАРЬЕР-Д - для детей (младший школьный возраст). Эта версия адаптирована для детей в смысле особого подхода к подбору учебных текстов. Как показывает опыт эксплуатации, одновременно с значительным (в 2-3 раза за один месяц) возрастанием скорости чтения у детей развивается память и правописание. Эта версия пользуется наиболее широкой популярностью.

БАРЬЕР-П - автоматизация производства и проектирования в машиностроении.

БАРЬЕР-Т - металлообработка (технология машиностроения).

БАРЬЕР-К - металлообработка (станки и инструмент).

БАРЬЕР-Р - радиотехника и электроника

БАРЬЕР-С - сельское хозяйство (растениеводство).

БАРЬЕР-О - общекультурное содержание. Эту версию заказывают при неопределенной профессиональной ориентации, например для школьников старших классов или для тех, кто связан с гуманитарными областями деятельности. Может быть рекомендована клубам, кружкам и т. п.

ПРИМЕЧАНИЕ: Если в заказе не указано конкретно, какая версия необходима заказчику, мы поставляем версию БАРЬЕР-В (вычислительная техника и программирование).

IV. СРОК ИСПОЛНЕНИЯ ЗАКАЗА.

Срок исполнения - 2 - 3 недели после поступления средств на наш р/с.

V. ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ К АППАРАТНО-ПРОГРАММНОМУ ОКРУЖЕНИЮ

1. Полная аппаратно-программная совместимость с IBM PC XT/AT. Надежность функционирования на отечественных модификациях не гарантируется и не обсуждается.
2. Наличие "жесткого" диска ("Винчестера") стандартного объема.
3. Дисковод гибких дисков 5,25".
4. Операционная система - MS DOS не ниже 3.20.
5. Русификация компьютера в стандарте гост (кодировка альтернативная).
6. Требования к монитору - не специфицируются. Желательно - EGA.

VI. ГАРАНТИЙНЫЕ ОБЯЗАТЕЛЬСТВА

Гарантийным свидетельством при поставке программного продукта является картонный альбом, в который вложены дискеты с указанной на нем датой продажи При его отсутствии поставка выполняется с заверенным гарантийным талоном.

Гарантиями обеспечивается:

- бесплатная замена поставочных дисков, неработоспособных в состоянии поставки (в течение меся после поставки);
- замена с минимальной оплатой при выходе программы из строя по вине пользователя (механическое или электромагнитное повреждение, поражение вирусом на машине пользователя и т. п.) или по истечении месяца после поставки. Минимальная оплата не превышает стоимость дисков + 5% текущей стоимости программного обеспечения + стоимость почтово-транспортных расходов и согласовывается с потребителем.

VIII. ПОРЯДОК ОФОРМЛЕНИЯ ЗАКАЗА.

а) направить в наш адрес письмо-заказ с указанием необходимого программного продукта и количества копий. Приложить копию платежного поручения.

Крайне желательно сообщать также номер банковского АВИЗО. Это позволяет проводить розыск перечисленных средств в случае их длительной задержи в структурных подразделениях центрального Банка.

Наш адрес: 121019, Москва, Г-19, а/я 16, "ИНФОРКОМ"

б) произвести предварительную оплату платежным поручением на наш р/с:
N 500461778 во Фрунзенском коммерческом банке г. Москвы. МФО 201412.

Стоимость комплекса на период май - август 1992г.

БАРЬЕР-М - 1100 руб. + 28%
БАРЬЕР-В - 1100 руб. + 28%
БАРЬЕР-Д - 1100 руб. + 28%
БАРЬЕР-П - 1100 руб. + 28%
БАРЬЕР-О - 1100 руб. + 28%
БАРЬЕР-Т - 1300 руб. + 28%
БАРЬЕР-И - 1300 руб. + 28%
БАРЬЕР-Р - 1300 руб. + 28%
БАРЬЕР-С - 1300 руб. + 28%

"ЗЕЛЕНЫЙ ПАКЕТ" ДИСТРИБУТОРА

О том, что такое "зеленый пакет" Вы можете подробно прочесть в N11-12 "ZX-РЕВЮ" за 1991 г.

Стоимость "зеленого пакета" по комплексу "БАРЬЕР" составляет для частных лиц 430 рублей. Высылается наложенным платежом. Никакой предоплаты делать не надо, достаточно прислать заявку.

Напоминаем, что затраты дистрибутора на *зеленый пакет" являются только залогом и возвращаются по требованию. Активно работающим дистрибуторам затраты возвращаются при выплате комиссионных и далее такие пакеты предоставляются бесплатно.

Содержание