ZX - РЕВЮ
N3,N4 1993

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

Вниманию читателей!

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

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

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

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

Вниманию дистрибуторов! Адреса наших оптовых покупателей публикуются бесплатно.

Спектрум в школе

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

Попробуйте выполнить простую Бейсик-программу:

```
10 LET A=100000001
20 LET B=100000000
30 PRINT A;"-";B;"=";A-B
```

На экране Вы увидите:

```
1E+8-1E+8=1
```

А желательно было бы видеть:

```
100000001-100000000=1
```

Осуществимо ли это?

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

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

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

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

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

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

"ПОИСК СОМНОЖИТЕЛЕЙ"
"ПОИСК ПРОСТЫХ ЧИСЕЛ"

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

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

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

Строка 2100 проверяет, все ли возможные сомножители проверены. Если все, то переход на строку 2200 для вывода результата. Строки 2110 или 2120 (в зависимости от заданного режима) проверяют, делится ли исследуемое число на сомножитель без остатка. В режиме анализа всех чисел, в том случае, если делится без остатка, то значит найден один из сомножителей, происходит вывод его на экран и установка флага F (строка 2120) и далее выполняется продолжение анализа остатка от деления для поиска остальных сомножителей. В режиме поиска только простых чисел, зафиксированное деление без остатка говорит о том, что это число не простое, поэтому дальнейший анализ не нужен и выполняется переход к проверке следующего числа (строка 2110).

Строка 2200 выполняет печать исследуемого числа и всех его сомножителей, если они есть или печать сообщения о том, что число простое. Собственно печать чисел выполняется подпрограммой GO SUB 40. Здесь, если число меньше, чем 100000, то происходит обычная его печать, а если больше, то печать выполняется в два приема. Сначала печатается то, что выходит за пределы 100000 (переменная N, строка 44), затем, для того, чтобы напечатать младшие разряды, происходит преобразование значения числа в стринг. Поясним происходящее на примере. Пусть нам надо напечатать число 4000000027. Сначала берутся старшие разряды числа: H=40000, теперь вычисляется "остаток", увеличенный на 100000, и происходит печать его за старшими разрядами:

```
надо:      4000000027
печать1:   40000          <- число
печать2:   100027         <- стринг
           ↑ эта единица не выводится, так как полученный стринг печатается (2 TO).
```

В заключение отметим, что поскольку программа производит многократные вычисления при поиске сомножителей, и рассчет выполняется в Бейсике, то получение результата для больших чисел может занять достаточно продолжительное время. Представьте себе, сколько сомножителей надо перебрать для того, чтобы убедиться в том, что десятизначное число является простым! Например, для числа 4294987291 - требуется 8 минут! Поэтому в качестве усовершенствования этой программы мы можем посоветовать читателям попробовать скомпилировать подпрограмму со строки 2000 для ускорения выполнения вычислений.

Маленькие хитрости

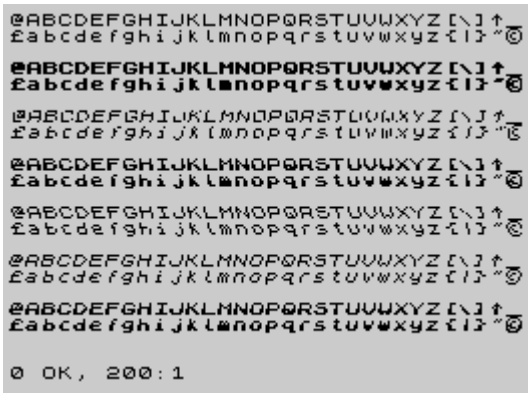
Стилизованные шрифты

Вы, наверное, видели, как во многих программах применяются «стилизованные» шрифты. Утолщённые, наклонные и другие разновидности придают программам особую привлекательность. Если для этой цели применять загружаемый символьный набор, который занимает почти килобайт памяти, то требуется соответствующее время при загрузке программы, которая порой меньше, чем сам символьный набор. Поэтому для получения оригинальных шрифтов можно воспользоваться несложными процедурами в машинных кодах. Правда, выигрыш во времени загрузки получается только в том случае, если используется символьный набор ПЗУ, поскольку русский шрифт всё равно приходится загружать. Однако в обоих случаях появляется возможность разнообразить программу, используя один базовый символьный набор и оригинально модифицируя его. Например, в одном режиме работы программы вывод осуществляется утолщённым шрифтом, в другом - наклонным, в третьем - готическим и т.д. Для этого не надо загружать такое число символьных наборов, а все их можно получить из основного - базового.

В качестве примера, поясняющего вышесказанное, приводим простую Бейсик-программу, построенную на использовании машиннокодовых процедур, в основе которых лежат операции с битами. Преобразования выполняются с символьным набором ПЗУ, но Вы можете использовать изложенные принципы и для модифицированных загружаемых символьных наборов.

```
4 GO TO 100
8 POKE 23606,0: POKE 23607,117: RETURN
9 POKE 23606,0: POKE 23607,60: RETURN
40 FOR n=64 TO 127: PRINT CHR$ n;: NEXT n: PRINT '': RETURN
50 RESTORE 50: FOR n=30000 TO 30029: READ a: POKE n,a: NEXT n
55 DATA 33,0,61,17,0,118,1,0,3,237,176,33,0,118,17,0,3,126,79,203,63,177,119,35,27,122,179,
      32,244,201
59 GO TO 80
60 RESTORE 60: FOR n=30000 TO 30041: READ a: POKE n,a: NEXT n
65 DATA 33,0,61,17,0,118,1,0,3,237,176,33,0,118,14,96,6,2,126,203,63,203,63,119,35,16,247,6,
      4,126,203,63,119,35,16,249,35,35,13,32,231,201
69 GO TO 80
70 RESTORE 70: FOR n=30000 TO 30035: READ a: POKE n,a: NEXT n
75 DATA 33,0,61,17,0,118,1,0,3,237,176,33,0,118,6,96,197,35,35,35,35,6,4,126,79,203,63,177,
      119,35,16,247,193,16,237,201
80 RANDOMIZE USR 30000: GO TO 40
100 BORDER 7: PAPER 7: INK 0: CLEAR 29999: PRINT AT 1,0;
110 GO SUB 9: GO SUB 40: GO SUB 8
120 GO SUB 50
130 GO SUB 60
140 GO SUB 70
150 POKE 30027,0: PAUSE 20: GO SUB 80
160 POKE 30026,39: PAUSE 20: GO SUB 80
170 POKE 30027,177: PAUSE 20: GO SUB 80
200 GO SUB 9
```

После набора запустите программу командой RUN. Вы увидите на экране различные символьные наборы. (См. рисунок).



Подпрограммы со строк 50, 60, 70 - формируют в памяти с адреса 30000 (7503h) и выполняют процедуры в машинных кодах, которые модифицируют символьный набор. После этого происходит вывод на экран фрагмента полученного символьного набора при помощи подпрограммы со стоки 40.

Переключение символьных наборов происходит при помощи подпрограмм GO SUB 8 или GO SUB 9. Первая включает полученный символьный набор, а вторая - символьный набор ПЗУ.

Рассмотрим подробно, что же происходит при работе блоков в машинных кодах с символьным набором.

Листинг 1

Подпрограмма GO SUB 60 создаёт утолщённый символьный набор:

```
7530    21 00 3D          LD HL,#3D00     ;Переброска базового
7533    11 00 76          LD DE,#7600     ;символьного набора
7536    01 00 03          LD BC,#0300     ;в адрес 7600h (30208)
7539    ED B0             LDIR            ;для модификации.
753B    21 00 76          LD HL,#7600     ;Начиная сначала, каждый бит символьного
753E    11 00 03          LD DE,#0300     ;набора, т.е. каждая линия символа
7541    7E          LOOP1 LD A,(HL)       ;загружается в аккумулятор,
7542    4F                LD C,A          ;сохраняется в регистре C, затем
7543    CB 3F             SRL A           ;сдвигается вправо на один пиксел
7545    B1                OR C            ;и объединяется по «ИЛИ» с прежним
                                          ;значением, т.е. включенные пикселы так
                                          ;и останутся включенными, но включатся
                                          ;ещё и те, которые получены путём сдвига.
7546    77                LD (HL),A       ;Полученное значение записывается
                                          ;обратно в символьный набор.
7547    23                INC HL          ;Переход к следующей ячейке, т.е. к
                                          ;следующей линии символьного набора.
7548    1B                DEC DE
7549    7A                LD A,D          ;Проверка на достижение конца символьного
754A    B3                OR E            ;набора, т.е. нуля в счётчике DE
754B    20 F4             JR NZ,LOOP1     ;Если нет, то повторение процедуры
754D    C9                RET             ;Если да, то возврат в Бейсик
```

Листинг 2

При помощи подпрограммы GO SUB 70 получается шрифт, напоминающий готический:

```
7530    21 00 3D          LD HL,#3D00     ;Переброска
7533    11 00 76          LD DE,#7600     ;символьного набора
7536    01 00 03          LD BC,#0300     ;в адрес 7600h (30208)
7539    ED B0             LDIR            ;для модификации.
753B    21 00 76          LD HL,#7600     ;Начиная сначала, для всех
753E    0E 60             LD C,#60        ;символов символьного набора
7540    06 02       LOOP1 LD B,#02        ;берутся две верхних линии (два байта).
7542    7E          LOOP2 LD A,(HL)       ;Байт записывается в аккумулятор,
7543    CB 3F             SRL A           ;сдвигается вправо
7545    CB 3F             SRL A           ;на два пиксела
```

```
7547    77              LD (HL),A       ;и записывается назад в память.
7548    23              INC HL          ;Переход к адресу следующего байта.
7549    10 F7           DJNZ LOOP2      ;Повторение для второго байта.
754B    06 04           LD B,#04        ;Берутся четыре следующих линии.
754D    7E        LOOP3 LD A,(HL)       ;Байт записывается в аккумулятор,
754E    CB 3F           SRL A           ;двигается вправо на один пиксел
7550    77              LD (HL),A       ;и записывается назад в память.
7551    23              INC HL          ;Переход к адресу следующего байта.
7552    10 F9           DJNZ LOOP3      ;Повторение для всех четырёх линий.
7554    23              INC HL          ;Оставшиеся две линии из
7555    23              INC HL          ;восьми не изменяются.
7556    0D              DEC C           ;Счётчик символов уменьшается на 1.
7557    20 E7           JR NZ,LOOP1     ;Повторение цикла для следующего символа.
7559    C9              RET             ;Возврат в Бейсик.
```

Листинг 3

Оригинальный шрифт получается, если утолщённой сделать только нижнюю часть символа, оставив верхнюю без изменения:

```
7530    21 00 3D        LD HL,#3D00     ;Переброска
7533    11 00 76        LD DE,#7600     ;символьного набора
7536    01 00 03        LD BC,#0300     ;базового в адрес 7600h (30208)
7539    ED B0           LDIR            ;для модификации.
753B    21 00 76        LD HL,#7600     ;Начиная сначала
753E    06 60           LD B,#60        ;для всех символов символьного набора
7540    C5        LOOP1 PUSH BC         ;запоминание счётчика цикла на стеке,
                                        ;так как регистр участвует ещё
                                        ;в одном, внутреннем цикле.
7541    23              INC HL          ;Четыре верхние
7542    23              INC HL          ;линии символа
7543    23              INC HL          ;оставляем
7544    23              INC HL          ;без изменений.
7545    06 04           LD B,#04        ;Цикл для четырёх оставшихся линий.
7547    7E        LOOP2 LD A,(HL)       ;Байт заносится в аккумулятор,
7548    4F              LD C,A          ;сохраняется в регистре C,
7549    CB 3F           SRL A           ;сдвигается вправо на один пиксел
754B    B1              OR C            ;и объединяется с прежним значением
                                        ;(это выполняется аналогично
                                        ;предыдущей программе).
754C    77              LD (HL),A       ;Полученное значение заносится в память
754D    23              INC HL          ;и переход к адресу следующего байта.
754E    10 F7           DJNZ LOOP2      ;Повторение цикла для 4-х нижних линий.
7550    C1              POP BC          ;Повторение цикла для всех остальных
7551    10 ED           DJNZ LOOP1      ;символов символьного набора.
7553    C9              RET             ;Возврат в Бейсик.
```

Полученный при помощи программы 3 символьный набор, за счёт смещения нижней части вправо, получается как бы наклоненным влево. Некоторая модификация этой программы, а именно, замена команды OR C в ячейке 754Bh, командой NOP, позволяет получить только наклон символов, без расширения нижней части. Это выполняется в строке 150 Бейсик-программы. Наклон можно получить и в правую сторону. Для этого нужно использовать в ячейке 7549h вместо сдвига вправо (SRL) сдвиг влево (SLA), что выполняется в строке 160. Наклон вправо с расширенной нижней частью получается, если восстановить команду OR C в ячейке 754Bh, сохранив команду сдвига влево в ячейке 7549h. Это делается в строке 170.

Этим, конечно, не исчерпывается возможности по модификации символьного набора. Мы только показали, как практически можно достичь простейших эффектов. Уверены, что читатели разовьют эту тему и придумают новые оригинальные варианты такой модификации.

Стюарт Николс Перевод с английского: Пашорин В.И.

ПРИМЕНЕНИЕ АССЕМБЛЕРА ДЛЯ СОЗДАНИЯ БЫСТРОРАБОТАЮЩИХ ПРОГРАММ

Продолжение. (Начало см. в "ZX-РЕВЮ-93" N 1,2)

Заканчивая рассмотрение вопросов выдачи информации на экран, мы должны еще обсудить возможность печати в нижней части экрана, т.е. в строках с номерами 22 и 23.

Для этого применяют те же способы, что были описаны выше для вывода на главную часть экрана. Отличие состоит лишь в том, что для вывода в нижнюю часть экрана необходимо открывать другой канал, прежде чем применять команду печати RST 16. Как это сделать, показано в демонстрационной программе 1.8.

Здесь для вывода в нижнюю часть экрана сначала открывается канал с номером "минус 3" (253). Чтобы информационное сообщение "0:O.K.", появляющееся после исполнения программы не переместило выведенную нами на экран строку, необходимо иметь эквивалент команды PAUSE 0 в машинных кодах.

Останов программы можно реализовать, если использовать команду ассемблера HALT, которая приостанавливает работу процессора до очередного прерывания (до очередного сканирования клавиатуры, выполняемого процедурой ПЗУ KEYSCAN). Для реализации команды, аналогичной PAUSE 0, необходимо после команды HALT выполнить проверку состояния 5-го бита системной переменной FLAGS, который включается при нажатии произвольной клавиши. Если этот бит сброшен, то выполняется возврат к команде HALT, в противном случае программа выполняется дальше. Обратите внимание на то, что здесь 22-я строка считается нулевой, а 23-я - первой.

При выводе данных на экран весьма полезной для Вас может оказаться одна процедура из ПЗУ, находящаяся там по адресу 3652. Она позволяет уладить заданное число строк с экрана, причем отсчет строк начинается с 23-й строки, т.е. с ее помощью Вы можете как стирать информацию в системном окне, так и на основном экране. Пользоваться этой процедурой придется, по-видимому очень часто.

Перед вызовом процедуры необходимо записать в регистр B число удаляемых строк. Атрибуты экрана остаются такими, какие записаны в системной переменной ATTR-P. Так, например, нижние две строки экрана могут быть очищены с помощью следующей короткой программы:

```
LD B,2
CALL 3652
RET
```

Листинг 1.8

```
АДРЕС     МАШ. КОД        АССЕМБЛЕР       КОММЕНТАРИЙ
                          ORG 23760
23760     3E FD           LD A,253        ;Подготовка к открыванию канала.
23762     CD 01 16        CALL 5633       ;Открываем канал "системного
                                          ;окна компьютера"
23765     11 EA 5C        LD DE,DATA      ;Адрес начала выводимых
                                          ;данных.
23768     01 1E 00        LD BC,30        ;Длина данных с учетом
                                          ;управляющих кодов.
23771     CD 3C 20        CALL 8252       ;Вызов процедуры ПЗУ для
                                          ;печати сообщения.
WAIT                                      ;Начало процедуры "задерж-
                                          ;ки", аналогичной PAUSE 0.
23774     76              HALT            ;Прекращение работы процессора
                                          ;до прихода системного прерывания
23775     FD CB 01 6E     BIT 5,(IY+1)    ;Проверка пятого бита сис-
                                          ;темной переменной FLAGS.
```

```
23779       28 F9             JR Z,WAIT          ;Если он выключен - клавиша
                                                 ;не нажималась, то возврат
                                                 ;на метку WAIT.
23781       FD CB 01 AE       RES 5,(IY+1)       ;Если он включен, то вы-
                                                 ;ключаем его и продолжаем работу.
23785       C9                RET                ;Выход из процедуры.
DATA        22 00 10                             ;Аналог "AT 0, 10"
            73 78 80 85 84                       ;текстовое сообщение
            32 76 73 78 69                       ;"INPUT LINE 0", записан-
            32 48                                ;ное в кодах ASCII.
            22 01 10                             ;Аналог "AT 1, 10"
            73 78 80 85 84                       ;Текстовое сообщение
            32 76 73 78 69                       ;"INPUT LINE 1", записан-
            32 49                                ;ное в кодах ASCII.
```

2. Команды PLOT, DRAW и CIRCLE

PLOT

Команда PLOT x,y позволяет поставить на экране точку (включить пиксел), координата которой определяются значениями x и y, а цвет зависит от установленного значения INK. Координата x определяет номер столбца, а координата y - номер строки той точки экрана, где будет включен пиксел.

Пиксел, координаты которого равны 0,0, находится в нижнем левом углу экрана, а пиксел с координатами 175,255 - в верхнем правом углу экрана.

Для этой цели в ПЗУ компьютера имеется процедура PLOT, у которой есть две возможные точки входа.

Первая точка входа имеет адрес 8933 DEC (22E5 HEX). Перед вызовом этой процедуры по команде CALL 8933, необходимо в регистр B записать значение координаты y (0...175), а в регистр C - значение координаты x (0...255). Пример записи показан в программе 2.1.

Листинг 2.1

```
АДРЕС       МАШ. КОД          АССЕМБЛЕР          КОММЕНТАРИЙ
                              ORG 23760
23760       08 7D             LD B,125           ;Координата y=125.
23762       0E 4B             LD C,75            ;Координата x=75.
23764       CD E5 22          CALL 8933          ;Печать точки (125,75).
23767       C9                RET                ;Возврат.
```

Использование десятиричных чисел вместо шестнадцатиричных мы применяем только для того, чтобы начинающие читатели быстрее адаптировались к сути работы в машинном коде. Они более наглядны, если применяется раздельная загрузка регистров B и C. Однако, если записывать координаты сразу же в регистровую пару BC, то лучше все же использовать шестнадцатиричные числа - при такой записи проще представить положение пиксела на экране.

Вторая точка входа имеет адрес 8924 DEC (22DC HEX). При этом значения координат x и y должны быть предварительно записаны не в регистровую пару BC, а на стек так называемого встроенного калькулятора, причем значение y должно находиться на вершине стека, а значение x - под ним.

Начинающий программистам надо сказать и несколько слов о стеке калькулятора.

Для работы с действительными числами (числами с десятичной точкой) целочисленных регистров процессора явно недостаточно. Более мощные машины применяют для этого математический сопроцессор. В "Спектруме" же реализован другой, более удобный и дешевый подход - в ПЗУ внедрена обширная программа-калькулятор, имеющая свою оригинальную систему команд. Когда Вы работаете в БЕЙСИКе, калькулятор работает автоматически и для Вас все происходит незаметно. При работе в машинном коде

приходится программировать его вручную.

Основным местом для хранения исходных данных и результатов расчета калькулятора является его стек, хотя кроме этого калькулятор имеет еще и несколько ячеек памяти. Подробно с использованием калькулятора Вы можете познакомиться в нашей книге "Программирование в машинных кодах. Первые шаги. Практикум. Справочник."

Остается, правда, пока открытым вопрос, а как же записать координаты своей точки на стек калькулятора? Очень просто, для этой цели тоже используется процедура ПЗУ компьютера? Она находится по адресу 11560 DEC (2D28 HEX). Вам нужно поместить желаемое число в регистр A процессора и вызвать эту процедуру. Число будет передано на стек, чтобы отправить туда два числа, операцию надо сделать дважды. Естественно, что то число, которое будет передано последним и будет размещено на вершине стека.

На первый взгляд кажется, что работа с первой точкой входа намного удобнее, т.к. не надо ничего перебрасывать через стек калькулятора. Но это только на первый взгляд. На самом же деле в программах очень редко приходится что-то печатать в заранее известных координатах. Почти всегда программа сама и рассчитывает эти координаты. Делается это с помощью встроенного калькулятора и потому взять результаты расчета со стека и использовать их напрямую оказывается весьма удобно.

Листинг 2.2

```
АДРЕС       МАШ. КОД          АССЕМБЛЕР          КОММЕНТАРИЙ
                              ORG 23760
23760       3E 4B             LD A,75            ;Координата x=75.
23762       CD 28 2D          CALL 11560         ;Поместили x на стек каль-
                                                 ;кулятора.
23765       3E 7D             LD A,125           ;Координата y=125
23767       CD 28 2D          CALL 11560         ;Поместили ее на стек.
23770       CD DC 22          CALL 8924          ;Печать точки (125,75)
23773       C9                RET                ;Возврат.
```

Листинг 2.3

```
АДРЕС       МАШ. КОД          АССЕМБЛЕР          КОММЕНТАРИЙ
                              ORG 23760
23760       3E 03             LD A,3             ;
23762       FD 77 57          LD (IY+87),A       ;Включение двух младших
                                                 ;битов сист. перем. P_FLAG
23765       06 28             LD B,40            ;Координата y=40.
23767       0E 19             LD C,25            ;Координата x=25
23769       CD E5 22          CALL 8933          ;Печать точки (40,25) в ре-
                                                 ;жиме OVER 1.
23772       AF                XOR A              ;Очистка регистра A.
23773       FD 77 57          LD (IY+87),A       ;Восстановление P_FLAG.
23776       C9                RET                ;Возврат.
```

Листинг 2.4

```
АДРЕС       МАШ. КОД          АССЕМБЛЕР          КОММЕНТАРИЙ
                              ORG 23760
23760       2E 7F             LD L,64            ;Номер символа "@"=64 DEC.
23762       26 00             LD H,00            ;Теперь в паре HL число 64
23764       29                ADD HL,HL          ;Умножили его на 2.
23765       29                ADD HL,HL          ;Умножили его на 4.
23766       29                ADD HL,HL          ;умножили его на 8.
23767       ED 5B 36 5C       LD DE,(23606)      ;Загрузили в DE адрес, на
                                                 ;который указывает систем-
                                                 ;ная переменная CHARS.
23771       19                ADD HL,DE          ;Прибавили к нему "смеще-
                                                 ;ние" нашего символа от
                                                 ;начала символьного набо-
                                                 ;ра, ранее вычисленное в
                                                 ;регистровой паре HL.
23772       06 08             LD B,8             ;Счетчик строк в шабло-
                                                 ;не символа, равный 8.
```

```
23774       E5                  PUSH HL           ;Запомнили адрес начала
                                                  ;шаблона на машинном стеке
23775       0E 08               LD C,B            ;Счетчик столбцов в шабло-
                                                  ;не символа, равный 8.
23776       7E                  LD A,(HL)         ;Принимаем в аккумулятор
                                                  ;байт шаблона символа.
23778       C5                  PUSH BC           ;Запомнили содержимое BC,
                                                  ;чтобы освободить BC для
                                                  ;других дел.
23779       ED 4B 0E 5D         LD BC,(COORD)     ;Ввод в BC координат пози-
                                                  ;ции печати.
23783       17                  RLA               ;Ротация регистра A влево.
                                                  ;Содержимое старшего бита
                                                  ;при этом переходит во
                                                  ;флаг CARRY регистра F.
23784       F5                  PUSH AF           ;Запомнили регистры A и F
                                                  ;на машинном стеке.
23785       30 05               JR NC,23792       ;Если флаг CARRY выключен,
                                                  ;то точку печатать не надо.
                                                  ;Делаем обход печати.
23787       C5                  PUSH BC           ;Перед печатью точки запо-
                                                  ;минаем координаты.
23788       CD E5 22            CALL 8933         ;Вызов процедуры PLOT.
23791       C1                  POP BC            ;Восстановили координаты.
23792       0C                  INC C             ;Переход к очередному
                                                  ;столбцу символа.
23793       ED 43 0E 5D         LD (COORD),BC     ;Запомнили новую позицию
                                                  ;печати.
23797       F1                  POP AF            ;Восстановление пары AF.
23796       C1                  POP BC            ;Восстановление счетчиков в BC
23799       0D                  DEC C             ;Уменьшение счетчика столбцов.
23800       20 E8               JR NZ,23778       ;Если он еще не обнулился
                                                  ;возвращаемся для печати
                                                  ;соседней точки в ряду.
23802       C5                  PUSH BC           ;Если же он равен 0, то с
                                                  ;этим рядом закончили и надо
                                                  ;переходить к новому.
                                                  ;Сперва временно запомним счетчики.
23803       ED 4B 0E 5D         LD BC,(COORD)     ;Возьмем координаты.
23807       05                  DEC B             ;Уменьшим номер ряда.
23808       3E F8               LD A,248          ;Число 248 - это то же
                                                  ;самое, что и "минус 8".
23810       81                  ADD A,C           ;В результате сложения из
                                                  ;содержимого C (коорд. x) будет вычтено 8.
                                                  ;Результат оставлен в A.
23811       4F                  LD C,A            ;Перенесем его в C
23812       ED 43 0E 5D         LD (COORD),BC     ;и запомним в переменной COORD.
23816       C1                  POP BC            ;Восстановили счетчики рядов и столбцов.
23817       E1                  POP HL            ;Восстановление указателя.
23818       23                  INC HL            ;Приняли из символьного
                                                  ;набора конструкцию очередного ряда.
23819       10 D1               DJNZ 23774        ;Уменьшаем на единицу сче-
                                                  ;тчик рядов в B и если он
                                                  ;еще не обнулялся, то возвращаемся назад.
23821       C9 RET                                ;Выход

COORD
23822                           DEFB 100          ;Координата y=100
23823                           DEFB 20           ;Координата x=20
```

Программа 2.2 показывает использование процедуры PLOT со второй точкой входа 8924 DEC (22DC HEX). Процедура PLOT снимет два верхних числа со стека калькулятора и использует их как координаты для включения пиксела на экране. Числа на стеке при этом не сохраняются.

Использование калькулятора будет еще обсуждаться более подробно при рассмотрении программы 2.8. А пока рассмотрим программу 2.3, в которой показано, как в машинных кодах реализовать команду

```
PLOT OVER 1; x,y.
```

В этой программе перед вызовом процедуры PLOT необходимо включить нулевой и первый биты системной переменной P-FLAG (IY+87). А затем, после вызова процедуры PLOT, восстановить прежнее значение этих битов. Если этого не сделать, то и все последующие точки и символы будут выводиться на экран в режиме OVER 1.

Включение режима INVERSE аналогично включению режима OVER. Отличие состоит лишь в том, что теперь необходимо включить 2-ой и 3-ий биты системной переменной P-FLAG, т.е. надо записать в эту переменную число 12.

Используя точку входа 8933, мы можем написать программу для построения символа в любом месте экрана по точкам. При этом код ASCII символа должен быть в диапазоне 32...127, а шаблон символа (его конструкция) должен быть задан в участке символьного набора (см. программу 2.4).

Чтобы определить, где хранятся конструкции (шаблоны) символов, служит системная переменная CHARS (23606 DEC = 5C36 HEX). Тот адрес, который в ней хранится, указывает на 256 байтов ниже, чем начало символьного набора.

В программе 2.4 код выводимого на экран символа записывается в регистр L. Затем это значение умножается на 8 (т.к. в символьном наборе на конструкцию каждого символа использовано по 8 байтов) и результат суммируется с числом, которое хранится в системной переменной CHARS. Таким образом определяется начальный адрес восьми байтов, описывающих форму нужного вам символа.

Вы знаете, что первые 32 символа (с 0 по 31) являются непечатными, поэтому очень удобно, что CHARS указывает на 256 байтов ниже, чем начало набора, т.к. благодаря этому они-то как раз и оказываются пропущенными (32*8=256).

Теперь регистровая пара HL используется как указатель адреса очередного байта, а регистровая пара BC - как счетчик элементов матрицы 8x8 битов, описывающей форму символа. Здесь в B хранится номер текущего ряда, а в C - номер столбца. Регистр A используется для манипуляций с текущим байтом.

Листинг 2.5

```
АДРЕС      МАШ. КОД          АССЕМБЛЕР         КОММЕНТАРИЙ
                             ORG 23760
23760      2E 7F             LD L,87           ;Номер символа "W"=87 DEC
23762      ..........        ..........        ;Строки 23762... 23791
           ..........        ..........        ;этой программы соответ-
23791      ..........        ..........        ;ствуют программе 2/4.
23792      04                INC B             ;Переход к очередной стро-
                                               ;ке экрана.
23793      ED 43 0E 5D       LD(COORD),0BC     ;Запомнили позицию печати
23797      F1                POP AF            ;Восстановление пары AF.
23798      C1                POP BC            ;Восстановление счетчиков.
23799      0D                DEC C             ;Уменьшение счетчика
                                               ;столбцов.
23800      20 E8             JR NZ,23778       ;Если он еще не обнулялся,
                                               ;возвращаемся для печати
                                               ;соседней точки в ряду.
23802      C5                PUSH BC           ;Переходя к очередному ря-
                                               ;ду, запомнили счетчики.
33803      ED 4B 0E 5D       LD BC,(COORD)     ;Возьмем координаты.
23807      05                DEC B             ;Уменьшим номер ряда.
23808      3E F8             LD A,248          ;Уменьшение координаты
23810      80                ADD A,B           ;y на 8.
23811      47                LD B,A            ;
23812      ED 43 0E 5D       LD (COORD),BC     ;Запомнили координату.
23816      C1                POP BC            ;Восстановили счетчики ря-
                                               ;дов и столбцов.
23817      E1                POP HL            ;Восстановление указателя.
```

```
23818       23              INC HL          ;Очередной ряд.
23819       10 D1           DJNZ 23774      ;Уменьшаем на единицу сче-
                                            ;тчик рядов в B и если он
                                            ;еще не обнулился, то воз-
                                            ;вращаемся назад.
23821       C9              RET             ;Выход

COORD
23822                       DEFB 100        :Координата y=100
23323                       DEFB 20         ;Координата x=20
```

Так, например, одной из манипуляций является ротация байта влево командой RLA, которая производит циклическое смещение всех битов байта влево и перенос седьмого бита во флаг CARRY. Если этот бит равен единице, то флаг CARRY включается, следовательно будет включен и соответствующий пиксел на экране, если же бит равен 0, то не включатся ни флаг, ни пиксел. Команда RLA выполняется по 8 раз для каждого из 8 байтов, описывающих шаблон символа.

После каждой команды RLA координаты очередного пиксела изменяются. Начальное значение этих координат соответствует верхнему левому углу матрицы 8*8 битов, хранящей форму символа. Понятно, что при этом начальное значение координаты y должно быть в диапазоне 7...175, а значение координаты x может быть в диапазоне 0...255. Если же это значение окажется больше, чем 255, то это равносильно тому, что значение равно 0, так как если C=255 и выполняется команда INC C, то результат будет C = 0.

Эта программа может быть легко модифицирована для вывода на экран символа, развернутого на 90 градусов против часовой стрелки (см. программу 2. 5).

В этой программе начальные значения координат x,y соответствуют нижнему левому углу матрицы, хранящей форму символа и исследование байта идет не по горизонтали, а по вертикали.

С этого момента мы можем прийти к очень интересным решениям. Дело в том, что одной из первых проблем, встающих перед начинающим пользователем "Спектрума" является печать нестандартными шрифтами.

Обычно каждый спрашивает "а почему компьютер печатает только символами 8X8?" Почему нельзя печатать символами 5X8, 6X8 и даже 3X6, как например в программе "THE LAST WORD 2". Почему нельзя печатать символами 17X18? Может быть для тех, кто пишет программу на японском языке это было бы очень удобно.

Оказывается, если Вы печатаете из БЕЙСИКа или из машинного кода командой RST 16 или процедурой ПЗУ, которая опирается на RST 16, Вы действительно ограничены системным требованием размера 8x8. Так уж заложено в системных процедурах ПЗУ. Но если Вы выполняете графическую печать, то есть как бы не печатаете свои символы, а рисуете их по точкам, то можете обойти эти процедуры и создать свои, которые позволят вам делать все, что захотите.

Листинг 2.6

```
5 OVER 0: INK 0: PAPER 6: BORDER 3: CLS
10 LET x=24: LET y=79: LET h=5: LET w =2: LET a$="STOP THE TAPE"
15 INK 2: RANDOMIZE USR 32393: PAUSE 100
20 LET w=8: LET y=150: LET x=0: LET a$="PLOT"
30 INK 1: RANDOMIZE USR 32393
35 LET a$ ="    ": LET y=140: INK 1: RANDOMIZE USR 32393
250 PRINT AT 8,0:"Высота 1-22?   ", "Ширина 1-32?   ", "x-коорд. 0-255?    ", "Y-коорд. 0-
      195?   ", "СООБЩЕНИЕ?", "                          INK 0-9?",
255 LET z$=" "
257 LET v$=" "
260 PRINT OVER 1; AT 8,0; z$
262 INPUT h: PRINT AT 8,14; h;" ":OVER 1; AT 8,0; v$
265 PRINT OVER 1; AT 9,0; z$: INPUT w; PRINT AT 9,14; w: " "; OVER 1; AT 9,0; v$
270 PRINT OVER 1; AT 10,0;z$: INPUT x: PRINT AT 10,17;x;" "; OVER 1; AT 10,0;v$
272 PRINT OVER 1; AT 11,0:z$: INPUT y: PRINT AT 11,17; y:" "; OVER 1: AT 11,0;v$
```

```
273 PRINT OVER 1; AT 12,0;z$; AT 13,0; z$: INPUT a$: PRINT AT 13,0: a$; : FOR k=LEN a$ TO
      31: PRINT " ";: NEXT k: PRINT OVER 1; AT 12,0; v$; OVER 1: AT 13,0; v$
275 PRINT OVER 1; AT 14,0; z$: INPUT 1: PRINT AT 14,0; INK 1; OVER 1; v$
280 PAUSE 50: CLS: INK 1: RANDOMIZE USR 32393: INK 0
300 PRINT #0; AT 0,0 "Q - конец работы"; AT 1,0; "Прочие клавиши - повтор"
310 PAUSE 0: IF INKEY$="Q" OR INKEY$ = "q" THEN STOP
320 CLS: GO TO 250
```

Освоив способы вывода по точкам произвольного символа, мы можем перейти к нашей первой серьезной программе, позволяющей выводить по точкам на экран в произвольном месте целое сообщение, высота и ширина символов которого может быть выбрана нами.

Эта программа состоит из двух блоков. Первый блок - настроечный, он выполнен на БЕЙСИКе, т.к. от него не требуется выстродействие. второй блок - основной, он выполнен в машинных кодах. Связь между блоками нужна для передачи основных параметров: высота и ширина символов (h,w), начальные координаты сообщения (x,y) и само сообщение (a$). И эта связь осуществляется благодаря тому, что эти параметры в БЕЙСИК-программе являются заданы программными переменными, а в подпрограмму в машинных кодах передается с помощью команд POKE.

Может быть, Вы желаете использовать только программу машинных кодах, тогда Бейсик-программу можно опустить, передавая при этом необходимые параметры прямо в область памяти, зарезервированной для буфера принтера. Это выглядит более профессионально, но наличие БЕЙСИК-программы позволяет упростить работу и дает читателю возможность поэкспериментировать с различными значениями вышеупомянутых параметров.

БЕЙСИК-программа приведена в листинге 2.6.

Блок в машинных кодах приведен в листингах 2.7.1 - 2.7.5 и представляет собой расширенную версию программы 2.4.

Машиннокодовый блок включает в себя следующие процедуры: FIND, SET, START, PLOT и SKIP. для простоты рассмотрим их по отдельности.

Процедура FIND (32335 - 32380) приведена в листинге 2.7.1. Она выполняет поиск значения БЕЙСИК-переменной по ее имени, которое должно быть предварительно задано. У нас имя искомой переменной задано в ячейке памяти 23728.

Это нужно сделать для того, чтобы передать из БЕЙСИКа в машинный код параметры x,y,h,w и само текстовое сообщение.

Сама по себе задача передачи параметров из БЕЙСИКа в машинный код - весьма интересная и важная. Для этого существуют много разных способов. Наиболее распространенный прием - через параметры функции пользователя FN () мы рассмотрели и нашей книге "Элементарная графика" ("ИНФОРКОМ", М:, 1992, с. 111).

Здесь С. Николс предлагает прием передачи параметров через БЕЙСИК-переменные. Вот их поиском и занимается процедура FIND. Если Вам покажется сложным то, как она это делает, значит Вы не вполне знакомы с форматом БЕЙСИК-переменных, в котором они хранятся в памяти компьютера. Тогда посмотрите "ZX-РЕВЮ-92" на с. 166.

Процедура SET (32381-32392) приведена в листинге 2.7.2. Она служит для того, чтобы передать параметры x,y,b,w, разысканные процедурой FIND в области БЕЙСИК-переменных в область буфера принтера (весьма удобное место для хранения данных, обычно ничем не занятое в компьютере. Эта область расположена всегда в адресах 23396-23551). Сюда же передается и количество символов в выводимой на экран сообщении, а также коды символов сообщения. Наша основная программа будет работать с параметрами, беря их из области буфера принтера.

Выполнение машинного кода в программе 2.7 начинается с адреса 32393. Это стартовый адрес процедуры START. Она приведена в листинге 2.7.3. Сначала (32393...32418) регистровая пара BC определяется в качестве указателя буфера принтера, после чего вызывается процедура SET для записи кода переменной в ячейку 23728. Затем (32419... 32442) определяется количество символов в переменной a$ и переписываются

коды всех символов в буфер принтера. Процедура (32443... 32456) проверяет текущее значение координаты y. Если это значение будет больше 175, то выполняется операция y-176 и новые значения координат x и y записываются в ячейки 23728/9 и 23296/7.

Процедура PLOT (Листинг 2.7.4) представляет собой расширенную версию процедуры вывода символа на экран по точкам с добавлением циклов вывода на экран точки шириной w пикселов в h строках подряд.

Обратим внимание на то, что здесь неиспользуемые "Спектрумом" ячейки памяти 23728/9 служат для хранения текущих экранных координат позиции печати, а не для того, для чего они использовались в процедуре FIND.

Листинг 2.7.1

```
АДРЕС       МАШ. КОД          АССЕМБЛЕР          КОММЕНТАРИЙ
                              ORG 32335
FIND
32335       2A 4B 5C          LD HL,(23627)      ;23627 - адрес системной перем. VARS,
                                                 ;в которой хранится адрес, с которого
                                                 ;начинается размещение переменных БЕЙСИКа.
L2
32338       3A B0 5C          LD A,(23728)       ;Код имени БЕЙСИК-переменной.
32341       BE                CP (HL)            ;Сравнение и выход, если
32342       C8                RET Z              ;переменная найдена, в HL
                                                 ;остается ее адрес.
32343       CB 6E             BIT 5,(HL)         ;Если пятый бит в имени
32345       20 08             JR NZ,L1           ;переменной равен нулю, то
32347       23                INC HL             ;это либо массив, либо
32340       5E                LD E,(HL)          ;символьная строка. Нас
32349       23                INC HL             ;они не интересуют, их на-
32350       56                LD D,HL            ;до пропустить. Следующие
32351       19                ADD HL,DE          ;два байта содержат их
32352       23                INC HL             ;длину. На нее мы и перескакиваем.
32353       18 EF             JR L2              ;Возврат на повтор поиска.
L1
32355       CB 76             BIT 6,(HL)         ;Если шестой бит не равен
32357       20 0C             JR NZ,L3           ;нулю, то это либо одно символьная
                                                 ;переменная, либо параметр цикла.
                                                 ;Выясним это, перейдя на L3.
32359       23                INC HL             ;Если же это многосим-
32360       7E                LD A,(HL)          ;вольная переменная, то
32361       CB 7F             BIT 7,(HL)         ;пропускаем и ее и воз-
32363       28 FA             JR Z,-6            ;вращаемся для дальнейших
32365       11 06 00          LD DE,6            ;проверок на метку L2.
32368       19                ADD HL,DE
38369       18 DF             JR L2
L3
32371       CB 7E             BIT 7,(HL)         ;Если седьмой бит равен
32373       28 F6             JR Z,-10           ;нулю, это односимвольная
                                                 ;переменная (хоть и
                                                 ;не наша.) Вернемся на
                                                 ;32365, пропустим 6 байтов
                                                 ;и повторим поиск с L2.
32375       11 13 00          LD DE,19           ;Седьмой бит равен единице
32378       19                ADD HL,DE          ; - это параметр цикла.
                                                 ;В этом случае пропускаем
                                                 ;19 байтов и повторяем
52379       18 D5             JR L2              ;поиск с L2.
```

Листинг 2.7.2

```
АДРЕС       МАШ. КОД          АССЕМБЛЕР          КОММЕНТАРИЙ

SET
32381       32 B0 5C          LD (23728),A       ;в стандартном "Спектруме"
                                                 ;ячейки 23728/9 не используются.
```

```
                                        ;Мы храним в них имя
                                        ;переменной (x, y, h...)
32384       CD 4F 7E        CALL FIND   ;Поиск значения переменной
                                        ;по ее имени.
32387       23              INC HL      ;За именем переменной в
32388       23              INC HL      ;БЕЙСИКе хранится ее зна-
32389       23              INC HL      ;чение в пятибайтной фор-
                                        ;ме. Для целых чисел
                                        ;меньших 255 это форма:
                                        ;---------------------
                                        ;|имя|00|Знак|nn|00|00|
                                        ;---------------------
                                        ;  |          |
                                        ; (HL)       (HL)+3
                                        ;Вот для чего нужны три
                                        ;команды INC HL подряд.
32390       7E              LD A,(HL)   ;Переброска параметра в
32391       02              LD (BC),A   ;буфер принтера.
32392       C9              RET         ;Выход из процедуры.
```

Процедура SKIP (Листинг 2.7.5) завершает подпрограмму. Она ведет учет изменения экранной координаты позиции печати. В частности, она осуществляет переход к следующей строке, а когда достигнут низ экрана (y>175), она выполняет переход на первую строку за счет операции "y-175".

Выше мы отметили, что в процедуре ПЗУ PLOT имеется еще одна точка входа по адресу 8924, с помощью которой можно напечатать на экране точку, координаты которой заданы на вершине стека калькулятора. Мы говорили о том, что это может быть удобным при печати результатов расчета калькулятора.

Не углубляясь в подробности работы процедуры CALCULATOR, рассмотрим работу программы, которая использует адрес 8924 для входа в процедуру PLOT.

Листинг 2.7.3

```
АДРЕС       МАШ. КОД        АССЕМБЛЕР       КОММЕНТАРИЙ

START                                       ;Точка входа в программу.
32393       01 00 5B        LD BC,23296     ;Начало буфера принтера.
32396       3E 7B           LD A,120        ;120 - код буквы "x".
32598       CD 7D 7E        CALL SET        ;в 23296 - коорд. "x".
32401       03              INC BC          ;Очередной байт буфера.
32402       3E 79           LD A,121        ;121 - код буквы y.
32405       CD 7D 7E        CALL SET        ;в 23297 - коорд. "y".
32407       03              INC BC          ;Очередной байт буфера.
32408       3E 68           LD A,104        ;104 - код буквы h.
32410       CD 7D 7E        CALL SET        ;В 23298 - высота "h".
32413       03              INC BC          ;Очередной байт буфера.
32414       3E 77           LD A,119        ;119 - код буквы w.
32416       CD 7D 7E        CALL SET        ;в 23299 - ширина "w".
32419       3E 41           LD A,65         ;65 - код буквы A.
52421       32 B0 5C        LD (23728),A    ;Подготовка и проведение
32424       CD 4F 7E        CALL FIND       ;поиска переменной a$.
32427       23              INC HL          ;
32428       5E              LD E,(HL)       ;В DE помещается длина
32429       23              INC HL          ;нашего сообщения и пере-
32430       56              LD D,(HL)       ;дается в буфер принтера
32431       ED 53 04 5B     LD(23300),DE    ;По адресу 23300.
32435       D5              PUSH DE         ;Переброска длины сообще-
32436       C1              POP BC          ;ния из DE в BC.
32437       23              INC HL          ;Переброска самого текста
32438       11 05 5B        LD DE,23301     ;сообщения в буфер принте-
32441       ED B0           LDIR            ;ра начиная с 23301.
32443       2A 00 5B        LD HL,(23296)   ;В HL - координаты x,y.
                                            ;Причем y находится в H.
```

```
32446      AF               XOR A           ;Сброс флага переноса (это
                                            ;необходимо перед операцией SBC).
32447      7C               LD A,H          ;Координата y.
32443      DE B0            SBC A,176       ;Проверка на >= 176.
32450      38 04            JR C,+4         ;Если нет, то переход на
                                            ;адрес 32456 (все O.K.)
32452      67               LD H,A          ;В противном случае остав-
32453      22 00 5B         LD (23296),HL   ;ляем "y минус 176" и за-
32456      22 B0 5C         LD (23728),HL   ;поминаем копию в 23728/9.
                                            ;Итак, в 23728 - "x";
                                            ;в 23729 - "y-176".
32459      21 05 5B         LD HL,23301     ;Указание на начало текста

LP5
32462      E5               PUSH HL         ;Запомнили начало текста.
32463      7E               LD A,(HL)       ;Взяли символ.
32464      26 00            LD H,0          ;Код символа передаем
32466      6F               LD L,A          ;в HL и
3246т      29               ADD HL,HL       ;умножаем его код на 8.
32468      29               ADD HL,HL
32469      39               ADD HL,HL
32470      11 00 3C         LD DE,15360
32473      19               ADD HL,DE       ;Нашли адрес, начиная с
                                            ;которого в ПЗУ хранится
                                            ;шаблон этого символа.
32474      06 08            LD B,8          ;Счетчик байтов в шаблоне
LP4
32476      C5               PUSH BC         ;Запомнили его.
52477      ED 4B 01 5B      LD BC,(23297)   ;в "B" - высота символа;
                                            ;в "C" - его коорд. "y".
LP3
32481      7E               LD A,(HL)       ;Байт шаблона символа.
32482      E5               PUSH HL         ;Запомнили его.
32483      C5               PUSH BC         ;Запомнили "h" и "y".
32484      06 08            LD B,8          ;Счетчик битов в байте
                                            ;шаблона.
LP2
32486      C5               PUSH BC         ;Запомнили текущий байт.
32487      17               RLA             ;Ротация байта.
32488      F5               PUSH AF         ;Запомнили результат рота-
                                            ;ции вместе с флагами.
32489      DA F9 7E         JP C,PLOT       ;Если флаг переноса вклю-
                                            ;чился, переходим на 32505
                                            ;для печати точки.
32492      2A 03 5B         LD HL,(23299)   ;В регистр "L" помещается
                                            ;ширина "w".
32495      3A B0 5C         LD A,(23728)    ;Координата "x".
32498      85               ADD A,L         ;Нашли "x+w"
32499      32 B0 5C         LD (23728),A    ;Запомнили в 23728.
32502      C3 0F 7F         JP SKIP         ;Обход.
```

Программа 2.8 демонстрирует построение по точкам синусоиды, описываемой формулой

```
88 + 80*SIN(A/128*PI)
```

с применением процедуры CALCULATOR.

С помощью этой процедуры можно выполнить сложные арифметические операции, используя так называемые коды калькулятора. Обычно при вызове этой процедуры выполняется операция над двумя числами, являвшимися верхними на калькуляторном стеке, но возможны операции и над символьными величинами.

Например, если на вершине стека записаны два числа: 1234 и 2, а после включения процедуры CALCULATOR (RST 40) записан код 4 (DEFB 4), то эти два числа будут извлечены из калькуляторного стека, перемножены между собой, а их произведение будет записано обратно на вершину калькуляторного стека. Теперь на стеке будет записано только одно

число, равное 2468, если до этого там кроме сомножителей ничего не было записано.

Целые положительные числа могут быть занесены на стек следующими способами:

STACK VAL A - CALL 11560

STACK VAL BC - CALL 11563,

т.е. число предварительно записывается в регистр a или регистровую пару BC, а затем вызывается соответствующая процедура.

В программе 2.8 используется вызов STACK VAL A для размещения на стеке чисел в следующем порядке:

```
переменная A
88
80
переменная A
128
```

Вызов процедуры CALCULATOR осуществляется по команде RST40. Байты, записанные после этой команды, определяют вид выполняемой операции. В программе 2.8 сразу же после команды RST 40 следует байт, равный 5, что означает "Деление". При этом из стека извлекаются два верхних числа, выполняется деление одного числа на другое и результат записывается обратно в калькуляторный стек. Следующий байт - 163 является кодом операции "stack PI/2"- он переписывает на стек значение PI/2, хранимое в ПЗУ. Затем следует код 49 (duplicate top value) - он дублирует верхнее число на стеке, а код 15 "add top two values" производит суммирование двух верхних чисел и записывает результат обратно на стек. Теперь на вершине стека записано число PI. Следующий код 4 (multiply top two values) перемножает два верхних числа, т.е. число PI и A/128, и результат возвращается на стек. Код 31 (SIN of top value) определяет синус числа на вершине калькуляторного стека, а последующие коды 4 и 15 завершает расчет по записанной формуле. Последний код 56 "end calc" выполняет выход из процедуры, т.е. как бы выключает калькулятор и осуществляет возврат в программу в машинных кодах.

Листинг 2. 7.4

```
АДРЕС       МАШ. КОД          АССЕМБЛЕР          КОММЕНТАРИЙ

PLOT                                             ;Процедура печати точки.
32505       ED 4B B0 5C       LD BC, (23298)     ;В "B" - ширина "w"
                                                 ;В "C" - высота "h".
LP1
32509       C5                PUSH BC            ;Запомнили на стеке.
32510       ED 4B B0 5C       LD BC,(23298)      ;Повторили на стеке еще
32514       C5                PUSH BC            ;раз.
32515       CD E5 22          CALL 8933          ;Вызов процедуры печати
                                                 ;точки.
32518       C1                POP BC             ;Переход к соседней
32519       0C                INC C              ;точке.
32520       ED 43 B0 5C       LD (23728),BC      ;Запомнили текущую коор-
32524       C1                POP BC             ;динату.
32525       10 EE             DJNZ LP1           ;Если не все точки напе-
                                                 ;чатаны, возврат на 32509
```

Листинг 2.7.5

```
АДРЕС       МАШ. КОД          АССЕМБЛЕР          КОММЕНТАРИЙ

SKIP
32527       F1                POP AF             ;Восстановили текущий байт
                                                 ;шаблона.
32528       C1                POP BC             ;Восстановили счетчик
32529       10 03             DJNZ LP2           ;Уменьшили его и, если он
                                                 ;не обнулялся, возврат
                                                 ;на LP2 для последующей
                                                 ;ротации.
32531       3A 00 5B          LD A,(23296)       ;Координата "x".
32534       21 B0 5C          LD HL,23728        ;
32537       77                LD (HL),A          ;Принимаем "x" в качестве
                                                 ;текущей экранной коорди-
```

```
                                        ;наты.
32536     23              INC HL        ;Указание на "y".
32539     AF              XOR A         ;Сброс флага переноса,
32540     7E              LD A,(HL)     ;Координата "y"
32541     DE B0           SBC A,176     ;Проверка на "y>175".
32543     38 03           JR C,+3       ;Если так, переход к 32546
32545     77              LD (HL),A     ;"y"
32546     18 08           JR +8         ;Переход на 32556.
32548     7E              LD A,(HL)     ;"y-176".
32349     FE 00           CP 0          ;Проверка на "ноль".
32551     20 02           JR NZ,+2      ;Если не 0, переход на ад-
                                        ;рес 32555.
32553     36 B0           LD (HL),176
32555     35              DEC (HL)
32556     C1              POP BC        ;
32557     E1              POP HL
32558     10 B1           DJNZ LP3
32560     23              INC HL        ;Переход к следующему бай-
                                        ;ту шаблона.
32561     C1              POP BC        ;Восстановили счетчик.
32562     10 A8           DJNZ LP4      ;Если он еще не обнулился,
                                        ;переход на LP 4.
32564     3A 03 5B        LD A,(E3299)  ;Ширина символа "w".
33567     87              ADD A,A       ;Умножаем
32566     67              ADD A,A       ;ее
32569     67              ADD A,A       ;на 8.
32570     6F              LD L,A        ;Запомнили в L.
32571     3A B0 5C        LD A,(25728)  ;Новая экранная коорд. "x"
32574     85              ADD A,L
32575     32 00 5B        LD (23296),A  ; Запомнили ее.
32578     32 B0 5C        LD (23728),A  ;- "" -- "" -
32581     3A 01 5C        LD A,(23297)  ;Новая экранная коорд. "7"
32584     3A B1 5C        LD (23729),A  ;Запомнили ее.
32587     E1              POP HL        ;Восстановили указатель на
                                        ;текущий символ.
32588     23              INC HL        ;Переход к новому символу.
32569     3A 04 5B        LD A,(23300)  ;Длина сообщения.
32592     3D              DEC A         ;Умножаем ее на 1.
32593     C8              RET Z         ;Проверка на конец. Если
                                        ;так, то выход.
32594     32 04 5B        LD (23300),a  ;Запомнили новую длину.
32597     C3 CE 7E        JP LP5        ;Возврат на печать очеред-
                                        ;ного символа.
```

Существуют еще много других кодов калькулятора, которые могут быть использованы в Ваших программах для различных целей. С некоторыми из них мы еще познакомимся в следующих главах.

И еще одно важное замечание. Перед возвратом в BASIC-систему после применения процедуры CALCULATOR необходимо полностью очистить калькуляторыый стек от всех записей. В программе 2.8 после последнего кода 56 на стеке все же остаются два числа: переменная A и результат вычисления по формуле 88+80*SIN(A/128*PI). Процедура PLOT(CALL 6924) извлекает эти два значения, тем самым полностью очищая стек и использует эти два числа как координаты для пиксела.

Код программы для построения синусоиды приведен в листинге 2.8.

Листинг 2.8

```
АДРЕС     МАШ. КОД        АССЕМБЛЕР     КОММЕНТАРИЙ

23760     AF              XOR A         ;Обнуление аккумулятора.
LOOP
23761     F5              PUSH AF       ;Запомнили его на стеке.
23762     CD 28 2D        CALL 11560    ;На кальк. стеке - текущее
                                        ;значение аргумента.
23765     3E 5B           LD A,88       ;На кальк. стеке -
```

```
23767     CD 28 2D          CALL 11560        ;88, A.
23770     3E 50             LD A,80           ;На кальк. стеке -
23772     CD 28 2D          CALL 11560        ;80, 88, A.
23775     F1                POP AF            ;В акк-ре текущее значение
                                              ;аргумента.
23776     F5                PUSH AF           ;Запомнили его на стеке.
23777     CD 28 2D          CALL 11560        ; A, 80, 88, A.
23780     3E 80             LD A,128
23782     CD 28 2D          CALL 11560        ; 128, A, 80, 88, A.
23765     EF                RST 40            ;Включение калькулятора.
23766     DEFB 05           division          ; A/128, 80, 88, A.
23787     DEFB 163          stack PI/2        ; PI/2, A/128, 80, 88, A.
23788     DEFB 49           duplicate         ; PI/2, PI/2, a/128, 80, 88, A.
23789     DEFB 15           add               ; PI, A/128, 80, 88, A.
23780     DEFB 04           multiply          ; PI*A/128, 80, 88, A.
23781     DEFB 31           Sin               ; SIN(A/128*PI), 80, 88,A.
23782     DEFB 04           multiply          ; 80*SIN(A/128*PI), 88, A.
23783     DEFB 15           add               ; 88+80*SIN(A/128*PI), A.
23784     DEFB 56           endcalc           ;Выключение калькулятора.
23795     CD DC 22          CALL 8924         ;Вызов процедуры PLOT.
23798     F1                POP AF            ;Текущее значение аргумента.
23799     3C                INC A             ;Приращение аргумента.
23800     FE 00             CP 0              ;Проверка на конец экрана.
23802     20 D5             JR NZ,LOOP        ;Возврат для расчета и
                                              ;печати следующей точки.
23804     C9                RET               ;Выход из процедуры.
```

DRAW x,y.

Две точки входа в ПЗУ существуют и для реализации команды DRAW x,y. Для входа через первую точку (CALL 9402) необходимо, чтобы в регистре B было записано ABS(y), а в регистре C - ABS (x). Регистровая пара DE служит для хранения знака приращения координат x и y. Регистр D хранит знак приращения по x (SGN dx), а регистр C - знак приращения по y (1, если число положительное и 255, если число отрицательное). Программа 2.9 является модифицированной версией программы 2.8, которая демонстрирует реализацию команды DRAW. Обратите внимание, что перед вызовом процедуры DRAW, начальное значение регистровой пары H'L' (альтернативная) должно быть сохранено. Для этого содержимое этой пары записывается в машинный стек.

Начальное значение регистровой пары H'L' не должно быть потеряно в ходе выполнения программы, написанной пользователем, иначе при возврате в BASIC-систему произойдет сбой в работе компьютера. Начальное значение регистровой пары H'L' восстанавливается после команды DRAW. Команды OVER и INVERSE включаются таким же образом, как и PLOT OVER/INVERSE.

Для реализации команды DRAW x,y через вторую точку входа (CALL 9335) необходимо, чтобы значения x и y были записаны в калькуляторном стеке, причем значение y должно быть верхним в стеке. Чтобы использовать эту точку входа, необходимо прочитать главу 7, для ознакомления со способом записи 5-ти байтных чисел в калькуляторный стек.

Листинг 2.9

```
АДРЕС     МАШ. КОД          АССЕМБЛЕР         КОММЕНТАРИЙ

23760     06 5F             LD B,95           ;Координата "y".
23762     0E 00             LD C,00           ;Координата "x".
23764     CD E5 22          CALL 8933         ;Печать исходной точки.
23767     D9                EXX               ;Включение регистров
                                              ;альтернативного набора.
23768     E5                PUSH HL           ;Сохранение пара H'L'.
23769     D9                EXX               ;Выключение альтернативного набора.
23770     06 00             LD B,0            ;Приращение по "y"=0.
23772     0E FF             LD C,255          ;Приращение по "x" = 255.
23774     16 01             LD D,1            ;Знак приращ. по "x"="+".
23776     1E 01             LD E,1            ;Знак приращ. по "y"="+".
```

```
23778       CD BA 24          CALL 9402          ;Рисуем прямую линию.
23781       D9                EXX                ;Включение регистров
                                                 ;альтернативного набора.
23782       E1                POP HL             ;Восстановление пары H'L'.
23783       D9                EXX                ;Выключение альтернативного набора.
23784       AF                XOR A              ;Обнуление аккумулятора.
LOOP
23765       F5                PUSH AF            ;Запомнили его на стеке.
23766       CD 28 2D          CALL 11560         ;На кальк. стеке - текущее
                                                 ;значение аргумента.
23789       3E 78             LD A,120           ;На кальк. стеке -
23791       CD 28 2D          CALL 11560         ;120,A.
23794       3E 28             LD A,40            ;На кальк. стеке -
23796       CD 28 2D          CALL 11560         ;40, 120, A.
23799       F1                POP AF             ;в акк-ре текущее значение
                                                 ;аргумента.
23600       F5                PUSH AF            ;Запомнили его на стеке.
23601       CD 28 2D          CALL 11560         ;A, 40, 120, A.
23804       3E 10             LD A,16
23806       CD 28 2D          CALL 11560         ;16, A, 40, 120, A.
23809       EF                RST 40             ;Включение калькулятора.
23810       DEFB 05           division           ; A/16, 40, 120, A.
23811       DEFB 163          stack PI/2         ; PI/2, A/16, 40, 120, A.
23813       DEFB 49           duplicate          ; PI/2, PI/2, A/16, 40, 120, A.
23813       DEFB 15           add                ; PI, A/16, 40, 120, A.
23814       DEFB 04           multiply           ; PI*A/16, 40, 120, A.
23815       DEFB 31           Sin                ; SIN(A/16*PI), 10, 120, A.
23816       DEFB 04           multiply           ; 40*SIN(A/16*PI), 120, A.
25617       DEFB 15           add                ; 120+40*SIN (A/16*PI), A.
23618       DEFB 56           endcalc            ;Выключение калькулятора.
23819       CD DC 22          CALL 8924          ;Вызов процедуры PLOT.
23822       D9                EXX                ;Включение альтерн. набора
23823       E5                PUSH HL            ;Сохраняя H'L'.
23824       D9                EXX                ;Выключение альтерн. наб.
23825       06 32             LD B,50            ;Приращение по "y".
23827       0E 00             LD C,0             ;Приращение по "x".
23829       16 FF             LD D,255           ;Знак приращения по "y"
                                                 ;"минус".
23831       1E 01             LD E,01            ;Знак приращения по "x" -
                                                 ;"плюс".
23833       CD BA 24          CALL 9402          ;Рисование линии.
23836       D9                EXX                ;Включение альтерн. набора
23837       E1                POP HL             ;Восстановление пары H'L'.
23838       D9                EXX                ;Выключение альтернатив-
                                                 ;ного набора.
23839       F1                POP AF             ;Текущее значение аргумента.
23840       3C                INC A              ;Приращение аргумента.
23841       FE 00             CP 0               ;Проверка на конец экрана.
23843       20 D5             JR NZ, LOOP        ;Возврат для расчета и
                                                 ;печати следующей точки.
23845 C9                      RET                ;Выход из процедуры.
```

DRAW x,y,a

Точкой входа в ПЗУ для реализации этой команды служит адрес 9106. В этом случае необходимо, чтобы значения x, y и a были записаны в калькуляторный стек в таком порядке, чтобы значение "a" было верхним на стеке. Опять же, начальное значение регистровой пары H'L' (альтернативной) должно быть сохранено.

CIRCLE x,y,r

Точкой входа в ПЗУ для реализации этой команды служит адрес 9005. При этом необходимо, чтобы значения x,y, и r были записаны в калькуляторном стеке. Начальное значение регистровой пары H'L' должно сохраниться в отдельном месте в течение работы

процедуры CIRCLE.

Программа 2.10 демонстрирует метод рисования концентрических окружностей. Обратите внимание на тот факт, что эта программа в машинных кодах выполняется также медленно, как и на языке BASIC. Это потому, что процедура CIRCLE в ПЗУ достаточно длинная. Если в Вашей программе возникнет необходимость построить окружность, то будет быстрее, если Вы будете строить ее по точкам, используя для постановки пиксела координаты, записанные как последовательность байтов с именем DATA.

Машинный код процедуры представлен в листинге 2.10. В качестве аналога выступает следующая БЕЙСИК-программа:

```
10 REM CIRCLE x,y,r
20 FOR b= 1 TO 2
30 FOR a= 1 TO 21 STEP 2
40 CIRCLE OVER 1; 128,88,a
50 NEXT a
60 NEXT b
```

Листинг 2.10

```
АДРЕС       МАШ. КОД          АССЕМБЛЕР          КОММЕНТАРИЙ

23760       06 02             LD B,2             ;FOR b=1 TO 2
L2
23762       C5                PUSH BC
23763       3E 01             LD A,1             ;FOR a=1 TO ...
L1
23765       F5                PUSH AF
23766       FD 36 57 03       LD (IY+87),3       ;OVER 1
23770       3E 80             LD A,128
23772       CD 28 2D          CALL 11560         ;На кальк. стеке - 128.
23775       3E 58             LD A,88
23777       CD 28 2D          CALL 11560         ;На кальк. стеке - 88,128.
23780       F1                POP AF             ;текущее "a".
23781       F5                PUSH AF
23782       CD 28 2D          CALL 11560         ;На к. стеке - a, 88, 128.
23785       D9                EXX
23786       E5                PUSH HL            ;Сохранение H'L'.
23787       D9                EXX
23788       CD 2D 23          CALL 9005          ;CALL CIRCLE
23791       D9                EXX
23792       E1                POP HL             ;Восстановление H'L'.
23793       D9                EXX
23794       F1                POP AF             ;Текущее "a".
23795       3C                INC A
23796       3C                INC A              ;STEP 2
23797       FE 17             CP 23              ;a = 23?
23794       20 DC             JR NZ, L1          ;Цикл не завершен.
23801       C1                POP BC             ;цикл по "A" завершен.
23802       10 D6             DJNZ L2            ;Конец цикла по "b".
23804       FD 36 57 00       LD (IY+87),D       ; - OVER 0.
23808       C9                RET                ;Выход
```

3. Счет

Во многих игровых программах необходимо постоянно вести счет, например, счет очков, потерянных "жизней", времени и т.д. Известны, по крайней мере, три способа организации счета в программах и вывода результатов счета на экран.

Первый настолько непригляден и занимает так иного памяти, что мы рассмотрим его кратко, лишь в общих чертах. Способ этот позволяет организовать счет в программе от 0 до очень большого числа. Максимально возможное число определяется только количеством байтов, зарезервированных для представления этого числа. Например, для того, чтобы получить счетчик от 0 до 999 999, необходимо зарезервировать 6 байтов, т. е. для каждого

разряда числа нужен 1 байт. Поскольку показание счетчика выводится на экран как значение символьной переменной, то на экране в начальный момент времени будет 6 нулей: 000 000. Затем, с увеличениен значения на 1, изменения произойдут только в разряде единиц. Если же последнее записанное в этом разряде число равно 9, то с добавлением 1 этот разряд обнулится, а единица добавится к числу в следующем разряде. Продедура проверки значения числа выполняется для каждого разряда и, если во всех шести разрядах записано число 9, то дальнейший счет прекращается. Для того, чтобы шаг счетчика был больше, чем 1, необходимо включить в процедуру счета специальные циклы.

Второй способ используется для создания счетчиков, показания которых не выходят за пределы диапазона 0...65535. Реализация этого способа показана в программе 3.1. В этой программе используется одна из процедур ПЗУ, позволяющая выводить на экран значение числа, записанного на вершине калькуляторного стека. Вызов этой процедуры осуществляется командой CALL 11747. При этом текущее значение счетчика может, например, храниться в неиспользуемой системной переменной 23726. Значение счетчика извлекается из этой системной переменной, увеличивается на 1, а затем вновь записывается в эту переменную и на вершину калькуляторного стека. После этого определяются параметры PRINT AT и вызывается процедура 11563, печатающая содержимое стека калькулятора.

Листинг 3. 1

```
        LD BC,0                 ;Инициализация
BEGIN                           ;счетчика по
        LD (23728),BC           ;адресу 23728.
        CALL 11563              ;Переброска на
                                ;стек калькулятора
        LD A,2                  ;Открыли канал
        CALL 5633               ;печати на экран.
        LD A,22                 ;
        RST 16                  ;Аналог ... AT 22
        LD A,16
        RST 16                  ;Аналог AT 22,16
        CALL 11747              ;Печать счетчика.
        LD BC,(23728)           ;Вызов счетчика.
        INC BC                  ;Приращение.
        LD A,B                  ;Проверка на
        OR C                    ;обнуление.
        RET Z                   ;Выход, если 0.
        BIT 5,(IY+1)            ;Проверка не была
                                ;ли нажата клавиша
        JR Z, BEGIN             ;Повтор, если не
                                ;было нажатия.
        RET                     ;Выход, если кла-
                                ;виша была нажата.
```

Если в регистровой паре значение числа становится равным 65535, то осуществляется возврат в BASIC систему. В эту программу включена также процедура EXIT (строки 23795...23800), благодаря которой имеется возможность прервать счет и вернуться в BASIC систему, если нажата какая-нибудь клавиша. Это удобно, так как счет от 0 до 65535 выполняется несколько минут.

Третий способ также основан на использовании калькулятора. Выше, когда мы говорили о калькуляторном стеке, мы не останавливались на детальном рассмотрении способа записи и хранения чисел в стеке. Числа в стеке хранятся в 5-байтной форме, что позволяет калькулятору легко выполнять действия с ними. Более подробно об этом будет сказано в седьмой главе. Здесь же приводим программу, которая выполняет счет от 0 до бесконечно большого числа с шагом 250. При достижении показания счетчика 99 999 999, форма вывода на экран меняется на E-форму (1E+9). Однако счетчик, у которого диапазон представляемых чисел достигает 100 млн. вполне пригоден для большинства программ и на этот факт можно не обращать внимание.

В программе 3.2 есть много важных процедур, которые необходимо объяснить. Начальное значение счетчика, равное нулю, записывается в стек с помощью кода калькулятора 160 (stack_zero). Это число занимает 5 верхних байтов стека, содержимое которых затем копируется в буфер принтера, используемого как "хранилище" для счетчика. Копирование выполняется командой LDIR. Значение, записанное в регистровой паре HL при этом считается равным STACKEND-5.

Значение счетчика выводится на экран с помощью процедуры 11747. Затем в стек записывается число 250 (шаг изменения показания счетчика) и число из "хранилища" в 5-байтной форме. Следующая команда вызова калькулятора RST 40 и код калькулятора 15(add) обеспечивают суммирование двух чисел, записанных на вершине стека, и осуществляется переход на начало цикла для передачи нового значения в "хранилище" и вывода этого значения на экран. В этой программе предусмотрена также процедура преждевременного выхода из цикла счета, если будет нажата любая клавиша. Перед возвратом в BASIC-систему после окончания счета стек обнуляется путем перемещения последнего записанного там значения.

Для создания счетчика, значение которого с каждым шагом уменьшается, необходимо в стек первоначально записать максимальное значение счетчика (запись в стек чисел больше, чем 65535, показана в главе 7), а вместо кода калькулятора для сложения - 15 (add) следует использовать код для вычитания 3 (subtract). Необходимо помнить, что при вычитании верхнее в стеке число вычитается из числа, записанного вторым. После каждой операции вычитания проверяется достигнут ли 0 (либо включен ли 7 бит результата) и если это так, то счет прекращается. Перед выводом очередного значения счетчика необходимо очищать экран, чтобы избежать ошибок, связанных с устареванием экранной информации. Например, если экран не очищать, то при изменении значения счетчика с 1000 на 999 на экране будет изображено не 999, а 9990.

Листинг 3.2

```
            ORG 23760
            RST 40                  ;включили кальк-р.
            DEFB 160                ;На стек - 0.
            DEFB 56                 ;Выключили кальк-р.
L1
            LD DE,23296             ;начало буфера
                                    ;принтера.
            LD BC,5                 ;Число перебрасы-
                                    ;ваемых байтов.
            LDIR                    ;Переброска.
            LD A,2                  ;Канал экрана.
            CALL 5633               ;Открыли канал.
            LD A,22                 ;... AT 22 ...
            RST 16
            LD A,11                 ;...AT 22,11
            RST 16
            CALL 11747              ;Печать показаний
                                    ;счетчика.
            LD A,250                ;Шаг счетчика.
            CALL 11560              ;Поместили его на
                                    ;стек кальк-ра.
            LD HL,23296             ;Начало буфера
                                    ;принтера.
            LD DE,(23653)
                                    ;23553 - системная
                                    ;переменная STKEND.
            LD BC,5                 ;Число перебрасы-
                                    ;ваемых байтов.
            LDIR                    ;Переброска.
            LD (23653),DE           ;Новое значение
                                    ;STKEND.
            RST 40                  ;Включили кальк-р.
            DEFB 15                 ;Код сложения (add)
```

```
DEFB 56                 ;Выключение к-ра.
BIT 5(IY+1)             ;Проверка не была
                        ;ли нажата какая -
                        ;либо клавиша.
JR Z,L1                 ;Если нет, то
                        ;повтор.
LD HL,(23653)           ;(STKEND)
DEC HL
DEC HL
DEC HL
DEC HL
DEC HL
LD (23553),HL           ;(STKEND)-5
RET                     ;Выход
```

И завершим мы материал этого номера небольшой игрой, в которой объединим то, о чем Вы сегодня прочитали. Программа 3.3 показывает возможность использования счетчика для измерения скорости Вашей реакции.

Программа состоит из двух частей: первая часть написана на БЕЙСИКе и служит в качестве интерфейса между пользователем и компьютером. Она не требует скорости в работе. Машинный код служит для выдачи сообщений и запросов нестандартным шрифтом и для измерения скорости реакции. Печать нестандартным шрифтом выполняется с помощью блока машинного кода, приведенного в листинге 2.7, рассмотренного ранее. Он должен быть подгружен, начиная с адреса 32335. измерение скорости реакции выполнит процедура 3.3.2, приведенная ниже.

Измерение скорости вашей реакции производится следующим образом. Сначала процедура 2.7 нарисует на экране большую букву от A до Z, после чего будет включен режим CAPS LOCK и Вам надо нажать соответствующую клавишу. Время Вашей реакции замерит процедура 3.3.2.

В ней применен один хитрый прием. Дело в том, что рисование с помощью PLOT буквы на экране происходит не очень быстро, и за то время, пока идет этот процесс, Вы можете в принципе догадаться, что это за буква и подготовиться к нажатию соответствующей клавиши. Чтобы этого не было, буква рисуется желтым цветом INK=6 по желтому фону PAPER=6 и ее на экране не будет видно. Когда же дело дойдет до измерения Вашей реакции, процедура 3.3.2 включит на экране цвет пикселов INK=1 (черный). Произойдет это с огромной скоростью и буква появится мгновенно.

БЕЙСИК-программа стартует со строки 9800, в которой записана подгрузка модулей в машинных кодах. Поэтому набрав Бейсик выгрузите его на ленту со строкой автостарта 9800:

```
SAVE "counter" LINE 9800
```

Листинг 3.3.1

```
12 PAPER 6: CLS
15 DATA 87,143,1,2, "COUNT"
20 DATA 95,63,1,2, "SLOW"
30 DATA 71,63,1,2, "AVERAGE"
40 DATA 95,63,1,2, "GOOD"
50 DATA 63,63,1,2, "VERY GOOD"
60 DATA 63,63,1,2, "EXCELLENT"
80 DATA 47,63,1,2, "NOT TRYING"
90 DATA 0,31,1,2, "ANOTHER GO? y/n"
95 DATA 55,125,5,2, "THANK YOU"
96 DATA 103,79,3,2, "for"
97 DATA 71,50,5,2, "PLAYING"
100 DATA 24,79,5,2, "STOP THE TAPE"
105 DATA 15,167,1,2, "REACTION TIMER"
110 DATA 13,164,2,2, "_____"
120 DATA 24,15,1,2, "PRESS ANY KEY"
125 RESTORE 100: FOR a=1 TO 4:PAUSE 25: INK 2: GO SUB 9000:NEXT a
180 PAUSE 0: BORDER 5: PAPER 6: INK 0: CLS
```

```
 190 RESTORE 105: INK 0: GO SUB 9000
 290 PRINT AT 20,5; "PRESS ANY KEY TO PLAY"
 300 IF INKEY$<>"" THEN GO TO 300
 302 IF INKEY$="" THEN GO TO 302
 310 CLS: RESTORE 105: INK 1: GO SUB 9000: INK 5: GO SUB 9000
 330 RESTORE 15: INK 2: GO SUB 9000
 350 INK 0: PLOT 102,114: DRAW 50,0: DRAW 0,-38: DRAW 50,0: DRAW 0,38
 380 LET z=INT (RND*26+65)
 390 LET x-111: LET y=111: LET h=4: LET w=4: LET a$=CHR$ z
 400 INK 6: LET c=USR 33393: FOR a=1 TO RND*200+50: NEXT a
 420 INK 0: LET c=USR 30000: LET a=PEEK 23728+256*PEEK 237729
 450 IF a>=140 OR a=0 THEN RESTORE 20
 460 IF a<140 THEN RESTORE 20
 470 IF a<120 THEN RESTORE 30
 480 IF a<100 THEN RESTORE 40
 490 IF a<85 THEN RESTORE 50
 500 IF a<70 THEN RESTORE 60
 510 GO SUB 9000
 7000 RESTORE 90:INK 1:GO SUB 9000
 7010 IF INKEY$<>"" THEN GO TO 7010
 7020 IF INKEY$="" THEN GO TO 7020
 7030 LET d$=INKEY$
 7040 IF d$ <> "y" THEN GO TO 8000
 7050 LET c=USR 30082: GO TO 350
 8000 RESTORE 95: POKE 30083,20: LET c=USR 30082: POKE 30083,18
 8030 FOR a=1 TO 3: GO SUB 9000: NEXT a: STOP
 9000 READ x, y, h, w, a$: LET c=USR 32393: RETURN
 9800 CLEAR 29999: LOAD ""CODE: GO TO 12
```

Листинг 3.3.2

```
АДРЕС       МАШ. КОД          АССЕМБЛЕР          КОММЕНТАРИЙ

30000       3E 7A             LD A,122           ;Код буквы "z"
30002       01 81 5C          LD BC,23681        ;Для хранения параметра "z"
                                                 ;выбирается неиспользуемая
                                                 ;в стандартном "Спектруме"
                                                 ;ячейка 23681.
30005       CD 7D 7E          CALL 32381         ;Параметр "z" перебрасыва-
                                                 ;ется в 23681 (см. проце-
                                                 ;дуры 2.7.1, 2.7.2.)
30008       FD CB 30 DE       SET 3,(IY+48)      ;Включение 3-го бита сис-
                                                 ;темной переменной FLAGS2
                                                 ;(23656) эквивалентно вклю-
                                                 ;чению режима CAPS LOCK.
30012       21 00 58          LD HL,22528        ;Первый байт области цвето-
                                                 ;вых атрибутов экрана.
L1
30015       7E                LD A,(HL)          ;Проверка этого байта.
30016       FE 36             CP 54              ;Переключение атрибутов
30018       20 02             JR NZ,L2           ;экранной области с ре-
30020       36 30             LD (HL),48         ;жима PAPER = 6, INK = 6
L2                                               ;на режим PAPER =6.
30022       23                INC HL             ;INK = 0.
30023       7C                LD A,H             ;Ранее нарисованная бук-
30024       FE 5B             CP 91              ;ва мгновенно "проявля-
30026       20 F3             JR NZ,L1           ;ется" на экране.
30028       01 00 00          LD BC,0            ;Инициализация счетчика,
L3
30031       C5                PUSH BC
30032       CD 2B 2D          CALL 11563         ;Помещение его на стек
                                                 ;калькулятора.
30035       3E 02             LD A,2             ;Открываем канал печати
30037       CD 01 16          CALL 5633          ;на экран.
30040       3E 16             LD A,22            ;Аналог ... AT ...
30042       D7                RST 16
```

```
30043      3E 06            LD A,6           ;Аналог ... AT 6...
30045      D7               RST 16
30046      3E 0E            LD A,14
30046      D7               RST 16           ;Аналог ... AT 6,14
30049      CD E3 2D         CALL 11747       ;Печать показаний счетчика.
30052      C1               POP BC
30053      03               INC BC           ;Увеличение счетчика.
30054      78               LD A,B           ;Проверка счетчика на об-
30055      B1               OR C             ;нуление.
30056      28 0F            JR Z,END         ;Выход, если он успел обну-
                                             ;литься (достиг 256).
30058      FD CB 01 6E      BIT 5,(IY+1)     ;Проверка не была ли нажата
                                             ;какая-либо клавиша.
30062      28 DF            JR Z,L3          ;Если нет, то возврат на
                                             ;повтор.
30064      3A 08 5C         LD A,(23560)     ;Код нажатой клавиши.
30067      21 81 5C         LD HL,23681      ;Код буквы, которая была
                                             ;показана на экране.
30070      BE               CP (HL)          ;Сравнили их между собой.
30071      20 06            JR NZ,L3         ;Если они не совпадают,
                                             ;значит нажата не та клави-
                                             ;ша. Следует повторить по-
                                             ;пытку.
END
30073      ED 43 B0 5C      LD(23728),BC     ;Результат испытания пере-
                                             ;дается из BC в адрес 23728
30077      FD CB 30 9E      RES 3,(IY+48)    ;Выключается ранее включен-
                                             ;ный режим CAPS LOCK.
30081      C9               RET              ;Выход из программы.
13082      06 12            LD B,18          ;Эта вспомогательная проце-
30084      CD 44 0E         CALL 3652        ;дура служит для очистки
30087      C9               RET              ;k нижних строк экрана, где
                                             ;k-число, установленное в
                                             ;адресе 30083 (исходно 18).
```

FORUM

Как эффективнее всего освоить машинные коды? Опыт показывает, что лучше всего освоение проходит, когда берешь какую-нибудь несложную готовую программу в машинных кодах и пытаешься понять, что и как там делается. При этом неизбежно возникает вопросы: "А почему так? Может можно все это сделать по-другому?" Начинаешь пробовать сделать по-другому и выясняется, что по-другому не выходит. Тогда становится понятно, почему автор организовал программу именно так, а не иначе. Но иногда встречаются ситуации, когда приходят действительно ценные идеи. Именно потому, что "ход мысли" начинающего еще не оформился в каких-то рамках, он не знает, что "можно", а что "нельзя". И он пытается выполнить задачу по-своему, а это зачастую приводит к находке новых, оригинальных решений.

Александр Балашов из г. Конаково Тверской обл. пишет нам о том, что он начал заниматься машинными кодами совсем недавно. Исследуя программы компрессии и декомпрессии, опубликованные в "ZX-РЕВЮ" N 1-2 за 1991 г., Александр обратил внимание, что под счетчик числа одинаковых байтов используется двухбайтный регистр BC. Учитывая организацию экранной области "Спектрума", он предположил, что выгоднее, видимо, использовать один байт, так как 256 байтов перекрывают 8-ю штрихами третью часть экрана и старший байт двухбайтного регистра BC используется лишь при практически пустом экране. Число, записанное в этом регистре занимает в скомпрессированном блоке 2 байта, один из которых почти всегда в нуле.

Этот факт побудил Александра разработать свою программу компрессии, в которой под счетчик числа одинаковых байтов он использовал только регистр B. Этот вариант позволяет получать длину скомпрессированного блока на 10-20% меньшую. Лишь при почти незаполненном экране, когда длина скомпрессированного блока меньше 300-400 байтов, длина его скомпрессированной картинки равна или уступает указанному варианту компрессора. Кроме того, его вариант не приводит к сбоям, которые могут иметь место при определенных ситуациях в варианте компрессора, упомянутом выше.

Александр приводит результаты сравнительных испытаний двух компрессоров для взятых наугад нескольких экранов игровых программ. Вот полученные длины скомпрессированных блоков (в байтах) соответственно для двух вариантов компрессоров:

"Death Star"	4619 и 5936
"Cauldron"	3228 и 2821
"Scooby Doo"	5958 и 5265
"XENO"	4985 и 4372
"Saboteur"	4027 и 3474

Вариант компрессора, который приводит Александр, заинтересовал нас и мы провели тестирование присланного им блока кодов, которое подтвердило основные моменты, изложенные выше. Но выяснилось, что полученный скомпрессированный блок кодов нельзя загружать для декомпрессии под любой адрес, а только под адрес, в котором он был скомпрессирован. Это серьезное ограничение для пользователей, которое может свести к нулю преимущество нового варианта компрессора. Хотя проблема в общем-то решается довольно просто. Те, кто регулярно читают "РЕВЮ", наверняка знают, как осуществить привязку к конкретным адресам загрузки.

Мы сделали такое усовершенствование программы, присланной Александром. Изменения коснулись программы декомпрессии (она стала длиннее на 7 байтов), из-за чего пришлось на 7 байтов отодвинуть программу компрессии в сторону младших адресов. В таком усовершенствованном виде мы предлагаем программу читателям (см. Лист. 1)

Листинг 1.

```
7F8B    210040          LD HL,#4000     ;В HL указатель адреса в экранной
                                        ;области.
7F8E    01001B          LD BC,#1B00     ;В BC счетчик байтов экрана.
7F91    110080          LD DE,#8000     ;В DE задан адрес, куда будет
                                        ;производиться компрессирование.
```

```
7F94    D5              PUSH DE         ;Он запоминается на стеке для то-
                                        ;го, чтобы потом можно было вычи-
                                        ;слить длину полученного скомпрес-
                                        ;сированного блока.
7F95    1B              DEC DE          ;Эта операция выполняется ввиду
7F96    13      L0      INC DE          ;того, что при возврате из опера-
                                        ;ции подсчета повторов и сравнений
                                        ;байтов в DE должен находиться
                                        ;адрес первого байта пустого места.
7F97    78      L1      LD A,B          ;Проверка BC на ноль, то есть опре-
7F90    B1              OR C            ;деление конца компрессирования.
7F99    2828            JR Z,#7FC3      ;Если ноль, то переход на END1
7F9B    7E              LD A,(HL)       ;в аккумулятор заносится байт из
7F9C    EDA0            LDI             ;экранной области и переносится по
                                        ;адресу, который указан в DE. DE и
                                        ;HL увеличиваются, а BC - уменьша-
                                        ;ется на единицу.
7F9E    BE              CP (HL)         ;Сравнение следующего байта с
                                        ;предыдущим.
7F9F    20F6            JR NZ,#7F97     ;Если не равны, то возврат на L1.
7FA1    0D              DEC C           ;Проверка BC на ноль (конец). B
7FA2    0C              INC C           ;данном случае нельзя использовать
7FA3    2004            JR NZ,#7FA9     ;аккумулятор, т.к. в нем находится
7FA5    05              DEC B           ;байт из экранной области, который
7FA6    04              INC B           ;сравнивается со следующим.
7FA7    281A            JR Z,#7FC3      ;
7FA9    12      L2      LD (DE),A       ;Загрузка первого повторяющегося
                                        ;байта в приемник.
7FAA    13              INC DE          ;Приращение DE для того, чтобы ор-
                                        ;ганизовать по этому адресу счет-
                                        ;чик повторов. Один повтор уже
                                        ;есть, поэтому сразу заносится еди-
                                        ;ница, а не ноль. Это оказывается
                                        ;удобнее при декомпрессировании.
7FAB    08              EX AF,AF'       ;Сохраняем в A экранный байт, вы-
                                        ;зываем альтерн. регистр для орга-
                                        ;низании в нем счетчика повторов.
7FAC    3E00            LD A,#00        ;Как было сказано выше, в счетчик
7FAE    3C      L3      INC A           ;сразу заносится единица. Здесь бу-
                                        ;дет организован счетчик повторов.
7FAF    28E5            JR Z,#7F96      ;Проверка на переполнение. Макси-
                                        ;мальное число повторов - 255.
7FB1    12              LD (DE),A       ;Запись в приемник числа повторов.
7FB2    23              INC HL          ;Переход к следующему байту экрана.
7FB3    0B              DEC BC          ;Уменьшение счетчика байтов экрана.
7FB4    0D              DEC C           ;Проверка BC
7FB5    0C              INC C           ;на ноль
7FB6    2004            JR NZ,#7FBC     ;без
7FB8    05              DEC B           ;задей-
7FB9    04              INC B           ;ствования
7FBA    2608            JR Z,#7FC4      ;аккумулятора.
7FBC    08      L4      EX AF,AF'       ;Обратный обмен регистров для срав-
                                        ;нения текущего байта с предыдущим.
7FBD    BE              CP (HL)         ;Сравнение.
7FBE    20D6            JR NZ,#7F96     ;Если не равны, то возврат на L1
7FC0    08              EX AF,AF'       ;Если равны, то восстанавливаем
7FC1    18EB            JR #7FAE        ;счетчик повторов и переходим на
                                        ;L3 для наращивания счетчика.
7FC3    1B      END1    DEC DE          ;Эта операция нужна для того, чтобы
                                        ;при выходе из программы компрессии
                                        ;в DE содержался адрес последнего
                                        ;байта скомпрессированного блока.
7FC4    1A      END2    LD A,(DE)       ;Берем последний байт из DE,
7FC5    2F              CPL             ;инвертируем его
7FC6    13              INC DE          ;и записываем в следующий адрес ;
```

```
7FC7        12              LD (DE),A       ;это сделано для упрощения декодиру-
                                            ;ющей процедуры, т.к. в ней можно
                                            ;будет избежать лишней проверки
                                            ;счетчика на ноль.
7FC8        EB              EX DE,HL        ;Обмен DE и HL для сохранения DE.
7FC9        D1              POP DE          ;Возвращаем со стека начальное
                                            ;значение DE.
7FCA        AF              XOR A           ;Сброс флага переноса, необходимо
                                            ;для выполнения следующей команды.
7FCB        ED52            SBC HL,DE       ;Отняв от конечной величины DE (ко-
                                            ;торая теперь в HL) начальную, по-
                                            ;лучим число байт скомпрессированн-
                                            ;ной области, но без учета послед-
                                            ;него контрольного байта, который
                                            ;является инверсной копией предпо-
                                            ;следнего.
7FCD        22E57F          LD (#7FE5),HL   ;Подставляем полученное значение в
                                            ;программу декомпрессии.
7FD0        012600          LD BC,#0026     ;Смещение, учитывающее длину
                                            ;декомпрессирующей процедуры.
7FD3        AF              XOR A           ;Сброс флагов.
7FD4        ED4A            ACD HL,BC       ;Получение в HL суммарной длины
7FD6        E5              PUSH HL         ;блока кодов с учетом декодирующей
7FD7        C1              POP BC          ;процедуры. Перезапись полученного
7FD8        03              INC BC          ;значения в регистр BC и увеличение
7FD9        C9              RET             ;его на 1 для учета последнего
                                            ;(контрольного) байта.
```

Последняя часть программы (с 7FD0H) позволяет получить выходной параметр в удобном для последующего использования виде. Для этого он был предусмотрительно помещен в регистр BC. Подав команду PRINT USR 32651 (7F8BH) результат работы компрессирующей процедуры (из регистра BC) будет выведен на экран: длина скомпрессированного блока (с учетом контрольного байта) плюс длина декомпрессора. А при подаче команды

```
LET L=USR 32651
```

- результат будет помещен в переменную L. На магнитную ленту должен быть выгружен блок кодов с адреса 32730 (7FDAH), длиной L.

```
7FDA        CD7C00          CALL #007C      ;Так выполняется привязка к
7FDB        3B              DEC SP          ;адресу загрузки. В результате
7FDE        3B              DEC SP          ;в HL получается адрес начала
7FDF        E1              POP HL          ;скомпрессированного блока кодов.
7FE0        012300          LD BC,#0023     ;Оно образуется в результате
7FE3        09              ADD HL,BC       ;увеличения адреса загрузки на
                                            ;величину, занимаемую программой
                                            ;декомпрессии.
7FE4        010000          LD BC,#0000     ;Сейчас здесь ноль, но при работе
                                            ;программы компрессии сюда будет
                                            ;подставлена длина скомпрессиро-
                                            ;ванного блока кодов.
7FE7        110040          LD DE,#4000     ;Адрес приемника - экрана.
7FEA        78          L0  LD A,B          ;Проверка BC
7FEB        B1              OR C            ;на ноль.
7FEC        C8              RET Z           ;Выход, если ноль.
7FED        7E              LD A,(HL)       ;В аккумулятор заносится байт из
                                            ;скомпрессированной области.
7FEE        EDA0            LDI             ;Перезапись байта в экран и пере-
                                            ;ход к следующему байту экрана и
                                            ;приемника.
7FF0        BE              CP (HL)         ;Сравнение его с предыдущим.
7FF1        20F7            JR NZ,#7FEA     ;Если не равен, то переход на L0 и
                                            ;повтор операции перезаписи и срав-
                                            ;нения.
```

```
7FF3        23                  INC HL              ;Переход к следующему байту, в котором
                                                    ;находится кол-во повторов.
7FF4        C5                  PUSH BC             ;Счетчик запоминается на стеке.
7FF5        46                  LD B,(HL)           ;Организуем в B счетчик повторов,
                                                    ;число которых берется из HL.
7FF6        12          L1      LD (DE),A           ;Запись байта в экран.
7FF7        13                  INC DE              ;Переход к следующему байту экрана.
7FF8        10FC                DJNZ #7FF6          ;Уменьшение счетчика, если не 0 -
                                                    ;то продолжение перезаписи байта
                                                    ;из аккумулятора в экран.
7FFA        C1                  POP BC              ;Снимаем BC со стека без проверки
                                                    ;на ноль, т.к. переход в цикл
                                                    ;загрузки одинаковых байтов может
                                                    ;быть только из положительных зна-
                                                    ;чений BC (нулевой байт инвертиро-
                                                    ;ван по отношению к последнему).
7FFB        23                  INC HL              ;Переход к следующему байту ском-
                                                    ;прессированной области.
7FFC        0B                  DEC BC              ;Теперь BC указывает на байт в HL,
                                                    ;в котором находится число повто-
                                                    ;ров и тоже не может быть в нуле.
7FFD        0B                  DEC BC              ;Уменьшение счетчика.
7FFE        18EA                JR #7FEA            ;Переход на проверку BC и пере-
                                                    ;запись байтов.
8000        ....                                    ;Скомпрессированный блок кодов.
```

Для того, чтобы можно было компрессировать экраны, записанные без заголовка, Александр рекомендует дополнить программу компрессора процедурой загрузки блока кодов без заголовка, которая использует процедуру 1386 (0556H) ПЗУ:

```
7F7E        DD210040            LD IX,#4000
7F6E        11001B              LD DE,#1B00
7F85        3EFF                LD A,#FF
7F87        37                  SCF
7F88        CD5605              CALL #0556
```

Эффективность той или иной программы во многом зависит от того, насколько продумана Бейсиковая ее часть. Мы приводим вариант программы "Компрессор", которая в основе содержит программу, присланную Александром. Она русифицирована способом, которым мы постоянно пользуемся (см. "ZX-РЕВЮ"-92 N 1,2, стр. 30).

```
1 GO TO 100
2 CLEAR 30000: LOAD "chr" CODE 64600
3 FOR N=32636 TO 32767: READ M: POKE N,M: NEXT N
4 RUN
5 SAVE "COMPR" LINE 2: STOP
6 POKE 23606,88: POKE 23607,251: RETURN
9 POKE 23606,0: POKE 23607,60: RETURN
100 BORDER 7: PAPER 7: INK 0: CLS
110 GO SUB 8
200 PRINT AT 7,1;"<1> - ЗАГРУЗКА ЭКРАННОГО ФАЙЛА" ' TAB 7; "С ЗАГОЛОВКОМ"
210 PRINT AT 10,1;"<2> - ЗАГРУЗКА БЛОКА КОДОВ"'TAB 7;"БЕЗ ЗАГОЛОВКА" TAB 7;"С КОНТРОЛЕМ
    СЧИТЫВАНИЯ"
220 PRINT AT 14,1;"<3> - ЗАГРУЗКА БЛОКА КОДОВ"'TAB 7;"БЕЗ ЗАГОЛОВКА" TAB 7;"БЕЗ КОНТРОЛЯ
    СЧИТЫВАНИЯ"
300 PAUSE 0: IF INKEY$<>"1" AND INKEY$<>"2" AND INKEY$<>"3" THEN GO TO 300
310 LET I$=INKEY$
400 CLS : PRINT #0; TAB 11; FLASH 1;" ЗАГРУЗКА ": GO SUB 9
410 IF I$="1" THEN LOAD ""CODE 16384,6912: LET L=USR 32651
420 IF I$="2" THEN POKE 32649,2: POKE 32650,8: LET L=USR 32638
430 IF I$="3" THEN POKE 32649,66: POKE 32650, 5: LET L=USR 32636
500 GO SUB 8: INPUT ; : PRINT #0; PAPER 0; INK 7; BRIGHT 1, FLASH 1;" ДЛИНА="; L.
510 BEEP .1,20: PAUSE 400
```

```
600 PRINT AT 21,0;"ИМЯ ФАЙЛА ДЛЯ ЗАПИСИ: ": GO SUB 9: INPUT F$
610 SAVE F$ CODE 32730,L
700 RUN
900 DATA 221,33,0,64,17,0,27,62,255,55,205,86,5
910 DATA 33,0,64,1,0,27,17,0,128,213,27,19,120,177,40,40,126
920 DATA 237,160,190,32,248,13,12,32,4,5,4,40,26,18,19,8,62,0
930 DATA 60,40,229,16,35,11,13,12,32,4,5,4,40,8,8,190,32,214,8
940 DATA 24,235,27,26,47,19,18,235,209,175,237,32,34,229,127,1,38,0,175,237,74,229,193,3,201
950 DATA 205,124,0,59,59,225,1,35,0,9,1,0,0,17,0,64,120,177,200
960 DATA 126,237,160,190,32,247,36,197,70,18,19,16,252,193,35,11,11,24,234
```

Со строки 200 выводится меню программы. Возможны три варианта загрузки. Первый - обычная загрузка экрана, записанного с заголовком. Второй и третий - загрузка экрана, записанного без заголовка. Эти два режима могут оказаться довольно удобными, так как во многих играх встречаются блоки кодов - экраны, записанные без заголовка. Режимы загрузки экрана без заголовка отличаются между собой тем, что в одном случае для загрузки вызывается подпрограмма 2050 из ПЗУ, которая в случае ошибки выводит сообщение: "Tape loading error", а в другом случае используется подпрограмма 1366 ПЗУ, при этом работа БЕЙСИКа не прекращается независимо от правильности считывания.

Сразу же после окончания загрузки внизу экрана Вы увидите результат работы компрессирующей программы: длину скомпрессированного блока кодов. После нажатия любой клавши программа предложит Вам ввести имя для записи полученного результата вместе с Программой декомпрессии в виде единого файла. После записи можете загружать скомпрессированной блок в произвольный адрес ADR. Вывод на экран - RANDOMIZE USR ADR.

Другой вариант компрессии.

Продолжая разговор о компрессорах экрана, приведем еще одну программу.

Очень часто при разработке каких-либо (например обучающих) программ, приходится решать вопрос размещения в памяти компьютера большого количества графической информации, рисунков, схем и т. д. Причем во многих случаях графика занимает не весь экран, а скажем, только средний сегмент экрана. При этом верхний и нижний сегменты используются для других целей. Поэтому за счет компрессии не полного экрана, а лишь одного его сегмента, можно значительно сократить длину скомпрессированного блока кодов. Кроме того, при декомпрессии можно вызывать сегменты из памяти независимо один от другого, не нарушая соседние, правда в этом случае придется компрессирование проводить за два этапа, отдельно для дисплейной части и атрибутов, так как между вами появляется разрыв.

Декомпрессирущий блок является единственным для всех скомпрессированных экранов. При таком подходе в памяти можно получать десятки (а если небольшие, то может быть даже сотни) экранов. Перед запуском декомпрессирующей программы надо номер нужного экрана занести в системную ячейку декомпрессора при помощи POKE, а затем вызывать декодирующую процедуру.

В основе приводимой программы содержится известный многим пользователям "Спектрума" блок кодов "compress" CODE 28000,650. Но Бейсиковая часть сделана практически заново, что позволило пользоваться программой с большим удобством и удовольствием. Кроме того, мы приводим два варианта программы: для магнитофона и для дисковода.

Этот вариант компрессора является более изощренным, чем приведенные до сих пор программы. (Достаточно обратить внимание на длину его кодового блока). Однако эта длина сама по себе ни о чем не говорит. Ведь важна длина конечного продукта - скомпрессированного экрана плюс программы декомпрессии. Этот вариант компрессора позволяет получить более высокую степень компрессии, чем те, которые мы приводили до сих пор. Мы провели сравнительные испытания этого компрессора для тех же экранов.

"Death Star"	3185 байт
"Cauldron"	2128 байт

"Scooby Doo" 3946 байт
"XENO" 3439 байт
"Saboteur" 2545 байт

Кроме более высокой степени компрессии, этот вариант компрессора позволяет получить следующие режимы, недоступные другим программам:

Возможно компрессирование как всего экрана, так и произвольно взятых одного любого или двух соседних сегментов экрана. Программа анализирует расположение графической информации на экране и выбирает, в зависимости от этого, наиболее выгодный способ компрессирования. При этом во время декомпрессии Вы можете увидеть, что некоторые экраны разворачиваются сверху вниз, аналогично тому, как это происходит при загрузке. А другие экраны - разворачиваются слева направо, (что несомненно усиливает зрительный эффект). Правда работает эта программа медленнее, чем другие варианты компрессоров. Компрессия полного экрана занимает порядка 1-3 секунд (однако это время не имеет значения для конечного продукта).

Так как кодовый блок программы (мы его полностью приводим ниже) расположен с адреса 28000, при использовании программы с дисководом Вам может не хватить свободного места. Для этого в программе используются приемы по сохранению памяти. Мы уже приводили такие материалы в "РЕВЮ". Это, в частности, замена числа 0 на NOT PI, числа 1 - на SGN PI, использование вместо числа выражения: VAL "число". Без таких преобразований оставлены только строки программы, реализующие работу меню, чтобы не замедлять время выполнения этих операций. Все эти приемы могут оказаться излишними для магнитофонного варианта, но они обязательны для дискового. Текст БЕЙСИК-программы вначале приведен для магнитофонного варианта. Затем будут даны дополнительные строки, которые позволят Вам получить дисковый вариант программы.

Вот Бейсиковая часть программы. Она традиционно русифицирована по известной Вам схеме.

```
1 GO TO VAL "100"
2 BORDER VAL "7": PAPER VAL "7": INK NOT PI: CLEAR VAL "27995": LOAD "chr"CODE 64600
3 LOAD "compress"CODE 28000,650
4 GO TO NOT PI
5 SAVE "COMPRESS"LINE 2: SAVE "compress"CODE 28000, 650: STOP
8 POKE VAL "23606", VAL "86": POKE VAL "23607", VAL "251":RETURN
9 POKE VAL "23606",NOT PI: POKE VAL "23607",CODE "<": RETURN
30 BORDER 1: PAPER 0: INK 7: CLS : GO SUB 8: GO SUB 40
31 FOR M=1 TO NN: READ M$: PRINT INK 6;AT (Y0+(M-1)*DY),X0;M$: NEXT M: PRINT INK 4; AT Y1,
      3; "SPACE,DOWN,UP,ENTER,BREAK"
32 IF MM<1 THEN LET MM=NN
33 IF MM>NN THEN LET MM=1
34 PRINT AT (Y0+(MM-1)*DY),X0-DX: PAPER 7; INK 2; BRIGHT 1; OVER 1;S$: BEEP .03,2*MM+10
35 PAUSE 0: LET I=(INKEY$=" " OR 1 INKEY$= "6" OR CODE INKEY$=10)+(INKEY$="7" OR CODE
      INKEY$=11)*2+(INKEY$ = "0" OR CODE INKEY$=12 OR CODE INKEY$=13)*3: GO TO (35+I)
37 PRINT INK 6;AT (Y0+(MM-1)*DY),X0-DX; OVER 1;S$: LET MM=MM-2*I+3: GO TO 32
38 FOR M=1 TO NN: PRINT PAPER 8; INK 8; BRIGHT 6; FLASH (M=MM); OVER (M=MM);AT (Y0+(M-
      1)*DY),X0-DX:S$: NEXT M: BEEP .1,36: PAUSE 10: RETURN
40  PRINT AT 2,9;"ЧИСЛО ЭКРАНОВ: ";SCR
42 PRINT AT 3,3;"КОМПРЕССИЯ СЕГМЕНТА : ";BEG;: IF END<>BEG THEN PRINT AT 3,21;"ОВ";AT
      3,26:"-";END
44 PRINT AT 4,6: "СВОБОДНОЙ ПАМЯТИ : ";65536-LAST
49 RETURN
50 BORDER VAL "7": PAPER VAL "7": INK NOT PI: CLS
52 PRINT AT VAL "21",NOT PI;"ИМЯ ФАЙЛА:": GO SUB VAL "9": INPUT F$: CLS
59 RETURN
100 LET BEG=SGN PI: LET END=VAL"3": LET SCR=NOT PI: LET FIRST=VAL "28350": LET LAST=VAL
      "28650": LET S=NOT PI
190 LET MM=SGN PI
200 DATA "ЗАГРУЗКА ЭКРАНА","РЕЖИМ КОМПРЕССИИ","ПРОСМОТР ЭКРАНА","УДАЛЕНИЕ ЭКРАНА","ЗАПИСЬ
      ФАЙЛА"
201 RESTORE VAL "200": LET NN=VAL "5": LET X0=VAL "8": LET V0=VAL "9": LET DY=SGN PI; LET
      DX=VAL "3": LET S$="                    ": LET Y1=VAL "19": GO SUB VAL "30"
```

```
210 IF SCR=NOT PI AND MM>=VAL "3" THEN PRINT #NOT PI; FLASH SGN PI;"          НЕТ ЭКРАНОВ!
      ": BEEP SGN PI, NOT PI: PAUSE VAL "100": GO TO VAL "190"
220 GO TO VAL "1E3"*MM
300 DATA " ВЕСЬ ЭКРАН","СЕГМЕНТЫ 1-2","СЕГМЕНТЫ 2-3"," СЕГМЕНТ 1"," СЕГМЕНТ  2"," СЕГМЕНТ
      3"
301 RESTORE VAL "300": LET NN=VAL "6": LET X0=VAL "10": LET Y0=VAL "8": LET DY=SGN PI: LET
      DX=VAL "3": LET S$="                    ": LET Y1=VAL "19": GO SUB VAL "30": RETURN
1000 GO SUB VAL "50"
1010 LOAD F$ CODE 16384,6912
1030 POKE VAL "28005",INT (LAST/VAL "256"): POKE VAL "28004",LAST-VAL "256"*PEEK VAL "28005"
1040 POKE VAL "28006",BEG-SGN PI: POKE VAL "28007",END-(BEG-SGN PI)
1050 RANDOMIZE USR VAL "28010"
1060 LET LEN=PEEK VAL "28008"+VAL "256"*PEEK VAL "26009": LET LAST=LAST+LEN+SGN PI: LET
      SCR=SCR+SGN PI: LET S=SCR-SGN PI
1070 GO SUB VAL "8": PRINT #NOT PI; PAPER NOT PI; INK VAL "7"; BRIGHT SGN PI; FLASH SGN PI;"
      ДЛИНА=";LEN;" "
1080 PAUSE VAL "400"
1090 GO TO VAL "200"
2000 LET MM SGN PI: GO SUB VAL "300"
2010 RESTORE VAL "2000": FOR N=SGN PI TO MM: READ BEG: READ END:NEXT N
2020 GO TO VAL "190"
2100 DATA SGN PI,VAL "3",SGN PI, VAL "2",VAL "2", VAL "3",SGN PI,SGN PI,VAL "2",VAL "2",VAL
      "3",VAL "3"
3000 LET S=S*(S<SCR)+(S<=SCR)
3010 INPUT ("НОМЕР ЭКРАНА (1-";SCR;"): ";S;" : "); LINE F$: IF F$<>"" THEN LET S=VAL F$
3020 IF S<SGN PI OR S>SCR THEN GO TO VAL "3010"
3030 BORDER VAL "7": PAPER VAL "7": INK NOT PI: CLS
3040 POKE VAL "28352",S: RANDOMIZE USR VAL "28350": PAUSE NOT PI
3050 GO TO VAL "200"
4000 IF SCR SGN PI THEN LET LAST=VAL "28650": GO TO VAL "4100"
4010 LET A=NOT PI: LET C=A: LET LEN=VAL "28649": FOR N=SGN PI TO SCR: LET LEN=LEN+A+C*VAL
      "256"+SGN PI: LET A=PEEK LEN: LET C=PEEK (LEN+SGN PI): NEXT N: LET LAST=LEN
4100 LET SCR=SCR-SGN PI: LET S=SCR-SGN PI
4110 GO TO VAL "200"
5000 GO SUB VAL "50"
5010 SAVE F$CODE 28350,LAST-FIRST
5030 VERIFY F$ CODE
5050 GO SUB VAL   "8" : PRINT #NOT PI; INVERSE SGN PI; "   ЗАПИСЬ  И ВЕРИФИКАЦИЯ - O.K.    "
5060 BEEP VAL ".1",VAL "26": BEEP VAL ".1",VAL "20": PAUSE VAL "100"
5070 GO TO VAL "190"
```

Программы, подобные этой, Вы уже встречали на страницах "РЕВЮ", поэтому не будем подробно останавливаться на каждой ее части. Перечислим только основные фрагменты:

Строка 2 - автостарт, загрузка кодовых блоков.

Строка 5 - самозапись программы на магнитную ленту.

Строки 8 и 9 - переключатели шрифтов, соответственно русского и латинского.

Строка 30 - подпрограмма работы меню. Она была подробно рассмотрена в "ZX-РЕВЮ" N 9-10 за 1992 г., стр. 197.

Строка 40 - вывод основных режимов и состояния компрессора. На экран выводится следующая информация: число скомпрессированных экранов, режим компрессии - какие сегменты заданы для компрессирования, количество оставшейся свободной памяти компьютера.

Строка 50 - ввод имени файла для загрузки и записи. Если при выполнении загрузки в ответ на запрос имени нажать ENTER, ничего не вводя, то загрузится первый встретившийся кодовый файл.

Строка 100 - инициализация.

Строка 200 - главное меню программы, задающее режимы работы компрессора.

Строка 300 - второе меню, задающее сегменты для компрессирования.

Строка 1000 - загрузка очередного экрана.

Строка 2000 - задание сегментов для режима компрессии (вызов второго меню).

Строка 3000 - просмотр скомпрессированных экранов. Здесь следует отметить организацию ввода номера просматриваемого экрана. При запросе предлагается вариант по соглашению. Если предложенное число Вас не устраивает, введите свое значение, если устраивает - просто нажмите ENTER. Предложение варианта по соглашению (строка 3000) организовано так, что после загрузки очередного экрана к просмотру предлагается этот последний экран. После его просмотра по соглашению в следующий раз будет предложен первый экран, после его просмотра - второй и так далее до последнего, затем опять первый и так далее. Так можно просмотреть все экраны, ничего не вводя, а можно в любой момент вмешаться в этот порядок и ввести свое значение.

Строка 4000 - удаление экрана. Надо ли здесь организовать запрос для подтверждения Вашего решения? Организуйте его сами, если хотите. Удаляется всегда только последний экран. Это может создавать некотрое неудобство. Хорошо, если бы можно было удалять любой из скомпрессированных экранов, но это потребует вмешательство в блок кодов. В общем-то ничего принципиально сложного здесь нет. Надо только рассчитать место удаляемого экрана, определить его длину и организовать при помощи машиннокодовой команды LDIR переброску следующих экранов на место удаляемого. Все, что необходимо для такого рассчета, находится в строках с 4000. Может быть кто-нибудь из читателей и сделает такое усовершенствование.

Строка 5000 - запись всех скомпрессированных экранов вместе с программой декомпрессии в виде единого файла. Если при задании имени файла нажать ENTER, ничего не вводя, то программа вернется в режим основного меню (см. строку 5000).

При работе с БЕТА-диском, должны быть изменены строки, связанные с загрузкой и выгрузкой. Мы приводим ниже вариант строк для адаптации программы под дисковод. Вы можете набрать и хранить их отдельно, а при адаптации подгрузить к основной БЕЙСИК-программе при помощи MERGE.

```
2 BORDER VAL "7": PAPER VAL "7": INK BIN : CLEAR VAL "27999": RANDOMIZE USR VAL "15619": REM
      : LOAD "chr"CODE 64600
3 RANDOMIZE USR VAL "15619": REM : LOAD "compress"CODE 28000,650
5 GO SUB VAL "90": STOP
6 RANDOMIZE USR VAL "15616"
7 STOP
90 RANDOMIZE USR VAL "15619": REM : ERASE "COMPRESS"
92 RANDOMIZE USR VAL "15619": REM : SAVE "COMPRESS" LINE 2
99 RETURN
1000 GO SUB VAL "50": IF F$="" THEN GO TO VAL "200"
1010 LET ERR USR VAL "15619": REM : LOAD F$ CODE 16364,6912
1020 IF ERR<>NOT PI THEN GO TO VAL "1500"
1500 GO SUB VAL "8": PRINT AT VAL "10",VAL "9"; FLASH SGN PI;" ОШИБКА ";ERR:" "' FLASH NOT
      PI'" ВЫВОДИТЬ КАТАЛОГ ДИСКА? (Y/N)"
1510 BEEP SGN PI, NOT PI: PAUSE NOT PI: IF INKEY$<>"y" AND INKEY$<>"Y" THEN GO TO VAL "200"
1520 GO SUB VAL "9": RANDOMIZE USR VAL "15619": REM : CAT
1530 PAUSE VAL "400"
1540 GO TO VAL "200"
5010 LET ERR=USR VAL "15619": REM : SAVE F$ CODE 28350,LAST-FIRST
5020 IF ERR<>NOT PI THEN GO TO VAL "1500"
5030 LET ERR=USR VAL "15619": REM : VERIFY F$ CODE
5040 IF ERR<>NOT PI THEN GO TO VAL "1500"
```

Строка 5 - как и раньше, самозапись программы (только Бейсиковой части) на диск. Она выполняется при помощи подпрограммы со строки 90, так как в одной пятой БЕЙСИК-строке не может выполняться более одного оператора TRDOS.

Строка 6 - переход в TR-DOS для выполнения каких-либо действий. После возврата оттуда по команде RETURN, попадаем на STOP в строке 7.

Строки с 1000 - загрузка экрана с диска. Здесь в случае ошибки при загрузке (например отсутствие файла на диске), происходит переход на строку 1500. Выведенный на экран код ошибки позволит Вам сориентироваться, почему произошла ошибка (коды ошибок см. в руководстве по TR DOS).

Строка 1500 предлагает вывести каталог диска, для того, чтобы Вы могли еще раз уточнить, есть ли нужный файл на этом диске или нет. На строку 1500 программа может перейти и из строк с 5000.

Строка 5000 - режим записи файла на диск, а также выполнение его верификации после записи.

Для тех читателей, которым не знаком кодовый блок программы "COMPRESS", мы приводим его полностью. (Тем, у кого он есть, можно его не набирать, а взять из имеющейся программы в готовом виде.) Для набора его Вы можете воспользоваться рекомендациями и программой для шестнадцатиричного ввода см. "ZX-РЕВЮ"-91, N 3, с. 59.

```
6D60: 30 01 00 00 EA 6F 00 01:58
6D68: 8D 03 FD 21 60 6D FD 36:83
6D70: 03 01 CD 85 6D E5 FD 36:B8
6D78: 03 00 CD 85 6D D1 ED 52:B7
6D80: D8 FD 36 03 01 FD 6E 04:6B
6D68: KD 66 05 11 04 00 19 1E:A9
6D90: 00 FD 7E 06 87 87 87 C6:D9
6D98: 40 57 FD 77 08 1A 2F FD:5E
6DA0: 77 00 FD 36 01 01 FD 36:EC
6DA8: 02 00 FD 7E 03 18 74 FD:1E
6DB0: 4E 00 1A FD 77 00 B9 28:DA
6DB8: 0E FD CB 02 46 28 08 FD:70
6DC0: CB 02 86 FD 36 01 01 A7:5C
6DC8: 20 16 FD CB 02 46 20 10:AB
6DD0: FD 7E 01 3D 23 04 23 71:B6
6DD8: 18 F9 3C 23 0E 00 18 13:EE
6DE0: B9 20 30 FD CB 02 46 20:86
6DE8: 15 FD 7E 01 3C FD 77 01:97
6DF0: FE 04 D8 36 00 23 77 23:2A
6DF8: 71 FD 36 02 01 C9 2B 34:34
6E00: 08 23 FD 36 01 01 08 C0:96
6E08: 7E 2F FD 77 00 FD 36 02:CC
6E10: 00 C9 71 23 FD 35 01 20:2E
6E18: F9 77 FD 36 01 01 FD 36:5E
6E20: 02 00 C9 77 06 01 FD CB:9F
6E28: 03 46 20 02 06 20 C5 FD:E9
6E30: 46 07 C5 06 08 C5 06 08:91
6E38: C5 06 01 FD CB 03 46 28:AB
6E40: 02 06 20 D5 CD AF 6D 13:A7
6E48: 10 FA D1 C1 14 10 E9 7A:D9
6E50: D6 08 57 7B C6 20 5F C1:74
6E58: 10 DB 7A C6 00 57 C1 10:21
6E60: D1 FD 56 08 1C C1 10 C6:AD
6E68: 11 00 58 0E 01 FD 46 07:98
6E70: FD 7E 06 82 57 C5 CD AF:79
6E78: 6D C1 0B 13 78 B1 20 F5:70
6E60: FD 5E 04 FD 56 05 A7 ED:39
6E83: 52 EB 73 23 72 FD 7E 06:BC
6E90: 23 77 23 FD 7E 07 77 EB:9F
6E98: 22 68 6D C9 00 00 00 00:C6
6EA0: 00 00 00 00 00 00 00 00:0E
6EA8: 00 00 00 00 00 00 00 00:16
6EB0: 00 00 00 00 00 00 00 00:1E
6EB8: 00 00 00 00 00 00 18 04:42
6EC0: 01 FF FF 0C CD 7C 00 38:BD
6EC8: 3B E3 33 33 A7 11 07 00:79
6ED0: ED 52 E5 FD E1 11 2A 01:7C
6ED8: 19 FD 7E 00 3D 28 06 5E:A3
6EE0: 23 56 19 18 F7 11 00 40:40
6EE8: 23 23 FD 7E 01 3C 20 04:78
6EF0: 7E FD 77 01 23 FD 7E 02:F1
6EF8: 3C 20 04 7E FD 77 02 23:DD
6F00: 7E FD 77 03 FD 7E 01 FE:DE
6F08: 03 38 04 FD 36 01 00 FD:E7
6F10: 7E 02 FE 04 38 04 FD 36:70
6F18: 02 01 FD 86 01 FE 04 38:48
6F20: 04 FD 36 02 01 FD 7E 02:46
6F28: A7 20 04 FD 36 02 01 23:BB
6F30: 1E 00 FD 7E 01 87 87 87:CE
6F38: C6 40 57 4F 06 01 FD CB:22
6F40: 03 46 20 02 06 20 C5 FD:02
6F48: 46 02 C5 06 08 C5 06 08:A5
6F50: C5 06 01 FD CB 03 46 28:C4
6F58: 02 06 20 D5 C5 18 43 C1:A5
6F60: 13 10 F9 D1 C1 14 10 E8:89
6F68: 7A D6 08 57 7B C6 20 5F:46
6F70: C1 10 DA 7A C6 08 57 C1:EA
6F78: 10 D0 51 1C C1 10 C7 1E:EA
6F80: 00 3E 58 FD 86 01 57 FD:5D
6F88: 46 02 4B FD CB 03 DE C5:F8
6F90: 18 10 C1 0B 33 78 B1 20:4F
6F98: F6 FD 36 01 FF FD 36 02:65
6FA0: FF C9 FD CB 03 56 28 0D:2D
6FA8: D9 05 79 D9 28 03 12 18:9C
6FB0: 18 FD CB 03 96 7E 23 A7:E0
6FB8: 20 0E 7E D9 47 D9 23 7E:6D
6FC0: D9 4F D9 23 FD CB 03 D6:F4
6FC8: 12 FD CB 03 5E 20 C3 18:6D
6FD0: 8E 08 10 11 12 12 14 00:2E
6FD8: 03 00 01 02 01 00 03 00:51
6FE0: 0F 19 0E 00 04 00 08 18:A9
6FE8: 22 22 00 00 00 00 00 00:9B
```

Михайленко В.С., Турович А.К.

Профессиональный подход

Методы Билла Гильберта.

По-видимому, среди программ, находящихся у пользователей, имеющих Синклер-совместимые компьютеры, ничто так не распространено, как программы, взломанные Биллом Гильбертом.

В этой статье Вашему вниманию будет предложена информация о специфике некоторых приемов оформления программ, применяемых этим хаккером. Несмотря на внешнюю схожесть эффектов, получающихся в ходе работы программ, Гильберт не всегда действует по одному шаблону. У него их достаточно много, хотя иногда они очень похожи и, разобравшись с одним из них, Вам не составит труда понять принципы действия остальных.

Как Вы уже вероятно знаете, Билл Гильберт всегда оставляет текстовые сообщения на взломанных им программах. Несмотря на то, что в последние годы он явно поскромнел, и большинство его сообщений можно перевести с английского, как "Дисковая версия Билла Гильберта" (по-английски: Disked by Bill Gilbert), в отличие от ранее использовавшегося автографа "Взломано Биллом Гильбертом" (по-английски: Cracked by Bill Gilbert), его методы остались теми же и основная цель, которую он преследовал вставкой подобных сообщений не изменилась. При изучении его "творчества" необходимо осознавать неэтичность действий Гильберта и относиться к этому так же, как и к тому, что некто на наших глазах роется в чужих карманах (хоть и делает это высокопрофессионально).

В то же время, опыт хаккера может быть использован при оформлении программ, совершенствовании их дизайна, что делает работу более простой, удобной и увлекательной.

При использовании приводимого нами материала необходимо учитывать, что большинство рассматриваемых программ написаны в машинных кодах. Это еще раз говорит о том, что высокого уровня профессионализма можно добиться только изучив программирование на языке Ассемблера. Несмотря на то, что большинство рассматриваемых процедур снабжены подробным комментарием, мы настоятельно рекомендуем Вам перед прочтением данного материала поближе познакомиться с программированием в машинных кодах. Одной из лучших книг, которые нам приходилось встречать по данное тематике, является методическая разработка "ИНФОРКОМа" "Практикум по программированию в машинных кодах".

Мы рекомендуем Вам для начала просто ознакомиться с текстом статьи, получить представление об основных приемах и методах, а потом уже детально изучать проблему по разделам. Мы старались подготовить информацию таким образом, чтобы она не требовала специального запоминания, а усваивалась постепенно по ходу внимательного прочтения.

Необходимо также учитывать, что данный материал тесно связан с той информацией, которая давалась в предыдущих статьях, особенно это касается методик взлома рассмотренных систем защиты. Впрочем, если Вы и не читали предыдущих статей, то не отчаивайтесь - нижеследующий текст отличается определенной автономией от предыдущего материала и не требует особой подготовки, интересен как для профессионалов, так и для начинающих. Желаем Вам удачи при изучении!

Современные разработки Билла Гильберта.

Билл Гильберт работает над взломом программ уже много лет. За это время ему удалось накопить достаточно большой опыт в этих вопросах. Постоянно совершенствуя свою технику программирования он достиг того уровня, который мы видим в последних взломанных им программах.

Наиболее любопытными среди них являются Iron Man и Sim City. В этих програмах использован оригинальный вывод текстовых сообщений о взломе, который свидетельствует о высоком профессиональном уровне программирования хаккера. В обеих программах используется совмещение программы на БЕЙСИКе с программой в машинных кодах, обе

эти программы схожи также и по структуре загрузки и по методике взлома, применненной хаккером. Это позволяет рассматривать обе программы как две оригинальные ветви одного типа взлома. В них можно найти очень много общего, в том числе: общие процедуры на Ассемблере, схожесть расположения процедур в адресном пространстве и многое другое. Но, в тоже время, существуют довольно разительные отличия, которые предопределяют необходимость индивидуального рассмотрения каждой программы. В первую очередь это касается методики вывода текстовых сообщений на экран и, во-вторых, методики создания оригинальных моделей шрифта.

Рассмотрим более подробно эти методы на обеих программах.

Iron Man.

```
10 PAPER NOT PI
20 INK NOT PI
30 POKE VAL "23624",NOT PI
40 CLEAR VAL "24499"
50 OUT VAL "254",NOT PI
60 RANDOMIZE USR VAL "23789"
70 LOAD "IRNMAN1" CODE
80 LOAD "IRONMAN2" CODE
90 RANDOMIZE USR VAL "25E3"
100 REM NO POKE
110 RANDOMIZE USR VAL "24036"
```

Как видим, данная программа практически не имеет стройной системы защиты; такой вид она имела сразу же после того, как ее обработал Билл Гильберт. В строке 100 хаккер сам предупреждает нас, что в программе отсутствуют защитные POKES. Структура прграмны на БЕЙСИКе достаточно проста. Строки 10-20 делают цвет INK и PAPER черный, строка 50 делает черным и цвет бордюра. Как Вы могли заметить, в программе трижды используется запуск процедур в машинных кодах, но нас будет интересовать только самый первый запуск в строке 60. Именно благодаря ему создается оригинальное текстовое сообщение характеризующееся необычным сочетанием цветов, и использованием необычного шрифта. Рассмотрим подпрограмму в машинных кодах более подробно.

```
            CALL N5D07
            CALL N5D23
            CALL N5D45
            CALL N5D64
            CALL N5D77
            CALL N5DA1
            LD HL,3D00
            DEC H
            LD (5C36),HL
            RET

N5D07       LD HL,3D00
            LD DE,9C40
            PUSH DE
            LD BC,0300
N5D11       LD A,(HL)
            RRA
            OR (HL)
            LD (DE),A
            INC HL
            INC DE
            DEC BC
            LD A,B
            OR C
            JR NZ,N5D11
            POP DE
            DEC D
            LD (5C36),DE
            RET
```

```
N5D23       LD HL,5DD6
            LD B,1E
N5D28       LD A,(HL)
            CP 0D
            JR Z,5D30
            INC HL
            DJNZ N5D28
            LD A,1E
            SUB B
            LD (5D40),A
            LD B,A
            LD A,20
            SUB B
            SRL A
            LD (5D41),A
            RET

N5D40       XOR A
            CALL 1601
            LD A,16
            RST 10
            XOR A
            RST 10
            LD A,(5D41)
            RST 10
            LD A,10
            RST 10
            XOR A
            RST 10
            LD A,(5D40)
            LD B,A
            LD HL,5DD6
N5D5E       LD A,(HL)
            RST 10
            INC HL
            DJNZ N5D5E
            RET

N5D64       LD HL,5CD0
            LD A,16
            RST 10
            LD A,01
            RST 10
            INC A
            RST 10
N5D6F       LD A,(HL)
            CP 0D
            RET Z
            RST 10
            INC HL
            JR N5D6F

N5D77       XOR A
            LD (5D42),A
            LD B,07
N5D7D       PUSH BC
            LD A,(5D42)
            LD HL,5AC0
            LD DE,5AC1
            LD BC,001F
            LD (HL),A
            LDIR
            CALL N5D9B
            LD A,(5D42)
            INC A
```

```
            LD (5D42),A
            POP BC
            DJNZ N5D7D
            RET
N5D9B       LD B,05
N5D9D       HALT
            DJNZ N5D9D
            RET

N5DA1       LD IX,0000
            LD (5D43),IX
            LD HL,5AF0
            LD BC,0011
            LD D,05
N5DB1       PUSH HL
            PUSH BC
            LD BC,(5D43)
            PUSH BC
            INC BC
            LD (5D43),BC
            POP BC
            PUSH HL
            SBC HL,BC
            LD (HL),D
            POP HL
            ADD HL,BC
            LD (HL),D
            POP BC
            POP HL
            HALT
            HALT
            HALT
            DEC BC
            LD A,B
            OR C
            JR NZ,N5DB1
            RET
```

Если Вы внимательно присмотритесь к программе в машинных кодах, то обнаружите, что текст ее составлен весьма профессионально. Программа на языке Ассемблера обычно характеризуется очень большим объемом. Чтобы облегчить работу с ней, некоторые логически законченные действия оформляются в виде отдельных процедур - подпрограмм, которые следуют одна за другой. Управление этими процедурами осуществляется из головной программы на языке Ассемблера, которая последовательно вызывает каждую процедуру, используя оператор CALL.

Внимательно присмотревшись к листингу программы, Вы обнаружите, что программа Билла Гильберта не исключение. В этом листинге очень четко просматривается головная программа управления и процедуры в машинных кодах.

Рассмотрим, что выполняют процедуры этой программы. Для этого проследим, какой визуальный эффект получается при работе программы в целом. После запуска программы экран становится черным, после этого на нижних строчках появляется надпись "IRON MAN" и постепенно высвечивается сообщение "DISKED BY BILL GILBERT 1991". После того, как последняя надпись высветилась полностью, все буквы начинают переливаться разными цветами. Особый колорит оформления достигается за счет использования в программе оригинального шрифта.

Как Вы видите, программа состоит из шести подпрограмм, каждая из которых выполняет определенную функцию для достижения визуального эффекта, описанного выше. Таким образом, одна из них формирует утолщенный шрифт, следующая затемняет экран, следующая за ней процедура ответственна за распечатку первого текстового сообщения, второе текстовое сообщение, как Вы помните, выводится на экран достаточно оригинальным способом - путем постепенного высвечивания информации от середины к

краям экрана, следующая процедура обеспечивает переливающуюся различными цветами окраску букв.

Однако, наибольший интерес для читателя, на наш взгляд, представляют не все процедуры, используемые Биллом Гильбертом, а лишь некоторые из них. В частности, формирование оригинального шрифта. Поэтому не будем здесь подробно расссматривать каждую процедуру программы в машинных кодах, а остановимся на рассмотрении лишь самых интересных из них, Оставленные за пределами нашего изучения процедуры достаточно просты и не нуждаются в подробном объяснении и комментарии.

Рассмотрим процедуру формирования утолщенного шрифта. Одним из недостатков знакогенератора Спектрума являются тонкие буквы - поэтому многие пользователи жалуются на нечеткость надписей, выводимых на экран. Самым практичным методом создания утолщенного шрифта можно считать моделирование нового знака, как наложение нормального знака и знака, сдвинутого на одну точку влево. Формирование нового шрифта обычно начинается перемещением встроенного знакогенератора "Спектрума" в свободную область оперативной памяти. Следующим этапом является изменение знакогенератора, находящегося в ОЗУ, в соответствии с нашими требованиями и завершающим этапом является переключение системной переменной, указывающей на место расположения нового знакогенератора, образовавшегося в ходе ваших переделок. Рассмотрим подпрограмму на Ассемблере, выполняющую данные функции.

```
N5D07       LD HL,3D00
            LD DE,9C40
            PUSH DE
            LD BC,0300
N5D11       LD A,(HL)
            RRA
            OR (HL)
            LD (DE),A
            INC HL
            INC DE
            DEC BC
            LD A,B
            OR C
            JR NZ,N5D11
            POP DE
            DEC D
            LD (5C36),DE
            RET
```

В начале в регистр HL мы загружаем начало встроенного знакогенератора Спектрума - 3D00. После этого в регистр DE загружается адресе нового месторасположения шрифта, а в регистр BC мы заносим общую длину области знакогенератора. Далее начинается работа в цикле. Мы загружаем в аккумулятор то, что находится в ячейке памяти, на которую указывает регистр HL, то есть первый байт области знакогенератора. Следующий этап - это ротация аккумулятора вправо, причем бит 0 перемещается во флаг переноса, а флаг переноса - в бит 7. Более подробно эта операция показана на структурной схеме, приведенной ниже.

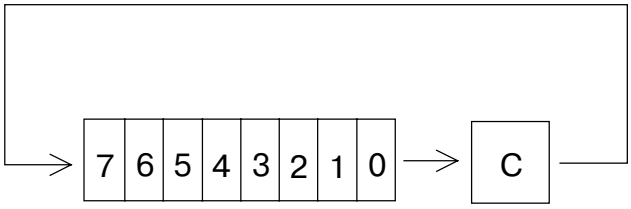


Схема 1.0 : C - флаг переноса.

Таким образом, мы получаем байт знакогенератора, который является смещенным вправо на один бит относительно первоначального байта. Теперь нам необходимо выполнить команду "ИЛИ" (OR) над образовавшимися байтами. Команда "ИЛИ" (OR) исполняется побитным сравнением двух двоичных чисел. Результат ее работы равен единице, если данный бит включен хотя бы в одном из операндов: в первом, или во втором,

или в обоих вместе. Таким образом, в результате может быть ноль только если в обоих операндах ноль, в противном случае получается единица.

Пример 1.0:

```
Первый операнд 1010 1010 (AA)
Второй операнд 1100 0000 (C0)
---------------------------
Результат  OR  1110 1010 (EA)
```

Продолжим рассмотрение программы, формирующей утолщенный шрифт. После того, как мы произвели операцию логическое "ИЛИ" над двумя байтами, мы загружаем образовавшийся байт в ячейку памяти, адрес которой находится в DE. Таким образом начинается формирование нового знакогенератора в свободной области оперативной памяти, поскольку именно в регистре DE содержится начало заданной для этих целей области ОЗУ.

Следующим этапом мы увеличиваем содержимое регистров HL и DE на единицу, т.е. переходим к следующему байту знакогенератора и записываем его измененный вариант в новое место оперативной памяти. В то же время, нам необходим контроль, чтобы модифицировать только байты знакогенератора и не больше. Для этой цели мы используем регистр BC, значение которого уменьшается на единицу после модернизаций очередного байта знакогенератора.

Следующим этапом программы является проверка достижения нуля в регистре BC. Если ноль не достигнут, то мы повторяем все операции сначала, только со следующим байтом шаблона символа. Однако, если в регистре BC остался ноль, это свидетельствует о том, что мы обработали всю область знакогенератора.

После этой проверки мы восстанавливаем со стека значение регистра DE, причем обратите внимание на то, что на стек засылалось значение регистра, которое соответствует месторасположению новой области знакогенератора в ОЗУ. Теперь нам достаточно уменьшить значение регистра D на единицу, и в регистре DE у нас окажется адрес, указывающий на 256 байтов ниже той области памяти, где расположен альтернативный знакогенератор, формируемый нами в ходе выполняемых операций. Теперь нам осталось занести это значение в область системных переменных, чтобы указать на месторасположение нового набора шаблонов символов. Последним шагом является возврат в головную программу в машинных кодах.

Как видим, ничего сложного в подобном преобразовании нет. При желании Вы можете использовать эту процедуру в своей работе. Для этого мы написали похожую процедуру в виде отдельной программы на БЕЙСИКе, запустив которую, Вы сможете подробно разобраться с работой генератора утолщенного шрифта.

```
10 INPUT "START ADRESS = ";ADRES
20 LET SUMMA=0
30 FOR N=ADRES TO ADRES + 68
40 READ BYTE:POKE N,BYTE
50 LET SUMMA = SUMMA + BYTE
60 NEXT N
65 IF SUMMA <> 6932 THEN STOP
100 DEF FN G(A$) = USR ADRES
110 PRINT AT 15,7;
115 LET ZM=FN G(IS IT GOOD ?)
120 PRINT AT 17,7;"AND NOW?"
200 DATA 42,11,92,35,35,35,35,126,92,35,126,67,35,70,235,94
210 DATA 123,254,31,216,254,127,208,22,0,197,229,33,160,0
220 DATA 237,75,123,92,9,235,41,41,41,237,75,54,92,9,6,8,126
230 DATA 203,63,128,18,19,35,16,247,62,2,205,1,22,62,164,215
240 DATA 225,35,193,16,203,201
```

Данная программа являтся перемещаемой: ее можно поместить по любому адресу.

Вызов : LET ZM = FN G ("ТЕКСТ")

Текст может содержать знаки с кодами от 32 до 127 включительно. Если присутствуют

другие символы, вывод на экран автоматически останавливается. Программа строит утолщенные знаки, как GRAPHICS "U", поэтому в своей программе нельзя переопределить этот символ, так как после выполнения команды вывода текста GRAPHICS "U" стирается.

На этом закончим рассмотрение того, как Билл Гильберт оформил программу "Iron Man" и теперь займемся программой "Sim City".

Sim City

```
10 PAPER NOT PI
20 INK NOT PI
30 POKE VAL "23624",NOT PI
40 CLEAR VAL "24499"
50 OUT VAL "254",NOT PI
60 RANDOMIZE USR VAL "23791"
70 LOAD "SIMCITY1" CODE VAL "4E4"
80 CLS
90 RANDOMIZE USR VAL "4E4"
100 POKE VAL "23739",CODE "o"
110 LOAD "SIMCITY2" CODE
120 CLS
130 LOAD "SIMCITY3" CODE
140 RANDOMIZE USR VAL "24500"
150 REM NO POKE
160 RANDOMIZE USR VAL "20480"
```

Как видим, БЕЙСИК-загрузчик данной программы очень схож с аналогичным загрузчиком программы "IRON MAN". В нем практически отсутствуют какие-либо хитроумные операции направленные на защиту программы от просмотра. Билл Гильберт сам предупреждает нас о том что, программа практически не защищена.

Фактически эффект, возникающий в ходе работы программы в машинных кодах очень схож с аналогичным эффектом в программе "Iron Man". Однако, профессиональный программист сразу же обратит внимание на то, что вся информация выводится своеобразным шрифтом. Причем, если в предыдущей программе для вывода сообщения использовался обычный утолщенный шрифт, в данном случае мы имеем дело с неким подобием готического шрифта. Начинающий программист может предположить, что этот шрифт загружается вместе с исходным текстом программы, однако дополнительные 768 байтов может себе дозволить далеко не каждый загрузчик. В нашем примере Биллу Гильберту с помощью использования специальных команд процессора удалось сформировать готический шрифт из обычного шрифта "Спектрума". Кроме того, подпрограмма в машинных кодах, работающая в программе "Sim City", обладает целым рядом новшеств, делающих ее очень увлекательным.

Рассмотрим листинг процедуры в машинных кодах более подробно.

```
              CALL N5D2D
              CALL N5D93
              CALL N5D74
              CALL N5D4A
              CALL N5D02
              CALL N5D69
              RET
              ;
N5D02         LD B,07
              XOR A
              LD (5D73),A
N5D08         LD HL,5AC0
              PUSH BC
              LD A,(5D73)
              INC A
              LD (5D73),A
              LD B,40
              PUSH BC
              LD A,(5D73)
              LD (HL),A
```

```
            INC HL
            POP BC
            CALL N5D25
            DJNZ N5D08
            RET
N5D25       PUSH BC
            LD B,05
N5D2B       HALT
            DJNZ N5D28
            POP BC
            RET
N5D2D       LD HL,5DFA
            LD B,1E
N5D32       LD A,(HL)
            CP 0D
            JR Z,N5D3A
            INC HL
            DJNZ N5D32
N5D3A       LD A,1E
            SUB B
            LD (5D72),A
            LD B,A
            LD A,20
            SUB B
            SRL A
            LD (5D71),A
            RET
            ;
N5D4A       XOR A
            CALL 1601
            LD A,16
            RST 10
            XOR A
            RST 10
            LD A,(5D71)
            RST 10
            LD A,10
            RST 10
            XOR A
            RST 10
            LD A,(5D72)
            LD B,A
            LD HL,5DFA
N5D63       LD A,(HL)
            RST 10
            INC HL
            DJNZ N5D63
            RET
            ;
N5D69       LD HL,3D00
            DEC H
            LD (5C36),HL
            RET
            INC C
            EX AF,AF
            RLCA
            ;
N5D74       XOR A
            CALL 1601
            LD HL,5CD0
            LD A,16
            RST 10
            LD A,01
            RST 10
            LD A,02
```

```
            RST 10
            LD A,10
            RST 10
            XOR A
            RST 10
            LD B,1C
N5D8B       PUSH BC
            LD A,(HL)
            RST 10
            INC HL
            POP BC
            DJNZ 5D8B
            RET
            ;
N5D93       DI
            LD A,R
            LD E,A
            EX DE,HL
            SUB L
            LD L,A
            OR 40
            LD H,A
            SUB H
            PUSH HL
            PUSH HL
            POP DE
            INC A
            LD E,A
            RRA
            CPL
            LD C,A
            SCF
            RLA
            DAA
            RLA
            RES 4,(IY+01)
            RLA
            LD A,(000E)
            INC A
            INC A
            LD B,A
            LD (HL),L
            LDIR
            POP HL
            LD A,H
            SUB 03
            LD H,A
            SUB H
            LD L,A
            XOR C8
            LD D,A
            LD E,L
            LD C,E
            LD B,L
            SET 0,B
            XOR C8
            SET 1A
            OR B
            LD B,A
            PUSH DE
            LDIR
            POP HL
            PUSH HL
            LD BC,(1A22)
            LD B,C
```

```
N5DD7       PUSH BC
            SUB A
            LD B,A
            SET 2,B
            INC HL
            INC HL
            INC HL
            INC HL
N5DE0       LD A,(HL)
            LD C,A
            SLA A
            OR C
            LD (HL),A
            INC HL
            DJNZ N5DE0
            POP BC
            DJNZ N5DD7
            POP DE
            EX DE,HL
            DEC H
            LD (5C36),HL
            EI
            RET
```

Данная программа на Ассемблере по своей структуре очень похожа на рассмотренную ранее в программе "Iron Man". Однако, это только внешнее сходство. Большинство процедур в машинных кодах выполнены по новому принципу. Однако, все они достаточно примитивны и мы надеемся, что читатель сам сможет разобраться с ними. Мы же остановимся лишь на процедуре создания готического шрифта. Она начинается с метки N5D93 (см. листинг выше).

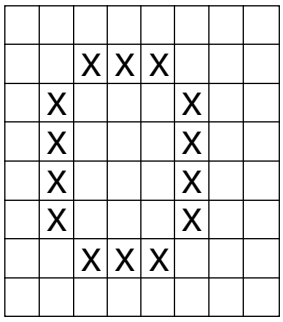
При создании этой процедуры Билл Гильберт, очевидно, не задавался целью сделать ее доступной для понимания широкого круга читателей, скорее наоборот. Он запутал ее как сумел. Нам пришлось провести работу по расшифровке и полученной информацией мы готовы поделиться с читателем.

В этой подпрограмме в машинных кодах Билл Гильберт совместил выполнение двух функций: затемнение экрана (экран становится черным) и создание оригинального готического шрифта. Нас мало интересует процедура затемнения экрана, поэтому все свое внимание мы сконцентрируем на подпрограмме создания готического шрифта. В то же время, мы не можем рассматривать эти процедуры разрозненно, поскольку они тесно связаны между собой, и находятся в единой подпрограмме, которая вызывается второй по счету из головвой программы в машинных кодах, с использованием оператора CALL.

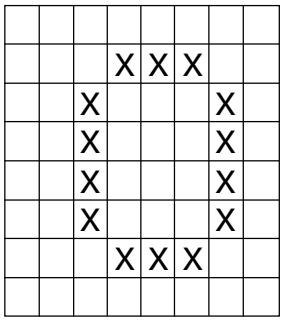
Четкого разграничения между этими процедурами не существует, потому что после выполнения одной Биллу Гильберту понадобилось изменить содержимое всех регистров микропроцессора с целью правильного выполнения второй процедуры. В то же время, вместо того, чтобы напрямую занести содержимое чисел в регистр микропроцессора, Билл Гильберт начинает "заметать следы", используя ряд ухищрений с употреблением лишних в данной ситуации команд микропроцессора. Вместе с тем, к концу махинаций в регистрах оказывается именно те значения, которые необходимы для правильной работы процедуры, создавшей готический шрифт. Это Вы можете легко проверить, используя дизассемблер со встроенной процедурой отладки.

Фактически подпрограмма создания готического шрифта берет свое начало с метки N5DD7. Именно относительно этой метки создается цикл, который преобразует весь встроенный шрифт "Спектрума" в оригинальный готический шрифт. Все это делается тем же способом, как и в программе "IRON MAN". К началу работы процедуры в регистре BC находится значение 0300H, что соответствует длине области встроенного в компьютер шрифта. Регистр HL несет в себе значение 3D00H, что соответствует началу этой области. В регистре DE содержится значение C800H (десятиричный эквивалент - 51200). Это адрес в оперативной памяти, куда осуществляется перенос нового шрифта.

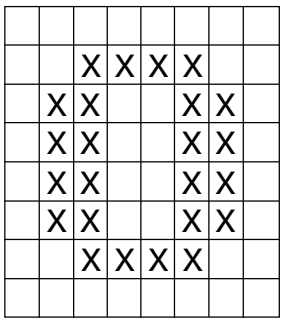
Если рассматривать любой символ шрифта как знакоместо 8X8, то мы можем обнаружить, что в предыдущей программе создания утолщенного шрифта образующийся символ создавался за счет смещения всего исходного символа вправо с сохранением исходного символа на своем месте. В этой программе мы смещаем не полное знакоместо, а лишь последний четыре байта, к тому же смещение этих байтов осуществляется не вправо, а влево, что наглядно демонстрируют приведенные ниже диаграммы.



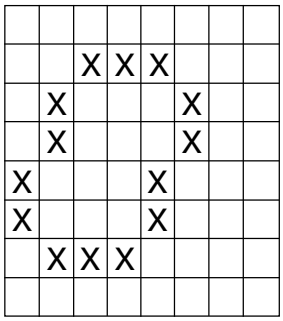
Диаг. 1. Стандартный символ "O" из области знакогенератора "Спектрума".



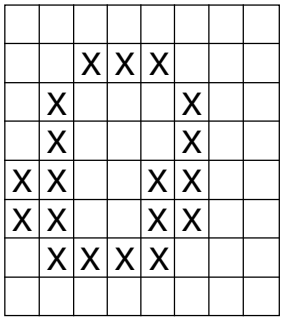
Диаг. 2. Изображение смещенной буквы "O", образующееся после сдвига вправо всех байтов шаблона 8X8. Подобный принцип сдвига реализован в программе "Iron Man".



Диаг. 3. Утолщенная буква "O", образующаяся в результате операции "ИЛИ" над стандартный символом "O" и смещенным символом "O".



Диаг. 4. Символ "O", в котором нижние четыре байта смещены влево операцией SLA.



Диаг. 5. Символ "О", образующийся в результате операции "ИЛИ" над стандартным символом "О" и символом, изображенным на диаг. 4. Принцип используется Биллом Гильбертом в программе "Sim City".

Теперь рассмотрим, каким образом это реализуются в процедуре, написанной в машинных кодах. Перед входом в процедуру мы сохраняем на стеке значение регистра DE, которое, как Вы помните, указывает на место расположения нового шрифта в оперативной памяти компьютера. После этого мы сохраняем на стеке значение регистра BC, которое указывает на номер обрабатываемого символа из таблицы стандартного шрифта. Далее следует очистка аккумулятора, после чего мы очищаем содержимое регистра B, засылая в него содержимое аккумулятора, равное 0. Затем "зажигается" второй бит регистра B. Таким образом Билл Гильберт записывает в регистр в число 4. Это достаточно простой прием, который направлен на то, чтобы запутать начинающего программиста. В данном случае вместо одной команды:

```
LD B,4
```

Гильберт использует целых три, причем результат работы в обоих случаях один и тот же.

На следующем этапе компьютер начинает анализировать стандартный шрифт. Первые четыре байта остаются неизменными, остальные четыре байта нам необходимо сместить на один бит влево и осуществить операцию логического "или" с их исходными аналогами. Делается это следующим образом. Для начала в аккумулятор заносится значение, содержащееся в ячейке памяти, на которую указывает регистр HL. После этого значение регистра A копируется в регистр C. Далее над содержимым регистра A производится команда арифметического сдвига влево: по этой команде все биты сдвигаются. В бит 0 (младший) засылается 0. Бит 7 сдвигается во флаг переноса C. Операция эквивалента умножению байта на 2. Если при этом образуется число большее, чем 255, включается флаг переноса C. Более подробно эта операция показана на структурной схеме.

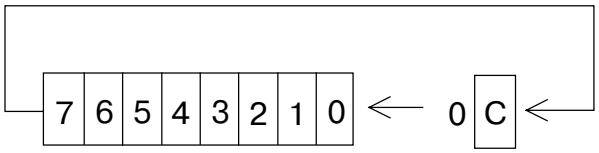


Схема 2.0 : C - флаг переноса.

После выполнения логического сдвига, мы выполняем функцию "ИЛИ" над содержимым регистров A и C. После этого новое значение сохраняем в памяти в области нового шрифта. Далее содержимое регистра HL увеличивается на единицу, чтобы он указывал на значение следующего байта из области стандартного шрифта.

Поскольку в регистре B содержалось 4, а следующей командой идет оператор DJNZ, который задает цикл, мы начинаем повторять все предыдущие операции, пока не очистится регистр B.

Следующим этапом нашей работы является восстановление со стека содержимого регистра DE, который, как Вы помните, указывал на место расположения сформированного нами шрифта. Для того чтобы поместить в соответствующую системную переменную указание на месторасположения данного шрифта, нам необходимо, чтобы в регистре HL содержалось значение на 256 байтов меньше, чем действительный адрес

месторасположения шрифта в ОЗУ. Это достигается уменьшением на единицу старшего регистра пары HL, то есть регистра H. После необходимых изменений мы заносим содержимое этого регистра в системную переменную, указывающую на месторасположение действующего в данный момент шрифта "Спектрума".

В заключение мы восстанавливаем прерывания и возвращаемся в вызвавшую нас процедуру с помощью команды RET.

SINCLAIR LOGO

"ИНФОРКОМ" всегда уделял большое внимание пропаганде языков программирования для пользователей "ZX-Spectrum". Только на страницах "ZX-РЕВЮ" уже были опубликованы "MEGA BASIC" и три версии языка "BETA BASIC". Не прекращаются наши публикации, посвященные АССЕМБЛЕРу. Еще ранее, в 1989 - 1990 г.г. в виде брошюр мы выпустили методички, посвященные языкам "LASER BASIC", ПАСКАЛЬ (HP4T) и "ZX-FORTH".

Сегодня мы начинаем публикацию языка программирования "ЛОГО". Об этом языке написано уже немало. Имеется достаточный выбор отечественной литературы и мы не стали бы повторяться, если бы не одно маленькое обстоятельство. Дело в том, что доступные книги по "ЛОГО" никак не привязаны к компьютеру "Спектрум". Мы же предлагаем Вашему внимание книгу, которая так и называется "LOGO on the SINCLAIR SPECTRUM". Таким образом, все, что Вы почерпнете из этой серии статей, вполне может быть использовано Вами в работе с вашим компьютером. Дело остается за малым - найти и запустить программу и освоить новый язык. Не знаем, как обстоит дело с этой программой на наших рынках сейчас, но в конце 80-х она была распространена очень широко и достать ее не было никаких проблем. Автор книги - известный англиийский программист и разработчик обучающего программного обеспечения Грэхем Филд. Авторизованный перевод с английского языка и редакция - наши.

1. Что такое "ЛОГО".

ЛОГО - это язык программирования, начало разработки которого было положено в Массачусетском Технологическом институте в 1967 году.

В его основе лежит концепция подготовки языка, удобного как для подготовки детей к основам компьтерной грамотности, так и для решения математических задач и задач, относящихся к области искусственного интеллекта. Использование этого языка программирования для этих целей стало возможным в результате значительного объема исследований, проведенных как в МТИ, так и в других научных и учебных центрах США и Европы, в частности, в Эдинбургском университете.

Уже первый опыт использования этого языка показал его высокую эффективность в обучении детей, а также особый интерес к нему не только со стороны учащихся, но и со стороны родителей и педагогов. Вместе с тем, ЛОГО не следует рассматривать только с этой стороны. Это не просто игрушка и в опытных руках он может быть серьезным инструментом. У ЛОГО есть ряд значительных преимуществ по сравнению, например с БЕЙСИКом и он может быть вполне использован в качестве универсального средства решения повседневных задач на компьютерах бытового класса. Итак, несмотря на то, что ЛОГО прост в освоении детьми и лицами, не имеющими никакой компьютерной подготовки, он является мощным инструментальным средством, обеспечивающим удобный доступ к ресурсам компьютера.

"Черепашка".

Для того, чтобы проще приобщить детей к ЛОГО, было изобретено специальное механическое устройство - "Черепашка". Оно представляет собой маленькую механическую тележку с электродвигателем, подключенную к компьютеру с помощью проводов. Тележка может кататься по полу, по столу и т.п., но как правило ее запускают по большому листу бумаги. На тележке установлен держатель, в котором закрепляется перо, карандаш, фломастер. Перо может подниматься и опускаться по командам от компьютера. В случае перемещения "черепашки" по бумаге с опущенным пером, вычерчивается линия, соответствующая траектории движения. Для управления "черепашкой" служит программа, написанная на ЛОГО.

В принципе, "Спектрум" тоже может управлять подобной "черепашкой", как и многими другими электромеханическими устройствами, но мы от нее отвлечемся и будем думать о ней, имея дело с кусором на экране телевизора, с помощью которого будем строить

различные простейшие геометрические фигуры, за которыми так и закрепилось в литературе название "черепашья графика".

Вся прелесть (и мощь) ЛОГО состоит в том, что если Вы один раз "объяснили" компьютеру как надо исполнять ту или иную задачу, Вы можете больше никогда ему этого не "объяснять". Для этого надо Вашему объяснению присвоить какое-либо имя. После этого данная задача становится для компьютера известной ему процедурой и впоследствии, когда Вам опять потребуется исполнить такую же задачу, достаточно только назвать имя нужной процедуры. Получается так, что по мере того, как в своем понимании программирования развиваетесь Вы, также параллельно вместе с Вами развивается и Ваш компьютер. Единственным ограничением при этом является размер памяти Вашей машины, но у Вас есть возможность сохранять готовые процедуры на магнитной ленте и загружать в память только те, которые нужны для решения текущей проблемы.

ЛОГО был разработан специалистами, основным направлением исследований которых является искусственный интеллект, главная проблема которого состоит в изучении приемов мышления человека и введении аналогичных приемов в компьютер. Это исследование привело в свое время к созданию языка программирования ЛИСП, основная задача которого состоит в обработке различный списков данных. К сожалению, оказалось, что этот язык труден для усвоения. Может быть он и не столь труден, как это кажется начинающим, но тем не менее он по крайней мере неудобен хотя бы тем, что содержит в своей основе многие специфические термины, которые относились к конкретным особенностям той ЭВМ, на которой он создавался и которая уже давним давно не существует. ЛОГО родился, как изящная альтернатива, сочетающая методический подход ЛИСПа с доступностью терминологии и естественной наглядностью.

К сожалению, по отношению к ЛОГО сложилось обывательское негативное представление о том, что "черепашья графика" - это основной аспект языка. Есть даже некоторые несведущие люди, убежденные в том, что это единственная область его применения. В последние годы даже появились некоторые программы, в той числе и высококлассные, которые занимаются "черепашьей графикой" и потому содержат слово "ЛОГО" в своих названиях. "СИНКЛЕР ЛОГО" к ним не относится. Эта версия языка для компьютера "Спектрум" содержит все возможности оригинала и даже добавляет некоторые (например обращение к подпрограммам в машинных кодах), которые связаны со спецификой компьютера и не содержатся в эталонном языке, но могут быть очень полезны авторам неординарных программ.

Как и другие языки программирования, ЛОГО имеет немало специфических терминов. Многие не любят новой терминологии и боятся ее во всех областях, кроме той области, в которой сами являются специалистами. Но, тем не менее, некоторые термины все-таки необходимы, если мы хотим серьезно обсуждать возможности нового языка и смотреть, как они соотносятся с другими языками. Вы ничуть не хуже будете водить машину, если не будете знать, что те четыре штуковины, которые торчат из двигателя, называются свечами зажигания. Но это знание немного улучшит ваше положение, когда в один прекрасный день Ваш автонобиль не заведется. Оно поможет понять что произошло, когда кто-то более опытный объяснит Вам ситуацию и пригодится при покупке запчастей.

Эта книга не перенасыщена терминологией и не шокирует начинающего читателя. Все необходимые технические термины будут разбираться по мере их появления на простых и понятных примерах.

ЛОГО и БЕЙСИК

Для тех, кто уже знаком с БЕЙСИКом, мы дадим краткое сравнение этих языков. Если же БЕЙСИК Вам еще неизвестен, то этот раздел Вы можете просто пропустить.

Эти языки во многом различаются. Наверное наиболее очевидным отличием, сразу бросающимся в глаза, является то, что в ЛОГО нет номеров строк в программе. Они просто не нужны. Если Вам нужно внести изменения в часть программы, написанной на ЛОГО, Вы должны воспользоваться программой-редактором, которая является частью ЛОГО-системы. Этот редактор отличается от того строчного редактора, с который Вы имеете дело

на "Спектруме". С его помощью можно обрабатывать не одну программную строку, а сразу целую процедуру или даже несколько. И этот редактор намного более удобен, чем тот, с которым Вы работаете в стандартном БЕЙСИКе.

Процедуры ЛОГО обычно короче и даже намного короче, чем программы БЕЙСИКа. Благодаря этому их легче писать, проще отлаживать и легче понимать, чем БЕЙСИК-программу. Редактор может одновременно обрабатывать большее количество строк, чем помещается на одном экране, но в ЛОГО не считается хорошей практикой делать процедуры столь длинными.

Еще одно необычное отличие состоит в том, что если Вы привыкли к тому, что в памяти компьютера может содержаться одновременно только одна БЕЙСИК-программа, то в ЛОГО одновременно в памяти могут быть несколько совершенно не связанных друг с другом процедур.

Еще одно, незаметное на первый взгляд отличие - то, что каждый оператор БЕЙСИКа имеет собственную структуру и всегда должен быть записан только в соответствии с ней. ЛОГО же, с другой стороны, имеет только одну допустимую структуру для своих операторов и две небольших вариации, которые не обязательны, а существуют только для большего удобства программиста.

Если Вы привыкли к тому, что операторы и функции БЕЙСИКа вызываются одним нажатием клавиши, то может быть будете несколько разочарованы, поскольку в ЛОГО этого нет. Это не позволяет сделать сама структура языка. Вам придется набирать все слова полностью по буквам. Впрочем, многие из стандартных инструкций ЛОГО имеют сокращения и к тому же в ЛОГО есть возможность самому задать сокращения для прочих слов, если Вам это хочется.

Есть еще много других отличий, о которых имеет смысл говорить после освоения обоих языков, но мы на них сейчас не будем останавливаться. Сказанного выше уже достаточно для того, чтобы Вы поняли, что ЛОГО - это совершенно особый язык программирования, не имеющий к БЕЙСИКу никакого отношения.

Диалекты ЛОГО.

Если Вы сравните те конструкции, которые увидите в этой книге, с другими книгами по ЛОГО, то найдете очень значительные отличия. Так происходит потому, что ЛОГО, по видимому, имеет большее количество диалектов, чем любые другие языки. Это, конечно, неприятно, но на самом деле ЛОГО это ведь не столько язык программирования, сколько необычный и весьма продуктивный образ мышления. Во всяком случае, основные структуры, как и возможности того или иного диалекта, будут как правило похожими. Разобравшись, Вы сможете конвертировать другие программы, написанные на ЛОГО для других компьютеров в форму, приемлемую для "Спектрума". Слова могут отличаться, но стояние за ними идеи будут теми же.

Как пользоваться этой книгой.

При написании программ на ЛОГО общепринято пользоваться прописными (заглавными) буквами и делать отступ в конструкциях языка. В этой книге будет использован тот же подход, но Вы можете им и не пользоваться, поскольку Синклер-ЛОГО - достаточно гибкий диалект. В нем можно без проблем использовать и строчные буквы, а отступ в строках - это просто удобная форма представления и чтения процедур, которая для Вас не обязательна.

Так что, если Вы наберете что-то вроде:

```
TO SAYHELLO
   PRINT [ДОБРО ПОЖАЛОВАТЬ В СИНКЛЕР-ЛОГО]
END
```

то на экране компьютера может получиться например так:

```
?to sayhello
>print [ДОБРО ПОЖАЛОВАТЬ В СИ!
>НКЛЕР-ЛОГО)
>end
```

Символы ? и > в начале строк и символ ! в конце - это специфические символы языка и

в свое время мы с ними разберемся.

При работе с книгой желательно попробовать приведенные в ней примеры на своем компьютере, чтобы почувствовать, как ЛОГО реагирует на те или иные команды и на ошибки. Тогда Вы научитесь использовать возможности языка наиболее эффективно. Все примеры в книге, за исключением самых примитивных, преднамеренно сделаны так, что их можно развивать и дорабатывать. Идеи о том, что еще можно доработать во многих случаях даны. В книге нет контрольных вопросов, да они и не нужны. Если в ЛОГО написанная Вами процедура работает и делает то, что должна делать, она написана верно.

2. Первые шаги.

В отличие от БЕЙСИКа, который уже встроен в ПЗУ Вашего "Спектрума", ЛОГО следует загружать с магнитной ленты. Это делается обычным способом:

```
LOAD ""
```

После загрузки на экране появится сообщение:

```
WELCOME TO SINCLAIR LOGO
```

Ниже Вы увидите вопросительный знак, служащий запросом к Вам и сигналом о том, ЛОГО ждет ваших команд. Рядом виден мигающий квадрат - это курсор. Кроме этого, Вы можете увидеть в правом нижнем углу символ "L", который означает, что от Вас ожидается ввод буквы или цифры.

Мы начнем наши эксперименты с "черепашьей графики". Наберите SHOWTURTLE и нажмите ENTER. Набор можно сделать и строчными буквами. Вы увидите, что на экране появится "черепашка" в виде стрелки. Одновременно экран разделится, оставив две нижние строки для ввода Ваших команд. Основная часть экрана будет использована для изображения результатов движений "черепашки".

Попробуйте дать команду

```
FORWARD 50
```

Вы увидите, что стрелка переместилась в том направлении, в котором она указывает. Здесь надо обратить внимание на то, что слова, из которых состоят команды языка ЛОГО должны разделяться пробелами. Так что, если Вы попробуете дать команду FORVARD50, то увидите, что ЛОГО Вас не поймет. Попробуйте это, когда появится сообщение об ошибке. Вы увидите изображение стрелки в правом нижнем углу экрана. Это означает, что Вам нужно нажать какую-либо клавишу, чтобы продолжить работу.

Теперь пойдем дальше:

```
BACK 50
```

Стрелка на экране вернется назад. Число 50 задает расстояние, на которое должна перенеститься "черепашка". Слова FORWARD и BACK - это процедуры, а число 50 - параметр. Трудно сказать, в каких единицах измеряется расстояние перемещения черепашки, поскольку это зависит от размеров экрана Вашего телевизора или монитора, но во всяком случае максимальное количество шагов по вертикали -175, а по горизонтали - 255. Мы можем считать, что расстояние измеряется в пикселах.

Попробуйте подвигаться вперед-назад, привыкая к чувству расстояния и, когда Вам надоест двигаться по одной прямой, мы начнем осваивать новые направления. Сначала наберите CLEARSCREEN. Если Вы немного знаете английский, то уже очевидно догадались, что эта процедура очищает экран и при этом выставляет "черепашку" в исходную позицию.

Чтобы повернуть "черепашку", можно воспользоваться процедурами RIGHT (направо) и LEFT (налево). Эти процедуры должны иметь при себе параметр - угол, на который следует повернуться. Угол измеряется в градусах, полная окружность - 360 градусов, поэтому команда RIGHT 180 развернет "черепашку" в противоположном направлении, а команда LEFT 90 - повернет ее на прямой угол.

Попробуйте теперь:

```
FORWARD 50
RIGHT 120
FORWARD 50
RIGHT 120
FORWARD 50
```

Вы увидите, что "черепашка" пройдет по замкнутому контуру и встанет в той точке, с которой начинала. Справедливости ради надо сказать, что ее положение не строго идентично исходному, поскольку смотрит она не туда. Но если Вы добавите еще одну команду RIGHT 120, то она встанет точно в исходное положение.

Может быть, Вы сообразили, что это простой и изящный метод рисования треугольников? Да, наверное, это так. А как быть с пятиугольниками или, скажем, с десятиугольниками? Не кажется ли Вам, что рисовать фигуру с десятью сторонами и десятью углами таким приемом было бы несколько утомительно? К счастью, ЛОГО имеет простое средство избавиться от этой неприятности. Опять исполните CLEARSCREEN и попробуйте такую команду:

```
REPEAT 3(FORWARD 50 RIGHT 120)
```

Здесь используются квадратные скобки, просьба не путать с круглыми. В ЛОГО разные виды скобок используются для разных целей.

Процедура REPEAT означает, что то, что стоит в квадратных скобках, должно быть повторено. Параметр 3 указывает на количество повторений.

Команды REPEAT могут быть довольно длинными. Для того, чтобы их было удобнее читать, строку разбивают в тех местах, где удобно и продолжение строки печатают с отступом.

Если при написании команды она дойдет у Вас до правого края экрана и начнется на новой строчке не пугайтесь, этот разрыв не страшен. ЛОГО не считает строку законченной до тех пор, пока не нажата клавиша ENTER. В качестве указателя того, что команда продолжается на новой строке компьютер поставит в конце экранной строки восклицательный знак "!", служащий здесь как бы символом переноса. Поэтому, если в этой книге Вам встретится например такая команда:

```
REPEAT 8 (FORWARD 20 RIGHT 90
          FORWARD 10 RIGHT 90
          FORWARD 20 LEFT 15]
```

Смело набирайте ее на экране в одну строку, давая ENTER только тогда, когда будет введено все до конца.

Но мощь ЛОГО, конечно же заключена не в способности многократно повторять одни и те же действия, а в способности создавать новые команды. Поскольку Вы теперь знаете, как изображается треугольник, Вы можете теперь создать команду для его изображения.

Сначала напечатайте:

```
TEXTSCREEN
```

По этой команде "черепашка" будет удалена с экрана и все 22 строки экрана будут доступны для написания текста. А теперь делайте так:

```
TO TRIANGLE
```

(появится приглашение ">")

```
REPEAT 3 [FORWARD 50 RIGHT 120]
```

На сей раз команда не будет выполнена непосредственно, поскольку оператор TO... свидетельствует о том, что Вы не рисуете треугольник в данный момент, а занимаетесь тем, что объясняете компьютеру, как это делается. Вместо этого появляется новое приглашение ">", оно означает, что Вы не закончили описание новой команды. Чтобы завершить дело, нужно набрать еще один оператор:

```
END
```

Компьютер ответит Вам на это следующим сообщением:

```
TRIANGLE defined
```

Это означает, что процедура для изображения треугольников под именем TRIANGLE определена и теперь она известна компьютеру.

После этого на экране появится приглашение ">". Оно означает, что ЛОГО находится в командном режиме и ждет от Вас ввода команды. Перейдите к "черепашке" командой SHOWTURTLE и дайте команду TRIANGLE. Если все было сделано правильно, треугольник будет нарисован. Отныне команда TRIANGLE не менее знакома компьютеру, чем команды FORWARD и BACK.

Более того, теперь Вы можете встраивать команду TRIANGLE в другие, еще более мощные команды. Попробуйте, например:

```
REPEAT 12 [TRIANGLE RIGHT 30]
```

- и посмотрите, что будет нарисовано на экране. Попробуйте также самостоятельно оформить эту команду в процедуру и дайте ей имя по вкусу.

Развивая мысль, мы с Вами попробуем нарисовать человечка. Он (или она) будет состоять из квадратной головы, туловища, которое будет изображено одной прямой линией, пары прямых рук и пары прямых ног (см. рис. 1).

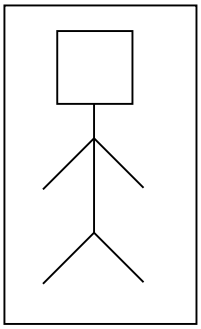


Рис. 1.

Программа выглядит очень просто:

```
TO MAN
   HEAD          (голова)
   BODY          (тело)
   LEGS          (ноги)
   ARMS          (руки)
END
```

Можете набрать эту процедуру. Пусть пока Вас не смущает, что ни мы, ни компьютер не знаем пока определения процедур HEAD, BODY, LEGS и ARMS. Мы их еще напишем. Если при их написании Вам захочется посмотреть, как они работают, просто перейдите в командный режим (курсор "?") и дайте имя команды.

Итак, "голову" мы изобразим, как прямоугольник и процедура очень похожа на ту, которая изображала треугольники. Но нам надо еще после всего поместить "черепашку" в центр нижней стороны и развернуть ее вниз, чтобы она была готова перейти к рисованию "тела". Для этого потребуются дополнительно три команды. В целом процедура HEAD будет выглядеть так:

```
TO HEAD
   REPEAT 4 [FORWARD 20 RIGHT 90]
   RIGHT 90
   FORWARD 10
   RIGHT 90
END
```

"Тело" еще проще:

```
TO BODY
   FORWARD 40
END
```

Для изображения "ног" развернем слегка "черепашку" влево, переместим вперед, вернем назад, повернем вправо на вдвое больший угол, чем в начале, опять вперед и назад и снова развернем влево, возвратив в первоначальное положение.

```
TO LEGS
   LEFT 30
   FORWARD 30
   BACK 30
   RIGHT 60
   FORWARD 30
   BACK 30
   LEFT 30
END
```

Если Вы посмотрите на рисунок, то увидите, что "руки" ничем принципиально от "ног" не отличаются, поэтому для изображения "рук" достаточно отогнать "черепашку" назад и пририсовать к "телу" пару "ног" на уровне "плеч".

```
TO ARMS
   BACK 35
```

```
    LEGS
END
```

Обратите внимание на саму методику работы. Вы видели, как мы упростили задачу рисования человечка до четырех задач рисования его отдельных частей. Это очень удобный метод программирования. Он состоит в расчленении одной большой задачи на группу более простых, с каждой из которых можно заниматься порознь. Их, в свою очередь, тоже можно дробить до тех пор, пока вся задача, как бы сложна она ни была, не будет сведена к элементарнейшим операциям.

ИМЕННО ДЛЯ ТАКОГО СТИЛЯ ПРОГРАММИРОВАНИЯ ЯЗЫК ЛОГО И ПРЕДНАЗНАЧЕН.

Если Вы в восторге от того, что нарисовали, можете вывести рисунок на принтер по команде COPYSCREEN.

Редактирование.

Если до сих пор Вы не сделали ни одной ошибки, считайте, что Вам повезло, но надо все-таки знать, каким образом можно внести изменения и исправления в свои процедуры. Для этого служит редактор (editor). Он вызывается по команде EDIT, после которой надо указать имя процедуры, которую Вы хотите переработать, например:

```
EDIT "LEGS
```

На экране появится текст процедуры LEGS. Курсор перемещается обычными "синклеровскими" курсорными клавишами 5,6,7,8 в сочетании с CAPS SHIFT.

Нажатие клавиши ENTER в той позиции, где стоит курсор, создает новую строку. Вы можете добавить новый текст, например PRINT "HELLO.

Стирание выполняется как обычно с помощью DELETE (CAPS SHIFT + 0).

При редактировании Вы можете воспользоваться и расширенным режимом (курсор "E"). После одновременного нажатия CAPS SHIFT и SYMB SHIFT в правом нижнем углу загорится символ "E".

В E-режиме курсорные клавиши действуют немного не так. Совместно с CAPS SHIFT клавиши 5 и 8 "перегоняют" курсор в начало или конец текущей строки, а клавиши 6 и 7 - в нижнюю или в верхнюю строку экрана. Можно перемещать курсор и на еще большие расстояния. Так, в этом режиме при нажатии клавиши "B" Вы попадете в начало текста (BEGIN), а при нажатии клавиши "E" - в конец текста, соответственно (END). Это очень удобно при редактировании процедур, длина которых больше, чем размер экрана.

Еще одно огромное удобство в этом режиме предоставляют клавиши "Y" и "R". С их помощью можно переносить строки программы с места на место. Попробуйте выставить курсор на строке RIGHT 60, затем войдите в E-режим и нажмите "Y". Строка RIGHT 60 исчезнет с экрана, но она не пропала окончательно, а осталась в памяти компьютера. Опустите курсор на две строки вниз и в E-режиме нажмите клавишу "R" - строка RIGHT 60 появится на новом месте.

Не бойтесь ничего испортить, поработайте с редактором. Любой редактор имеет свои особенности и к ним надо привыкнуть. Время, затраченное на эксперименты, многократно окупится в будущем.

Выйти из редактора можно двумя способами.

Если Вы не хотите, чтобы все сделанные Вани изменения отразились на программе, Вы можете нажать CAPS SHIFT и SPACE одновременно.

С другой стороны, если Вам надо, чтобы ваши изменения были зафиксированы, выход выполняется нажатием клавиши "C" в E-режиме.

У Вас есть возможность использования клавиш редактора и тогда, когда Вы не находитесь в самом редакторе. Так, например, если при наборе текста Вы заметите, что сделали ошибку, можете "перегнать" курсор и внести исправление до того, как была нажата клавиша ENTER и строка ушла в память компьютера.

Если Вы войдете в редактор, задав имя процедуры, которая до сих пор еще не была определена, то компьютер подумает, что Вы хотите создать новую процедуру. В этом случае перед вами будет пустой экран и слово "TO, напечатанное в его начале. Таким образом, у Вас есть еще один метод создания процедур, отличающийся от того, который мы рассмотрели ранее. В принципе, этот путь более удобный, т.к. к вашим услугам все

имеющиеся ресурсы редактора, в том числе и такие, как вставка новых строк. Те, кто попробовал программировать самостоятельно, знают как это важно.

Сохранение программ на кассете.

Мы дошли то той точки, где Вам может быть захочется сохранить только что написанные процедуры для дальнейшего использования, выключить компьютер и немного отдохнуть. Если просто выключить "Спектрум", то все, что хранилось в его памяти, будет навсегда утрачено. Даже если Вам и нет нужды сохранять результаты своих экспериментов, Вы все же попробуйте это сделать, чтобы убедиться, что все идет нормально.

Как и большинство языков программирования, ЛОГО имеет дело с "файлами" (так мы будем называть блоки информации, записанные на магнитофонной ленте (или на диске и т.п.). Каждый файл имеет свое собственное имя. В ЛОГО имена файлов (в отличие от прочих ключевых слов) не могут иметь более 7 символов. Это ограничение в данном случае не связано с ЛОГО, оно относится к самому "Спектруму". Так, например, если мы хотим сохранить процедуры, которые использовались при рисовании человечка в файле под именем PERSON, наберите:

```
SAVE "PERSON [MAN HEAD BODY LEGS!
       ARMS] <ENTER>
```

Включите магнитофон на запись и нажмите любую клавишу. Выгрузка идет, как обычно. По окончании сохранения файла на экране появится приглашение "?".

Если теперь Вам надо будет загрузить этот пакет процедур обратно в память компьютера, Вы можете делать следующим образом: LOAD "PERSON <ENTER>

Обслуживание памяти.

Система ЛОГО занимает часть памяти Вашего компьютера. Еще часть памяти расходует сам ЛОГО, отслеживая ход исполняемой работы. Нужна также память для хранения промежуточных результатов вычислений. Так, например при расчете 2+5-3 компьютер сначала долже найти 2+5 и запомнить где-то этот промежуточный результат. Оставшаяся часть памяти может быть использована для хранения Вашей информации и созданных Вами процедур. Мы назовем эту часть памяти "рабочей областью".

Рано или поздно программист так или иначе столкнется с проблемой нехватки памяти в рабочей области. В этот момент ему было бы чрезвычайно важно узнать, какие же процедуры хранятся у него сейчас в памяти. Так, например, возможным выходом из создавшегося положения могло бы быть удаление части процедур, если они для работы не нужны.

Для этой цели ЛОГО имеет несколько встроенных команд. Вы можете попробовать их в работе, но для того, чтобы увидеть, как они действуют, надо, чтобы в памяти сейчас у Вас было несколько Ваших процедур. Так что если их нет, то либо наберите их, либо загрузите с ленты (если при прочтении прошлого параграфа Вы их отгрузили).

Первая команда, которую мы рассмотрим - это команда POTS. Название может быть выглядит не очень осмысленным, но это даже хорошо, так ее проще запомнить. Расшифровывается она, как Print Out TitleS (распечатать заголовки). По этой команде Вы можете получить, например вот такой результат:

```
TO TRIANGLE
TO MAN
TO HEAD
TO ARMS
TO BODY
TO LEGS
```

Конечно это не все процедуры, которые знакомы для ЛОГО. Мы, например, знаем, что ЛОГО понимает такие процедуры, как FORWARD, BACK, RIGHT, LEFT и др. Но они по команде POTS не распечатываются, поскольку являются встроенными в систему. Это обязательные элементы языка, их еще называют "примитивами".

Другая команда с не менее нелепым названием - POPS означает Print Out ProcedureS (распечатать процедуры) выполняет печать всех Ваших процедур одну за другой. Так Вы можете их просмотреть.

Просмотреть одну какую-либо процедуру Вы можете в редакторе, вызвав ее по имени, но можно это сделать также и командой PO(Print Out). Например:

```
PO "LEGS
```

Может быть, Вы хотите, чтобы печать Ваших процедур производилась не на экран, а на принтер? Тогда Вы можете дать команду PRINTON. Начиная с этого момента принтер станет устройством вывода информации и результат работы POTS, POPS и PO будет распечатываться на принтере. Отключить принтер и вновь перевести весь вывод на экран можно командой PRINTOFF.

Итак, Вы научились просматривать рабочую область. Освободить ее от ставших ненужными процедур можно с помощью команды ERASE. Например:

```
ERASE "HEAD
```

Попробуйте это сделать, а потом дайте команду POTS для того, чтобы убедиться, что все прошло нормально. Нелишне напомнить, что перед удалением иэ памяти временно ненужных процедур стоит подумать о том, чтобы сохранить их на ленте.

И, наконец, рассмотрим команду для полного освобождения рабочей области. Это команда ERPS (ERase ProcedureS). После нее можно начинать работу сначала.

Все команды, рассмотренные в этом разделе, имеющие дело с конкретной процедурой, например EDIT, PO, ERASE должны иметь открывающие кавычки перед именем этой процедуры. Но они могут иметь после своего имени в квадратных скобках еще и список входящих в них процедур, так, как это было при рассмотрении команды SAVE. В этом случае поданная команда распространяется не только на головную процедуру, но и на все входящие, например:

```
EDIT "ARMS [LEGS]
```

даст Вам возможность одновременно редактировать не только процедуру ARMS, но и процедуру LEGS. Это может быть весьма удобно например в тех случаях, когда Вы хотите перенести строку из одной процедуры в другую.

Команда PO [HEAD BODY] распечатает обе процедуры, причем первой пойдет HEAD.

Команда ERASE [MAN TRIANGLE] удалит обе процедуры из рабочей области.

Печать текста.

Если Вы хотите распечатать на экране какое-либо сообщение, Вы можете воспользоваться командой ЛОГО - PRINT. Обратите внимание на то, что если включен режим PRINTON, то с тем же успехом печать будет выполнена на принтере.

Если надо распечатать всего одно слово, то оно может быть написано после кавычек, например:

```
PRINT "HELLO
```

В этом случае есть разница, какими буквами прописными или строчными Вы набрали свое слово. Напечатано оно будет точно так, как было набрано.

Если Вам надо напечатать целое предложение (иди список слов), то весь список должен быть заключен в квадратные скобки:

```
PRINT [WELCOME TO LOGO]
```

Если Вы хотите напечатать пустую строку, воспользуйтесь командой PRINT " или PRINT [].

Мы продемонстрируем возможность этой команды на следующем примере, который назовем TZAR ("Царь").

```
TO TZAR
   PRINT []
   PRINT [Жил был царь]
   PRINT [у царя был двор]
   PRINT [На дворе был кол]
   PRINT [На колу мочало]
   PRINT [Не начать ли сказку!
          сначала?]
   TZAR
END
```

Запустите процедуру командой TZAR. Остановить ее можно будет только нажав BREAK (CAPS SHIFT + SPACE).

3. Переменные.

В ЛОГО, как и в большинстве других языков, можно пользоваться словами для выражения каких-либо данных, например чисел. Так, например, предположим, перед Вами стоит задача рассчитать величину налога на добавочную стоимость (НДС). Предположим, что стоимость изделия - 30 руб. в этом случае Вы можете присвоить переменной PRICE значение 30. Это делается командой MAKE:

```
MAKE "PRICE 30
```

Кавычки перед словом PRICE обозначают, что это имя переменной, а не имя процедуры.

Теперь попробуйте дать команду

```
PRINT "PRICE
```

Компьютер напечатает слово PRICE. Да это и естественно, ведь кавычки перед ним говорят о том, что это просто слово, а не переменная. В то же время, есть возможность напечатать не PRICE, а то, что этим словом обозначено. Для этого может служит двоеточие. Например:

```
PRINT :PRICE
```

Теперь компьютер напечатает число 30. Итак, в команде PRINT кавычки " обозначают само слово, а двоеточие (:) обозначает значение переменной, обозначенной этим словом.

Теперь мы можем перевести на язык ЛОГО следующее утверждение: "НДС равен 20-ти процентам от стоимости изделия". Оно будет выглядеть так:

```
MAKE "NDS :PRICE/100*20
```

Итак, команда MAKE требует наличия двух аргументов. Первый - это имя переменной (слово), а второй - это то значение, которое эта переменная теперь будет иметь (например число). Не забывайте о кавычках перед словом, которое идет после MAKE. Есть случаи, когда эти кавычки не нужны, но это особые случаи.

Обратите внимание и на то, что перед словом PRICE в нашем примере стоит двоеточие. Это потому, что не слово PRICE нас интересует, а то, что за ним скрывается, его числовое значение, равное 30, которое мы хотим разделить на 100 и умножить на 20 (НДС в России в 1993 г. равен 20-ти процентам). Как видите символы умножения (*) и деления (/) совпадают с БЕЙСИКом. После того, как переменная NDS создана командой MAKE, мы можем ее напечатать:

```
PRINT :NDS
```

- и опять двоеточие, поскольку нам нужно не слово, а его значение.

В компьютерной терминологии слова PRICE и NDS называются переменными, поскольку они представляют заранее неизвестные значения, которые к тому же по ходу расчетов могут и изменяться. Посмотрите, например, как оперируют с переменной, выражающей изменяющуюся длину:

```
TO TRISPI
   SHOWTURTLE
   MAKE "LENGTH 10
   REPEAT 21 [FORWARD :LENGTH RIGHT 120  MAKE "LENGTH :LENGTH +5 ]
END
```

Внимательно посмотрите на последнюю команду в квадратных скобках. Этой командой переменной LENGTH присваивается новое значение путем прибавления к ранее имевшемуся значению числа 5. Запустите эту процедуру и посмотрите, как она работает.

При работе с переменными ЛОГО различает прописные и строчные буквы. Так,

```
MAKE "NDS 7
MAKE "nds 4
```

создают две совершенно различные переменные.

```
PRINT :NDS даст 7
PRINT :nds даст 4
```

Это может вызвать проблемы, если Вы печатаете все в нижнем регистре и забываете ставить двоеточия в кавычки там, где это нужно. Так, наберите:

```
print nds
```

и компьютер сам трансформирует запись в

```
PRINT NDS
```

поскольку он будет полагать, что и print и nds - имена процедур.

Поставить двоеточие перед NDS будет недостаточно, придется в редакторе обратно переправить NDS на nds.

Арифметика.

Кроме арифметических знаков (*),(+) и (/), с которыми мы уже встречались, существуют также и арифметические процедуры. Так, например, процедура SUM отыскивает сумму чисел, следующих за ней. Обычно это бывают два числа, но в некоторых случаях их бывает и больше. Тогда надо ставить круглые скобки так, как показано ниже. Примеры:

```
PRINT SUM 2 3
    5
PRINT (SUM 1 2 3 4 5)
    15
```

Процедура PRODUCT выполняет умножение. Примеры:

```
PRINT PRODUCT 7 3
    21
PRINT (PRODUCT 2 3 5 7)
    210
```

Внимание: таким путем для встроенных функций могут вводиться более, чем по два аргумента. Это может вызвать проблему, если их использовать внутри квадратных скобок, например в команде REPEAT. Таких случаев следует избегать.

Продедура DIV выполняет деление. Она может использоваться только с двумя аргументами, например:

```
PRINT DIV 10 5 2
```

Внимание: в ЛОГО, как и во всех других языках деление на ноль невозможно. Проверьте и посмотрите, что будет:

```
PRINT DIV 5 0
```

Это обычная ошибка начинающих программистов, особенно часто возникающая, когда делятся две переменные:

```
PRINT DIV :TOTAL :NUMBER
```

Программисты забывают учесть, что при определенных ситуациях переменная NUMBER может принимать нулевое значение.

Для вычитания нет своей процедуры. Поэтому это действие можно выполнить либо с помощью знака (-) либо с помощью суммирования SUM с отрицательным аргументом.

```
PRINT 12 - 4
```

дает 8

Знак "минус" выполняет двойную роль. Между двумя числами он обозначает вычитание, но если он стоит перед целым числом, то обозначает, что это число отрицательное.

Во всех случаях ЛОГО пытается разобраться с тем, что Вы написали и во многие случаях делает это весьма успешно. Так,

```
PRINT 12-4
```

или

```
PRINT 12 - 4
```

будет принято успешно.

Вы можете вычесть и отрицательное число, например:

```
PRINT 12 - - 4
```

или

```
PRINT 12--4
```

Тем не менее, все-таки существут случаи, когда ЛОГО может быть введен в заблуждение неадекватным использованием знака "минус", поэтому на всякий случай выработайте в себе привычку всегда ставить пробелы перед знаком "минус" у отрицательного числа и не оставлять пробела между знаком "минус" и самим числом. Так, последний пример наиболее грамотно записать так:

```
PRINT 12 - -4
```

В одном выражении Вы можете объединять все четыре арифметических действия.

Например:

```
PRINT 3*4 + 2
```

дает 14

И дело вовсе не в том, что вычисления выполняются слева направо. Сравните:

```
PRINT 2 + 3*4
```

тоже дает 14

Порядок арифметических действий, принятый в ЛОГО выглядит так:

- действия в скобках;
- деление;
- умножение;
- вычитание;
- сложение.

Таким образом, Вы можете пользоваться скобками () для того, чтобы изменять естественный порядок действий. Например:

```
PRINT (2 + 3) * 4
```

даст 20

При расчете выражения, стоящего в скобках, ЛОГО придерживается естественного порядка действий. Если Вы не уверены при написании выражения в порядке его расчета, смело расставляйте скобки, чтобы порядок был именно таким, какой Вам нужен. Еще два примера:

```
MAKE "CENTIGRADE (:FAHRENHEIT - 32) * 5 / 9
```

выполняет сначала вычитание, а

```
MAKE "FAHRENHEIT :CENTIGRADE *9/5 + 32
```

выполняет деление, умножение и последним сложение.

С точки зрения арифметики все равно выполнять сначала умножение или деление, но с точки зрения ЛОГО деление лучше выполнять раньше, чтобы избежать появления слишком больших чисел, с которыми будет трудно работать.

Одна из полезнейших возможностей в использовании переменных состоит в том, что благодаря им можно создавать процедуры со вводными параметрами. Например, Вы хотите иметь процедуру SQUARE для изображения на экране квадратов различной величины. Так, SQUARE 50 должна рисовать квадрат со стороной 50 единиц, а SQUARE 100 - со стороной 100 единиц. В этом случае мы можем ввести параметр SIZE, от которого будет зависеть размер квадрата и, меняя это параметр при вызове процедуры, изменять результат ее работы. Тогда определение процедуры SQUARE будет выглядеть так:

```
TO SQUARE :SIZE
   REPEAT 4[FORWARD :SIZE RIGHT 90]
END
```

В заключение в порядке эксперимента напишите процедуру POLYGON, которая должна иметь два входных параметра :SIZE и :NUMBER. Параметр SIZE задает размер стороны правильного многоугольника, а параметр NUMBER - число его сторон. Можете это сделать, модернизировав приведенную выше процедуру SQUARE. Правда, придется помозговать насчет угла поворота "черепашки" после изображения каждой стороны. Вам может помочь тот факт, известный из геометрии, что сумма углов выпуклого многоугольника равна 180*(n-2) градусов, где n - число его сторон.

А пока мы прощаемся до следующего выпуска.

(Продолжение следует).

AFRICAN SEEDS

Уважаемые читатели! Мы получаем массу писем с просьбой больше давать программ на БЕЙСИКе для самостоятельного набора начинающими. Многие полагают, причем совершенно справедливо, что это является прекрасной школой обучения программированию. Мы с этим утверждением полностью солидарны. И дело даже не в том, что при наборе программы начинающий программист осваивает клавиатуру, использует новые для себя операторы и конструкции языка, узнает новые приемы. Основная прелесть состоит в том, что как бы ни старался читатель набрать программу без ошибок, в 99 случаях из 100 у него это не выйдет. Практически все при любом опыте наделают ошибок. А вот поиск этих ошибок и отладка программы и являются по-настоящему ценной школой начинающего программиста.

Интерпретатор БЕЙСИКа устроен так, что от большинства ошибок он Вас застрахует. Если при наборе строки Вы нарушите синтаксис языка, он Вас немедленно предупредит и строка не будет введена в программу клавишей ENTER до тех пор, пока Вы эту ошибку не исправите. Тем не менее, существуют еще сотни возможных видов ошибок, которые он не увидит. Например, он ничего не сможет поделать, если Вы пропустите какую-либо строку. Об этом Вы узнаете только запустив программу, причем ошибка проявится совсем не в том месте, в котором Вы пропустили строку. Она проявится там, где компьютеру надо будет воспользоваться данными, содержавшийся в пропущенной строке.

Поэтому прежде, чем Вы приступите к набору программы, нам хотелось бы дать Вам некоторые приемы, которые помогут сделать мучительный процесс отладки чуть менее мучительным. Тем не менее, настройтесь все-таки на тяжелую и кропотливую работу.

Во-первых, когда программу набирают со страниц журнала или книги, большой процент ошибок возникает в связи с плохой различимостью символов. Так, например, часто путают буквы "i", "l", "I" и цифру "1". Путают букву "O" и цифру "0". Можно спутать букву "B" и цифру "8". Имея дело с этими символами относитесь к ним предельно осторожно.

Теперь о ходе отладки. Вы, конечно понимаете, что предусмотреть все возможные виды ошибок, которые Вы наделаете, заранее никто не сможет, хотя кое-какая статистика у нас имеется. Можно сказать, что до 90 процентов самых больших неприятностей связано с тремя видами ошибок:

Variable not found (переменная не найдена).
Out of screen (печать вне пределов экрана).
End of DATA (нехватка данных).

Что касается Variable not found, - это наиболее часто встречающаяся неприятность. К ней, как правило, и приводит путаница в символах, о которой мы писали выше. Хотя это, конечно, не единственный способ напороться на данную ошибку.

Когда компьютер сообщает об этой ошибке, он указывает и номер строки, в которой она произошла. Внимательно сверьте эту строку с исходным текстом. Если все в порядке, попробуйте распечатать переменные "подозрительного" выражения прямой командой. Та, которая ошибочна, распечатана не будет и Вы опять получите сообщение "Variable not found". Теперь надо найти программную строку, в которой эта переменная была в первый раз определена. Может быть эту строку Вы и пропустили или неправильно набрали переменную.

Иногда эта ошибка появляется не тогда, когда Вы что то пропустили, а наоборот набрали что-то лишнее, что было воспринято, как имя переменной.

Ошибка Out of screen возникает при печати (PRINT, PLOT, DRAW, CIRCLE), если координаты печати выходят за пределы экрана. Бороться с этой ошибкой весьма трудно, поскольку ошибка возникает в тех случаях, когда параметры координат позиции печати выражены не числом, а переменными, распечатайте их прямой командой PRINT.

Посмотрите, какой параметр "врет". Вернитесь по программе назад, найдите строку, в которой этот параметр вычисляется. Пусть это будет например строка 5550. Поставьте после нее отладочную строку:

```
5551 PRINT <имя перем.>: PAUSE 0
```

Так, постепенно "раскручивая" эту переменную, Вы и найдете в какой строке вычисления с ней портят Вашу программу.

Ошибка "Out of DATA" возникает в случае несоответствия количества данных, прочитанных программой по оператору READ и имеющихся в строках DATA. Борьба с ней довольно проста и требует только внимательности и точности.

А теперь, закончив с этим затянувшимся вступлением, мы предлагаем нашим читателям старинную африканскую игру "AFRICAN SEEDS". Игра относится к направлению, называемому "Манкала". В этом семействе имеются сотни разных игр. Выбранная нами игра отличается относительной простотой правил и несложной алгоритмизацией. Несмотря на свою простоту, компьютерная версия игры отличается тем, что победить ее непросто. Мы не будем здесь рассказывать правил игры, Вы их прочитаете непосредственно после запуска программы. Укажем только, что играть можно не только с компьютером. Воспользовавшись камешками или ракушками Вы можете играть с друзьями на пляже или с помощью сосновых шишек в лесу. В связи с приближением отпускного сезона, Вам это может пригодиться.

Программа переведена на русский язык. Если Ваш "Спектрум" имеет русский регистр, то набор русского текста не вызовет у Вас трудностей. Но, учитывая тот факт, что большинство компьютеров у наших пользователей нерусифицированные, мы использовали в программе загружаемый символьный набор "НС" в кодах КОИ-7 в стандарте ASCII (он приведен ниже). Это "утолщенный" русско-латинский символьный набор, предложенный для русификации программы "MF-09" (см. статью Алексеева А.Г. в "ZX-РЕВЮ" N 1-2 за 1992 г. стр. 29). Для тех читателей, которые подписались на "РЕВЮ" только в этом году, мы повторяем этот способ русификации. Мы будем всякий раз ссылаться на него при публикации в дальнейшем БЕЙСИК-программ для самостоятельного набора.

Для формирования символьного набора используется программа для шестнадцатиричного ввода (см. "ZX-РЕВЮ"-93, N 1-2, стр. 5-7).

СИМВОЛЬНЫЙ НАБОР "НС" КОИ-7

```
FC58:00 00 00 00 00 00 00 00 :54
FC60:00 18 18 18 18 00 18 00 :D4
FC68:00 6C 6C 6C 00 00 00 00 :A8
FC70:00 6C FE 6C 6C FE 6C 00 :18
FC78:00 18 7E 58 7E 1A 7E 18 :90
FC80:00 62 64 08 10 26 46 00 :C6
FC88:00 30 58 30 7A CC 76 00 :F8
FC90:00 18 30 00 00 00 00 00 :D4
FC98:00 0C 18 18 18 18 0C 00 :0C
FCA0:00 30 18 18 18 18 30 00 :5C
FCA8:00 00 6C 38 FE 38 6C 00 :EA
FCB0:00 00 18 18 7E 18 18 00 :8A
FCB8:00 00 00 00 00 18 18 30 :14
FCC0:00 00 00 00 7C 00 00 00 :38
FCC8:00 00 00 00 00 18 18 00 :F4
FCD0:00 00 06 0C 18 30 60 00 :86
FCD8:00 3C 66 6E 76 66 3C 00 :FC
FCE0:00 18 38 18 18 18 3C 00 :B0
FCE8:00 3C 66 06 3C 60 7E 00 :A6
FCF0:00 3C 66 0C 06 66 3C 00 :42
FCF8:00 0C 1C 2C 4C 7E 0C 00 :1E
FD00:00 7E 60 7C 06 66 3C 00 :FF
FD08:00 3C 60 7C 66 66 3C 00 :25
FD10:00 7E 66 0C 18 30 30 00 :75
FD18:00 3C 66 3C 66 66 3C 00 :FB
FD20:00 3C 66 66 3E 06 3C 00 :A5
FD28:00 00 18 18 00 18 18 00 :85
FD30:00 00 18 18 00 18 18 30 :BD
FD38:00 00 0C 18 30 18 0C 00 :AD
FD40:00 00 00 7C 00 7C 00 00 :35
FD48:00 00 30 18 0C 18 30 00 :E1
FD50:00 3C 66 0C 18 00 18 00 :2B
FD58:00 7C CE D6 DE C0 7C 00 :8F
FD60:00 3C 66 66 7E 66 66 00 :AF
FD68:00 7C 66 7C 66 66 7C 00 :0B
FD70:00 3C 66 60 60 66 3C 00 :71
FD78:00 7C 66 66 66 66 7C 00 :05
FD80:00 7E 60 7C 60 60 7E 00 :15
FD88:00 7E 60 7C 60 60 60 00 :FF
FD90:00 3C 66 60 6E 66 3C 00 :9F
FD98:00 66 66 7E 66 66 66 00 :11
FDA0:00 3C 18 18 18 18 3C 00 :75
FDA8:00 0E 06 06 66 66 3C 00 :C7
FDB0:00 66 6C 78 78 6C 66 00 :41
FDB8:00 60 60 60 60 60 7E 00 :13
FDC0:00 C6 EE D6 D6 C6 C6 00 :A9
```

```
FDC8:00 66 66 76 6E 66 66 00 :41
FDD0:00 3C 66 66 66 66 3C 00 :DD
FDD8:00 7C 66 66 7C 60 60 00 :59
FDE0:00 3C 66 66 6E 6E 3E 00 :FF
FDE8:00 7C 66 66 7C 6C 66 00 :7B
FDF0:00 3C 60 3C 06 66 3C 00 :60
FDF8:00 7E 18 18 18 18 18 00 :EB
FE00:00 66 66 66 66 66 3C 00 :38
FE08:00 66 66 66 66 3C 18 00 :F2
FE10:00 C6 C6 D6 D6 FE 44 00 :88
FE18:00 66 3C 18 18 3C 66 00 :8A
FE20:00 66 3C 18 18 18 18 00 :20
FE28:00 7C 0C 18 30 60 7C 00 :D2
FE30:00 1E 18 18 18 18 1E 00 :CA
FE38:00 C0 60 30 18 0C 06 00 :B0
FE40:00 78 18 18 18 18 78 00 :8E
FE48:00 18 3C 5A 18 18 18 00 :3C
FE50:00 00 00 00 00 00 00 FF :4D
FE58:00 00 DC F6 F6 F6 DC 00 :F0
FE60:00 00 3C 66 7E 66 66 00 :4A
FE68:00 00 7C 60 7C 66 7C 00 :A0
FE70:00 00 6C 6C 6C 6C 7E 06 :A2
FE78:00 00 3C 6C 6C 6C FE C6 :BA
FE80:00 00 7E 60 7C 60 7E 00 :B6
FE88:00 00 7C D6 D6 7C 10 00 :3A
FE90:00 00 3E 30 30 30 30 00 :8C
FE98:00 00 66 3C 18 3C 66 00 :F2
FEA0:00 00 66 66 6E 76 66 00 :B4
FEA8:00 18 66 66 6E 76 66 00 :D4
FEB0:00 00 66 6C 78 6C 66 00 :CA
FEB8:00 00 1E 36 36 36 66 00 :DC
FEC0:00 00 C6 EE D6 C6 C6 00 :D4
FEC8:00 00 66 66 7E 66 66 00 :DC
FED0:00 00 3C 66 66 66 3C 00 :78
FED8:00 00 7E 66 66 66 66 00 :EC
FEE0:00 00 3E 66 3E 36 66 00 :5C
FEE8:00 00 7C 66 66 7C 60 00 :0A
FEF0:00 00 3C 66 60 66 3C 00 :92
FEF8:00 00 7E 18 18 18 18 00 :D4
FF00:00 00 66 66 3E 06 3C 00 :4B
FF08:00 00 D6 D6 7C D6 D6 00 :DB
FF10:00 00 7C 66 7C 66 7C 00 :4F
FF18:00 00 60 60 7C 66 7C 00 :35
FF20:00 00 C6 C6 F6 DE F6 00 :75
FF28:00 00 3C 66 0C 66 3C 00 :77
FF30:00 00 C6 D6 D6 D6 FE 00 :75
FF38:00 00 78 0C 3C 0C 78 00 :7B
FF40:00 00 C6 D6 D6 D6 FF 03 :89
FF48:00 00 66 66 3E 06 06 00 :5D
FF50:00 00 70 30 3C 36 3C 00 :9D
```

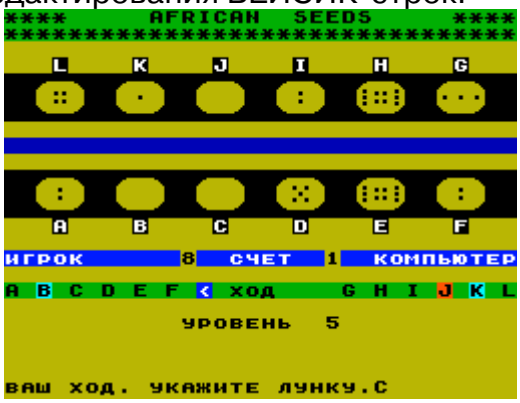
После того, как работа по вводу этих кодов будет завершена, запишите полученный символьный набор на магнитную ленту: SAVE "chr" CODE 61600,768. Он пригодится Вам и для других программ.

Перед тем, как набирать текст БЕЙСИК-программы, сделайте следующее:

```
CLEAR 64599: LOAD "chr"CODE
POKE 23606,88: POKE 23607,251
```

Теперь включен загруженный символьный набор. Попробуйте набирать БЕЙСИК программу. Токены ключевых слов "Спектрума", состоящие из заглавных латинских букв, будут печататься как обычно. Текст, набираемый при помощи буквенных клавишей при отключенном режиме CAPS LOCK, будет набираться русскими буквами. Так же он будет выглядеть и на экране. При включений регистра CAPS LOCK, текст будет набираться латинскими буквами. При этом, правда, придется обходиться только заглавными русскими и латинскими буквами, но зато это позволит легко совмещать русский и английский текст внутри одного оператора PRINT. Имена переменных для удобства чтения набираются заглавными латинскими буквами.

Кроме формирования самого символьного набора, в программу вводятся дополнительные строки. Это строка 2, загружающая символьный набор. С нее должен происходить автостарт программы. Еще это переключатели шрифта, строки 8 и 9 (могут быть любые другие номера, такие, какие удобно вписываются в программу). Используя команду GO SUB 8, можно включить загруженный символьный набор, а командой GO SUB 9 - выключить его. Эти команды можно как включать в БЕЙСИК-строки, так и подавать напрямую, находясь в режиме редактирования БЕЙСИК-строк.



Следует также предупредить тех пользователей, которые имеют БЕТА-ДИСК

интерфейс. В программах, в которых используется блок UDG-графики, стандартно расположенный после рестарта компьютера с адреса 65368, полезно бывает перед стартом программы включать еще и конструкцию, задающую системную переменную UDG:

```
POKE 23675,88: POKE 23676,255
```

Это, казалось бы лишнее, так как после рестарта компьютера в указанные ячейках и так находятся указанные числа. Дело в том, что значения этих ячеек могли быть изменены в процессе работы дискового загрузчика "boot". Так некорректно ведет себя, например, в общем-то очень хороший и широко распространенный загрузчик "MOA-SERVICE". Эта рекомендация убережет Вас от возможных "накладок" и в других случаях.

А теперь, когда есть определенность с вопросом русификации, можно переходить непосредственно к тексту программы "AFRICAN SEEDS".

```
1 GO TO 1000
2 CLEAR 64599:LOAD "CHR" CODE 64600
3 POKE 23675,88:POKE 23676,255:RESTORE 2100:FOR P=0 TO 95:READ G:POKE USR "A"+P,G:NEXT P
4 RUN
5 SAVE "AFRICA"LINE 2:SAVE "CHR" CODE 64600,768:STOP
8 POKE 23606,88:POKE 23607,251:RETURN:REM RUS
9 POKE 23606,0:POKE 23607,60:RETURN:REM LAT
100 PRINT AT 17, CODE A$(H);BRIGHT 1;PAPER 2;SCREEN$(17, CODE A$(H)):LET S=H(H):LET
      H(H)=0:LET D=D+1
110 IF S THEN LET S=S-1:LET H=H+1-12*(H=12):LET H(H)=H(H)+1:GO TO 110
200 PRINT AT 17, CODE A$(H);BRIGHT 1;INK 0;PAPER 2;SCREEN$(17, CODE A$(H)):IF H(H)<>1 THEN
      FOR B=10 TO 10+H(H):BEEP .002,B+D:NEXT B:RETURN
210 PRINT AT 17, CODE A$(H);BRIGHT 1;PAPER 5;SCREEN$(17, CODE A$(H));:LET END=1:LET OP=13-
      H:FOR B=20 TO 20+H(OP):BEEP .02,B:NEXT B:IF (NOT ZX AND OP<7) OR (ZX AND OP>6) OR NOT
      H(OP) THEN RETURN
220 LET WIN=H(OP):LET H(OP)=0:LET S(2-ZX)=S(2-ZX)+WIN*SC:PRINT AT 17, CODE A$(OP);FLASH
      1;BRIGHT 1;PAPER 5;SCREEN$(17, CODE A$(OP)):FOR B=30 TO 30+WIN:BEEP .025,B:NEXT
      B:RETURN
300 PRINT AT 19,25;"       ";AT 19,11;BRIGHT 1;PAPER (D-8*INT (D/8));INK 9;" ХОД  ";D;"
      ";:FOR P=1 TO 12:RESTORE 2300+B(P):READ N$
330 PRINT AT  CODE X$(P), CODE Y$(P);N$:LET H(P)=B(P):BEEP .005,D+B(P):NEXT P
340 PRINT AT 15,11;S(2);:PRINT AT 15,20;S(1):PRINT AT 19,11;L$:RETURN
400 BRIGHT 0:PRINT AT 17,0;PAPER 4;"A B C D E F   ХОД    G H I J K L":RETURN
500 PRINT ("ЧИСЛО ПРОИГРАННЫХ ВАМИ ОЧКОВ: "+STR$ (S(1)-S(2)) AND S(1)>S(2))+("ЧИСЛО
      ВЫИГРАННЫХ ВАМИ ОЧКОВ: "+STR$ (S(2)-S(1)) AND S(1)<S(2))
510 GO SUB 400
520 PRINT AT 0,0:INPUT "ХОТИТЕ СЫГРАТЬ ЕЩЕ РАЗ? (Y) ";LINE U$:IF  CODE U$=121-32*(U$="Y")
      THEN GO TO 1040
530 STOP
580 PRINT AT 18,0;"ЛУНКИ ПРОТИВНИКА ПУСТЫ.  ВСЕ ОС-ТАВШИЕСЯ ФИШКИ ПЕРЕШЛИ К НЕМУ."
600 PRINT AT 20,0;"ИГРА ";("ПРОДОЛЖАЕТСЯ." AND S(1)=S(2));("ЗАКОНЧЕНА." AND S(1)<>S(2)):GO
      TO 500
700 IF S(1)>24 OR S(2)>24 THEN GO TO 600
710 IF S(1)+S(2)>39 AND NOT WIN THEN LET C=C+1:IF C>10 THEN GO TO 600
720 RETURN
1000 GO SUB 8:POKE 23658,8:LET D=0:DIM S(2):DIM B(12):DIM H(12):DIM A$(12):DIM C$(12):DIM
      X$(12):DIM Y$(12)
1020 GO SUB 2000:PRINT AT 19,1;INK 0;"ВЫ БУДЕТЕ СМОТРЕТЬ ИНСТРУКЦИЮ?";AT 21,4;"(ЕСЛИ ДА, ТО
      ВВЕДИТЕ Y)":INPUT LINE U$:IF  CODE U$+32*(U$="Y")=121 THEN
GO SUB 2400
1030 RESTORE 2160:FOR N=1 TO 12:READ A:LET A$(N)=CHR$ A:NEXT N
1040 GO SUB 2020
1060 GO SUB 2080:INPUT LINE U$:IF U$<"0" OR U$>"9" THEN GO TO 1060
1070 LET L=VAL U$+20*(U$="0"):PRINT AT 15,0;"                    ":FOR P=15 TO 21 STEP
      2:PRINT AT P,17;"            ":NEXT P:LET L$="УРОВЕНЬ  "+U$+" "
1080 RESTORE 3000:FOR N=1 TO 12:READ X,Y:PRINT AT X,Y;"   ";AT X+2-4*(N>6),Y+1;INK 7;PAPER
      0;BRIGHT 1;CHR$ (64+N):LET X$(N)=CHR$ X:LET Y$(N)=CHR$ Y:NEXT N
1090 PRINT AT 15,0;BRIGHT 1;PAPER 1;INK 7;"ИГРОК         СЧЕТ     КОМПЬЮТЕР"
1100 DIM M(12):LET D=0:LET T=0:LET SC=0:LET C=0:LET S(1)=0:LET S(2)=0:FOR P=1 TO 12:LET
      B(P)=4:LET H(P)=4:NEXT P
1110 RANDOMIZE :LET ZX=INT (RND *2)
```

```
1120 GO SUB 300:GO SUB 400
1150 IF ZX THEN PRINT AT 17,19;BRIGHT 1;INK 7;PAPER 1;">":INPUT "ПЕРВЫЙ ХОД ВЫПАЛ
      КОМПЬЮТЕРУ.    НАЖМИТЕ ENTER, ЕСЛИ ГОТОВЫ. ";LINE U$:LET PREF=INT (RND *7+6):GO TO
      1400
1200 IF NOT ZX THEN GO TO 1500
1210 GO SUB 700
1220 IF NOT (B(7)+B(8)+B(9)+B(10)+B(11)+B(12)) THEN LET S(1)=S(1)+48-S(1)-S(2):GO TO 580
1230 LET OP=0:LET SC=0:LET WIN=0:DIM M(12)
1250 PRINT AT 17,19;BRIGHT 1;INK 7;PAPER 1;">":INPUT "НАЖМИТЕ ENTER, ПОЖАЛУЙСТА.";LINE U$:GO
      SUB 400:IF U$=" STOP " THEN GO TO 600
1300 PRINT AT 19,26;"АНАЛИЗ":FOR P=7 TO 12:LET D=0:LET END=0:LET WIN=0:LET OP=0:IF NOT B(P)
      THEN GO TO 1350
1320 LET H=P:LET M=P:FOR K=1 TO 12:LET H(K)=B(K):NEXT K
1330 GO SUB 400:GO SUB 100:IF D<1 AND NOT END THEN GO TO 1330
1340 LET M(P)=WIN*END
1350 NEXT P
1360 LET PREF=7:PRINT AT 19,26;"ОЦЕНКА":FOR P=7 TO 12:IF M(P)>M(PREF) THENLET PREF=P
1370 NEXT P
1380 LET B=0:FOR P=7 TO 12:IF P<>PREF THEN IF B(P)>0 THEN IF M(P)=M(PREF) THEN LET B=B+1:LET
      M(B)=P
1390 NEXT P:IF B THEN LET PREF=M(INT (RND *B)+1)
1400 LET END=0:LET D=0:LET OP=0:LET SC=1:LET H=PREF:LET WIN=0:GO SUB 400:FOR P=1 TO 12:LET
      H(P)=B(P):NEXT P
1410 PRINT AT 17, CODE A$(H);BRIGHT 1;PAPER 2;SCREEN$(17, CODE A$(H)):GO SUB 100
1440 FOR P=1 TO 12:LET B(P)=H(P):NEXT P:GO SUB 300:IF NOT END THEN GO SUB 400:GO TO 1410
1450 GO SUB 700
1490 IF NOT (B(1)+B(2)+B(3)+B(4)+B(5)+B(6)) THEN LET S(2)=S(2)+48-S(1)-S(2):GO TO 580
1500 PRINT AT 17,12;BRIGHT 1;INK 7;PAPER 1;"<":POKE 23658,8:INPUT "ВАШ ХОД. УКАЖИТЕ
      ЛУНКУ.";LINE U$:IF U$=" STOP " THEN GO TO 600
1540 LET H= CODE U$-64-32*(U$>"Z"):GO SUB 400:IF H<0 OR H>6 THEN PRINT #0;"ЗАПРЕЩЕННЫЙ
      ХОД.""ПОВТОРИТЕ, ПОЖАЛУЙСТА.":BEEP 1.5,0:GO TO 1500
1550 LET D=0:LET ZX=0:LET SC=1:LET WIN=0:LET OP=0:IF NOT B(H) THEN GO TO 1500
1560 PRINT AT 17, CODE A$(H);BRIGHT 1;PAPER 2;SCREEN$(17, CODE A$(H)):GO SUB 100
1570 FOR P=1 TO 12:LET B(P)=H(P):NEXT P:IF NOT OP THEN GO SUB 300:GO SUB 400:GO TO 1560
1580 GO SUB 300:LET ZX=1
1590 GO TO 1200
2000 BORDER 4:PAPER 4:INK 2:CLS:PRINT ''                    "
      ▄▄▄▄ ▄▄▄▄  ▄▄▄  ▌ ▄▄▄ ▄▄▄▄ ▄▄  ▐
      ▄▄ ▌ ▄▄   ▐▌ ▌ ▌ ▄▄  ▄▄ ▌ ▄▄▄▄ ▌
      ▄▄▄▄ ▄▄▄   ▄▄▄▄ ▌ ▄▄  ▄▄▄▄ ▄▄ ▄▄▄
      ▌  ▌ ▄▄    ▌  ▌ ▌ ▄▄▄ ▌  ▌ ▄▄  ▄▄"
2005 PRINT ''                "
        ▄▄▄▄  ▄▄▄▄  ▄▄▄▄  ▄▄▄▄  ▄▄▄▄
        ▌    ▄▄   ▄▄   ▄▄ ▌  ▌
        ▄▄▄  ▄▄▄  ▄▄▄  ▄▄ ▌  ▄▄▄
          ▄▄  ▌   ▌    ▄▄ ▌    ▄▄
        ▄▄▄▄  ▄▄▄▄  ▄▄▄▄  ▄▄▄▄  ▄▄▄▄   "
2010 PRINT AT 16,3;INK 0;"MIKE EDWARDS + ""ИНФОРКОМ""":RETURN
2020 BORDER 6:PAPER 6:INK 0:CLS:PRINT PAPER 4;"****     AFRICAN  SEEDS
      **********************************"
2030 PRINT '                 "
      ▄▄J▀I▄▄J▀I▄▄J▀I▄▄J▀I▄▄J▀I▄▄J▀I▄▄
      ▄▄   ▄▄   ▄▄   ▄▄   ▄▄   ▄▄   ▄▄
      ▄▄K▄L▄▄K▄L▄▄K▄L▄▄K▄L▄▄K▄L▄▄K▄L▄▄"
2040 PRINT PAPER 1'"                          "
2050 PRINT '                 "
      ▄▄J▀I▄▄J▀I▄▄J▀I▄▄J▀I▄▄J▀I▄▄J▀I▄▄
      ▄▄   ▄▄   ▄▄   ▄▄   ▄▄   ▄▄   ▄▄
      ▄▄K▄L▄▄K▄L▄▄K▄L▄▄K▄L▄▄K▄L▄▄K▄L▄▄"
2060 RETURN
```

```
2080 PRINT AT 15,0;"ВВЕДИТЕ УРОВЕНЬ: 1-3=НОВИЧОК"''TAB17;"4-6=ЛЕГКИЙ"''TAB17;"7-
     9=ТЯЖЕЛЫЙ"''TAB19;"0=ЭКСПЕРТ"
2090 RETURN
2100 DATA 0,0,0,24,24,0,0,0
2101 DATA 0,24,24,0,0,24,24,0
2102 DATA 24,24,0,24,24,0,24,24
2103 DATA 0,102,102,0,0,102,102,0
2104 DATA 195,195,0,24,24,0,195,195
2105 DATA 102,102,0,102,102,0,102,102
2106 DATA 102,102,0,219,219,0,102,102
2107 DATA 219,219,0,102,102,0,219,219
2108 DATA 255,255,255,255,31,7,3,1
2109 DATA 255,255,255,255,248,224,192,128
2110 DATA 128,192,224,248,255,255,255,255
2111 DATA 1,3,7,31,255,255,255,255
2160 DATA 0,2,4,6,8,10,21,23,25,27,29,31
2300 DATA "   "
2301 DATA " A "
2302 DATA " B "
2303 DATA "AAA"
2304 DATA " D "
2305 DATA " E "
2306 DATA "BBB"
2307 DATA "BCB"
2308 DATA "BDB"
2309 DATA "CCC"
2310 DATA "CDC"
2311 DATA "CEC"
2312 DATA "DDD"
2313 DATA "DED"
2314 DATA "EDE"
2315 DATA "FCF"
2316 DATA "FDF"
2317 DATA "GCG"
2318 DATA "FFF"
2319 DATA "HCH"
2320 DATA "HDH"
2321 DATA "GGG"
2322 DATA "HFH"
2323 DATA "HGH"
2324 DATA "HHH"
2325 DATA " 25"
2326 DATA " 26"
2327 DATA " 27"
2328 DATA " 28"
2329 DATA " 29"
2330 DATA " 30"
2400 BORDER 4:PAPER 4:INK 0:CLS:PRINT "****     AFRICAN  SEEDS
     **********************************"
2410 PRINT ''"
       ИГРА ОСНОВАНА  НА  АФРИКАНСКИХ
     ИГРАХ ТИПА  ""МАНКАЛА"", ХОТЯ ЕЕ
     ПРАВИЛА ПРОЩЕ, ЧЕМ У БОЛЬШИНСТВА
     ЕЕ ПРЕДШЕСТВЕННИЦ."
2420 PRINT '"
       ИМЕЕТСЯ  12 ЛУНОК  С  ЧЕТЫРЬМЯ
     ФИШКАМИ КАЖДАЯ. ОНИ ПОРОВНУ РАЗ-
     ДЕЛЕНЫ МЕЖДУ ИГРОКАМИ. ВАША ЦЕЛЬ
     - НАБРАТЬ ОЧКИ,  СОБИРАЯ КАК МО-
     ЖНО БОЛЬШЕ ФИШЕК.  25 ОЧКОВ ОБЕ-
     СПЕЧИВАЕТ ПОБЕДУ."
2430 INPUT "ДЛЯ ПРОДОЛЖЕНИЯ НАЖМИТЕ ENTER ";LINE U$
2440 BORDER 4:PAPER 4:INK 0:CLS:PRINT "****     AFRICAN  SEEDS
     **********************************"
```

```
2450 PRINT ''"
        ИГРОКИ ПО ОЧЕРЕДИ БЕРУТ  ФИШКИ
      ИЗ ЛЮБОЙ СВОЕЙ ЛУНКИ И РАССЫПАЮТ
      ИХ ПО  ОДНОЙ  В  СОСЕДНИЕ  ЛУНКИ
      ПРОТИВ ЧАСОВОЙ СТРЕЛКИ,  НАЧИНАЯ
      С ЛУНКИ, НАХОДЯЩЕЙСЯ  ПО-СОСЕД-
      СТВУ С НАЧАЛЬНОЙ."
2460 PRINT '"
        ЕСЛИ ПОСЛЕДНЯЯ ФИШКА  ОКАЖЕТСЯ
      В ЗАНЯТОЙ ЛУНКЕ, ТО ИГРОК  БЕРЕТ
      ФИШКИ ИЗ НЕЕ И ПРОДОЛЖАЕТ ХОД  В
      ТОМ ЖЕ ПОРЯДКЕ.  ТАК ПРОДОЛЖАЕТ-
      СЯ, ПОКА ХОД НЕ ЗАВЕРШИТСЯ В ПУ-
      СТОЙ ЛУНКЕ."
2470 PRINT '"
        ЕСЛИ ХОД ЗАКАНЧИВАЕТСЯ В  ПУС-
      ТОЙ ЛУНКЕ ПРОТИВОПОЛОЖНОГО РЯДА,
      ИЛИ НАПРОТИВ  ПУСТОЙ  ЛУНКИ,  ТО
      ХОД ПЕРЕХОДИТ К ПРОТИВНИКУ."
2480 INPUT "ДЛЯ ПРОДОЛЖЕНИЯ НАЖМИТЕ ENTER ";LINE U$
2490 BORDER 4:PAPER 4:INK 0:CLS:PRINT "****     AFRICAN  SEEDS
      **********************************"
2500 PRINT ''"
        ЗАХВАТ ФИШЕК  ВОЗМОЖЕН  ТОЛЬКО
      ЕСЛИ ПОСЛЕДНЯЯ ФИШКА В ХОДЕ  ПО-
      ПАДАЕТ В ПУСТУЮ ЛУНКУ СВОЕГО РЯ-
      ДА, НАПРОТИВ ЗАНЯТОЙ ЛУНКИ  ПРО-
      ТИВНИКА. ТОГДА ФИШКИ ПРОТИВНИКА
      ИЗЫМАЮТСЯ, А СЧЕТ УВЕЛИЧИВАЕТСЯ."
2510 PRINT '                 "
        ИГРА ПРОДОЛЖАЕТСЯ,  ПОКА  ОДИН
      ИЗ ИГРОКОВ НЕ НАБЕРЕТ  БОЛЕЕ  24
      ОЧКОВ ИЛИ ОБА ПАРТНЕРА НЕ СОГЛА-
      СЯТСЯ НА НИЧЬЮ."
2520 PRINT '                   "
        ВНИМАНИЕ!   ЕСЛИ  ВЫ  ОСТАВИТЕ
      СВОЕГО ПРОТИВНИКА  БЕЗ  ФИШЕК  И
      БЕЗ ХОДА, ТО ВСЕ ФИШКИ, ИМЕЮЩИЕ-
      СЯ НА ДОСКЕ, ПЕРЕХОДЯТК НЕМУ."
2530 INPUT "ДЛЯ ПРОДОЛЖЕНИЯ НАЖМИТЕ ENTER ";LINE U$
2540 BORDER 4:PAPER 4:INK 0:CLS:PRINT "****     AFRICAN  SEEDS
      **********************************"
2550 PRINT ''                  "
        ПРАВО ПЕРВОГО ХОДА  ОПРЕДЕЛЯЕТ
      КОМПЬЮТЕР СЛУЧАЙНЫМ ОБРАЗОМ."''"
        ВЫ ИМЕЕТЕ ЛУНКИ ОТ A ДО F. КОМ
      ПЬЮТЕР ИМЕЕТ ЛУНКИ ОТ G ДО L."
2560 PRINT '                   "
        КОГДА БУДЕТ ВАШ  ХОД,  ВВЕДИТЕ
      СИМВОЛ ТОЙ ЛУНКИ, ИЗ КОТОРОЙ  ВЫ
      ХОТИТЕ  СДЕЛАТЬ  ХОД.  КОМПЬЮТЕР
      ЖДЕТ НАЖАТИЯ  ENTER  ДЛЯ  НАЧАЛА
      СВОЕГО ХОДА."
2570 PRINT '                   "
        КОГДА  ИГРА  БУДЕТ  ЗАВЕРШЕНА,
      КОМПЬЮТЕР ПОДСЧИТАЕТ  И  ОБЪЯВИТ
      РЕЗУЛЬТАТ. МОЖНО  ПРЕРВАТЬ  ИГРУ
      В ЛЮБОЕ ВРЕМЯ, ВВЕДЯ: "" STOP "".""
2580 INPUT                    "
      ВВЕДИТЕ R ДЛЯ ПОВТОРА ИНСТРУКЦИИ
      ИЛИ ENTER, ЧТОБЫ НАЧАТЬ ИГРУ. ";LINE 3$:IF  CODE 3$=114-32*(3$="R") THEN GO TO 2400
2590 RETURN
3000 DATA 11,2,11,7,11,12,11,17,11,22,11,27,5,27,5,22,5,17,5,12,5,7,5,2
```

Набирая программу, Вы периодически можете записывать результат на магнитную ленту, выполняя: RUN 5. При наборе программы обратите внимание, пожалуйста, на те места, где набор происходит с использованием графического регистра. Это строки 2000-2005, в которых используется символ с кодом 143 (регистр [G], затем CS+8). Это также символы с кодами 131, 140, 143 блочной графики и символы "I"-"L" UDG-графики в строках 2030, 2050. Символы UDG-графики "A"-"H" используются при наборе строк 2300-2324.

Автостарт программы происходит со строки 2, где прежде всего резервируется место дли расположения загружаемого символьного набора, путем переустановки значения RAMTOP оператором CLEAR. После загрузки символьного набора происходит задание банка символов UDG-графики. На этом подготовку можно считать законченной. Далее строка 4 передает управление на начало программы. При отладке, после остановки программы, ее можно запустить командой RUN с начальной строки.

Так как мы используем русско-латинский символьный набор, то перед тем, как будет произведен ввод параметров при помощи оператора INPUT, корректно будет принудительно включать режим CAPS LOCK (устанавливать курсор [C]), так как вводиться должны параметры в виде английских букв. Это происходит, например, в начале выполнения программы, в строке 1000 подачей команды POKE 23658,6, а также в других местах.

Подпрограмма GO SUB 2000 в строке 1020 - это вывод на экран титульной заставки. После этого происходит запрос для вывода инструкции. Результат запроса (переменная U$) анализируется, причем результат не зависит от того, нажата ли клавиша "Y" в режиме CAPS LOCK или нет. В том случае, если нажата клавиша "Y", происходит вызов подпрограммы GO SUB 2400, которая выводит на экран инструкцию с правилами игры.

Подпрограмма GO SUB 2020 в строке 1040 обеспечивает прорисовку игрового поля - игровые лунки играющего и компьютера.

В строке 1060 происходит ввод уровня сложности игры. Кстати, при тестировании программы выяснилось, что эта величина нигде дальше в программе не участвует. Непонятно, для чего автор ввел этот параметр. Может быть, рассчитывая в дальнейшем на усовершенствование игры, или на то, что "пытливые умы" сами сделают такое усовершенствование. В общем, есть простор для творческого поиска. А можно пойти по другому пути: просто исключить все, что касается ввода уровня. Но нам кажется, что вариант с заданием уровня (пусть фиктивным) смотрится динамичнее.

Далее идет подготовка игрового поля и предустановка исходных данных. В строке 1110 происходит определение того, кто делает первый ход. Если право первого хода выпало компьютеру, тогда после нажатия ENTER программа переходит на строку 1400. Кстати, свой первый ход компьютер делает случайным образом выбирая одну из своих лунок. Это происходит путем задания переменной PREF в строке 1150. Но это касается только первого хода. В дальнейшем, перед тем, как сделать ход, компьютер рассчитывает наиболее удачный вариант. Это осуществляется в строках, начиная с 1200. В этой строке, если право очередного хода имеет человек, то происходит переход на строку 1500. Если ход принадлежит компьютеру, то ожидается нажатие ENTER, после чего ситуация анализируется (со строки 1300) а затем, в строке 1360, выбирается наиболее предпочтительный вариант PREF. Далее, со строки 1400 происходят действия, непосредственно связанные с выполнением кода.

Если очередной ход делает человек, то происходит переход на строку 1500. На эту строку мы попадаем сразу после того, как будет закончен ход компьютера (в строках с 1400) или в начале игры, в случае первого хода, (см. строки 1110-1200). В конце этой части программы происходит передача хода компьютеру (строка 1580) и повторение цикла со строки 1200.

И, в заключение, предлагаем еще одну игру для тех, кто любит программы, развивающие внимательность, наблюдательность и быстроту реакции.

CASH-FLOW

Игра имитирует игровой автомат. Монета падает вниз, ударяясь о неподвижные штыри. Таким образом, путь монеты в значительной мере случаен. В нижней части игрового поля монета попадает в один из шести монетоприемников, в каждом из которых уже может находиться до четырех монет. Ваша задача - довести их количество до пяти. Тогда монеты будут выданы Вам в качестве приза (разумеется, при наличии у вашего "Спектрума" соответствующей приставки для выдачи "наличных"). Выигранные монеты можно использовать для продолжения игры. Результат игры в основном зависит от момента, когда Вы бросаете монету. В то же время, у Вас есть возможность подтолкнуть падающую монету в любом направлении, но только один раз и только в верхней части доски со штырями. Монетоприемник N 4 "проглатывает" все монеты, поэтому его следует избегать всеми силами. Всего на игру Вам дается 6 монет по 10 пенсов каждая. Постарайтесь как можно раньше получить выигрыш и продолжать игру как можно дольше.


```
1 GO TO 10
2 CLEAR 64599:LOAD "CHR" CODE 64600
3 POKE 23606,88:POKE 23607,251
4 GO SUB 3000:RUN
5 SAVE "CASH-FLOW"LINE 2:SAVE "CHR" CODE 64600,768:STOP
10 POKE 23658,8:BORDER 5:PAPER 6:INK 1
15 LET OW=0
20 LET A$="AB":LET B$="CD"
25 GO SUB 2500
30 CLS
35 FOR N=0 TO 21:DIM I$(26):PRINT PAPER 6;AT N,3;I$:DIM J$(2):PRINT PAPER 5;AT N,0;J$;AT
      N,30;J$:NEXT N
40 FOR N=0 TO 192 STEP 32
45 CIRCLE 32+N,77,2:BEEP .05,10:CIRCLE 32+N,109,2:BEEP .05,20:CIRCLE 32+N,141,2:BEEP .05,30
50 NEXT N
55 FOR N=0 TO 160 STEP 32
60 CIRCLE 48+N,93,2:BEEP .05,40:CIRCLE 48+N,125,2:BEEP .05,20
65 NEXT N
70 DEF FN T()=INT ((65536*PEEK 23674+256*PEEK 23673+PEEK 23672)/50)
72 LET T1=FN T()
75 FOR N=0 TO 21
80 PRINT INK 4;AT N,2;"F";AT N,29;"E"
85 IF N>12 THEN PRINT INK 2;AT N,3;"EF  EF  EF  EF  EF  EF  EF"
90 NEXT N
95 PLOT INK 3;24,159:DRAW INK 3;207,0
100 LET W=0:LET L=0
120 GO TO 1000
130 PRINT AT X,Y;"  ";AT X+1,Y;"  "
135 IF INKEY$="Q" AND L>40 THEN LET NW=W-L:GO TO 2000
140 LET K=2*COS (PI*(INT (RND *2)))
150 LET X=X+2
151 IF I=0 AND M<4 AND Y<=25 AND INKEY$="P" THEN BEEP .05,32:LET Y=Y+2:LET I=1:GO TO 160
152 IF I=0 OR M<4 AND Y>=5 AND INKEY$="O" THEN BEEP .05,32:LET Y=Y-2:LET I=1:GO TO 160
154 IF K>0 AND Y=27 THEN LET Y=Y-K:GO TO 160
155 IF K>0 AND Y<26 THEN LET Y=Y+K
```

```
157 IF K<0 AND Y=3 THEN LET Y=Y-K:GO TO 160
158 IF K<0 AND Y>4 THEN LET Y=Y+K
160 PRINT INK 1;AT X,Y;A$;AT X+1,Y;B$
165 BEEP .05,20
166 FOR N=1 TO 100:NEXT N
170 LET M=M+1
180 IF M=6 THEN GO TO 500
190 GO TO 130
500 IF Y=5 THEN LET G=A:LET A=A+1:GO TO 530
505 IF Y=9 THEN LET G=B:LET B=B+1:GO TO 530
510 IF Y=13 THEN LET G=C:LET C=C+1:GO TO 530
515 IF Y=17 THEN LET G=0:GO TO 530
520 IF Y=21 THEN LET G=E:LET E=E+1:GO TO 530
525 IF Y=25 THEN LET G=F:LET F=F+1
530 IF G=4 THEN GO TO 600
540 FOR N=0 TO 3-G:PRINT INK 1;AT 12+2*N,Y;"  ";AT 13+2*N,Y;"  ";AT 14+2*N,Y;A$;AT
      15+2*N,Y;B$
550 PAUSE 20
560 NEXT N
565 IF Y=17 THEN PRINT AT 20,17;"  ";AT 21,17;"  ":BEEP 1,0
568 IF CR<=0 THEN GO TO 2000
570 PRINT INK 2;FLASH 1;AT 1,20;"НАЖМИТЕ A"
575 IF INKEY$="A" THEN PRINT INK 2;FLASH 1;AT 1,28;"Z":GO TO 1100
576 LET T=FN T():IF T>T1+240 THEN GO TO 1990
578 IF INKEY$="Q" AND L>40 THEN LET NW=W-L:GO TO 2000
580 GO TO 575
600 BEEP 1,40
604 FOR N=0 TO 4
606 LET W=W+10:LET NW=W-L:LET CR=NW+60
608 PRINT AT 12+2*N,Y;"  ";AT 13+2*N,Y;"  ":BEEP .1,30:PAUSE 20
610 PRINT AT 0,15;"    "
612 PRINT BRIGHT 1;FLASH 1;AT 0,15;CR
615 NEXT N
618 IF CR<=0 THEN GO TO 2000
620 IF Y=5 THEN LET A=0
621 IF Y=9 THEN LET B=0
622 IF Y=13 THEN LET C=0
624 IF Y=21 THEN LET E=0
625 IF Y=25 THEN LET F=0
630 PRINT INK 2;FLASH 1;AT 1,20;"НАЖМИТЕ A"
635 IF INKEY$="A" THEN PRINT INK 2;FLASH 1;AT 1,28;"Z":GO TO 1100
638 IF INKEY$="Q" AND L>40 THEN GO TO 2000
640 GO TO 635
1000 LET A=INT (RND *3):FOR N=1 TO A:IF A<>0 THEN PRINT INK 1;AT 23-2*N,5;B$;AT 22-
      2*N,5;A$:BEEP .05,0:NEXT N
1010 LET B=INT (RND *3):FOR N=1 TO B:IF B<>0 THEN PRINT INK 1;AT 23-2*N,9;B$;AT 22-
      2*N,9;A$:BEEP .05,10:NEXT N
1020 LET C=INT (RND *5):FOR N=1 TO C:IF C<>0 THEN PRINT INK 1;AT 23-2*N,13;B$;AT 22-
      2*N,13;A$:BEEP .05,20:NEXT N
1040 LET E=INT (RND *5):FOR N=1 TO E:IF E<>0 THEN PRINT INK 1;AT 23-2*N,21;B$;AT 22-
      2*N,21;A$:BEEP .05,30:NEXT N
1050 LET F=INT (RND *3):FOR N=1 TO F:IF F<>0 THEN PRINT INK 1;AT 23-2*N,25;B$;AT 22-
      2*N,25;A$:BEEP .05,30:NEXT N
1055 LET CR=60:PRINT BRIGHT 1;FLASH 1;AT 0,15;CR
1060 PRINT INK 2;FLASH 1;AT 1,20;"НАЖМИТЕ A"
1065 IF INKEY$="A" THEN PRINT INK 2;FLASH 1;AT 1,28;"Z":GO TO 1100
1068 IF INKEY$="Q" AND L>40 THEN GO TO 2000
1070 GO TO 1065
1100 LET L=L+10:LET NW=W-L:LET CR=NW+60:LET I=0
1101 PRINT AT 0,15;"    "
1102 PRINT BRIGHT 1;FLASH 1;AT 0,15;CR
1104 PRINT INK 1;AT 0,3;A$;AT 1,3;B$:PAUSE 6
1105 FOR N=3 TO 27
1110 PRINT AT 0,N;"  ";AT 1,N;"  "
1115 IF N=27 AND CR<=0 THEN GO TO 2000
```

```
1120 IF N=27 THEN GO TO 1060
1130 PRINT INK 1;AT 0,N+1;A$;AT 1,N+1;B$
1140 PAUSE 6
1150 IF INKEY$="Z" AND INT (N/2)<>INT ((N-1)/2) THEN LET Y=N+1:PRINT AT 1,20;"         ":GO
     TO 1200
1160 NEXT N
1200 LET X=0:LET M=0
1210 PRINT AT X,Y;"  ";AT X+1,Y;"  "
1212 PRINT AT 0,15;"    "
1215 PRINT BRIGHT 1;FLASH 1;AT 0,15;CR
1220 LET X=X+2:LET M=M+1
1230 PRINT INK 1;AT X,Y;A$;AT X+1,Y;B$
1240 BEEP .05,20:FOR N=1 TO 50:NEXT N
1250 IF INT ((Y+1)/4)=INT ((Y+3)/4) THEN GO TO 1300
1260 IF M=2 THEN GO TO 1300
1270 GO TO 1210
1300 PRINT AT X,Y;"  ";AT X+1,Y;"  "
1310 PLOT INK 3;24,159:DRAW INK 3;207,0
1320 GO TO 135
1990 PRINT AT 0,15;"    ";AT 1,20;"          "
1992 PRINT INK 1;FLASH 1;AT 1,8;" ИГРА ОКОНЧЕНА "
1994 FOR N=0 TO 400:NEXT N
2000 PRINT AT 0,15;"    ";AT 1,20;"          "
2002 LET OW=OW+NW
2003 IF OW>=0 AND INT ((OW-10)/100)=INT (OW/100) THEN PRINT INK 7;PAPER 1;BRIGHT 1;AT
     1,4;"ИТОГОВЫЙ ВЫИГРЫШ  #";OW/100;"0 "
2004 IF OW>=0 AND INT ((OW-10)/100)<>INT (OW/100) THEN PRINT INK 7;PAPER 1;BRIGHT 1;AT
     1,4;"ИТОГОВЫЙ ВЫИГРЫШ  #";OW/100;".0"
2006 IF OW<0 AND INT ((OW-10)/100)=INT (OW/100) THEN PRINT INK 7;PAPER 1;BRIGHT 1;AT
     1,4;"ИТОГОВЫЙ ПРОИГРЫШ #";-OW/100;"0 "
2007 IF OW<0 AND INT ((OW-10)/100)<>INT (OW/100) THEN PRINT INK 7;PAPER 1;BRIGHT 1;AT
     1,4;"ИТОГОВЫЙ ПРОИГРЫШ #";-OW/100;".0"
2009 FOR N=0 TO 8
2010 DIM I$(26)
2020 PRINT PAPER 2;BRIGHT 1;AT 13+N,3;I$
2030 NEXT N
2040 PRINT INK 7;PAPER 2;AT 14,4;" ВЫ ИЗРАСХОДОВАЛИ - ";BRIGHT 1;L
2050 PRINT INK 7;PAPER 2;AT 16,7;" И ПОЛУЧИЛИ - ";BRIGHT 1;W
2060 IF NW>=0 THEN PRINT INK 7;PAPER 2;AT 18,5;"ВАША ПРИБЫЛЬ - ";INK 7;PAPER 0;BRIGHT 1;AT
     18,22;NW
2070 IF NW<0 THEN PRINT INK 7;PAPER 2;AT 18,7;"ВАШИ ПОТЕРИ - ";INK 7;PAPER 0;FLASH 1;BRIGHT
     1;-NW
2080 PRINT INK 7;PAPER 1;BRIGHT 1;FLASH 1;AT 20,4;"СЫГРАЕТЕ ЕЩЕ РАЗ? ДА-<Y>"
2082 IF T>T1+240 AND INKEY$="Y" THEN LET T1=FN T():LET OW=0:GO TO 2100
2085 IF INKEY$="Y" THEN GO TO 2100
2090 GO TO 2040
2100 PRINT AT 1,3;"                              "
2105 FOR N=0 TO 8
2110 DIM I$(26)
2120 PRINT PAPER 6;AT 13+N,3;I$
2130 NEXT N
2140 GO TO 75
2500 DIM I$(704):PRINT AT 0,0;I$
2520 PLOT INK 6;26,144:DRAW 0,-32,11*PI/10:BEEP .05,10
2523 PAUSE 30
2524 PLOT 32,112:DRAW 12,32:BEEP .05,10:DRAW 12,-32:BEEP .05,10:PLOT 35,120:DRAW 18,0:BEEP
     .05,10
2526 PLOT INK 6;80,136:DRAW -8,-8,3*PI/2:BEEP .05,10:PLOT INK 6;64,120:DRAW 8,8,3*PI/2:BEEP
     .05,10
2527 PAUSE 30
2528 PLOT 88,112:DRAW 0,32:BEEP .05,10:PLOT 104,112:DRAW 0,32:BEEP .05,10:PLOT 88,128:DRAW
     16,0:BEEP .05,10
2529 PAUSE 30
2530 PLOT 116,128:DRAW 8,0:BEEP .05,10
2531 PAUSE 30
```

```
2532 PLOT 136,112:DRAW 0,32:BEEP .05,10:DRAW 16,0:BEEP .05,10:PLOT 136,128:DRAW 10,0:BEEP
     .05,10
2533 PAUSE 30
2534 PLOT 176,112:DRAW -16,0:BEEP .05,10:DRAW 0,32:BEEP .05,10
2535 PAUSE 30
2536 PLOT INK 6;194,144:DRAW 0,-32,5*PI/6:BEEP .05,10:DRAW 0,32,5*PI/6:BEEP .05,10
2537 PAUSE 30
2538 PLOT 214,144:DRAW 8,-32:BEEP .05,10:DRAW 8,16:BEEP .05,10:DRAW 8,-16:BEEP .05,10:DRAW
     8,32:BEEP .05,10
2540 FOR N=13 TO 21
2545 PRINT INK 2;AT N,3;"EF  EF  EF  EF  EF  EF  EF"
2550 PRINT INK 4;AT N,2;"F";AT N,29;"E"
2555 NEXT N
2558 LET Q=5
2560 FOR N=0 TO 3:PRINT INK 1;AT 12+2*N,Q;"  ";AT 13+2*N,Q;"  ";AT 14+2*N,Q;A$;AT
     15+2*N,Q;B$:BEEP .05,Q:NEXT N
2561 FOR N=0 TO 2:PRINT INK 1;AT 12+2*N,Q;"  ";AT 13+2*N,Q;"  ";AT 14+2*N,Q;A$;AT
     15+2*N,Q;B$:BEEP .05,Q:NEXT N
2562 FOR N=0 TO 1:PRINT INK 1;AT 12+2*N,Q;"  ";AT 13+2*N,Q;"  ";AT 14+2*N,Q;A$;AT
     15+2*N,Q;B$:BEEP .05,Q:NEXT N
2563 PRINT INK 1;AT 14,Q;A$;AT 15,Q;B$:BEEP .05,Q
2565 LET Q=Q+4:IF Q<29 THEN GO TO 2560
2570 IF Q=29 THEN PRINT FLASH 1;BRIGHT 1;INK 1;AT 10,1;"ВАМ -НУЖНА ИНСТРУКЦИЯ? ДА - <Y>"
2575 PAUSE 0:IF INKEY$="Y" THEN GO TO 2600
2580 RETURN
2600 CLS
2610 PRINT INVERSE 1;AT 1,10;" CASH-FLOW "
2615 PRINT AT 3,0;"
        ВАША ЦЕЛЬ - СОБРАТЬ  5 МОНЕТ В
      ОДНОМ  МОНЕТОПРИЕМНИКЕ.  В  ЭТОМ
      СЛУЧАЕ ВСЕ ОНИ ДОСТАНУТСЯ ВАМ."
2620 PRINT AT 6,0;"
        У ВАС ЕСТЬ 6 МОНЕТ ПО 10  ПЕН-
      СОВ ПЛЮС ВСЕ, ВЫИГРАННЫЕ ВАМИ."
2622 PAUSE 100:PRINT INK 6;PAPER 2;FLASH 1;AT 9,4;" НАЖМИТЕ  ЛЮБУЮ КЛАВИШУ "
2624 PAUSE 500:PRINT AT 9,4;"                              "
2628 PRINT INK 6;PAPER 2;AT 9,9;" УПРАВЛЕНИЕ "
2630 PRINT AT 11,0;"A - ВСТАВИТЬ МОНЕТУ В АВТОМАТ";AT 13,0;"Z - ЗАПУСТИТЬ МОНЕТУ"
2635 PRINT AT 15,0;"O И P - ПОДТОЛКНУТЬ МОНЕТУ      ВЛЕВО ИЛИ ВПРАВО (ТОЛЬКО 1 РАЗ)"
2640 PRINT AT 18,0;"
      Q - ПРЕКРАЩЕНИЕ ИГРЫ (ТОЛЬКО ПО-
      СЛЕ ТОГО,  КАК  ВЫ  ИЗРАСХОДУЕТЕ
      ПЕРВЫЕ 50 ПЕНСОВ"
2642 PAUSE 100
2644 DIM I$(160):PRINT PAPER 6;AT 3,0;I$
2650 PRINT INK 2;FLASH 1;AT 5,3;" НАЧИНАТЬ ИГРУ?  ДА - <Y> "
2660 IF INKEY$="Y" THEN RETURN
2670 GO TO 2660
3000 RESTORE :LET H=0
3005 POKE 23675,88:POKE 23676,255
3010 LET U=PEEK 23675+256*PEEK 23676
3020 READ J:IF J=.5 THEN RETURN
3030 POKE U+N,J:LET N=N+1:GO TO 3020
3040 DATA 7,31,63,112,119,228,239,231,224,248,252,14,230,247,247,247,231,224,231,97,112,
     63,31,7,247,247,231,198,14,252,248,224
3050 DATA 127,127,127,127,127,127,127,127,254,254,254,254,254,254,254,254,.5
```

Программа русифицирована по методике, описанной выше. Автостарт программы - со строки 2, где происходит загрузка символьного набора. В строке 3 происходит включение символьного набора, а в строке 4 - формирование банка символов UDG-графики при помощи подпрограммы 3000. Формируются только символы "A" -"F", так как только они применяются в программе. После этого происходит старт программы с начальной строки. Так Вы будете запускать программу при остановке во время отладки: командой RUN.

Фактическое начало выполнения программы - строка 10. Здесь происходит принудительное включение регистра CAPS LOCK для того, чтобы упростить процедуру опроса нажатых клавишей при помощи INKEY$. Например, чтобы проверить нажатие клавиши "A", обычно применяется конструкция:

```
IF INKEY$="a" OR INKEY$="A" ...
```

Включив CAPS LOCK, можно исключить первую половину этого выражения, так как невозможно будет нажатие "a".

В строке 15 обнуляется счетчик суммарного выигрыша на протяжении нескольких игр, проведенных подряд. Далее задается изображение верхней и нижней половины монеты при помощи символов UDG-графики. В строке 25 выполняются действия, связанные с прорисовкой титульной страницы игры при помощи подпрограммы 2500. В строке 2500 фактически выполняется очистка экрана, но, в отличие от CLS, это происходит плавно, для более приятного зрительного эффекта. Строки 2520 - 2538 воспроизводят название игры, причем это делается довольно оригинальный способом.

Строки 2540 - 2555 прорисовывают монетоприемники, а строки 2560 -2565 заполняют их монетами.

В строке 2570 начинаются действия, связанные с выводом на экран инструкции игры. Для этого надо нажать "Y". Если это зафиксировано, то подпрограмма переходит на строки 2600 - 2670 для вывода краткой инструкции по игре. Если нет, то подпрограмма завершается и выполняется возврат на продолжение выполнения программы со строки 30.

Строки 30 - 95 обеспечивают прорисовку игрового поля. 35 -голубые полосы слева и справа экрана. 40 - 65 - штыри для изменения направления падения монеты. 75 - 90 - монетоприемники. 95 -направляющая для движения монеты.

В строке 100 обнуляются счетчики выигрыша и проигрыша, затем происходит переход на строку 1000. Здесь монетоприемники заполняются монетами случайным образом. Ну не совсем случайным, все-таки имеются некоторые ограничения. В крайних монетоприемниках задается не более двух монет, а в третьем и пятом их количество может достигать четырех. Количество монет в монетоприемниках 1-6 определяется соответственно переменными A-F. Так как в 4-м монетоприемнике не может быть монет по условиям игры, то переменная D не задается.

Строка 1055 задает Ваш "начальный капитал". Далее выводится предписывающая надпись и автомат ждет, когда Вы опустите монету в автомат. Если Вы это сделаете (нажатием "A"), то программа переходит на строку 1100.

В строках с 1100 происходит движение монеты по направляющей, проходящей внутри автомата. Теперь Вы должны в нужный момент сбросить ее вниз, нажав "Z". Если Вы не сделаете это вовремя, пока она не докатится до конца направляющей, то Вы ее просто потеряете. При нажатии "Z" программа переходит на строку 1200. Это начало падения монеты. Дистанция по вертикали, которую монета прошла от верхней направляющей, обозначена переменной M. При достижении M=2, строки с 1300 обеспечивают "закрывание щели", через которую монета провалилась в автомат. Далее программа переходит на строку 135.

В строках 151 и 152 происходит проверка нажатия клавишей "O" или "P", при помощи которых можно подтолкнуть монету. Обратите внимание на условия, при которых возможно подталкивание монеты. Это, прежде всего, флажок, запоминающий факт подталкивания - I. Монету можно подтолкнуть только один раз. Кроме того, это дистанция, на которой монета находится от верхней направляющей. Если монета упала слишком глубоко (M больше или равно 4), то подтолкнуть монету уже не удастся. Факт подталкивания монеты сопровождается коротким звуковым сигналом. Строка 160 обеспечивает прорисовку монеты на новом месте. Строка 166 определяет скорость падения монеты.

При падении монеты, когда она достигнет уровня монетоприемников (что определяется по величине дистанции M), программа переходит на строку 500. В этих строках вычисляется, сколько же монет стало в том монетоприемнике, куда упала монета. Если там к этому времени уже находилось 4 монеты, то программа переводит на строку 600. Если монета упала в четвертый монетоприемник, то она пропадает (строка 565). Если это

была последняя монета, то переход на финал игры, строку 2000 (определяется в строке 568). Строка 570 обеспечивает продолжение игры.

Если в монетоприемнике оказалось 5 монет, то программа продолжается со строки 600. Здесь Вам достаются все выигранные 5 монет, а монетоприемник освобождается. В строках 620-625 обнуляются счетчики монет в соответствующем монетоприемнике. Со строки 630 все действия повторяются.

Финальная часть программы - это строки с 2000. Здесь производится рассчет результата игры - разницы между израсходованными и выигранными монетами, рассчет суммарного результата по итогам нескольких игр и вывод всех этих данных на экран. В строке 2060 Вам предлагается продолжить игру. В том случае, если Вы согласитесь, нажав "Y", игра будет продолжена.

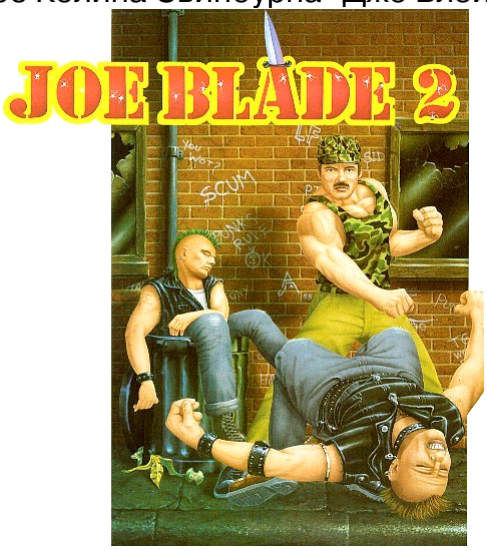
Начиная набивать программу, наберите строки 1-3. Потом сделайте RUN 2 и загрузите заготовленный русско-латинский символьный набор. После его включения продолжайте набивать программу. Имена переменных набирайте в режиме CAPS LOCK. Обратите внимание, что символы "A" - "O" в строке 20 набираются при помощи графического регистра. То же относится и к символам "E" - "F" в строках 80, 85 и 2545, 2550. Периодически выполняйте RUN 5 для записи результатов работы на магнитофон.

Компьютерная новелла

Матвеев Ю. А.

ОТВЕТНЫЙ УДАР КРАКСА

(по игре Колина Свинбурна "Джо Блэйд-2").



Было около полудня, когда в квартире Джо Блэйда резко зазвонил телефон. Разгоняя остатки сна, Джо успел пару раз перевернуться с боку на бок прежде, чем дотянулся затекшей рукой до холодной трубки телефона.

- Говорите, - прохрипел он сонно.

- Все спишь? - после небольшой паузы спросил грубый незнакомый голос.

- Нет... По телефону разговариваю. - Джо почесал щеку и подумал о том, что побриться сегодня не помешало бы. - А кто это?

На другом конце провода послышался смех. Джо уже хотел швырнуть трубку, как вдруг услышал фразу, которая окончательно привела его в чувство:

- Смотри, а то смерть проспишь, щенок…

Незнакомец отключился. Джо положил трубку, потянулся и сел на кровати.

Так с ним никто еще не шутил. Джо понял, что дело серьезное и, прежде чем переходить к активным действиям, решил принять душ и хорошенько все обдумать.

Еще вчера вечером звонили из мэрии и просили подъехать на прием завтра к часу дня. Джо хотел, чтобы его соединили с мэром напрямую, но секретарь отказал. Там шло совещание.

С тех пор, как Джо взорвал тюрьму Кракса, мэр практически исчез из поля зрения. Он видел его лишь один раз на церемонии награждения. Мэр приколол на грудь Джо Блэйда медаль Почета, пожал руку и, как-то вяло поблагодарив, стал поспешно зачитывать фамилии сотрудников безопасности, получивших благодарность.

Церемония проходила в узком кругу доверенных лиц и прессой не освещалась. Отчасти поведение мэра объяснялось тем, что Краксу удалось тогда выкрутиться и отделаться легким испугом. Он не потерял места в городской Думе, а также остался президентом военно-промышленной компании. Его влияние в городе, старые связи с ушедшими в отставку после той истории, по-прежнему позволяли сохранять в своих руках основные рычаги власти.

Но Кракс стал осторожнее, он на время ушел в тень, стал реже мелькать на телеэкранах и страницах газет. Было очевидно, что рано иди поздно мэру придется туго, ибо Кракс никогда не забывал обид. И вчера подготовка к ответному удару, видимо была закончена. Иначе зачем мэру опять потребовался Джо Блэйд? Чем он мог помочь?

Приняв душ и перекусив на скорую руку, Джо подошел к телефону и набрал номер приемной мэрии. Пока в трубке шли длинные гудки, Джо прикурил сигарету.

- Вас слушают.

- Здравствуйте. Говорит Джо Блэйд. Соедините меня с мэром.

- Это невозможно. Идет совещание.

- Послушайте, я сегодня… В трубке что-то щелкнуло и Джо услышал голос мэра:

- Это ты? Боюсь, что мы опоздали.

- В чем дело? - Джо выдохнул густое облако дыма.

- В городе беспорядки. Толпа каких-то молодчиков шатается по улицам, крушит витрины магазинов, терроризируют население… Полиция бездействует.

- Почему? - спросил Джо, догадываясь, что главный виновник происходящего уже обо всем позаботился.

- Силы моих людей небезграничны, а остальные не в моем подчинении…

Джо заметил, что мэр перешел на шепот и старается не называть имен.

- Он рядом?

- В зале заседаний. Там все слышно. Я вышел позвонить, а тут ты… - Мэр тяжело вздохнул.

- Что же вы теперь решаете?

- Они предлагают ввести чрезвычайное положение, чтобы навести порядок.

- Я так и думал. Но вы, конечно, не соглашаетесь?

- Голосов поровну и пока ничья. А ничья в мою пользу. Теперь они требуют собрать расширенное заседание Думы и я не ручаюсь за исход битвы. Им будет достаточно перевеса в один голос и тогда он станет комендантом города на время действия чрезвычайного положения.

- Сколько еще придет горожан на расширенное заседание?

- Шестнадцать человек.

- И какое у них настроение?

- Ни за что не ручаюсь. Я доверяю им, конечно, но где гарантия, что Кракс не подкупил и их.

Джо Блэйд посмотрел на часы.

- Когда начнется заседание?

- Через час… - обреченно ответил мэр.

Джо Блэйд задумался. Что можно сделать в этой ситуации? Выловить заседателей где-нибудь на подходе к мэрии и объяснить обстановку? Они сочтут это за прямое давление и, чего доброго, согласятся с Краксом. Разогнать всю толпу хулиганов? Одному это вряд ли по силам. Для того, чтобы хоть что-то сделать, нужно знать намерения Кракса Бладфингера в отношении этих шестнадцати горожан. Они, конечно, "людишки" для него, но их голоса сейчас стоят очень дорого.

Мысли Джо прервал тихий голос мэра:

- Я видел краем глаза на столе Кракса пакет уже готовых документов и распоряжений. Среди прочих, там есть приказ о твоем аресте…

Эти слова мэра Джо оценил как руководство к действию,

- Понятно, - сухо ответил он. - До свидания.

- Что собираешься делать? - спросил мэр, но ответа не дождался: Джо бросил трубку.

Еще с минуту он собирался с мыслями, потом подошел к окну. Джо жил в стареньком трехэтажном домике на втором этаже. Окна его квартиры выходили на узкую серую улочку. Обычно многолюдная, улица сегодня была абсолютно пустынной. Только в самой ее конце у витрины галантерейного магазина наблюдалось какое-то движение.

Джо быстро оделся и заглянул в шкаф, где на верхней полке среди прочих вещей поблескивал хромом распылитель. Решив, что с оружием он привлечет к себе лишнее внимание, джо не стал брать его с собой. Спустя минуту он уже стоял под окнами своего дома.

С другого конца улицы доносились крики и хохот. Потом Джо услышал звон разбитого стекла и чью-то брань.

Человек пять или шесть шли прямо к нему, разбивая витрины магазинов и кидая в окна домов камни. Им, видимо, было очень весело и свободно на этих пустынных улицах, где

они чувствовали себя хозяевами. Все как один были пьяны.

Джо двинулся им навстречу. Его, казалось, не замечали.

Какой-то панк, высоко подпрыгнув, попытался разбить уличный фонарь, но не достал, а пролетев вперед пару метров чуть не столкнулся с Джо.

Окинув его стеклянным взглядом, и видимо приняв за своего, панк пошел дальше.

"Это хорошо", - подумал Джо и буквально нокаутировал ударом ноги в челюсть подошедшего к нему наркомана.

Никто на действия Джо не прореагировал. Вся группа прошла мимо, пребывая в каком-то экстазе разрушения.

"Без наркотиков здесь не обошлось", - подумал Блэйд, глядя им вслед.

Развернувшись, он быстро пошел по пустынной улице в сторону центра города, где заходилось здание мэрии. Джо надеялся перехватить заседателей на подходе.

По пути он встретил еще одного панка, видимо отбившегося от той шумной группы. Ударом ноги в прыжке, Джо надолго успокоил наркомана.

Неожиданно он заметил у стены дома на тротуаре небольшой круглый предмет с циферблатом на лицевой стороне, очень похожий на будильник. Прибор вел обратный отсчет времени, все это было до боли знакомо. Джо покопался в собственной памяти. Ну конечно! Это был военный таймер-универсал, имевший свою передающую антенну и способный задействовать любой электронный прибор на расстоянии. Может применяться в самых различных целях.

Как оказалась эта штука на улице? Какой механизм она должна привести в действие через девять минут, если верить ее показаниям?

Джо не нашел ответа на эти вопросы. У него возникло желание вдребезги разбить таймер, но, подумав, он нажал кнопку на задней стенке прибора, установив тем самым счетчик времени на первоначальное значение. Что бы ни должно было произойти, Джо оттянул время еще на десять минут. Этот отрезок времени был задан программой таймера.

Без сомнения, Кракс затеял опасную игру и теперь во что бы то ни стало нужно было помешать ему достичь своей цели.

Джо подошел к двери подъезда и толкнул ее. Он рассчитывал выйти с черного хода на другую улицу, чтобы не обходить кругом целый квартал. Однако, дверь была заперта. Жители города в панике закрылись в своих домах в надежде спастись от буйных панков. Никто не знал, что это может помешать Джо осуществить его замысел.

Впрочем, у Джо Блэйда всегда имелось в запасе штук двадцать всякого рода отмычек. Одной из них он и воспользовался.

Джо вышел на соседнюю улицу. Здесь так же бесчинствовали панки, алкоголики и наркоманы. Нескольких, самых буйных, Джо завалил своим коронным приемом. Во всеобщей неразберихе никто ни на что не обращал внимания.

Пройдя вперед несколько метров, Джо заметил какого-то гражданина почтенного возраста в сером плаще и черной шляпе. Он явно куда-то торопился, однако ему все время приходилось прижиматься к краю тротуара, чтобы пропустить пробегающих мимо молодчиков. Джо подошел, к нему сзади и тихо, чтобы не испугать человека, окликнул:

- Здравствуйте, я от мэра.

Человек в плаще вздрогнул и обернулся. Это был один из членов городской Думы. Джо знал его в лицо. Такие люди часто появляются на экране телевизора и спутать их с кем-нибудь невозможно.

- Ваш мэр, - нахмурив брови ответил старичок. - Просто полный идиот... Я не хочу с вами разговаривать.

- Постойте, - Джо схватил его за руку. - Вы же всегда были за мэра, что случилось?

- А сейчас против, и дай Бог Краксу сил и здоровья!

Джо еле сдержал себя, чтобы не съездить уважаемому члену городской Думы по физиономии, но он заставил себя быть вежливым и лишь сурово посмотрел на собеседника.

- Да! Я буду голосовать за Кракса Бладфингера! - завизжал старик. - Только он может навести порядок в этом городе. Все! Разговор окончен!

Джо нехотя отпустил руку. В чем дело? Откуда такая ненависть? Неожиданно в голову

пришла сумасшедшая мысль. Перед глазами у Джо проплыл таймер, найденный на улице и картина стала резко проясняться.

- Минуточку. - Джо быстро заломил руку заседателя за спину и, пока тот сопротивлялся и пыхтел, в темпе проверил его карманы. И он не ошибся. В левом кармане плаща лежал небольшой, со спичечный коробок, предмет. В одно мгновение он оказался в руке у Джо.

Его подозрения оправдались. Кракс Бладфингер занялся электронным гипнозом, чтобы заполучить голоса членов Думы.

Отмахиваясь от озверевшего старичка, как от назойливой мухи, Джо нажал кнопку "ВЫКЛ.", но опять столкнулся с любимой идеей Кракса - секретным кодом. Кракс защищал себя как мог - он предвидел, что кто-нибудь может обнаружить гипнотизатор и тогда, при ошибке ввода кода, гибель нашедшего эту штуковину неизбежна.

Шестьдесят секунд между жизнью и смертью. Джо торопился, нервничал и это чуть его не подвело. Но он сумел взять себя в руки и справился с этой задачей. Приборчик замолк. Джо обернулся, старичок, улыбаясь, стоял сзади и хитро подмигивал.

- Я вас узнал, Джо Блэйд. Мэр должен сегодня победить!

Потом он приложил палец к губам и, надвинув на глаза шляпу, отправился своей дорогой на голосование.

Джо от радости лихо подпрыгнул и одним ударом свалил в сточную канаву сразу двух панков.

Теперь все прояснилось. Джо не сомневался, что люди Кракса Бладфингера каким-то образом сумели подбросить всем шестнадцати заседателям электронные гипнотизаторы

Кракс хотел полной победы и, вполне вероятно, что даже те, кто был настроен к нему лояльно, не избежали этой участи.

Гипнотизатор сильная штука, и, чтобы не убить человека резким вторжением в его мозг, на улицы города, по которым пойдут депутаты, подбросили таймеры, которые поддерживали десятиминутный щадящий режим работы гипотизатора. По прошествии этого срока процесс становился необратимым и человек уже не смог бы избавиться от наваждения.

Джо мчался по улице, сбивая на своем пути всех хулиганов и наркоманов. Он засек время и видел, что оставалось шесть минут до окончательного перехода заседателей Думы на сторону Кракса. Шесть минут - если только он не найдет таймер. Как назло двери всех переходов между улицами были заперты. Джо стал подозревать и в этом Кракса. Отмычки кончались со скоростью света и Джо понял, что такое дело нельзя пускать на самотек. На одной из улиц с огромным отвращением он перевернул мусорный бак в надежде найти хотя бы кусок проволоки. К сожалению, ничего подобного он не обнаружил.

Он пересек еще несколько улиц и увидел таймер. Прибор сразу бросился в глаза. Джо подбежал и, уже готовый сбросить его на прежнее время, заметил заседателя городской Думы. Тот неспеша проходил мимо.

Сколько всего таймеров подброшено на улицы города? Джо не находил ответа на этот вопрос. Может быть по одному на каждого заседателя, а может быть всего два, и поэтому он не стал рисковать. Оставив на время таймер, Джо подкрался к заседателю и, не говоря ни слова, принялся его обыскивать. Прибор он нашел быстро и ситуация повторилась. Не обращая внимания на возмущенные крики горожанина, Джо занимался кодом. Надо сказать, что здесь код оказался сложнее и Джо изрядно попотел, прежде чем решил эту задачку.

Очнувшись от чар электронного гипноза, заседатель любезно раскланялся с Блэйдом и заверил его в своей преданности мэру.

Теперь настало время сбросить таймер и отсрочить развязку еще на десять минут. Но Джо не расслаблялся. Все улицы, прилегающие к зданию мэрии, необходимо было тщательно контролировать, чтобы не пропустить ни одного заседателя, а свои отмычки Джо почти израсходовал. Опрокинув пять мусорных баков, Джо пополнил использованные запасы десятью ключами.

Время шло, но по мере того, как Джо встречал все новых и новых членов городской

Думы, его действия приобретали почти автоматический характер: внезапное нападение, гипнотизатор, установка кода, вежливое прощание…

Позже, вспоминая эти сумасшедшие минуты, он улыбался своей мысли о том, что со стороны его поведение могло показаться странным. Это он был загипнотизирован, ибо действовал как робот…

Джо Блэйд вздохнул свободно только тогда, когда шестнадцатый человек ушел на голосование, глядя на мир не через пленку, накинутую Краксом Бладфингерон, а своими глазами.

* * *

Из протокола N29 экстренного заседания Думы в расширенном составе:

"… считать нецелесообразным введение чрезвычайного положения. С учетом принятого решения по этому вопросу, передать все полномочия на время, необходимое для наведения порядка, мэру города…"

THE DARK WHEEL

Станция "Кориолис" представляет собой огромный мир, воздвигнутый на шести панелях и заполняющий образованное таким образом обширное замкнутое пространство. На противоположных панелях расположены два крупнейших центра - Саут Сити и Норт Сити. Огни, сверкающие в ночи над головой жителя Саут Сити - это ничто иное, как ярко освещенные дома и улицы противоположного города.

Алекс отметился в регистрационном пункте своего причала и взял воздушное такси. Юркий автоматический корабль аккуратно скользнул между громадами прибывающих и отходящих кораблей. В восторженном настроении Алекс наблюдал, как из серой дымки над ним проступали массивы городских строений Саут Сити. Слева от него очень хорошо просматривались улицы и парки другого конгломерата, известного под названием Коммандер Сити. Этот город располагался как раз напротив входа на станцию и традиционно там проживали высокопоставленные чины администрации станции, а также послы и представители других планетных систем. Условия их проживания были особо комфортными. Этот город имел специально сконструированный ландшафт с озерами, реками и парками. Здесь были даже склоны для катания на горных лыжах, покрытые настоящим снегом.

"Немезида" под ним превратилась сначала в бесформенную темную колючку, а потом и совсем исчезла из виду, растворившись на фоне посадочной платформы, а над ним, словно гигантские сталактиты, нависли городские массивы Саут Сити.

Кораблик развернулся и после мгновенной потери ориентации все опять стало на свое место. Дома оказались внизу, а "Немезида" стала едва заметной точкой в темном небе. Такси мягко спустилось на уровень городских улиц между двумя монолитными сооружениями. Вокруг сверкали разноцветные огни и казалось, что тонкий слой атмосферы, покрывавшей город, сам туманно мерцал вместе с этими огнями.

Улицы ночного города были запружены толпами народа и Алекс быстро понял, что на этой станции Саут Сити был своеобразным злачным местом. Здесь, по-видимому, процветала торговля всевозможным запрещенным товаром - рабами, роботами, наркотиками, сенсостимуляторами и замороженными органами. Инопланетники медленно и осторожно продвигались по улицам: многие из них были одеты почти в полный космический костюм, что само по себе указывало на небезопасность окружения. Здесь и там громоздились рекламные объявления, обещавшие все мыслимые и не мыслимые, в большинстве своем запрещенные удовольствия. Высоко в небе, среди рекламных кораблей парили несколько полицейских катеров. Улицы города были заполнены толпами, суетой и грязью.

Темный куб Комплекса Магеллана расположился среди этого хаоса, как огромный монстр. Никаких окон снаружи не просматривалось. Здесь и там по стенам скользили наружные лифты. При виде их медленно перемещавшихся зеленых огоньков казалось, что это сооружение - живое.

Алекс прибыл сюда без оружия и уже начал об этом жалеть. Практически все, кого он видел, были вооружены, несмотря на то, что ношение оружия в пределах орбитальных станций было запрещено. Он аккуратно пробирался сквозь толпу рептилиоидов, амфибиоидов, вооруженных инсектоидов, гротескных робо-танков, похожих то на гигантских моллюсков, то на червей.

Он вошел в здание и здесь ему остро ударил в нос отвратительный запах, объединивший в себе выделения тысяч различных живых форм. Бывают, правда, случаи, когда существа, питающиеся жидким метаном, потеют эфирами с запахом сирени, но увы так бывает далеко не всегда.

Частный торговый центр представлял собой огромный зал, вокруг которого расположились конторы и склады. В этом зале, запруженном толпой, продавались как правило такие товары, которые выставить на открытом рынке было бы слишком рискованно

или вследствие своей изощренной специфики они все равно не нашли бы там покупателя. Во всяком случае, торговые корабли, загружавшие свой товар здесь, немедленно регистрировались официальными службами и более жесткий таможенный мониторинг экспортной службы перед отлетом им был обеспечен. Более того, многим из них был уготован гораздо более плотный контакт со спецслужбами в порту назначения, чем им хотелось бы.

Алекс внимательно пригляделся к стенам в поисках вывески складов Мак Греви. Наконец, он отыскал ее и направился ко входу, когда путь ему преградили два высоких и довольно решительно выглядевших инсектоида. Их тела были покрыты светло серым панцирем, а фасеточные глаза немигая следили за Алексом. Они о чем-то совещались, пощелкивая и потрескивая в своеобразной манере общения. Алекс шагнул в сторону, сердце бешено забилось и кровь хлынула к голове - "... Фасеточные глаза, шарнирные конечности, антенны на голове, двухрядные челюсти... - Таргоиды! - эта мысль мгновенно родилась в его голове. - Таргоиды, здесь на космической станции?!".

Таргоиды были смертельно опасны. У пилотов их кораблей удалялись железы, вырабатывающие вещества, которые вызывают чувство страха. Это совершенно бесстрашные и безжалостные противники. Абсолютно негуманоидные, они были самыми злобными и непримиримыми врагами людей в известном космосе. Награда за уничтожение таргоида была огромной, но еще выше объявленная премия тому, кто доставит в исследовательский центр образцы форм детенышей таргоидов - тарглетов.

Но что могут делать таргоиды здесь, на космической станции? Оба чудовища продолжали беседу, холодно рассматривая Алекса. Алекс обратил внимание на придатки пластины, закрывающей грудную клетку, за которой таргоиды хранят свои ручные лазерные пистолеты.

- Назад, - зашептал голос ему на ухо и Алекс обернулся. Мак Греви стоял рядом. Только сейчас Алекс узнал о его невысоком росте - он едва доставал ему по грудь.

- Таргоиды... - прошептал Алекс.

- Ерунда, сказал Мак Греви и потянул Алекса в сторону. - Это оресрианцы и единственная их опасность - в том, что они очень похожи на таргоидов и их часто путают точно так же, как спутал их и ты. А таргоиды, кстати, их злейшие враги. В следующий раз обращай внимание на пластину, закрывающую грудную клетку и на форму четвертого сустава задней ноги, прежде чем делать поспешные выводы.

Алекс с облегчением последовал за Мак Греви, прочь от продолжавших шептаться странных существ.

Склад Мак Греви был небольшим, до предела загроможденным и очень вонючим. Алекс прошел в тускло освещенный коридор и почувствовал явный дискомфорт, когда Мак Греви закрыл входную дверь. В нескольких больших и прозрачных клетках шевелились и урчали странные создания, обеспокоенные неожиданным вторжением.

Это то, что ты хотел мне предложить? - низким голосом промолвил Алекс. Мак Греви кашлянул. Он подошел к одной из клеток и включил свет, более ярко высветивший то, что находилось внутри.

Алекс застыл. Создание выглядело явно знакомым, но память отказывалась служить. У животного был плотный панцирь, аккуратно разделенный на ячейки. Из образованного панцирем костяного домика через регулярно расположенные отверстия высовывались конечности. Втянув конечности, существо оказалось полностью под защитой своей оболочки.

- Кто это?

- Мимурты,- ответил Мак Греви. - Если тебе они кажутся знакомыми, так это потому, что они выглядят точно так же, как и животные Древней Земли. Кажется, тех называли черепахами или что-то в этом роде. У этих две головы, четыре ноги и еще две каких-то внешних органеллы, которые неизвестно для чего нужны. Их называют по имени той планеты, откуда они родом - Мимурт. Твоя задача - перевезти их на Сираг. У сирагианцев особое отношение к мимуртам.

- Они их едят? - догадался Алекс.

- Они им молятся, - движением губ поправил его догадку Мак Греви.

- Молятся?

Мак Греви кивнул. - Для жителей Сирага мимурты - это нечто вроде живого воплощения их богов. Слышал когда нибудь о реинкарнации? Ну так вот, есть у них такой особый бог Аватар, который возрождается в облике мимурта. Так что мимурты - это живые формы их бога. Мимурты очень похожи на древние представления легенд и мифов Сирага об их боге. Конечно же, мимурты родом с другой планеты и никакого отношения ни к Сирагу, ни к их богам не имеют, но любая сирагская семья отдаст маленькое состояние, чтобы иметь себе мимурта в домашнем храме.

Алекс почувствовал интерес, а успокоенные создания опять начали шевелиться. Из отверстия в корпусе появились и задвигались розоватые конечности. Животные поползли по слякоти, заполнявшей клетку.

- А "маленькое состояние", это сколько?

- Каждый из них потянет на сотню кредов, может чуть больше. Здесь их двадцать восемь штук. Двадцать восемь сотен, этого тебе хватит на на любые защитные экраны и лазеры.

- А почему ты не хочешь доставить их сам?

- С моим-то криминальным списком? Нет уж, благодарю покорно, но я не желаю выходить в космос. У меня теперь другой бизнес. Обычно у меня уходит год, чтобы собрать такую партию, как сейчас, и обычно к этому времени Рейф Зеттер присылает ко мне торговца, нуждающегося вроде тебя в быстром зарабатке для какого-то дела.

Алекс поймал себя на том, что спокойно и без прежнего отвращения смотрел прямо в изуродованное лицо. Он больше не обращал внимания на пульсацию чужеродной жизни под тонкой кожей человека. Он чувствовал, что должен верить этому знакомому Рейфа, ему даже хотелось верить и все-таки он не мог.

- Сделай мне такое предложение, чтобы я не смог отказаться,- продолжил Мак Греви. И Алекс с огорчением вернулся к обыденной прозе жизни.

- Три сотни, - сказал он.

Мак Греви прокашлялся и покачал головой.

- Нет, парень, ты не понял. Идея вовсе не в том, чтоб обобрать тебя. Наоборот, именно ты должен иметь с этого дела основной доход. Откуда ты его возьмешь, если будешь давать мне за голову в три раза больше, чем получишь сам?

- Я имел в виду… три сотни за всех.

На секунду Мак Греви застыл, уставясь на молодого человека

- Это что, шутка?

- Нет, не шутка. У меня всего лишь триста кредов. я не тот человек, который вам нужен, мистер Мак Греви.

- Да ты же только что продал груз шанаскилского меха?

- Да, и купил вооружение и оборудование. К тому же, закупал я эти меха с потерей, в общем, я вам говорю, что я не торговец, а боец. - Алекс еще раз взглянул на мимуртов и произнес - Я беру восемь штук. Договорились?

- Я продаю все или ничего и мне нужно за них пятнадцать сотен. Рейф Зеттер говорил, что…

- Значит, он был неправ. Поищи себе другого сопляка.

Алекс развернулся, собираясь уходить и со злорадным удовольствием услыаал нотки паники в голосе Мак Греви.

- Я берег их для Рейфа. Где я найду еще кого-то, кто будет торговать мимуртами?

- За три сотни я возьму десять штук. Чем больше ты будешь думать, тем меньше я дам.

Торговля начинала нравиться Алексу.

- Но я должен отправить на Сираг именно всю партию.

- Интересно, где этот Сираг, - подумал Алекс. Это название не вызывало в его голове никаких ассоциаций.

- Тогда тебе придется доверить мне так же, как ты доверял Рейфу. Я даю триста

кредов авансом и потом еще треть того, что получу за них на Сираге. Расплачусь, когда вернусь.

Мак Греви молча разглядывал юношу. - Треть вряд ли покроет мои расходы. Пятьдесят процентов.

- Сорок процентов, - сказал Алекс. - Это последнее предложение.

Мимурты опять сосредоточенно завозились в клетках. Мак Греви огорченно пожал плечами и кивнул. Он тут же вызвал по видеоканалам двух свидетелей и в их присутствии контракт был подписан.

Дело обещало быть выгодным для Алекса, если религиозные фанатики этого Сирага раскупят всех мимуртов. Но где же этот Сираг, черт побери?.. Да, теперь "Немезида" будет вооружена лучшими лазерами, снабжена дополнительными энергетическими отсеками и энергетической бомбой. Вот когда начнется настоящая охота!

Алекс вернулся на корабль с докладом о результатах торговли.

ГЛАВА 7.

Теперь они были на нуле. И, надо сказать, сами играючи залезли в такую петлю. Оставалось только найти Сираг и двигать к нему. Алекс проверил официальный планетный регистр, но Сираг там не значился. Так вот почему название казалось ему незнакомым! Вообще-то невключение в официальный регистр еще ничего не значило. Туда включались только обитаемые планеты. Существовали миллионы планет, интересные только для шахтеров, охотников и добытчиков руд, которые упоминались только в галактическом справочнике. Но ведь Сираг совсем иное дело, это планета с разумной жизнью!

Это означало только одно. Значит, Сираг был независимым миром и не признал Федерацию. Стало быть, это было очень опасное и, может быть, даже смертельно опасное место для посещения. Скорее всего, это рай для пиратов, преступников и прочих бандитов всех мастей. По-видимому, это система, в которой сперва стреляют, а потом разговаривают.

- Мы, кажется рехнулись... - сказала Элиссия.

Алекс не возражал. - Слушай, а может быть Сираг - это и есть Раккcла? Может быть, его имел в виду мой отец перед смертью?

- Не может быть. Сираг - это Сираг, а Раккcла, если она вообще существует, находится где-то в другой галактике. Ты ведь знаешь легенды. Судя по всему, Сираг - это чертова дыра. Верни этому парню его мимуртов и давай лучше попробуем толкнуть эти окаменелые кости.

Но Алекс отказался. Он чувствовал, что вокруг него что-то происходит. То, как его "направляли" и как им манипулировали, развило в нем непонятную тягу к этому предприятию. В конце концов, его ждали впереди немалые деньги и наконец-то появлялась возможность закончить вооружение "Немезиды" и начать настоящую охоту, начать жизнь мстителя.

- В общем, пан или пропал? Так? И, как говорит Рейф, если нас ждет неудача, то нас не будет, чтобы ее переживать.

- Мы, наверное, сошли с ума... - опять повторила Элиссия.

- По крайней мере, будет нам наука, как общаться с незнакомыми людьми.

* * *

Перед ними парил Сираг, пастельно желтая планета с темными пятнами то ли гор, то ли пустынь, рисунок которых напомнил Алексу о костях. Девятнадцать световых лет от Ксезавра "Немезида" покрыла с двумя дозаправками и сейчас в ее баках оставалось горючего на прыжок в два световых года, а ближайшая система находилась вдвое дальше.

Впрочем, теперь это не имело значения. С новыми топливозаборниками им достаточно пройти над солнечной короной и горючего будет достаточно.

Солнце Сирага было большой желтой звездой, достаточно старой, но еще вполне активной. Когда Элиссия, сидевшая у навигационной консоли, развернулась к звезде, два

мощных протуберанца вспыхнули на ее поверхности и устремились в космос. Эти воронкообразные вихри очень красиво смотрелись через поляризованные фильтры "Немезиды".

- Давай немного подзаправимся, - сказала Элиссия и дала полное ускорение, но лететь им пришлось не более минуты.

- Святая Звездная Матерь! - Алекс бросил взгляд на экран сканера, и почувствовал, как желудок выворачивается наизнанку. Яркие метки на экране были столь велики, что они могли принадлежать либо крейсерам класса "Боа", либо "Анаконде". Четыре ярких точки, расположенные в боевом строю и рой сопровождающих истребителей не оставляли сомнения о намерениях их экипажей.

- "Боа", - сказала Элиссия. Ведут себя, как боевые крейсеры. Хорошо хоть они медленные. Держись...

Алекс вцепился в кресло. Хорошо, что он был пристегнут. Вселенная вздрогнула, а внутренние органы сделали сальто. Элиссия исполнила лихую петлю и строй истребителей, а это были "Мамбы", распался и начал расходиться веером. Это означало погоню. Но Элиссия не останавливаясь сделала еще один вираж и обманула преследователей, перейдя в лобовую атаку.

Спокойно и аккуратно она поднырнула под брюхо ведущего корабля. Казалось, что это было сделано с удовольствием. Противник открыл огонь с нижней полуплоскости и "Кобра" развернулась вокруг оси, приведя к бою бортовые лазеры. Они как раз пролетали под брюхом огромного крейсера.

- Опознавательные знаки незнакомы, - сказал Алекс. Это были черно-зеленые флаги с изображением ярком солнечной вспышки и неземными иероглифами по бокам.

- Зато их намерения очень знакомы, - вздохнула Элиссия. К этому временя две преследующие "Мамбы" завершили маневр охвата и приближались сзади. Вспышки огня лазеров прорезали черное пространство вокруг яркого солнечного диска. Более крупные корабли к этому времени уже тоже развернулись и угрожающе приближались. И Алекс и элиссия прекрасно понимали, что теперь им не дойти до звезды на дозаправку.

Элиссия развернула корабль. Во время разворота она нацелила и выпустила первую ракету. Ближайший истребитель мгновенно исчез в облаке сверкающей пыли. Несколько зарядов, принятых лобовым экраном, заставили их корабль дрогнуть, но два коротких нажатия ловкими пальцами на кнопку бортового лазера остановили второй истребитель и он закувыркался. Элиссия приблизилась для последнего удара...

Убит!..

Из темноты вышел "Боа". Он медленно вращался и лучи лазера играли у носовой надстройки гигантского корабля. Элиссия нацелила еще одну ракету. По ее лицу текли крупные капли пота, пальцы побелели от напряжения. Алекс, чувствуя свою бесполезность, сидел, напряженно вцепившись в кресло, всем телом повторяя маневры Элиссии.

"Боа" уничтожил ракету, едва она прошла десятую часть дистанции. Но "Немезида" вновь скользнула под брюхо огромного корабля и, развернувшись боком, и уровняв скорости, начала разносить чувствительные участки гиганта огнем из бортового лазера.

Но наконец то, что рано или поздно должно было случиться, произошло. Страшный удар в кормовую часть потряс их корабль. "Немезида" задрожала и начала быстрое беспорядочное вращение. Алекс выругался, почувствовав, как в тело впились пристяжные ремни. Неожиданный удар чуть не оторвал ему голову. С трудом он выпрямился, пытаясь разобраться в ситуации. Сзади приближались две "Мамбы", а неподалеку парила гигантская "Анаконда", готовая их поглотить.

- Мы еще посмотрим кто кого, - громко сказал Алекс и обернулся к Элиссии, недоумевая почему она не маневрирует.

Она обмякла в своем кресле. С головы и носа текла кровь. Она, во-видимому, была слабо пристегнута, когда удар потряс "Кобру" и разбила голову о приборную консоль.

Рывком Алекс покинул кресло второго пилота, освободил Элиссию и буквально бросил ее на пол. Сейчас было не до вежливости. Он круто изогнул траекторию корабля и всадил заряд в распахнутый грузовой отсек "Анаконды". Увернувшись от огня лазера, он тут

же увернулся и от ракеты, сбивать которую уже не было времени. На мнгновение диск планеты проскочил перед ним и Алекс развил полную скорость, пытаясь уйти от смертельной опасности.

Во время этой отчаянной попытки оторваться от врагов неприятная мысль поразила его. "А где гарантия, что на этой планете станция будет его защищать, если он окажется в ее зоне?" Такой гарантии у него не было. Вполне возможно, что и станция будет против него. Надо только дать им знать, какой груз он везет. Если они будут знать, что у него на борту груз их обожаемых богов, они вышлют корабли и прогонят пиратов.

Справа неизвестно откуда вынырнула "Мамба". Он развернул "Немезиду" и встретил ее кормовым лазером, сбросил скорость, опять развернул корабль и поразил противника огнем с правого борта, Корабль не взорвался, но закувыркался и вышел из строя.

Ах, если бы он мог бросить груз, выкинуть в космос контейнеры с системой жизнеобеспечения мимуртов и, может быть, все прекратилось бы. Они с Элиссией потеряют на этом три сотни кредов, ту и что? Они ведь в конце концов еще не "элита".

Новые залпы потрясли корабль. Это опять "Мамба". Алекс нацелил ракету, но запускать ее не стал, открыв огонь бортового лазера.

В это время Элиссия начала приходить в сознание. Она поднялась на ноги и сквозь залитые кровью глаза наблюдала за битвой. Сираг был все ближе и ближе. Маленькая точка серебристого света мигнула им навстречу, но не наполнила Алекса радостью.

- Слушай, в этих контейнерах должно быть не мимурты, - тихо произнесла Элиссия.

- Позже поговорим, - ответил Алекс, маневрируя и выходя из под огня более крупных кораблей противника.

Элиссия покинула мостик. Спасая их жизни, она направилась в грузовой отсек...

И вдруг внезапно атака прекратилась. Алекс чуть не подпрыгнул от изумления. Еще мгновение назад его кормовые экраны едва сдерживали огонь противника. Его бортовой лазер перегрелся от непрерывного огня и вдруг... ничего, тяжелые массы пиратских кораблей отошли назад. Две легких "Мамбы", вцепившиеся в их хвост, выпустили по последнему отчаянному заряду вдогонку и исчезли, промелькнув в солнечных лучах и растаяв в черноте пространства.

Алекс замедлил ход и проверил степень повреждений. К счастью, ничего серьезного. Они лишились двух ракет, низок энергетический уровень, зато груз не поврежден. А раз пираты прекратили преследование, значит Сираг будет защищать своих визитеров.

На мостик вернулась Элиссия, в руках у которой была просвечивающая камера.

- Слушай, эти животные выглядят как черепахи, ведут себя как черепахи и от них воняет как от черепах, но я сделала пару просвечивающих снимков, чтобы посмотреть нет ли там еще чего-нибудь.

- Отличная идея. Давай посмотрим.

- Сейчас, еще две--три минуты.

Она поставила камеру, присела в кресло второго пилота и взглянула на Алекса.

- Ты в порядке?

Алекс кивнул.

- А ты?

- Чувствую себя чертовски разбитой. Мы в безопасной зоне?

- Вроде да.

Перед ними неспешно вращалась станция "Кориолис", ярко освещенная солнцем. Она отбрасывала пятна серозеленой тени на поверхность своей планеты. Неподалеку парили несколько пришвартованных к буям кораблей. Они выглядели достаточно безмятежно. На станции вспыхнули огни. Вскоре все вокруг приветливо засверкало.

Алекс грациозно пролетел сквозь орбитальный город парящих судов и приблизился к точке входа.

Но входа не было!..

- Что за черт?

Он сидел неподвижно. Корабль вращался, уровняв скорость своей ротации со станцией, но вместо входа перед ними серел металл. Увеличив разрешение приборов.

Алекс все-таки сумел различить признаки входного отверстия, но оно было плотно закрыто.

- Может, они опасаются чужестранцев? - пожала плечами Элиссия.

- Но нам очень нужно топливо. Ну нельзя же нас опасаться до такой степени.

Наконец, раздался треск в устройстве громкоговорящей связи:

- "Назовитесь! Назовитесь! Говорит орбитальная станция планеты Сираг."

- "Немезида", торговый корабль класса "Кобра", - сообщил Алекс. - Мы несем груз мимуртов. Просим открыть шлюз.

На некоторое время последовала тишина, хотя потрескивание в канале свидетельствовало о том, что связь не прервана, и затем:

- Внимание "Немезида"! Торговля мимуртами на станции запрещена!

- Что?

- Избавьтесь от груза перед входом на станцию. Бросьте груз. Вы получите компенсацию.

Алекс взглянул на Элиссию.

- Что же нам теперь делать?

- Мне кажется, что все это выглядит как-то непрофессионально. Что-то здесь есть фальшивое.

Она взяла в руки камеру и извлекла оттуда обработанную пленку. Внезапно она застыла, уставившись в два готовых отпечатка, как будто не в силах оценить то, что увидела и наконец произнесла:

- Бог мой… - с этими словами она передала отпечатки Алексу.

В это время начало постепенно открываться приемное отверстие станции. Два ярких прожектора проступили из темноты отверстия, как два горящих глаза.

Алекс вгляделся в отпечатки и на секунду растерялся при виде необычного, гротескного зрелища. Просветив тела живых мимуртов, камера выхватила какие-то паукообразные формы, живущие внутри этих безобидных черепашек. Зрелище было очень неприятным. Суставчатые конечности протянулись в каждую из конечностей мимуртов, заполняя все внутренне пространство. На темном теле просматривались фасеточные глаза. Два длинных отростка протянулись от него внутрь голов мимурта.

- Что это? - прошептал Алекс.

- Это беда. Это детеныши таргоидов.

Сердце Алекса учащенно забилось. Так это тарглеты?! Значит, он транспортировал тарглетов - личиночную стадию самой опасной жизненной формы в известных галактиках. Он подставлен? Нет, "подставка" - это слишком мягкое слово для того, что с ним сделали на Ксезавре. Чего же теперь удивляться, что пираты накинулись на него столь отчаянно?

- За тарглетов дают неплохое вознаграждение. Военные охотно принимают их для исследовательских целей.

- Пираты выращивают их и используют в качестве пилотов на своих истребителях. Мы привезли на Сираг будущих пилотов пиратских кораблей. Понятно, что они на нас накинулись. Им нельзя оставлять свидетелей.

Алекс уставился на станцию. На какое-то время слова Элиссии пролетели мимо и никак не отложились в его голове. Он задумался о пиратах, атаку которых они отбили, он думал, что все позади и раз они находятся рядом со станцией, то бояться больше нечего, кроме обвинения в торговле запрещенным товаром. Да, он думал, что они в безопасности.

Два желтых глаза выскользнули из черной утробы станции, а за ними потянулось и само темное тело корабля. За ним вспыхнул свет и темная тень незнакомого судна упала на "Немезиду". Змеиная тень, тень "Кобры".

Этот корабль он узнал бы где угодно. Прошли месяцы после встречи с ним, но не было ни одной ночи, когда бы мысль о нем не возвращалась к нему в ночных кошмарах. Корабль, погубивший "Авалонию" медленно приближался и ни малейшего сомнения Алекс не испытывал. Не было сомнений и у Элиссии.

Она затаила дыхание и подалась вперед всем телом. - Отдай мне его. Пусти меня к приборам.

- Сядь. - холодно ответил Алекс и Элиссия зло на него посмотрела.

- У меня с ними большие счеты…

- Пилот этого корабля убил моего отца.

- Они убили всю мою семью! Мы бежали с Теорга и обратились к ним за помощью, мы просили немного припасов. Они захватили меня и сестру в качестве рабов, а корабль моего отца расстреляли. Мне удалось бежать, а сестра погибла. Алекс, отдай мне этого ублюдка!

- Сейчас уже поздно…

Вспышка полыхнула со стороны "Кобры". "Немезиду" тряхнуло. Алекс нацелил ракету и всадил мощный заряд из носового лазера в противника. Желтым цветком рассеялась энергия на защитных экранах "Кобры".

Она ускорила свой ход. Алекс тоже прибавил скорость, но завис над станцией и противником.

"Мы не можем сражаться с ней. У нас пока нет ни достаточого вооружения, ни средств защиты. Пока еще нет! Что же делать?" - проносились мысли в его голове.

На заднем экране Алекс видел контур мрачного убийцы, выплывающего из-за орбитальной станции. Вспыхнул сигнал приближавшейся ракеты и Алекс задействовал систему ECM на ее подавление. Сделав это, он развернул корабль. Оба корабля прошли встречными курсами, разрывая друг друга на части. Два величественных металлических галеона, поливающих друг друга огнем. Снова разворот и очередное сближение.

Дважды они схватывались в такой дуэли. "Немезида" стонала под тяжестью лазерных ударов. Энергетические отсеки начали сдавать и вновь сомнения захватили Алекса. Эта "Кобра" знает, кто перед ней и не даст ему уйти. Он тоже хочет ее смерти. Но ведь он еще не готов! Нет, еще нет!

Так, несмотря на упреки Элиссии, Алекс развернул корабль и направил его к звезде. "Кобра" начала преследование. Оба корабля маневрировали, сплетая в петли траектории полета. То тормозясь, то ускоряясь Алекс не упускал ни одной возможности для стрельбы кормовым лазером и это немного сдерживало напор пирата. Алекс сбил еще три выпущенных противником ракеты. Хотелось думать, что весь запас ракет пирата исчерпан, но Алекс не позволял себе расслабиться. Собственная ракета оставалась нацеленной и готовой к пуску. Алекс медлял, сознавая, что ее ждет быстрая и бессмысленная кончина.

Звезда придвинулась. Она сияла в своем огромном великолепии. Температура в кабине "Немезиды" начала постепенно повышаться. Сверхестественными чудовищами, порожденными расплавленным океаном, всплывали вихрящиеся рукава раскаленной плазмы необъятных размеров. К одному из них направил свой корабль Алекс, подготовив топливозаборное устройство.

"Кобра" сделала еще выстрел и вновь заскрежетали силовые экраны. Сражающиеся корабли начали погружение в раскаленную преисподнюю…

- Смотри, оно работает, - сказал Алекс.

Стрелка датчика горючего поползла вверх, по мере того, как топливозаборное устройство всасывало потоки сырой плазмы и конвертировало ее в форму, необходимую для работы гиперпространственных двигателей. "Немезида" скользила по краю огненного океана. Рукава выбросов солнечной короны достигали миллионов миль в длину и тысяч миль в поперечнике. Они свивались в огромные воронки, похожие на водовороты. В центре такой воронки образовывается относительно спокойная, более безопасная и менее раскаленная область.

Туда и направил корабль Алекс. Рубка наполнилась сверхестественным сиянием, солнце светило непереносимым блеском и температура корабля начала катастрофически повышаться. Пламя заиграло на обводах корпуса, застонали силовые экраны.

- Только не долго, - сказала Элиссия. В конце концов и она начала понимать, что они еще не готовы к последней битве с врагом. Сейчас им надо уйти отсюда и побыстрее. До ближайшей звезды шесть световых лет, а у них топлива, судя по показаниям датчика - всего на четыре, но стрелка продолжала подниматься.

Выбрав спокойный участок в океане огня, корабль остановился. Где-то в ярком потоке слепящей плазмы, окружившей их, продолжал свои поиски смертельный враг. По-видимому, здесь они были в безопасности - никакие электронные средства обнаружения не

могли работать в потоке интенсивной радиации, излучаемой солнечной короной. Враг остался без глаз и без ушей.

- Пять световых лет, заряд нормальный. Приготовься к старту. Курс введен…

- Готов, - ответил Алекс. Он старался не думать о возможных последствиях такого дальнего неконтролируемого гиперпрыжка. В первый момент ему пришла в голову мысль попробовать перемещаться малыми скачками, но он решил этого не делать - гиперпривод мог не выдержать многократных запусков.

Алекс развернул корабль и придал ему небольшое вращение, внимательно просматривая окружающее пространство в поисках возможной опасности.

- Пять точка пять световых лет. Еще одна минута. Всего шестьдесят секунд…

- Всего тридцать секунд…

Корабль вибрировал всем корпусом. По лицу Алекса сбегали крупные капли пота.

- Еще двадцать секунд, Алекс и мы полетим.

Едва различимое пятнышко мигнуло на экране сканера, выдавая присутствие пиратской "Кобры" где-то рядом. Она находилась по другую сторону плазменного вихря. Занавес сплошного огня разделил их. "Немезида" и убийца стояли друг против друга, отделенные стеной солнечного пламени.

- Мы готовы, - доложила Элиссия. - Вперед, Алекс! Пошел!

Алекс Райдер отрицательно тряхнул головой. - Нет, пока еще нет…

- Алекс!

Он тронул корабль вперед, в направлении огня. Мерцающее, еле различимое изображение на экране сканера тоже переместилось. Началоеь сближение.

Дикий крик вырвался из груди Алекса, когда он дал максимальное ускорение кораблю, вложив всю возможную мощь в двигатели. Корабль рванулся вперед навстречу стене огня и плазмы. Исчезли все изображения на экранах, единственное, что видел Алекс перед собой - это лицо отца и яркую вспышку пламени, в котором погибла "Авалония".

Горечь, гнев и ненависть - только эти чувства он ощущал в это мгновенье. Он знал, что пока у него есть ракета, уже нацеленная на вражескую "Кобру, у него еще остается один последний отчаянный шанс.

Корабли сближались. Теперь их разделяла только тонкая стена плазменного водоворота. Плазма играла на корпусе "Немезиды" и силовые поля уже не выдерживали напора. Дальше двигаться было нельзя. Глубже - нельзя… опасность!.. И Алекс запустил ракету.

Крошечный снаряд нырнул в солнечное пламя, отыскивая свой путь к цели. Алекс не видел ракету на экране, не видела ее и пиратская "Кобра", пока не стало слишком поздно.

В последний момент "Кобра" включила ECM. За вспышкой света последовала детонационная волна и еще через мгновенье ракета превратилась в стремительно вращающийся огненный клубок, вобравший в себя инерцию ракеты, силу взрыва и потоки раскаленной солнечной плазмы. Развивая скорость и мощь, этот клубок рванулся к вражескому кораблю и не было никакой силы защитных экранов, способной остановить этот клубок солнечной энергии. Через мгновенье "Кобра" купалась в потоках плазмы, захваченная вихрей протуберанца. Алекс не отрывал глаз от экранов и вдруг… все кончилось. "Кобра" была уничтожена. Она погибла, сгинула навсегда.

Медленно развернувшись, "Немезида" легла на обратный курс. Никто на мостике не проронил ни слова, но в ярких лучах старевшего светила блестели слезы на двух счастливых лицах.

ГЛАВА 8.

Галоизображение Рейфа Зеттера дрожало в боевой рубке корабля и гордо светилось. За ним на переднем экране просматривался гостеприимный диск планеты Лейа. Остатки мимуртов с их драгоценными паразитами к этому времени уже были перегружены на два "Эспа" военного флота. Сумма окончательного вознаграждения, правда, еще не была согласована, но получалось не меньше, чем по сотне кредов за экземпляр.

- Я знал, что ты сможешь это сделать, - сказал Рейф, счастливо пережевывая свою

жвачку и лихо сплевывая ее в сторону. - Я был уверен настолько, что направил тебя на Сираг немного раньше, чем ты окончательно подготовился.

- Но мы могли погибнуть. От этой системы мурашки бегут по коже.

- Настоящий боец, даже если он "Элита", знает, когда надо отступить и как надо отступать. Я горжусь тобой. Ты отступил … и победил.

В это время пришло сообщение с "Кориолис-6" из штаб квартиры галактической полиции.

"От имени Галактического Содружества Миров поздравляем Алекса Райдера и приносим благодарность за проявленное умение и мужество при уничтожении пиратских кораблей. Согласно представленному рапорту, подтвержденному пленкой бортовой регистрационной V-системы, присваиваем боевой ранг "Смертельный". Ваш правовой статус "В розыске" снимается, новый боевой рейтинг "Смертельный" будет введен в каналы Галактической сети в течение стандартного дня. Желаем быть мудрым и сильным в бою."

Вот так и случилось, что не достигнув и двадцати лет, Алекс оказался всего на шаг от боевого класса, о которой большинство людей даже и не мечтают. Он вошел в класс "Смертельных". Он уничтожил "Кобру", которая неизвестно за что убила его отца. Почему это было сделано, Алекс конечно не знал, а спросить у пилота-убийцы не подумал. Он предполагал, что это было просто заказное убийство, за которое бандит получил вознаграждение.

Вместо этого, он спросил Зеттера: - Ты знал, что этот корабль на Сираге?

- Предполагал. Алекс, вот почему мы послали тебя с грузом тарглетов. Ты знаешь, никто, абсолютно никто не в состоянии устоять перед искушением захватить такую добычу. Я знал, что ты соберешь всех пиратов в радиусе светового года, но я также знал и то, что ты с ними справишься.

Более того, я был даже уверен, что такая наживка привлечет и эту "Кобру".

Ты бился отлично. Я вижу в тебе те же инстинкты бойца, которые были и в Джейсоне. И он был прав. Ты именно тот человек, который сможет пойти по его стопам.

- Куда пойти по его стопам?

Рейф кашлянул и покачал головой. - Знашь, это сложный вопрос. Твой отец искал следы мифической Раккслы. Он хотел узнать, существует ли она на самом деле? Если легенды не врут, то там должна быть установка, созданная инопланетным разумом, с помощью которой открывается проходы в другие Вселенные, ко всем их сокровищам.

Джейсон Райдер был уверен, что Раккса реально существует. Вот почему он прошел специальную подготовку и был включен в состав лиги Темного Колеса, лиги легендарных первопроходцев. На несколько лет он исчез и я не имел никаких сведений ни от него, ни о нем до самого последнего времени перед его гибелью. Тогда он рассказал мне, что нашел реальные доказательства существования Раккслы. Он вернулся из глубин космоса, чтобы сформировать команду. - Рейф горько улыбнулся. - Но перед самым отлетом он взял несколько дней для отдыха в спокойном месте вместе с сыном… где его и встретил наемный убийца.

- Но почему? Почему его убили из-за Раккслы?

- Потому, что на Раккса уже давно люди. Правда, это только догадка, но судя по тому, что они сделали с Джейсоном, она недалека от истины. Мы давно подозревали, что корпус "Элиты" создал там базу и широко использует переход в иные Вселенные. Это могущественные и решительные люди. У них достаточно средств, чтобы нанять негодяя и уничтожить того, кто несет угрозу их превосходству.

Рейф приблизился чуть вперед, его глаза сияли.

- Я контролировал твои шаги, Алекс, а также и Элиссии. Темное Колесо нуждается в вас. В обоих вас. Но поверь мне, то, через что вы уже прошли - ничто по сравнению с тем, что Вас ждет в будущем. Алекс, вы должны стать "Элитой". А это означает многие месяцы, а может быть и годы тревировок и сражений, но после этого Вселенная распахнется перед Вами так, как Вы не можете себе даже представить.

Алекс стоял и глядел на старика в молчании и задумчивости. Наполовину скрытая в тени, в уголке стояла Элиссия, напуганная тем, что только что услышала.

- Ну как, твое горе утихло? - спросил Рейф и Алекс кивнул. Старый торговец улыбнулся.

- Скажи, что ты чувствуешь став богатым?

- Опустошенность. - и Рейф рассмеялся.

- Да, ты подойдешь для Темного Колеса, это точно!

Содержание