МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ СССР

Тульский государственный педагогический
институт имени Л.Н.Толстого

„ЯМАХА"

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

по использованию графических средств
диалогового языка программирования MSX-БЕЙСИК

Для физико-математических специальностей
педагогических институтов

Москва,
1986

МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ СССР

Тульский государственный педагогический
институт имени Л.Н.Толстого

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

по использованию графических средств
диалогового языка программирования MSX-BASIC

Для физико-математических специальностей
педагогических институтов

Москва,
1986

Составители: А.Р.Ковиа, Л.П.Лапицкая, В.И.Ефимов

Ответственный редактор: А.Р.Ковиа

УТВЕРЖДЕНО

Управлением учебных заведений
Министерства просвещения СССР

Методические рекомендации адресованы студентам вузов, учащимся средних учебных заведений, а также всем желающим ознакомиться с графическими средствами ведущего диалогового языка программирования MSX-БЕЙСИК. В них обсуждаются основные команды и функции экранной графики и на примерах иллюстративного характера разъясняется их синтаксис и семантика. Рекомендации рассчитаны на использование персонального MSX-компьютера "Ямаха".

I. МАСШТАБНАЯ СЕТКА ЭКРАНОВ 2 и 3

Вывод графических объектов реализуется на экраны 2 и 3, на которых имеется по n столбцов и m строк. Любая пара

$$(X_0, Y_0) \qquad X_0 \in \{0, 1, \ldots n-1\}, \quad Y_0 \in \{0, 1, \ldots m-1\}$$

задает на экране конкретную точку (см. рис.I.I).

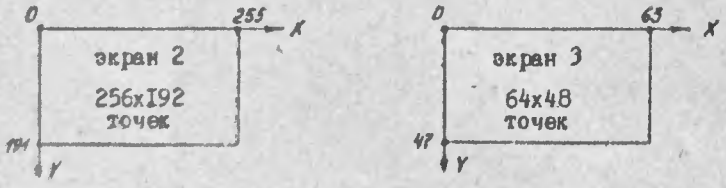


Рис.I.I Схематическое указание системы координат и масштабной сетки для экранов 2 и 3

Для экрана 2:

$n = 256$, $m = 192$,
количество точек = 256x192.

Для экрана 3:

$n = 64$, $m = 48$,
количество точек = 64x48.

В любой конкретный момент времени одна из точек (α, β) экрана находится в особом состоянии или, как говорят, отмечена графическим курсором. α и β в этом случае называют текущими координатами курсора. При включении компьютера графический курсор устанавливается в положение (0,0) – верхний левый угол экрана.

Пусть (α, β) – текущая позиция курсора, X и Y – арифметические выражения, у которых используются целые части X_0 и Y_0 соответствующих значений.

В описании графических команд будут встречаться следующие элементы.

1. @ – необязательный элемент команд, никоим образом не влияющий на их выполнение.

2. (X, Y) без параметра STEP – необязательный элемент, определяющий точку экрана (X_0, Y_0). Если (X_0, Y_0) вне пределов экрана, то элемент, как

правило, не используется или вызывает прерывание при выполнении соответствующей команды.

3. *STEP (X,Y)* – элемент, задающий смещение относительно текущей позиции курсора и определяющий точку ($\alpha + X_0$, $\beta + Y_0$). Если эта точка лежит вне пределов экрана, то элемент, как правило, не используется или вызывает прерывание при выполнении соответствующей команды.

4. *C* (или *C*) – арифметическое выражение, целая часть значения которого лежит в диапазоне [0,15] и определяет цвет точки, области и т.п. По умолчанию *C* есть цвет изображения.

В последующих пунктах дается описание каждой из системы базовых графических команд и функций MSX-БЕЙСИКа. Заметим, что запятую, когда она – запрещающая в команде, указывать не обязательно.

2. УСТАНОВКА И СТИРАНИЕ ТОЧЕК

В этом пункте речь пойдет о двух командах *PSET* и *PRESET*, используемых для перемещения курсора в требуемую точку с возможным изменением ее цвета, а также функции *POINT*, позволяющей узнавать цвет конкретных точек.

Команда *PSET* (установить точку).

Записывается команда в виде

PSET [@][*STEP*](X,Y)[,C]

При выполнении *PSET* графический курсор устанавливается в точку (X_0, Y_0), определяемую элементами (*X*, *Y*) или *STEP(X,Y)* и раскрашивает ее в цвет *C*. Следует иметь ввиду, что это может привести к изменению цвета до 8 близлежащих точек, расположенных в строке X_0.

Пример 1. Точки с разной раскраской

```
10 COLOR 1,15,5:SCREEN 2
20 PSET(0,95)
30 FOR X=0TO15
40 PSET STEP(10,0),X
50 NEXT
60 GOTO 60
```

Пример 2. Синусоида

```
10 COLOR1,15,8:SCREEN 2:PSET(0.95)
20 FOR I=0TO255
30 PSET STEP(1,0),1
40 NEXT
50 PSET(128,0)
60 FOR I=0TO191
70 PSET STEP(0,1),1
80 NEXT
90 FOR I=0TO255
100 PSET(I,50*SIN(6.28318*I/256)+95),5
110 NEXT
120 GOTO 120
```

Команда *PRESET* (стереть точку).

Записывается команда в виде

PRESET [@][STEP](X,Y)[,C]

При выполнении команды *PRESET* без параметра *C* соответствующая ей точка "стирается", то есть закрашивается под цвет фона. Если параметр *C* присутствует, то действия команд *PRESET* и *PSET* идентичны.

Пример 3. Установка и стирание точек (бегущий отрезок)

```
10 COLOR 1,15,8:SCREEN 2
20 FOR I=10TO265
30 PSET(I,95)
40 PRESET(I-10,95)
50 NEXT:GOTO 20
```

Функция *POINT* (точка).

Записывается функция *POINT* в виде

POINT (X,Y)

Ее значением является цвет соответствующей ей точки или -1, когда последняя выходит за пределы экрана.

Пример 4.

```
10 COLOR 1,15,8:SCREEN 2
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
```

3. КРУГИ. ЭЛЛИПСЫ. ДУГИ

По команде *CIRCLE*, записываемой в виде

CIRCLE [@][*STEP*](*X,Y*),*R*,[*C*],[*α*],[*β*],[*e*] ,(I)

вычерчиваются окружности, эллипсы или их дуги. Смысл параметров в (I) следующий.

1. *CIRCLE* (круг, окружность) – служебное слово.
2. *R* – арифметическое выражение, целая часть значения которого определяет радиус окружности.
3. *α, β* – арифметические выражения, фиксирующие углы в радианах, в пределах которых рисуется дуга окружности. По умолчанию:
   $\alpha = 0$
   $\beta = 6.28318\ (\approx 2\pi)$.
4. *e* – арифметическое выражение, задающее коэффициент сжатия окружности. Значение *e* должно лежать в пределах
   $1/260 \leq e \leq 260$
   При $e < 1$ происходит сжатие вдоль оси *OY*, а при $e > 1$ – вдоль оси *OX*. По умолчанию $e = 1$.

Пример 1. Смещенные круги

```
10 COLOR 1,15,8:SCREEN 2
20 PSET(0,85)
30 FOR I=1TO50
40 CIRCLE STEP(5,0),50,4
50 NEXT
60 GOTO 60
```

Пример 2. Смещенные полуокружности

```
10 COLOR 1,15,8:SCREEN 2
20 PSET(0,85)
30 FOR I=1TO25
40 CIRCLE STEP(10,0),50,4,0,3.1415
50 NEXT
60 GOTO 60
```

Пример 3. Смещенные эллипсы

```
10 COLOR 1,15,8:SCREEN 2
20 PSET(0,85)
30 FOR I=1TO25
40 CIRCLE STEP(10,0),50,4,,,5
50 NEXT
60 GOTO 60
```

Пример 4.

```
10 COLOR 1,15,0:SCREEN 2
20 FOR I=1TO100
30 C=INT(RND(1)*16):E=RND(1)
40 CIRCLE (120,95),80,C,,,E
50 NEXT
60 GOTO 60
```

4. ОТРЕЗКИ ПРЯМЫХ. ПРЯМОУГОЛЬНИКИ

По команде *LINE*, записываемой в виде

$$LINE\,[[(@)](STEP)(X_1,Y_1)]-[(@)](STEP)(X_2,Y_2),[C]\left\{\begin{matrix}[,B]\\[,BF]\end{matrix}\right\} \qquad (2)$$

вычерчиваются отрезки прямых и прямоугольники с одновременной раскраской их в тот или иной цвет. Смысл параметров в (2) следующий.

1. *LINE* (линия) – служебное слово.
2. *B* – параметр, определяющий рисование прямоугольника цветом *C* без закраски его внутренней части.
3. *BF* – параметр, определяющий рисование прямоугольника с одновременной закраской его цветом *C*.

Элементы (X_k, Y_k) или $STEP(X_k,Y_k)$ (k=1,2) задают на экране две точки *P* и *Q*. Если первый из элементов отсутствует, то в качестве соответствующей точки берется та, которая отмечена графическим курсором.

На выполнение команды *LINE* существенное влияние оказывают параметры *B* и *BF* или их отсутствие. Рассмотрим три случая.

A. *B* и *BF* отсутствуют. Тогда на экране цветом *C* проводится линия, соединяющая точки *P* и *Q*.

B. Есть параметр *B*. Тогда цветом *C* со сторонами, параллельными осям координат, рисуется прямоугольник. Точки *P* и *Q* оказываются его противоположными вершинами. Внутренняя часть прямоугольника не закрашивается.

C. Есть параметр *BF*. Тогда так же, как и в случае B, рисуется прямоугольник, но внутренняя часть его закрашивается в цвет *C*.

Пример 1. Смещенные прямоугольники

```
10 COLOR 1,15,0:SCREEN 2
20 FOR X=0TO15
30 LINE (30+8*X, 40+8*X)-(120+8*X,90+8*X), 15-X,B
40 NEXT
50 GOTO 50
```

Пример 2. Вложенные прямоугольники

```
10 COLOR 1,15,8:SCREEN 2
20 FOR X=0TO15
30 LINE (5+8*X,10+6*X)-(250-8*X,185-6*X),15-X,BF
40 NEXT
50 GOTO 50
```

Пример 3.

```
10 COLOR 1,15,8:SCREEN 2:PSET(-4,96)
20 FOR X=0TO15
30 LINE STEP(8,-130)-STEP(8,130),X,BF
40 NEXT
50 GOTO 50
```

Пример 4. Приведенный ниже пример рекомендуется прогнать при $P = 50$ и $R = 2$.

```
10 '  З В Е З Д Н О Е   Н Е Б О
20 SCREEN 0:CLS
30 INPUT"плотность (10-200)";P
40 IF P<10ORP>200ORP<>INT(P) THEN 20
50 INPUT"max. размер (0 - 5)";R
60 IF R<0ORR>5ORR<>INT(R) THEN 50
70 COLOR 7,1,8:SCREEN 2
80 I=1:J=2:E=1:DIMF(200),G(200)
90 F(I)=INT(RND(1)*256)
100 G(I)=INT(RND(1)*192)
110 C=INT(RND(1)*R)
120 LINE(F(I),G(I))-STEP(C,C),E,BF
130 LINE(F(J),G(J))-STEP(R,R),1,BF
135 I=(I+1)MOD(P+1):J=(J+1)MOD(P+1)
140 E=(E+1)MOD16
150 FOR K=1TO40:NEXT:GOTO 90
```

5. ЯЗЫК *GML*

Команда *DRAW* (чертить, рисовать) использует так называемый графический макроязык *GML* (*Graphics Macro Language*) и позволяет впрок заготавливать специальные строковые значения для быстрого вывода на экран требуемых рисунков. Записывается она в виде

DRAW β

где β – строковая константа, определяющая рисунок.

Значение β составляется из последовательности команд *GML*. Длина β не должна превышать 255 байтов.

Каждая команда *GML* представлена одиночной ключевой буквой, за которой следует один или два числовых параметра. Последние, как правило, задаются целыми числами. Команды *GML* (см. табл. 5.1) в основном предназначены для перемещения курсора из одних

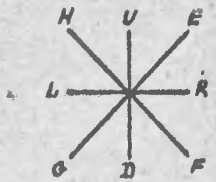
	Команда	Название и выполняемые действия
I	2	3
I 2 ... 8	*Dn*	Движение курсора по направлению *D*. Здесь: 1. *D* – указатель одного из *8* возможных направлений: $D \in \{R, E, U, H, L, G, D, F\}$ Углы между соседними направлениями равны по 45° (см. схему).  2. *n* – целое число, задающее величину смещения курсора (в точках экрана). При выполнении команды курсор смещается от своего текущего положения на *n* единиц по направлению *D*.
9	*M X,Y*	Движение курсора из текущего положения в точку с координатами (*X, Y*).
I0	*M±X,±Y*	Смещение курсора из текущего положения на *±X* единиц по абсциссе и *±Y* единиц по ординате. Указание знака "+" или "–" обязательно.
II	*Cp*	Определение цвета вычерчиваемых отрезков (*p* = 0,1,...15 – код цвета). Действует команда до нового назначения цвета. По умолчанию *P* есть цвет изображения.
I2	*Sm*	Задание масштабного множителя *m*. (1≤*m*≤255). При выполнении *Sm* числовые параметры всех команд движения получают масштабный множитель *m*/4. Действует *Sm* во всех командах *DRAW* до нового назначения.
I3	*At*	Поворот изображения вокруг точки, с которой начиналось рисование на (90·*t*)° против часовой стрелки. Действует *At* во всех командах *DRAW* до нового назначения.
I4	*Xα;*	Интерпретация значения строковой переменной α как последовательности команд *GML*. Знак ";" обязателен.

Таблица 5.I Команды языка *GML*.

точек экрана и другие с одновременным вычерчиванием соединяющих их отрезков. Впрочем, они могут модифицироваться следующим образом.

1. Перед командой "движения" ставится параметр *B*. Тогда положение курсора изменяется, но линия не рисуется.
2. Перед командой "движения" ставится параметр *N*. Тогда курсор возвращается на место, которое он занимал до выполнения данной команды.

Там, где в команде *GML* в качестве параметров используются целые числа, можно указывать и числовые переменные α, в том числе и элементы массивов. Но α при этом следует записывать в форме

$$=\alpha;$$

Примеры команд *GML*

M 100,100 *C7* *XY$;* *N=-A;,-=B;*	*M+5,-20* *S40* *M=X;,=Y;* *C=A8;*	*R 50* *A4* *U=A(3);*

Пример 1. Команды *Un, Cp, Sm*.

```
10 COLOR 1,15,8:SCREEN 2:PSET(80,100)
20 DRAW"c13a12u10e10r10f10d10g10l10h10"
30 GOTO 30
```

Пример 2. Команда *M±X, ±Y*, параметр *N*.

```
10 COLOR 1,15,8:SCREEN 2:PSET(100,85)
20 DRAW"s12c13nu10ne10nr10nf10nd10ng10nl10nh10m+0,30"
30 GOTO 30
```

Пример 3. Команда *At*, параметр *B*.

```
10 COLOR 1,15,8:SCREEN 2
20 FOR I=0TO3:L=40
30 DRAW"a=I:s4c13bm120,85e10r10f10d10g10l10h10u10"
40 NEXT
50 GOTO 50
```

Пример 4. Орнамент

```
10 COLOR 1,15,8:SCREEN 2
20 FOR J=25TO160 STEP 40
30 FOR I=18TO180 STEP 40
40 DRAW"s8bm=I;,=J;e10r10f10d10g10l10h10u10"
50 NEXT I,J
60 GOTO 60
```

6. ЗАКРАШИВАНИЕ ОБЛАСТЕЙ

Для закрашивания областей, ограниченных замкнутыми одноцветными линиями используется команда *PAINT* (красить), записываемая в виде:

$$PAINT\ [@][STEP](X,Y)[,C]$$

Заметим, что цвет C раскраски области должен совпадать с цветом ограничивающего ее контура.

Пусть элементами (X, Y) или $STEP(X,Y)$ определяется некоторая точка (X_0, Y_0). Обозначим через ω области, ограниченные замкнутыми одноцветными линиями, для которых точка (X_0, Y_0) – внутренняя или находится на границе экрана. Если таковых областей нет, то примем за ω весь экран. Далее, пусть ω_0 – минимальная по площади из областей ω. Тогда при выполнении команды *PAINT* область ω_0 закрашивается в цвет C.

<u>Пример I.</u> Раскраска эллипсов.

```
10 COLOR 1,15,0:SCREEN 2
20 FOR I=0TO15
30 CIRCLE(128,95),127-8*I,15-I,,,[illegible]
40 PAINT(128,95),15-I
50 NEXT
60 GOTO 60
```

7. СПРАЙТЫ

Здесь речь пойдет о некоторых дополнительных средствах вывода на экраны графической информации, позволяющих формировать значения специальных переменных и затем выводить их в виде определенных образов на экраны I, 2 и 3.

<u>7.I Формирование значения спрайта</u>

В рамках рассматриваемой версии БЕЙСИКа допускается использование до S специальных нумерованных строковых переменных длины L, называемых спрайтами.

При работе с экранами 0 и I

$$S = 256, \quad L = 8,$$

а номера спрайтов суть:

$$0,\ 1,\ 2,\ \ldots\ 255.$$

При работе с экранами 2 и 3

$$\delta = 64, \quad L = 32,$$

а номера спрайтов:

$$0, \ I, \ 2, \ \ldots \ 63.$$

Имена спрайтов записываются в виде

$$SPRITE\$(n)$$

где:

I. *SPRITE* (светлячок) - служебное слово,

2. *n* - арифметическое выражение, целая часть значения которого определяет номер спрайта.

Значения спрайтов формируются командой

$$SPRITE\$ \ (n) = \beta \qquad (I)$$

где β - строковое выражение.

При выполнении (I), если это требуется, значение β выравнивается до длины спрайта за счет отбрасывания его лишних правых байтов или, наоборот, за счет добавления справа необходимого количества нулевых байтов (00000000). Полученное значение присваивается спрайту с заданным номером.

<u>Примеры</u>.

```
1)  10 sprite$(8)="12345678"

2)  20 sprite$(255)=y$

3)  10 y$=space$(32)
    20 for i=1to32
    30 mid$(y$,i,1)="w"
    40 next
    50 sprite$(32)=y$

4)  10 for s=1 to 63
    20 sprite$(s)="ABCDEFGH"
    30 NEXT
```

Для значений спрайтов отводится определенное место в оперативной памяти, называемое спрайтовой областью и имеющее объем 2 Кбайта.

Заметим, что допустимы присваивания вида

$$\alpha = SPRITE\$ \ (n) \quad ,$$

где α - строковая переменная. Это позволяет при недостатке строкового пространства создавать и хранить значения строковых простых переменных и элементов массивов в спрайтовой области.

7.2 Вывод спрайта на экран

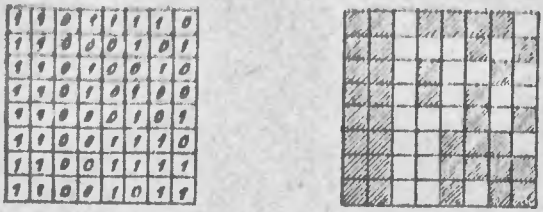
Значения спрайтов обычным путем можно выводить на экраны 0 и 1.

Пример.

```
10 SCREEN 1
20 SPRITE$(50)="screen 1"
30 PRINT SPRITE$(50)
RUN
SCREEN 1
```

Однако, как правило, выводятся не сами значения спрайтов, а некоторые геометрические образы, порождаемые специальной двоичной интерпретацией этих значений. Вывод организуется на экраны 1, 2 и 3.

Пусть длина спрайта 8 байтов. Расположим его двоичное значение в виде квадрата размером 8x8: в первой строке – 1-й байт, во второй – 2-й байт и т.д. На рисунке 7.1 показана двоичная интерпретация значения "чертенок" и соответствующий ей геометрический образ. По такому же принципу любому значению спрайта



а) б)

Рис. 7.1 Специальная двоичная интерпретация значения спрайта (а) и его геометрический образ (б)

длиной в 8 байтов можно сопоставить некоторый геометрический образ на экране, выделив там блок из 8x8 точек и раскрасив в один и тот же цвет все его точки, которым соответствует бит "1" и не изменив цвета других точек.

Пусть теперь длина спрайта 32 байта. Расположим его двоичное значение в виде квадрата 16x16 в две колонки по 16 байтов в строке так, как это показано на рис. 7.2. Тем же способом, что и

I-й байт	I7-й байт
2-й байт	I8-й байт
... ...	... ...
I6-й байт	32-й байт

Рис. 7.2 Схема двоичной интерпретации значения спрайта длиной 32 байта

раньше, этому значению можно сопоставить на экране некоторый блок из I6xI6 точек и в нем конкретный образ.

Средство для вывода на экраны I, 2 и 3 образов спрайтов (коротко спрайтов) имеется. Это команда

PUT SPRITE t,[[@] [*STEP*](X,Y)],[c],[n] (2)

Смысл параметров в (2) следующий.

I. *PUT* (разместить) *SPRITE* - служебные слова.

2. *t* - арифметическое выражение, целая часть значения которого может быть равна:
0, I, 2, ... 3I
и задает "экранный" номер спрайта.

3. *n* - арифметическое выражение, целая часть значения которого определяет номер одного из сформированных в оперативной памяти спрайтов, образ которого и будет выведен на экран под номером *t*. По умолчанию *n = t*.

4. (X,Y) или *STEP(X,Y)* - элементы, определяющие координаты верхнего левого угла блока точек на экране, в котором размещается образ спрайта.

При выполнении команды (2) спрайт с номером *n* цвета *C* получает "экранный" номер *t* и размещается с позиции, определяемой элементами *(X,Y)* или *STEP(X,Y)*. В цвет *C* раскрашиваются точки, соответствующие его единичным битам.

Заметим, что одному значению *n* может соответствовать несколько значений параметра *t*, находящегося в одной или более командах *PUT SPRITE*. Это позволяет при необходимости "размножать" спрайт в виде нескольких идентичных образов на экране (см. пример I).

В одной "строке" можно разместить не более 4 спрайтов.

В отличие от других графических команд здесь элементы *(X,Y)* и *STEP(X,Y)* могут задавать точку (X_0,Y_0) с координатами, удовлетворяющими условиям

$$-32768 \le X_0,Y_0 \le 32767 .$$

К требуемому диапазону (X_0,Y_0) приводится вычислением X_0 и Y_0 по модулю соответствующего экранного параметра (ширины и длины в точках).

Размер выводимых спрайтов определяется вторым параметром *S* в команде

SCREEN [*n*],[*s*],[*c*],[*v*],[*p*] ,

где *S* - арифметическое выражение, целая часть значений которого может быть равна:

0, 1, 2, 3.

Именно эти числа и определяют размеры спрайтов.

S = 0. Спрайты длиной 8 байтов, занимающие 8x8 точек экрана.

S = 1. Спрайты длиной 8 байтов, такие же, как и при *S* = 0, но с двухкратным увеличением размеров по обоим направлениям и занимающие поэтому 16x16 точек экрана.

S = 2. Спрайты длиной 32 байта, занимающие 16x16 точек экрана.

S = 3. Спрайты длиной 32 байта, такие же, как и при *S* = 2, но с двухкратным увеличением размеров по обоим направлениям и занимающие поэтому 32x32 точки экрана.

Некоторые частные вопросы работы со спрайтами можно понять из приведенных ниже примеров. Особенно следует обратить внимание на способ формирования образа спрайта операторами *DATA* .

Пример 1. Двенадцать одинаковых спрайтов

```
10 COLOR 1,15,8:SCREEN 2,3
20 W$="ДВЕНАДЦАТЬСПРАЙТОВ-ЧЕТЫРЕВСТРОКЕ"
30 SPRITE$(40)=W$:S=0
40 FOR I=20TO220 STEP 60
50 FOR J=20TO170 STEP 60
60 PUT SPRITE S,(I,J),13-S,40
70 S=S+1:NEXT J,I
80 GOTO 80
```

Пример 2. Случайные спрайты

```
10 COLOR 1,15,8:SCREEN 2,3
20 FOR I=1TO12
30 Y$="":FOR Z=1TO32
40 Y$=Y$+CHR$(INT(RND(1)*256))
50 NEXT Z:SPRITE$(I)=Y$
```

```
60 NEXT K:S=1
70 FOR I=20TO230 STEP 60
80 FOR J=20TO170 STEP 60
90 PUT SPRITE S,(I,J),13-S
100 S=S+1:NEXT J,I
110 GOTO 110
```

Пример 3. Движение спрайтов

```
10 COLOR 1,15,8:SCREEN 2,3
20 X$="диапазон"
30 SPRITE$(0)=X$+X$+X$+X$
40 FOR K=1TO255
50 PUT SPRITE 5,(K,30),4,0
60 PUT SPRITE 6,(255-K,120),13,0
70 PUT SPRITE 7,(70,K),6,0
80 PUT SPRITE 8,(180,255-K),12,0
90 NEXT:GOTO 40
```

Пример 4. Формирование образа спрайта с помощью команд

```
10 COLOR 1,15,8:SCREEN 2,3
15 DATA 11111111
20 DATA 10000001
25 DATA 10111101
30 DATA 10100101
35 DATA 10100101
40 DATA 10111101
45 DATA 10000001
50 DATA 11111111
55 FOR S=1TO8:READ R$
60 Y$=Y$+CHR$(VAL("&b"+R$))
65 NEXT
70 SPRITE$(0)=Y$+Y$+Y$+Y$
75 PUT SPRITE 0,(110,85),4
80 GOTO 80
```

7.3 Столкновения спрайтов

К возможностям организовывать переходы к подпрограммам по совершению таких событий, как истечение определенного интервала времени, нажатие какой-либо функциональной клавиши или пары клавиш *STRL* и *STOP* можно добавить еще одну – переход к подпрограммам при столкновении спрайтов.

Столкновением спрайтов мы считаем наложение на экране их образов (по максимально возможным размерам).

По команде

ON SPRITE GOSUB n ,

где: 1. *ON* (по), *SPRITE*, *GOSUB* (переход к подпрограмме) – служебные слова.
2. *n* – номер программной строки,

в состояние готовности приводится режим передач управления подпрограмме, расположенной со строки n, при любых столкновениях спрайтов. Этот режим включается и выключается соответственно параметрами *ON* и *OFF* команды

$$SPRITE\begin{Bmatrix} ON \\ OFF \end{Bmatrix}$$

Пример.

```
10 COLOR 1,15,8:SCREEN 1,3
20 ON SPRITE GOSUB 90: SPRITE ON
30 X$="чертенок"
40 SPRITE$(0)=X$+X$+X$+X$
50 FOR Y=0TO255
60 PUT SPRITE 0,(Y,70),4,50
70 PUT SPRITE 1,(120,Y),13,50
80 NEXT:GOTO 50
90 '  ПОДПРОГРАММА
100 SPRITE OFF
110 PRINT "столкновение"
120 RETURN
```

8. ПРОГРАММА "БАЗОВЫЕ СТРУКТУРЫ"

Основные структуры действий, используемые при описании алгоритмов, это

1. последовательность,
2. ветвление,
3. цикл.

На рисунках 8.1 – 8.5 приведены блок-схемы этих структур.

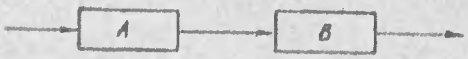


Рис. 8.1 Структура "последовательность"

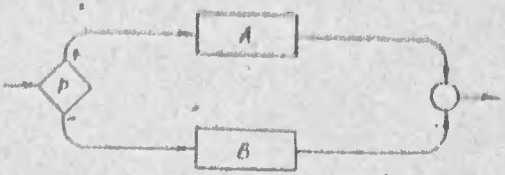


Рис. 8.2 Структура "ветвления" (полная)

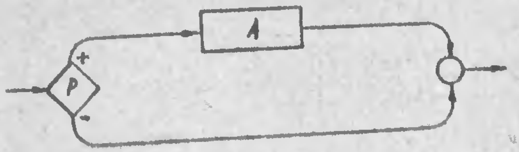


Рис. 8.3 Структура "ветвления" (неполная)

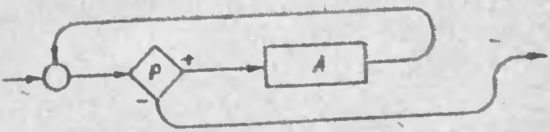


Рис. 8.4 Структура цикла (цикл "пока")

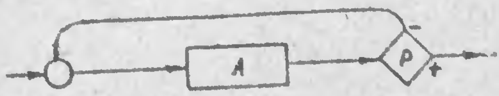


Рис. 8.5 Структура цикла (цикл "до")

По программе "базовые структуры" организуется рисование всех пяти схем, указанных на рис. 8.I - 8.5 при нажатии на функциональные клавиши *F1-F5*. Завершается работа с программой при нажатии на любую из клавиш *F6-F10*.

Схемы, графики, рисунки или их отдельные фрагменты, выводимые на графические экраны обычно нуждаются в обозначениях и поясняющих сообщениях. Сформировать их оператором *PRINT* невозможно. На экранах 2 и 3 последний не действует. Поэтому, например, буквы *A*, *B* и *P*, которыми обозначены арифметические и логические блоки в схемах, индицируемых программой "базовые структуры", рисуются оператором *DRAW*.

Вывод текстов на экраны 2 и 3 можно организовать и иными путями. Один из них демонстрирует следующий фрагмент.

```
10 COLOR 1,1[illegible],[illegible]:SCREEN 2
20 OPEN "GRP:"FOR OUTPUT AS#1
30 PSET(100,100),15
40 PRINT #1,"[illegible]"
50 FOR T=1TO5000:NEXT
```

Этот способ позволяет выводить символы фиксированного формата в блоках 8x8. Если требуется регулировать и размер букв, и

```
10 REM  Б А З О В Ы Е   С Т Р У К Т У Р Ы
20 COLOR 1,15,8:SCREEN 0
30 REM  Б А З О В Ы Е   С Т Р У К Т У Р Ы
40 INPUT "ФОН (темный - 1,светлый - 2) ";X
50 IF X<>1ANDX<>2 THEN 40
60 IF X=1THEN 80
70 COLOR 1,15,8:SCREEN 2:GOTO90
80 COLOR 15,1,13:SCREEN 2
90 C$="h2f3g3e3"
100 F$="u8r30d16l30u8br30"
110 E$="e10f10g10h10"
120 H$="h3f3e3g3"
130 ON KEY GOSUB 300,160,240,370,450,600,600,600,600,600
140 FOR S=1 TO 10:KEY(S)ON:NEXT
150 GOTO150
160 REM СТРУКТУРА 'ВЕТВЛЕНИЕ (полное)'
170 CLS:PSET(10,95):GOSUB 530
180 DRAW"[illegible]"
190 GOSUB 500:GOSUB 570
200 X=141:Y=50:GOSUB 560
210 X=137:Y=134:GOSUB 570
220 X=69:Y=91:GOSUB 580
230 RETURN
240 REM СТРУКТУРА 'ВЕТВЛЕНИЕ (неполное)'
250 CLS:PSET(10,95):GOSUB 530:DRAW"l70"
260 GOSUB 540:GOSUB 570
270 X=141:Y=50:GOSUB 560
280 X=69:Y=91:GOSUB 580
290 RETURN
300 REM СТРУКТУРА 'ПОСЛЕДОВАТЕЛЬНОСТЬ'
310 CLS:PSET(10,95)
320 DRAW"s8r10Bc$;F$;r20xc$;"
330 DRAW"F$;r10xc$;"
340 X=79:Y=91:GOSUB 560
350 X=174:Y=91:GOSUB 570
360 RETURN
370 REM СТРУКТУРА 'ЦИКЛ (пока)'
380 CLS:PSET(10,95)
390 DRAW"[illegible]"
400 DRAW"[illegible]"
410 DRAW"[illegible]"
420 X=141:Y=71:GOSUB 540
430 X=78:Y=91:GOSUB 580
440 RETURN
450 REM СТРУКТУРА 'ЦИКЛ (до)'
460 CLS:PSET(17,95)
470 DRAW"[illegible]"
480 DRAW"[illegible]"
490 DRAW"[illegible]"
500 X=70:Y=91:GOSUB 560
510 X=180:Y=91:GOSUB 580
520 RETURN
530 DRAW"[illegible]"
540 DRAW"[illegible]":RETURN
550 DRAW"[illegible]":RETURN
560 DRAW"[illegible]":RETURN
570 DRAW"[illegible]":RETURN
580 DRAW"[illegible]":RETURN
590 DRAW"[illegible]":RETURN
600 END
```

расстояние между ними, то можно рекомендовать прием, предложенный в программе "алфавит". Там предполагается, что в массиве A$( ) значение каждого элемента A$(n) языком GWL задает рисование символа с десятичным кодом ASCII , равным n . Для иллюстрации в программе показано формирование элементов "К", "Д", "А" и "С".

```
10 REM   А   Л   Ф   А   В   И   Т
20 REM (X,Y)- ЛЕВЫЙ ВЕРХНИЙ УГОЛ
30 REM        ПЕРВОЙ БУКВЫ ТЕКСТА
40 REM  S   - РАЗМЕР БУКВ
50 REM  L   - РАССТОЯНИЕ МЕЖДУ БУКВАМИ
60 COLOR 1,15,8:SCREEN 2
70 CLEAR 1000:GOSUB 150
80 X=70:Y=80:S=2:L=9:S=4*S
90 TE$="КЛАСС":P$="s=S;bm=X;,=Y;":GOSUB 110
100 GOTO 100
110 FOR K=1 TO LEN(TE$)
120 Z$=MID$(STR$(ASC(MID$(TE$,K))),2)
130 P$=P$+"x#$("+Z$+");"+"BR=L;"
140 NEXT:DRAW "XP$;":RETURN
150 DIM A$(255)
160 A$(32)="BR1"
170 A$(225)="BD6U4E2F2D4U2L4BE4BR1"
180 A$(226)="BD6E1UTE2R1D6U6BR2"
190 A$(235)="D6U2R1F3H3E3DR1"
200 A$(243)="BR1R2F1BD4G1L2H1U4E1BR4"
210 RETURN
220 RUN
```

№	КОМАНДА, ФУНКЦИЯ	СТР	№	КОМАНДА, ФУНКЦИЯ	СТР
1	2	3	1	2	3
1	CIRCLE	7	8	PRESET	4
2	DRAW	9	9	PSET	5
3	GMD	10	10	PUT SPRITE	15
4	LINE	8	11	SCREEN	14
5	ON SPRITE GOSUB	17	12	SPRITE OFF	16
6	PAINT	12	13	SPRITE ON	16
7	POINT	6	14	SPRITE$	13

ТАБЛИЦА А. УКАЗАТЕЛЬ КОМАНД И ФУНКЦИЙ

Содержание