ZX - РЕВЮ
N9,N10 1992

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

СПЕКТРУМ В ШКОЛЕ

Сегодня мы предлагаем Вашему вниманию несложную экзаменующую программу, которая может быть использована для проверки элементарных знаний учащихся по любому предмету. Она пригодится и тем, у кого есть маленькие дети. Опыт показал, что при всей ее простоте она привлекает к себе детское внимание. Программа воспринимается ими, как игра "Викторина". Может так случиться, что именно с этой программы Ваш ребенок начнет свой путь в большую компьютеризацию.

Программа управляется от несложной системы меню и не требует никаких инструкций по работе - она самообеспечена. В ней есть два основных режима работы:

1. Режим тестирования.
2. Режим заполнения вопросов и ответов.

После первого запуска командой RUN программа еще не содержит ни вопросов, ни ответов. Сначала вы должны их ввести. Вы можете иметь 5 серий (вариантов) вопросов и ответов на разные темы. Каждая серия содержит 8 вопросов и ответов.

Выбрав, какой вариант Вы хотите заполнить, вводите вопросы и ответы по указанию от компьютера. Закончив заполнение варианта, проверьте его на отсутствие ошибок. Если не все в порядке, Вам надо будет снова войти в этот режим. На этот раз компьютер предложит Вам выбор:

1 - внести изменения
2 - заменить вариант

В режиме тестирования компьютер предложит учащемуся выбрать вариант, с которым он хочет поработать. Разумеется, для выбора будут предложены только те варианты, которые уже заполнены вопросами и ответами.

Учащийся сам может выбрать, в каком порядке он будет отвечать на предложенные вопросы. В случае правильного ответа на экране появляется веселая физиономия и играет веселая музыка. При неправильном ответе и музыка и физиономия становятся печальными.

При вводе программы у Вас будут, конечно, проблемы с вводом символов русского алфавита. Мы не можем сделать это за Вас, поскольку не можем заранее знать Ваших возможностей. Многие работают на компьютерах с русифицированным ПЗУ и имеют команды для переключения с одного шрифта на другой. У некоторых даже есть как бы дополнительный символьный регистр. Если у Вас ничего в этом смысле в компьютере нет, то Вам надо русифицировать компьютер программно. Как это делается, мы уже многократно писали, но должны предостеречь Вас от русификации путем использования символов графики пользователя UDG. Дело в том, что программа уже использует графические символа от "A" до "F" для изображения "физиономий". Эти символы в распечатке программы подчеркнуты и набирать их следует в графическом режиме. Блок данных, задающий "конструкцию" этих символов, расположен в строках 1000. . . 1050.

Может быть, Вы решите отказаться от изображения этих "физиономий" и тогда вернете возможность использования символов UDG для изображения прописных букв русского алфавита. В этом варианте тоже есть своя прелесть, т.к. тогда возможны двуязычные экраны и Вы сможете применить программу для проверки знаний по иностранному языку.

Может быть, Вы захотите организовать несколько банков символов графики пользователя и оперативно переключаться между ними, когда это надо. Как это делается, мы писали в недавно вышедшем первом томе, посвященном графике "Спектрума" - "Элементарная графика" (М:,"ИНФОРКОМ", 1992г., 208 стр.).

Но по всей видимости, Вам лучше русифицировать компьютер полной сменой символьного набора. Из наших последних публикации на эту тему посмотрите пожалуйста статью Алексеева А.Г., посвящённую полной русификации программы "MASTERFILE-09" в "ZX-РЕВЮ-92" на стр. 29-32,71-75. Полезными будут советы этого автора, приведенные и в данном выпуске "ZX-РЕВЮ" в статье "Профессиональный подход".

При вводе и отладке программы обратите особое внимание на следующие обстоятельства:

1. Подчеркнутые символы A...F являются символами графики пользователя UDG и вводиться должны в графическом режиме (курсор G).

2. Только первый запуск программы можно выполнять командой RUN. После того, как вы заполните хотя бы одну серию вопросов и ответов, эту команду уже подавать нельзя, так как по команде RUN обновляется содержимое программных переменных и массивов и Вам придется снова заполнять вопросы и ответы.

При втором и последующем запусках вместо RUN давайте команду GO TO 70.

ВИКТОРИНА

```
10 REM здесь Вы можете разместить
20 REM необходимые Вам
30 REM процедуры, например для
40 REM русификации компьютера.
49 REM
50 DIM h$(5,32): DIM q$(5,8.28): DIM a$(5,8,13): DIM p$(8): DIM t$(5): DIM e$*(1)
60 LET t$="00000"
70 BORDER 7: PAPER 7: INK 0: CLS
80 RESTORE 1000: GO SUB 1000: LET new=0
90 REM *** Начало работы
100 DIM c$(8): CLS
120 FOR y=1 TO 13 STEP 6
130 FOR x=1 TO 29 3TKP 28
140 PRINT AT y,x; INK 4;"AB";AT y+1,x;"CD";AT y+3,x; INK 2; "AB"; AT y+4,x;"EF"
150 NEXT x
160 NEXT y
170 PRINT INK 4;AT 19,1: "AB"; AT 20,1 ;"CD"; AT 19,29; "AB"; AT 20,29;"CD"
180 PRINT AT 8,5; " Выберите режим работы:"
190 PRINT AT 11,4; "1 = Ответы на вопросы"
200 PRINT AT 13,4; "2 - Ввод новых вопросов"
210 PRINT AT 15,4; "3 = Конец работы"
220 PRINT AT 19,5; BRIGHT 1;"Нажмите нужную клавшу"
230 IF INKEY$="1" THEN GO TO 280
240 IF INKEY$="2" THEN GO TO 600
250 IF INKEY$="3" THEN GO TO 900
260 GO TO 220
270 REM
230 REM ** Ответы на вопросы**
281 REM
290 CLS : INPUT "": PRINT AT 0,7; BRIGHT 1; "Подумай и ответь"
300 GO SUB 1300
310 PAUSE 30: PRINT #0;AT 0,0;"Введи номер варианта "
320 GO SUB 1210: LET n=i: IF n>5 THEN GO TO 320
330 IF t$(n)="0" THEN PRINT AT 6,6; FLASH 1; "Этот вариант не готов": INPUT "": PAUSE 150:
      GO TO 90
340 CLS: LET p$=" 12345678"
350 FOR x=1 TO 8
360 LET r=INT (RND*8)+1: IF p$(r)<>" " THEN GO TO 380
370 GO TO 360
380 IF x=r THEN GO TO 360
```

```
390 LET c$(x)=p$(r): LET p$(r)=" ": NEXT x
400 GO SUB 1100: PRINT INK 1;AT 0, 0;h$(n): FOR x=1 TO 8
410 PRINT AT 2*x+2,0;x;AT 2*x+2,2;q$(n,x)(1 TO 14);AT 2*x+3,2;q$(n, x) (15 TO 28);AT
      2*x+2,17; a$(n, VAL c$(x));AT 2*x+2,31;x
420 NEXT X
430 LET p$="12345678": LET t=0
440 INPUT "": PRINT #0;AT 0, 0; "Номер вопроса ?"
450 GO SUB 1210: LET q=i
460 IF p$(q)=" " THEN PRINT #0;AT 0,0; BRIGHT 1;"На этот вопрос Вы уже ответили": BEEP 1,-
      12: PAUSE 50: GO TO 440
470 PRINT BRIGHT 1: AT 2*q+2,2; q$(n,q)(1 TO 14);AT 2*q+3,2;q$(n,q)(15 TO 28)
480 INPUT "": PAUSE 30: PRINT #0; AT 0,0; "Правильный ответ ?"
490 GO SUB 1210: LET a=i
500 PRINT AT 2*a+2,31; FLASH 1;a: PAUSE 50
510 IF VAL c$(a)=q THEN PRINT #0; AT 0.0; "Правильный ответ    " ; INK 4;"AB
      CD": GO TO 530
520 PRINT #0;AT 0,0; "Нет, ответ неверный "; INK 2; "AB                                    EF":
      GO SUB 2200: PRINT AT 2*a+2,31;a: GO TO 480
530 PRINT AT 2*a+2,31;a;AT 2*a+2,17; BRIGHT 1;a$(n,VAL c$(a)):GO SUB 2000
540 PRINT AT 2*q+2,2;q$(n,q)(1 TO 14); AT 2*q+3,2;q$(n,q)(15 TO 28); AT 2*a+2,17; a$(n,VAL
      c$(a))
550 LET p$(q)=" ": LET t = t+1: IF t = 8 THEN GO TO 570
560 GO TO 440
570 INPUT "": PRINT#0;AT 0,0; BRIGHT 1; "Нажми любую клавишу"
580 PAUSE 0
590 GO TO 90
599 REM
600 REM**Ввод новых вопросов**
605 REM
610 CLS : PRINT AT 0,7; BRIGHT 1; "Ввод новых вопросов"
620 PRINT AT 2,0; "Эта программа может содержать   до 5 вариантов вопросов и отве- тов.
      Каждый вариант имеет номер"
625 PRINT AT 5,0; "от 1 до 5. Здесь вы можете       узнать, какой вариант готов или внести
      нужные изменения и дополнения."
630 GO SUB 1300: PRINT AT 19,3; "для выхода нажмите клавишу 6"
640 INPUT "": PRINT #0; AT 0,0;" Какой вариант будем заполнять?"
650 GO SUB 1210: LET n=i: IF n>6 THEN GO TO 650
660 INPUT "": IF n=6 THEN GO TO 90
670 IF t$(n)="0" THEN GO TO 790
680 INPUT "Исправление или замена? Нажмите i или z (прочие клавиши - конец."; LINE e$
690 IF e$ = "z" THEN GO TO 790
700 IF e$ <>"i" THEN GO TO 90
710 CLS : GO SUB 1100: PRINT INK 1;AT 0,0;h$(n)
720 FOR x=1 TO 8: GO SUB 880: GO SUB 890: NEXT x
730 INPUT "Какой вопрос будем исправлять? (Клавиша 9 - конец работы)"; x
740 LET x=INT x: IF x<1 OR x>9 THEN GO TO 730
750 IF x=9 THEN GO TO 90
760 PRINT AT 2*x+2,0; BRIGHT 1;x: GO SUB 860
770 PRINT AT 2*x+2,31; BRIGHT 1;x: GO SUB 870
780 LET new=1: GO TO 730
790 CLS : INPUT "Заголовок (до 32 букв)? ", LINE h$(n)
800 GO SUB 1100: PRINT INK 1;AT 0,0;h$(n)
810 FOR x=1 TO 8: GO SUB 860: GO SUB 870: NEXT x
820 LET t$(n)="1": LET new=1
830 INPUT "": PRINT #0;AT 0,0; BRIGHT 1: "Нажмите любую клавишу"
840 PAUSE 0
850 GO TO 90
860 INPUT "Вопрос (до 28 букв)? ", LINE q$(n,x): GO SUB 880: RETURN
870 INPUT "Ответ (до 13 символов)? ", LINE a$(n,x): GO SUB 890: RETURN
880 PRINT AT 2*x+2,0;x;AT 2*x+2,2;q$(n,x)(1 TO 14); AT 2*x+3,2;q$(n,x)(15 TO 28): RETURN
890 PRINT AT 2*x+2,17: a$(n,x);AT 2*x+2,31; x: RETURN
899 REM
900 REM ** Конец программы **
905 REM
910 IF new THEN SAVE "Victorina" LINE 70: PAUSE 30
```

```
920 CLS : STOP
990 REM
995 REM *Данные UDG-графики*
999 REM
1000 DATA "a",7,31,48,96,76,204,192,193
1010 DATA "b", 224,248,12,6,50,51,3,131
1020 DATA "c", 193,192,216,79,99,46,31,7
1030 DATA "d",131,3,27,242,198,12,248,224
1040 DATA "e",193,192,195,71,108,,46,31,7
1050 DATA "f",131,3,195,226,54,12,248,224
1060 FOR x=1 TO 6: READ e$
1070 FOR y=0 TO 7
1080 READ i: POKE USR e$+y,i
1090 NEXT y: NEXT x: RETURN
1095 REM
1100 REM **Дизайн экрана$**
1101 REM
1102 PLOT 0,164: DRAW 255,0
1104 PLOT 0,163: DRAW 255,0
1110 PLOT 0,148: DRAW 255,0
1120 PLOT 0,147: DRAW 255,0
1130 PLOT 0,10: DRAW 255,0
1140 PLOT 0,11: DRAW 255,0
1145 PLOT 0,148: DRAW 0,15
1150 PLOT 131,11: DRAW 0,152
1160 PLOT 132,11: DRAW 0,152
1165 PLOT 255,146: DRAW 0,15
1170 PLOT 11,11: DRAW 0,136
1180 PLOT 244,11: DRAW 0,136
1190 PRINT INK 1;AT 2,4;"ВОПРОСЫ"; AT 2,20; "ОТВЕТЫ"
1200 RETURN 1204 REM
1206 REM *Прием нажатой клавиши*
1208 REM
1210 LET e$=INKEY$
1220 LET i=CODE e$-48
1230 IF i>0 AND i<9 THEN RETURN
1240 GO TO 1210
1299 REM
1300 REM **Готовность данных**
1305 REM
1310 PRINT AT 11,8; BRIGHT 1; " ВАРИАНТ       ГОТОВНОСТЬ "
1320 FOR i=1 TO 5
1330 PRINT AT i+12, 11;i;AT i+12,19;("ГОТОВ" AND t$(i)="1")+("НЕ ГОТОВ" AND t$(i)="0")
1340 NEXT i
1350 RETURN
1999 REM
2000 REM** Правильный ответ**
2005 REM
2010 LET w=0.07
2020 BEEP 3*w,12: BEEP w,16: BEEP 2*w,14: BEEP w,17: BEEP 3*w,16: BEEP 5*w,12
2030 PAUSE 100
2040 RETURN
2199 REM
2200 REM **неправильный ответ**
2205 REM
2210 LET w=0.1
2220 BEEP 1.5*w,7: BEEP w,4: BEEP 1.5*w,0: BEEP w,4: BEEP .8*w,2: BEEP 2*w,2
2230 RETURN
```

BETA BASIC

Продолжение. (Начало см. на стр. 3,47,91,135)

53. SAVE <строка TO строка;> устройство;> имя

SAVE DATA <УСТРОЙСТВО;>имя

См. также DEFAULT <устройство>

В отличие от стандартного БЕЙСИКа, Бета-Бейсик позволяет выгружать не всю программу, а только ее часть, а также выгружать отдельным блоком программные переменные. Параметр <строка TO строка> указывает, начиная с какой строки производится выгрузка и по какую. Если он не указан, то выгружается вся программа целиком.

В форме SAVE DATA этот оператор служит для выгрузки только программных переменных. Если номер устройства, на которое должна происходить выгрузка, не указан, то выгрузка производится на ленту (если ранее оператором DEFAULT не было задано какое-либо иное устройство в качестве основного). Если же номер устройства задан, то выгрузка производится на соответствующий микродрайв (если командой DEFAULT в качестве устройства ввода/вывода не были назначены локальная сеть или последовательный порт RS232).

Примеры

SAVE 10 TO 200; "fragment" -часть программы, начиная с десятой строки по строку 200 включительно выгружается на ленту под именем "fragment".

SAVE 900 TO;"box" - под именем "box"выгружается часть программы, начиная со СТРОКИ 900 и до конца.

SAVE DATA "vars3" - под именем "vars3" выгружаются все программные переменные.

SAVE 20 TO 70;2;"bit" - под именем "bit" на микродрайв номер 2 выгружается часть программы, начиная со строки 20 по 70-ую.

ВНИМАНИЕ!

1. Если Вы захотите загрузить ранее выгруженную часть программы или блок программных переменных, то имейте в виду, что после команды LOAD стирается имевшаяся в компьютере Бейсик-программа, включая и нулевую строку. Блок программных переменных тоже трактуется как программа, не имеющая номеров строк.

Во избежание подобных коллизий Вам целесообразно подгружать ранее отгруженные фрагменты с помощью команды MERGE.

2. Одно из важнейших назначений команды SAVE строка TO строка состоит в том, чтобы Вы могли отгружать свои процедуры по-отдельности и формировать из них на кассете библиотеки процедур для последующего использования. Узнать начальные и конечные номера строк для каждой процедуры можно с помощью ранее рассмотренной команды LIST PROC.

Может быть, вы сочтете целесообразным перед выгрузкой процедуры переместить ее в конец программы, поменяв в ней номера строк с помощью команды RENUM и впоследствии использовать с помощью команды MERGE.

3. Наиболее целесообразное применение оператора SAVE DATA для отгрузки состояния программы. Если вы написали игровую программу, имеющую большую продолжительность, то с помощью такой команды сможете дать пользователю возможность отложить игру. Впоследствии он сможет начать ее сначала или продолжить, загрузив отложенный блок программных переменных.

54. SCROLL код направления <,число> <;x,y; ширина, длина>

Клавиша: S

См. также ROLL

Команда SCROLL имеет синтаксис очень похожий на синтаксис команды ROLL (следует сначала прочитать раздел о команде ROLL). Основное отличие состоит в том, что команда SCROLL может быть использована без параметров, а команда ROLL не может. В этом случае SCROLL вызывает скроллинг экрана на одно знакоместо вверх.

Если за командой стоит код направления 5,6,7 или 8, то текущее окно (а обычно это весь экран) будет передвинуто в заданном направлении (направление смещение стандартно для "Спектрума"):

5 - влево 6 - вниз

7 - вверх 8 - вправо

Когда часть изображения выходит за пределы экрана, она безвозвратно теряется. С противоположной стороны экрана вытягивается чистое поле.

Команда может действовать не на весь экран, а только на заданное окно. В этом случае следует задать параметры.

X, Y - координаты левого верхнего угла окна (задаются в пикселах), система координат та же, что и для команд PLOT и DRAW.

Ширина - размер окна по горизонтали (задается в знакоместах).

Длина - размер окна по вертикали (задается в пикселах).

Обе команды и SCROLL и ROLL широко применяются при разработке игровых программ, а также в различных графических приложениях. Попробуйте поэкспериментировать с теми примерами, которые были приведены для команды ROLL, заменив команду на SCROLL и посмотрите на разницу их действия.

А вот пример небольшой программы, которая передвигает по экрану символьную строку.

```
100 LET a$="HAPPY NEW YEAR"
110 FOR c = 1 TO LEN a$
120 PRINT AT 10,31; INK 7; A$(c)
130 FOR p=1 TO 8
140 SCROLL 5; 0,95;32,8
150 NEXT p
160 NEXT c
170 FOR p=1 TO 255
160 SCROLL 5; 0,95; 32,8
190 NEXT p
```

Обратите внимание на то, что в строке 120 устанавливается белый цвет INK (предполагается, что исходно цвет PAPER тоже белый). В этом случае символы, печатаемые в позиции 10,31 оказываются невидимыми и только по мере смещения влево командой SCROLL проявляются на экране. Печать символов по одному выполняет цикл по "c" (строки 100...160). Смещение их влево на одно знакоместо выполняет первый цикл по "p" (строки 130...160), а второй цикл по "p" (строки 170...190) выводит текст за пределы экрана.

Эти же принципы могут быть использованы для изящной выдачи текстов на экран, например при печати информационных сообщений.

```
200 DATA "Данным давно в далекой"
210 DATA "галактике жили были..."
300 FOR k=1 TO 2: READ a$
310 PRINT AT 21,0; INK 7; a$
320 FOR P=1 TO 8:
330 SCROLL 7
340 NEXT p
350 NEXT k
360 FOR p=1 TO 176
370 SCROLL 7
380 NEXT p
```

55. SORT

или

SORT INVERSE строковый массив
или числовой массив
или символьная строка

Клавиша: M

Команда SORT переорганизует символьные строки или символы или числа в восходящем или в нисходящем порядке. Рассмотрим для начала ее работу с символьными массивами на примере следующей программы, которая генерирует 100 десятибуквенных символьных строк. (Вы можете ускорить работу этой программы, если воспользуетесь вместо стандартной функции БЕЙСИКа RND функцией Бета-Бейсика RNDM, о которой речь пойдет ниже.)

```
100 DIM a$ (100, 10)
110 FOR s=1 TO 100
120 FOR p=1 TO 10
130 LET a$(s,p) = CHR$(RND*25+65)
110 NEXT p
150 NEXT s
160 GO TO 200
170 SORT a$
200 FOR s = 1 TO 100
210 PPINT a$(s)
220 NEXT s
```

Как только массив будет сгенерирован (а это займет определенное время), программа распечатает его в том порядке, в каком он получится.

Теперь дайте прямую команду GO TO 170 (только не RUN, а то массив будет утрачен) и Вы увидите, как тот же массив будет распечатан в алфавитном порядке.

Сортировка 100 строк займет 0.2 секунды и это время очень мало зависит от длины строк, но сильно зависит от их количества. Сортировка массива длиной 200 строк займет примерно 0.7 сек., а для массива в 400 строк - около трех секунд.

Строки сортируются в порядке возрастания кодов первых символов. Если Вы не знаете, какому символу какой код соответствует, то распечатайте себе на память эту таблицу:

```
100 FOR i=32 TO 127
110 PRINT i, CHR$ i
120 NEXT i
```

Если в строке 170 SORT a$ заменить на SORT INVERSE a$, то массив будет отсортирован и распечатан в обратном порядке.

Вы можете сортировать не весь массив, а только его часть, например:

```
SORT a$ (1 TO 20)
```

отсортирует только первые 20 элементов массива, а команда

```
SORT a$(30 TO)
```

отсортирует все элементы, начиная с тридцатого и до конца.

Можно поступить еще хитрее и отсортировать массив не по первому символу, а например по второму и всем последующим.

```
SORT a$ () (2 TO)
```

В этом случае первый символ не будет приниматься во внимание. Как видите, нам пришлось применять скобки дважды.

Команда SORT позволит Вам создавать и эксплуатировать простые, надежные и гибкие базы данных.

Когда мы говорим о базах данных, то массив, о котором шла речь, будем считать файлом, а его символьные строки - записями. В записи можно выделить различные области для разной информации, назовем их полями. Например, первые 20 символов записи отведем для имени вашего партнера. Это будет поле "ИМЯ". Следующие 20 символов отведен для его адреса - поле "АДРЕС", и, наконец еще один символ - для записи возраста - поле "ВОЗРАСТ". Всего на запись уйдет 41 символ.

Встает вопрос, каким образом одним символом выразить двузначный возраст. Это возможно. Одного символа достаточно для выражения возраста от 0 до 255, если сделать так:

```
LET a$(s;41) - CHR$ n,
```

где s - номер записи в Вашей базе, а n- возраст партнера.

Такая форма хранения чисел достаточно проста и экономит память. Но что делать, если нам понадобится хранить более сложную информацию, например размер сберегательного счета. Вы можете воспользоваться следующий приемом:

```
LET a$ (s,41 TO 46) = STR$ b
```

где b - содержимое расчетного счета. Это число будет храниться в виде строки, например: "100" или "22375".

Правда, при этом возникает один недостаток, связанный с тем, что числа будут выровнены по левому полю и сортировка сработает неправильно. Посмотрите, если у Вас есть три записи "9", "75" и "500", то после сортировки они расположатся в порядке:

```
500
75
9
```

Причина в том, что левое поле при сортировке является первич-[1]

Первый выход из положения прост, но неудобен. Вам всегда придется помнить об этой особенности и. вводя числа в свою базу, добавлять необходимое число ведущих нулей:

```
OOOOO9
000075
000500
```

Более грамотный выход - следующий. Прежде, чем заносить данные по полю "СЧЕТ" в массив, программа должна переформатировать их так, чтобы они были выровнены не по левому полю, а так, чтобы их десятичные знаки занимали одинаковые положения. Сотни - под сотнями, десятки - под десятками, тысячи - под тысячами и т.п. Тогда наш пример по результатам сортировки выглядел бы так:

```
  9
 75
500
```

Такое форматирование можно сделать с помощью функции Бета-Бейсика USING$, о которой мы расскажем чуть ниже. Пример ее использования выглядит так:

```
LET a$(s, 41 TO 46) = USING$; ("000.00",b)
```

Теперь вы можете сортировать свои записи по полям "ИМЯ", "АДРЕС", "СЧЕТ". Вы можете, например, отсортировать базу по именам, а затем первые двадцать записей - по размеру счета и т.п. Вы можете сортировать их как в восходящем, так и в нисходящем порядке.

Конечно, вам следует принять меры предосторожности при заполнении базы, чтобы не нарушить ее структуру, т.е. внося имя партнера следует его всегда записывать с первой позиции и никогда не продолжать за двадцатую, иначе нарушится поле "АДРЕС". Впрочем, все меры предосторожности следует делать программно, проверяя вводимые с помощью INPUT данные и тогда ошибки можно исключить.

Мы разобрались с символьными массивами, но команда SORT может работать и со строковыми переменными:

```
INPUT s$: SORT s$: PRINT s$
```

Если Вы здесь по команде INPUT введете строку "Fred Bloggs", то на печать получите " BFdegglors".

Это не выглядит очень полезным, но позволяет работать с числовыми данными в тех случаях, когда они представлены символьными строками, а мы на наших страницах уже неоднократно упоминали о том, что это весьма экономичный способ хранения чисел в памяти компьютера.

Команда SORT может работать и с числовыми массивами, как с одномерными, так и с двумерными. Синтаксис ее применения тот же, что и при работе с символьными массивами. Двумерные массивы мы можем представлять для себя в виде таблиц, в которых первая размерность - это номер ряда, а вторая - номер столбца. Так, команда

```
SORT b (1 TO 20) (2)
```

---

[1] В оригинале пропущена строка (Прим. OCR)

отсортирует первые двадцать рядов таблицы по второму столбцу. Обратите внимание на то, что мы всегда должны использовать хотя бы одну пару скобок при имени массива b() для того, чтобы компьютер отличал массив от простой переменной b, даже и в тех случаях, когда внутри скобок ничего не стоит.

```
SORT b()
```

Если используется и вторая пара скобок (для указания номера столбца, по которому производится сортировка), в ней должно быть не более одного числа, в отличие от сортировки символьных массивов.

Сортировка числовых массивов идет примерно в четыре раза медленнее, чем сортировка строковых массивов. Это связано с необходимостью иметь дело с интегральной (пятибайтной) формой записи чисел в "Спектруме" (см. "Программирование в машинных кодах"; М:, "ИНФОРКОМ", 1990, 1992).

В отличие от сортировки символьных массивов и строк, при числовой сортировке первыми идут более высокие числа, а затем низкие. Сделать порядок следования чисел возрастающий можно очевидно командой SORT INVERSE.

56. SPLIT (не ключевое слово).

Фактически вместо этой команды вводится символ "<>".

Клавиша SYMBOL SHIFT + W (не в графическом режиме, а в обычном).

Фактически это не оператор языка, а дополнительная возможность редактирования программы. Если Вы редактируете очень длинную строку (она находится в нижней части экрана) и хотите часть ее ввести в программу, а с оставшейся частью продолжить работу, то можете самым первый символом в любом операторе строки поставить символ "<>" и нажать ENTER. в этом случае начало строки до этого символа перейдет в программу со своим номером строки, а оставшаяся часть останется в области редактирования с тем же номером строки и курсором справа от него, чтобы вы могли первым делом изменить его так, как Вам надо.

Например, в нижней части экрана у вас было:

```
10 PRINT "hello": GO TO 10: <> PRINT "goodbye"
```

Если Вы теперь нажмете ENTER, то в листинг программы пойдет:

```
10 PRINT "hello": GO TO 10
```

а в нижней части останется:

```
10 (курсор) PRINT "goodbye"
```

Теперь Вы можете изменить номер строки и, соответственно, отправить эту строку в ту часть программы, в какую хотите, но можете передумать и "подшить" ее к десятой строке с помощью команды JOIN.

57. TRACE номер строки

или

TRACE: оператор: оператор:..

Клавиша: T.

Эта команда относится к разряду отладочных. С ее помощью вы можете запускать БЕЙСИК-программу на выполнение с постоянной распечаткой результатов заданной строки, заданного оператора или заданной переменной. При этом Вы можете изменять (замедлять) скорость исполнения программы.

Существуют две формы оператора TRACE.

1. TRACE номер строки - вызывает переход GO SUB к этой строке перед исполнением любого оператора в Вашей программе. Это не относится к операторам самой этой строки, а то программы бы зациклилась.

2. Вторая форма: TRACE оператор, оператор: ....

В этой случае перед исполнением любого оператора вашей программы исполняется

последовательность операторов, стоящих после TRACE.

И в том и в другом случае, при использовании оператора TRACE Вам доступны две вспомогательные переменные - lino и stat.

LINO - это номер строки, которая сейчас будет выполняться. STAT - номер оператора в этой строке, который сейчас будет исполняться.

При исполнении трассирущей подпрограммы режим TRACE естественно отключается и вновь будет включен по оператору RETURN, который завершает эту подпрограмму. Содержимое подпрограммы можете избрать любое, например:

```
9000 PRINT INVERSE 1: lino:":"; stat: RETURN
```

Введите эту подпрограмму в свою программу и введите команду TRACE 9000 в ту точку программы, с которой хотите начать отладку.

Если Вы хотите воспользоваться второй формой оператора TRACE, то можете не вводить строку 9000, а вставить в ту строку, с которой хотите начать отладку, последовательность операторов:

```
TRACE: PRINT INVERSE 1; lino; ":"; stat: RETURN
```

Отключить режим трассирования, начиная с какой-то строки в Вашей программе можно вставив туда оператор TRACE 0.

Приведенная выше TRACE-подпрограмма позволит получить на экране номера строк и номера операторов, исполняемых Вашей программой в любой момент. Для того, чтобы отличать их от тех данных, которые программа должна выдавать на экран сама по себе, включен режим инверсной печати INVERSE 1.

Если Вы хотите, чтобы не только номера исполняемых строк, но их содержимое было постоянно перед глазами, Вы можете использовать команды:

```
LIST lino TO lino
```

или

```
LIST lino-1 TO lino
```

Если у Вас задан интервал между строк больше, чем 1, вторая команда будет отличаться от первой тем, что при печати содержимого строк не будет изображаться позиция курсора.

Команды RUN и CLEAR также отбивают режим трассирования, как и TRACE 0.

Если Вы хотите замедлить исполнение программы, введите в подпрограмму TRACE оператор PAUSE или сделайте это иным доступным Вам способом, например через BEEP.

Вы можете распечатать также и содержимое интересующих Вас переменных, задав их печать в подпрограмме, но тогда постарайтесь, чтобы эти переменные были объявлены как можно ранее в Вашей программе, иначе можете получить сообщение "Variable not found".

Чтобы отличать ту печать, которую делает оператор TRACE от той, которую ведет сама программа, Вам может быть захочется использовать оператор PRINT AT (а это может пригодиться, если программа строит некоторое графическое изображение и Вы не хотите его нарушать посторонней печатью). Но в этом случае Вам придется позаботиться о том, чтобы перед вызовом трассирующей подпрограммы запоминались координаты текущей позиции печати, а после ее исполнения они бы восстанавливались. Нижеприведенная подпрограмма позволит это сделать:

```
9000 LET POS = DPEEK(23688)
9010 PRINT AT 0,0; lino;":";slat; "    ","a$= "; a$
9020 DPOKE 23688, POS: RETURN
```

Системная переменная SPOSN, расположенная в адресах 23688 и 23689 хранит информацию о текущих координатах позиции печати. Запоминая и восстанавливая ее через переменную POS, Вы добьетесь того, чего хотели.

Поработав какое-то время с Бета-Бейсиком, Вы сами для себя выработаете наиболее удобную отладочную процедуру и тогда сможете присвоить ее с помощью команды DEF KEY какой-либо клавише и отгрузить вместе с самим Бета-Бейсиком на ленту, чтобы впоследствии вызывать одним нажатием тогда, когда надо.

58. UNTIL условие.

Клавиша: K.

Оператор позволяет исполнять циклы DO - LOOP до тех пор, пока "условие" не станет

справедливым.

Подробности см. в описании операторов DO - LOOP.

59. USING, USING$

Клавиша: U.

И USING и USING$ служат для того, чтобы задать формат, в котором Вы хотите распечатать числа. Оператор USING самостоятельно не используется, а употребляется только в качестве квалификатора оператора PRINT или LPRINT, в то время, как USING$ является функцией, имеет самостоятельное значение в виде символьной строки. Функция USING$ выдает символьную строку такой, какой она могла бы быть напечатана при использовании оператора PRINT USING. Благодаря этому появляется возможность использовать форматированные строки не только с командой PRINT, но и с любой другой, например LET, которая может работать со строками.

При использовании USING или USING$ желаемый формат задается в виде форматной строки. Это символьная строка, в которой знаком "хэш" (#) обозначены ведущие пробелы, символ "ноль" обозначает ведущие нули и любой из них может служить для указания значащих цифр после десятичной точки.

```
100 FOR n=1 TO 30:
110 LET X-RND*100
120 PRINT x, USING "###.##";x
130 NEXT n
```

Вы получите на экране два столбца. Слева - неотформатированная печать, а справа - отформатированная. Обратите внимание насколько она выглядит аккуратнее. Попробуйте в строке 120 изменять форматную строку и посмотрите, как будет меняться результат. Обратите также внимание, что при форматировании чисел с помощью USING происходит их округление до заданной в формате десятичной цифры.

А вот еще несколько примеров того, как работает форматная строка для числа 12.3456.

```
"##.#"          12.3
"###.#"         _12.3
"####.##"       __12,35
"000.00"        012.35
"00"            12
"$00.00"        $12,35
"0.00"          %..3
```

Предпоследний пример демонстрирует возможность использования в форматной строке произвольных символов, а не только знаков "#" и "0".

Последний пример демонстрирует случай переполнения формата. Нельзя одним знаком выразить двузначную целую часть - об этом свидетельствует символ %, выводимый на печать, как индикатор ошибки.

Функция USING$ очень похожа на оператор USING. Разницу продемонстрируем на примерах. Вместо

```
PRINT USING a$; number
```

можно сделать

```
PRINT USING$ (a$, number)
```

или

```
LET b$ = USING$ (a$,number) PRINT b$
```

60. VERIFY <строка TO строка;> устройство;> имя

VERIFY DATA <устройство:>имя

Как и стандартная команда VERIFY, эта команда с параметрами служит для проверки правильности выгруженного на ленту или микро-драйв блока данных (программы или ее фрагмента).

Мы не будем останавливаться на синтаксисе команды - он в точности соответствует синтаксису команд SAVE и SAVE DATA, рассмотренных ранее.

61. WHILE условие.

Клавиша: J.

Оператор позволяет исполнять циклы DO - LOOP до тех пор, пока "условие" справедливо.

Подробности см. в описании операторов DO - LOOP.

62. WINDOW номер окна <,x,y,w,l>

Клавиша: 5.

См. также CLS, CSIZE.

Команда WINDOW позволяет Вам назначить часть экрана для выдачи информации при печати или листании. Если окон несколько, то каждое имеет текущую координату позиции печати, а также статусы OVER, BRIGHT, FLSH, INK, PAPER и CSIZE.

В качестве номера окна Вы можете использовать любое число от 1 до 127, причем заданное Вами число не имеет никакого специального значения - это просто опознавательный номер, присущий данному окну.

Все параметры, присвоенные каждому окну, хранятся выше адреса RAMTOP (который, если надо, понижается). Это защищает данную информацию от уничтожения командой NEW и позволяет выгрузить ее на ленту вместе с кодом самого Бета-Бейсика.

Нулевое окно - это весь экран и именно оно является "текущим окном" в момент первого запуска Бета-Бейсика. Это означает, что исходный размер символов и установка цветов - стандартные. Чтобы задать иное окно, Вы можете делать например так:

```
WINDOW 1,0,175,128,176
```

Заданное окно имеет координаты левого верхнего угла 0,175. Ширина окна - 128 пикселов (пол-экрана), а высота - 176 пикселов (весь экран). Текущий установленный размер символов - стандартный (8x8), а установка цветовых атрибутов соответствует нулевому окну, хотя их можно и изменить.

Мы только что задали первое окно, но это еще не сделало его текущим. Чтобы сделать это, надо дать команду WINDOW, указав при ней только номер окна:

```
WINDOW 1
```

Указанное окно стало текущим (если оно было задано). Если же вы забыли его предварительно задать, то получите сообщение об ошибке устройства ввода/вывода - "Invalid I/O device". Единственное окно, которое задается автоматически, без Вашего участия - нулевое окно.

После того, как первое окно стало текущим, весь вывод на печать по командам PRINT, LIST, PLOT, DRAW и т.п. будет производиться только в пределах области данного окна, то есть, только на левой половине экрана. Если хотите поэкспериментировать с окнами другого размера, можете изменить определение первого окна или задать любое другое. В любом случае все изменения, которые Вы произведете, будут видны на экране только после того, как Вы дадите команду WINDOW n, даже если окно с номером n перед этим и так было текущим. Только после этой команды начинают действовать параметры окна, установленные при его задании.

При выходе из окна, а точнее говоря при вызове другого окна, сохраняются все параметры, соответствовавшие данному окну CSIZE, INK, PAPER и т.п., включая и координаты текущей позиции печати. Они сохраняются в областях памяти выше уровня RAMTOP.

Когда вы вновь войдете в это окно, вызвав его командой WINDOW n, все эти параметры будут для него восстановлены. Таким образом, Вы можете менять размер печатаемых символов CSIZE и цветовые атрибуты переключением между окнами.

В нижеприведенном примере задаются два окна и поочередно в них производится печать.

```
10 WINDOW 1,0,175,128,176
20 WINDOW 2,128,151,128,80
30 WINDOW 1: INK 1: PAPER 6
40 WINDOW 2: INK 7: PAPER 1:CSIZE 4,8
50 WINDOW 0
60 PRINT WINDOW 1;"one";WINDOW 2; "two";:GO TO 60
```

Если Вам надо очистить какое-либо окно, Вы можете использовать оператор CLS n, где n - номер окна, подлежащего очистке. В самом начале программы целесообразно давать команду CLS 0, очищая весь экран и подготавливая его к работе.

Если Вам надо удалить из памяти параметры задания какого-либо окна, для этого служит оператор ERASE WINDOW n

63. XOS, XRG, YOS, YRG.

Это не ключевые слова, а своеобразные переменные, с помощью которых можно изменять масштаб экрана и начало координатной сетки, используемой операторами PLOT, DRAW, DRAW TO, CIRCLE, GET и FILL.

XOS - смешение оси X от стандартной.

YOS - смешение оси Y от стандартной.

У этих переменных есть одна особенность. В отличие от прочих, команды CLEAR и RUN их не уничтожают, а устанавливают в заранее заданное фиксированное значение.

Попробуйте дать команду CLEAR, а затеи сделать PRINT XOS или PRINT xos и вы получите "0", а не "Variable not found", как это было бы для обычных переменных.

Оба смещения имеют нулевые значения, если Вы не зададите иные с помощью команды LET.

```
LET XOS=128, YOS=88
```

Такое задание начала координат очень удобно для команды PLOT, т.к. теперь координата X изменяется не от 0 до 255, а от -128 до 127 и, соответственно, координата Y изменяется от -88 до 87.

Команда CLS, как Вы знаете, не только очищает весь экран, но еще и устанавливает текущую позицию печати в координаты 0,0. Если Вы изменили начало координат с помощью xos и YOS, то начальная позиция печати будет устанавливаться после CLS в новое начало координат.

Переменные XRG и YRG определяют масштаб, в котором исполняются команды PLOT, DRAW и пр. Исходное значение XRG = 256 (Вы можете напечатать до 255 различных точек вдоль оси X), а для YRG = 176. Изменив XRG и YRG, Вы меняете масштаб изображения.

```
10 GO SUB 100: REM normal
20 LET XRG = 128: GO SUB 100
30 LET YRG = 88: GO SUB 100
40 LET XRG = 256: GO SUB 100:
50 STOP
100 CLS: PLOT 0,0: DRAW 50,0
110 DRAW 0,50: DRAW -50,0
120 DRAW 0,-50: PAUSE 100
130 RETURN
```

По этой программе сначала рисуется нормальный квадрат, затем он вытягивается по оси X, затем по осям X и Y и, наконец, только по оси Y.

Нижеприведенная программа рисует график функции "синус", используя как смещение начала координат, так и изменение масштаба.

```
100 LET XRG = 2*PI: REM 360 град.
110 LET YRG = 2.2
120 LET YOS = 1.1
130 FOR n=0 TO 2*PI STEP 2*PI/256
140 PLOT n, SIN n: NEXT n
```

Обратите внимание на то, что величина начального смещения начала координат также подвергается масштабированию в заданном диапазоне.

При изображении дуг или окружностей есть одна особенность. Так, последняя точка дуги будет напечатана в соответствии с новой системой координат и с масштабом, но сама кривая - не подвергнется изменениям. То же относится и к окружности. Центр ее будет помещен в точку в соответствии с новой системой координат и с масштабом, но у Вас нет средств промасштабировать радиус, так что если Вы захотите подобным способом получить растянутую вдоль какой-либо оси окружность, то у Вас ничего не получится.

Есть интересная возможность использования масштабирования. Предположим, что вы подготовили программу для того, чтобы получать графическое изображение на полном

экране, после этого Вам захотелось разделить экран пополам (по вертикали) и в правой половине экрана распечатывать листинг программы, а в левой половине - результат ее работы. Это сделать несложно, задав два окна. Но теперь есть проблема в том, что раз графические команды у Вас были рассчитаны на полный экран, то они не смогут работать нормально в его одной половине. В этом случае при переключении на окно Вы можете поменять и масштаб LET XRG = 128.

РАЗДЕЛ 3. ФУНКЦИИ

Бета-Бейсик версии 3.0 имеет более 20 новых функций. Эти функции определены в нулевой строке (которая при листинге не воспроизводится). Там же содержится и указание на машинный код, выполняющий непосредственные расчеты для этих функций.

В программе эти функции существуют как обычные функции, заданные пользователем, а в листинге они являются обычными ключевыми словами. Например, если строка программы содержит FN S$, в листинге это проявляется, как STRING$ и курсор проскакивает это слово за одно нажатие. Функции, заданные пользователем и не являющиеся составной частью Бета-Бейсика, действуют как обычно.

Может случиться так, что Вами ранее уже была подготовлена программа, использующая пользовательские функции, совпадающие с функциями Бета-Бейсика. В этом случае Вам придется поменять обозначения своих функций во избежание конфликта. Это легко можно сделать с помощью команды ALTER.

Ввод имен функций Бета-Бейсика может осуществляться полностью, (если Вы работаете в режимах KEYWORDS 3 или 4) или вводом FN, затем буквы, а затем символа "$" или символа "(". "FN" можно получить нажатием клавиши "Y" в графическом режиме или по буквам. Дополнительные функции Бета-Бейсика не будут работать, если в памяти отсутствует машиннокодовая часть программы Бета-Бейсик. С другой стороны, если будет отсутствовать нулевая строка, то будут потеряны определения функций и Вы получите сообщение об ошибке "FN without DEF". В этом случае остальная часть языка будет работать, но воспользоваться новыми функциями Вы не сможете.

При выгрузке написанной Вами программы нулевая строка выгружается вместе с ней и поэтому если Вы загрузите программу, написанную под Бета-Бейсиком, то нулевая строка будет присутствовать на месте, в то же время, если Вы загрузите программу, написанную ранее в стандартном Бейсике, то нулевая строка погибнет. Поэтому если Вы хотите загрузить в Бета-Бейсик программу, написанную не в нем, то сначала очистите память командой NEW - уберутся все строки, кроме нулевой, а потом подгружайте свою программу с помощью MERGE, что не затронет нулевую строку.

Если же Вы захотите специально удалить нулевую строку, то это можно сделать командой DELETE 0 TO 0.

Ниже мы приводим список новых функций Бета Бейсика.

AND	FN A(
BIN$	FN B$
CHAR$	FN C$
COSE	FN C(
DEC	FN D(
DPEEK	FN P(
EOF	FN K(
FILLED	FN F(
HEX$	FN H$
INARRAY	FN U(
INSTRING	FN I(
ITEM	FN T(
LENGTH	FN L(
MEM	FN M(
MEMORY$	FN M$

MOD	FN V(
NUMBER	FN N(I
OK	FN O(
RNDM	FN R(
SCRN$	FN K$
SHIFT$	FN Z$
SINE	FN S(
STRING$	FN S$
TIME$	FN T$
USING$	FN U$
XOR	FN X(

1. AND (число, число)

FN A (число, число)

По написанию эта функция похожа на обычное ключевое слово AND, но в программе их можно различить по различному синтаксису. Она выполняет побитную операцию AND для двух чисел от 0 до 65535. Если какой-то бит и в первом числе и во втором равен единице, то только в этом случае соответствующий бит результата тоже будет равен единице. Если хотя бы один из них равен нулю, то и в результате этот бит будет равен нулю.

Чтобы лучше понять, как все это происходит, мы воспользуемся функцией BIN$, о которой речь пойдет ниже.

```
BIN$ (254) = "11111110"
BIN$ (120) = "01111000"
BIN$(AND(254,120)) = "01111000"
```

Таким образом, с помощью функции AND можно выключать (маскировать) нежелаемые биты, например:

```
PRINT AND(BIN 00000111, ATTR(line,column))
```

Эта конструкция позволит получить значение INK, установленное для знакоместа с координатами (line, column), а прочие цветовые атрибуты в байте окажутся замаскированы. Поскольку BIN 00000111 = 7, то мы могли записать эту команду и так:

```
PRINT AND (7,ATTR(line,column))
```

Нижеприведенный пример напечатает на экране слово "Bang", если на клавиатуре будет нажата клавиша "F". При этом не имеет значения, какие еще клавиши будут нажаты вместе с ней. В примере клавиатура рассматривается как серия внешних портов. Если Вы не знакомы с тем, как это происходит, читайте нашу книгу ("Программирование в машинных кодах"; М:. "ИНФОРКОМ", 1990, 1992).

```
10 IF AND(BIN 00001000, IN 65022) = 0 THEN PRINT "Bang! ";
20 GO TO 10
2. BIN$ (число) FN B$ (число)
```

Функция дает двоичный эквивалент для заданного числа в виде восьмисимвольной строки, если число меньше 256 или в виде шестнадцатисимвольной строки, если число больше 255, но меньше 65535. Речь идет только о целых числах.

Эта функция очень полезна для тех, кто осваивает программирование в машинных кодах и (или) пользуется функциями Бета-Бейсика для битовых операций AND, OR, XOR.

Кроме того, эта функция может пригодиться при работе с генератором символов, с графикой пользователя UDG и при анализе цветовых атрибутов, системных переменных или при опросе клавиатуры, например:

```
10 PRINT AT 10, 10; BIN$(IN65022)
20 GO TO 10
```

Символьная строка, выдаваемая этой функцией, - это последовательность нулей и единиц. Может быть, вам захочется, чтобы это были иные символы, например, при распечатывании на экране конструкции шрифта знакогенератора очень удобно вместо единиц иметь крестики, а вместо нулей - пробелы, это тоже можно сделать, если принудительно заслать (POKE) в ячейку 62865 код буквы "X", а в ячейку 62869 - код пробела " ".

3. CHAR$ (число)

FN C$ (число)

Эта функция позволяет конвертировать целые числа до 65535 в двухсимвольные строки, что позволяет более экономно использовать память и более компактно хранить данные. Эквивалентом из обычного БЕЙСИКа будет следующая конструкция:

```
LET a = INT(number/256): LET b = number - a*256 LET c$ = CHR$ a + CHR$ b
```

Полученная в результате строка не всегда может быть распечатана, т.к. не всегда в результате получается печатный символ. В этом случае может, например, появиться сообщение об ошибке " invalid colour", если в итоге получился символ, являющийся управляющим кодом. Перед печатью на экран желательно сделать обратную конверсию. Это может сделать функция NUMBER, которая заменит двухсимвольную строку на соответствующее ей число. Поскольку на хранение двухсимвольной строки расходуются только два байта, а не пять, как на число, то мы рекомендуем Вам очень серьезно подумать о том, чтобы использовать эти возможности, если Ваша программа обслуживает и хранит большие массивы чисел (что часто бывает в научных расчетах).

Если Вы не можете работать только с целыми числами, то попробуйте воспользоваться преобразованием их. Например, вам надо хранить число 87.643. Умножив его на 100, вы получите 8764.3. Округлите и преобразуйте 8764 в двухсимвольную строку перед сохранением этого числа в памяти. Когда же оно вновь Вам понадобится, сделайте обратное преобразование, с помощью NUMBER преобразуйте двухсимвольную строку в 8764 и поделите результат на 100. Полученное значение 87.64 имеет неплохую точность по сравнению с исходным числом, эта точность достаточна для большинства практических приложений.

Одним словом, действия над числами Вы можете выполнять как обычно, но перед закладкой числа на хранение Вы его конвертируете, а после вызова вновь восстанавливаете. Это в некоторой степени (в небольшой) снизит скорость расчетов, но очень значительно сэкономит расход памяти. Если рассматривать искусство программирования как вечный компромисс между скоростью операции и расходом памяти, то здесь вам есть поле для деятельности.

Нижеследующий пример демонстрирует последовательность манипуляций по преобразованию числового массива так, как было описано выше.

```
100 DIM a$ (500,2)
110 FOR i = 1 TO 500
120 LET a$(i) = CHR$ (i*10)
130 NEXT i
140 PRINT "Array is ready!"
100 PRINT "Press any key!'
160 PAUSE 0
170 FOR i=1 TO 500
180 PRINT i, NUMBER(a$(i))
190 NEXT i
```

Полученный в результате массив будет занимать только 1 килобайт, в то время, как при хранении чисел стандартным способом его размер был бы 2.5 килобайта.

Оператор SORT работает с конвертированными в символьные строки массивами совершенно правильно.

4. COSE (ЧИСЛО)

FN C (число)

Это функция "косинус". Она отличается от функции COS стандартного БЕЙСИКа тем, что имеет меньшую (но достаточную в большинстве практических приложений) точность, зато выполняется в шесть раз быстрее.

5. DEC (символьная строка)

FN D (символьная строка)

См. также HEX$ (число).

Эта функция преобразует символьную строку, выражающую шестнадцатиричное

число в целое десятиричное число от 0 до 65535. При этом шестнадцатиричное число должно быть выражено символьной строкой длиной от 1 до 4-х символов, а регистр символов - не имеет значения.

DEC ("FF") - 255
DEC ("10") = 16
DEC ("4000") = 16384
DEC ("e") = 14

Эта функция предназначена для обеспечения ввода в программу шестнадцатиричных данных. Например:

```
INPUT a$: POKE address,DEC(a$)
```

Использование "пустой строки" или строки, содержащей символы, не являющиеся шестнадцатиричной формой записи дает сообщение об ошибке "Invalid argument".

(Окончание следует).

Михайленко В. С.

ЗАЩИТА ПРОГРАММ

Продолжение.
(Начало: 9-16, 53-60,97-104, 141-146)

3.2 Работа со встроенными машинными кодами.

В предыдущей статье мы с Вами рассмотрели, как блокировать действие защитных управляющих кодов. (Для этих целей была использована специальная программа, существуют и другие способы просмотра, которым будет посвящена следующая статья данного пособия). Однако просмотра содержимого одного лишь Бейсика бывает явно недостаточно для анализа принципа работы программы в целом и становится необходимым изучение встроенных в Бейсик машиннокодовых процедур. Именно этому вопросу и посвящен данный раздел.

Прежде, чем приступать к работе, нам необходимо определиться с целью, которую вы преследуете при вскрытии процедуры в машинных кодах. Если Вы хотите осуществить какие-либо изменения, то Вам необходимо достаточно кропотливое изучение каждого программного блока. В том случае, если Вы просто изучаете принципы программирования тех или иных авторов, то на первое место по степени приоритетности выступает анализ общей структуры программы.

Здесь мы постараемся рассмотреть оба подхода с тем, чтобы иметь представление о каждом из них. Многие из Вас, уважаемые читатели, наверняка сталкивались с программами, вскрытыми хаккерами. Наиболее известным из них является BILL GILBERT, поскольку программы, вскрытые им, получили наибольшее распространение. И многие из Вас наверняка пытались пробраться сквозь дебри его защиты, чтобы попытаться изменить что-либо, или же просто понять ее принцип действия.

Возможно, что это удалось не всем. Однако, не огорчайтесь. Сегодняшний пример мы построим на исследовании программы, вскрытой этим хаккером, а в последующих выпусках исследуем приемы этого взломщика еще более подробно.

В этой статье мы рассмотрим применение процедур в машинных кодах для защиты программ на примере игры GAME OVER (IMAGINE/DINAMIC). Судя по дате, проставленной хаккером, эта программа была взломана им в 1987 году.

Для того, чтобы с наименьшими усилиями вскрывать программы Билла Гилберта, необходимо уяснить несколько деталей его специфической защиты. В будущем это может очень сильно пригодиться. Так, практически все программы, вскрытые этим хаккером, не имеют защиты от BREAK, т.е. программу можно остановить, нажав клавишу BREAK.

Второе - почти все программы в связи с отсутствием защиты от BREAK, имеют мощную защиту от листинга. В большинстве случаев используются защитные POKES совместно с методом зануления и совместно со встроенными процедурами в машинных кодах. Причем, во многих случаях бывает невозможно работать с программой, не блокировав защитные POKES.

И третье - буквально все программы, вскрытые им, имеют защиту от MERGE, аналогичную описанной Главе 3 первого тома.

Защита от MERGE необходима для того, чтобы не удалось достаточно легко блокировать систему защитных POKES.

Зная эти три особенности, будем грамотно осуществлять взлом, не оставляя без внимания ни одну из них. Наиболее разумным методом ввода первого Бейсик-файла в компьютер является ввод его через программу блокировки автозапуска, рассмотренную в разделе 2.3 второго тома. Тогда защитные POKES ни оказали бы никакого влияния на просмотр листинга данной программы. Однако, можно ввести данный Бейсик-файл и просто подав команду LOAD "" . После его загрузки остановить работу программы нажатием

клавиши break. Теперь весь экран окрасился в черный цвет, а немного ниже середины экрана в прямоугольной рамке можно увидеть надпись:

Естественно, что после загрузки данного Бейсик файла с помощью программы блокировки автозапуска надпись в прямоугольной рамке не появится, поскольку ни одна Бейсик-команда не запустилась на выполнение. Для нас это положение выгодное, поэтому в случае загрузки с использованием LOAD "" нам необходимо наверстать упущенное. Когда это будет сделано, Вам будет сообщено дополнительно.

Во первых, сразу после загрузки, желательно подать команду BORDER 7. Это позволит Вам видеть все команды вводимые с клавиатуры. Желательно еще привести в порядок цвета INK и PAPER, подав команды:

```
INK 0: PAPER 7
```

Теперь, подав команду LIST, вы столкнетесь с первичной защитой Билла Гилберта. В данном случае в качестве защиты использована системная переменная 23570. Ее действие немного рассмотрено в Главе 1 первого тома и достаточно основательно в "ZX-РЕВЮ" N10,1991 (стр. 211). Характерным признаком, свидетельствующим об использовании этой системной переменной, является выползание текста снизу вверх вместо раскрутки его сверху вниз, а также появление сообщения:

```
5 OUT OF SCREEN
```

(вне экрана) - во время Ваших попыток автоматического листинга. Как Вам уже вероятно известно, подобный эффект достигается после подачи команды

```
POKE 23570,16
```

Нормальным состоянием данной системной переменной является наличие там числа 6. Поэтому защиту можно блокировать, подав с клавиатуры команду:

```
POKE 23570,6.
```

Только теперь, изменив цвета INK, PAPER и BORDER, а также блокировав действие защитной системной переменной, мы достигли того эффекта, который был получен при использовании для загрузки программы блокировки автостарта. Поскольку все эти изменения осуществила данная программа в ходе своей работы, изменение цветовых атрибутов осуществилось из встроенной подпрограммы в машинных кодах, а POKES, блокирующий защиту, был установлен прямо в Бейсике.

Теперь настало время подробно изучить структуру Бейсик-файла, чтобы через него выйти на программу в машинных кодах - основную цель нашего исследования. Для этих целей используем программу блокировки действия управляющих кодов. Ввиду того, что данный Бейсик-файл имеет значительный объем, необходимо расширить область обработки моей программы. Для этого, после загрузки ее с адреса 62030, необходимо подать команду с клавиатуры:

```
POKE 62032,1
```

Если в будущем Вам и этого окажется недостаточно, можно увеличить это число, либо использовать более точный метод, описанный в предыдущем разделе данной главы.

Программа действует очень быстро и после команды

```
RANDOMIZE USR 62030
```

практически сразу появляется сообщение O.K.

Теперь, после подачи команды LIST, Вы увидите приблизительно такую картину:

```
0>REM FORMAT *BG1987 !REM \NOT
 CONTINUE GO SUB VAL NOT PNOT !NEW
NOT "[\CODE 2\2H\vPIa! RESTORE
STEP ATN !STORE CODE OPEN
#RETURN >STEP ! !PLOT !XCOPY
"[\>x2\> 2\" STEP INT $ STEP INT
$ STEP INT $ COPY STEP INT
$!X'INK <> COPY !!!!AND
))/<>:BCBB:^P^LLIST LLIST > = %CHR$
```

```
""CHR$
   PPFPP^LIST REM
J*FOR BRIGHT RUN DATA POKE >
t c<

   0>REM ! @ GO SUB VAL ICOPY
NOT @! VAL K "SAVE GO SUB VAL NOT
l#

   0>GO TO USR (PEEK VAL
"23635"+VAL "256"*PEEK VAL
"23636"+ VAL "17")
   1 POKE VAL "23570", VAL "16":
LOAD "GAME OVER$"CODE VAL "5E4":
CLS: RANDOMIZE USR VAL "5E4"
    2 LOAD "Game over1 "CODE: LOAD
"Game Over2"CODE: CLS: LOAD "Game
OVER3" CODE: RANDOMIZE USR VAL
"24100"
```

Здесь первые две строки с нулевыми номерами - это встроенные процедуры в машинных кодах. Третья строка, имеющая номер 0, запускает машинный код на выполнение командой:

```
GO TO USR
```

Если вы вызовите эту строку командой EDIT для редактирования и замените GO TO USR оператором PRINT, стерев предварительно номер строки, то получите, что Вы просите компьютер напечатать адрес старта процедуры в кодах. После нажатия клавиши ENTER вы получите число 23772. Это и есть адрес запуска встроенной процедуры в кодах. Именно туда идет реальное обращение Бейсик-программы, т.к. это первая реально выполняющаяся строка, ввиду того, что все предыдущие начинались командой REM.

Для того, продисассемблировать данный машинный код, необходимо воспользоваться специальной системной программой, предназначенной для этих целей. Я использовал программу MONITOR 48, причем открыл в ней дополнительный режим, намного упростивший работу в данном конкретном случае, а в принципе он упрощает процесс изучения и отладки любых программ в машинных кодах.

* * *

ПРИМЕЧАНИЕ

В большинстве справочной литературы к программе MONITOR 48, данный режим работы этой программы не описан. Отсутствует эта информация и в фирменной инструкции, перевод которой был подготовлен "ИНФОРКОМом". Поэтому сегодня в конце раздела приведена специальная статья с описанием этого нового режима работы.

Поскольку сегодня мы работаем с программой, вскрытой Биллом Гилбертом, то надо указать на одну из особенностей его программирования встроенных в Бейсик машиннокодовых процедур. Перед началом работы, вся подпрограмма в машинных кодах переносится в область памяти, начиная с 50000 и лишь потом она начинает работать уже в этой области оперативной памяти.

Не является исключением и наш случай, программа в машинных кодах начинается следующими командами:

```
5CDC            LD HL,5CEA
5CDE            LD DE,C350
5CE2            LD BC, 03E8
5CE5            LDIR
5CE7            JP C350
5CEA            LD HL, C3E6
```

Как видно из приведенной выше распечатки, строки 5CDC - 5CE2 подготавливают регистры процессора для переноса блока памяти командой LDIR. После этого

осуществляется безусловный переход на начальный адрес уже перемещенного блока, т.е. 50000 (C350H). Строка 5CEA дана для того, чтобы Вы имели представление о точке, с которой начинается перенос кодов на новое место

Необходимый для понимания принципа работы программы дисассемблер приведен ниже уже для новых адресов.

```
C350  21E6C3          LD HL,LC3E6
C353  227B5C          LD (#5C7B),HL
C356  AF              XOR A
C357  328D5C          LD (#5C8D),A
C35A  32485C          LD (#5C48),A
C35D  76              HALT
C35E  01A761          LD BC,#61A7
C361  210313          LD HL,#1303
C364  E5              PUSH HL
C365  CDB71E          CALL #1EB7
C368  21761B          LD HL,#1B76
C36B  E5              PUSH HL
C36C  AF              XOR A
C36D  D3FE            OUT (#FE),A
C36F  3E02            LD A, #02
C371  CD0116          CALL #1601
C374  21C9C3          LD HL,C3C9
C377  7E        LC377 LD A,(HL)
C378  FEFF            CP #FF
C37A  CA81C3          JP Z,LC381
C37D  D7              RST #10
C37E  23              INC HL
C37F  18F6            JR LC377
C381  2158FF    LC381 LD HL,#FF58
C381  227BC5          LD (#5C7B),HL
C387  3E78            LD A,#78
C389  3E905C          LD (#5C90),A
C38C  3E20            LD A,#20
C38E  32915C          LD (#5C91),A
C391  0E36            LD C,#36
C393  060E            LD B,#0E
C395  CDE522          CALL #22E5
C398  0E9B            LD C,#9B
C39A  0600            LD B,#00
C39C  1E01            LD E,#01
C39E  1600            LD E,#00
C3A0  CDBA24          CALL #24BA
C3A3  0E00            LD C,#00
C3A5  060B            LD B,#0B
C3A7  1E00            LD E,#00
C3A9  1601            LD D,#01
C3AB  CDBA24          CALL #24BA
C3AE  0E9B            LD C,#98
C3B0  0600            LD B,#00
C3B2  1EFF            LD E,#FF
C3B4  1600            LD D,#00
C3B6  CDBA24          CALL #24BA
C3B9  0E00            LD C,#00
C3BB  060B            LD B,#0B
C3BD  1E00            LD E,#00
C3BF  16FF            LD D,#FF
C3C1  CDBA24          CALL #24BA
C3C4  215827          LD HL,#2758
C3C7  D9              EXX
C3C8  C9              RET
C3C9  161307    LC3C9 DEFB #16,#13,#07
C3CC  110210          DEFB #11,#02,#10
C3CF  071301          DEFB #07,#13,#01
C3D8  909192          DEFB #90,#91,#92
```

```
C3D5  939495            DEFB #93,#94,#95
C3D8  969798            DEFB #96,#97,#96
C3DB  999A9B            DEFB #99,#9A,#9B
C3DF  9C9D9E            DEFB #9C,#9D,#9E
C3E1  9FA0A1            DEFB #9F,#A0,#A1
C3E4  A2FF              DEFB #A2,#FF
C3E6  000000      LC3E6 DEFB #00,#00,#00
C3E9  000000            DEFB #00,#00,#00
C3EC  000000            DEFB #00,#00,#00
C3EF  1D2121            DEFB #1D,"!","!"
C3F2  21211D            DEFB "!","!",#1D
C3F5  0000C6            DEFB #00,#00, #C6
C3F8  29292F            DEFB ")",")","/"
C3FB  C92900            DEFB #C9, ")",#00
C3FE  003A              DEFB #00,":"
```

Если Вы внимательно изучите приведенный дисассемблированный код, то достаточно быстро поймете, что это не вершина программирования в машинных кодах. Хотя следует признать, что подход, примененный в этой программе достаточно оригинален. Я не буду подробно излагать здесь описание данной процедуры, а лишь немного расскажу о ней, чтобы Вы имели обшее представление о принципах ее работы.

После необходимых подготовительных процедур осуществляется печать текстовой информации. Причем любопытен сам метод формирования этой текстовой информации. Данный текст сформирован с помощью UDG графических символов пользователя. Именно в них содержится надпись CRACKED BY BILL GILBERT. Это объясняет и необычно мелкий шрифт данного сообщения. Кроме того, данный прием позволяет обойти попытки будущих взломщиков найти, где находится этот текст с помощью средств поиска заданной символьной строки, которые есть в любой отладочной программе.

Теперь Вы имеете доступ к изменению данного сообщения (разумеется, исключительно из чистого любопытства). Причем, следует отметить, что банки UDG символов не переносятся в верхние области памяти, а всего лишь изменяется значение системной переменной, указывающей на месторасположение этих символов.

После распечатки текста идут процедуры вычерчивания линий. Поскольку это прямоугольная рамка, то линий достаточно 4. Эти процедуры легко обнаружить, как четыре почти однотипных блока.

Как видите, нет ничего сложного. Я надеюсь, что приведенная информация развеет миф о надежности защиты Билла Гилберта. Хочется также надеяться, что наши уважаемые читатели не будут злоупотреблять полученной информацией (об этом мы уже писали в Главе 2 первого тома).

В заключение надо добавить, что здесь рассмотрен лишь один из по меньшей мере трех известных автору способов защиты информации, практикуемых Биллом Гилбертом. Практически аналогичные приемы применены во вскрытых им EXOLON, DEATH WISH 3 и др. А в программах EAGLES NEST и SAS, вскрытых этим хаккером, реализован уже совершенно иной принцип защиты, несмотря на схожесть картинки с сообщением. К более подробному рассмотрению типов защиты программ у Била Гилберта мы еще вернемся.

Глава 4. Изучение блоков в машинных кодах.

4.1 Введение.

Как Вы уже вероятно догадались, данный раздел посвящен вскрытию блоков, целиком записанных в машинном коде. Наши читатели по-видимому понимают, что за изучением блоков машинного кода последует самостоятельное программирование в машинном коде, а это уже приципиально новый уровень Вашей работы с компьютером. Только поднявшись до этого уровня Вы сможете полностью почувствовать всю мощь и быстродействие Вашей машины, только здесь Вы сможете полностью использовать все ее возможности.

При изучении программирования в машинных кодах, вам обязательно захочется исследовать какие-либо фирменные программные разработки, чтобы узнать новые приемы в программировании и обогатить свой арсенал. Но большинство программ имеют

высококлассную защиту, обойти которую не так просто.

А вот еще одна более реальная ситуация. Предположим, Вы открываете "ZX-РЕВЮ" или другую специальную литературу и видите POKES, т. е. информацию о том, как ввести в игру бессмертие, изменить количеству оружия и пр. Но во многихслучаях первые попытки ввести POKES в программу ничего не дают.

Почему? Да потому, что большинство программ, в которые приятно поиграть на досуге, имеют высококлассные загрузчики в машинных кодах, которые и запускают загруженную программу. А Ваши попытки изменить содержимое ячеек, в которые впоследствии будет загружаться программа, естественно, ни к чему не приведут. И в этом случае нельзя винить авторов, давших POKES. Они изложили всю необходимую информацию, а теперь вашей задачей является правильно ею воспользоваться.

Данная статья имеет среди прочих и цель научить вас, уважаемые читатели, самостоятельно пользоваться необходимой информацией как для изучения новых приемов программирования, так и для установки необходимых POKES. Причем последний вопрос настолько актуален сейчас, что мы рассмотрим два примера предлагаемых POKES непосредственно из "ZX-РЕВЮ".

4.2 Адаптация фирменных программ под индивидуальный вкус.

4.2.1 Адаптация программы GREEN BERET.

Если Вы внимательно читаете "ZX-РЕВЮ", то вам должно быть известно, что такое POKES и для чего они применяются. Однако, если Вы внимательно изучите таблицы POKES, то обратите внимание, что в большинстве случаев предложено лишь значение ячейки памяти и ее содержимое. Собственно говоря, больше Вам ничего знать и не понадобится за одним исключением - необходимо знать как осуществить изменение содержимого указанных ячеек памяти на необходимые значения.

Проблема эта возникает в связи с тем, что почти все хорошие программы имеют загрузчик в машинных кодах и с помощью функции Бейсика POKE Вам ничего достичь не удастся. Не всегда помогает и COPY COPY, поскольку не все программы имеют начальный адрес загрузки 23896, а чтобы узнать этот начальный адрес опять-таки необходимо изучать загрузчик в кодах. Конечно, хорошо бы иметь DISK-MONITOR, позволяющий прервать работу программы и, осуществив необходимые изменения, вновь возвратиться в нее. Однако не у всех читателей он есть и не все собираются в перспективе его устанавливать.

Итак, рассмотрим в качестве примера программу GREEN BERET, одну из лучших "стрелялок" для "ZX SPECTRUM", которая несмотря на некоторую политическую окраску достаточно популярна и у нас в стране. Однако многим читателям явно не хватает предоставленных игрой трех жизней и трех выстрелов из огнемета. Казалось бы, если изменить хотя бы один из этих параметров, то игру удастся одолеть без труда, не говоря уже о том случае, когда удастся изменить их оба сразу. Причем, что самое любопытное - вся необходимая для этой цели информация есть в "ZX-РЕВЮ" (см. N7,8,10 - 1991). Остается лишь изменить содержимое указанных там ячеек, но именно здесь и возникает проблема, поскольку в программе применен собственный загрузчик в машинных кодах, который и осуществляет потом загрузку и запуск программы.

Для того, чтобы правильно вести работу, Вам в первую очередь понадобится узнать структуру всей программы. Это можно осуществить, загрузив ее в один из копировщиков, например ZX COPY 87, TF COPY и т. д. Информация для GREEN BERET, полученная таким образом на копировщике ZX COPY 87:

```
01 GREENBERET  BASIC        10        67
02                                     67
03 T.P.15006   CODE      33475       794
04                                    794
05                                     17
06                                   6912
07                                  10303
08                                  30832
```

Здесь Вы видите, что небольшая Бейсик-программа загружает и запускает загрузчик в машинных кодах, который и осуществляет дальнейшую загрузку файлов, а в конце концов их запуск. Причем, как видим, файлы, загружавшиеся после загрузчика, идут без хэдера, т. е. правильная загрузка их без "родного" загрузчика практически невозможна, поскольку именно он знает адрес памяти, с которого необходимо осуществить загрузку.

Любопытна судьба блока кодов длиной 17 байтов, идущего сразу после загрузчика. Если Вы внимательно изучите программу-загрузчик, то достаточно быстро поймете, что этот коротенький файл в перспективе затирается более мощным программным блоком. Т.е. фактически для работы программы он не нужен. Кроме того, если Вы обратили внимание, он загружается вызовом специальной подпрограммы, загружаемой вместе с загрузчиком. Я думаю, что это не случайно, как не случаен и тот факт, что длина этого блока совладает с видимой длиной "хэдера". По моим предположениям этот блок служил для защиты от копирования, поскольку некоторые версии копировщиков принимали его за "хэдер", после чего не могли правильно интерпретировать его содержимое и копирование становилось невозможным. Однако появившиеся в последние годы польские копировщики ZX COPY 87, TF COPY, TF COPY-2 свели на нет данный метод защиты, отголоски которого мы можем наблюдать в этой программе, однако вернемся к ее более подробному исследованию, поскольку именно оно поможет нам осуществить желаемые изменения.

Ниже представлен листинг Бейсика данной программы. (Сразу следует оговориться, что версий программы GREEN BERET, распространенных в нашей стране по-видимому несколько. Здесь рассматривается версия, вскрытая TOMI & DALI. Об этом свидетельствует надпись слева на картинке CRACKED BY TOMI & DALI.

GREEN BERET LLIST

```
10 BORDER 0: PAPER 0: INK 0: CLEAR 65535
20 LOAD ""CODE: RANDOMIZE USR 34132
```

Как видим, данная программа на Бейсике достаточно проста и не имеет никакой защиты. Это может объясняться тем, что используется загрузчик в машинных кодах и авторы настолько уверены в психологическом барьере, отпугивающем пользователей от машинных кодов, что не потрудились даже установить какую-либо защиту.

С другой стороны, возможно, что защита БЕЙСИКа все же была, но ее уже снял кто-то из взломщиков.

Наиболее ценной информацией, которую мы почерпнули из данной Бейсик-программы является то, что загрузчик в машинных кодах запускается с адреса 34132.

Рассмотрим теперь дисассемблер загрузчика, начиная со стартового адреса. Необходимо сразу предупредить читателя, что встроенный загрузчик не рассматривается здесь как таковой, а исследуется лишь как подпрограмма, к которой осуществляется обращение. Нам необходимо знать, в какой последовательности загружаются программные файлы и благодаря этой информации найти в программе место, с которого осуществляется запуск программы на выполнение.

Правильная адаптация программы и предполагает именно проведение всей загрузки, а перед самим запуском внесение изменений в машинный код, т.е. исполнение POKES.

На этом пути Вас ждут многие проблемы, но основные среди них - поиск точки запуска программы и правильное внесение изменений, сопряженное с программированием в машинном коде.

Программа-загрузчик использует для своей работы две процедуры - встроенную в ПЗУ и загружаемую в компьютер. Имея перед собой информацию, полученную из копировщика, Вы легко разберетесь, как работает программа.

Фальшхэдер загружается специальной процедурой программы загрузчика. Больше данная процедура нигде не используется. За это отвечают строки 34147 - 34157.

Как видим, данная процедура мало чем отличается от стандартной. Она загружает 17 байтов, начиная с ячейки памяти 36864, о чем свидетельствуют значения, загружаемые в регистры DE и IX соответственно.

Следующий блок программы 34160 - 34174 осуществляет загрузку экрана. Об этом свидетельствует длина файла 6912 и начало загрузки с адреса 16384.

После этого исполнение программы "передается" на адрес 33639 (информация о дисассемблере данной области приведена ниже).

Здесь осуществляется загрузка третьего и четвертого блоков кодов с использованием встроенной процедуры, расположенной в ПЗУ по адресу 1366.

Далее в строках 33665-33670 изменяется содержимое одной из ячеек памяти, после чего управление возвращается на 34180.

После включения прерывания командой IM 1 осуществляется корректировка содержимого памяти, т.е. перенос содержимого ячеек из одной области в другую. Это осуществляется с использованием команды LDIR. по всей видимости, это тоже один из видов защиты.

Далее идут команды изменения содержимого отдельных ячеек памяти и, наконец, переход JP 32768.

Это и есть переход в программу для запуска игры, поскольку все блоки кодов уже загружены, и кроме этого осуществлены необходимые защитные корректировки. Здесь этот безусловный переход осуществляется в "новую" область, т.е. в один из блоков загруженной программы. Однако, не всегда это может оказаться так. Чтобы быть уверенным в достоверности полученной информации, Вам необходимо проверить свое предположение на практике. Проверка показала, что высказанное здесь предположение с успехом подтвердилось.

```
34132 F3              DI
34133 31FFFF          LD SP,65535
34136 310058          LD HL, 22528
34139 010300          LD BC, 00003
34148 3600      L855E LD (HL),#00
34144 23              INC HL
34145 10FB            DJNZ L855E
34147 37              SCF
34148 3EFF            LD A,#FF
34150 DD210090        LD IX,36864
34154 111100          LD DE,00017
34157 CDE684          CALL 34082
34160 DD210040        LD IX,16384
34164 37              SCF
34165 11001B          LD DE,06912
34168 215884          LD HL,33860
34171 225684          LD (33878),HL
34174 CD5605          CALL 01366
34177 C36783          JP 33639
34180 B7              OR A
34181 00              NOP
34182 00              NOP
34183 00              NOP
34184 ED47            LD I,A
34186 215827          LD HL,10072
34189 D9              EXX
34190 FD213A5C        LD IY,23610
34194 ED56            IM 1
34196 2I4083          LD HL,33600
34199 1141S3          LD DE,22601
34202 015402          LD BC,00596
34205 3600            LD (HL),#00
34207 EDB0            LDIR
34209 AF              XOR A
34210 320A80          LD (32778),A
34213 320B80          LD (32779),A
34216 C30080          JP 32768
34219 D1 POP          DE
34220 7C LD           A,H
34221 FE01            CP #01
```

```
34223 F5              PUSH AF
34224 AF              XOR A
34225 E638            AND #38
34227 0F              RRCA
3422B 0F              RRCA
34229 0F              RRCA
34230 D3FE            OUT (#FE),A
23232 F1              POP AF
34233 D8              RET C
34234 C34083          JP 33600
34237 14              INC D
34238 282A            JR Z,34282
34240 1EFF            LD E, #FF
34242 1A              LD A,(DE)
34243 1000            DJNZ L85C5
34245 5B        L85C3 LD E,E
34246 3F              CCF
34247 2810            R Z,L85D9
34249 00              NOP
34250 88              ADC A,B
34251 1F              RRA
34252 4E              LD C,(HL)
34253 00              NOP
34254 00              NOP
34255 00              NOP
3425E 27              DAA
34257 23              INC HL
34258 00              NOP
34259 00              NOP
34260 00              NOP
34261 00              NOP
34262 00              NOP
34263 00              NOP
34264 D0              NOP
34265 00        L85D9 NOP
34266 00              NOP
33639 DD21005B        LD IX,23296
33643 113F28          LD DE,10303
33646 37              SCF
33647 3EFF            LD A,#FF
33649 CD5605          CALL 01366
33652 DD21DD85        LD IX,34269
33656 11147A          LD DE,31252
33659 37              SCF
33660 3EFF            LD A,#FF
33662 CD5605          CALL 01366
33655 3E00            LD A,#00
33667 323385          LD (34099),A
33670 3A3385          LD A,(34099)
33673 C38485          JP 34180
```

Ниже приведены изменения, которые необходимо сделать, чтобы осуществить одновременное изменение количества жизней и оружия. Рассмотрим эту информацию более подробно.

```
34216 03CB85          JP L85CD
34219 D1              POP DE
34220 7C              LD A,H
34221 FE01            CP #01
34223 F5              PUSH AF
34224 AF              XOR A
34225 E638            AND #38
34237 0F              RRCA
34228 0F              RRCA
34229 0F              RRCA
34230 D3FE            OUT (#FE),A
```

```
34232 F1              POP AF
34233 D6              RET C
34234 C340B3          JP 33600
34237 14              INC D
342338 282A           JR Z,34282
34240 1EFF            LD E,#FF
34242 1A              LD A,(DE)
34243 1000            DJNZ L85C5
34245 5E        L85C5 LD E,E
34246 3F              CCF
34247 2810            JR Z,L85D9
34E99 00              NOP
34250 88              ADC A,B
34251 1F              RRA
34252 4E              LD C,(HL)
34253 21ECB6    L85CD LD HL,46828
34256 77              LD (HL),A
34257 23              INC HL
34256 77              LD (HL),A
34259 23              INC HL
34260 77              LD (HL),A
34261 215CA4          LD HL,42076
34264 77              LD (HL),A
34265 C30080    L85D9 JP 32766
34268 85              ADD A,L
34269 00              NOP
```

Как нам уже известно, строка 34216 осуществляет запуск игры, т.е. к этому моменту загрузка закончилась и закончился также необходимый перенос информации в нужные ячейки командой LDIR. Это именно та точка, которая нужна для осуществления необходимых изменений. Теперь будем рассуждать с точки зрения рядового пользователя компьютером. Если по адресу 34216 осуществляется запуск игры, то значит команды, размещенные в ячейках 34219-34252 просто не нужны. Однако, с другой стороны, весьма маловероятно, что они являются "мусором". Не разбираясь глубоко в принципе работы программы, можно предположить, что они используются загрузившейся программой в процессе ее работы, либо же служат, как переменные загрузчика. Такие логические рассуждения необходимы для того, чтобы безошибочно определить место размещения будущей специальной подпрограммы, изменяющей содержимое ячеек памяти. Как видим, наиболее приемлемым местом является область 34253-34268, поскольку здесь есть немного "пустого" места и мы ножем не опасаться, что чему-либо помешаем.

Верхний предел связан с тем, что с адреса 34269 начинается загрузка блока машинных кодов, - информация, которую Вы могли узнать, внимательно ознакомившись с загрузчиком. Теперь нам необходимо достаточно экономно расходовать память при составлении своей процедуры изменения, чтобы уложиться в отведенное пространство 13 байтов.

Как вам уже вероятно известно, в этой программе необходимо внести следующие изменения:

- для бесконечного оружия

```
46328,0
46329,0
46330,0
```

- для изменения количества жизней

```
42076,0
```

Машинный код, который выполнит необходимые изменения. Вы можете видеть в строках 34253 - 34265.

Таким образом, в том месте, где загрузчик должен запускать игру, мы поставили "жучка", который запускает процедуру для изменения количества жизней и оружия, расположенную по адресу 34253. А после осуществления необходимых изменений в ячейках памяти процедура сама запускает игру на исполнение.

После таких изменений игра становится даже не интересной - Вы можете дойти до

конца буквально за 10 минут. Поэтому мы рекомендуем Вам просто изменить количество выстрелов из огнемета - тогда в игру станет играть чуть легче, но не настолько, чтобы быстро потерять к ней интерес.

Для тех, кто решит сохранить данные изменения, рекомендуем записать информацию на магнитофон, подав команду:

```
SAVE "G.B.CORR." CODE 33475,794
```

Для тех читателей, которые не имеют ни малейшего представления о машинных кодах Z80 и не желают их изучать, Я предлагаю измененный вариант Бейсик-загрузчика, который перед запуском программы в кодах вносит в нее все вышеописанные изменения.

```
10 BORDER 0: PAPER 0: INK 0:CLEAR 65535
20 LOAD ""CODE
30 POKE 34217,205: POKE 34218,133
40 FOR i=34253 TO 34267
50 READ A: POKE I,A
60 NEXT I
70 DATA 33,236,182,119,35,119,33,92,164,119,195,0,128
80 RANDOMIZE USR 34132
```

4.2.2. Новые возможности программы "RENEGADE".

Если Вы внимательно изучили содержимое предыдущего раздела и достаточно подробно разобрались с предложенной вашему вниманию программой, то теперь должны достаточно хорошо разбираться в подобного рода вопросах, поскольку программа GREEN BERET имеет достаточно сложный загрузчик. Однако, возможно, что не у всех хватило терпения и желания подробно изучить предложенное выше описание и потому для тех, кто просто пролистал материал предыдущего раздела, мы предлагаем разобраться с более простой программой RENEGADE. (Конечно, слово "простой" не относится к программе, а только к ее загрузчику).

В этой программе, как и в GREEN BERET не требуется много размышлять, но эта игра достаточно популярна. Здесь так же, как и в GREEN BERET количества жизней, предложенных вначале, бывает недостаточно для прохождения программы до конца.

Бейсик-загрузчик имеет вид:

```
10 POKE 23624,71: POKE 23693,71:
      CLS: LOAD ""CODE: LOAD "SCREEN$:
      RANDOMIZE USR 64000
```

Запуск машиннокодового загрузчика выполняется, как видите, с адреса 64000. Посмотрим, что там содержится.

```
64000 DD21005B          LD IX,23296
64004 11009F            LD DE,40704
64007 37                SCF
64008 3EFF              LD A,#FF
64010 CD5605            CALL 01366
64013 31FFFF            LD SP,65535
61016 C3CB5C            JP 23755
64019 32485C            LD (23624),A
64022 CD6B0D            CALL 03435
64025 21003D            LD HL,15616
61028 113000            LD DE,00048
64031 00                NOP
64032 00                NOP
```

Если Вы внимательно изучите и эту короткую программу, то достаточно быстро поймете, что в адресах 64000-64010 осуществляется непосредственная загрузка главного блока (как Вам уже вероятно известно, в программе RENEGADE загружаются два блока в машинных кодах). Однако нас сейчас интересует только первый, являющийся

машиннокодовым загрузчиком.

Почти сразу после загрузки главного блока осуществляется безусловный переход по команде: JP 23755.

Чтобы осуществить необходимые изменения, нам необходимо воспользоваться данным переходом. Вместо того, чтобы переходить на адрес 23755, мы перейден на адрес 64031, где разместим программу, осуществляющую необходимые изменения, и, осуществив их, перейдем на адрес 23755.

Измененный загрузчик для RENEGADE, набранный в ассемблере "EDITAS-48".

```
10          ORG 64000
20          LD IX,23296
30          LD DE,40704
40          SCF
50          LD A,255
60          CALL 1366
70          LD SP,65535
80          JP NEW
90          LD (23624),A
100         CALL 03435
110         LD HL,15616
120         LD DE,0048
130   NEW   LD HL,41047
140         LD A,162
150         LD (HL),A
160         LD HL,30301
170         LD A,195
180         LD (HL),A
190         JP 23755
200         END
```

Перед вами дисассемблер измененного загрузчика. Эта версия проверена автором и вполне работоспособна.

ПРИМЕЧАНИЕ.

В "ZX-РЕВЮ" N7,8 за 1991г. была дана информация об изменении количества жизней и времени в программе RENEGADE. Однако приведенная там БЕЙСИК-программа не работает с имеющейся у автора версией программы. Для тех, у кого тоже есть такие проблемы дается другой вариант адаптации. Подчеркнуты те байты, которые нуждаются в изменении.

```
64000 DD 21 00 5B   ]!.[
64004 11 00 9F 37   ...7
64008 3E FF CD 56   >.MV
64012 05 31 FF FF   .1..
64016 C3 1F FA 32   C.z2
64020 48 5C CD 6B   H\MK
64024 0D 21 00 3D   .!.!
64028 11 30 00 21   .0.!
64032 57 A0 3E B6   W >6
64036 77 21 5D 76   w!]v
64040 3E C3 77 C3   >CvC
54044 CB 5C 00 00   K\..
64046 00 00 00 00   ....
64052 00 00 00 00   ....
64056 00 00 00 00   ....
64060 00 00 00 00   ....
64064 00 00 00 00   ....
64068 00 00 00 00   ....
64072 00 00 00 00   ....
64076 00 00 00 00   ....
```

Новый же БЕЙСИК-загрузчик, способный внести эти изменения выглядит так:

RENEGADE NEW LOADER LLIST

```
10 POKE 23624,71: POKE 23693,71:CLS: LOAD ""CODE: LOAD ""SCREEN$
20 POKE 64017,31: POKE 64018,250
30 FOR I = 64031 TO 64045
40 READ A: POKE I, A
50 NEXT I
60 DATA 33,87,160,62,182,119,33,93,118,62,195,119,195,203,92
100 RANDOMIZE USR 64000
```

Том 3. Методы известных взломщиков компьютерных программ к ZX SPECTRUM.

Введение.

Эта книга написана для тех, кто внимательно изучил информацию первого и второго тома и теперь желают расширить свои знания путем исследования методов известных хаккеров.

Начало взлома компьютерных программ к "ZX SPECTRUM" относится приблизительно к 1986-1987 г.г. По всей видимости, это связано с тем, что фирмы-изготовители программного обеспечения с этого периода начали интенсивно защищать свои продукты от несанкционированного просмотра. Это, в свою очередь, было связано с существенными прорывами в технике программирования для даного компьютера и, кроме того, к такому поведению фирмы подтолкнуло широкое распространение самого компьютера.

Комментарий "ИНФОРКОМА"

Конечно же техника защиты программ, как и техника их взлома начали разрабатываться раньше. Можно сказать, что только в 1982 году выпускались программы без защиты, но и компьютеров в то время было еще очень немного.

В 1983 г. техника программирования была еще недостаточно развита для серьезных защит и, как правило, программы защищались только на уровне БЕЙСИК-загрузчика (совмещение цвета, внедрение управляющих кодов, искажение содержимого некоторых системных переменных, защита от MERGE).

1984 год стал годом разгула средств защиты. Появились такие варварские изобретения, как "стукалка", "спидлок", "лензлок". Вся мерзость их состояла в том, что не то, что скопировать программу было невозможно, ее и загрузить-то было трудно (в случае "лензлока" с загрузкой не было проблем, но были проблемы с запуском). Массовые жалобы покупателей и отказ от приобретения программ у фирм, запятнавших себя такими приемами стали коммерческой проблемой уже в 1985 году, когда убытки от средств защиты превысили убытки от пиратства.

Автор статьи относит начало появлений средств защиты к 1986 году и мы с ним согласны, если рассматривать квалифицированные средства защиты, построенные на изощренной логике программирования, а не на искажении параметров загрузки, что очень плохо отражается на самой загрузке.

Разумеется, техника взлома, техника защиты и техника программирования совершенствовались одновременно. В частности, если сравнить по графике игры 1986 года и 1989, то без сомнения в большинстве случаев Вы отдадите предпочтение последним. То же самое можно сказать и об уровне защищенности компьютерных программ.

Однако то же самое мы можем сказать и о технике взлома. В самом деле, фирмы совершенствуют свою защиту и, чтобы вскрыть ту или иную программу, приходится прикладывать максимум изобретательности.

Среди "хаккеров" наибольшую известность приобрел BILL GILBERT. Можно с уверенностью сказать, что программы взломанные им есть у каждого обладателя

компьютера "ZX SPECTRUM". Достаточно сложно судить о том, что заставило этого человека заниматься такой деятельностью, однако судя по распространению программ вскрытых Билом Гилбертом можно предположить следующее.

В нашу страну программы попадают уже пройдя через руки этого "хаккера", а поскольку наибольший приток программ сюда наблюдается из Польши, то не исключено, что он проживает именно там. Не исключено так же, что человека с таким именем не существует, а хаккер в своей деятельности использует псевдоним.

однако это всего лишь гипотеза, и если кому-либо известна достоверная информация о работе "хаккера", то я надеюсь, он поделится ей на страницах "ZX-РЕВЮ".

Комментарий "ИНФОРКОМа"

Не имея достоверной информации о Б. Гилберте, мы должны, тем не менее, отметить тот факт, что вопросы пиратского распространения копий программного обеспечения несколько лет назад широко обсуждались на страницах зарубежных журналов, посвященных Синклер-совместимым машинам. Сейчас острота этой проблемы спала, поскольку лидирующее место на рынке заняли другие компьютеры.

Исследования западных журналистов показали, что центром международного пиратства является Голландия, где по-видимому и следовало бы искать Б. Гилберта. Оттуда поток разделялся на три ветви. За океан в Южную Америку, в частности в Бразилию, где ситуация очень напоминала нашу - очень дорогие компьютеры при очень дешевых программах и при полном отсутствии элементарной защиты авторских прав. Второй поток - на юг в Израиль, далее куда угодно, и третий поток - на Восточную Европу, в частности в Польшу, Венгрию и Югославию, а из Польши - к нам.

Как показали журналистские расследования, эта деятельность - отнюдь не невинные развлечения энтузиаста "хаккера". В нее были вовлечены колоссальные средства, измерявшиеся миллионами долларов. Достаточно сказать, что во многих случаях пиратский тираж программ из Голландии в десяток раз превосходил основной тираж в Англии. Организация имела мощные сети и неоднократно начинала продавать вскрытые программы за несколько месяцев до того, как в Англии выходил первый защищенный оригинал.

Достаточно сложно выделить "хаккеров", которые бы занимали второе место после Билла Гилберта. Фактом является то, что никому не удалось еще подняться на его уровень. Тем не менее, в коллекции автора второе место занимает "PEGAS SOFTWARE". Несмотря на то, что никакой информации о месте нахождения данной "компании" нет. Это не помешает нам разобрать приемы, используемые ими в работе.

Ступенькой ниже идут отдельные программы, в которых окозалось сообщение о том, что их кто-то взломал. К таким хаккерам мы можем отнести "ROBI CRACKING SERVISE", "TOMI & DALI" и др.

Автор считает своим приятным долгом выразить благодарность тов. Кириллову С.В. из пос. Мурмаши Мурманской обл., предоставившему очень интересную информацию из журнала "Bajtek", касавшуюся данного вопроса.

ГЛАВА 1.

Техника защиты совершенствуется постоянно. Поэтому Глава 4 первого тома к моменту прочтения Вами изложенного в ней материала по всей видимости безвозвратно устарела. Именно по этой причине мы будем говорить о современных методах защиты как можно более часто, стараясь, чтобы изложенная информация сохраняла свою актуальность.

Итак, сегодня мы рассмотрим некоторые приемы, которые были взяты на вооружение фирмами-изготовителями программного обеспечения. Для начала исследуем очень любопытный прием, разработанный фирмой ULTIMATE.

Вы знаете, что "SPECTRUM" имеет "встроенные часы", роль которых выполняет системная переменная FRAMES. Ее значение увеличивается каждую 1/50 сек. Таким образом, мы можем контролировать процессы, протекающие строго определенное время. Именно на этом принципе основана одна из защит фирмы ULTIMATE (изготовитель

программ ATIC ATAC, SABRE WOLF, KNIGHTLORE, ALIEN 6, PENTAGRAM, NIGHT SHADE и мн. др.)

Поскольку при загрузке программ система прерываний отключается, то мы можем контролировать - осуществлялось или нет вмешательство в данную программу, следя за состоянием системной переменной FRAMES. В самом деле, для осуществления вмешательства необходима остановка программы, которая, что вполне естественно, приведет к несовпадению контрольных значений в системной переменнной FRAMES.

Рассмотрим, как это осуществляется на практике. Обычно, после достаточно простого загрузчика на БЕЙСИКе на ленте идет сама программа, после которой следуют три небольших блока, которые собственно и осуществляют защиту. Первый - это однобайтовый код инструкции JP(HL), осуществляющий старт необходимых процедур, второй блок программной защиты состоит из нескольких байтов - он осуществляет декодирование всей программы. (Вопросы кодирования и декодирования программ будут подробно рассмотрены в следующем разделе), и, наконец, третий блок имеет длину два байта и загружается в область 23627, т.е. в системную переменную FRAMES. Загружаемые значения представляют собой контрольную сумму, по величине которой и осуществляется проверка: останавливалась исходная программа или нет. Обычно этим занимается специальная процедура в машинных кодах, которая при обнаружении взлома обнуляет память компьютера.

Данная защита не получила широкого распространения и обычно ее можно обнаружить лишь в программах фирмы ULTIMATE. Это объясняется тем, что ее можно достаточно легко свести на нет, если придерживаться специального принципа при взломе.

Вмешательство в программу такого типа весьма просто. Достаточно загрузить все блоки, за исключением последнего, а после осуществления просмотра или проведения в ней определенных изменений (например, вписание необходимых POKES) достаточно ввести:

LOAD "" CODE:RANDOMIZE USR ADRESS(24064)

Эту команду необходимо обязательно ввести с клавиатуры одной строкой, разделяя инструкции двоеточием, чтобы вложиться в интервал времени соответствующий новому значению системной переменной FRAMES.

Поскольку в большинстве вышеприведенных программ фирмы ULTIMATE адресом старта является 24064, то именно он приведен в качестве примера. Однако, в каждом конкретном случае значение может быть другим. Необходимо каждый раз перед вводом этой серии команд уточнять его и вносить необходимые изменения.

После такого пояснения мы надеемся, что у читателя не возникнет затруднения при вводе POKES для изменения игр фирмы ULTIMATE.

(Продолжение следует)

МОНИТОР 48 - новые возможности.

Читателям, которые работают с машинными кодами Z80, наверняка приходилось использовать пакет программ фирмы PICTURESQUE "EDITAS/MONITOR". Существуют две версии данного пакета программ, рассчитанные на работу в компьютерах с оперативной памятью 16 и 48 килобайт. Они имеют соответственно названия MONITOR 16 и MONITOR 48. Естественно, программа, рассчитанная на 16 килобайтную машину, подходит к компьютеру, имеющему объем оперативной памяти 48 Кбайт, поэтому в отечественных моделях возможно и желательно использование обеих типов данных программ, т.к. вследствие своего расположения в низких адресах оперативной памяти программа MONITOR 16 может применяться для дисассемблирования блоков, занимающих верхние области памяти.

Однако было бы по меньшей мере наивно предполагать, что в программе MONITOR 48 существенным отличием является лишь новое расположение в оперативной памяти компьютера. Эта версия является одним из лучших отладчиков для программ в нашинных кодах. Она, безусловно, имеет меньше режимов работы, чем версии программы MONS из

пакета DEVPAC, однако это компенсировано разработчиками введением дополнительного режима, обеспечивающего работу программы через отладчик. Рассмотрим его более подробно.

Этот режим вызывается после нажатия клавиши T <ENTER> и служит для пошаговой отладки программы, написанной в машинном коде. Назовем этот режим режимом трассирования (T - TRACE).

После этого Вы попадаете в основное меню данного режима. На экране Вы можете видеть следующую картину (см. рис. 1):

- два дисассемблированных оператора текущего программного адреса;
- содержимое всех регистров микропроцессора в т.ч. и альтернативных;
- состояние всех флагов регистра F, в том числе и альтернативного.
- содержимое нескольких ячеек вершины машинного стека;
- содержимое определенных ячеек памяти, которые заранее выбраны пользователем.

Следует отметить, что содержимое основных регистров процессора отделено от содержимого альтернативных регистров. Кроме этого, напротив каждой основной регистровой пары приведено содержимое ячеек памяти, на которые указывает содержимое данной пары.

```
STEP
2D28     AF       LD C,A
IR       3F50               STACK
          SZ H PNC          0009
A'F'   >0001000100         000D
B'C'    1721               10ED
D'E'    369B               F39C
H'L'    2758              *3365
          SZ H PNC
AF      0F01010100
BC      F70F (BC) C3 45 F1 1A CD
DE      61C4 (DE) 00 00 0F F7 00
HL      2D2B (HL) FD 21 3A 5C AF

IX      03D4
IY      5C3A
SP      E8D6
PC      2D28
M000   F3 AF 11 FF FF C3 CB 11 2A
```

Экран режима "T" программы "MONITOR 48" при входе в него сразу после начала работы отладчика.

Рис. 1

Все это стандартная информация, которую можно получить и в MONS за исключением, пожалуй, достаточно интересной информации о содержимом стека. Фактически этот режим создан исключительно для изучения работы программ в машинных кодах. Здесь можно изучать пошаговое выполнение программы, поскольку после нажатия клавиши ENTER происходит выполнение текущей команды.

Теперь настало время рассмотреть более подробно работу программы в данном режиме. Я предполагаю, что читатель, заинтересовавшийся этим материалом уже имеет необходимое представление о машинных кодах и поэтому не считаю нужным давать здесь какую-либо информацию о процессоре Z80. Если Вы хотите получить эту информацию, рекомендую трехтомник "ИНФОРКОМа".

Итак, вы уже вероятно поняли, что информация, появившаяся на экране после перехода в режим "T" отражает текущее состояние микропроцессорной системы. Естественно, она будет изменяться после выполнения компьютером очередной команды. Сверху вы можете видеть две дисассемблированные команды, начиная с адреса, указанного в регистре PC микропроцессора. Вы сможете изменить этот адрес для того, чтобы

осуществить выполнение программы оттуда, откуда вам надо. Но об этой чуть позже, а теперь краткая сводка команд, использующихся в этом режиме.

S - установка программного счетчика (регистра PC) на заданный шестнадцатиричный адрес.

M - распечатка в нижней строке экрана содержимого ячеек памяти, начиная со значения, заданного после M. Всего можно увидеть 9 значений. В момент первого запуска данного режима программа автоматически устанавливает значение M, указывающее на нулевую ячейку памяти.

Курсорные клавиши - изменяют местоположение стрелки от одной регистровой пары к другой. Таким образом, можно указать на любой регистр микропроцессора, за исключением регистра программного адреса PC. Это необходимо, чтобы изменять содержимое данных регистров, а как Вам уже известно, содержимое регистра PC изменяется с помощью команды S.

1 - Если Вы нажмете на 1, то в строке ввода информации (самая нижняя строка экрана, она находится под строкой информации о содержимом памяти) появится содержимое текущей регистровой пары микропроцессора, на которую указывает стрелка-курсор. Если вы хотите изменить содержимое данной регистровой пары, то наберите шестнадцатиричное число, которое бы Вы хотели видеть вместо текущего значения. Если же Вы не хотите ничего изменять, то не набирая шестнадцатиричного числа нажмите ENTER. Кроме этого, для выхода из режима можно использовать команду X.

X - если Вы набираете какую-либо команду и увидели, что набираете ее неправильно, то ввиду того, что в программе MONITOR 48 отсутствует функция DELETE. Вам необходимо использовать функцию X. После нажатия этой клавиши вы возвращаетесь в исходный режим, который был до начала набора данной команды, примечательно, что эта клавиша используется и для выхода из режима "T" в основной режим MONITORa. Поэтому будьте осторожны и внимательны при использовании этой функций в данном режиме. Но в крайнем случае не отчаивайтесь: ничто не помешает Вам вновь нажать "T" и продолжить работу.

ENTER - нажатие этой клавиши приводит к выполнению компьютером текущей программной команды. Если вы несколько раз нажали клавишу ENTER, перед этим установив программный счетчик, то можете включить два вспомогательных режима.

R - после нажатия этой клавиши и ENTER справа вверху появляется надпись SCIP TO RET. Надо признаться, что этот режим еще недостаточно хорошо исследован, однако есть предположение, что по этой команде начинается исполнение программы и при обнаружении ближайшей команды RET осуществляется возврат в MONITOR 48. Предлагаю Вам самостоятельно проверить правильность этой гипотезы и подтвердить ее или опровергнуть.

Следующий режим исследован наиболее тщательно и, на взгляд автора, является тем преимущественным фактором, который вызывает предпочтение MONITORa 48 другим отладчикам. Это так называемый режим B.

B - нажатие этой клавиши и ENTER приводит к появлению слева вверху надписи BREAKPOINT, а повторное нажатие ENTER приводит к включению режима. После B можно набрать шестнадцатиричный адрес, с которого Вы бы хотели осуществить включение режима.

Данный режим работы служит для относительно медленного контроля работы исходной программы в машинных кодах с возможностью в любое время прервать выполнение программы нажатием клавиши BREAK. Причем в данном случае медленная работа программы Вам на руку, поскольку программы в машинных кодах в нормальном режиме выполняются очень быстро, а здесь происходит замедление в 100, а может и более раз. К тому же несомненное удобство представляет возможность остановки программы с

возвратом в отладчик. Вам не придется кусать себе локти в случае зависания отлаживаемой программы

Однако, есть несколько нюансов, которые необходимо учитывать при работе. Во-первых, если Вы наберете B0000, то, быть может, Вам вновь придется загружать программу с внешнего носителя. Это необходимо учитывать в том случае, если проверяемая Вами процедура использует в своей работе верхнюю область памяти (нельзя допускать каких-либо изменений в области, где хранится программа MONITOR 48).

Во-вторых, есть одна особенность, без учета которой бывает достаточно сложно включить режим "B". В частности, он не всегда начинает работать, если вы включаете его сразу после изменения содержимого программного счетчика, используя команду S. Если это происходит, то необходимо перед нажатием "B" "прогнать" несколько шагов программы нажатием клавиши ENTER.

Безусловно данная статья не в состоянии охватить полностью всех возможностей этого режима. Однако этого вполне достаточно, чтобы дать толчок для ваших исследований, уважаемые читатели. Мы надеемся, что открыв что-нибудь новое, Вы поделитесь своей информацией на страницах "ZX-РЕВЮ". Мы также надеемся, что быть может откликнется человек, имеющий необходимое описание, сделанное фирмой производителем данной программы, чтобы мы с вами могли более полно использовать все возможности великолепной программы-отладчика машинных кодов MONITOR 48.

Алексеев А. Г.

ПРОФЕССИОНАЛЬНЫЙ ПОДХОД

"ДЕБЮТ ПРОГРАММЫ"

Продолжая разговор о повседневных задачах, начатый в одной из номеров "ZX-РЕВЮ", следует поподробнее остановиться вот на чем. Несмотря на то, что Ваши программы различаются в зависимости от конкретной задачи, практически во всех программах встречается повторяющиеся моменты. Это может быть введение русского шрифта, титульная заставка с фамилией автора (Вашей), блок кодов "ON ERROR GO TO" с механизмом его инициирования, который описан в предыдущей статье (см. ZX-РЕВЮ 5-6 за 1992 г. стр. 113), какое-то стандартное начало программы. Неплохо также предусмотреть элементарный сервис для себя, в частности упрощение сохранения программы на ленте в процессе написания и отладки. То есть речь идет о "дебюте" программы. Наверное, у многих програмистов есть свой "дебют" или стандартное начало своих программ, а также одинаковые приемы, используемые практически во всех или в большинстве своих программ.

Предлагаю вниманию начинающих достаточно универсальный "дебют" программы. Я сам уже долгое время пользуюсь этим "дебютом" при разработке своих программ и оказывается, что при составлении новой программы требуется минимум переделок. Новую программу я всегда начинаю с загрузки "дебюта", затем набиваю остальные строки программы, не отвлекаясь на "второстепенные" задачи по формированию символьного набора и т.п.

Итак, сначала текст заготовки программы, назовем ее "PROG", затем - комментарии.

```
0 REM (машинные коды)
1 GO TO 100
2 RANDOMIZE 2: GO SUB 10:CLEAR 49999: LOAD "prog"COOE 50000,1000
3 GO SUB 8: PRINT . . . (вывод на экран заставки)
4 GO SUB 20: GO TO 0
5 GO SUB 9: SAVE "PROG"LINE 2: SAVE "prog"CODE 50000,1000
6 VERIFY "PROG": VERIFY "prog" CODE: PRINT INK 9; TAB 20; "O.K." : BEEP 0.5,32: GO TO 5
7 POKE 23675,208: POKE 23676,95: RETURN : REM UDG
8 POKE 23606,208: POKE 23607,91: RETURN : REM RUS
9 POKE 23606,0: POKE 23607,60: RETURN : REM LAT
10 POKE 24746, PEEK 23670: POKE 24749,PEEK 23671: RANDOMIZE USR 24696: RETURN :REM ON ERROR
      GO TO
20 PRINT #0; TAB 5;"НАЖМИТЕ ЛЮБУЮ КЛАВИШУ": PAUSE 0: INPUT;
22 IF INKEY$="q" THEM GO TO 9999
24 RETURN
100 RANDOMIZE 3: GO SUB 10: GO SUB 7: GO SUB 8: BORDER 1: PAPER 0: INK 6: BRIGHT 0: CLS
1000 PAUSE 50: GO SUB 20
1010 PRINT AT 12, 10;"ЗВУК 1":RANDOMIZE USR 24769
1020 PAUSE 50: GO SUB 20 1030 PRINT AT 12,10;"ЗВУК 2":RANDOMIZE USR 24801
9999 BORDER 7: PAPER 7: INK 0: BRIGHT 0: INVERSE 0
```

В нулевой строке программы расположены блоки кодов, необходимые для ее работы.

О формировании нулевой строки мы подробно поговорим позже. А сейчас - о том, что же это за блоки кодов.

1. Символьный набор расположен с адреса 23760. Длина - 768 байт (по адрес 24537). Системная переменная CHARS при этом равна: 23760-256=23504. Символьный набор может быть любым, по вашему вкусу. Я использую русско-латинский символьный набор в кодах ASCII КОИ-7 "НС". Подробно об этом - в статье о русификации программы "МАСТЕРФАЙЛ-09" в ZX-РЕВЮ 1-2 за этот год, стр. 31. Там же программа по формированию такого символьного набора.

2. Символы UDG графики расположены с адреса 24528. Длина -168 байт (по адрес 24695). Этот набор может быть совершенно различным в зависимости от Ваших требований. Попутно замечу, что часто могут требоваться изображения стрелок влево, вправо, вниз (вверх - уже есть в символьном наборе), элемента для изображения рамок, волнистые линии и т.д. Все зависит от требований Вашей программы. Поэтому я не буду специально останавливаться на этом.

3. Блок кодов "ON ERROR GO TO" расположен с адреса 24696. Длина - 73 байта. Подробно работа этого блока была изложена в предыдущей статье в ZX-РЕВЮ N 5-6 за этот год на стр.113. Номер Бейсик-строки для перехода задается в ячейках 24746,24749.

4. SOUND 1 - так назовем коротенький блок в машинных кодах, выдающий звуковой сигнал, который может использоваться для индикации какого-либо положительного эффекта в программе (это может быть правильный ответ на вопрос или находка "клада" и т.п.). SOUND 1 расположен с адреса 24759 (по адрес 24800). Длина - 32 байта. Вызов - RANDOMIZE USR 24769.

5. SOUND 2 - аналогично предыдущему пункту - звуковой сигнал, индицирующий отрицательный эффект в программе, т.е. неправильный ответ на вопрос, ошибка в действиях игравшего и т.п. SOUND 2 расположен с адреса 24801. Длина - 32 байта. Вызов - RANDOMIZE USR 24801.

Для получения блоков кодов SOUND 1 и SOUND 2 приведена программа в конце этой статьи, а более подробно о них будет рассказано в следующем выпуске "РЕВЮ".

Конечно, это не означает, что Вы должны копировать этот порядок. Вам просто надо определиться со своими требованиями и добавить свои блоки кодов или изменить имеющиеся.

Теперь о других строках программы.

Автостарт программы происходит со строки 2. Строки 2, 3, 4 – выполняют те действия, которые необходимо сделать, чтобы обеспечить старт программы "с нуля", после включения компьютера. Иными словами, это "горячий старт". Сюда входит следующее. Предположим, что для работы Вашей программы требуется загрузка некоторого блока в машинных кодах. Пусть это будет блок "prog"CODE 50000,1000.

Тогда предварительно надо переустановить RAMTOP для резервирования места под блок кодов командой CLEAR (ADDR-1), где ADDR - адрес загрузки.

Далее (строка 3) может идти вывод сообщения об авторах программы, дате ее последней коррекции или какие-либо рекламные комментарии, а также краткие сообщения по работе программы. (Перед тем, как это сделать, скорее всего придется включить русский символьный набор.) Если Вам недостаточно одной строки для этого, то используйте GO SUB ...

Строка 4 передает управление на начало программы, обеспечивая "холодный" старт программы. В процессе отладки Вы будете пользоваться "холодным" стартом программы, подавая команду RUN. При этом нет необходимости опять загружать блок машинных кодов. В том случае, если у вас нет блока машинных кодов, который необходим для работы Вашей программы и "холодный" и "горячий" старт - это одно и то же, то исключите строки 1 и 2, а в строке 4 GO TO 0 исправьте на GO TO 100. 100 - это адрес начала непосредственного выполнения программы.

Зато, если вы в дальнейшем захотите усовершенствовать свою программу, например, "озвучить" ее с помощью музыкального редактора "WHAM", то Вы всегда сможете восстановить строки 1 и 2 для загрузки скомпилированная кодов мелодий.

Строки 5 и 6 обеспечивают вам возможность периодического сохранения программы (вместе с блоком кодов, если это необходимо) на ленте в процессе написания и отладки. Для этого надо сделать RUN 5.

Строка 7 - переключатель символов UDG-графики. Она включает набор символов UDG, размещенный в нулевой строке.

Строка 8 - включение символьного набора, который расположен в нулевой строке.

Строка 9 - выключение символьного набора нулевой строки (т.е. включение символьного набора из ПЗУ).

Строка 10 - подготовка и активизация блока кодов "ON ERROR GO TO", подробно об этом было изложено в предыдущей статье в ZX-РЕВЮ 5 -6 за этот год.

В строках 20, 30, 40, ... по 99) - располагаются недлинные подпрограммы, которые часто используются на протяжении всей программы, типа вывода таблички "нажмите любую клавишу" или последовательности операторов BEEP, играющих мелодию из нескольких нот и т.п. В общем, их можно было бы располагать и в другом месте, например в строках 6000 или 9000, но я использую именно строки с 20 по 99 просто потому, что короче набирать GO SUB 20, чем GO SUB 8000.

Сама программа начинается со строки 100. Причем в строках 100 999 располагаются следующие фрагменты. Это установка исходных параметров, когда происходит переключение блока UDG-графики на тот, который загружен с ленты, включение нового символьного набора, устанавливаются необходимые цвета PAPER, INK, BORDER и т.п. Могут задаваться массивы для переменных, присваиваются численные значения переменным и т. д.

Со строки 200 может быть расположено главное меню программы. Подробно об универсальном варианте такого меню будет рассказано в следующей статье. Это меню, в свою очередь, может передавать управление на другие "подменю", располагаемые со строк 300, 400 ... , при помощи которых управление передается на непосредственное выполнение различных фрагментов программы. Последние располагаются, начиная со строк 1000, 2000 и т.д. (Для примера, в строкая с 1000 демонстрируются звуки SOUND 1 и SOUND 2).

Остановка программы, в случае использования блока кодов "ON ERROR GO TO", возможна только благодаря "жучку" в строке 22. Нажав клавишу "Q", когда появляется табличка: "нажмите любую клавишу", переходим на строку 9999. Эта строка обеспечивает восстановление белого цвета PAPER и черного INK при остановке программы для удобства редактирования. Добавив в строку 9999: GO SUB 9, можно включать опять символьный набор Спектрума, но если Вы используете русско-латинский символьный набор "HC" в кодах КОИ-7, то это переключение не требуется. Наоборот, в этом случае вы будете одинаково хорошо видеть текст программы, печатаемый английскими буквами и текстовые сообщения в операторах PRINT, печатаемые русскими буквами. Надо только для удобочитаемости программы имена переменных набирать, используя режим CAPS LOCK, тогда они будут печататься английскими буквами. (Впрочем, может быть Вам больше понравится набирать их русскими словами? Дело вкуса.)

Теперь поговорим подробнее о нулевой строке. Вспомним, как располагается в памяти строка Бейсика. На начало первой строки программы указывает системная переменная PROG (2 байта по адресу 23655). Обычно это 23755. 2 байта занимает номер строки, (младший и старший байты здесь расположены не как обычно, сначала – старший, затем - младший) затем 2 байта занимает длина строки (длина текста плюс 1 байт для кода 13, завершающего строку), затем идет текст программы на Бейсике, затем код 13 - ENTER - конец строки.

Если набрать 1 REM , а после REM, скажем, 5 пробелов, то после ввода в память строка будет иметь вид:

```
23755 0 номер строки: 1
23756 1
23757 7 длина строки: 7
23756 0
23759 234 REM
23760 32
23761 32
23762 32
23763 32
23764 32
23765 13 <ENTER>
```

Область за REM начинается с адреса 23760 и заканчивается адресом 23764 (5 байт).

Теперь на это место можно загрузить блок кодов (конечно, если его длина не превышает 5 байт, иначе Бейсик-программа будет запорчена). Так мы делали, располагая в начальной строке программы блок кодов "ON ERROR GO TO" (см. предыдущую статью в ZX-РЕВЮ 5-6 на стр. 113). Там для этого надо было 73 байта памяти и после REM мы набирали 73 пробела.

Для размещения тех кодов, о которых говорилось выше, надо зарезервировать следующий объем памяти:

Символьный набор:	768 байт
UDG-графика:	168 байт
ON ERROR GO TO:	73 байта
SOUND 1:	32 байта
SOUND 2:	32 байта
Всего:	1073 байта

То есть после REM должно быть набрано 1073 пробела или любых других символов. Если Вы попытаетесь вручную набрать такое количество пробелов, то это займет уйму времени, к тому же, при наборе, с определенного момента времени начнет раздаваться предупредительный звуковой сигнал, еще более замедляющий работу. То есть, набрать вручную такое количество пробелов практически невозможно.

В этом случае могла бы помочь опять же программа "SUPERCODE". Под номером 84 в ней находится блок кодов "EXPAND REM" (или "REM FILL" в "SUPERCODE 2"). Этот блок кодов должен расширять область REM на величину до 9999 байт. К сожалению, этот блок кодов имеет программные ошибки, из-за чего он не работает так, как должен. Поэтому я для создания REM-области пользуюсь своей программой в машинных кодах, которую хочу предложить вниманию читателей.

Для тех, кто интересуется машинными кодами, приводится текст программы с комментариями. Если Вас это не интересует, то пропустите описание работы блока кодов.

Программу назовем так же, как и в "SUPERCODE" - "REM FILL".

Блок кодов "REM FILL"

Этот блок кодов можно загружать в любое место памяти, для примера, он загружен в буфер принтера, в адрес 23296 (#5B00). В отличие от программы N84 в "SUPERCODE", здесь длина получаемой REM-области не ограничена 9999 байтами, а ограничена только памятью компьютера, отведенной под Бейсик-программу (системной переменной RAMTOP). Кроме того, сам блок кодов почти вдвое короче, чем в "SUPERCODE". Единственное ограничение на работу блока кодов - это то, что получаемая REM-область должна быть не менее двух байт (что вряд ли кому-то может придти в голову).

Теперь - блок кодов "REM FILL", затем - комментарии.

```
5B00 016400             LD BC,#0064       (1)
5B03 2A4B5C             LD HL,(#5C4B)     (2)
5B06 09                 ADD HL,BC         (3)
5B07 224B5C             LD (#5C4B),HL     (4)
5B0A 2A555C             LD HL,(#5C55)     (5)
5B0D 09                 ADD HL,BC
5B0E 22555C             LD (#5C55),HL
5B11 2A595C             LD HL,(#5C59)     (6)
5B14 09                 ADD HL,BC
5B15 22595C             LD (#5C59),HL
5B18 2A5B5C             LD HL,(#5C5B)     (7)
5B1B 09                 ADD HL,BC
5B1C 225B5C             LD (#5C5B),HL
5B1F 8A5D5C             LD HL,(#5C5D)     (8)
5B22 09                 ADD HL, BC
5B23 225D5C             LD (#5C5D),HL
5B26 2A615C             LD HL,(#5C61)     (9)
```

```
5B29 09              ADD HL,BC
5B2A 22615C          LD (#5C61),HL
5B2D 2A635C          LD HL,(#5063)      (10)
5B30 09              ADD HL,BC
5B31 22635C          LD (#5C63),HL
5B34 DD2A535C        LD IX,(#5C53)      (11)
5B38 DDE5            PUSH IX
5B3A DD360000        LD (IX+0),#00      (12)
5B3E DD360100        LD (IX+1),#00
5B42 DD6603          LD H,(IX+3)        (13)
5B45 DD6E02          LD L,(IX+2)
5B48 E5              PUSH HL            (14)
5B49 09              ADD HL,BC          (15)
5B4A DD7403          LD (IX+3),H        (16)
5B4D DD7502          LD (IX+2),L
5B3O D1              POP DE             (17)
5B51 13              INC DE
5B52 13              INC DE
5B53 13              INC DE
5B54 E1              POP HL             (18)
5BS5 19              ADD HL,DE          (19)
5B56 C5              PUSH BC            (20)
5B57 C5              PUSH BC
5B58 EB              EX DE,HL           (21)
5B59 2A655C          LD HL,(#5C65)      (22)
5B5C E5              PUSH HL            (23)
5B5D ED52            SBC HL,DE          (24)
5B5F E5              PUSH HL            (25)
5B60 C1              POP BC
5B61 D1              POP DE             (26)
5B62 E1              POP HL
5B63 19              ADD HL,DE          (27)
5B64 22655C          LD (#5C65),HL      (28)
5B67 EB              EX DE,HL           (29)
5B68 2B              DEC HL             (30)
5B69 1B              DEC DE             (31)
5B6A EDBB            LDDR               (32)
5B6C 23              INC HL             (33)
5B6D 3600            LD (HL),#00        (34)
5B6F E5              PUSH HL            (35)
5B70 D1              POP DE
5B71 13              INC DE             (36)
5B72 C1              POP BC             (37)
5B73 0B              DEC BC
5B74 EDB0            LDIR               (38)
5B76 00              NOP                (39)
5B77 210100          LD HL,#0001        (40)
5B7A CD6E19          CALL #196E
5B7D E5              PUSH HL
5B7E 211027          LD HL,#2710
5B81 CD6E19          CALL #196E
5B84 D1              POP DE
5B65 CDE519          CALL #19E5
5B88 C9              RET
```

Комментарии к программе.

Сначала (1) в регистр BC заносится длина будущей REM-области - это та величина, на которую удлинится начальная строка Бейсик-программы, или иными словами, это то число байтов, которое добавится между текстом начальной строки и кодом 13 - <ENTER>, завершающим строку. (Для примера задано 100 байтов - то есть #0064.)

Далее следует ряд действий по изменению значений системных переменных. На величину добавляемых байтов должна увеличиться значения следующих системных

переменных:

VARS - адреса переменных Бейсика (23627 или #5C4B)
NXTLIN - адрес следующей строки Бейсик-программы (23637 или #5C55)
E_LINE - адрес выведенной команды (23641 или #5C59)
K_CUR - адрес курсора (23643 или #5C5B)
CH_ADD - адрес следующего интерпретируемого символа (23645 или #5C5D)
WORK_SP - адрес временной рабочей области (23649 или #5C61)
STK_BOT - адрес дна программируемого стека (23651 или #5C63)
STK_END - адрес начала резервной области памяти (23653 или #5C65)

Все эти системные переменные имеют длину по два байта и изменяются следующим образом. Сначала (2) в регистр HL заносится содержимое системной переменной (VARS), затем (3) к нему прибавляется число добавочных байтов из регистра BC и затем результат сложения (4) из регистра HL пересылается назад в память компьютера, в ячейку этой системной переменной.

Аналогичные действия (5)-(10) производятся и с остальными перечисленными системными переменными, кроме STK END. О ней - чуть позже.

Далее (11) в регистр IX заносится значение системной переменной PROG (23635). Теперь в IX -23755. Это число нам еще понадобится для дальнейших рассчетов, поэтому сохраним его на стеке.

Записывая 0 по адресу (PROG) и (PKOG)+1 (12) изменяем номер начальной строки программы, теперь это 0.

Затем (13) в регистры H и L побайтно заносятся числа из адресов (PROG)+2 и (PROG)+3. Теперь в регистре HL - длина начальной строки Бейсик-программы. Запомним это число на стеке (14) для дальнейших действий. Далее содержимое HL также увеличивается на величину добавляемых байтов и затем (16) новое значение длины строки из регистров H и L побайтно записывается в память.

Для дальнейших рассчетов нам надо получить адрес символа <ENTER> - конца строки. Начиная с этого места вся дальнейшая область Бейсик-системы должна быть отодвинута на величину добавляемых байтов. Для этого снимаем со стека в регистр DE бывшую длину начальной строки и увеличиваем ее на три (два байта - номер строки плюс два байта - длина строки и минус один байт - символ <ENTER>; итого 3 байта). Затем (18) снимаем со стека значение PROG и добавляем к нему (19) полученное число из регистра DE. Теперь в регистре HL - адрес символа <ENTER>. Теперь, перед дальнейшими расчетами, сохраним на стеке (20) длину добавочных байт.

Затем (21) переписываем содержимое HL в DE (это адрес символа <ENTER>), а в регистр HL загружаем (22) значение системной переменной STK END число из ячейки 23653. Бейсик-система заканчивается адресом (STK END)-1, а с адреса (STK END) начинается свободная область памяти. Сохраним (23) эту величину на стеке. Теперь (24) из HL вычтем DE. Получим длину блока, который надо отодвинуть для размещения добавочных байтов. Перепишем (25) это число через стек из HL в BC. далее (26) снимем со стека значение STK END в регистр DE и число добавочных байтов в регистр HL. сложив их (27), получим (в регистре HL) новое значение STK END. Теперь (28) его можно занести в память в таблицу системных переменных.

Далее (29), поменяв между собой содержимое регистров HL и DE, получим в HL - бывшее значение STK END, а в DE - новое значение STK END. Уменьшив HL на единицу (30), получим последний адрес Бейсик-системы, которую надо отодвинуть (путем переброски) на величину добавляемых байтов. Адрес "места назначения" перебрасываемого блока кодов получится (31) в результате уменьшения на единицу содержимого DE. В BC к этому моменту находится длина перебрасываемого блока - начиная с символа <ENTER> - конца начальной строки до величины STK END.

Для переброски блока кодов используется (32) команда LDDR, а не LDIR, так как LDDR начинает переброску со старших адресов и заканчивает младшими, что сохраняет массив

кодов при частичном наложении (см. "Программирование в машинных кодах"; М.:"ИНФОРКОМ", 1990,1992, 271 стр.).

К моменту завершения переброски массива, в регистре HL будет 23759 - адрес последнего "непереброшенного" байта. Увеличив (33) это значение на единицу, получим адрес первого "добавочного" байта. Иными словами - это адрес первого байта за REM (то есть 23760).

Для того, чтобы полученную REM-область очистить от прежних кодов - остатков переброшенной Бейсик-программы (заполнить например нулями), воспользуемся командой LDIR. Для этого сделаем следующее. Запишем (34) в первую ячейку за REM - ноль. Затем (35) перенесем через стек содержимое HL в регистр DE и увеличим (36) на единицу содержимое DE. Теперь (37) снимем со стека число добавочных байтов в регистр BC и уменьшим это число на единицу (так как в одну ячейку мы уже занесли ноль). При выполнении LDIR (38) происходит следующее. Из ячейки 23760 ноль переписывается в 23761, затем из 23761 - в 23762 и так далее, пока не заполнится нулями вся добавочная REM-область кодов.

Вместо нуля можно задать любой другой код, изменив числовой параметр в команде (34). Задав, например, 32 (#20), вся добавленная REM-область заполнится пробелами, а задав 137 - графическими символами типа "шахматная клетка".

На этом работа по формированию REM-области заканчивается. Если на месте команды NOP (39) поставить RET, то произойдет возврат из машинного кода в вызывающую Бейсик-программу. Но в данном варианте происходит переход на дальнейшее выполнение программы в машинных кодах (40). Этот небольшой фрагмент служит для удаления строк с 1 до 10000 вызывающей Бейсик-программы. Строка с REM-областью имеет теперь номер 0 и не удаляется. Этот блок кодов (40) взят из ZX-РЕВЮ N3 за 1991 г. со стр. 50, поэтому я не останавливаюсь на нем подробно. В результате действия этого блока кодов в памяти компьютера останется только нулевая строка с REM-областью. Ее можно теперь записать на магнитофон или соединить с необходимой Бейсик-программой при помощи MERGE.

Для того, чтобы получить блок кодов "REM FILL", наберите программу на Бейсике:

```
10 LET N=23296: LET S=0
20 FOR X=N TO N+136
30 READ Y
40 POKE X.Y
50 LET S=S+Y
60 NEXT X
70 IF S<> 12985 THEN PRINT FLASH 1;"ERROR": STOP
80 SAVE "REM FILL"CODE 23396,137
90 STOP
100 DATA 1,2,0,42,75,92,9,34,75,92,42,85,92,9,34,85,92,42,89,92,9,34,89,92
110 DATA 42,91,92,9,34,91,92,42,93,92,9,34,93,92,42,97,92,9,34,97,92,42,99,
      92,9,34,99,92
120 DATA 221,42,83,92,221,229,221,54,0,0,221,54,1,0,221,102,3,221,110,2,229,
      9,221,116,3,221,117,2
130 DATA 209,19,19,19,225,25,197,197,235,42,101,92,229,237,82,229
140 DATA 193,209,225,25,34,101,92,235,43,27,237,184,35,54
150 DATA 0,229,209,19,193,11,237,176,0
160 DATA 33,1,0,205,110,25,229,33,16,39,205,110,25,209,205,229,25,201
```

Если Вы все набрали правильно, то программа сформирует и выдаст для записи на магнитофон блок кодов под именем "REM FILL". В строке 150 первое значение DATA определяет код символа, которым будет заполнена REM-область.

Бейсиковая часть программы "REM FILL" выглядит следующим образом:

```
1 REM 10 LOAD "REM FILL"CODE 23296
20 INPUT "No. of extra bytes:"; n
30 RANDOMIZE n: POKE 23297, PEEK 23670: POKE 23298, PEEK 23671
40 CLEAR : RANDOMIZE USR 23296
```

Программа стартует сначала, загружая блок кодов "REM FILL". Затем в строке 20 запрашивается, на сколько байтов надо увеличить начальную строку. (При этом в строке 1 после REM уже может находиться какая-либо информация, например, Ваша фамилия и телефон. Добавочные байты будут расположены после этой информации.) Строка 30 преобразует число добавочных байтов в двухбайтную форму и записывает полученное значение в ячейки 23297, 23296, подготавливая данные для блока кодов. Затем стартует блок кодов "REM FILL". В результате, после сообщения 0 OK. будет сформирована строка с заданным числом байтов после REM, номер этой строки станет 0, а все остальные строки Бейсик-программы будут уничтожены.

А теперь вернемся к нашему дебюту программы "PROG". В этом случае для формирования нулевой строки надо на запрос о числе добавочных байтов ответить: 1073. После того, как нулевая строка необходимой длины будет сформирована, наберите (или догрузите) текст программы "PROG". (Но не запускайте ее. Кроме этого, во избежание ошибок, подставьте RETURN: в начало строки 10. Это отключит пока блок кодов "ON ERROR GO TO" до тех пор, пока не будет набита, отлажена и работать без ошибок вся программа. Только после этого можно начинать вторую часть работы - отладку программы с блоком "ON ERROR GO TO", удалив RETURN из начала строки 10.)

Теперь можно загрузить в REM-область заранее подготовленные символьный набор, блок UDG и другие блоки кодов:

```
LOAD "CHR"CODE 23760,768
LOAD "UDG"CODE 24528,168
LOAD "ON ERR"CODE 24696,73
LOAD "SOUND 1"CODE 24769,32
LOAD "SOUND 2"CODE 24801,32
```

Если в начальной строке вы расположили после REM свою фамилию, то адреса загрузки кодовых блоков должны быть пересчитаны. Они должны увеличиться на столько, сколько занимает текст с Вашей фамилией.

Что касается блоков кодов "SOUND 1" и "SOUND 2", то они сформированы при помощи программы "SPECSOUND" фирмы "OZ SOFTWARE". Те читатели, которые не имеют этой программы, могут получить указанные блоки (для записи на магнитофон), набрав и запустив следующую Бейсик-программу:

```
10 LET N=23296: LET S=0
20 FOR X=N TO N+63
30 READ Y
40 POKE X,Y
50 LET S=S+Y
60 NEXT X
70 IF S<>6020 THEN PRINT FLASH 1;"ERROR": STOP
80 PRINT AT 10,12;"SOUND 1": RAHDOMIZE USR 23296: PAUSE 50: PRINT AT 10,18;"2": RANDOMIZE
      USR 23328: PAUSE 100
90 CLS : SAVE "SOUND 1"CODE 23296,32: SAVE "SOUND 2"CODE 23328,32
100 DATA 14,1,6,5,33,224,1,197,17,70,0,229,205,181,3,225,17,40,0,237,
      82,193,16,239,62,2,12,65,184,32,227,201
110 DATA 14,1,6,100,33,200,0,197,17,10,0,229,205,181,3,225,17,2,0,237,
      90,193,16,239,62,2,12,65,184,32,227,201
```

После старта, если Вы все набрали правильно, программа сформирует блоки кодов и выдаст их для записи на магнитофон.

В одной из следующих статей я подробно остановлюсь на работе этих блоков кодов, а также предложу усовершенствованный вариант программы "SPECSOUND", переведенный на русский язык.

И, в заключение, еще один момент. Если Вы размещаете блоки кодов в нулевой строке Бейсик программы, то отсутствует жесткая привязка этих кодов к конкретным адресам памяти компьютера, а есть только привязка к началу Бейсик-программы -

системной переменной PROG. Если Вы работаете с магнитофоном и у Вас нет планов обзаводиться дисководом, то все будет нормально. Если же Вы решите в будущем адаптировать Вашу программу для работы с дисковой операционной системой TR-DOS, то вы должны знать, что для работы этой системы в памяти компьютера выделяется дополнительно 112 байт ОЗУ, в которых хранятся новые системные переменные, связанные с работой TR-DOS. Эти дополнительные байты расположены непосредственно перед Бейсик-программой, поэтому последняя отодвинута на 112 байт в памяти компьютера и системная переменная PROG имеет значение не 23755, а 23867. Так что при работе с дисковой операционной системой все обращения к кодам нулевой строки должны быть смещены на 112 байт.

Аналогичные сюрпризы могут возникнуть не только при работе с дисковой операционной системой. В частности - если Вы пользуетесь ПЗУ "TURBO-90", которое имеет встроенный МОНИТОР и возможность загрузки программ с магнитофона с удвоенной скоростью. В некоторых режимах работы этого ПЗУ происходит сдвижка Бейсик-программы в область более старших адресов.

Ключей к избавлению от этих "подводных камней" является тот факт, что куда бы ни была сдвинута Бейсик-программа, ее место всегда указывается в системной переменной PROG. Исходя из этого, начало символьного набора в нулевой строке можно представить как (PROG)+5. При этом CHARS=(PROG)+ 5-256=(PROG)-251, начало UDG-графики (PROG)+5+768=(PROG)+773, начало блока "ON ERROR GO TO" будет: (PROG)+5+768+168=(PROG)+941 и т.д.

Теперь надо изменить некоторые строки нашего "дебюта":

```
7 RANDOMIZE PEEK 23635+256*PEEK 23636+773:POKE 33675, PEEK 23670: POKE 23676, PEEK 23671:
      RETURN : REM UDG
8 RANDOMIZE PEEK 23635+256*PEEK 23636-251: POKE 23606, PEEK 23670: POKE 23607, PEEK 23671:
      RETURN : REM RUS
10 POKE PEEK 23635+256*PEEK 23636+993, PEEK 23670: POKE PEEK 23635+256*PEEK 23636+994,PEEK
      23671: RANDOMIZE USR (PEEK 23635+256*PEEK 23636+941): RETURN
```

Строку 9 изменять не надо. Вызов звуков SOUND 1 и SOUND 2 может осуществляться через GO SUB. Определим для этого, скажем, строки 11 и 12:

```
11 RANDOMIZE USR (PEEK 23635+256*PEEK 23636+1014): RETURN
12 RANDOMIZE USR (PEEK 23635+256*PEEK 23636+1046): RETURN
```

Тогда вызов SOUND 1 будет GO SUB 11, а SOUND 2 GO SUB 12.

Этот вариант обращения к кодам нулевой строки предпочтительнее, он гораздо более универсален, так как не требует какой-либо переделки для адаптации программы под БЕТА-ДИСК интерфейс, да и в других случаях не принесет Вам сюрпризов. Но, поскольку ничего в природе не дается даром, то такой вариант имеет и свой недостаток - он медленнее работает. Работая над программой, оцените сами, существенно ли это замедление для Вас, и решите, какой вариант Вам более подходит.

* * *

Рассматривая дебют программы "PROG", я упомянул о том, что со строк 200, 300... могут быть расположены меню программы. В следующей статье мы подробно рассмотрим возможный вариант такого меню. Оно достаточно эффектно выглядит на экране и легко переналаживается на разные режимы работы.

УНИВЕРСАЛЬНОЕ МЕНЮ

Итак, Вы получили весь необходимый аппарат, который можете использовать, начиная дописывать к "дебюту" текст Вашей программы. У Вас уже есть переключатели

UDG-графики, русского и латинского символьного наборов. Вы можете заблокировать остановку программы и имеете кое-какие удобства для пользования - автоматизм сохранения программы на ленте.

Итак, предположим, что, загрузив "дебют", Вы исправили строку 3 программы. Теперь она вывела на экран название вашей будушей программы, Вашу фамилию, дату написания, и внизу экрана появилась табличка: "нажмите любую клавишу". Что же дальше?

Как правило, ни одна игра не начинается сразу. Необходимо сначала задать управление, выбрать уровень сложности и т.д. Для этого можно использовать цифровые и буквенные клавиши. Обычно это делается по принципу:

```
100 INPUT "Введите уровень сложности (1-5) : "; n
```

или так:

```
100 PRINT "Бесконечные жизни (Y/N)?": PAUSE 0
110 IF INKEY$ "y" OR INKEY$ "Y" THEN ...
```

Однако наиболее профессиональным является вариант меню. Меню - это перечень возможных вариантов ответа, например:

```
KEMPSTON
SINCLAIR
PROTEK
CURSOR
KEYBOARD
```

Здесь каждая строка – это пункт меню. Один из пунктов меню всегда выделен (яркость, цвет, инверсия). При этом переход от одного пункта к другому происходит при нажатии клавишей "ВВЕРХ" или "ВНИЗ", а выбор выделенного пункта осуществляется и клавишами "ENTER" или "0" ("ОГОНЬ"). Таким образом, в различных ситуациях в зависимости от требуемого запроса будет меняться текст меню, а управление будет всегда одинаковым и может осуществляться, например, только курсор джойстиком, то есть без помощи клавиатуры. Я хочу предложить вниманию читателей достаточно отработанный вариант меню, которое применяю в своих программах. Оно хорошо смотрится на экране и легко перенастраивается под разные конкретные варианты. Чтобы пример не был абстрактным, предположим, что мы разрабатываем программу для обучения детей устному счету. В этой программе, (назовем ее "PRIM") компьютер задает примеры на сложение, вычитание и т.д. и контролирует результат игравшего. Кроме того, в игре предусмотрим режим, когда играющий сможет сам задавать примеры компьютеру. При этом компьютер превращается в калькулятор (забегая вперед, скажу, что моя дочь, например, с удовольствием проверяет таким способом домашнее задание). Этот режим делает игру более разнообразной, и к тому же, психологически это как бы ставит компьютер и играющего в равные условия. Зададим еще режим "КОНЕЦ РАБОТЫ", когда компьютер поблагодарит Вас за уделенное ему время.

Главное меню этой программы может выглядеть следующим образом:

```
ПРИМЕРЫ ЗАДАЕТ КОМПЬЮТЕР
ПРИМЕРЫ ЗАДАЕТ ИГРАЮЩИЙ
КОНЕЦ РАБОТЫ
```

Если Вы выберете первый пункт меню, то будет сделан переход на следующее меню, которое позволит выбрать тип решаемых примеров:

```
СЛОЖЕНИЕ
СЛОЖЕНИЕ И ВЫЧИТАНИЕ
УМНОЖЕНИЕ
УМНОЖЕНИЕ И ДЕЛЕНИЕ
ВСЕ ЧЕТЫРЕ ДЕЙСТВИЯ
ТАБЛИЦА УМНОЖЕНИЯ
КОНЕЦ РАБОТЫ
```

Алгоритм работы программы должен быть следующим. При выборе одного из первых пяти пунктов, уровень сложности должен постепенно увеличиваться при правильных ответах

и уменьшаться при определенном количестве неверных ответов, а при выборе шестого пункта уровень сложности на протяжении всей игры должен оставаться постоянным. При выборе последнего пункта происходит возврат в предыдуюее, главное меню.

Продолжая придерживаться структуры "дебюта" "PROG", определим для главного меню строки с 200 по 299, а для второго меню - строки с 300 по 399. При этом строки начиная с 30 (до 40), являются подпрограммой, выполняющей действия, непосредственно связанные с работой меню.

Итак, текст программы, затем - дальнейшие комментарии.

```
4 GO TO 100
30 RANDOMIZE 3: GO SUB 10: BORDER 1: PAPER 7: INK 0: CLS
31 GO SUB 8: FOR M= 1 TO NN: READ M$: PRINT AT ,(Y0+(M-1)*DY),X0;M$: NEXT M: PRINT AT Y1,3;
      INK 1; INVERSE 1; "SPACE,DOWN,UP,ENTER,BREAK"
32 IF MM<1 THEN LET MM=NN
33 IF MM>NN THEN LET MM=1
34 PRINT AT (Y0+(MM-1)*DY),X0-DX; PAPER 2; INK 6; BRIGHT 1; OVER 1;S$: BEEP .03,2*MM+10
35 PAUSE 0: LET I=(INKEY$=" " OR INKEY$:"6" OR CODE INKEY$=10)+(INKEY$="7" OR CODE
      INKEY$=11)*2*(INKEY$="0" OR CODE INKEY$=12 OR CODE INKEY$=13)*3: GO TO (35+I)
37 PRINT AT (Y0+(MM-1)*DY),X0-DX; OVER 1;S$: LET MM=MM-2*I+3:GO TO 32
38 BEEP .1,36: RETURN 190 LET MM=1
200 RESTORE 200: LET NN=3: LET Y0=8: LET X0 = 4: LET DY=2: LET DX=2: LET S$="
                      ": LET Y1=18: GO SUB 30:GO TO 200+MM*10
210 LET MM= GO TO 300 220 GO TO 8000
230 GO TO 9000
299 DATA "ПРИМЕРЫ ЗАДАЕТ КОМПЬЮТЕР", "ПРИМЕРЫ ЗАДАЕТ ИГРАЮЩИЙ","      КОНЕЦ РАБОТЫ"
300 RESTORE 300: LET NN = 7: LET Y0=7: LET X0 = 6: LET DY=1: LET DX=2: LET S$="
                    ": LET Y1 = 19: GO SUB 30: GO TO 300+MM*10
360 GO TO 1000
370 LET MM=3: GO TO 200
399 DATA "СЛОЖЕНИЕ", "СЛОЖЕНИЕ И ВЫЧИТАНИЕ","УМНОЖЕНИЕ","УМНОЖЕНИЕ И ДЕЛЕНИЕ", "ВСЕ ЧЕТЫРЕ
      ДЕЙСТВИЯ","ТАБЛИЦА УМНОЖЕНИЯ","КОНЕЦ РАБОТЫ"
```

Текстовые сообшения в программе напечатаны по-русски. Они будут так выглядеть на Вашем экране, если вы включите русско-латинский символьный набор командой GO SUB 8. Однако следует учесть, что имена переменный вводятся латинскими буквами, для этого надо включить режим CAPS LOCK.

В программе меню используются следующие переменные:

NN - число пунктов в меню.

X0 - левая граница (по горизонтали) текста меню.

Y0 - верхняя граница (по вертикали) текста меню.

DX - отступ выделяющей строки влево по отношению к тексту меню.

DY - шаг строк меню по вертикали.

Y1 - вертикальная координата вспомогательной строки с указанием управляющих клавишей.

MM - указатель меню.

Указатель меню MM - это основной параметр меню. Это выходной параметр, получаемый в результате работы меню. Это также и входной параметр, определяющий выделенный пункт при старте меню.

Перед тем, как начнется выполнение меню со строки 200 или 300, необходимым предварительным условием является задание указателя меню MM. Это сделано для того, чтобы можно было, обращаясь к меню из разных мест программы, выделять тот пункт, выбор которого в данный момент наиболее вероятен. Это создает повышенное удобство в работе и придает "профессионализм" программе. Поясню это на нашем примере. Так, при старте программы выделяется первый пункт главного меню: "ПРИМЕРЫ ЗАДАЕТ КОМПЬЮТЕР" (в строке 190 задается MM=1) потому, что именно этот режим наиболее вероятен для выбора. Далее, нажав ENTER, мы перейдем ко второму меню. Если теперь выбрать "КОНЕЦ РАБОТЫ", то на экране появится главное меню с выделенным третьим пунктом, а не первый, как было при старте программы. Повторное нажатие ENTER, для

подтверждения, и без лишних хлопот для оператора программа завершена. То есть в каждой конкретном случае обращения к меню, может быть выделен тот пункт, который наиболее вероятно будет выбран оператором, то есть программа отличаясь "деликатностью", как бы "подыгрывает" оператору, облегчая его действия.

Учитывая вышесказанное, процедура вызова меню всегда одинакова и имеет вид:

```
LET MM=...: GO TO ...
```

Это, например, строка 210 (вызов второго меню) или 370 (вызов главного меню). Кроме того, это строка 190 (это тоже вызов главного меню; здесь нет необходимости указывать GO TO 200, так как следующей выполняемой строкой и так является строка 200).

Теперь конкретно о главном меню. Строка 200 задает начальные параметры: число пунктов в главном меню NN=3. Левая и верхняя границы текста меню X0 и Y0 - Вы можете подбирать их экспериментально, добиваясь симметричного расположения текста меню на экране. То же относится и к величине DX и количеству пробелов в переменной S$. Что касается параметра DY, то если меню состоит из небольшого количества пунктов (2... 5), то лучше смотрится вариант, когда текст написан через строчку, то есть DY=2. Если число пунктов в меню большое, то строки следуют вплотную друг к другу (DY =1).

Далее в строке 200 идет выполнение подпрограммы меню: GO SUB 30. Такое выделение процедур выполнения меню в подпрограмму позволяет значительно сэкономить память компьютера в программах, где требуется сложная сеть многоступенчатых, взаимосвязанных между собой меню. Если в вашей программе меню используется только один раз, то, в принципе, можно было бы раскрыть подпрограмму, расположив весь текст меню в строках с 200. Но я не рекомендую это делать, вполне вероятно, что в будущем Вы захотите усовершенствовать программу, введя какие-то дополнительные режимы, для которых потребуются новые меню. Тогда Вы оцените такой подход.

Теперь подробно рассмотрим подпрограмму меню.

Строка 30 задает возврат при ошибке на перезапуск программы сначала при нажатии клавиши BREAK (со строки 3; подробно об этом говорилось в ZX-РЕВЮ). Далее идет задание цветов BORDER, PAPER, INK и очистка экрана.

Строка 31 выводит на экран текст меню, а также вспомогательный текст с указанием управляющих клавишей. При этом команда GO SUB 8 включает русско-латинский символьный набор.

Строки 32 и 33 проверяют указатель меню MM на допустимые пределы и "зацикливает" переход от одного пункта меню к другому, то есть, если нажать клавишу "ВНИЗ", то, дойдя до последнего пункта, при следующем нажатии будет сделан скачок к первому пункту. Аналогично - при движении вверх. Вы можете ликвидировать "зацикливание", сделав замену:

```
32 IF MM<1 THEN LET MM=1
33 IF MM>MM THEN LET MM=NN
```

Строка 34 выделяет пункт меню, определенный указателем MM. При этом звуковой сигнал как бы подтверждает этот факт, а тон сигнала ориентировочно указывает на предлагаемый выбор.

Строка 35 - режим ожидания нажатия клавиши. При этом в зависимости от нажатой клавиши меняется параметр I.

I=1, если нажаты клавиши "SPACE" или "6" или "ВНИЗ" (CAPS SHIFT+6)

I=2, если нажаты клавиши "7" Или "ВВЕРХ" (CAPS SHIFT+7)

I=3, если нажаты клавиши "0" или "DELETE" (CAPS SHIFT+0) или "ENTER"

I=0, если нажата любая другая клавиша.

Далее в строке 35 следует переход на строку 35+I. То есть, в последнем случае (I=0), будет сделан переход на эту же строку 35 и ожидание нового нажатия клавиши, в случае I=1 - переход на строку 36, но так как ее нет, то будет сделан переход на ближайшую следующую строку, то есть 37, как и в случае I = 2.

В строке 37 (если I=1 или 2) выделенный пункт меню становится невыделенным, указатель меню MM в зависимости от величины I либо увеличивается на единицу, либо

уменьшается на единицу и работа меню повторяется со строки 32, то есть проверка MM на допустимые пределы, выделение нового пункта меню и ожидание нажатия клавиши.

В случае, когда I=3, строка 38 осуществляет выход из меню и действия, связанные с этим, например, звуковой сигнал. Экспериментируя с меню, попробуйте изменить строку 38:

```
38 FOR M=1 TO NN: PRINT PAPER 8: INK 8. BRIGHT 8; OVER (M=MM);FLASH (M=MM);AT (Y0+(M-
      1)*DY),X0-DX;S$: NEXT M: PRINT AT Y1, 0;"          (32 пробела)          ": BEEP
      .1,36: PAUSE 10: RETURN
```

Такой вариант интереснее смотрится на экране. Попробуйте сами придумать какие-нибудь другие варианты.

По команде RETURN в конце строки 38 закончится работа меню (GO SUB 30) и мы возвращаемся на строку 200, где после GO SUB 30 следует переход на строку, номер которой вычисляется выражением 200+MM*10, то есть для трех возможных вариантов главного меню: строки 210,220,230.

Как уже говорилось выше, в том случае, если выбран первый пункт главного меню, то вызывается второе меню (строка 210). В двух других случаях управление передается непосредственно на соответствующие фрагменты программы (строки 220, 230).

Второе меню (строки 300...399) полностью идентично первому. Отличие состоит только в числе пунктов, работа его абсолютно такая же, как и главного меню.

Строка 300, в случае нажатия "ENTER" или "0", адресует программу к строкам, начиная с 310 по 370, в зависимости от величины указателя MM. Отсюда уже выполняются переходы на дальнейшие фрагменты программы. Так как строки 310, 320, 330, 340 и 350 - отсутствуют, то в случае MM=1...6 будет переход на строку 360, а отсюда должен быть организован переход на начало основной части программы - запрос уровня сложности игры.

Теперь отдельно обсудим момент, связанный с работой блока кодов "ON ERROR GO TO". Я использую следующий прием. Если в тот момент, когда на экране находится меню, нажать клавишу "BREAK", то программа перезапускается со строки 3 (см. строку 30), выводя на экран вступительную заставку с названием, фамилией и т.д. Почему именно со строки 3 - см. статью о структуре программы и дебюте "PROG". При этом обеспечивается "полухолодный" (или "полугорячий") старт программы, в отличие от полностью "холодного" старта со строки 2, где может быть размещена загрузка каких-либо кодовых кусков, например, скомпилированных кодов мелодии для озвучивания игры, приготовленных при помощи музыкального редактора "WHAM". Это могут быть и какие-нибудь другие усовершенствования, поэтому лучше не занимать строку 2.

Так выглядит работа универсального меню.

Для тех читателей, кого больше интересует конкретная игровая программа, чем отдельные куски, из которых она собрана, немного ниже приводится окончание программы "PRIM" для обучения детей устному счету. Это та часть программы, которая не имеет отношения к работе меню. Поэтому она выделена в отдельный листинг.

А теперь небольшое отступление от темы и возврат к теме предыдущей номера. А именно, с чего практически начинать написание (набивание) новой программы.

Вообще, я делаю так. Дебют "PROG" набран и хранится отдельно. Начиная воплощать в жизнь какую-то идею, я прежде всего загружаю его. Затем решаю, понадобится ли для этой программы меню. Если понадобится, то догружаю при помощи "MERGE" текст меню, который также набран и хранится отдельно (строки с 30 по 38). При этом вовсе не обязательно определяться с необходимостью меню в начале написания программы. Например вы пишете какую-то простенькую вспомогательную программу для своих целей, и вот, когда задача успешно выполнена, Вы решаете, что хорошо было бы добавить в программу какие-то новые режимы. Вот тут то Вы имеете возможность догрузить меню в вашу программу и с его помощью организовать новые режимы работы. Сами же отдельные фрагменты программы, тоже могут быть набраны и опробованы отдельно, так как для каждого блока программы определяются свои области, которые не должны перекрываться (со строк 1000, 2000 и т.д.). Смысл в том, что придерживаясь идеи "дебюта" и структуры программы, вы в любой момент можете усовершенствовать или видоизменить ее по

Вашему желанию. В этом и заключаются преимущества структурного программирования, о котором уже говорилось на страницах "ZX-РЕВЮ".

Прежде чем предложить листинг программы "PRIM", еще одно отступление. Точнее это не отступление, а целое маленькое исследование, но без него непонятен будет смысл некоторых изменений, которым подверглись знакомые уже Вам (по дебюту "PROG") фрагменты и приемы.

Наберите для этого простенькую программку:

```
10 RANDOMIZE
20 LET X=RND*100
30 LET Y=RND*100
40 PLOT X,Y
50 GO TO 20
```

Запустите ее. То, что вы видите на экране, является результатом работы функции случайной величины RND. Вы видите, что картина подобна той, которую рисует на асфальте начинающийся дождь. То есть, распределение в достаточной степени случайно.

Для того, чтобы задать начальный параметр для функции случайной величины, служит оператор RANDOMIZE n, где n - числовой параметр. Если этот параметр равен нулю (RANDOMIZE 0 или просто RANDOMIZE), то начальный параметром для RND служит значение системной переменной FRAMES - счетчика кадров. Так обеспечивается практически случайное число.

Измените теперь строку 50:

```
50 GO TO 10
```

И запустите программу RUN. то, что Вы видите сейчас, вовсе не похоже на дождь. То есть для нормального функционирования RND нельзя включать RANDOMIZE внутрь цикла работы RND. А то как бы все время функция RND запускается заново а первое ее значение совсем не случайно, оно определяется системной переменной FRAMES.

Продолжаем наше исследование. Измените строку 10:

```
10 RANDOMIZE 13345
```

(можете подставить любое число) и запустите программу. Теперь вообще нет никакого "дождя", все "капли" попадают в одну и ту же единственную точку на экране. Сейчас мы каждый раз в начале цикла задаем для RND одно и то же число и поэтому каждый раз получаем одинаковый результат. Все эти детали нужны вот для чего. В основе программы "PRIM" лежит использование функции RND для задания примеров ("ПРИМЕРЫ ЗАДАЕТ КОМПЬЮТЕР"). А сплошь и рядом в тексте программы будут встречаться: RANDOMIZE или RANDOMIZE USR ... это вызов SOUND или SOUND 2, подготовка двухбайтных данных для блока кодов "ON ERROR GO TO", запуск самого этого блока. Естественно, все эти RANDOMIZE попадают внутрь цикла работы RND, нарушая его функционирование.

Измените строку 10:

```
10 RANDOMIZE USR 124
```

По адресу 124 в ПЗУ находится команда RET, то есть сразу выполняется возврат, ничего не делая, но имитируется выполнение программы в машинных кодах. Запустите программу. На экране по-прежнему одна единственная точка. Как же выполнить программу в машинных кодах, не прибегая к помощи RANDOMIZE? Это сделать можно, используя вместо RANDOMIZE USR ... другой вариант:

```
10 LET S=USR 124
```

Примечание "ИНФОРКОМа"

Возможно использование также RESTORE USR, PRINT USR, PLOT USR и т.д. Кратко принцип формулируется следущим образом: "Если в программе используется генерация случайных чисел, то использовать RANDOMIZE для запуска машинного кода нельзя".

Подробности вы можете посмотреть в нашей новой книге "Элементарная графика". М:."ИНФОРКОМ", 1993, 207 стр. (это первый том готовящегося четырехтомника).

Измените строку 10 и запустите программу. Вот теперь на экране опять "дождь". RND работает правильно.

Кроме того, придется отказаться от услуг RANDOMIZE по превращению чисел в двухбайтную форму. Для этого сделаем новую подпрограмму в строке 15. Число, подлежащее преобразованию в двухбайтную форму обозначим W. Тогда старший (HI) и младший (LO) байты будут вычисляться в строке 15:

```
15 LET HI=INT (W/256): LET LO=W-HI*256: RETURN: REM 2-BYTE CONVERTER
```

Учитывая все вышесказанное, начальные служебные строки (с 7-й) будут теперь такими:

```
7 LET W=PEEK 23635+256*PEEK 23636+773: GO SUB 15: POKE 23675,LO: POKE 23676,HI: RETURN: REM
      UDG
8 LET W=PEEK 23635+256*PEEK 23636-251: GO SUB 15: POKE 23606,LO: POKE 23607,HI: RETURN: REM
      РУС
9 POKE 25606,0: POKE 23607,60: RETURN: REM ЛАТ
10 GO SUB 15: POKE PEEK 23635+256*PEEK 23636+993, LO: POKE PEEK 23635+256*PEEK 23636+994,HI:
      LET S=USR (PEEK 23635+256*PEEK 23636+941): RETURN
11 LET S = USR (PEEK 23635+256*PEEK 23636+1014): RETURN: REM SODND 1
12 LET S = USR (PEEK 23635+256*PEEK 23636+1046): RETURN: REM SOUND 2
15 см. выше
```

Для инициирования блока кодов "ON ERROR GO TO" например в строке 30 вместо:

```
RANDOMIZE 3: GO SUB 10
```

теперь используется другая конструкция:

```
LET W=3: GO SUB 10
```

а в начало строки 10 теперь подставлено GO SUB 15, где число и преобразуется в двухбайтную форму.

Если Вы догружаете программу меню с магнитофона, то не забудьте изменить начало строки 30.

Программа "PRIM"

Теперь, наконец, переходим к тексту остальных строк программы "PRIM". Комментарии - потом.

```
0 REM коды
3 LET W=3: GO SUB 10: GO SUB 7: GO SUB 8: BORDER 1: PAPER 0: INK 6: BRIGHT 0: CLS : PRINT
      BRIGHT 1;AT 2,2: "ПРОГРАММА ДЛЯ ОБУЧЕНИЯ СЧЕТУ": PAPER 2; INK 7; INVERSE 1; AT 7,9: "
      ";AT 8,9; "    P R I M    ";AT 9,9;"               "; PAPER 0: INK 2: INVERSE 0:AT
      14,6:"АВТОР: АЛЕКСЕЕВ А.Г."
4 PRINT INK 4: BRIGHT 1; AT 17,12;"01,06,92": BEEP .1,26: BEEP .1,20: GO SUB 20: GO TO 100
5 GO SUB 9: SAVE "PRIM" LINE 2: VERIFY "PRIM"
6 GO TO 5
7
.
. см. выше
.
15
20 INPUT ;: PRINT #0; "     НАЖМИТЕ ЛЮБУЮ КЛАВИШУ"
22 LET W=22: GO SUB 10: PAUSE 0
24 IF INKEY$="q" OR INKEY$="Q" THEN GO TO 9999
26 RETURN
30
.
. см. текст программы меню
.
38
40 LET T=0: RETURN
41 LET T=INT (RND*2): RETURN
42 LET T=2: RETURN
43 LET T=INT (RND*2+2): RETURN
44 LET T=INT (RND*4): RETURN
```

```
45 LET T=2: RETURN
50 LET W=50: GO SUB 10: INPUT O
52 IF O=0 THEN GO TO 4000
54 PRINT O;
56 LET W=3: GO SUB 10: RETURN
60 PRINT TAB 20; PAPER 2: INK 7; BRIGHT 1; " ОШИБКА "''
62 GO SUB 12
64 LET E=E+1: LET U=U+1
66 RETURN
70 PRINT TAB 20; PAPER 4; " ВЕРНО "''
72 GO SUB 11
74 LET R=R+1: LET I=I+1
76 RETURN
100 RANDOMIZE
190 LET MM=1 200
.
. см. текст программы меню
.
399
1000 BORDER 7: PAPER 7: INK 0; CLS
1010 LET W=1010: GO SUB 10: INPUT "ЗАДАЙТЕ УРОВЕНЬ : "; LINE L$
1020 IF L$="" OR L$<="0" THEN LET MM=2: GO TO 200
1030 LET L = VAL L$
1040 LET W=3: GO SUB 10: LET N=L: LET R = 0: LET E=0: LET I=1
1050 PRINT AT 11,0;"ДЛЯ ЗАВЕРШЕНИЯ РАБОТЫ ВВЕДИТЕ 0": BEEP .05,32
1100 PRINT AT 19,10; "УРОВЕНЬ ";INT L
1110 LET U=0
1120 FOR C=1 TO 5
1130 LET X=INT (RND*.7*L+.3*L+.5)
1140 LET Y=INT (RND*.7*L+.3*L+.5)
1150 PRINT AT 21,1; I;"."
1160 GO SUB 40+MM-1
1170 GO SUB 2000+100*T
1180 NEXT C
1200 PRINT '''''
1210 IF U=0 THEN PRINT AT 17,10:PAPER 6; INK 2; BRIGHT 1; " ОТЛИЧНО ": LET L =
      L+.2*L"(MM<>6)
1220 IF U=1 THEN PRINT AT 17,10;PAPER 5; INK 1;"  ХОРОШО  "
1230 IF U>=2 THEN PRINT AT 17,10; PAPER 1; INK 5; BRIGHT 1; " ПЛОХОВАТО ": LET
      L=L=.2*L*(MM<>6)
1240 GO TO 1100
2000 LET Z=X+Y
2010 PRINT AT 21,5;X;"+";Y;"=";
2020 GO SUB 50
2030 IF O<>Z THEN GO SUB 60: GO TO 2010
2040 GO TO 70
2100 LET Z=X+Y
2110 PRINT AT 21,5;Z;"-";X;"=";
2120 GO SUB 50
2130 IF O<>Y THEN GO SUB 60: GO TO 2110
2140 GO TO 70
2200 LET Z=X*Y
2210 PRINT AT 21,5;X;"x";Y;"=";
2220 GO SUB 50
2230 IF O<>Z THEN GO SUB 60: GO TO 2210
2240 GO TO 70
2300 LET Z=X*Y
2310 PRINT AT 21,5;Z;":";X;"=";
2320 GO SUB 50
2330 IF O<>Y THEN GO SUB 60: GO TO 2310
2340 GO TO 70
4000 LET W=3: GO SUB 10: BORDER 4: CLS : PRINT AT 1,5;"ТИП ПРИМЕРОВ: "; AT 3,6;
4010 RESTORE 300: FOR A=1 TO MM: READ A$: NEXT A: PRINT A$
4020 PRINT AT 7,5; "НАЧАЛЬНЫЙ УРОВЕНЬ : ";N;AT 9,5;"КОНЕЧНЫЙ УРОВЕНЬ : "; INT L
```

```
4030 PRINT AT 14,10; PAPER 4;" ВЕРНО "; PAPER 7;" : ";R;AT 15,10; PAPER 2; INK 7; BRIGHT 1;
      " ОШИБОК "; PAPER 7; INK 0; BRIGHT 0; " : "; E
4040 BEEP .1,36: BEEP .1,20: PAUSE 0
4050 LET MM=3: GO TO 200
8000 LET W=3: GO SUB 10: BORDER 7: PAPER 7: INK 0: CLS
8010 PRINT AT 11,0;"ДЛЯ ЗАВЕРШЕНИЯ РАБОТЫ ВВЕДИТЕ 0": BEEP .05,32
8020 PRINT AT 21,0;"ВВЕДИТЕ ПРИМЕР ( БЕЗ ЗНАКА = ) :"
8030 LET W=8030: GO SUB 10: INPUT LIKE X$: IF X$="0" THEN LET MM=1: GO TO 200
8040 LET X=VAL X$
8050 PRINT ''AT 21,0;X$;"=";X'''
8060 GO SUB 11: GO TO 8020
9000 LET W=9030: GO SUB 10: BORDER 3: PAPER 7: INK 0: CLS
9010 PRINT AT 11,10: INK 1;"ДО СВИДАНИЯ."'''" ПРИЯТНО БЫЛО С ВАМИ ПОРАБОТАТЬ!"
9020 BEEP .1,26: BEEP .1,20: PAUSE 0
9030 RANDOMIZE USR 0 9999 BORDER 7: PAPER 7: INK 0: BRIGHT 0: INVERSE 0: POKE 23658,8
```

Переменные, используемые в программе.
L$, L - уровень чисел для счета
N - начальный уровень
R - счетчик правильных ответов
E - счетчик ошибочных ответов
I - порядковый номер примера
U - счетчик ошибочных ответов в "столбике" из пяти примеров
C - параметр цикла, определяющий число примеров в "столбике"
X,Y,Z - операнды для задания примеров
T - тип примера
T=0 - сложение
T=1 - вычитание
T=2 - умножение
T=3 - деление
O - ответ, вводимый играющим.

В этой программе собраны все моменты, освещенные в предыдущих статьях цикла. Набирая и отлаживая программу, Вы еще раз остановитесь на них.

На подпрограммах в строках 40…66 мы остановимся по ходу работы основной части программы. Основная часть - режим задания примеров компьютером - начинается со строки 1000.

В строке 1010 запрашивается начальный уровень чисел для счета. Это величина, которую не могут превышать числа, например слагаемые при выполнении сложения.

Если не задавая никакого уровня нажать "ENTER", или задать уровень 0, то произойдет возврат в главное меню с выделением второй строки меню (установка режима задания примеров играющим). Чтобы реализовать возможность простого нажатия ENTER, в строке 1010 вместо INPUT L стоит INPUT LINE L$ и далее в строке 1030 числовой переменной L присваивается значение текстовой переменной L$.

Обратите внимание на применение блока кодов "ON ERROR GO TO". В строке 1010 перед INPUT стоит конструкция LET W=1010: GO SUB 10, которая в случае ошибки при вводе уровня возвратит программу на строку 1010 и повторит ввод. Такая ошибка может произойти в начале строки 1030, если, например, вместо числа, задать букву. Когда опасность такой ошибки миновала, корректно будет возвратить блок "ON ERROR GO TO" в исходное состояние (то есть при ошибке программа перезапускается со строки 3, как мы договаривались выше). Это происходит в строке 1040. Здесь также запоминается начальный уровень для подведения итогов в конце игры, обнуляются счетчики верных и ошибочных ответов, счетчику примеров присваивается номер 1.

Строка 1050 информирует играющего о том, как он может завершить работу программы.

Со строки 1100 начинается следующий этап в работе программы - задание "столбика" из пяти примеров. Вначале выводится текущее значение уровня чисел для счета. Затем в

строке 1110 обнуляется счетчик ошибок в одном столбике (U). Затем (в строке 1120) организуется цикл, определяющий пять примеров в "столбике".

В строках 1130 и 1140 задаются значения операндам, участвующим в примерах, при этом числа для счета получаются в диапазоне от 0.3*L до L. Такое ограничение позволяет, если уровень чисел для счета задан, например 100, избежать примеров типа: "1+2=".

Строка 1150 выводит на экран номер решаемого примера.

Подробнее разберемся со строкой 1160. Здесь, в зависимости от того, какой вариант из второго меню был выбран (указатель MM), выполняется соответствующая подпрограмма, задавая тип примера (T). Если было выбрано только сложение (MM=1), то выполняется GO SUB 40, то есть T становятся равным 0. Если сложение и вычитание (MM=2), то выполняется GO SUB 41, где T может принять одно из двух значение: 0 или 1 и т.д. В зависимости от величины T находится тип конкретного задаваемого примера:

T=0 - сложение
T=1 - вычитание
T=2 - умножение
T=3 - деление

Далее, строка 1170 в зависимости от величины T, вызывает выполнение соответственно одной из четырех подпрограмм из строк 2000, 2100, 2200 или 2300 для каждого типа примеров. Они идентичны, за исключением некоторых деталей. Для того, чтобы при вычитании не получалось отрицательных чисел, а при делении - дробных чисел, используется простой прием. Сначала подсчитывается результат, а пример строится, используя результат, как исходные данные.

Рассмотрим, для примера, подпрограмму со строки 2000. В строке 2000 осуществляется подсчет результата, в 2010 - вывод примера на экран. В следующей строке GO SUB 50 - это ввод играющим ответа и проверка его на ноль для выхода из игры. В этом случае будет переход на строку 4000 (см. строку 52). Так как при работе INPUT опять возможны всякие неприятности, в строке 50 активизируется блок "ON ERROR GO TO", заставляя программу возвращаться на эту строку, на ввод ответа. И опять корректно будет, завершая подпрограмму ввода ответа вернуть блок "ON ERROR GO TO" к переходу по ошибке на строку 3 перед тем, как выполнить RETURN в строке 56. Далее в строке 2030 идет анализ величины ответа, и, если ответ неверный, то выполнение подпрограммы GO SUB 60 - это реакция программы на ошибку и далее, возврат на повторение этого примера еще раз (со строки 2010).

В подпрограмме со строки 60 идет вывод таблички "ОШИБКА", затем звуковой сигнал SOUND 2, а также на единицу увеличивается счетчик ошибок (E) и счетчик ошибок в этом столбике (U).

Строка 2040 - выполнение подпрограммы при правильном ответе. Вообще-то наверное понятнее в строке 2040 было бы записать:

```
GO SUB 70: RETURN
```

То есть выполнение подпрограммы "ВЕРНО", затем возврат в вызывавшую подпрограмму - на строку 1180. Но результат одинаковый. В том варианте, который приведен в листинге, в вызывающую программу (на строку 1180) управление вернет оператор RETURN из строки 76. Подпрограмма со строки 70 - это вывод таблички "ВЕРНО", звуковой сигнал SOUND 1, затем увеличение на единицу счетчика правильных ответов (R) и увеличение на единицу сквозной нумерации примеров (I).

После возврата в вызывающую программу на строку 1180 происходит либо повтор цикла, то есть вывод нового примера из "столбика", либо "столбик" завершен и пора оценить промежуточный результат.

Строки 1210, 1220 и 1230 выполняют анализ числа неправильных ответов в одном столбике. Если неправильных ответов нет (строка 1210), то выводится оценка "отлично". Уровень чисел увеличивается на 20 процентов (но только в тех случаях, если не была выбрана таблица умножения, то есть если MM<>6), иначе уровень сложности не

увеличивается.

Если в столбике один неправильный ответ (строка 1220), то выводится сообщение "ХОРОШО", а уровень - не изменяется.

Если в столбике два или более неправильных ответов (строка 1230), то выводится табличка "ПЛОХОВАТО", а уровень чисел понижается на 20 процентов (если только это не режим таблицы умножения).

Когда будете придумывать текст сообщений компьютера оператору, следует воздержаться от резких и категоричных высказываний типа: "ОЧЕНЬ ПЛОХО" или "ВЫ НИЧЕГО НЕ ЗНАЕТЕ". Компьютер должен быть вежлив в общении с оператором, тем более, если оператор - ребенок. Даже если он совсем ничего не знает или невнимателен при вводе ответа, компьютер не ругает его нехорошими словами, а лишь слегка "недоумевает" по поводу неправильного результата. В этом случае у ребенка не возникает раздражения и не "отпадает охота", а появляется желание играть в эту игру и дальше. А ведь как раз это и нужно для того, чтобы развить способности к устному счету.

Кстати, вместо "ПЛОХОВАТО" можно выводить совсем нейтральное: "УПРОСТИМ ЗАДАЧУ".

Далее (строка 1240), работа программы возобновляется со строки 1100, то есть задание нового "столбика" примеров.

Если при выполнении подпрограммы ввода ответа (GO SUB 50) будет принят ноль (строка 52), то управление передается на строку 4000 - это подведение итогов работы, где на экран выводится следующая информация, число решенных примеров, их тип, начальный и конечный уровни решенных примеров, а также число верных и ошибочных ответов. Затем (строка 4050) вызывается главное меню, предлагая завершить программу (выделен третий пункт меню).

Строки с 8000 - эта часть программы выполняет режим калькулятора и может использоваться, например, при проверке учеником домашнего задания. В случае ошибки при вводе ответа, блок "ON ERROR GO TO" вернет программу на ввод примера, на строку 8030.

Строки с 9000 (по 9030) - финальная часть программы. Сюда можно добавить выполнение нехитрой мелодии (при помощи нескольких операторов BEEP), исполнение которой зациклено до нажатия на какую-нибудь клавишу. После нажатия на клавишу - рестарт компьютера. Попробуйте эту часть программы придумать сами.

Для того, чтобы в процессе отладки можно было останавливать программу, установлен "жучок" в строке 24, переводящий на строку 9999, которая обеспечит удобные для работы цвета бумаги, чернил и т.д., включение режима курсора [C] и остановку программы с сообщением 0 OK, так как это последняя строка программы. Практически, для остановки программы надо войти в любое меню, затем нажать BREAK (для перехода в титульную заставку), затем нажать "Q".

Остановку программы можно обеспечить по-другому, например непосредственно из режима меню.

Для этого измените строку 35:

```
35 PAUSE 0: LET I=(INKEY$=" " OR INKEY$="6" OR CODE INKEY$=10) +(INKEY$="7" OR CODE
   INKEY$=11)* 2+(INKEY$="0" OR CODE INKEY$=12 OR CODE INKEY$=13)*3+(INKEY$="q" OR
   INKEY$="Q")*4: GOTO (35+I)
```

И добавьте строку 39:

```
39 GO TO 9999
```

В общем, чем больше Вы будете экспериментировать, тем лучше. Это всегда полезнее, чем просто скопировать готовую программу. Важно понять подход, принцип, а уж как его развить и использовать дальше - это на Ваше усмотрение.

* * *

В тексте часто упоминаются блоки кодов SOUND 1 и SOUND 2. Они сформированы при помощи программы "SOUND". За основу этой программы была взята программа "SPECSOUND" фирмы "OZ SOFTWARE", которая была "доведена до ума" и переведена на русский язык. Следующая статья будет посвящена этой теме. Я подробно остановлюсь на работе этих блоков кодов, а также предложу читателям программу "SOUND".

Маленькие Хитрости

Признаться по совести, нам очень понравилась идея создания универсального программного "дебюта", предложенная автором предыдущей статьи. Конечно, "универсального" не в том смысле, что он должен быть один на всех и всех устраивать, а в том смысле, что каждый, кто программирует на "Спектруме", сможет сделать для себя собственную домашнюю заготовку и "подшивать" ее всякий раз, когда ему придет в голову написать какую-нибудь программу.

Идея "дебюта" - это не только программистская идея, ее ценность идет гораздо дальше. Из мемуаров известно, с каким трудом многие творческие личности берут себя по утрам за шиворот и тянут к письменному столу работать, а ноги упирается. Куда приятнее: подошел к столу, нажал пару кнопок и готово дело, дебют уже сделан, можно с чистой совестью отправляться обедать, а потом и вовсе отдыхать.

Многие писатели практикуют даже такой прием: отходя ко сну, они бросают своих героев на самом интересном месте, обрывая фразу на полуслове. А на следующий день спокойно с утра добьют своих героев, похоронят, если получится, и таким жизнерадостным дебютом начинают новый день в самой работоспособном настроении.

Вот и мы прочитали статью уважаемого автора и захотелось "плодотворную дебютную идею" (как говаривал высокочтимый нами Остап Ибрагимович) как-то развить и упрочить. Захотелось и свой вклад в большое дело внести. Попробовали и так и этак, ничего в голову не идет - надо все-таки программировать, а без этого ну просто никуда.

И тут нас осенило: ну зачем нам программировать!? Ведь в компьютере уже есть куча программ в ПЗУ. Может быть ими и воспользуемся? Зачем велосипед изобретать?

Взялись за дело. Все ПЗУ прошарили, все обыскали, пачку бумаги испачкали - но нет там ничего такого, чтоб компьютер сам за нас программу написал. Эх, лучше было бы это время на "ЭЛИТУ" потратить. И вдруг…

Самое интересное всегда происходит неожиданно. Вдруг по адресу 196ЕН (6510 - десятиричное) мы нашли процедуру, которая выдает адрес начала любой БЕЙСИК-строки в оперативной памяти, особенно если вы предварительно зашлете номер этой строки в регистровую пару HL процессора. После того, как процедура отработает, она оставит искомый адрес в той же регистровой паре HL.

НУ вот, решили мы, половина дела сделана. Адрес начала любой строки компьютер найдет сам, осталось только, чтобы он сам и строку туда записал. Но не тут то было - ничего для этого в ПЗУ нет. Тут что-то Клайв Синклер недодумал, за что только его лордом сделали?

Но мы не отчаялись и нашли еще одну интересную процедуру, которая начинается с адреса 19Е5Н (6629 DEC). Эта процедура способна уничтожить все, что находится в БЕЙСИК-программе между адресом, который установлен в регистровой паре DE, и адресом, который установлен в регистровой паре HL. В общем, шикарный метод для стирания программ. Не совсем то, что мы искали, но хоть что-то. Уж если не попрограммируем, зато хоть постираем от души, благо все делается автоматически.

А ведь если честно, то какое же программирование без хорошего стирания? Вы когда-нибудь хоть строчку написали, чтобы ее потом не стирать? Вот то то же! Всякое правильное программирование всегда начинается со стирания того, что ранее было запрограммировано неправильно.

А уж объединить эти две процедуры в одну программу в машинных кодах - дело нехитрое. Вот и получилась такая миленькая программа, размером всего в 19 байтов, которая хотите верьте, хотите нет, а сэкономит массу времени любому, кто хоть когда-нибудь написал что-то длиннее, чем тридцать строк.

Можете теперь ее встроить в свой собственный "дебют", если хотите, можете держать ее отдельно. Делайте с ней, что хотите, храните в любых адресах, можете встроить в БЕЙСИК-строку, воспользовавшись приемом, которым поделился автор предыдущей статьи.

Мы же, не зная заранее, куда Вы предпочтете ее препроводить, разместили код в буфере принтера, начиная с адреса 23300, а БЕЙСИК-загрузчику выделили строки выше 9990, чтобы под ногами не путался.

Итак, распечатка того, что у нас вышло, помещена в Листинге 1. А распечатка БЕЙСИК-загрузчика приведена в Листинге 2. Самое интересное - то, что самой этой же программой можно стереть и этот БЕЙСИК-загрузчик. Правда, при стирании строки 9996 будет выдано сообщение C: Nonsense in BASIC, но оно уже никакого значения для нас не имеет.

Так что стирайте в собственное удовольствие все, что хотите. Обратите только внимание на нашу статью, посвященную ошибкам в ПЗУ - там в разделе, посвященном функциям пользователя FN () кое-что сказано о том, почему нельзя код для стирания строк оформлять в виде пользовательских функций.

Успешного Вам DELETa!

Листинг 1

```
210000      LD HL,00          ; Номер начальной строки помещается в HL.
                              ; Сейчас там нули, но когда Вы введете
                              ; свое число, оно поступит туда.
CD6E19      CALL 196EH        ; Вызов процедуры ПЗУ для определения
                              ; адреса начала строки.
E5          PUSH HL           ; Запомнили полученный адрес на стеке.
210000      LD HL,00          ; Номер конечной строки помещается в HL.
                              ; Сюда мы тоже введем свое число.
33          INC HL            ; Указание на следующую строку, которая не
                              ; будет удалена.
CD6E19      CALL 196EH        ; Определили адрес конца последней строки.
D1          POP DE            ; Адрес начальной строки сняли со стека в DE.
CDE519      CALL 19E5         ; Удаление строк из заказанного интервала.
C9          RET               ; Возврат в БЕЙСИК.
```

Листинг 2

```
9991 DATA 33,0,0,205,110,25,229,33,0,0,35,205,110,25,209,205,229,25,201
9992 RESTORE 9991: FOR n=0 TO 18: READ a: POKE 23300+n,a: NEXT n
9993 INPUT "Введите номер начальной СТРОКИ"; a
9994 POKE 23301,a-256*INT(a/256): POKE 23302, INT(a/256)
9995 INPUT "Введите номер конечной строки"; b
9996 IF b<a THEN GO TO 9992
9997 POKE 23308,b-256*INT (b/256): POKE 23309, INT (b/256)
9998 RANDOMIZE USR 23300
```

ВЕКТОРНАЯ ГРАФИКА

Сегодня мы предлагаем Вашему вниманию небольшой отрывок из главы готовящейся сейчас книги "Прикладная графика". Книга является логическим продолжением ранее вышедшего тома "Элементарная графика", а глава, отрывок из которой здесь приведен, посвящена одной из проблем трехмерной векторной графики.

Вам, конечно, неоднократно приходилось сталкиваться с векторной графикой в игровых программах. Практически полностью на ней построена любимая игра тысяч наших читателей "ELITE", та же графика в программах "ACADEMY", "STARION" и в очень необычной, увлекательной программе, требующей тонкого расчета и стратегического мышления - "CENTINELL". Список игр, инкорпорирующих векторную графику, мог бы быть очень и очень обширным и среди них многие относятся к лучшим из лучших.

Вы, уважаемые читатели, обратили, конечно, внимание на то, что векторная графика в этих играх одноцветная, угловатая и художественными достоинствами очевидно не отличается. Так почему же они пользуются таким успехом, с чем он связан?

Да, конечно, векторная графика выглядит на экране победнее, чем многоцветная растровая графика, но у нее есть два огромных преимущества.

Во-первых, это очень быстрая графика. Цикл освежения экрана и перестроения изображения происходит намного быстрее, чем в программах с растровой графикой.

Во-вторых, это вычисляемая графика, то есть не надо хранить в памяти компьютера заранее подготовленные экраны. Все изображения рассчитываются по заданным алгоритмам и практически никогда не повторяются. Благодаря этому Вы можете иметь в таких программах тысячи планетных систем, десятки возможных кораблей противника и нескончаемое разнообразие игровых ситуаций.

Вспомним программу "ELITE". Да, конечно, нужно иметь воображение, чтобы принять угловатую "морковку" на экране Вашего монитора за роскошный корабль "Fer-de-Lance", нашпигованный чудесами науки и техники и отделанный изнутри лучшими породами дерева и самыми дорогими материалами.

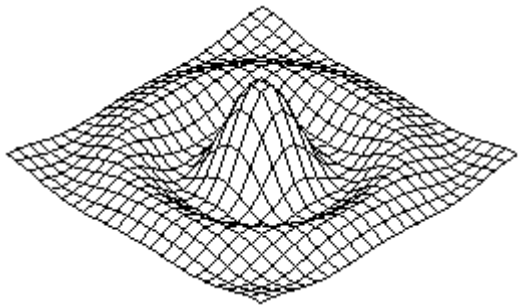


Рис. 1. График функции Z=SIN(R)/R, где
R = SQR (X*X + Y*Y)
Параметры приняты такими, как приведенные в БЕЙСИК-распечатке. Другие параметры дадут иную поверхность.

Но зато когда он вращается вокруг всех собственных осей и при этом летит в пространстве, изменяя свои координаты относительно Вашего корабля, а Вы вместе с ним при этом перемещаетесь и маневрируете относительно планеты, звезды, станции и прочих кораблей и этот клубок пронзают залпы лазеров, в нем летят и находят свою цель ракеты, здесь Вы забываете обо всем - и о "морковке" и о черно-белой графике. Перед Вами реальный, хорошо вооруженный противник - это вызов Вашему мастерству. Динамика игры, острота схватки и неповторяемость ситуаций делают возможным для вас эффект реального присутствия и вам уже не нужно художественное впечатление от богатства красок. Ваш мозг, увлеченный переживаниями, сам домыслит столько, сколько ему надо.

В этой статье мы коснемся только одной маленькой проблемы, которая связана с изображением на экране трехмерной векторной графики. Те из наших читателей, которые захотят скрупулезно изучить вопросы векторной (и не только векторной) графики прочитают книгу, а здесь мы рассмотрим один полезный алгоритм, который может быть принят на вооружение по крайней мере теми, кто использует свой "Спектрум" в практической работе, например при написании курсовых и дипломных проектов.

Сокрытие невидимых линий контура

При работе с трехмерной векторной графикой часто встает одна важная проблема - как изображать невидимые линии трехмерного объекта? Эта задача имеет непосредственное отношение к системам автоматизированного проектирования и наибольшее развитие получила именно в теории этих систем и, надо сказать, для ее решения привлекают довольно сложный аппарат из той области высшей математики, которая называется аналитической геометрией. Для нас с Вами, поскольку мы занимаемся обычной прикладной графикой, эту задачу можно несколько упростить. На данном этапе нас не интересует как изобразить невидимые линии - нам просто нужно их НЕ ИЗОБРАЖАТЬ.

Приемов и методов для достижения этой цели немало и мы рассмотрим один из наиболее простых, поддающийся несложной алгоритмизации.

Посмотрите на рис. 1. На нем изображен некоторый трехмерный ландшафт. Фактически это график функции:

Z = SIN (R)/R, где R = SQR (X*X + Y*Y).

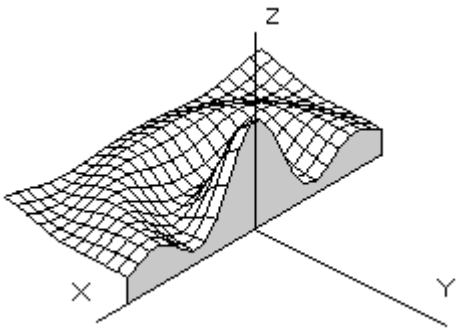


Рис.2

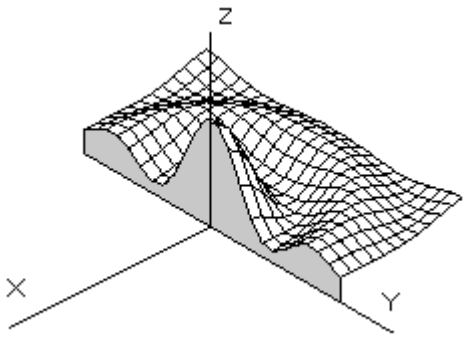


Рис.3

В своих экспериментах вы можете изменить эту функцию и поработать с другими. Важно только, чтобы она имела вид Z=f(x,y), т.е. чтобы была возможность составить однозначный алгоритм для вычисления координаты Z по заданным координатам X и Y. Самое интересное в этом графике - то, что скрытые детали - на самом деле скрыты. Все точки, которые находятся за гребнем или за вершиной - не показаны.

Алгоритм.

Давайте рассмотрим алгоритм, с помощью которого может быть достигнут желаемый эффект. Во-первых, надо отметить, что наша трехмерная поверхность изображается в два приема. На первом проходе изображаются все линии, параллельные оси X (рис. 2), а на втором проходе - линии, параллельные оси Y (рис. 3). Именно благодаря такому порядку изображения кривых и оказывается возможным скрыть невидимые детали.

Линии изображаются с некоторым шагом по X - Xh и по Y - Yh (см. рис. 4). Конечно, чем мельче шаг, тем детальнее будет проработано изображение, но слишком мельчить тоже не надо - существует некоторый оптимум, который можно установить методом проб и ошибок. Во всяком случае, параметры

Xh=(Xmax-Xmin)/20

и

Yh=(Ymax-Ymin)/20

выглядят достаточно удачными.

Если какая-то часть вычерчиваемой в текущий момент линии оказывается за ранее проведенной кривой, то она не изображается и есть достаточно простой прием, который позволяет в программе принять такое решение.

Если мы выстраиваем изображение в виде семейства кривых, начиная от ближайшей к наблюдателю и удаляясь от него назад (т.е. идем от т.т. X2 и Y2 к т.т. X1 и Y1, как показано на рис. 4), то фактически те точки текущей линии, которые оказывается на экране ниже, чем точки ранее изображенных линий и являются невидимыми и должны быть скрыты.

Чтобы решить этот вопрос программно, мы создаем в оперативной памяти буфер размером 256 байтов, а дальше действуем следующим образом. Поскольку экран "Спектрума" имеет в ширину 256 пикселов, то мы будем считать, что он образован как бы из 256-ти узких однопиксельных вертикальных столбцов. Каждому столбцу отведем по одной ячейке памяти в нашем буфере и теперь всякий раз, когда будем печатать на экране точку, будем смотреть, что же содержится в буфере для данного столбца. Если то значение, которое есть там - меньше, чем вертикальная координата экрана, в которой мы будем печатать точку, то новая координата запоминается в данном буфере и точка печатается. Если же хранящееся там значение больше, чем текущая вертикальная координата позиции печати, то точка должна быть скрыта и не печатается, а значение в буфере не изменяется.

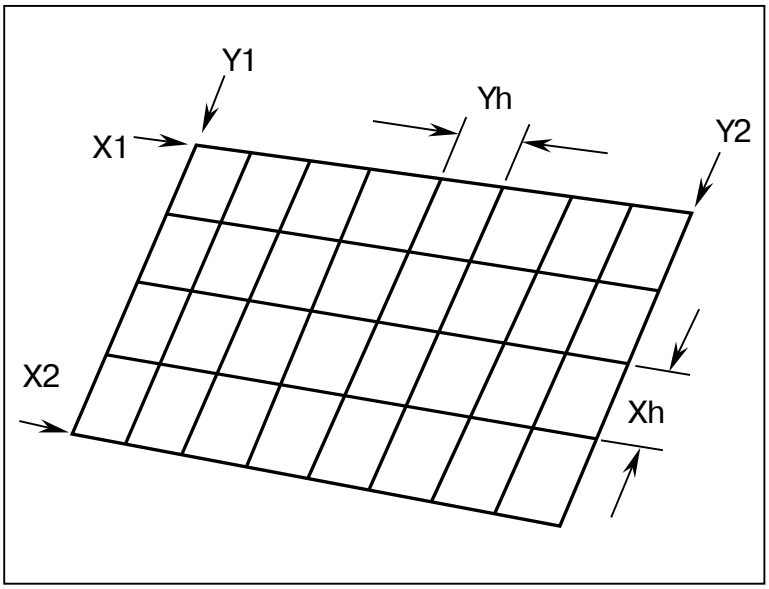


Рис. 4

Итак, алгоритм имеет следующий вид:

1. Задаем максимальные значения координат X2 и Y2.
2. Задаем минимальные значения координат X1 и Y1.
3. Определяем шаг по осям X и Y - Xh,Yh.

Xh =(Xmax-Xmin)/23
Yh=(Ymax-Ymin)/23

Мы специально делили здесь на "некруглое" число 23, а не на 20. Это помогает избежать прохождения через нулевую точку в том случае, если максимальные и минимальные параметры заданы симметрично относительно нуля. Все-таки неуютно себя чувствуешь, когда программа должна посчитать SIN(R)/R, когда R равно нулю. Хотя в принципе и этот случай можно было бы предусмотреть. Для тех, кто еще пока не изучал высшую математику, подскажем, что SIN(R)/R приближается к единице, когда R стремится к нулю.

4. Определяем масштаб по осям X и Y.

Для системы координат, показанной на Рис. 1...3 масштаб по X и Y - одинаков. Он зависит от ширины экрана и от угла между осями X, Y и осью Z. Для того, чтобы при любых

допустимых значениях X и Y точка умещалась бы на экране, нам необходимо избрать масштаб:

(X2-X1) + (Y2-Y1)*255/SQR(3)*2

5. Задаем масштаб для оси Z (его можно менять).

6. Создаем буферный массив из 256 элементов, обнуляем их.

7. Начинаем строить семейство кривых, "параллельных" оси X. Организуем цикл по Y от Y2 до Y1 с шагом Yh.

8. Внутри этого цикла строим кривую "параллельную" оси X. Организуем цикл от X2 до X1 с шагом по Xh.

9. Внутри этого цикла для текущих значений X и Y определяем Z по заданной формуле функции.

10. Полученный результат для Z умножаем на масштаб, получаем координату Z для графика.

11. Выполняем преобразование систем координат. По трехмерным координатам X, Y, Z находим значения X и Y для плоскости экрана.

Формулы для этого преобразования будут зависеть от того, какую проекцию трехмерной системы координат на плоскость Вы изберете. Другими словами, они зависят от того, под какими углами Вы смотрите на трехмерный объект. Для случая, показанного на Рис. 1.. Рис. 3 подойдут формулы:

X = SQR(3)* (Y-X)/2 + 127

Y = Z - (Y+X)/2 + 87

12. Мы готовы поставить на экране точку в координатах x, y. Но сначала проверим, что есть в буфере для данной координаты x. Если там значение меньше, чем y, то точку x, y на экране ставим и значение y обновляем в буфере, а если оно больше, то точка - невидима, мы ее не ставим и значение в буфере не обновляем.

13. Вычислив экранные координаты точки x, y, мы готовы соединить ее линией с предыдущей точкой x', y' (если наша точка не первая). Хорошо бы для этого воспользоваться командой БЕЙСИКа DRAW или процедурой изображения отрезков в машинных кодах, но делать этого, к сожалению, нельзя. Причина в том, что этот отрезок (или его часть) может быть невидимым. Значит, надо строить его по точкам и для каждой точки проверять по буферу видима она или нет. Поэтому опять же надо организовать цикл для изображения отрезка по точкам.

14. Теперь надо определиться с параметром этого цикла. Он может изменяться по горизонтали (по x), а может и по вертикали (по y). Надо понять, что больше - приращение dx (равное x-x') или dy (равное y-y'). То, которое больше, и следует принять в качестве параметра цикла.

15. Определившись с параметром, организуем цикл и внутри него вычисляем координаты текущих точек, проверяем для них y сравнением с буфером и, если точка видима, печатаем ее и обновляем буфер, а если нет, то не печатаем и не обновляем буфер.

Здесь есть маленькая хитрость, которая несколько усложняет жизнь программисту. Дело в том, что эти соединительные отрезки можно проводить слева-направо, а можно и справа налево. В принципе это все равно, но есть один нюанс. Допустим мы будем их проводить слева направо. Все будет в порядке, пока нам не придется провести круто падающий отрезок. На один шаг по x для него происходит несколько шагов по y. И бывает так, что одной координате x соответствуют несколько точек y. Если бы мы рисовали этот отрезок снизу вверх, все было бы в порядке, а при движении сверху вниз (слева-направо) ранее напечатанная точка может "блокировать" печать следующих, забив в буфере свое координату. Поэтому круто падающие отрезки программа должна строить наоборот - справа-налево. Тогда отрезок становится как бы не "падающим", а "восходящим".

16. Соединив две точки на экране, переходим к очередной точке, отстоящей на Xh, и возвращаемся на шаг 8.

17. Построив кривую, "параллельную" оси X, переходим к следующей, отстоящей от нее на шаг Yh. Возвращаемся на шаг 7.

18. Когда все семейство кривых, "параллельных" оси X построено, половина дела

сделана. Теперь надо построить семейство кривых, параллельных оси Y.

19. Для этого сначала переинициализируем буфер, обнулив все его значения, а затем повторим все то же, что мы делали для семейства кривых, идущих вдоль оси X (шаги 7 - 18). Правда, при этом вместо шагов по X будем делать шаги по Y и наоборот.

```
10 LET xmax = 10: LET ymax = 10
20 LET xmin =-10: LET ymin = -10
30 LET xh = -(xmax-xmin)/23: LET yh = -(ymax-ymin)/23
40 LET scale = 255/SQR(3)*2/((xmax-xmin) * (ymax-ymin))
50 LET zscale = 40
60 DIM c(256): FOR i = 1 TO 256: LET c(i)=0: NEXT i
70 FOR y=ymax TO ymin STEP yh: LET y1 = y*scale
80 FOR x=xmax TO xmin STEP xh: LET x1=x*scale
90 GO SUB 5000
100 IF x=xmax THEN GO SUB 8100: NEXT x
110 LET dy=yt-yold: LET dx=xt-xold
120 IF ABS(dy) >= ABS(dx) THEN GO SUB 6000: GO TO 140
130 GO SUB 7000
140 NEXT x
150 NEXT y
155 FOR i=1 TO 256: LET c(i)=0: NEXT i
160 FOR x=xmax TO xmin STEP xh: LET x1 = x*scale
170 FOR y=ymax TO ymin STEP yh: LET y1 = y*scale
180 GO SUB 5000
190 IF y=ymax THEN GO SUB 8100: NEXT y
200 LET dy=yt-yold: LET dx=xt-xold
210 IF ABS(dy) >= ABS(dx) THEN GO SUB 6000: GO TO 230
220 GO SUB 7000
230 NEXT y
240 NEXT x
250 STOP
4997 REM
4998 REM*********************************************
4999 REM
5000 LET r= SQR(x*x + y*y) : LET z = SIN(r)/r
5010 LET z = z*zscale
5020 LET xt=127+SQR(3)*(y1-x1)/2
5030 LET yt=87+z-(y1+x1)/2
5040 RETURN
5997 REM
5998 REM*********************************************
5999 REM
6000 IF dy<0 THEN GO TO 6100
6010 FOR i=0 TO dy
6020 GO SUB 6500
6030 NEXT i
6040 GO SUB 8100: RETURN
6100 FOR i=dy TO 0
6110 GO SUB 6500
6120 NEXT i
6130 GO SUB 8200: RETURN
6497 REM
6498 REM*************************************************
6499 REM
6500 LET xt = xold+dx/dy*i
6510 LET yt=yold + i
6520 GO SUB 8000: RETURN
6997 REM
6998 REM*************************************************
6999 REM
7000 IF dx<0 THEN GO TO 7100
7010 FOR i=0 TO dx
7020 GO SUB 7500
```

```
7030 NEXT i
7040 GO SUB 8100: RETURN
7100 FOR i=dx TO 0
7110 GO SUB 7500
7120 NEXT i
7130 GO SUB 8200: RETURN
7497 REM
7498 REM ************************************************
7499 REM
7500 LET xt=xold+i
7510 LET yt = yold+dy/dx*i
7520 GO SUB 8000: RETURN
7997 REM
7998 REM************************************************
7999 REM
8000 IF yt > c(xt+1) THEN LET c(xt+1)=yt: PLOT xt,yt
8010 RETURN
8097 REM
8098 REM************************************************
8099 REM
8100 LET xold=xt: LET yold=yt: RETURN
8197 REM
8198 REM************************************************
8199 REM
8200 LET xold=xold+dx: LET yold=yold+dy: RETURN
```

СПИСОК ПРОГРАММНЫХ ПЕРЕМЕННЫХ

xmax - максимально-допустимое значение координаты X (задаетсяпользователем).
ymax - то же для координаты Y.
xmin - минимально-допустимое значение координаты X (задается пользователем).
ymin - то же для координаты Y.
xh,yh - шаг между узлами сетки.
scale - масштаб по координатам X и Y (зависит от углов в пространстве, под которыми наблюдатель смотрит на трехмерный объект). Рассчитывается исходя из соображений оптимального использования плоскости экрана.
zscale - масштаб по оси Z (задается пользователем "по вкусу", но так, чтобы изображение по вертикали не вышло за пределы экрана). Возможно и автоматическое определение zscale в программе, во для простоты это не было сделано.
c(256)- буферный массив на 256 элементов.
x1, y1 - отмасштабированные значения трехмерных координат X и Y.
r - математический комплекс, нужный для вычисления Z.
x,y - текущие трехмерные координаты X и Y в узлах сетки.
xt,yt - текущие экранные координаты (двумерные).
xold,yold - экранные (двумерные) координаты предыдущего узла.
dx,dy - приращения экранных координат на очередном шаге (расстояние на экране между узлами).

* * *

Вот практически и весь алгоритм. Его описание выглядит страшнее, чем текст программы на БЕЙСИКе (см. распечатку), и это не случайно - ведь БЕЙСИК гораздо лучше подходит для описания алгоритмов и программистских идей, чем нормальный человеческий язык.

Конечно, скорость работы этой программы на БЕЙСИКе оставляет желать лучшего, но для иллюстрации самой концепции он неплох.

Мы попробовали - у нас получилось время работы программы что-то порядка пятнадцати минут, но тем не менее не пожалейте этого времени, поэкспериментируйте с

графиком.

Вы можете менять максимальные и минимальные значения X и Y. Вы можете менять масштаб по Z. В наших экспериментах мы получали кроме представленной на рис. 1 "шляпы" еще и "верблюда", "русалку", "замок в горах", "пепельницу", "ковер-самолет" и пр. и пр.

Более того, Вы можете менять и саму функцию, исследуя другие поверхности. Если Вы в приведенном нами примере замените вторую степень при X и Y на четвертую, то почувствуете, как холодок бежит по спине, когда компьютер изобразит очень натуральную свежую могилку с не менее натуральным каменным валуном в изголовье. Хочется пожелать Вам найти что-либо менее мрачное, особенно если вы работаете с компьютером по ночам.

Те же, кто предпочтут использовать для программирования машинный код, получат прекрасные результаты, но надо учесть, что программа выполняет большой объем чисто математических вычислений (это вообще характерно для трехмерной векторной графики). Здесь и масштабирование (чтобы график аккуратно занимал плоскость экрана) и пересчет из одной системы координат в другую (из трехмерной системы координат в двумерную систему координат плоскости экрана) и, конечно же, расчет самой функции Z=f(X,Y).

Процессор Z-80 не может оперировать с действительными числами, не может выполнять математических расчетов и здесь используют программирование в кодах калькулятора.

Пример программы в машинном коде мы здесь не даем, оставив для тех, кто интересуется, возможность ознакомиться с ним в книге. Подробный комментарий всех входящих процедур явится хорошим пособием для тех, кому необходимо программирование в кодах калькулятора.

Маленькие хитрости

В этом разделе мы коснемся некоторых путей повышения быстродействия часто встречающихся операций.

Так, например, в предыдущей статье мы с Вами затронули маленький вопрос об очистке 256-байтного буфера. Давайте посмотрим, как бы мы делали эту операцию, если бы программировали в машинном коде.

Мы бы загрузили в регистровую пару HL адрес NN, с которого начинается наш буфер. Затем в регистре B организовали бы счетчик на 256 байтов (FFH), обнулили бы аккумулятор командой XOR и затем в цикле поместили бы содержимое аккумулятора в ячейки буфера, на которые указывает HL. При этом на каждом шаге увеличивали бы HL на единицу.

Мы специально пишем столь подробно об этих элементарных вещах, потому что рассчитываем, что нас могут читать и те, кто только подумывает об освоении машинного кода.

```
       LD HL,NN          (10)
       LD B, FFH         (7)
       XOR A             (4)
LOOP   LD (HL),A         (7)*256
       INC HL            (6)*256
       DJNZ LOOP         (8)*256
```

Давайте посмотрим сколько времени займет это мероприятие. В скобках проставлено время, необходимое для выполнения каждой из приведенных команд процессора. Это время измеряется в тактах работы процессора, а пересчет в секунды возможен, если знать частоту задающего генератора в вашем компьютере. Но нам достаточно и тактов, чтобы сравнить между собой различные приемы.

Итак, приведенный выше пример займет: 21 + 21*256 = 5397 тактов, т.е. в среднем 21,08 такта на очистку одного байта в буфере.

Можно ли быстрее? Скорее проще, чем быстрее. Те, кто знают машинный код, осведомлены о наличии команды LDIR, которая служит для автоматической очень быстрой переброски блоков данных из одной области памяти в другую. Эта команда перебрасывает блок байтов, длина которого установлена в регистровой паре BC из области, начинающейся с адреса, установленного в регистровой паре HL в область, на начало которой указывает содержимое DE. Остается открытым вопрос, а как можно использовать LDIR для очистки, ведь если эта команда может быстро перебросить блок нулевых байтов, то ведь такой блок надо сначала создать - в общем придем к тому, от чего ушли (мы не можем в общем случае рассчитывать на то, что где-то в компьютере есть пространства с нулевым содержимым ячеек, откуда можно черпать "пустые" массивы).

Оказывается, использовать LDIR все же можно, хоть и делается это несколько необычно. Рассмотрим пример:

```
LD HL, NN         (10)
LD DE,NN+1        (10)
LD BC, 00FFH      (10)
LD (HL),B         (7)
LDIR              (21)*256
```

То, что здесь происходит, может показаться чепухой. В HL установили адрес начала нашего буфера, в DE - адрес второго байта буфера, в BC - счетчик на 255. Командой LD (HL),B очистили первый байт буфера, а потом зачем-то передвинули все содержимое буфера на один байт вверх. Ну получим в итоге, что и второй байт станет нулевым, а дальше-то что?

Вся хитрость состоит в том, что во время работы команды LDIR содержимое HL, DE и BC не остаются неизменными. После переброски каждого очередного байта HL и DE

увеличиваются на единицу, а BC на единицу уменьшается. Благодаря этому после переброски первого байта во второй, DE уже укажет на третий, а HL "подхватит" нулевой байт из второй ячейки и так далее. В общем, весь буфер будет вычищен.

Итак, при работе с командой LDIR мы затратим: 37+21*256= 5413 тактов - это хоть и не быстрее, но, главное, изящнее. Получается в среднем 21.14 такта на байт.

Те, кто любят "маленькие хитрости", наверное, ждут, что мы предложим что-либо еще более скоростное, чем LDIR. Возможно, что большинство профессионалов сказали бы, что это невозможно, ничего быстрее LDIR для манипуляции с большими блоками памяти не бывает.

Но нет, оказывается, бывает, причем не просто быстрее, а быстрее в несколько раз. Есть одно гениальное решение, которое мы не постесняемся так назвать, поскольку не мы его придумали, а "выудили" его в результате анализа машинного кода программы "STARION" фирмы "Melbourne House". На поиски натолкнул тот факт, что программа в части межзвездных сражений не уступает "ELITE", а вот графика более динамичная, гладкая и плавная. Там это решение применяется не для очистки маленьких таблиц, а для освежения всей экранной памяти, а ее размер велик и там каждая тысячная доля секунды на счету.

Рассмотрим используемый алгоритм на нашем примере с очисткой 256-байтного буфера.

```
        LD (MM),SP        (20)
        LD HL,0000        (10)
        LD SP,KK          (10)
        LD B,80H          (7)
LOOP    PUSH HL           (11)*128
        DJNZ LOOP         (8)*128
        LD SP,(MM)        (20)
```

Вся хитрость состоит в использовании стека. Исходное положение вершины стека надо запомнить в каком угодно адресе MM затем приготавливаем регистровую пару HL - обнуляем ее. Новый стек организуем в нашем буфере. Но надо помнить, что стек в памяти компьютера "растет" сверху вниз. Поэтому, чтобы очистить весь буфер, мы должны начать не с его начала, а с конца. LD SP, KK - прогружает в качестве вершины стека конец нашего буфера. Счетчик байтов создаем в регистре B, как обычно, но поскольку здесь байты будут перебрасываться не по одному, а парами, то счетчик надо настраивать не на 256, а на 128 - (LD B,80H) перемещений.

Все подготовительные операции сделаны. Теперь в цикле 128 раз помещаем содержимое HL на вершину стека, то есть обнуляем буфер. Закончив операцию, надо не забыть восстановить старое значение указателя стека, временно сохраненное в ячейке MM.

Всего мы затратили:

67+19*128 = 2499 тактов, т.е. примерно по 9.76 такта на байт!

Встает старый вопрос: "А нельзя ли еще быстрее?"

Пожалуйста, отвечаем мы. Идею Вы уловите сами очень легко и поймете, в каком направления надо идти, из следующего примера:

```
        LD (NN),SP        (20)
        LD HL,0000        (10)
        LD SP,KK          (10)
        LD B,20H          (7)
LOOP    PUSH HL           (11)*32
        PUSH HL           (11)*32
        PUSH HL           (11)*32
        PUSH HL           (11)*32
        DJNZ LOOP         (8)*32
        LD SP,(MM)        (20)
```

Всего имеем: 67+52*32 = 1731 - по 6,76 такта на байт.

Подумать только, с чего мы начинали! А ведь это еще не предел. При приличном

количестве команд PUSH HL, как в STARION'е, можно работать быстрее, чем с LDIR в четыре раза! И, конечно, такая скорость не нужна для освежения буфера - на самом деле это нужно для работы с экраном.

Прощаясь, признаемся, что там же мы "выудили" прием не только очистки, но и перестроения экрана, работающий в два раза быстрее, чем LDIR. В основу положена та же идея манипуляций со стеком, хотя сделано это намного сложнее. При случае мы вернемся к этому вопросу, хотя многие наверное уже и сами сообразят, как это должно быть. У нас ведь народ тертый, ему только намекни…

ОШИБКИ ПЗУ

Обзор по материалам зарубежной печати

В этой статье рассмотрены ошибки и неточности, имеющиеся в ПЗУ "СПЕКТРУМА". Может быть не все они, строго говоря, и являются ошибками, а просто особенностями компьютера, но о них надо знать и уметь обходить в тех случаях, когда они могут помешать нормальной работе.

1. Ограничение по использованию регистровой пары IY.

Эта ошибка (неточность) связана с тем, как в "Спектруме" обрабатываются маскируемые прерывания. Вы знаете, что обычно компьютер 50 раз в секунду приостанавливает исполнение своей текущей программы и обращается по адресу 0038H = 56 DEC для запуска процедуры обработки маскируемого прерывания. Эта процедура увеличивает на единицу показания системных часов компьютера (трехбайтную системную переменную FRAMES) и вызывает подпрограмму сканирования клавиатуры в поисках нажатой клавиши. Благодаря этому и возможен диалог между Вами и компьютером.

Ошибка связана с тем, что увеличение на единицу системных часов производится некорректно. Младшие два байта увеличиваются командой INC HL и с ними все в порядке, но когда они переполняются и надо увеличить старший байт, расположенный по адресу 5C7AH (33674), для этого используется команда процессора

```
INC (IY+40)
```

Программист, который писал эту процедуру, предполагал, что во время ее работы в регистровой паре IY должно быть значение 5C3AH (23610) - указание на системную переменную ERR-NR. в принципе так оно и должно бы быть.

Когда вы вызываете подпрограмму, написанную в машинном коде с помощью RANDOMIZE USR nn, процедура калькулятора, занимающаяся обработкой функции USR nn (она расположена по адресу 34B3H) автоматически сохраняет на стеке адрес 2D2EH. Поэтому, когда Вы вернетесь по RET из своей подпрограммы, то попадете сначала в адрес 2D2BH (процедура STACK_BC). Здесь восстанавливается нормальное содержимое пары IY = 5C3AH.

Таким образом, у Вас все было бы в полном порядке, если бы не одно "но". Ведь во время работы самой Вашей процедуры тоже может пройти системное прерывание. И если Вы изменили содержимое IY, то вместо приращения системных часов получите что угодно.

Ошибка ПЗУ состоит в том, что в процедуре обработки маскируемого прерывания следовало бы выставлять в IY значение 5C3A, а только потом наращивать старший байт системных часов, а перед выходом восстанавливать в IY Ваше значение.

Достаточно обидно иметь в своем распоряжении регистровую пару IY и не иметь возможности ею воспользоваться. И пользоваться ею можно, если при этом соблюдать некоторые меры предосторожности.

1. Если Вы намерены использовать регистровую пару IY, то возьмите себе за правило прежде, чем что-либо изменять в этой паре, отключать системные прерывания командой процессора DI - DISABLE INTERRUPTS, а перед выходом из процедуры восстанавливать их командой EI - ENABLE INTERRUPTS.

2. Может быть, Вам захочется даже создать свою собственную процедуру для обработки прерываний.

2. Особенности регистровой пары H'L' (альтернативной).

Эта ошибка коснется тех, кто программирует в машинных кодах, а еще точнее - тех, кто в своих программах объединяет БЕЙСИК и машинный код, а так поступают очень многие.

Они пишут логику программы на БЕЙСИКе, а операции, требующие большой скорости, пишут в машинном коде. Вероятность споткнуться об эту ошибку практически стопроцентная. Рано или поздно она испортят Вам кровь, хотя избежать ее очень просто.

В двух словах, суть состоит в том, что если Ваша процедура, написанная в маш. коде изменит содержимое регистровой пары H'L' (альтернативной), то в БЕЙСИК Вы уже не вернетесь и программа сбросится или зависнет.

Почему так происходит? Чтобы ответить на этот вопрос, надо знать, как происходит вызов пользовательских машиннокодовых процедур. Вы конечно знаете, что они вызываются командой RANDOMIZE USR nn, где nn - адрес ее старта. В принципе вместо RANDOMIZE USR может использоваться RESTORE USR, PRINT USR и т.п., сейчас нам это все равно. Суть состоит в том, что когда интерпретатор БЕЙСИКа встретит USR nn, то он должен рассчитать чему равно nn и для этого вызывает встроенный калькулятор. Это делает процедура ПЗУ, которая занимается расчетом выражений. По адресу 2756H она содержит вызов калькулятора, а по адресу 2758H - выход из калькулятора.

При вводе в калькулятор содержимое вершины машинного стека (а это есть адрес возврата из калькулятора) запоминается в альтернативной регистровой паре H'L'. Это делает процедура, находящаяся по адресу 3362H.

Затем процедура калькулятора, занимающаяся обслуживанием функции USR nn (34B3H) помещает на вершину стека адрес 2D2Bh и адрес "nn".

По команде RET со стека снимается адрес "nn" - выполняется переход в Вашу машиннокодовую процедуру.

Когда она отработает, по ее команде RET со стека снимется адрес 2D2BH и выполнится переход по этому адресу в ПЗУ. Мы писали, рассматривая предыдущую ошибку о том, что расположенная там процедура восстанавливает "стандартное" значение в регистровой паре IY, но она ничего не делает, чтобы восстановить содержимое альтернативной пары H'L', а ведь она тоже могла быть нарушена при исполнении пользовательской процедуры.

Таким образом, исполнение пользовательской процедуры в машинных кодах начинается с вызова системного калькулятора для расчета адреса старта, а кончается выходом из него. Выход из калькулятора производится помещением на стек того, что хранилось в H'L' и переходом по этому адресу. Это делает процедура, обслуживающая команду калькулятора end-calc, расположенная по адресу 369BH. И содержаться в этот момент в H'L' может только адрес 2758H.

Итак, если Ваша процедура нарушает содержимое регистровой пары H'L' (альтернативная) и Вы рассчитываете вернуться в БЕЙСИК, то потрудитесь перед выходом из процедуры восстановить в этой паре значение 2758H, иначе ничего у Вас не получится.

3. Особенности пользовательской функции FN.

Недоразумение, связанное с пользовательскими функциями может коснуться и тех, кто программирует на БЕЙСИКе и тех, кто программирует в машинном коде, но скорее всего с ним столкнутся опять те же энтузиасты, которые используют и БЕЙСИК и машинный код одновременно, причем пользуются для передачи параметров из БЕЙСИКа в маш. код пользовательскими функциями. Это весьма удобный и прогрессивный метод и мы подробно осветили его в первом томе, посвященном графике - "Элементарная графика".

Суть этой ошибки состоит в том, что если во время исполнения пользовательской функции произойдет какое-либо перемещение программной области БЕЙСИКа вверх или вниз, то будет выдано сообщение C; Nonsense in BASIC.

Конечно, не так-то просто выполнить перемещения БЕЙСИК-области и "нарваться" на такую ошибку, но это вполне возможно, если Вы используете USR внутри вашей функции.

На практике это может означать вот что. Если Вы, например, зададите пользовательскую функцию FN D(m,n) = USR NN, предназначенную для удаления из БЕЙСИК-программы строк m и n, то она у вас работать не сможет.

Причина появления ошибки - в статическом характере системной переменной CH ADD (23645=5C5DH), которая при работе интерпретатора содержит адрес следующего

интерпретируемого символа. На время исполнения пользовательской функции она "сохраняет" свое состояние на стеке и, если во время работы функции все адреса, имеющие отношения к БЕЙСИКУ "поплывут", то после окончания ее работы интерпретатор не сможет продолжить исполнение БЕЙСИК-программы.

4. Ошибка деления.

Эта ошибка встречается настолько часто, что о ней даже и не говорят, как об ошибке деления, а считают ее ошибкой округления. Однажды в разделе "ФОРУМ" мы отвечали на письмо читателей, связанное с этой ошибкой (см. "ZX-РЕВЮ-91, с. 222).

Кратко проиллюстрировать ее можно на примере:

```
IF 1/2 <> 0.5 THEN PRINT "Ku-Ku"
```

Попробовав, Вы сами убедитесь, что 1/2 не равна 0.5.

Конечно, в инженерных расчетах следует вводить понятие точности вычислений и сравнивать действительные числа между собой только в пределах установленной точности, но все-таки где-то что-то в ПЗУ не совсем то, раз компьютер выдает такие непонятные результаты.

Ошибка содержится в процедуре калькулятора, выполняющей деление и связана с неправильным округлением результата. Не вдаваясь в подробности как происходит деление двух действительных чисел, скажем только, что процедура работает с 32-мя битами, что и определяет точность числа. Но "за кадром" она учитывает еще 33-ий и 34-ый биты для правильного округления. Так вот, именно при добавлении 34-го бита и происходит ошибка - он не учитывается из-за неправильного перехода в адресе 3200H. Там записана команда JR Z, 31E2, а должно быть JR Z, 31DB.

Наилучший путь обхода - установить требуемую точность для своих расчетов, например:

```
LET eps = 0.0000001
```

и затем вместо

```
IF X = Y
```

употреблять:

```
IF (X-Y) < eps
```

5. Ошибка "-65536".

Это довольно известная ошибка. Она легко вызывается оператором:

```
PRINT INT(-65536)
```

который дает -1.

Причина скрыта в процедурах калькулятора, которые работают с пятибайтной формой представления чисел. Одни из них считают, что целые числа могут быть только в диапазоне от -65535 до +65535.

Некоторые, например процедура, выполняющая сложение, полагают, что и -65536 - тоже допустимое целое число, в частности, возможно, что сложив -65000 и -536, можно получить число -65536, которое внутри компьютера будет представлено не как действительное, а как целое число.

Процедура же INT - одна из тех, которые " разоблачают" эту двусмысленность.

В известной книге Я. Логана и Ф. О'Хары "The Complete SPECTRUM ROM Disassembly" подробному разбору этой ошибки посвящена целая глава приложения. Объяснения достаточно мудреные для непрофессионалов и мы отошлем тех, кому эти знания существенно важны, к этой книге.

6. Ошибка оператора PLOT.

Может быть, это и не ошибка в том смысле, что к фатальным последствиям она не приведет. Дело в том, что если в операторе PLOT X,Y Вы зададите координаты X и Y отрицательными числами, то все равно получите точку на экране.

На самом деле получается так, что команда PLOT печатает точку не в координатах X и Y, а в координатах ABS(X),ABS(Y). Причина - в том, что процедура ПЗУ PLOT (22DCH)

вызывает процедуру, перемещающую целые числа из вершины стека калькулятора в регистровую пару BC (процедуру STK_TO_BC - 2307H). Но после того, как STK_TO_BC отработает, не учитывается, что она еще поместила в регистровую пару DE знаки тех чисел, которые пошли в BC, и при возвращении в процедуру PLOT надо было бы проверить DE на знак, что не было сделано.

7. Ошибка первого экрана в компьютерах 128K.

Мы уже писали об этой ошибке в предыдущем выпуске "ZX-РЕВЮ" на стр. 156-159. Напомним, что 128-килобайтные машины имеют две экранных области памяти. Нулевой экран расположен в обычных адресах - 4000H, а дополнительный первый экран - в адресе 7C000H. Их можно переключать. Команда POKE 23388,24 включит первый экран, а команда POKE 23388,16 включит нулевой экран.

Ошибка происходит в тех случаях, когда пользователь работает с верхними областями памяти этой машины, как с электронным RAM-диском и сохраняет в ней данные в виде многочисленных файлов. Файлы "располагаются" в памяти снизу вверх и в это же время сверху вниз в памяти растет каталог этих файлов. В компьютере ничего не сделано для того, чтобы каталог или сами файлы не "перехлестнулись" с первым (дополнительным) экраном.

Чтобы этого не произошло, не храните более 216 файлов, как бы малы они не были, чтобы экран не нарушился каталогом. Вам также нельзя иметь в сумме более 64K данных в этих файлах, иначе экран-1 будет испорчен самими файлами.

(Окончание в следующем выпуске).

Blinky's

Сегодня мы представляем Вашему вниманию экспертную проработку программы BLINKY'S. Ее сделал наш читатель из г. Днепропетровска - Садошенко Денис. Вы сами увидите, что исполнена она в необычном жанре - назовем его "компьютерной новеллой". Денис пишет, что его друзьям и близким идея очень понравилась. Понравилась она и нам и мы надеемся, что понравится и Вам, уважаемые читатели.

Программа относится к жанру аркадных адвентюр (ARCADE/ADVENTURE). Должны сказать, что с экспертизой этого жанра у нас всегда были проблемы. И дело вовсе не в том, что разобраться с этими играми и пройти их от начала до конца не всегда просто. У нас так много поклонников этого жанра, что всегда есть кто-то, кто разобрался с той или иной игрой. Трудность состоит в том, как представить результаты своего исследования. Ведь если просто расписать подробный порядок прохождения игры, то и читать и играть будет неинтересно.

В свое время мы пробовали представить описание аркадной адвентюры в виде многоуровневой системы подсказок (см. "PIJAMARAMA" в ZX-РЕВЮ-91, стр. 255) и полагали этот вариант удачным, но сейчас должны признать, что идея Д. Садошенко выглядит привлекательнее.

Нам кажется, что ему удалось найти тот баланс, который позволил сочетать и художественную привлекательность и фактическое содержание. Надеемся, что и наши читатели это оценят.

Садошенко Д.
г. Днепропетровск.

Здравствуйте, дорогие читатели!

Это еще одна запись в моем дневнике. Как Вы знаете, меня зовут BLINKY'S и я, если честно, не совсем живой человек, я - привидение. Сейчас вот сижу у любимого камина, в котором догорают сосновые брусочки и вспоминаю свое прошлое.

Давным-давно жил я в вашей жизни, где есть зеленая трава, деревья, прекрасный чистый воздух и прочие блага. Здесь, в потустороннем мире такого не увидишь. Служил я у своего хозяина - короля придворным шутом. Мы с его слугами жили в старинном замке, который достался королю от деда. При дворе жилось легко и привольно. У меня была постоянная работа и постоянная еда. А большего и не надо. Трудности начинались, когда в замок приезжал очередной гость, ведь наш король был очень общительным человеком. Приходилось работать в поте лица, веселя гостей, за что нередко получал в качестве подачки еду и мелкие монетки. Так бы и продолжалась моя беззаботная жизнь, но вот беда - не любила меня старая королева, вторая жена короля. Может, и не напрасно не любила, иногда я вместо того, чтобы веселить гостей, развлекался сам, делая разные гадости королю и королеве. Например, на кухне я перепутал все специи и рассовал их по баночкам с разными надписями, так что в баночке с этикеткой "соль" можно было увидеть перец, а с этикеткой "сахар" - корицу. Это было так здорово!

За эту в принципе безобидную выходку меня выпороли. Вдобавок я еще и лишился работы на неделю, так как при виде меня, покрытого синяками и ссадинами, все плакали, а не смеялись. Отношение короля ко мне ухудшалось с каждым днем. А я день ото дня все больше наглел. Дошел до того, что стал изображать его во всех подробностях - с короткой левой ногой, плешью и тупым выражением лица. Королю донесли, и меня бросили в тюрьму.

На следующий день состоялся местный суд. Прокурором была королева и этим все уже сказано. Меня приговорили к смерти, и в тот же день приговор привели в исполнение…

Вот так я оказался среди мертвых. И нахожусь здесь уже более 4 лет. Каждый год в день моей смерти здесь отмечается небольшой праздник. Прилетает много моих друзей, которых я знал давно, и которые умерли до меня. И каждый год ровно на три часа (с 12 до 3 ночи) меня отпускают к живым.

Обычно я прилетаю в родной дворец и шляюсь по нему. Меня никто не видит, а я вижу всех. Но по ночам, когда все спят, из темных углов вылезают пауки, мыши, улитки. Вылетают вампиры и осы. Проклятый замок стареет с каждым днем - с потолка капает вода, сыпется песок и падают тяжелые плиты. Возле дворца есть ров с водой. Там живут зубастые пираньи, которые никогда не спят. Даже ночью они плавают, отражая тусклыми глазами рассеянный свет далекой луны.

Очень интересно и опасно ночью во дворце. Целый год я жду этой ночи, и каждый раз она себя оправдывает. Вот и та история, которую я хочу вам рассказать, была очень смешно и хитро задумана.

Я решил разбудить и сильно напугать старого короля. Ну что еще может сделать привидение!? Коварно выбрал время - 3 часа утра. В этот час он наиболее крепко спит, а у меня как раз кончается "отпуск".

Сделать это вроде бы и нетрудно, но пришлось попотеть. Каждое прикосновение к живому существу, будь то человек или мерзкий паук, отбирает у меня немного магической энергии. Если энергии не хватит, или я задержусь в замке до рассвета, тогда одним привидением на земле станет меньше. Меня просто не будет ни среди живых, ни среди мертвых. Вот так.

Но я был полон оптимизма и от всей души взялся за дело. На эту ночь, в качестве подарка, мне дали 4 дополнительных жизни. Если я использую их все до 3 утра, то никогда больше не вернусь к своим мертвым друзьям. Однако, во время прошлых посещений замка я научился избегать неприятных контактов, ведущих к повторной смерти. Даже привидение можно убить. Поэтому я был очень осторожен в этот раз. Мне совсем не хотелось раствориться в ночной воздухе из-за малейшей небрежности.

Почти всегда после перемещения я оказываюсь в подвале, в самом темном месте. А король спит наверху в своей любимой башне. Приходится проходить длинный и опасный путь. Во дворце летать я не могу, т.к. там очень тесно. Могу только ходить и прыгать.

Замок кишит тайными ходами - например, есть подводный ход, ведущий в замок снаружи. Нечисть даже построила там свою систему сообщения. Лифтом служит унитаз, а ключом - туалетная бумага, причем, где бы ты ни сел в лифт-унитаз, тебя всегда выбросит на месте старта - в подвале. Всего имеется два унитаза и три рулона бумаги.

В некоторых комнатах нет освещения. В некоторых из пола торчат острые колья. Практически везде ползают пауки. В общем, на каждом шагу меня поджидает смерть. И я начал было уже сетовать на малое количество дополнительных жизней, когда оказался во дворце.

Первым делом я огляделся и заметил небольшое послание на стене. Так как я видел его и раньше, то проигнорировал его и пошел из подвала направо. Тут я наткнулся на ядовитых пауков, прыгавших туда-сюда. Неприятности добавляли падающие с потолка плиты. Стремительно пробежав эту зловещую комнату, я попал в другую, не менее ужасную - здесь тоже были пауки, а из пола торчали острые колья. Зато здесь была и нужная мне туалетная бумага. Я взял ее, поднялся по плитам наверх и стал обладателем магического напитка. Какой-то бедняга, бродивший здесь до меня, оставил его и, по-видимому, уже никогда за ним не вернется.

Вернувшись на место старта, что стоило мне немного энергии, я поднялся выше, потом еще выше, вошел в комнату, где ползала улитка и взял там мешок с крупой. Так как больше трех предметов одновременно я поднять не могу, то мне стало тяжеловато. Спустившись вниз, я прошел налево через картинную галерею с одной-единственной картиной и добрался до входа на второй этаж. Но лестницы не оказалось - сломалась, наверное. Зато там был котел, в котором варилось какое-то варево.

Осторожно миновав улиток и ядовитых ос, я запрыгнул на котел и положил в него крупу и магический напиток. Не знаю, чего я от этого ждал, наверное ничего, просто устал их носить на себе. Так оно и вышло - ничего и не произошло. Разочарованно вздохнув, я отправился дальше.

Вернувшись в ту комнату, где я брал туалетную бумагу, я пошел направо. Перепрыгнул колья, обошел пауков и очутился в новом месте. Здесь была банка с клубничным джемом. Когда-то я очень любил этот джем, но сегодня брать его не стал. Решил, что прихвачу на обратном пути и правильно сделал. Начались такие неприятности, что мне стало не до джема.

Сначала я упал в шахту и при этом чуть не наступил на ядовитых крыс. Со страха сунулся было налево, но там была такая темнота, что идти дальше было бы чистым безумием. Пришлось собраться с духом и осторожно обходить крыс. Так я оказался в комнате с рыцарскими доспехами. Здесь мне наконец повезло - я нашел фонарик. Поспешно схватив его, я бросился в темную комнату и осветил мрачную обстановку заброшенного подвала.

Луч фонарика выхватил из темноты старую засушенную рыбу. Прихватив ее на всякий случай, я осторожно пошел дальше. В соседней комнате меня ожидали бутылка с соком и целый ряд острых кольев. Так как сок я все равно взять не мог, то со спокойной душой проследовал дальше, в комнате со статуей были плиты, а чуть ниже - пауки. Осторожно обойдя их, я очутился в зловещей комнате с двумя проломами в полу. Между проломами прыгал паук и лежала кассета. Логически поразмыслив, я подумал, что кассета мне вроде бы ни к чему. Оставалось падать в один из проломов. Я выбрал дальний и, как оказалось, - не напрасно.

Недолгое падение в шахту, и я очутился в самом низком месте замка. Здесь лежала туалетная бумага. Она у меня уже есть и новую я брать не стал, а решил запомнить этот путь, чтобы вернуться за ней в случае необходимости. Поскольку больше идти было некуда, я пошел налево и увидел... унитаз. Я уже упоминал о двойном назначении этого замечательного устройства. Подойдя к нему, я запрыгнул на сиденье, дернул за ручку и меня доставили на место старта. Бумага при этом у меня исчезла, а рыба и фонарик - остались.

Отнеся рыбу к котлу, я вернулся известным путем за бутылкой сока. Вооружился туалетной бумагой, воспользовался унитазом, оказался на месте старта и положил сок в котел. На этот раз мне повезло, в котле все забулькало, забурлило и я обрел возможность ЛЕТАТЬ!!! Взлетев, я оказался на втором этаже.

* 2 этаж *

Я сразу пошел налево. Осторожно обойдя в следующей комнате пауков и увернувшись от падающих плит, я нашел непонятный глаз, оставшийся здесь от какого-то доисторического чудовища. Поднявшись по плитам наверх, я увидел карликовую собачку, но брать ее не стал. Так я пробирался все выше и выше, стараясь при этом держаться правее. Дорога привела меня на крепостную стену. Прихватив невесть откуда взявшийся там гамбургер и увернувшись от летучих мышей, я снова спустился в замок.

В комнате, левее моей, я увидел цель своего путешествия - спящего короля. И здесь меня осенила идея. Моя месть будет еще более страшной, если я разбужу его с помощью будильника. Насколько я помнил, в мире живых это устройство считалось самым кошмарным после бормашины. Где взять бормашину я не знал, а вот будильник в замке когда-то был. Я сам устраивал его веселые похороны на старом кладбище, развлекая короля и гостей.

Пойдя направо, я перепрыгнул улитку и оказался в комнате с рыцарем. Там тоже было какое-то послание, но читать его мне было некогда. Я поспешно спустился вниз к котлу, положил в него глаз и гамбургер, вернулся на дворцовую стену и взял там баллон с кислородом.

Возвратившись обратно, я не стал пробираться к котлу, а спустился в шахту на одну

комнату вниз. Здесь я стал счастливым обладателем вкусной конфеты. Только теперь я пошел к котлу. Выложил в него и баллон и конфету, но ничего не произошло. Продолжив поиски, я нашел этажом выше воздушный шарик. С ним я направился к выходу на первый этаж - в ту комнату, где стоит первый котел. Но как я ни запрыгивал на него в надежде пробраться дальше в глубины замка, меня всегда относило наверх. Тут мне пришлось расстаться с одной моей дополнительной жизнью. До сих пор думаю, что что-то я сделал не так. Хоть все и кончилось хорошо, но гложет червячок сомнения мою неуспокоенную душу.

Очнулся я в картинной галерее. Обрадовавшись, побежал к комнате с доспехами, где в прошлый раз брал фонарик. Упав вниз, я взял третий рулон туалетной бумаги, перепрыгнул паука и проследовал дальше.

В комнате, куда я упал, было два хода - направо и налево. Пройдя налево, я увидел второй "лифт", но уезжать пока не стал, т.к. помнил, что меня выбросит на место старта. Поэтому пошел направо. Там и обнаружился подводный ход, ведущий наружу, о котором говорилось в древних легендах. Я смело прыгнул в холодную воду и не утонул, благо у меня был шарик. Немного поплавав, я выбрался из воды на старом кладбище вне пределов замка. Побродив по нему, я наконец обнаружил за мраморными статуями долгожданный будильник, заведенный на три часа утра…

Остальное было делом техники. Я без осложнений пробрался сквозь стаи голодных пираний, использовал унитаз, очутился на месте старта, поднялся на второй этаж, добрался до спальни короля и взобрался на кровать. Потом я зловеще расхохотался и нажал на кнопку будильника. Вы бы видели, какие в тот момент у короля были глаза!!!

Так закончилась эта смешная и грустная история, приключившаяся со мной совсем недавно. До новой встречи друзья!

СОВЕТЫ ЭКСПЕРТОВ

Сегодня этот раздел игровых программ полностью посвящен авиаимитаторам. Все описания подготовлены одним автором - это наш постоянный корреспондент из города С.-Петербург - Фокин С.А.

OPERATION HORMUZ

"Durell" 1988г.
Эксперт Фокин С.А.
г. С.-Петербург

В этой игре Вы будете управлять легким штурмовиком с вертикальным или укороченным взлетом и посадкой - AV-8A "Харриер", который используется в военно-морских силах Великобритании. Вам предстоит очень много стрелять, бомбить, пускать ракеты, и все это для успешного выполнения операции "Хормуз", цель которой - уничтожение ракетных баз противника, а точнее - его шахт, в которых базируется стратегические ракеты.

Имитация полета выполнена довольно условно: Ваш самолет летает как бы на плоскости, а Вы управляете им, глядя со стороны.

Запускается игра нажатием на "1", а управление осуществляется стандартными клавишами, хотя у Вас будет возможность их переназначить. После старта на экране Вы увидите свой авианосец с самолетом на борту. Слева и справа от основного экрана расположены индикаторы повреждений самолета. Внизу расположена приборная панель, у которой слева находится радар, посередине - экран предупреждений и сообщений, над которым находятся указатели различных типов оружия. Указатель выбранного оружия включен всегда. В правой части приборной панели находятся: счетчик очков, индикатор горючего и количество оставшихся самолетов.

Клавиши "1","2","3","4" - выбор оружия.

Самолет управляется клавишами:

"A", "Z" - вверх, вниз (управление по тангажу);

"N","M" - наклон по и против часовой стрелки (управление углом крена);

"SPACE" - стрельба выбранным оружием;

"F" - запуск тепловой ракеты /FLARE/;

"Q" - окончание игры.

Нажав "S", Вы можете наблюдать на карте вражеские базы, расположение вашего авианосца и количество оставшихся самолетов.

Итак, вы произвели взлет, оторвались от палубы и полетели направо к 1-ой вражеской

базе. Время от времени Вам будет выдаваться сообщение, что впереди находятся вражеские корабли. Они оснащены противокорабельным ракетным комплексом "Экзосет" и если Вы не хотите по возвращении обнаружить свой авианосец затонувшим (CARRIER SUNK), то уничтожайте их немедленно, иначе будьте уверены - начнется обстрел авианосца и будут попадания (CARRIER HIT).

Другая постоянная угроза - истребители МИГ-21 советского производства. Они атакуют ракетами "воздух-воздух" с инфракрасным наведением. Чтобы избежать попадания, необходимо произвести пуск тепловых ракет, которые уведут самонаводящиеся ракеты от Вашего самолета. Компьютер предупреждает об этом сообщением "LAUNCH FLARE".

О приближении к ракетной базе предупреждает сообщение - "BASE1". Тут Вам придется показать все свое искусство в заходе на цель, бомбометании и маневрировании, т.к. ракетные шахты (они все имеют квадратную форму) можно поразить только бомбами. Когда Вы разрушите все шахты, компьютер выдаст сообщение "DESTROYED". По мере того, как Вы будете удачно продвигаться по территории врага, Ваш авианосец будет двигаться вслед за Вами. Так что для дозаправки и пополнения боеприпасов не нужно будет далеко лететь, на карте боевых действий свое местоположение Вы можете определить по мигающему номеру базы: если мигают 2 номера, значит Вы находитесь между базами.

Операция будет успешно завершена, если Вы очистите всю территорию от ракетных баз противника.

ACE

"Cascade", 1986 г.
Эксперт Фокин С. А.
г. С.–Петербург

"ACE" - это типичный имитатор воздушного боя с довольно упрощенной техникой пилотирования, но с разнообразными типами вооружения. Отличительная черта боя - высокая агрессивность.

Агрессор предпринял попытку вторжения на Вашу территорию как сухопутными, так и военно-морскими силами. Все его боевые операции сопровождаются мощной поддержкой со стороны авиации. Вам предстоит нелегкая задача отразить нападение врага и защитить суверенитет своего государства.

После загрузки нажмите "SPACE" и на экране появится шифр. Его можно корректировать клавишами "5" и "6". Когда будете готовы, нажмите "SPACE" и компьютер запросит коды. До Вас здесь уже поработали хаккеры, так что подходит довольно много вариантов из 2-х букв, цифр или символов. Наберите, например, "12" или "88", а можете подыскать какие-то свои варианты, после этого на экране появится основная заставка, на которой Вам покажут различные вражеские объекты, встречающиеся в ходе сражения. При дальнейшем нажатии любой клавиши Вы переходите в основное меню:

1 - старт игры
2 - уровень сложности
3 - выбор пилота-одиночки или пилота со стрелком
4 - выбор летных условий (лето, зима, ночь)

5 - загрузка таблицы лучших результатов
6 - показать таблицу лучших результатов
7 - включение/выключение кемпстон-джойстика

После старта появится небольшое меню, в котором Вы можете сценарий и полетное задание:

1 - многоцелевой вариант;
2 - цель - воздушное превосходство;
3 - поддержка сухопутных операций;
4 - поддержка военно-морских операций.

Затем вы оказываетесь на взлетной полосе одной из ваших баз. Управление самолетом очень простое:

"CAPS SHIFT", "Z" - обороты двигателя;
"E","R" - влево, вправо;
"W", "S" - вверх, вниз;
"X" - "огонь";
"ENTER" - выбор оружия;
"U" - шасси;
"M" - карта;
"Q" - окончание игры.

Приборная панель имеет в центре радар и указатель угла крена самолета. Слева вверху находятся индикаторы оборотов двигателя и горючего. Слева внизу - указатели высоты и скорости, а также индикатор положения шасси и указатель курса. Здесь же находится счетчик очков.

В правой части панели расположены: вверху - табло сообщений, внизу - камера заднего вида и указатель выбранного оружия.

Теперь Вы можете непосредственно начать сражение. Чтобы взлететь, необходимо набрать скорость не менее 150 миль в час, но не злоупотребляйте нагрузкой на шасси и не забывайте, что после взлета его следует убрать.

Определите по карте ближайшее сосредоточение вражеских сил, и немедленно направляйтесь в зону боевых действий. В зависимости от того, как быстро Вы будете обнаруживать и уничтожать цели, во многом и зависит исход сражения.

Наземные цели можно уничтожать как из пулемета, так и ракетами "воздух - земля". Большое неудобство доставляют вертолеты противника. У них есть средства защиты от Ваших ракет, и лучше их расстреливать из пулемета. Но главная помеха - это вражеские самолеты. Истребители летают звеньями (2 самолета); количество нападающих звеньев определяется уровнем сложности игры.

Истребители непрерывно атакуют ракетами, о чем выдается предупреждение - "MISSILE WARNING". Чтобы избежать попадания, надо вовремя выпустить уводящую помеху - "DECOY FLARE" и когда ракета будет уведена от Вашего самолета, вы получите сообщение - "MISSILE AVOIDED", в противном случае, после какого-то количества попаданий (DAMAGED SUSTAINED), у Вас могут быть повреждения.

Истребители противника уничтожить очень непросто, но это надо сделать, чтобы до подхода очередной волны авиации, успеть провести подавление сухопутных сил на каком-либо фронте.

Время от времени на высоте 20500 футов будет появляться дозаправщик (REFUELER). Чтобы пополнить горючее в воздухе, необходимо набрать его рабочую высоту, зайти ему в хвост, подлететь как можно ближе и совместить конец всасывающего устройства с заправочной форсункой. Не забывайте, что скорости при этом должны быть одинаковыми.

У Вас в резерве есть один головокружительный прием: нажав клавишу "J" в любой

момент сражения, Вы окажетесь на взлетной полосе одной из Ваших баз. Но, к сожалению, за всю игру Вы не можете нажать ее более 3-х раз.

Противник будет стремиться уничтожить ваши взлетные полосы, и, если Вы плохо ведете бой, то можете получать сообщения: "AIR FIELD DESTROYED". Когда все Ваши взлетные полосы будут разрушены, то считайте, что вы проиграли, т. к. Вам просто негде будет пополнить горючее и боеприпасы. Итак, желаем удачи в боевой пилотировании.

ACE - 2

"Cascade", 1987 г.
Эксперт Фокин С. А.
г. С.-Петербург

Игра "ACE-2" является имитатором воздушного боя, в котором основную роль играет правильное оснащение Вашего самолета.

Сюжет "ACE-2", в отличие от игры "ACE", довольно прост: есть наземная цель противника, есть прикрытие со стороны авиации. Надо уничтожить и то и другое. Время от времени предстоят возвращения на базу для смены систем вооружения, пополнения боекомплекта и для дозаправки горючим.

После загрузки программы нажмите клавишу "SPACE", и на экране появится обширное меню. В него входят следующие пункты:

- начать воздушный бой:
- выбор противника (партнер или компьютер);
- сценарии сражения:

1 - ближний бой - оба самолета уже находятся в воздухе и вооружены только ракетам малого радиуса действия;

2 - полная имитация боевых действий по схемам "воздух-воздух" и "воздух-земля";

- количество самолетов для каждой стороны (от 1 до 20); высвечивание места аварии;
- количество попаданий ракет для уничтожения самолета;
- выбор джойстика.

Курсор перемещается клавишами "Q" и "A", изменение пункта производится клавишей "F". Если в меню выбран синклер-джойстик, то им можно пользоваться одновременно с клавиатурой.

Выбрав параметры боя и переместив указатель на первый пункт, Вы начинаете игру. Перед тем, как непосредственно начать бой, Вам будет предоставлена возможность выбора вооружения.

Если Вы играете против компьютера, то Ваше меню находится в верхней части экрана. Манипулируя курсором также, как и в основной меню, Вы можете выбрать необходимые Вам типы ракет и после этого идти на взлет.

В нижней части экрана расположена приборная панель, в центральной части которой

находятся радар, авиагоризонт, а между ними - указатель угла тангажа.

В левой части панели находится экран, на котором внизу указан выбранный тип оружия и его комплектность, а вверху - высвечиваются сообщения и предупреждения.

В правой части панели находятся индикаторы уровня топлива и мощности двигателей, а под ними - указатели скорости и высоты самолета. В самом правом конце расположен указатель курса.

Ваша задача - уничтожение вражеских самолетов, а также ликвидация наземной цели противника. В то же время, надо оберегать свой корабль в случае попытки потопить его. Расположение всех объектов в любой момент можно увидеть, включив карту (клавиша "C").

Органы управления программой:

"Q", "A" - угол тангажа;

"Z", "X" - угол курса;

"F" - "огонь";

"S", "D" - обороты двигателей;

"W" - выбор оружия.

Надо отметить, что после пуска ракеты по цели на экране появляется сообщение "TARGET LOCKED" - "цель захвачена" и, пока ракета не достигнет цели, вторую выпустить невозможно, на каждый вылет самолет может взять ракеты только 2-х типов, поэтому периодически придется возвращаться на базу.

Чтобы вернуться на базу, необходимо подлететь к левому краю карты и иметь при этом высоту менее 1000 футов.

Самолет может быть вооружен ракетами трех типов:

"HEAT AIR-AIR" - ракета класса "воздух-воздух" с тепловой головкой самонаведения.

"RADAR AIR-AIR" - ракета "воздух-воздух", с наведением по отраженному лучу подсвечивающего радара.

"AIR GROUND" - ракета класса "воздух-земля".

Пуск ракет можно осуществить только тогда, когда система наведения "захватит" цель. Об этом свидетельствует появление придела на лобовом стекле. Ракеты 2-го и 3-го типа требуют "подсветки" цели. Поэтому надо сопровождать цель и после пуска ракеты, т.е. держать ее в области прицела. Ракета с тепловой головкой самонаведения дойдет до цели сама, хотя ее недостатком является малый радиус действия - до 10 миль.

Необходимо отметить, что наземная цель противника надежно защищена системой ПВО, и Вас собьют раньше, чем Ваша ракета достигнет цели. Чтобы этого не произошло, Вам необходимо ставить радиопомехи. Они собьют с курса ракеты ПВО и не позволят уничтожить Вас и Ваши ракеты.

Чтобы уничтожить наземную цель, необходимо попадание 2-х ракет. Радиопомеха ставится клавишей "R", о чем свидетельствует сообщение - "CHAFF LAUNCH". Для этой же цели служат тепловые активные фальшцели (FLARE), но они отводят только ракеты с тепловой головкой самонаведения. Они запускаются тоже клавишей "R".

В заключение дадим ряд полезных советов. У Вас есть сильный козырь - это вид из кабины самолета противника, и Вы всегда сможете увидеть выпущенную в Вас ракету. От ракеты, ведомой по отраженному лучу, можно уйти, совершив маневр и выйдя из луча подсвечивающего радара. Если вы не уверены, чья ракета выпущена раньше Ваша или противника, то советуем Вам после пуска снизить скорость.

При игре вдвоем конфигурация приборной панели 2-го самолета отличается от первой, но ее нетрудно освоить, т.к. она содержит все те же элементы.

Клавиши управления второго самолета следующие:

"P","L" - угол тангажа;

"N","M" - угол курса;
"H" - "огонь";
"O" - выбор оружия;
"J","K" - мощность двигателей;
"U" - постановка помех.

Перевод важнейших сообщений:

"ENEMY PLANE" - вражеский самолет
"RANGE" дистанция
"ALT" - высота
"ENEMY MISSILE" - вражеская ракета
"OUT OF RANGE" - вне зоны досягаемости
"CRASH DANGER" - угроза аварии
"STALL DANGER" - угроза вхождения в "штопор".
"MISSILE LAUNCH" - пуск ракеты

TOMAHAWK

"Digital Integration" 1985 г.
Эксперт Фокин С.А.
г. С.-Петербург

Игра "Томагавк" - это один из самых качественных имитаторов боевого вертолета. Техника управления и полета выполнена на очень высоком уровне и максимально приближена к реальности.

Идет война, силы противника перевешивают. Командование объединенных сил решило применить последний шанс - операцию под кодовым названием "Томагавк", в основу которой положена поддержка наземных операций с воздуха с помощью вертолетов огневой поддержки типа "Апач". Вам и предстоит управлять одним из таких вертолетов.

После загрузки нажмите 2 раза "ENTER", и на экране появится меню настройки:

1 - выбор вариантов сражения (см. ниже)
2 - день или ночь
3 - чистое небо или облачно
4 - высота облачности
5 - ветренность и турбулентность
6 - включение/выключение звука
7 - рейтинг пилота (курсант, новичок, инструктор, ас)
8 - выбор управления.

После настройки нажмите "ENTER", игра начинается с того, что Вы находитесь на посадочной площадке. Для начала необходимо освоить управление:

Клавиши "W", "S" увеличивают и уменьшают обороты двигателя;
"Q", "A" - изменяют крутящий момент на винте.

Если вертолет висит на месте или имеет малую скорость, то "CAPS SHIFT" и "Z" разворачивают его.

Чтобы вертолет набрал скорость, надо наклонить его вперед (клавиша 7), но не забудьте набрать достаточную высоту. Аналогично можно погасить скорость, наклонив его назад (клавиша 6).

Клавиши 5 и 8 управляют креном вправо и влево.

Зону боевых действий в любой момент можно увидеть, нажав клавишу "M".

Клавишей "C" осуществляется наведение на различные типы объектов:

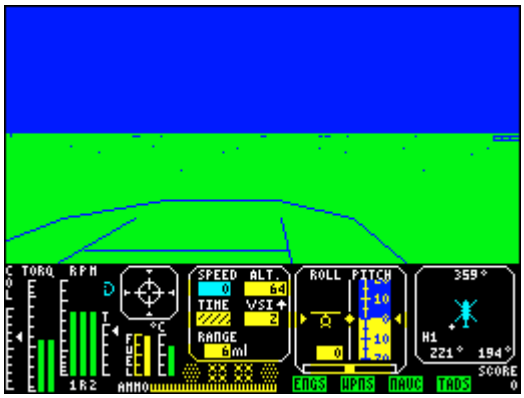
"H" - посадочная площадка:

"B" - база;

"T" - танк или орудие:

Две стилизованные буквы "S" -это вертолет.

Если целей выбранного типа несколько, то все их можно просмотреть клавишей "N". При выборе в качестве цели вертолета, танка или орудия, на лобовом стекле появляется прицел. Каждому виду оружия соответствует прицел определенной конфигурации:



"X" - пулемета;

"Крест" - для неуправляемых ракет;

"Квадрат" - для управляемых ракет.

Выбор оружия осуществляется клавшей "P", а стрельба из него клавишей "O".

Игра может быть остановлена клавиши "H" и продолжена клавишей "J". Одновременное нажатие "CAPS SHIFT" И "SPACE" - окончание игры.

Приборная панель вертолета расположена в нижней части экрана и достаточно сложна, поэтому требует отдельного описания.

В левой части экрана панели находятся вертикальные индикаторы крутящего момента винта (TORQ), оборотов двигателей (RPM), горючего (FUEL) и температуры системы охлаждения (C). Над индикаторами горючего и температуры расположен экран захвата цели, по которой можно произвести выстрел.

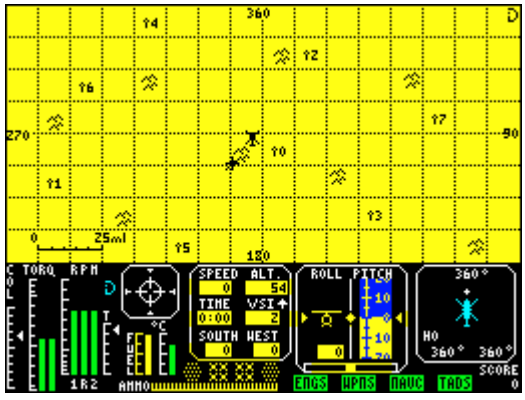
В центре панели находятся указатели скорости, высоты, скорости изменения высоты, времени полета до выбранной цели при установленной скорости, а также указатель расстояния до выбранного объекта.

В правой части панели расположены два экрана: левый показывает тангаж и крен, а правый - курс полета вертолета, азимут цели, тип и номер выбранной цели. В самом низу панели наводятся данные о боеприпасах и несколько табло повреждений вертолета (двигатели, вооружение, навигационные приборы и хвостовое оперение).

Как уже было упомянуто, цель игры - очищение каждого квадрата зоны боевых действий (на карте они мигают) от танков и орудий врага.

В зависимости от выбранного в основном меню варианта сражения, у Вас будут немного отличающиеся задачи.

1. Тренировочный полет: вам предоставляется возможность летать, стрелять, но боевые действия между войсками не ведутся



2. Все пространство поделено на две части. Сражение идет в средней полосе линии фронта.

3. Все оперативное пространство занято противником, и Вам нужно очищать область, начиная с центрального квадрата.

4. Этот вариант подобен варианту 2, но бои ведутся по всей линии фронта. Все пространство условно разбито на 128 квадратов, так что Вам потребуется огромное терпение, чтобы довести начатое дело до конца.

Цифрами на карте обозначены базы. На них Вы можете произвести ремонт в случае

повреждения оборудования. Пополнение боеприпасов происходит на посадочных площадках. Следует знать, что при возникновении аварийных условий, вам выдается звуковое предупреждение, и, чтобы избежать катастрофы, необходимо как можно быстрее вывести вертолет на нормальный режим полета. Например, при пикировании на цель, у Вас может развиться скорость более 200 миль/час. Если вы не погасите скорость, то у вас попросту оторвутся лопасти.

В заключение следует напомнить, что Вы можете навестись на цели только в пределах квадрата, в котором находитесь, и не пытайтесь сесть на базу, если она находится в квадрате, захваченном врагом. Если противник прорвал фронт и завоевал все квадраты от левого края до правого, то Вы уже не сможете отвоевать эту линию обратно.

Танки можно подбить только ракетами.

Наиболее удобно сбрасывать скорость быстрыми поворотами вправо-влево.

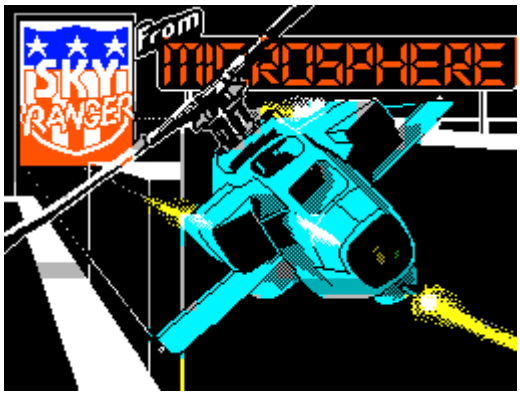
Вертолеты противника появляются по-одному в том квадрате, где Вы находитесь.

Ваши ракеты имеют дальность действия 3 мили.

На этом можно поставить точку, а все остальное вы обнаружите и прочувствуете сами. Успеха вам!

SKY RANGER

"Microsphere", 1985г.
Эксперт Фокин С. А.
г. С.-Петербург



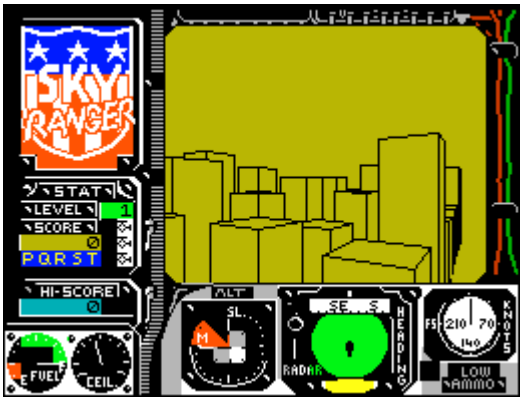
Игра "SKY RANGER" - это упрощенный имитатор полета на вертолете с элементами погони и стрельбы. Полет проходит в очень динамичной форме (особенно на высоких уровнях), поэтому игра захватывает на длительное время.

В современном городе на севере Америки появились загадочные объекты в виде черных шаров до 1 метра в диаметре. Они с поражающей быстротой пожирают атмосферный кислород, но не брезгуют также и людьми. Правительство поручило очистить город от этой заразы Вам - военнослужащему воздушного диверсионно-разведывательного подразделения.

После загрузки, если не нажимать на клавиши, начнется демонстрация полета. При нажатии любой клавиши компьютер запрашивает: будете переназначать клавиши? (да/нет). После этого необходимо ввести пароль уровня сложности.

При нажатии на "ENTER", запускается 1-й уровень. Перед Вами вид из кабины вертолета. Приборная панель проста: справа - указатель скорости, под которым находится табло наличия боеприпасов, в центре панели - радар с компасом, под радаром расположен индикатор захвата цели. Слева от радара находится высотомер. В левой части панели

расположены указатели горючего и высоты облачности.



Управление от стандартных клавиш следующее:

"CAPS SHIFT" ... "V" - поворот влево;

"B" ... "SPACE" - вправо;

"A" ... "G" - уменьшение скорости;

"Q" . . . "T" - увеличение скорости;

"H" ... "ENTER" и "Y"... "P" - изменение высоты;

Весь верхний ряд - "огонь".

Чтобы перейти на следующий уровень, Вам надо сбить 16 шаров. На каждый уровень Вам дано 4 вертолета. Чем выше уровень, тем сложнее догонять и сбивать шары, т.к. их скорость возрастает. По шару можно произвести выстрел, когда мигает индикатор под радаром. Для этого шар должен быть по высоте немного ниже средней линии лобового стекла.

Если вы внезапно попали в облачность, снизьте скорость и опуститесь ниже.

Сажайте вертолет плавно, т.к. иначе на стекле будут трещины, которые впоследствии будут Вам мешать.

Если горючее кончается, то лучше всего сесть на землю и подождать, когда оно кончится окончательно. При этом вы лишаетесь вертолета, но зато у Вас не будет трещин.

При переходе на следующий уровень компьютер выдаст Вам его код. Не забудьте записать его или запомнить, чтобы в следующий раз не начинать сначала. Хочется надеяться, что Вы оправдаете надежды жителей города.

TYPHOON

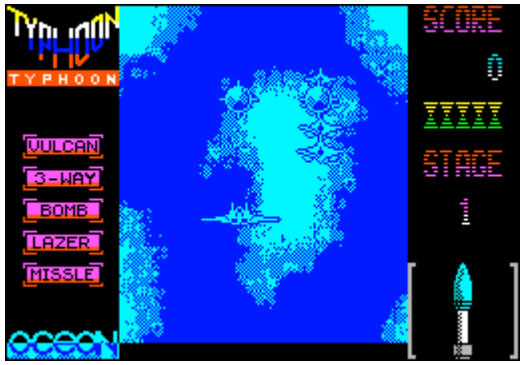
"Ocean", 1988г.

Эксперт Фокин С. А.

г. С.-Петербург

"TYPHOON" - это аркадная многоуровневая игра, которую можно отнести к разряду "боевиков" ("Action"). На всех отдельно загружаемых уровнях Вам придется вести активные боевые действия с воздуха, в самой разнообразной боевой обстановке, встречаясь с невиданным огнем противника, используя различные виды оружия. Вы должны добиться победы. В противном случае - смерть!

Отличная графика, единая связь всех уровней в общем контексте игры, и в то же время их разнообразие - вот, что Вас ожидает.



Итак, после загрузки основного блока надпись вверху начнет мерцать. Выберите нажатием соответствующей клавиши Ваш вариант управления (никакой реакции на экране не будет), а затем нажмите клавишу "N". После этого Вы можете загрузить 1-й уровень, после загрузки появится предупреждающая надпись, после чего вы оказываетесь непосредственно в игре.

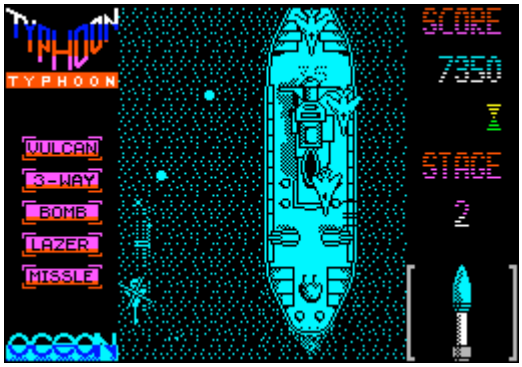
Задача 1-го уровня - уничтожение авианосца противника. Ваш самолет будет снижаться сквозь облака, встречая яростный огонь врага. А после снижения окажется непосредственно над авианосцем. Произведя 3 метких выстрела в машинное отделение

(это около трубы), Вы сможете насладиться видом разрушенного корабля. На этом уровне Вам понадобятся клавиши:

"Q","A" - вперед, назад;
"O","P" - влево, вправо;
"N" - "огонь".

Надо отметить, что конфигурация игры не очень удобная: если Вы не прошли какой-либо уровень (даже 1-й), Вам придется опять произвести загрузку с 1-го уровня, впрочем это недостаток многих программ, имеющих подзагружаемые уровни сложности.

2-й уровень уже сложнее 1-го. Ваша задача пролететь над всеми оборонительными укреплениями, уничтожая как можно больше на своем пути. Когда укрепления закончатся, картинка на экране остановится, и Вам нужно будет уничтожить главную цель в верхней части экрана. На этом уровне выбор оружия более разнообразен: наземные цели можно уничтожать бомбами (клавиша "B"). Иногда Вам могут попадаться контейнеры с буквой "E". Это батареи для зарядки лазера. Если их перехватить, то вместо скорострельной пушки "VULCAN", будет производиться стрельба лазером.



В списке вооружений, который находится на экране слева, всегда указывается задействованный тип оружия. С потерей "жизни" теряется то, что Вы приобрели. Теперь о самой интересном: вы, конечно, обратили внимание на изображение в нижней правой части экрана. Это сверхмощная ракета-аннигилятор, которая уничтожает все в пределах видимости. Она дается только одна (на каждую жизнь) и запускается клавишей "M".

3-й уровень отличается от предыдущего только ландшафтом и конечной стратегической целью. Здесь Вы также сможете использовать более широкий выбор оружия.

По-видимому, нет смысла подробно описывать оставшиеся уровни. В конце каждого из них вас ждет загадочный объект врага, который Вам предстоит уничтожить. После уничтожения всех объектов, на экране появится пилот, который сообщит об удачном выполнении миссии.

Желаем удачи!

TOP GUN

"Ocean", 1986г.
Эксперт Фокин С.А.
г. С.-Петербург

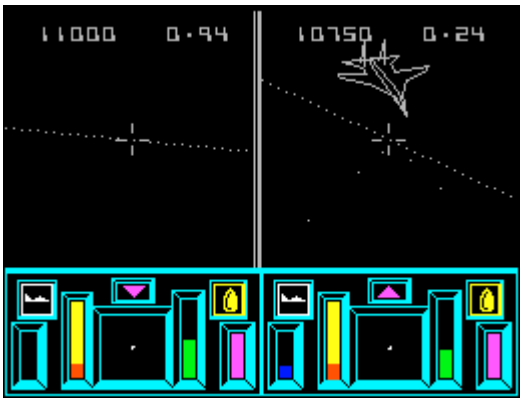
Игра "TOP GUN" представляет собой нечто среднее между имитатором полета истребителя и боевиком жанра "ACTION".

Для того, чтобы быть чистым имитатором, она слишком проста в управлении, не думаю, чтобы кто-то именно так представлял себе управление современным истребителем. Для обычного "ACTION" она все-таки сложновата: требует каких-то представлений о полете и маневрировании в воздухе. Но, как бы то ни было, именно такое сочетание и оказалось весьма удачным с точки зрения игрового впечатления.

Термин "TOP GUN" обозначает школу асов-истребителей военно-морских сил США. Туда принимают самых лучших пилотов ВМС и делают из них истинных мастеров воздушного боя. Если будет возможность посмотреть одноименный фильм, обязательно посмотрите - не пожалеете (у нас фильм известен под названием "Воздушная гвардия" - Прим. "Инфоркома"). Увидев заставку этой игры, вы сразу же узнаете (или не узнаете) изображенный самолет - это 2-х местный тяжелый истребитель морского базирования F-14 "Томкэт". Он приспособлен для взлета и посадки на авианосцы и действует в основном именно с таких кораблей.

Сражаться придется, как ни странно, против однотипного самолета. Это не то маневры (странно, когда пораженный самолет взрывается), не то враг обладает такими же машинами. Впрочем, если кому-то захочется, то он может вообразить себе, что сражается на СУ-27 морского базирования, которые взлетают с авианосца типа "Кремль", против вражеских F-14.

Во время воздушного боя в вашем распоряжении - пушка, ракеты, радиолокационные ловушки и, конечно, маневр. Для выполнения 1-й миссии достаточно сбить 3 самолета противника. Вражеские летчики в 1-й миссии маневрируют весьма слабо и плохо умеют использовать оружие. Они всегда стреляют из пушки и не применяют ни ракет, ни радиолокационных ловушек против Ваших ракет. Учитывая, что с помощью пушки самолет сбить достаточно сложно (необходимо много попаданий), то лучше использовать ракеты. Для этого необходимо, чтобы самолет противника несколько секунд удерживался в центре прицела.



На ракеты Вы должны переключаться заранее с помощью клавиши переключения оружия. При этом прицел имеет форму квадрата. Головка самонаведения должна захватить цель и, если теперь нажать клавишу "огонь", то вы увидите удаляющуюся ракету, которая сама наводится на самолет противника. Если самолет противника находится слишком близко от Вас, то сколько бы мы ни держали его в центре прицела, захвата цели не происходит. Так что оружие это очень эффективное и капризное одновременно. Поэтому во время выполнения миссии 2 (MISSION 2) советуем Вам пользоваться преимущественно пушкой, т.к. стоит только пустить ракету, как опытный враг тут же пускает ловушку и ракета сбивается с правильного курса. При этом вражеский ас, пока Вы возитесь с ракетным прицелом, успевает всадить в вас две-три очереди из своей пушки.

Как ни странно, противник почти не пользуется ракетами. Во 2-й миссии асы противника воюют на порядок лучше, чем в 1-й. Сбить их очень трудно, они выписывают немыслимые пируэты, когда сближаются с Вами и все время стремятся сесть Вам на хвост. Стряхнуть их очень трудно, здесь может выручить только мастерское маневрирование.

Несколько полезных советов по маневрированию во время боя:

Чтобы развернуть самолет, надо выполнить поворот самолета вправо, либо влево и при этом изменять высоту полета. При этом, чтобы ориентироваться и контролировать выполнение разворота, полезно наблюдать за самолетами противника на радаре: Вы должны стремиться, чтобы из нижней части круга они попали в верхнюю.

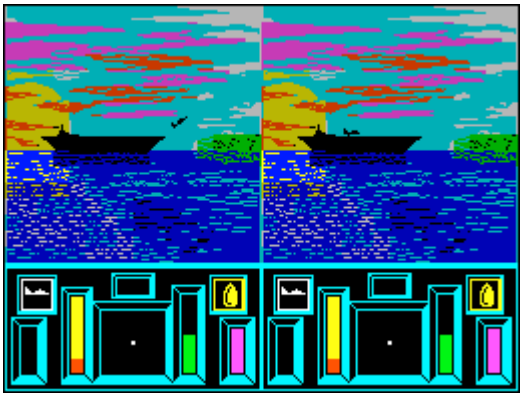
Во время боя полезен также маневр по высоте. Однако, следует иметь в виду, что при крутом пикировании высота падает очень быстро, а при достижении значения 000 Вы сами знаете, что будет с вашим самолетом. При крутом наборе высоты следует следить за тягой двигателя: если она недостаточна, самолет начинает терять скорость и может сорваться в штопор.

Вообще-то вы уже, видимо, заметили необычную особенность этой игры: на экране вид из двух кабин самолетов: вашего и вражеского. Такое встречается довольно редко и здорово облегчает жизнь, можно не только видеть вражеский самолет, но и наблюдать, как враг видит Вас. Это помогает ускользать от вражеского прицела. Полезно также во время боя смотреть на приборы противника и поучиться у него некоторым приемам. Особенно обратите внимание на то, как часто он меняет тягу двигателя – это позволяет ему зайти Вам в хвост.

Приборы помогут Вам контролировать обстановку. Вверху от основного экрана находятся высотомер (в футах) и индикатор мощности двигателя. Под основным экраном расположен радар. В его центре жирной точкой обозначен Ваш самолет, а маленькой - самолет противника. Интересно, что стоит только сбить вражеский самолет, как тут же появляется другой. Они всегда появляется сзади или сбоку. Так что до того, как они сблизятся с Вами, не плохо было бы развернуться.

Разберемся дальше с изображением в нижней части экрана. Верхний квадрат слева с силуэтом самолета показывает положение относительно горизонта. Квадратик справа содержит стилизованное изображение включенного оружия. При переключении оружия меняется вид прицела на основном экране. Радиолокационные ловушки прицела не имеют, средняя полоска внизу показывает количество повреждений самолета, а полоска справа - тяга двигателя.

После загрузки программы появится меню, в котором можно выбрать количество игроков (необходимо нажать 1 или 2). В общем-то, на начальной стадии игры полезно выбрать режим 2-х игроков и играть одному. При этом вражеский самолет - неуправляем, он не маневрирует и не стреляет. Идеальная мишень для начинающего пилота!



Чтобы войти сразу в игру (если управление устраивает), то надо нажать клавишу "SPACE", а если нет, то "ENTER". При этом появится новое меню для 1-го игрока. Курсорными клавишами можно выбрать нужную опцию и, если выбрать больше нечего, нажать "SPACE" или "ENTER", если надо выбрать управление для второго игрока.

При назначении клавиатуры появляется соблазнительная надпись: "PRESS D TO DEFINE KEYS". При этом идти на поводу у компьютера не следует: если переназначить клавиши, то игра зависнет на картинке с авианосцами, ничего не поделаешь, так уж с ней поработали компьютерные хаккеры. Дело в том, что эта игра в оригинальной версии работала только на фирменных "ZX SPECTRUM", а на наших моделях она зависала.

И, в заключение осветим клавиши управления:

"W","S" - вниз, вверх;

"R","E" - влево, вправо;

"A","Z" - обороты двигателя;

"T" - "огонь"

"CAPS SHIFT" - переключение оружия.

Счастливого полета!

THE DARK WHEEL

Для любителей программы "ELITE" мы продолжаем печать научно-фантастической новеллы английского писателя Роберта Хольдстока "The Dark Wheel" (перевод наш - "ИНФОРКОМ"). По расчетам новелла будет закончена в номере 5-6 "ZX-РЕВЮ" за 1993 год, а на очереди приключенческий боевик "Конференция", написанный по мотивам известного игрового сериала Колина Свинбурна JOE BLADE.

Продолжение. Начало см. на стр. 175, 176.

Наконец-то они вышли из гиперпространства.

В то же самое мгновение ловкие пальцы Джейсона взметнулись над клавиатурой управляющей консоли. "Авалония" рванулась вперед и одновременно начала закручиваться вокруг оси. Маленьким зеленоватым диском поплыла на экране планета Листи. Алекс видел, как отец навел и выпустил обе имевшихся на борту "Авалонии" ракеты и уже положил руку на триггер многоцелевого лазера.

Так значит это пират!

Когда до Алекса дошла в полной мере неотвратимость предстоящего боя, во рту пересохло, но мысли приобрели необычайную остроту и четкость. Никогда еще он не участвовал в боях, разве только на имитаторе. Конечно, отец рассказывал ему об этом, но в его рассказах это выглядело отнюдь не великолепным.

Итак, пиратский корабль, преследовал свою жертву на всем протяжении гиперперехода ради загруженного на борт фруктового сока!? Какой-то голос из глубины сознания подсказывал Алексу, что здесь что-то не так. Корсары так себя не ведут. Обычно они крейсируют на границах планетных систем, внимательно прощупывая пространство своими сканерами и придирчиво выбирая объект для атаки. Пираты могут повстречаться где угодно, но они редко оказываются в пределах корпоративных и демократических систем - здесь очень эффективно действует полиция. Их излюбленные системы - анархические и феодальные.

Определенно в поведении противника было что-то не то. Что-то не то, если это действительно пиратский корабль.

Алекс оторвал взгляд от проплывающей планеты и взглянул на сосредоточенное, посеревшее лицо отца. Судя по его выражению, их положение было далеко не безопасным.

- Одень дислок и займи спасательную капсулу, - буркнул Джейсон Райдер. - Выполняй!

- Я буду сражаться.

- Черта-с-два! Выполняй, я сказал! - с этими словами Джейсон сунул сыну в руки небольшую черную маску дистанционного локатора.

Защитные экраны "Авалонии" приняли первые удары ракет и пальцы Джейсона опять забегали по клавишам, приводя в действие защитные системы. Корабль дрожал от напряжения в последнем спасительном рывке. Начала работать система подавления вражеских ракет, а "Кобра" сделала второй залп.

Задний экран вспыхнул и залился ослепительным светом, затем сквозь яркое пятно вновь стали проступать серые очертания приближающегося корабля-убийцы.

Все произошло настолько быстро, что впоследствии Алекс много раз пытался восстановить точную последовательность событий и не мог. В ослепительной, но бесшумной схватке корабли вращались один относительно другого и оба вместе вокруг планеты. Пространство между ними полыхало. Оружие наносило удар за ударом, оружие отражало удары. А затем вселенная раскололась, огни "Авалонии" мигнули последний раз и погасли. Послышался свист уходящего воздуха, по управляющей консоли забегали огни индикаторов: перегрев лазера, низкий уровень энергии защитных полей, груз поврежден, температура в рубке падает…

Сильные руки подхватили Алекса, рванули вверх и грубо втолкнули в люк спасательной капсулы - вот все, что осталось в памяти о последних секундах жизни "Авалонии". Маска дислока в этот момент уже была на лице, закрывая глаза, нос и рот.

Корабль дрожал и скрежетал, топливо хлестало в космос. В последний раз отец и сын глядели в глаза друг другу.

-Я не понимаю, кому надо было…! - кричал Алекс сквозь грохот погибающего корабля.

- Раккса. - успел сказать Джейсон. - Помни это, Алекс. Раккса! Не забывай меня. Я не хотел тебе такой судьбы. Раккса!

Сработала система катапультирования. Капсула закружилась в пространстве. В последний раз мелькнули гладкие обводы "Авалонии" и все застыло в ослепительной вспышке, а затем, сменяя друг друга, перед глазами пошли картины белого жара и черного холода.

В секунду ушло все, что было так близко и дорого - корабль, отец и частица прожитой жизни. Все распалось в огненной вспышке выстрела пиратского корабля. Языки пламени рванулись к спасательной капсуле. Жар, боль, холод - чувства менялись с калейдоскопической быстротой. Раздираемая на части, капсула была отброшена и, разваливаясь на куски, начала падение на планету. Последние капли жизни покидали бессознательное тело Алекса.

ГЛАВА 2.

Космос безмолвен, но крик о помощи услышит каждый.

Для того и служит спасательная маска дислока. В то самое мгновение, как капсула потеряла герметичность, струи пластифибры хлынули из сопел, заполняя кабину. Мгновенно твердеющая масса защитила безжизненное тело и от вакуума и от холода. Расход кислорода снизился до минимально необходимого для питания сердца и мозга. Включилась система поддержания жизнедеятельности, готовая сделать инъекцию адреналина или успокаивающего наркотика в случае необходимости. А дислок кричал о помощи на весь космос.

Это стандартное устройство выдавало мощный сигнал, мгновенно распознаваемый как крик о помощи безжизненного тела. Крик разносился на сорока каналах, четырежды в секунду меняя частоту на каждом из них. Сто двадцать шансов из ста за то, что он будет услышан.

Неуклюжий "Боа", загруженный от киля до рубки индустриальным оборудованием

замедлил ход и начал сканирование пространства в поисках источника тревожного сигнала.

Два полицейских "Вайпера" прервали дежурное патрулирование в окрестностях звезды и ринулись в установленный сектор.

Межзвездная пассажирская шхуна "Мори", переоборудованная в космический госпиталь, о чем свидетельствовала огромная золотая звезда на обшивке верхней палубы, начала медленно набирать ход.

Тысячи радиосообщений, непрерывно циркулирующие между кораблями, планетой и кольцом орбитальных станций были внезапно прерваны, уступив все эфирное пространство этому отчаянному крику о помощи. Повсюду прекратились телетрансляции. Изображения на экранах сменились картой близлежащих секторов с фиксацией координат аварийного дислока. Даже рекламные корабли прервали свою трансляцию, переключившись на поддержку поиска.

Тысячи людей устремили взгляд в звездное небо. Этот зов о помощи был слишком хорошо известен каждому и никто не мог оставаться спокойным.

Не прошло и двадцати секунд, как два автоматических спасательных зонда зависли над безжизненным телом попавшего в беду астронавта. Эти маленькие аппараты были, по-существу, автоматическими роботами и несли на себе запас кислорода на один час и до 40 доз различных медицинских препаратов, рассчитанных на оказание первой помощи. Когда спасательный трос притянул и зафиксировал тело, роботы приступили к работе. Проткнув защитный пластик, иглы вошли в тело и живительный кислород вместе с глюкозой и адреналином стал поступать непосредственно в кровь, а когда Алекс впервые открыл глаза, то сразу получил укол тенваля - сильного успокаивающего средства.

В ушах раздался голос говорящего робота.

- Брэнди? Скотч? Водка, сэр? Любые стимулянты на время ожидания.

- Что… случилось … с кораблем?.. - с трудом выдыхал звуки сквозь маску Алекс.

- Так, значит брэнди - ответил робот и, ласково мигнув огоньками приборной панели, выдал больному двойную порцию квитирианского коньяку.

Через час Алекс уже был на борту космического госпиталя, дрейфующего над планетой. Медики позаботились об ожогах на руках и на лице, восстановили лопнувшие поверхностные кровеносные сосуды. Алекс чувствовал себя растоптанным и избитым, но физически был уже в норме.

Тем не менее, образ взрывающегося корабля продолжал преследовать его, как наваждение. Он стоял у широкого окна госпитальной палаты и сосредоточенно наблюдал за медленным вращением серо-зеленой планеты и суетливой толкотней челноков и грузовиков, снующих по своим делам в атмосферу и обратно. Разноцветные следы их пролета в верхних слоях атмосферы постепенно таяли и расплывались. Но куда бы Алекс ни обратил свой взор, везде ему казалось, что он видит "Кобру", ту самую "Кобру" и еще лицо отца.

Что-то было не так. Да, нападение было внезапным. Тревога, вспышка гнева, огонь и все кончено, но Алекс чувствовал, что все это время Джейсон Райдер ЗНАЛ.

Сейчас его сын ворошил онемевшую память и с каждой минутой все отчетливее сознавал, что отец гораздо лучше понимал нависшую над ними опасность, чем можно было судить по его виду. Теперь стало ясно, что это было в его лице, в напряженной атмосфере тревожного ожидания, нависшей в рубке, в отрывистых фразах, когда они еще только приближались к гипертуннелю.

Джейсон знал об угрозе и был готов к атаке. Он все подготовил для того, чтобы в последний момент спасти сына.

Это выглядело бессмысленно, но это было так. Теперь, когда Алекс потерял отца, у него не стало обоих родителей. У него вообще никого не осталось и родная планета сразу стала казаться чужим, негостеприимным миром.

За спиной мягко открылась дверь и вошла медсестра в сером халате. Слегка пожурив пациента за то, что тот покинул кровать, она по всей видимости осталась вполне удовлетворенной его состоянием. Затем поток посетителей стал непрерывным.

Сначала пришел доктор. Его, по-видимому, больше всего интересовало психическое

состояние больного и он, кажется, остался не вполне доволен.

- Молодой человек, - сказал врач, - вы потеряли отца и его уже не вернуть. Постарайтесь расслабиться. Нет ничего стыдного, если вы поплачете. Знаете, слезы смывают все - и горе и печаль. Не пытайтесь себя сдержать, это не пойдет вам на пользу.

- Я не плачу, я рыдаю о своем отце, и я еще сильнее буду рыдать, когда превращу в пепел того пирата, который его убил! И не раньше!

- Даже так?

- Именно так.

Следующим пришел агент Всегалактической службы медицинского страхования. Он взял необходимые данные и вскоре убедился, что Алекс застрахован на все случаи жизни, включая расходы на лечение и доставку на родную планету.

Затем пришла полиция. Два мужчины в серых плащах с серебряными поясами и с одинаковыми ничего не выражающими лицами представляли Департамент Борьбы с Распространением Наркотиков.

- Какой груз несла "Авалония"? Почему пират преследовал ее в пределах Корпоративной системы? Не занимался ли отец когда-нибудь перевозкой наркотиков? А оружия? Не возил ли он рабов? Какие инопланетные товары были на корабле? Манихуаза? Марсианский вирт? А может быть фиргланды? Что сказал отец перед смертью? Сможет ли Алекс узнать пиратский корабль? Отличительные приметы?

Алекс рассказал им все, что смог вспомнить, все, что видел, все, что слышал ... кроме того, что отец, очевидно, знал о готовящемся нападении, и ни слова не сказал о Раккcле.

Наконец полицейские ушли. Они совершенно не были удовлетворены тем, что смогли узнать.

- Молодой человек, мы понимаем, что Вы вполне самостоятельный пилот, имеете лицензию и можете сами выбрать путь домой, но мы настоятельно просим Вас перед отправлением согласовать с нами маршрут.

Раккcла!

Алекс наблюдал, как юркий, зловещий "Вайпер" отошел от госпиталя, круто развернулся и резко набрал скорость. Его серый цвет как нельзя лучше соответствовал мрачным штормовым облакам, затягивающим океан, над которым они пролетали.

Раккcла!

"Что это может быть? Что это может значить?"

После полуночи, когда Алекс еще не спал, в комнате замигал маленький зеленый огонек. Алекс зажмурился и понял, что за ним наблюдают.

- В чем дело? - обратился он в пустоту комнаты.

- По халофаксу для вас пришло сообщение. Просят дать ответный луч. Вы будете выходить на связь? - ответил голос медсестры.

Алекс присел. Дела развивались все любопытнее. Никто не мог знать, где он находится. Это факт. Он вновь нахмурился и ответил: - Да, конечно.

- Счет за переговоры записать на ваш кредит?

Опять загвоздка. Не было у него никакого кредита. У него вообще ничего не было, пока он не получит какую-то страховку. Ну, а раз так, то Алекс со спокойной душой ответил: - Да.

В центре комнаты воздух завибрировал, стал белым непрозрачным, потом рассыпался на мириады белых кристаллов, из которых постепенно стала собираться трехмерная голограмма мужской фигуры. Он был высоким, но слегка сутулился. Постепенно изображение приобрело цвет, но белизна фигуры сохранилась. Длинные седые волосы, всклокоченная борода, смуглое лицо. Маленькие блестящие глаза скрывались за глубокими морщинами. Человек улыбался. На нем была одета застиранная униформа торгового флота, одна рука безжизненно свисала вдоль тела. Даже ботинки были изрядно поношены и кое-где начала отставать подошва. Ручной лазер на боку тоже знавал лучшие дни, как и все в облике этого незнакомца.

- Это ты парень Райдера? - спросило поношенное изображение хриплым голосом человека, который подышал на своем веку глубоким вакуумом.

- Да, я Алекс Райдер. А вы?

Алекс слез с постели и подошел к фигуре. Старик спокойно смотрел на него и что-то жевал. Затем сплюнул. Плевок вроде бы пролетал мимо плеча Алекса и тот сделал шаг в сторону, забыв, что это только трехмерное изображение.

- Ты меня не помнишь, - сказал старик. - Да это и понятно. А я вот тебя помню.

- Как вас зовут?

- Рейф Зеттер. В прошлом торговец. Мы много лет работали в паре с твоим отцом, пока не разделили компанию, разойдясь во мнении по одному щепетильному вопросу.

- Рабы? - быстро отреагировал Алекс. Теперь и он вспомнил Рейфа. Но что же с ним стало! Он преждевременно состарился. Ему ведь столько же лет, сколько было и Джейсону Райдеру, но выглядел он лет на двадцать старше.

- Верно, парень. Рабы. Я прожил жизнь на конце вайперской удавки и был на пол шага впереди закона. К тому времени, как я позволил себе эту прихоть, у меня уже была железная задница и я сумел проскочить преисподнюю. Таково мое положение.

- В преисподней?

- Нет, я разорен.

Алекс кивнул, постепенно до него начинал доходить жаргон космических торговцев. "Железная задница" означает вооруженный до зубов корабль: силовые поля, ракеты, боевые лазеры. Такой может сделать пробежку по любой системе, даже по анархическому раю вроде Сотикью. "Проскочить преисподнюю" означает сорвать солидный куш на нелегальных операциях, хорошо погулять, а потом потерять все. Так они обычно и кончают.

Рейф продолжал.

- Мне очень жаль, что это произошло с Джейсоном. Он был хороший человек. Старый верный друг. Человек, которого я всегда буду уважать.

- Но это произошло не более, чем восемь часов назад. Откуда, черт возьми, вы можете об этом знать?!

Рейф закашлялся и сплюнул. Алекс не удержался и опять попытался уклониться от плевка, который растаял на границе голограммы. Холодок отвращения прошел по спине.

- Парень, твой темперамент такой же, как был у Джейсона. Не знаю, может быть даже ты унаследовал часть его способностей?

- Ответь на мой вопрос, старик. Как ты умудрился узнать об отце. Как ты меня нашел? - повысил тон Алекс.

Наблюдая из голограммы, Рейф пожевал и улыбнулся. Алекс невольно напрягся в ожидании очередного гиперпространственного плевка.

- Я повторяю, Алекс! Я с глубочайшим уважением отношусь к Джейсону. За то, кем он был и за все, что он сделал.

- Да, он был честный человек, - сказал Алекс, - и он был честным торговцем.

- Нет, он был много больше, чем ты думаешь, чертовски больше, - громко воскликнул РейФ и сплюнул. Алекс вздрогнул, завибрировало и стало тускнеть изображение.

- Что это значит?

Рейф Зеттер подался вперед, приблизившись к Алексу.

- Он был бойцом, Алекс, Одним из лучших. Он не должен был умереть так…

- Отец был торговцем, а не бойцом, - испуганно возразил Алекс.

- Подумай лучше, сынок.

- Но он не выносил стрельбы!

- Может быть, может быть, но это не останавливало его. А как ты думаешь он смог столько лет заниматься торговлей? Черт побери, Алекс, да ты можешь возить сметану и пряники и все равно рано или поздно найдется кто-то, кто захочет отнять их у тебя. Твой отец был бойцом высочайшего класса…

Алекс проглотил подступивший к горлу комок.

- Высочайшего класса?..

Рейф кивнул.

- Да, Алекс. - мягко продолжал он. - Ты можешь быть смертоносным, ты можешь быть опасным, но ты все равно превратишься в собачьи консервы на орбите какой-нибудь

собачьей планеты вроде Извивы. Но если ты Элита и если ты погибаешь, то значит для твоей смерти есть веские причины.

"Что говорит этот старик? Элита? Боец класса Элита?" У Алекса закружилась голова. Он слышал о пилотах, которые дослужились до этого рейтинга - их было очень немного. Очень многие были Опасными, иначе и нельзя заниматься торговлей. Очень многие были Смертоносными. Их много как среди торговцев, так и среди пиратов. Но Элита?! Таких единицы.

Его отец, Джейсон Райдер был Элитой, а никто в семье даже и не догадывался!

- Джейсон был одним из самых лучших. Ты, вероятно, никогда не видел его корабль. Это просто крепость. Он торговал в таких местах, которые мы видели разве что в кошмарах. - Рейф восхищенно покачал головой. - Один из лучших... Боец высочайшего калибра... - Его взгляд вновь упал на Алекса. - Весь вопрос в том, сможешь ли ты стать таким же?

- Почему ты сомневаешься?

- Джейсон никогда не рассказывал о тебе. Я думаю, он берег тебя. Беда в том, что мне не с чего теперь начать. По твоим глазам я вижу, что ты будешь мстить за отца, но для меня это означает лишь то, что еще один Райдер станет космической пылью еще до того, как сумеет навести ракеты.

Алексу не понравился этот тон.

- Я провел многие часы на тренажерах и у меня высшие баллы.

Рейф рассмеялся и смачно сплюнул, а затем сказал серьезно.

- Алекс, хотел бы я знать, не собираешься ли ты ...

- ... превратиться в собачьи консервы на орбите Извивы?

- Да, что-то в этом роде. Единственный человек, который знал, на что ты годишься, был твой отец. А теперь. Алекс, ответь мне. Скажи мне правду... Отец тебе ничего не сказал в тот момент... ну, в общем, что он сказал тебе перед смертью? Может быть, он на что-то намекнул?

- Он много чего говорил, - промямлил Алекс и почувствовал горькую боль, вспомнив глаза отца и его последние слова: "Не забывай меня. Алекс..." - Я думаю, он знал, что погибнет, последнее слово, которое он произнес, было "Раккcла". Я не знаю, что это может означать. Думаю, что что-то инопланетное.

Рейф улыбнулся и покачал головой, в его глазах вспыхнул блеск.

- Раккcла - это не что-то инопланетное. Это призрачный мир. Планета - легенда. - Он еще немного поколебался и продолжил: - отец на самом деле это сказал?

Алекс кивнул: - За мгновение до... Он сказал это перед смертью, это было его последнее слово.

- Значит, он знал и меня это радует. Тогда так. Завтра, Алекс, ты вылетаешь на Тионислу. Там возьмешь орбитальный челнок и полетишь на кладбище погибших кораблей. Скажешь, что прибыл посетить могилу звездопроходца Флейшера и внимательно смотри по сторонам. Сделай это, парень. Завтра я буду тебя ждать.

- Ждать ради чего?

Рейф закашлялся. - А как ты собираешься охотиться на "Кобру? Ты будешь летать по космосу на попутках? А воевать будешь чем? Размахивая дубиной? Тебе нужен корабль. Будь на свалке в Тионисле. Я знаю один корабль, который тебе нужен. И никому ни слова. Молча отправляйся завтра на Тионислу.

- Но...

- Прощай, Алекс!

И Рейф сплюнул в последний раз.

(Продолжение следует)

Содержание