ГАЙВАН А.В.

КОМПЬЮТЕРНАЯ ГРАФИКА И ЧИСЛЕННЫЕ МЕТОДЫ

ZX SPECTRUM

КОМПЬЮТЕРНАЯ ГРАФИКА И ЧИСЛЕННЫЕ МЕТОДЫ

BASIC

Продукция фирмы "ВА Принт"

Москва

1994

ББК 32.973.2-01
К42

По вопросам оптового приобретения продукции издательства "ВА Принт" обращайтесь по телефонам:

в Москве
(095) * 491-24-06
(095) * 415-55-52 (рассылка почтой)
в Санкт - Петербурге
(812) * 254-68-63
в Ростове - на - Дону
(863) * 252-09-86

или по адресу: г. **Москва, 123362, а/я 26**

ISBN 5-85734-032-2

ПРЕДИСЛОВИЕ

Дорогой читатель, Вы наверняка уже познакомились со многими программами при работе на своем компьютере, видели сложные графические изображения, решали какие-то задачи, играли в компьютерные игры. Заинтересовавшись "устройством" готовых программ, Вы далеко не всегда сможете в них разобраться — либо программы очень сложные и в них отсутствуют комментарии, либо у Вас имеются только загрузочные модули программ в кодах компьютера.

Книга, которую Вы держите сейчас в руках, поможет Вам войти в удивительный мир компьютерной графики, познакомиться с несложными численными методами решения задач. Пролистав эту книгу, Вы увидите множество рисунков, которые почти все созданы компьютером по приведенным в книге программам.

В качестве языка программирования выбран бейсик, который является широко распространенным языком, реализованным практически на всех типах персональных компьютеров. И хотя представленные в книге программы написаны для SPECTRUM'а, они легко могут быть модифицированы для любого другого компьютера. Кроме того, бейсик является хорошим средством для записи алгоритмов и позволяет перевести программы на любой другой язык программирования.

Простота бейсика дает возможность даже начинающим быстро освоиться в мире программ. Большинство рассмотренных в книге программ работает в диалоговом режиме, позволяя Вам путем задания различных комбинаций входных параметров проводить исследования при решении математических задач, создавать множество рисунков на плоскости и в трехмерном пространстве.

Все приведенные программы являются автономными и для своего выполнения не требуют загрузки каких-либо других программных модулей. Вы можете пропустить некоторые разделы книги без заметного ущерба для восприятия материала последующих разделов.

В первой главе рассмотрены способы построения различных элементов, применяемых в компьютерной графике, таких как спирали, различные узоры, мозаики и калейдоскопы, гистограммы и графики функций.

Вторая глава посвящена символьной графике, примеры применения которой приведены в этой главе, а также в третьей и шестой главах книги.

В третьей главе приведены примеры создания движущихся объектов на экране дисплея по принципу мультипликации.

Четвертая глава освещает вопросы реализации графики трехмерного пространства с помощью преобразования координат и некоторых искусственных приемов.

В пятой главе дается описание способов создания звуковых эффектов с помощью компьютера; приведенные программы позволяют воспроизводить и сочинять различные мелодии.

Шестая глава посвящена проблеме разработки компьютерных игр; приведены и подробно проанализированы программы двух игр.

В седьмой главе рассмотрены некоторые численные методы решения математических задач.

Глава 1

ВИДЫ КОМПЬЮТЕРНОЙ ГРАФИКИ

В этой главе Вы познакомитесь с различными способами создания рисунков на плоскости.

Сначала рассмотрим три основных графических оператора: PLOT, DRAW и CIRCLE, с помощью которых строится большинство рисунков. С остальными графическими операторами мы будем знакомиться по мере их применения в программах.

Графические операторы

Графический экран SPECTRUM'а содержит 22 строки (с 0 по 21) по 32 символа (с 0 по 31) в каждой, что составляет 22x32 = 704 символьные позиции. Каждая символьная позиция представляется квадратом 8x8 точек, называемых пикселями.

Пиксель задается своими координатами (X,Y). Первое число задает координату X, то есть удаление пикселя от левой границы экрана, а второе — координату Y, то есть удаление пикселя от нижней границы экрана. Углы графического поля имеют следующие координаты:

- *левый нижний — (0,0);*
- *правый нижний — (255,0);*
- *левый верхний — (0,175);*
- *правый верхний — (255,175).*

Оператор PLOT X,Y вызывает высвечивание закрашивающим цветом (INK) пикселя с координатами (X,Y).

Оператор PLOT широко применяется для точного вычерчивания функциональных кривых, для указания особых точек рисунков. С его помощью можно чертить непрерывные линии.

На листинге 1.1 приведена программа, демонстрирующая применение оператора PLOT. Результаты работы этой программы вы видите на рисунке 1.1.

Listing 1.1

```
10  REM 'PLOT'
15  PRINT AT 0,7; "USE OF THE 'PLOT'"
20  REM FUNCTION'S CURVE
30  FOR X=40 TO 200: PLOT X,120:NEXT X: PRINT AT 6,26;"X"
40  FOR Y=85 TO 155: PLOT 40,Y:NEXT Y: PRINT AT 1,5;"Y"
50  FOR X=0 TO 4*PI STEP PI /50
60  LET Y=35*SIN X
```

```
70  PLOT 40+12*X,120+Y
80  NEXT X
90  REM FRAME
100 FOR X=40 TO 200: PLOT X,10:PLOT X,70: NEXT X
110 FOR Y=10 TO 70: PLOT 40,Y:PLOT 200,Y: NEXT Y
120 REM RANDOM POINTS
130 FOR I=1 TO 500
140 LET X=41+RND *159: LET Y=11+RND * 59
150 PLOT X,Y:NEXT I
160 PRINT AT 6,2;"A": PRINT AT 16,2;"B"
170 STOP
```

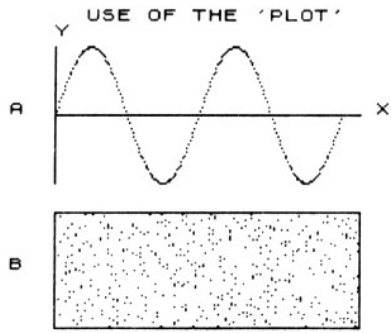


Рис. 1.1

В 20 - 80 строках программы организовано построение функции $y = 35sinx$ на двух ее периодах по 200 точкам (рис. 1.1а). Коэффициент 12 в строке 70 введен для растяжения графика по оси X.

Обратите внимание на то, что координатные оси имеют вид сплошных линий, хотя нарисованы по пикселям с помощью оператора PLOT (30 и 40 строки программы), который, правда, выполняет эту задачу медленнее, чем оператор DRAW.

Операторы в 90 - 110 строках выполняют построение рамки для рисунка 1.1б аналогично тому, как строились координатные оси.

В 120 - 150 строках реализовано построение 500 точек, координаты которых задаются по случайному закону с помощью функции RND в пределах нарисованной рамки. Функция RND выдает случайные числа, равномерно распределенные на интервале (0,1), а коэффициенты 159 и 59 (строка 140) "вытягивают" эти числа по полю, ограниченному рамкой.

Оператор DRAW X,Y рисует прямую линию от начального пикселя (пикселя, на котором завершился последний из операторов PLOT, DRAW или CIRCLE), до пикселя с приращением X по горизонтали и приращением Y по вертикали относительно начального пикселя. Начальный пиксель называют еще ***текущей графической позицией***.

Оператор DRAW X,Y,A рисует линию от текущей графической позиции до точки с приращениями (X,Y) по дуге в A радиан. При A < 0 дуга рисуется по часовой стрелке относительно центра дуги, при A > 0 — против часовой стрелки. Если A = 0, то рисуется прямая линия (аналог DRAW X,Y).

На листинге 1.2 Вы видите программу примера использования оператора DRAW, с помощью которой получен рис. 1.2.

Listing 1.2

```
10  REM 'DRAW'
15  PRINT AT 0,7; "USE OF THE 'DRAW'"
20  REM LINE
30  FOR Y=0 TO 140 STEP 20: PLOT 10,10:DRAW 100,Y:
    NEXT Y
40  FOR X=0 TO 90 STEP 20: PLOT 10,10:DRAW X,140:
    NEXT X
50  REM ARC
60  FOR A=-3.5 TO 3.5 STEP 0.5
70  PLOT 140,80:DRAW 110,0,A
80  NEXT A
90  PRINT AT 21, 7;"A": PRINT AT 21,24;"B"
100 STOP
```

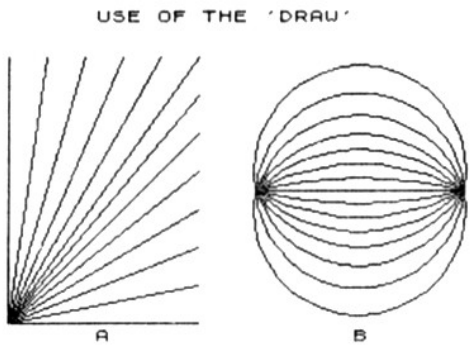


Рис. 1.2

С помощью 20 - 40 строк программы строится семейство прямых линий, выходящих из точки (10,10) с приращениями 100 пикселей по горизонтали и по 20 пикселей по вертикали (строка 30) и 140 пикселей по вертикали и по 20 пикселей по горизонтали (строка 40). В результате получается изображение, приведенное на рис. 1.2а.

В 50 - 60 строках реализовано построение семейства дуг, соединяющих точки (140, 80) и (250, 80). Обратите внимание на процесс построения верхних дуг (A < 0) и нижних дуг (A > 0). Результат построения Вы видите на рис. 1.2б.

Оператор CIRCLE X,Y,R рисует окружность радиуса R пикселей с центром в точке (X,Y).

На листинге 1.3 представлена программа применения оператора CIRCLE для получения некоторых графических элементов. Результат работы программы Вы видите на рис. 1.3.

Listing 1.3

```
10  REM 'CIRCLE'
15  PRINT AT 0,6; "USE OF THE 'CIRCLE'"
20  FOR R=0 TO 30 STEP 5
30  CIRCLE 45,122,R:NEXT R
40  FOR I=1 TO 10
50  CIRCLE 45,42,40/EXP (0.3*I): NEXT I
60  FOR I=0 TO 80 STEP 8
70  CIRCLE 125,45+I,35:NEXT I
80  FOR I=0 TO 104 STEP 8
90  CIRCLE 205,45+I,35-0.3*I: NEXT I
100 PRINT AT 6, 0;"A": PRINT AT 16, 0;"B":
    PRINT AT 21,15;"C": PRINT AT 21,25;"D"
110 STOP
```

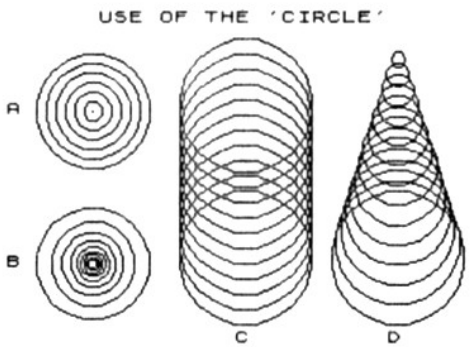


Рис. 1.3

С помощью 20 - 30 строк рисуется семейство концентрических окружностей с равномерно возрастающим радиусом (рис. 1.3а).

В 40 - 50 строках реализовано построение концентрических окружностей с радиусом, изменяющимся по экспоненте (рис. 1.3б).

Операторы в 60 - 70 строках строят семейство окружностей одного радиуса, центры которых равномерно смещаются по оси Y (рис. 1.3в).

Построение группы окружностей с постоянно убывающим радиусом, центры которых равномерно смещаются по оси Y (рис. 1.3г) реализовано в 80 - 90 строках программы.

Программы, приведенные в этом разделе, являлись демонстрационными и работали без диалога с пользователями. Целью их написания было показать некоторые приемы использования графических операторов в последующих программах.

Многоугольники

В этом разделе Вы познакомитесь с двумя программами. Первая из программ позволяет строить правильные многоугольники, а вторая — многоугольники (полигоны) произвольной формы.

На листинге 1.4 приведена программа построения правильных многоугольников.

Listing 1.4

```
10  REM N-GON
15  PRINT AT 0,13;"N-GON"
20  INPUT "N(3-20)=";N; " R(30-80)=";R
30  PRINT AT 0,25;"N=";N: PRINT AT 1,25;"R=";R
40  LET DF=2*PI /N
50  LET X=127:LET Y=80+R: PLOT X,Y
60  FOR F=0 TO 2*PI +0.1 STEP DF
70  LET XT=127+R*SIN F: LET YT= 80+R*COS F
80  DRAW XT-X,YT-Y
90  LET X=XT:LET Y=YT
100 NEXT F
110 INPUT "S (Y/N)?";A$
120 IF A$<>"Y" AND A$<>"y" THEN GO TO 150
130 LET S=0.5*N*R*R*SIN DF
140 PRINT AT 2,25;"S=";INT S
150 STOP
```

Программа имеет 2 входных параметра:

- *N — число вершин многоугольника;*
- *R — радиус окружности, описанной вокруг многоугольника.*

Алгоритм построения заключается в том, что окружность радиуса R разбивается на N равных дуг, концы которых последовательно соединяются друг с другом, образуя вершины правильного N-угольника. Сама окружность при этом не строится. При желании можно вычислить площадь построенного многоугольника, утвердительно ответив на вопрос "S (Y/N)". Пример работы программы вы видите на рис. 1.4.

Построение многоугольника начинается с верхней вершины заданием текущей графической позиции (127,80+R) оператором PLOT (строка 50) и осуществляется оператором DRAW (строка 80). Обратите внимание на дополнительную величину 0.1 в 60 строке — она необходима для того, чтобы последняя вершина N-угольника соединилась с первой вершиной.

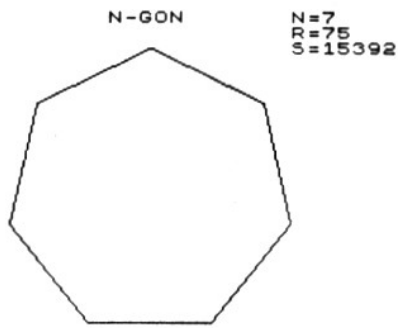


Рис. 1.4

Программа построения произвольного полигона приведена на листинге 1.5.

Listing 1.5

```
10 REM POLIGON
20 INPUT "N=";N: DIM X(N):DIM Y(N)
30 FOR I=1 TO N: INPUT " X(I)=";X(I); " Y(I)=";Y(I):
   PRINT " X(";I;")=";X(I); " Y(";I;")=";Y(I): NEXT I
40 INPUT "DRAW ?";H$
50 IF INKEY$ ="" THEN GO TO 50
60 CLS : PRINT AT 0,12;"POLIGON": PRINT AT 0,27;"N=";N
70 FOR I=1 TO N-1
80 PLOT X(I),Y(I): DRAW X(I+1)-X(I), Y(I+1)-Y(I)
90 IF I=N-1 THEN DRAW X(1)-X(N),Y(1)-Y(N)
100 NEXT I
110 STOP
```

После запуска программы запрашивается число вершин полигона N, а затем поочередно координаты всех вершин, причем для контроля эти координаты выводятся на экран дисплея. После ввода всей информации программа спрашивает "DRAW?" ("Рисовать?"). При нажатии клавиши "Enter" или при вводе любого символа программа очищает экран оператором CLS и строит полигон по заданным координатам вершин. Способ построения полигона отличается от способа построения правильного многоугольника — оператор PLOT задает текущую графическую позицию для каждой вершины, которая соединяется с соседней вершиной оператором DRAW (строка 80). В 90 строке организовано соединение последней вершины полигона с его первой вершиной. Пример построения полигона вы видите на рис. 1.5.

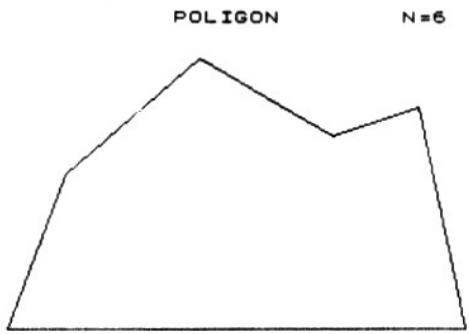


Рис. 1.5

Эллипсы

Как известно, эллипс может быть задан следующими параметрическими уравнениями:

$$x = x_0 + a \cdot sin\ \varphi\ ,$$

$$y = y_0 + b \cdot cos\ \varphi\ ;$$

где: (x_0, y_0) - координаты центра эллипса;
a, b - полуоси эллипса;
φ - угол, изменяющийся в пределах от 0 до 2π радиан.

На листинге 1.6 приведена программа построения эллипса по 300 точкам с помощью оператора PLOT по заданным полуосям A и B, расположенным параллельно координатным осям X и Y соответственно. Центр эллипса помещен в точку (128,82). Пример построения эллипса с помощью данной программы Вы видите на рис. 1.6.

Listing 1.6

```
10  REM ELLIPSE
15  PRINT AT 0,12;"ELLIPSE"
20  INPUT " A(10-125)=";A;" B(10-80)="; B
30  PRINT AT 1,26;"A=";A: PRINT AT 2,26;"B=";B
40  FOR Q=0 TO 2*PI STEP PI /150
50  LET X=128+A*SIN Q: LET Y= 82+B*COS Q
60  PLOT X,Y:NEXT Q
70  STOP
```

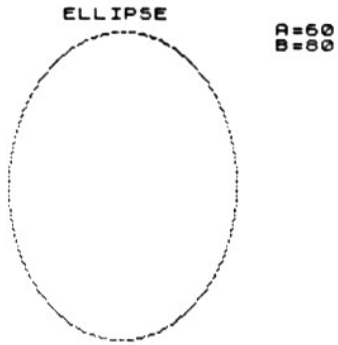


Рис. 1.6

Программа на листинге 1.7 строит эллипс с применением оператора DRAW.

Изображение эллипса формируется по отрезкам прямых, поэтому для получения его приемлемого вида вполне достаточно 40 точек. Более того, с увеличением числа точек (уменьшением шага DF) изображение эллипса ухудшается и все больше приближается к многоугольнику. В этом состоит одно из главных отличий способа построения функциональных кривых аппроксимирующими отрезками прямых от способа их построения по точкам.

Пример работы программы Вы видите на рис. 1.7.

Listing 1.7

```
10  REM ELLIPSE
15  PRINT AT 0,12;"ELLIPSE"
20  INPUT "A(10-125)=";A; " B(10-80)="; B
30  PRINT AT 1,26;"A=";A: PRINT AT 2,26;"B=";B
40  LET DF=PI/20
50  LET X=128:LET Y=84+B: PLOT X,Y
```

```
60  FOR F=0 TO 2*PI +.01 STEP DF
70  LET X1=128+A*SIN F: LET Y1= 84+B*COS F
80  DRAW X1-X,Y1-Y: LET X=X1:LET Y=Y1
90  NEXT F
100 STOP
```

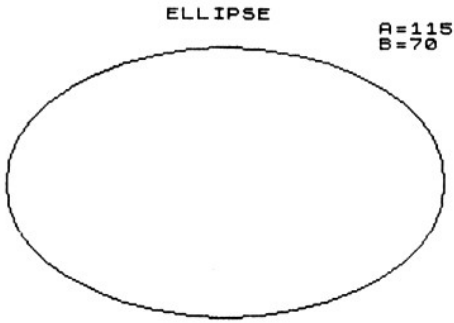


Рис. 1.7

Спирали

С давних времен в искусстве, дизайне люди применяли спирали — простые, сходящиеся, расходящиеся, составленные из дуг окружностей, стилизованные и т.д. Рассмотрим некоторые способы построения спиралей.

Простая спираль в параметрической форме имеет вид:

$$x = r \cdot sin\ \alpha\ ,$$

$$y = r \cdot cos\ \alpha\ ,$$

$$r = k \cdot \alpha\ ;$$

где: k - постоянный коэффициент;

α - угол, изменяющийся в заданных пределах;

r - текущий радиус спирали.

На листинге 1.8 приведена программа, рисующая по точкам простую спираль, заданную уравнениями:

X = 127 + 0.8 * A * SIN A ,

Y = 87 + 0.8 * A * COS A ;

где: A - угол, изменяющийся в пределах от 0 до 90 радиан с шагом 0.1;

(127,87) - координаты центра спирали.

Соответствующее изображение спирали Вы видите на рис 1.8.

Listing 1.8

```
10  REM SIMPLE SPIRAL
20  FOR A=0 TO 90 STEP 0.1
30  LET X=SIN A: LET Y=COS A
40  LET R=0.8*A
50  PLOT 127+R*X,87+R*Y
60  NEXT A
70  STOP
```

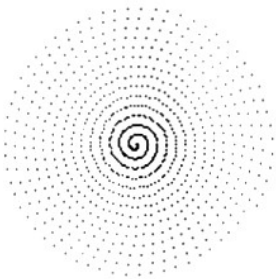


Рис. 1.8

Недостатком приведенной выше программы является ее жесткость, т.е. невозможность варьировать какие-либо параметры спирали. Следующая программа (листинг 1.9) позволяет строить простые спирали по параметрам, задаваемым Вами. Причем в данной программе изменен принцип определения спирали и спираль характеризуется следующими параметрами:

- *X0, Y0 — координаты центра спирали;*
- *R — максимальный радиус спирали;*
- *N — число витков спирали;*
- *GAMMA — начальный угол, который спираль образует с положительным направлением оси X.*

Listing 1.9

```
10  REM SPIRAL
15  PRINT AT 0,8; "SIMPLE SPIRAL"
20  INPUT "X0=";X0;" Y0=";Y0:
    INPUT "R=";R;" N=";N; " GAMMA=";GAMMA
30  PRINT AT 0,26;"X0=";X0: PRINT AT 1,26;"Y0=";Y0:
    PRINT AT 2,26;"R =";R : PRINT AT 3,26;"N =";N :
```

```
    PRINT AT 4,26;"G =";GAMMA
40  LET A=GAMMA: LET AD=PI /50: LET RD=R/(N*100)
50  LET X1=X0:LET Y1=Y0
60  FOR I=RD TO R STEP RD=ЧЭ
70  LET X=X0+I*SIN A: LET Y=Y0+I*COS A
80  PLOT X1,Y1: DRAW X-X1,Y-Y1
90  LET X1=X:LET Y1=Y: LET A=A+AD
100 NEXT I
110 STOP
```

Спираль строится аппроксимирующими отрезками прямых. В 40 строке программы определяются шаг приращения угла AD = PI/50 и шаг приращения радиуса RD = R/(N*100), а в 60 - 100 строках организован цикл построения спирали по заданным параметрам. Пример построения спирали по этой программе представлен на рис. 1.9.

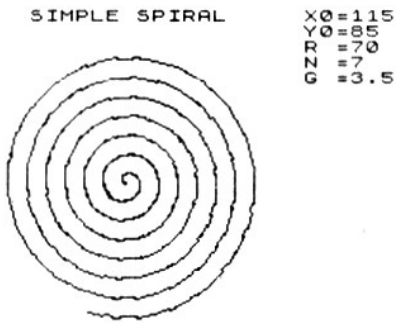


Рис. 1. 9

Расходящаяся спираль в параметрической форме имеет вид

$$x = r \cdot \sin \alpha ,$$

$$y = r \cdot \cos \alpha ,$$

$$r = k1 \cdot e^{\alpha / k2} ;$$

где: *k1* и *k2* - постоянные коэффициенты;

α - угол, изменяющийся в заданных пределах;

r - текущий радиус спирали.

На листинге 1.10 приведена программа, позволяющая построить расходящуюся спираль с параметрами *k1* = 0.1, *k2* = 30, при изменении α от 0 до 200 радиан. Обратите внимание на то, что программа позволяет построить именно одну спираль.

Listing 1.10

```
10 REM SPIRAL
15 PRINT AT 0,7; "DIVERGENT SPIRAL"
20 INPUT "FORM=";F$;" STEP=";S
30 PRINT AT 0,26;"F=";F$: PRINT AT 1,26;"S=";S
40 LET X1=120:LET Y1=80
50 FOR A=0 TO 200 STEP S
60 LET R=0.1*EXP (A/30): LET X=120+R*SIN A:
   LET Y= 80+R*COS A
70 IF F$="P" OR F$="p" THEN GO TO 90
80 PLOT X1,Y1: DRAW X-X1,Y-Y1: LET X1=X:LET Y1=Y:
   GO TO 100
90 PLOT X,Y
100 NEXT A
110 STOP
```

А сейчас Вы убедитесь, как сильно восприятие изображения зависит от способа его построения. В программу введены два входных параметра:

- *F$ — символьная переменная, задающая форму построения спирали — по точкам (P) или по отрезкам (L);*
- *S — шаг изменения параметра A, определяющий число точек, по которым строится спираль.*

При работе с программой надо иметь в виду, что первые витки расходящейся спирали имеют очень маленький радиус и рисуются для наблюдателя медленно.

На рис. 1.10а представлен пример классической расходящейся спирали, построенной по отрезкам прямых — внутренние витки плотно сжаты, по мере возрастания радиуса расстояние между витков увеличивается.

Спираль на рис. 1.10б строится по точкам с достаточно большим шагом S = 0.78. Возникает иллюзия построения восьми спиралей, расходящихся из общей точки.

Рис. 1.10а

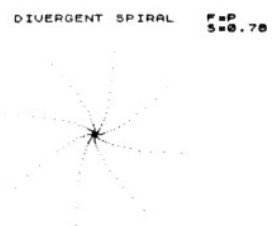


Рис. 1.10б

Спираль на рис. 1.10в строится с шагом S = 1.57 по отрезкам прямых — создается стилизованное изображение расходящейся спирали в виде скрученных четырехугольников.

Еще один пример стилизованного изображения расходящейся спирали Вы видите на рис. 1.10г.

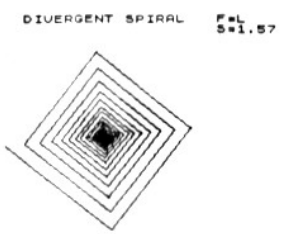


Рис. 1.10в

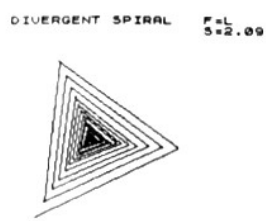


Рис. 1.10г

Сходящаяся спираль в параметрической форме имеет вид

$$x = r \cdot \sin \alpha ,$$

$$y = r \cdot \cos \alpha ,$$

$$r = k1 \cdot (1 \cdot e^{-\alpha / k2}) ;$$

где параметры аналогичны параметрам расходящейся спирали.

На листинге 1.11 приведена программа построения сходящейся спирали с параметрами $k1 = 70$, $k2 = 25$, при изменении α от 0 до 150 радиан.

Listing 1.11

```
10  REM SPIRAL
15  PRINT AT 0,7; "CONVERGENT SPIRAL"
20  INPUT "FORM=";F$;" STEP=";S
30  PRINT AT 0,26;"F=";F$: PRINT AT 1,26;"S=";S
40  LET X1=120:LET Y1=80
50  FOR A=0 TO 150 STEP S
60  LET R=70*(1-EXP (-A/25)): LET X=120+R*SIN A:
    LET Y= 80+R*COS A
70  IF F$="P" OR F$="p" THEN GO TO 90
80  PLOT X1,Y1: DRAW X-X1,Y-Y1: LET X1=X:LET Y1=Y:
    GO TO 100
90  PLOT X,Y
100 NEXT A
110 STOP
```

В данной программе имеются два входных параметра, аналогичные параметрам программы расходящейся спирали. Примеры различных способов изображения одной сходящейся спирали Вы видите на рис. 1.11а - 1.11д.

Рис. 1.11а

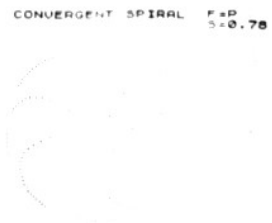


Рис. 1.11б

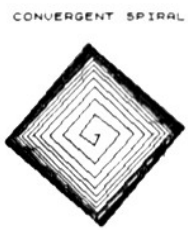


Рис. 1.11в

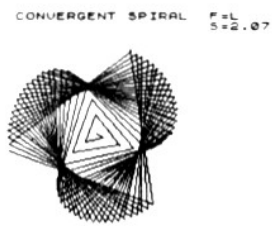


Рис. 1.11г

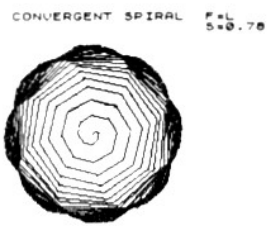


Рис. 1.11д

Перепишите приведенные программы построения спиралей так, чтобы можно было задавать различные значения коэффициентов *k1* и *k2*, диапазоны изменения угла α. Помимо рассмотренных, существует ряд других видов спиралей. Найдите их описания или изображения и попробуйте составить соответствующие программы, рисующие эти спирали.

Узоры

В этом разделе рассматриваются способы создания довольно сложных графических изображений с помощью простых алгоритмов, реализованных соответствующими программами.

Программа, представленная на листинге 1.12, создает узор из окружностей.

Listing 1.12

```
10 REM PATTERN
15 PRINT AT 0,12;"PATTERN"
20 INPUT "N=";N;" K=";K; " R(10-30)=";R
30 PRINT AT 0,26;"N=";N: PRINT AT 1,26;"K=";K:
   PRINT AT 2,26;"R=";R
40 LET A=0:LET DA=2*PI /N: LET MK=66-R:LET DM=MK/K
50 FOR M=0 TO MK STEP DM
60 FOR A=0 TO 2*PI STEP DA
70 LET X1=M*COS A: LET Y1=M*SIN A
80 CIRCLE 128+X1,88+Y1,R
90 NEXT A:NEXT M
100 STOP
```

Окружности радиуса R расходятся из общего центра по N направлениям, число окружностей в каждом направлении — K. Этот простой алгоритм позволяет получить узоры, подобные приведенным на рис. 1.12а, 1.12б.

Цикл с управляющей переменной A организует построение N окружностей радиуса R, равноудаленных от общего центра. Цикл с управляющей переменной M обеспечивает изменение удаления N окружностей от общего центра.

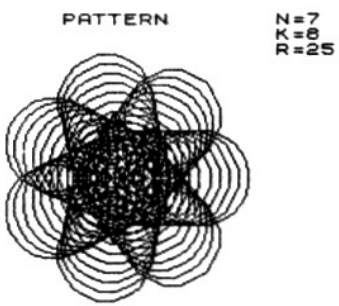


Рис. 1.12а

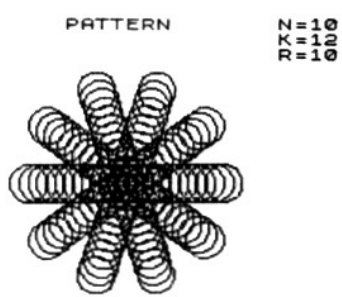


Рис. 1.12б

На листинге 1.13 приведена программа создания узоров, образуемых дугами.

Listing 1.13

```
10 REM PATTERN
15 PRINT AT 0,12;"PATTERN"
20 INPUT "F(1-4.5)=";F: PRINT AT 0,26;"F=";F
30 INPUT "S(0.1-1)=";S: PRINT AT 1,26;"S=";S
40 INPUT "L(20-110)=";L: PRINT AT 2,26;"L=";L
50 LET H=L/2 : OVER 1
60 FOR G=-F TO F+0.05 STEP S
70 PLOT 120,50:DRAW 0,L,G
80 PLOT 120-H,50+H:DRAW L,0,G
90 NEXT G
100 STOP
```

Узор строится с помощью оператора DRAW X,Y,G, который рисует линию от текущей графической позиции в точку с приращениями (X,Y) по дуге в G радиан.

В центральной части графического поля экрана выбираются два отрезка длиной L (один из них вертикальный, другой — горизонтальный), пересекающиеся в своих серединах. Концы этих отрезков соединяются дугами от -F до +F радиан с приращением S радиан при переходе к каждой новой дуге. Обратите внимание на использование оператора OVER 1 (50 строка). Запустите программу без этого оператора и посмотрите, что получится.

Примеры узоров, созданных этой программой, Вы видите на рис. 1.13а, 1.13б.

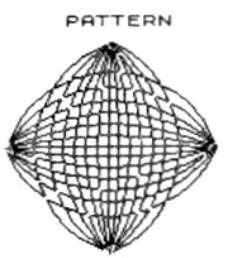


Рис. 1.13а

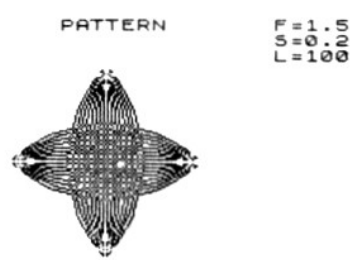


Рис. 1.13б

Программа, приведенная на листинге 1.14, демонстрирует еще один способ построения узоров с помощью дуг.

Listing 1.14

```
10 REM PATTERN
15 PRINT AT 0,13;"PATTERN"
20 INPUT "N=";N; " R=";R;" A=";A
30 PRINT AT 0,26;"N=";N: PRINT AT 1,26;"R=";R:
   PRINT AT 2,26;"A=";A
40 CIRCLE 127,82,R
50 LET DF=2*PI /N
60 FOR F=0 TO 2*PI STEP DF
70 PLOT 127,82
80 LET X=R/2*COS F: LET Y=R/2*SIN F
90 DRAW X,Y,-PI /A: DRAW X,Y, PI /A
100 NEXT F
110 STOP
```

На экране строится окружность радиуса R. Из центра этой окружности по N направлениям рисуются дуги, второй конец которых лежит на окружности. Дуги строятся так, что сначала они поворачивают на угол -PI/A, а затем на угол PI/A, где A является входным параметром программы. С помощью этой программы можно создавать графические изображения, подобные приведенным на рис. 1.14а, 1.14б.

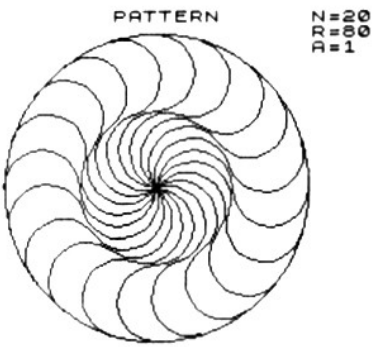


Рис. 1.14а

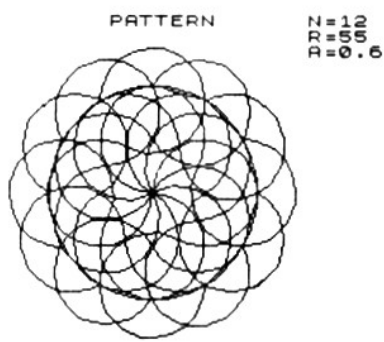


Рис. 1.14б

Рассмотрим способ создания узоров с помощью программы, представленной на листинге 1.15.

Listing 1.15

```
10  REM PATTERN
15  PRINT AT 0,12;"PATTERN"
20  INPUT "K /150-500/=";K: PRINT AT 0,26;"K=";K:
    INPUT "X /40-250/=";X: PRINT AT 1,26;"X=";X:
    INPUT "Y /40-160/=";Y: PRINT AT 2,26;"Y=";Y:
    INPUT "H /-Y/2_+Y/2/=";H: PRINT AT 3,26;"H=";H
30  LET X1=128:LET Y1= 82: LET SX=X/2:LET SY=Y/2:
    LET S=2*PI/K:OVER 1
40  FOR I=0 TO 2*PI +0.01 STEP S
50  LET X2=128+SX*SIN I
60  LET Y2= 82-SY*COS I
70  PLOT X1,Y1: DRAW X2-X1,Y2-Y1
80  PLOT X1,Y1: DRAW 128-X1,82+H-Y1
90  LET X1=X2:LET Y1=Y2
100 NEXT I
110 STOP
```

Для построения узоров используется эллипс с осями X и Y как геометрическое место K точек концов отрезков, соединяющих эти точки с выбранной точкой H на вертикальной оси эллипса.

При задании значений параметров K, X, Y, H в скобках указаны примерные диапазоны их изменения. Уберите оператор OVER 1 в 30 строке программы и посмотрите на соответствующую реакцию.

Примеры полученных с помощью программы узоров Вы видите на рис. 1.15а, 1.15б.

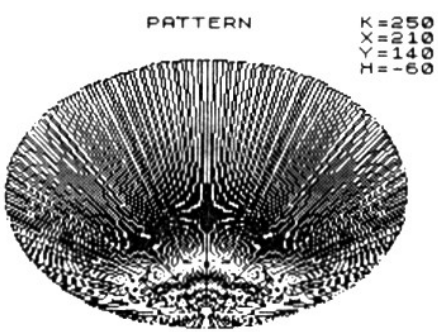


Рис. 1.15а

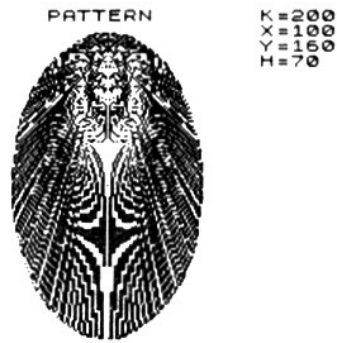


Рис. 1.15б

На листинге 1.16 Вы видите программу построения рисунков, использующих окружность радиуса R как геометрическое место точек расположения вершин правильного N-угольника, вписанного в эту окружность. Каждая вершина N-угольника последовательно соединяется со всеми остальными вершинами, в результате чего получается узор, подобный приведенному на рис. 1.16.

В 60 - 80 строках программы формируются массивы координат вершин многоугольника по введенным параметрам N и R, а в 90 - 110 строках реализуется алгоритм соединения вершин.

Listing 1.16

```
10  REM PATTERN
15  PRINT AT 0,13;"PATTERN"
20  INPUT "N(5-20)=";N; " R(30-80)=";R
30  PRINT AT 0,26;"N=";N: PRINT AT 1,26;"R=";R
40  DIM X(20):DIM Y(20)
50  LET DF=2*PI/N:LET F=0
60  FOR I=1 TO N
70  LET X(I)=127+R*SIN F: LET Y(I)= 82+R*COS F:
    LET F=F+DF
80  NEXT I
90  FOR I=1 TO N-1: FOR J=I+1 TO N
100 PLOT X(I),Y(I): DRAW X(J)-X(I),Y(J)-Y(I)
110 NEXT J:NEXT I
120 STOP
```

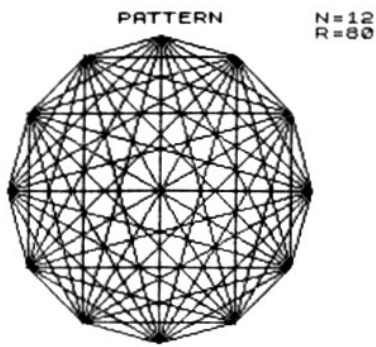


Рис. 1.16

Познакомимся еще с одним алгоритмом, реализованным программой на листинге 1.17.

Listing 1.17

```
10 REM ENVELOPE
15 PRINT AT 0,12;"ENVELOPE"
20 INPUT "SX(20-250)=";SX; " SY(20-160)=";SY
30 PRINT AT 1,26;"SX=";SX: PRINT AT 2,26;"SY=";SY
40 PLOT 128-SX/2,82:DRAW SX,0
50 PLOT 128,82-SY/2:DRAW 0,SY
60 INPUT "N(1-20)=";N: PRINT AT 3,26;"N =";N
70 FOR I=1 TO N
80 LET DX=INT (SX/2*I/N): LET DY=INT (SY/2*(N+1-I)/N)
90 PLOT 128+DX,82: DRAW -DX, DY:DRAW -DX,-DY:
   DRAW DX,-DY:DRAW DX, DY
100 NEXT I
110 STOP
```

Данная программа использует принцип развертки для построения рисунков и имеет три параметра:

- *SX — размер изображения по оси X;*
- *SY — размер изображения по оси Y;*
- *N — число точек разбиения отрезков SX/2 и SY/2.*

После задания параметров SX и SY плоскость экрана разбивается на четыре части двумя пересекающимися отрезками прямых (один из них горизонтальный длиной SX, другой — вертикальный длиной SY).

После задания параметра N каждая из четырех нарисованных полуосей разбивается на N частей. Четыре точки каждого I разбиения (строка 80) соединяются между собой (строка 90), образуя ромб. В результате повторения этой операции N раз строятся N ромбов, у которых постепенно увеличивается горизонтальная диагональ и уменьшается вертикальная.

Пример работы программы Вы видите на рис. 1.17.

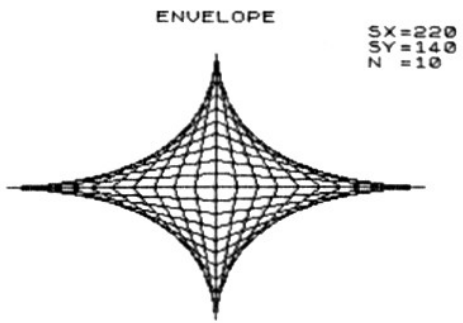


Рис. 1.17

Калейдоскопы

Многим знакома детская игрушка под названием калейдоскоп — зеркальная трехгранная призма, на одном из оснований которой расположены осколки цветного стекла, мелкие цветные бумажки и другие предметы. Медленно поворачивая калейдоскоп, можно получать разнообразные картинки за счет многократного отражения предметов внутри зеркальной призмы.

Рассмотрим несколько программ, позволяющих создавать постоянно меняющееся изображение на экране, реализуя принцип калейдоскопа по неповторимости рисунка.

Программа, приведенная на листинге 1.18, реализует калейдоскоп, который получается поворотом прямой линии с малым шагом X, изменяющимся по случайному закону при переходе к каждой новой картинке.

Listing 1.18

```
10 REM KALEIDOSCOPE OF LINES
20 FOR I=0 TO 7
30 BORDER I:PAPER I:INK 7-I: BRIGHT 0:CLS
40 LET X=1+3*RND
50 FOR J=0 TO 255 STEP X
60 PLOT J,0
70 DRAW OVER 1;255-2*J,175
80 NEXT J
90 FOR J=0 TO 175 STEP X
100 PLOT 0,J
110 DRAW OVER 1;255,175-2*J
120 NEXT J
130 INPUT "END? (Y/<ENTER>)";A$
140 IF A$="Y" OR A$="y" THEN GO TO 170
150 NEXT I
160 GO TO 20
170 INK 0:PAPER 7:BORDER 7
180 STOP
```

Каждый новый экран формируется со своими цветами фона, закрашивания и бордюра (рамки) экрана (30 строка программы). Значение шага X вычисляется в 40 строке с использованием функции RND. Изображение формируется с помощью двух циклов (50 - 80 строки и 90 - 120 строки) с применением оператора DRAW с включенным режимом OVER.

После создания каждой картинки программа запрашивает об окончании ("END?") — для продолжения работы нажмите "Enter", для окончания — введите символ Y.

Запрос введен в программу по двум причинам — он дает возможность рассмотреть каждое созданное изображение и осуществить нормальный выход из программы с восстановлением цветов (170 строка). Если запрос убрать, то программа будет работать бесконечно долго и выходить из нее придется с помощью прерывания, причем в этом случае сохранятся текущие цветовые параметры экрана. Данная программа позволяет получить узоры, один из которых Вы видите на рис. 1.18.

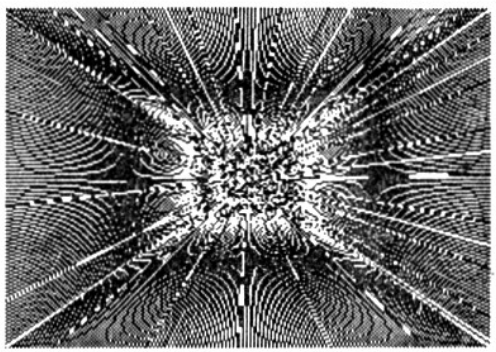


Рис. 1.18

Еще один калейдоскоп реализует программа, приведенная на листинге 1.19.

Listing 1.19

```
10  REM KALEIDOSCOPE
15  PRINT AT 0,5; "ROTATIONAL KALEIDOSCOPE"
20  CIRCLE 127,80,80
30  FOR Q=0 TO 1E30 STEP 0.02
40  LET X=75*SIN Q: LET Y=75*COS Q
50  OVER 1:PLOT 127+X,80+Y: DRAW -2*X,-2*Y
60  OVER 1:PLOT 127-X,80+Y: DRAW 2*X,-2*Y
70  NEXT Q
80  STOP
```

Калейдоскоп создается непрерывным вращением с шагом 0.02 радиана двух скрещивающихся прямых относительно центра окружности, причем прямые вращаются навстречу друг другу. Измените программу так, чтобы создать корректный выход из нее с помощью функции INKEY$.

На рис. 1.19 представлен один из множества узоров, полученных с помощью этой программы.

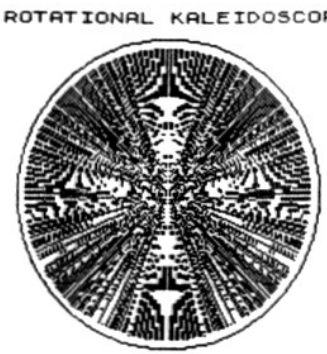


Рис. 1.19

На листинге 1.20 представлена программа реализации калейдоскопа с помощью окружностей.

Listing 1.20

```
10 REM KALEIDOSCOPE
15 PRINT AT 0,5; "KALEIDOSCOPE OF CIRCLES"
20 BORDER 5:PAPER 7:BRIGHT 0
30 FOR I=0 TO 7
40 FOR J=2 TO 66 STEP 2+I/2
50 OVER 1:INK I:PAPER 7-I
60 CIRCLE 105,67,J: CIRCLE 122,97,J: CIRCLE 139,67,J
70 NEXT J:BEEP 1,7:PAUSE 0
80 NEXT I
90 INK 0:PAPER 7:BORDER 7
100 STOP
```

В основе данного калейдоскопа лежит построение трех групп концентрических окружностей. Центры каждой группы окружностей расположены в вершинах правильного треугольника со стороной 34 пикселя. Радиусы окружностей изменяются от 2 до 66 с постоянным малым шагом для каждой пары цветов фона и закрашивания, при переходе к новой паре цветов шаг изменения радиуса возрастает.

В результате наложения одних окружностей на другие по мере увеличения их радиусов возникают различные узоры, причем особый эффект достигается при использовании цветного монитора.

После создания очередного узора раздается звуковой сигнал, и для продолжения работы программы достаточно нажать любую клавишу.

В 90 строке программы осуществляется установка нормальных цветов (черные знаки на белом фоне) перед завершением работы программы. На рис.1.20 Вы видите один из узоров, получаемых с помощью данной программы.

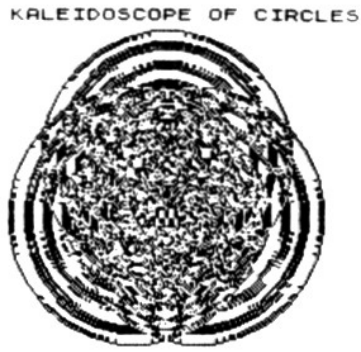


Рис. 1.20

Программа, приведенная на листинге 1.21, позволяет создать "музыкальный калейдоскоп", состоящий из символов с кодами от 32 до 91, выводимых на поле экрана. Причем как позиции каждого символа на экране, так и его код, цвет, цвет знакоместа, параметры яркости и мерцания выбираются случайно с помощью функции RND. Вывод каждого символа на экран сопровождается звуковым сигналом, продолжительность и тон которого также задаются случайным образом.

Listing 1.21

```
10  REM KALEIDOSCOPE
15  PRINT AT 0,3; "KALEIDOSCOPE OF CHARACTERS"
20  PAPER 6:BORDER 5
30  LET F=INT (2*RND ): LET B=INT (2*RND ):
    LET P=INT (8*RND ): LET I=INT (8*RND )
40  LET R=INT (20*RND )+2: LET C=INT (32*RND )
50  LET S=INT (60*RND ): LET T=0.2+0.3*RND : LET H=50*RND
60  PRINT AT R,C;FLASH F; BRIGHT B;PAPER P;INK I;
    CHR$ (32+S)
70  BEEP T,H
80  IF INKEY$ ="" THEN GO TO 30
90  INK 0:PAPER 7:BORDER 7: FLASH 0:BRIGHT 0
100 STOP
```

Программа работает до тех пор, пока Вы не нажмете любую клавишу. Перед завершением программы восстанавливаются нормальные параметры экрана.

На рис. 1.21 Вы видите пример работы программы (к сожалению, полиграфические возможности данного издания не позволяют хорошо воспроизвести этот рисунок).

Рис. 1.21

Рассмотрим еще одну программу (листинг 1.22), создающую калейдоскоп с помощью постоянно движущейся точки.

Listing 1.22

```
10 REM MOVING POINT
20 INK 7:PAPER 0:BORDER 6: OVER 1:CLS
30 LET DR=2: LET X=0: LET Y=0
40 LET DX=DR: LET DY=DR
50 PLOT X,Y
60 LET X=X+DX: LET Y=Y+DY
70 IF X>255-DR OR X<DR THEN LET DX=-DX
80 IF Y>175-DR OR Y<DR THEN LET DY=-DY
90 IF INKEY$ ="" THEN GO TO 50
100 INK 0:PAPER 7:BORDER 7
110 STOP
```

На темном фоне экрана движется светлая точка, которая отражается от границ графического поля экрана и оставляет след своего движения.

Изображение на экране насыщается до определенного предела, а затем точка начинает стирать свой собственный след, и этот процесс длится до полного исчезновения изображения.

После этого начинается новый цикл построения изображения. Выход из программы осуществляется нажатием любой клавиши. Пример работы программы Вы видите на рис. 1.22.

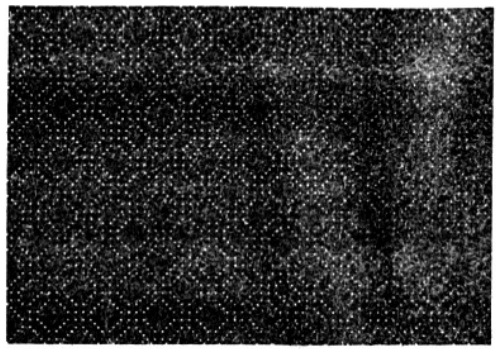


Рис. 1.22

Мозаики

Под мозаикой можно понимать, например, каждую из отдельных картинок, созданных программами 1.18 - 1.22. Рассмотрим мозаики, построенные с помощью графических элементов.

На листинге 1.23 представлена программа построения одной из таких мозаик.

Listing 1.23

```
10  REM MOSAIC
20  FOR Y=8 TO 167 STEP 16
30  FOR X=6 TO 243 STEP 12
40  GOSUB 60
50  NEXT X:NEXT Y:STOP
60  PLOT X,Y: DRAW 6, 8: DRAW 6,-8: DRAW -6,-8:
    DRAW -6, 8: DRAW 6, 8,2:DRAW 6,-8,2:
    DRAW -6,-8,2:DRAW -6, 8,2
70  RETURN
```

Рассмотрим алгоритм создания мозаики. В нижнем левом углу экрана рисуется ромб с диагоналями 12 пикселей (по оси X) и 16 пикселей (по оси Y). Соседние вершины ромба поочередно соединяются дугами в 2 радиана.

Затем эта процедура повторяется таким образом, что каждый новый ромб рисуется вплотную к предыдущему в порядке слева-направо и снизу-вверх по полю экрана до тех пор, пока рисунок не заполнит весь экран. Полученное при этом изображение Вы видите на рис. 1.23.

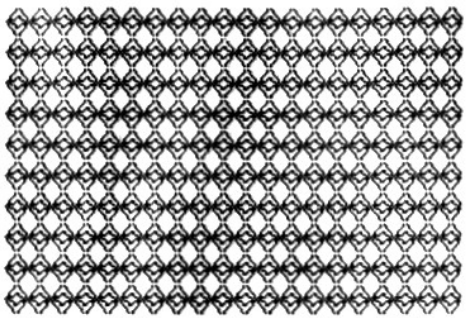


Рис. 1.23

Давайте теперь модифицируем эту программу так, чтобы диагонали ромбов и параметр дуги задавались в диалоговом режиме, и посмотрим, что получится. Модифицированная программа приведена на листинге 1.24.

Listing 1.24

```
10 REM MOSAIC
15 PRINT AT 0,8; "VARIABLE MOSAIC"
20 INPUT "SX /5-50/="; SX: PRINT AT 21,0;"SX=";SX:
   INPUT "SY /5-40/="; SY: PRINT AT 21,7;"SY=";SY:
   INPUT "R(0.5-11.5)=";R: PRINT AT 21,14;"R="; R
30 FOR Y=20+SY/2 TO 155-SY STEP SY
40 FOR X=5+SX/2 TO 245-SX STEP SX
50 GOSUB 80
60 NEXT X:NEXT Y
70 STOP
80 PLOT X,Y: DRAW SX/2, SY/2: DRAW SX/2,-SY/2:
   DRAW -SX/2,-SY/2: DRAW -SX/2, SY/2
90 DRAW SX/2, SY/2,R: DRAW SX/2,-SY/2,R:
   DRAW -SX/2,-SY/2,R: DRAW -SX/2, SY/2,R
100 RETURN
```

Программа имеет три входных параметра: **SX** — диагональ ромба по оси X; **SY** — диагональ ромба по оси Y; **R** — угол поворота дуги.

Параметры SX и SY лучше выбирать четными для достижения симметричности ромба; параметр R может быть как положительным, так и отрицательным. Для каждой пары SX и SY существует свое предельное значение R, когда изображение крайних элементов мозаики не выходит за границы графического поля экрана.

На рис. 1.24а - 1.24г приведены несколько вариантов мозаик, созданных с помощью модифицированной программы.

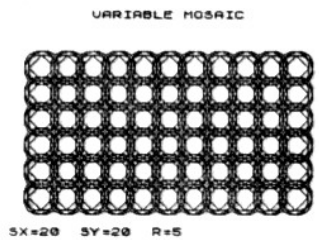


Рис. 1.24а

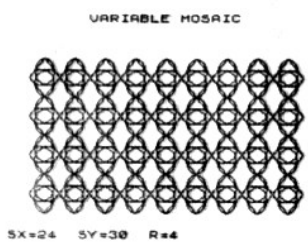


Рис. 1.24б

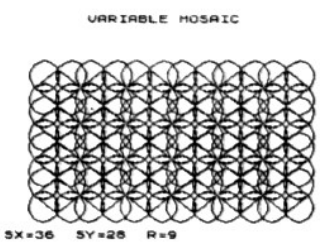


Рис. 1.24в

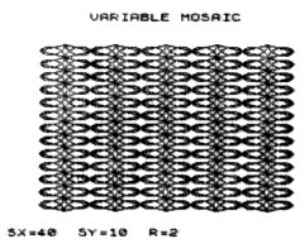


Рис. 1.24г

Вы можете придумать любой другой элемент мозаики и написать соответствующую программу построения мозаики. Причем необязательно регулярно заполнять весь экран элементами, их можно размещать другим образом (создайте различные варианты расположения элементов мозаики).

Во второй главе этой книги будет рассмотрен способ создания мозаик с помощью графических символов, определяемых пользователем.

Гистограммы

Одной из наиболее распространенных форм графического отображения данных является гистограмма (столбиковая диаграмма). Гистограмма представляется в виде ряда столбцов, высоты которых пропорциональны определенным числовым значениям.

На листинге 1.25 приведена программа построения простейших гистограмм.

Listing 1.25

```
10 REM HISTOGRAM
15 PRINT AT 0,11;"HISTOGRAM"
20 INPUT "N=";N: IF N>20 THEN GO TO 20
30 FOR I=1 TO N
40 LET L=0: IF I<10 THEN LET L=1
50 PRINT AT I+2,L;I
60 INPUT "A=";A: IF A>29 THEN GO TO 60
70 FOR J=1 TO A
80 PRINT AT I+2,J+2;"*"
90 NEXT J:NEXT I
100 STOP
```

Гистограмма строится в виде горизонтально расположенных N столбиков, длина каждого из которых пропорциональна соответствующему значению A, задаваемому в диалоговом режиме. Столбики строятся с помощью символов "*", количество которых равно заданному значению A.

При вводе данных для гистограмм осуществляется их контроль — если N > 20 или A > 29, то организуется повторный запрос соответствующего значения параметра. Пример построения гистограммы Вы видите на рис. 1.25.

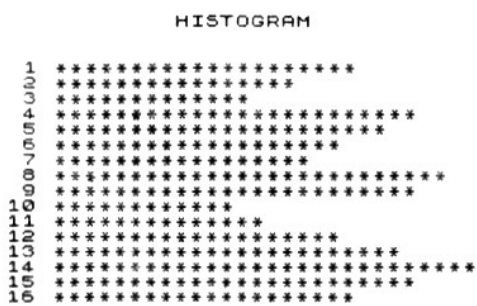


Рис. 1.25

Измените гистограмму на листинге 1.25 так, чтобы на экран выводился не только номер столбика, но и соответствующее ему числовое значение.

На листинге 1.26 представлена еще одна программа построения гистограмм.

Listing 1.26

```
10  REM HISTOGRAM
20  INPUT "N=";N: IF N>80 THEN GO TO 20
25  PRINT AT 0,26;"N=";N
30  DIM Y(N)
40  FOR I=1 TO N
50  INPUT "Y=";Y(I): IF Y(I)>160 THEN GO TO 50
55  PRINT "Y(";I;")=";Y(I)
60  NEXT I:CLS
70  PRINT AT 0,11;"HISTOGRAM"
80  PLOT 5,5:DRAW 240,0: PRINT AT 21,31;"X"
90  PLOT 5,5:DRAW 0,160: PRINT AT 0,0;"Y"
100 LET DX=240/N: LET X=0: LET YMIN=1000: LET YMAX=0
110 FOR I=1 TO N
120 IF Y(I)<YMIN THEN LET YMIN=Y(I)
130 IF Y(I)>YMAX THEN LET YMAX=Y(I)
140 PLOT X+5,5:DRAW 0,Y(I)
150 LET X=X+DX
160 DRAW DX,0:DRAW 0,-Y(I)
170 NEXT I
180 PRINT AT 0,24;"N =";N: PRINT AT 1,24;"YMIN=";YMIN:
    PRINT AT 2,24;"YMAX=";YMAX
190 STOP
```

Гистограмма строится в виде вертикально расположенных столбцов, создаваемых с помощью прямых линий.

Сначала задается количество столбцов N, а затем поочередно числовые значения Y(I) для формирования столбца соответствующей высоты, причем каждое значение Y(I) выводится на экран. В программе осуществляется контроль вводимых параметров: количество столбцов — не более 80 (для получения наглядного изображения гистограммы), высота столбца — не более 160.

По окончании ввода данных экран очищается, строится гистограмма и выводятся минимальное и максимальное значения величины Y, определяемые программой (120 - 130 строки).

На рис. 1.26 Вы видите пример построения гистограммы с помощью этой программы.

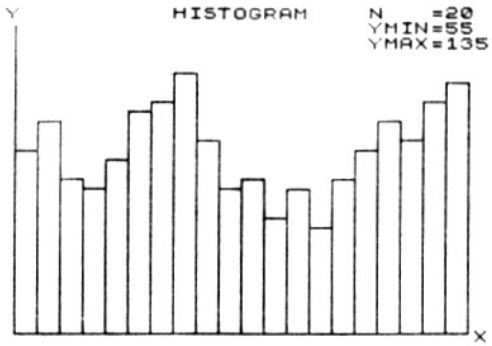


Рис. 1.26

Если в программе (листинг 1.26) задавать значения Y(I) в диапазоне, например, от 1 до 20, то полученная гистограмма будет не очень наглядной — почти все поле экрана будет пустым, а столбики "прижмутся" к оси X. Модифицируйте программу так, чтобы устранить этот недостаток — введите масштабирующий коэффициент для Y(I), вычисляемый, например, по формуле: K = 160/Ymax, и пересчитайте значения Y(I) по формуле: Y(I) = K*Y(I) перед построением гистограммы.

Рассмотрим еще один способ построения гистограмм, реализованный программой на листинге 1.27.

Listing 1.27

```
10 REM HISTOGRAM
20 INPUT "N=";N: IF N>14 THEN GO TO 20
30 PRINT AT 0,11;"HISTOGRAM": PRINT AT 0,26;"N=";N
40 INPUT "C=";C: IF C>6 THEN GO TO 40
50 LET M=INT ((32-2*N)/2)
60 FOR I=1 TO 4
70 LET L=M-3: IF I<10 THEN LET L=L+1
80 PRINT AT 22-5*I,L;5*I
90 NEXT I
100 FOR I=1 TO N
110 INPUT ("H("+STR$ I+")=");H: IF H>20 THEN GO TO 110
120 FOR J=1 TO H
130 PRINT AT 22-J,M+2*I-1; PAPER C;" "
140 NEXT J:NEXT I
150 STOP
```

Гистограмма состоит из вертикальных столбцов, образованных символьными блоками цвета С (строка 130). Столбцы располагаются с промежутком между собой в один символьный блок. Число столбцов N (от 1 до 14), цвет столбцов С (от 1 до 6) и высота каждого столбца Н (от 1 до 20) задаются в диалоговом режиме. Пример работы программы представлен на рис. 1.27.

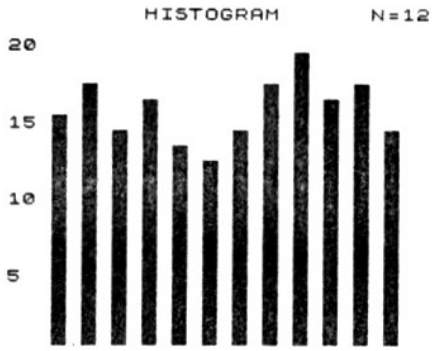


Рис. 1.27

Измените эту программу так, чтобы столбцы располагались вплотную друг к другу и можно было задавать цвет каждого из них отдельно.

Графики функций

Распространенной задачей является построение графиков функций. Написать программу для построения графика функции одной переменной несложно — необходимо правильно выбрать систему координат на графическом поле экрана в зависимости от вида функции, от пределов изменения ее аргумента и значений самой функции; затем по точкам или аппроксимирующим отрезкам построить график, изменяя аргумент и вычисляя соответствующее значение функции. Более сложной является задача создания универсальных программ, позволяющих автоматизировать выбор координатных осей и строить произвольные функции.

Пример одной из таких программ приведен на листинге 1.28.

Listing 1.28

```
10 REM CONSTRUCTION OF THE FUNCTION'S GRAPHS
20 CLS
30 INPUT "XMIN=";XMIN; " XMAX=";XMAX
40 IF XMIN>=XMAX THEN GO TO 30
50 INPUT "YMIN=";YMIN; " YMAX=";YMAX
```

```
60  IF YMIN>=YMAX THEN GO TO 50
70  REM COORDINATE'S AXIS
80  LET XF=XMAX-XMIN: LET YF=YMAX-YMIN
90  LET DX=XF/192: LET DY=YF/144
100 LET SX=ABS (XMIN*(XMIN<0)): LET SX=SX/DX+32
110 LET SY=ABS (YMIN*(YMIN<0)): LET SY=SY/DY+16
120 PLOT 32,SY:DRAW 210, 0: PLOT SX,16:DRAW 0,150
130 LET YSC=INT ((175-SY)/8)+1: LET XSC=INT (SX/8)-4
140 FOR N=0 TO 5
150 LET XSIM=(INT (10*(XMIN+ N*XF/5)+0.5))/10
160 PRINT OVER 1; AT YSC,5*N+4;XSIM: PLOT (5*N+4)*8,SY-1
170 NEXT N
180 FOR N=0 TO 6
190 LET YSIM=(INT (10*(YMAX- N*YF/6)+0.5))/10
200 LET A$=STR$ YSIM
210 IF LEN A$<4 THEN FOR I=LEN A$ TO 3:
    LET A$=" "+A$: NEXT I
220 PRINT OVER 1; AT 3*N+1,XSC;A$: PLOT SX+1,(3*N+2)*8
230 NEXT N
240 INPUT "AXIS FITS (Y)?";B$: IF B$<>"Y" AND B$<>"y"
    THEN GO TO 20
250 REM GRAPH OF THE FUNCTION
260 INPUT "Y(X)=";F$
270 LET X=XMIN: LET YN=(VAL F$-YMIN)/DY: PLOT 32,YN+16
280 FOR N=1 TO 200
290 LET X=XMIN+N*DX
300 LET Y=(VAL F$-YMIN)/DY: IF Y>=159 THEN LET Y=159:
    LET YN=Y: GO TO 330
310 LET XK=N+32: LET YK=Y: IF YK<=-15
    THEN LET YK=-15:     LET YN=-15: GO TO 330
320 PLOT N+31,YN+16: DRAW 1,YK-YN:
    LET YN=YN+INT ((YK-YN)+0.5)
330 NEXT N
340 INPUT "NEW GRAPH (Y/N)?";C$
350 IF C$="Y" OR C$="y" THEN GO TO 260
360 STOP
```

Программа позволяет строить графики произвольных функций в диалоговом режиме, причем на одних координатных осях можно построить несколько графиков в заданных диапазонах изменения аргумента и значений функции.

В начале диалога программа запрашивает диапазоны изменения аргумента XMIN, XMAX и функции YMIN, YMAX.

При неправильном задании параметров (XMIN > XMAX, YMIN > YMAX) соответствующий запрос повторяется.

По выбранным диапазонам изменения аргумента и функции программа вычисляет положение координатных осей на поле экрана (80 - 110 строки), рисует их (120 строка), оцифровывает (130 - 230 строки) и выдает запрос "AXIS FITS (Y/N)?" ("оси подходят?"). При отрицательном ответе экран очищается и запрашиваются новые диапазоны изменения аргумента и функции. При утвердительном ответе запрашивается функция, после задания которой строится соответствующий график по 200 вычисляемым точкам. Причем значения функции, выходящие за графическое поле экрана отсекаются до границ поля (300 - 310 строки).

График строится оператором DRAW (320 строка) с приращениями в 1 пиксель по оси X и YK-YN пикселей по оси Y относительно текущей графической позиции, задаваемой оператором PLOT для каждой из 200 точек графика.

После построения графика программа выдает запрос "NEW GRAPH (Y/N)?" ("новый график?"). При утвердительном ответе Вы можете на тех же координатных осях построить новый график, при отрицательном ответе программа завершает свою работу.

На рис. 1.28а приведен пример построения графика синуса и графика смещенной параболы со следующими параметрами:

XMIN = -10, XMAX = 10, YMIN = -30, YMAX = 60,
Y(X) = 20*SIN(X), Y(X) = 0.5*X*X-10.

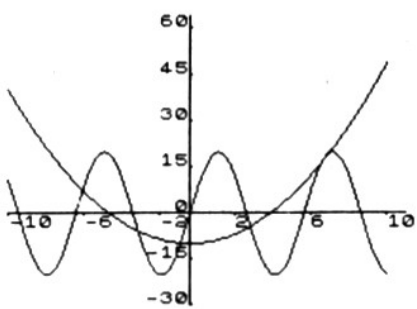


Рис. 1.28а

На рис. 1.28б Вы видите семейство экспонент, имеющих параметры:
XMIN = 0, XMAX = 5, YMIN = 0, YMAX = 30,
Y(X) = 25*EXP(-X), Y(X) = 25*EXP(-0.8*X),
Y(X) = 25*EXP(-0.6*X), Y(X) = 25*EXP(-0.4*X),
Y(X) = 25*EXP(-0.2*X).

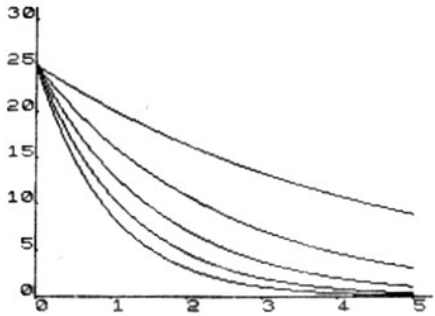


Рис. 1.28б

Закрашивание областей

Закрашивание отдельных областей поля экрана осуществляется по знакоместам (символьными блоками) или по произвольным границам (пикселям), причем границы могут задаваться во входных параметрах, а также определяться самой программой.

На листинге 1.29 приведена программа, позволяющая закрашивать цветом C прямоугольные области размерами X символов по горизонтали и Y символов по вертикали и координатами (X0,Y0) левого нижнего угла области, задаваемыми в символьных блоках.

Listing 1.29

```
10 REM PAINTING OF THE AREA
20 FOR I=21 TO 0 STEP -4: PRINT AT I,0;(21-I):
   NEXT I:PRINT AT 0,1;"Y"
30 FOR I=4 TO 28 STEP 4: PRINT AT 21,I;I:NEXT I:
   PRINT AT 21,31;"X"
40 PLOT 16,8: DRAW 0, 167:DRAW 239,0: DRAW 0,-167:
   DRAW -238,0
50 INPUT "X0(2-31)=";X0; " Y0(1-21)=";Y0: PLOT X0*8,Y0*8
55 INPUT "X=";X;" Y=";Y; " COLOUR(0-6)=";C
60 IF X0+X>32 THEN LET X=32-X0
70 IF Y0+Y>22 THEN LET Y=22-Y0
80 FOR I=X0 TO X0+X-1: FOR J=Y0 TO Y0+Y-1
90 PRINT AT 21-J,I; PAPER C;" "
100 NEXT J:NEXT I
```

```
110 INPUT "NEW AREA?(Y/N)";A$
120 IF A$="Y" OR A$="y" THEN GO TO 50
130 STOP
```

После запуска программы на экране дисплея рисуются координаты оси с оцифровкой в символьных блоках и запрашиваются координаты (X0,Y0) левого нижнего угла области закрашивания, после задания которых в соответствующем месте экрана высвечивается точка. Затем запрашиваются размеры X, Y области и цвет закраски C. По этим параметрам закрашивается определенная Вами область, после чего делается запрос о новой области "NEW AREA? (Y/N)", при утвердительном ответе на который процесс повторяется.

При задании областей, выходящих за правую и верхнюю границы экрана, происходит отсечение части областей до границ экрана. При использовании белого цвета (C = 7) восстанавливается нормальный цвет областей экрана.

Пример работы программы Вы видите на рис. 1.29 (полиграфические возможности данного издания не позволяют различать цвета).

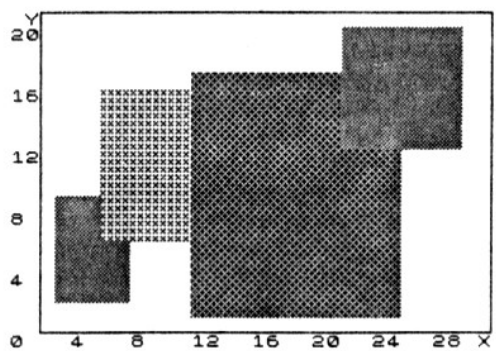


Рис. 1.29

Внесите в программу небольшие изменения для осуществления контроля за нижним левым углом области закрашивания с тем, чтобы области не выходили за левую и нижнюю границы рамки.

Следующая программа (листинг 1.30) позволяет закрасить произвольную прямоугольную область, определяемую с точностью до пикселя.

Listing 1.30

```
10  REM PAINTING OF THE AREA
20  LET K=0:LET L=50
30  FOR I=0 TO 24 STEP 6: PRINT AT 21,I;K:
```

```
    LET K=K+50:NEXT I: PRINT AT 21,31;"X"
40  FOR I=15 TO 3 STEP -6: PRINT AT I,0;L:
    LET L=L+50:NEXT I: PRINT AT 0,1;"Y"
50  PLOT 24,8: DRAW 0, 167:DRAW 231,0: DRAW 0,-167:
    DRAW -230,0
60  INPUT "X0(24-255)=";X0: IF X0<24 THEN GO TO 60
70  INPUT "Y0(8-175)="; Y0: IF Y0<8 THEN GO TO 70
80  PLOT X0,Y0
90  INPUT "X=";X;" Y=";Y
100 IF X0+X>256 THEN LET X=256-X0
110 IF Y0+Y>176 THEN LET Y=176-Y0
120 LET DY=Y-1
130 FOR M=X0 TO X0+X-1: PLOT M,Y0:DRAW 0,DY: NEXT M
140 INPUT "NEW AREA?(Y/N)";A$
150 IF A$="Y" OR A$="y" THEN GO TO 60
160 STOP
```

После запуска программа рисует координатные оси и оцифровывает их в пикселях. Затем запрашиваются координаты (X0,Y0) левой нижней границы области (после ввода которых высвечивается точка с этими координатами) и размеры области по осям X и Y.

В данной программе не используется параметр цвета, так как цвет является атрибутом знакоместа, а не отдельного пикселя. И если части различных областей попадают в одно знакоместо, то происходит наложение цветов. Как и в предыдущей программе области, выходящие за правую и верхнюю границы экрана, отсекаются до границ экрана.

Закрашивание заданной области производится вертикальными отрезками прямых длиной Y, перемещаемыми с шагом в 1 пиксель вдоль горизонтальной оси в пределах размера X. Пример использования программы — на рис. 1.30.

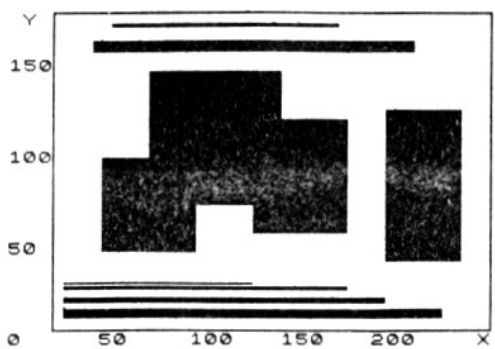


Рис. 1.30

Иногда бывает необходимо закрасить часть фигуры произвольной формы, когда одна или несколько границ области закрашивания заранее не известны.

В этом случае для определения границ используется функция POINT (X,Y), принимающая значение 1, если точка экрана с координатами (X,Y) закрашена, и 0, если эта точка имеет цвет фона.

Программа на листинге 1.31 использует функцию POINT для определения верхней и нижней границ закрашиваемой части фигуры, образуемой окружностью.

Listing 1.31

```
10  REM PAINTING OF THE AREA
15  PRINT AT 0,6; "PAINTING OF THE AREA"
20  INPUT "R(30-80)=";R:
    PRINT AT 2,24;"R =";R: CIRCLE 116,82,R
30  INPUT "LB(-1_+1)=";LB; " RB(-1_+1)=";RB
35  IF LB>RB THEN GO TO 30
40  PRINT AT 3,24;"LB=";LB: PRINT AT 4,24;"RB=";RB
50  LET XL=116+LB*R: LET XR=116+RB*R:
    IF LB=-1 THEN LET XL=XL+1
60  FOR X=XL TO XR
70  LET YB=81:LET YT=82
80  IF POINT (X,YT)=0 THEN PLOT X,YT:PLOT X,YB:
    LET YT=YT+1: LET YB=YB-1:GO TO 80
90  NEXT X
100 STOP
```

После задания параметра R на экране строится окружность этого радиуса и запрашиваются параметры LB и RB (соответственно левая и правая границы закрашиваемой области, определяемые в радиусах окружности. Например, при LB = -1 и RB = 1 закрашивается весь круг, а при LB = 0 и RB = 1 — его правая половина.

После ввода параметров начинается закрашивание определенной Вами области. Начиная с левой границы, строятся два вертикальных отрезка — один вверх, а другой вниз от горизонтального диаметра окружности — до тех пор, пока функция POINT не укажет на то, что достигнута точка, лежащая на окружности.

После этого на горизонтальном диаметре выбирается точка, отстоящая на 1 пиксель вправо от предыдущей точки, и начинается построение новых вертикальных отрезков. Процедура продолжается до достижения правой границы области закрашивания.

Пример использования программы Вы видите на рис. 1.31.

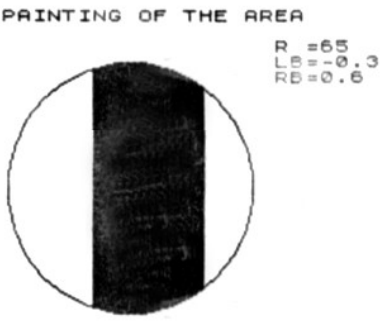


Рис. 1.31

Вы можете написать программу, которая закрашивает, например, области, ограниченные синусоидой и осью X, применяя для этого функцию POINT.

Некоторые другие построения

В заключение главы приведем еще несколько программ, создающих рисунки на экране дисплея. Программа на листинге 1.32 создает рисунки на основе архимедовой спирали, построенной по аппроксимирующим отрезкам прямых.

Listing 1.32

```
10 REM PICTURE
15 PRINT AT 0,12;"PICTURE"
20 INPUT "RMAX=";RMAX: PRINT AT 0,24;"RMAX=";RMAX
30 INPUT "FMAX=";FMAX: PRINT AT 1,24;"FMAX=";FMAX
40 INPUT "N=";N: PRINT AT 2,27;"N=";N
50 LET K=RMAX/FMAX: LET DF=FMAX/N
60 LET X1=128:LET Y1=80
70 FOR F=0 TO FMAX STEP DF
80 LET R=K*F: LET X=R*SIN F: LET Y=R*COS F
90 PLOT 128,80:DRAW X,Y
100 LET X=X+128:LET Y=Y+80
110 PLOT X1,Y1:DRAW X-X1,Y-Y1
120 LET X1=X:LET Y1=Y
130 NEXT F
140 STOP
```

Программа имеет три входных параметра:

- *RMAX — максимальный (конечный) радиус спирали;*
- *FMAX — конечный угол поворота спирали;*
- *N — число точек аппроксимации.*

Алгоритм создания рисунка заключается в том, что на каждом шаге конец аппроксимирующего отрезка соединяется с точкой центра спирали (90 строка программы). Спираль задается уравнениями, приведенными в 80 строке, и строится с помощью операторов 100 - 120 строк программы.

Примеры получаемых изображений Вы видите на рис. 1.32а и рис. 1.32б.

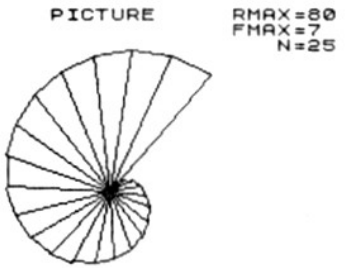


Рис. 1.32а

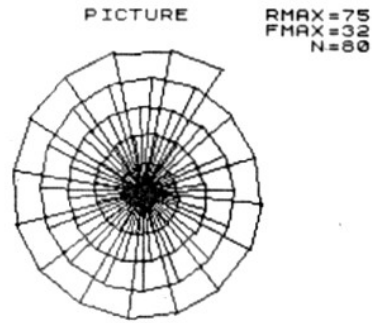


Рис. 1.32б

Фигуры Лиссажу можно получить с помощью программы на листинге 1.33.

Listing 1.33

```
10 REM FIGURE OF LISSADGU
15 PRINT AT 0,6; "FIGURE OF LISSADGU"
20 INPUT "K=";K: PRINT AT 0,0;"K =";K
30 INPUT "G=";G: PRINT AT 1,0;"G =";G
40 INPUT "A1 /5-125/=";A1: PRINT AT 0,26;"A1=";A1
50 INPUT "A2 /5-80/="; A2: PRINT AT 1,26;"A2=";A2
60 FOR I=0 TO 100 STEP 0.25
70 LET F=2*PI*I/100
80 LET X=127+A1*SIN F: LET Y= 80+A2*COS (K*F+G)
90 PLOT X,Y
100 NEXT I
110 STOP
```

В электротехнике фигуры Лиссажу можно наблюдать с помощью осциллографа, если на его входы X и Y подать два синусоидальных сигнала с амплитудами A1 и A2 соответственно, кратностью K и фазовым сдвигом G радиан. Все эти параметры задаются в программе в диалоговом режиме. Построение фигур Лиссажу выполняется по 400 точкам (90 строка) с координатами, вычисляемыми в 80 строке программы.

Примеры получаемых фигур Вы видите на рис. 1.33а, 1.33б.

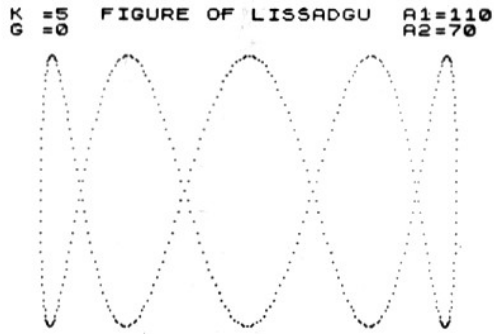


Рис. 1.33а

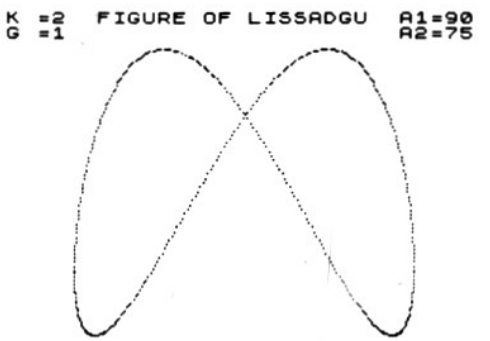


Рис. 1.33б

На листинге 1.34 приведена программа, создающая рисунок, который условно можно назвать "зубчатое колесо".

Listing 1.34

```
10  REM GEAR
15  PRINT AT 0,13;"GEAR"
20  INPUT "N=";N;" H=";H; " RS=";RS;" RV=";RV
30  PRINT AT 0,26;"N =";N: PRINT AT 1,26;"H =";H:
    PRINT AT 2,26;"RS=";RS: PRINT AT 3,26;"RV=";RV
40  FOR I=0 TO 2*PI STEP PI/200
50  LET F=RS+H/2*SIN (I*N)
60  PLOT 127+F*COS I,87+F*SIN I
70  NEXT I
80  CIRCLE 127,87,RV
90  STOP
```

Программа имеет 4 входных параметра:

- *N — число "зубьев" колеса;*
- *H — высота "зубьев";*
- *RS — радиус средней окружности "зубьев";*
- *RV — радиус внутренней окружности.*

Рисунок получается по 400 точкам, координаты которых вычисляются в 50 - 60 строках программы. Пример получаемого изображения приведен на рис. 1.34.

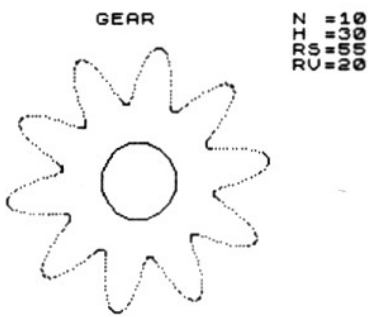


Рис. 1.34

Глава 2

СИМВОЛЬНАЯ ГРАФИКА, ОПРЕДЕЛЯЕМАЯ ПОЛЬЗОВАТЕЛЕМ

Любой символ формируется с помощью задания информации в символьном блоке, представляющим собой восемь двоичных чисел по восемь разрядов в каждом.

Символьный блок соответствует матрице 8x8 пикселей, которая может быть помещена в определенное место экрана дисплея оператором PRINT, а точнее в любую из 32 позиций (0, 1, ..., 31) любой из 22 строк (0, 1, ..., 21).

В SPECTRUME имеются три вида символов:

- *96 символов стандартного набора;*
- *16 графических блок-символов;*
- *21 символ, определяемый пользователем.*

Стандартные символы имеют коды с 32 по 127 и их набор хранится в ПЗУ (постоянном запоминающем устройстве) компьютера. Информация для графических блоков не хранится в ПЗУ, а каждый раз вычисляется по коду символа (от 128 до 143).

Графические блоки можно использовать для получения изображений средней разрешающей способности с графическим полем размерами 64x44 блоков.

Генерация графических символов

В данной работе нас интересует третий набор символов — символы, определяемые пользователем (коды с 144 по 164), которые заносятся в таблицу, расположенную в 168 ячейках в конце памяти, и в графическом режиме могут печататься с клавиатуры.

При включении компьютера в эту таблицу копируется 21 символ от A до U. Для изменения информации в ячейках данной таблицы используется функция USR, которая при вводе любого символа от A до U выдает адрес первой ячейки, хранящей информацию о графическом эквиваленте этого символа. Затем с помощью функций BIN, VAL и оператора POKE можно переопределить все 8 байтов соответствующего символа.

На листинге 2.1 представлена программа, позволяющая генерировать графические символы, которые затем можно использовать в других программах.

При этом не следует забывать, что сгенерированные символы сохраняются в системе только до момента выключения компьютера.

Listing 2.1

```
10  REM GRAPH'S CHARACTERS
15  PRINT "GRAPH'S CHARACTER"
20  INPUT "CHARACTER=";A$: IF CODE A$>96 THEN
    LET A$=CHR$ (CODE A$-32)
30  IF A$="X" THEN GO TO 210
40  IF A$<"A" OR A$>"U" THEN GO TO 20
50  LET MM=USR A$:PRINT
60  FOR I=0 TO 7
70  INPUT "BIN=";LINE F$
80  LET VV=VAL ("BIN "+F$): POKE MM+I,VV
90  PRINT "VAL ";I;"=BIN ";F$
100 NEXT I
110 LET K=11:LET L=0
120 PRINT AT K,L;FLASH 1;"G"
130 INPUT "<CHAR> OR * ";I$
140 IF I$="*" THEN CLS : GO TO 15
150 IF CODE I$>96 THEN LET I$=CHR$ (CODE I$-32)
160 IF I$<"A" OR I$>"U" THEN GO TO 130
170 LET I$=CHR$ (CODE I$+79)
180 PRINT AT K,L;I$
190 LET L=L+1:IF L=32 THEN LET L=0:LET K=K+1:
    IF K=22 THEN LET K=11
200 GO TO 120
210 STOP
```

После запуска программа запрашивает символ "CHARACTER=". Обратите внимание на второй оператор 20 строки, который еще не раз встретится Вам в следующих программах. Этот оператор переводит строчные латинские буквы в прописные, если Вы работаете в режиме курсора L, а не C.

При вводе любого символа от A до U программа последовательно запрашивает 8 байтов двоичного представления графического эквивалента этого символа и выводит их на экран.

После переопределения символа Вы можете посмотреть на графическое изображение на экране (для этого нажмите клавишу этого символа и клавишу "Enter"), либо перейти к переопределению нового символа ("*"+"Enter"). Для выхода из программы введите символ "X".

Пример работы программы приведен на рис. 2.1.

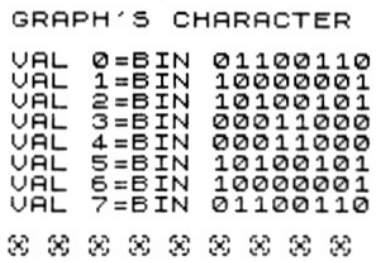


Рис. 2.1

Для сохранения графических символов их необходимо определять в самой программе, использующей графические символы. Это можно сделать, например, с помощью следующего программного фрагмента:

```
K     DATA 0,66,24,36,36,24,66,0
K+1   LET MEMORY=USR"L"
K+2   FOR I=0 TO 7
K+3   READ VALUE
K+4   POKE MEMORY+I,VALUE
K+5   NEXT I
```

Этот фрагмент переопределяет символ L, используемый в двух следующих программах (листинги 2.2 и 2.3).

В строке K задаются значения восьми байтов, определяющих символ L в графическом режиме. В строке K+1 функция USR вычисляет адрес первой ячейки, хранящей информацию о графическом эквиваленте символа L; переменной MEMORY присваивается значение этого адреса. Затем в циклическом режиме переменной VALUE оператором READ присваивается значение текущего байта из строки K, и оператор POKE записывает это значение по соответствующему адресу.

Применение графических символов

Графические символы очень широко применяются в компьютерных играх (некоторые примеры Вы увидите в главе 6), а также для построения рисунков.

На листинге 2.2 приведен пример программы построения мозаики с помощью определяемых пользователем графических символов.

Listing 2.2

```
10 REM MOSAIC
15 PRINT AT 0,13;"MOSAIC"
20 FOR I=2 TO 18 STEP 4
30 FOR J=0 TO 28 STEP 4
40 PRINT AT I ,J;"KLML"
50 PRINT AT I+1,J;"LKLM"
60 PRINT AT I+2,J;"MLKL"
70 PRINT AT I+3,J;"LMLK"
80 NEXT J:NEXT I
90 STOP
```

При вводе программы не забудьте набрать символы K, L, M в 40 - 70 строках в графическом режиме (переход в графический режим осуществляется нажатием <CAPS SHIFT>+9, а выход из него — повторным нажатием этих клавиш). Если Вы выполните программу, то увидите, что поле экрана заполнится упорядоченными символами K,L,M.

Теперь запустите программу генерации графических символов и переопределите символы, например, следующим образом:

```
0 0 0 1 1 0 0 0      0 0 0 0 0 0 0 0      1 1 0 1 1 0 1 1
0 0 1 0 0 1 0 0      0 1 0 0 0 0 1 0      1 0 0 1 1 0 0 1
0 1 0 1 1 0 1 0      0 0 0 1 1 0 0 0      0 0 1 0 0 1 0 0
1 0 1 0 0 1 0 1      0 0 1 0 0 1 0 0      1 1 0 1 1 0 1 1
1 0 1 0 0 1 0 1      0 0 1 0 0 1 0 0      1 1 0 1 1 0 1 1
0 1 0 1 1 0 1 0      0 0 0 1 1 0 0 0      0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 0      0 1 0 0 0 0 1 0      1 0 0 1 1 0 0 1
0 0 0 1 1 0 0 0      0 0 0 0 0 0 0 0      1 1 0 1 1 0 1 1
      K                    L                    M
```

После этого снова выполните программу 2.2, и на экране дисплея появится мозаика, приведенная на рис. 2.2.

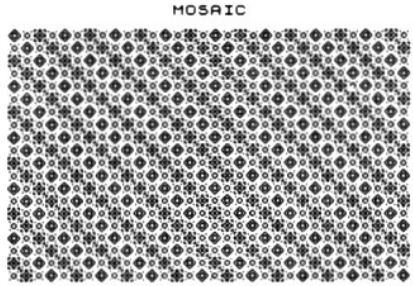


Рис. 2.2

Обратите внимание на то, что символьная мозаика строится намного быстрее, чем мозаики в главе 1.

Для получения разнообразных мозаик можно модифицировать программу 2.2, например, как это сделано в программе на листинге 2.3.

Listing 2.3

```
10 REM MOSAIC
15 PRINT AT 0,8; "VARIABLE MOSAIC"
20 INPUT "N(1-5)=";N: PRINT AT 0,26;"N=";N
30 DIM A$(N):DIM B$(N,N)
40 FOR I=1 TO N
50 LET F=INT (154+3*RND )
60 LET A$(I)=CHR$ F
70 NEXT I
80 LET K=-1
90 FOR I=1 TO N: FOR J=1 TO N
100 LET K=K+1: LET B$(I,J)=A$(I+K)
110 IF I+K=N THEN LET K=-I
120 NEXT J:NEXT I
130 FOR R= 1 TO 21-N STEP N: FOR C=-1 TO 31-N STEP N
140 FOR I=1 TO N: FOR J=1 TO N
150 PRINT AT R+I,C+J;B$(I,J)
160 NEXT J:NEXT I
170 NEXT C:NEXT R
180 INPUT "THAT'S ALL(Y/N)?";I$
190 IF I$<>"Y" AND I$<>"y" THEN CLS :GO TO 15
200 STOP
```

Программа имеет один входной параметр N. Мозаика строится из блоков NxN символов, причем N элементов первой строки блока формируются случайным образом с помощью функции RND (50 строка программы), а остальные строки внутри блока упорядочиваются (аналогично тому, как это было сделано в программе 2.2) по алгоритму, реализованному в 80 - 120 строках программы.

Сформированные таким образом блоки выводятся на экран, заполняя все его поле (130 - 170 строки программы).

После создания мозаики программа делает запрос: "THAT'S ALL (Y/N)?" ("Все?"). При нажатии "Enter" экран очищается, и Вы можете построить новую мозаику. Если ввести символ "Y", то программа завершит свою работу.

Пример работы данной программы Вы видите на рис. 2.3а, 2.3б.

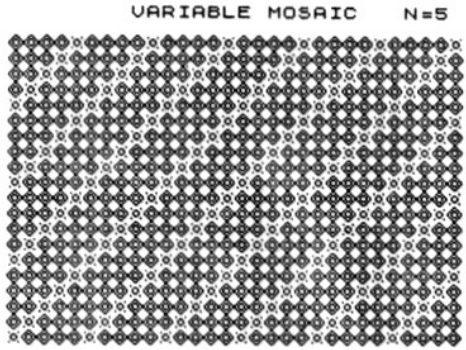


Рис. 2.3а

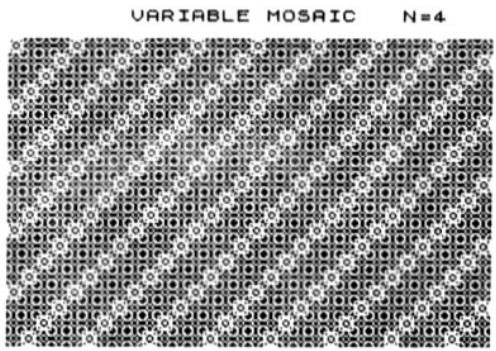


Рис. 2.3б

Составьте свои программы, использующие символьную графику.

Глава 3

ДВИЖЕНИЕ ОБЪЕКТОВ

В компьютерных играх Вы наблюдаете быструю смену событий на экране — перемещаются действующие лица, меняются показания приборов при управлении автомобилем и ландшафт за его стеклами. Причем кажется, что события происходят на экране одновременно.

На самом деле в каждый момент времени реализуется только одно событие, но высокое быстродействие компьютера при выполнении программ, написанных в машинных кодах, создает эффект одновременности действий.

Скорость выполнения бейсик-программы намного меньше, поэтому с их помощью трудно продемонстрировать перемещение сложных объектов. Тем не менее мы попробуем сделать это для достаточно простых объектов, поскольку во всех случаях применяются общие алгоритмы для решения данных задач.

Принцип мультипликации

Для создания объектов, перемещающихся тем или иным образом по полю экрана, применяется принцип мультипликации, когда объект строится, наблюдается, стирается, перемещается в новое положение и затем снова повторяется последовательность перечисленных выше операций.

Процесс построения, например, графика функции по точкам можно рассматривать как движение точки по заданной траектории с сохранением предыдущего изображения точки при переходе к каждому новому положению точки, что и позволяет нам увидеть на экране график функции. Для реализации принципа мультипликации достаточно внести в этот процесс операцию стирания каждого предыдущего изображения точки, и тогда мы увидим на экране не график функции, а движущуюся точку, которая перемещается по невидимой функциональной линии.

Перейдем к более сложным объектам. Для создания впечатления плавного непрерывного перемещения необходимо быстро строить объект. Поскольку графические операторы бейсика выполняются медленно, для построения объектов лучше не использовать окружностей, дуг, а также ограничить количество прямых линий, образующих объект, числом 15 - 20.

Программа, приведенная на листинге 3.1, последовательно строит на экране четыре стрелы (рис. 3.1).

Listing 3.1

```
10 REM ARROWS
15 PRINT AT 8,12;"ARROWS"
20 FOR X=8 TO 240 STEP 65
30 PLOT X,80
40 DRAW -8, 4:DRAW 12,-3: DRAW 36, 0: DRAW 0, 3:
   DRAW 12,-4: DRAW -12,-4: DRAW 0, 3: DRAW -36, 0:
   DRAW -12,-3: DRAW 8, 4
50 NEXT X
60 STOP
```

Рис. 3.1

Графическое изображение стрелы относительно базовой точки (середины хвостового оперения), задается последовательностью операторов DRAW в 40 строке программы. При помощи цикла с управляющей переменной X организуется вычисление текущей базовой точки и построение стрелы в соответствующем месте экрана.

Введем теперь в этот процесс операцию стирания изображения стрелы при переходе к ее новому положению, что выполнено в программе на листинге 3.2.

Listing 3.2

```
10 REM MOTION OF ARROW
15 PRINT AT 8,7; "MOTION OF ARROW"
20 FOR X=8 TO 195 STEP 15
30 GOSUB 90:PAUSE 1
40 OVER 1:GOSUB 90
50 IF INKEY$ <>"" THEN GO TO 80
60 NEXT X
70 GO TO 20
80 STOP
90 PLOT X,80
100 DRAW -8, 4:DRAW 12,-3: DRAW 36, 0: DRAW 0, 3:
    DRAW 12,-4: DRAW -12,-4: DRAW 0, 3: DRAW -36, 0:
    DRAW -12,-3: DRAW 8, 4
110 RETURN
```

Графическое изображение той же стрелы задается в подпрограмме, начинающейся с 90 строки, а в 20 - 70 строках реализуется принцип мультипликации:

- *с помощью управляющей переменной цикла X задается текущая базовая точка изображения стрелы; относительно базовой точки на экране строится стрела (обращением к подпрограмме 90);*
- *изображение наблюдается в течение 0.02 секунд с помощью оператора задержки PAUSE 1;*
- *изображение стрелы стирается с помощью оператора OVER 1 и повторного обращения к подпрограмме 90 (стрела рисуется в цвете, совпадающем с цветом фона экрана);*
- *вычисляется новое значение переменной X, и вся процедура повторяется сначала.*

При работе программы на экране возникает изображение стрелы, движущейся от левой до правой границы экрана (рис. 3.2). Для выхода из программы достаточно нажать любую клавишу.

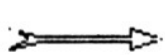


Рис. 3.2

На листинге 3.3 приведена программа, создающая изображение креста, который вращается внутри окружности (рис. 3.3).

Listing 3.3

```
10  REM CROSS
15  PRINT AT 0,8; "ROTATIONAL CROSS"
20  CIRCLE 127,80,80
30  FOR Q=0 TO 1E30 STEP 0.25
40  LET X1=75*SIN Q: LET Y1=75*COS Q
50  LET X2=75*SIN (Q+PI /2): LET Y2=75*COS (Q+PI /2)
60  FOR N=1 TO 2
70  OVER 1: PLOT 127+X1,80+Y1: DRAW -2*X1,-2*Y1
80  PLOT 127+X2,80+Y2: DRAW -2*X2,-2*Y2: IF N=1 THEN
    PAUSE 5
90  NEXT N:NEXT Q
100 STOP
```

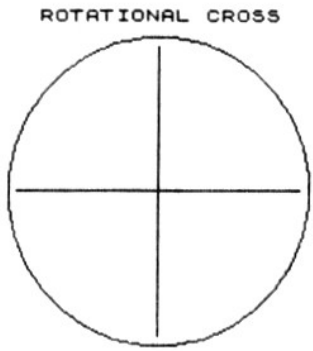


Рис. 3.3

В центре экрана строится окружность радиуса 80. Затем вычисляются координаты (X1,Y1) и (X2,Y2) двух точек таких, что прямые, проведенные через каждую из этих точек и центр окружности образуют два перпендикулярных отрезка. С помощью управляющей переменной N организуется цикл, в котором строится, сохраняется и стирается изображение красного креста. Цикл с управляющей переменной Q организует вращение изображения с шагом 0.25 радиана.

На листинге 3.4 представлена программа, создающая изображение небольшого перекрестия, которое движется по полю экрана и зеркально отражается от границ этого поля (рис. 3.4).

Listing 3.4

```
10  REM MOVING CROSS
15  PRINT AT 0,9;"MOVING CROSS"
20  PLOT 0,0: DRAW 0, 165:DRAW 255,0: DRAW 0,-165:
    DRAW -255,0
30  LET X=128:LET Y=88: GOSUB 120
40  LET DR=3: LET DX=DR:LET DY=DR
50  GOSUB 120:OVER 1
60  LET X=X+DX: LET Y=Y+DY
70  IF X>252-DR OR X<DR THEN LET DX=-DX
80  IF Y>162-DR OR Y<DR THEN LET DY=-DY
90  GOSUB 120
100 IF INKEY$ ="" THEN GO TO 50
110 STOP
120 PLOT X-3,Y:DRAW 6,0: PLOT X,Y-3:DRAW 0,6
130 RETURN
```

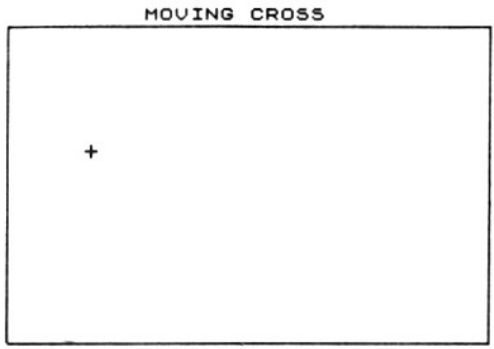


Рис. 3.4

Принцип мультипликации в этой программе реализован аналогично тому, как это было сделано для движущейся стрелы. Выход из программы осуществляется нажатием любой клавиши.

Рассмотрим еще один способ реализации мультипликации — с помощью определяемых пользователем символов и оператора PRINT. Этот способ, который является наиболее быстродействующим (вспомните символьные мозаики, получаемые с помощью графических операторов), реализован в программе на листинге 3.5.

Listing 3.5

```
10 REM FROG
15 PRINT AT 6,12;INK 4; "F R O G"
20 DATA 13,11,11,13:DIM K(4)
30 FOR I=1 TO 4: READ K(I):NEXT I
40 DATA 0,0,0,7,15,63,63,62, 48,104,252,248,240,240,216,28,
   0,0,0,7,15,127,248,224, 4,27,255,254,252,255,239,0
50 FOR I=0 TO 3
60 IF I=0 THEN LET M=USR "A"
61 IF I=1 THEN LET M=USR "B"
62 IF I=2 THEN LET M=USR "C"
63 IF I=3 THEN LET M=USR "D"
70 FOR J=0 TO 7
80 READ V:POKE M+J,V
90 NEXT J:NEXT I
100 PLOT 0,46:DRAW INK 4;255,0
110 PRINT AT 15,30;"  ": PRINT AT 15,0;INK 4;"AB": PAUSE 0
120 FOR I=0 TO 30 STEP 10
```

```
130 PRINT AT 15,I;INK 4;"AB": IF I=30 THEN GO TO 170
135 PAUSE 10: PRINT AT 15,I;"  "
140 FOR J=I+2 TO I+8 STEP 2
150 LET L=1:LET N=K(L): PRINT AT N,J;INK 4;"CD":
    PAUSE 10: PRINT AT N,J;"  ": LET L=L+1
160 NEXT J:NEXT I
170 INPUT "THAT'S ALL(Y/N)?";I$
180 IF I$<>"Y" AND I$<>"y" THEN GO TO 110
190 STOP
```

Программа рисует лягушку, которая прыжками перемещается по полю экрана. Изображение лягушки формируется из двух кадров — состояния покоя и прыжка, причем каждый кадр состоит из двух графических символов. При запуске программы в левой части экрана возникает изображение сидящей лягушки, которая после нажатия любой клавиши в три прыжка перемещается в правую часть экрана. Затем программа выдает запрос "THAT'S ALL (Y/N)?" ("Все?") об окончании работы — при нажатии "Enter" все повторяется сначала, при вводе символа "Y" программа завершает свою работу.

Один из моментов работы программы Вы видите на рис. 3.5.

Рис. 3.5

Рассмотрим эту программу более подробно. С помощью оператора DATA в 20 строке задается траектория прыжка лягушки — номера строк, в которые помещается изображение прыгающей лягушки; эти данные заносятся в массив K(4) оператором READ.

Оператор DATA в 40 строке определяет графические эквиваленты символов A, B, C и D, из которых формируется изображение лягушки: символы A и B — состояние покоя, символы C и D — состояние прыжка. С помощью двух вложенных циклов в 50 - 90 строках информация заносится в соответствующее место памяти. Затем рисуются горизонтальная площадка (100 строка) и сидящая лягушка (110 строка) и с помощью двух вложенных циклов в 120 - 160 строках организуется перемещение изображения по принципу мультипликации.

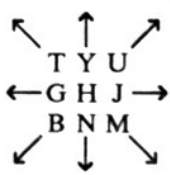
Управление движением

Рассмотренные в этой главе программы реализации принципа мультипликации были пассивными, т.е. работали без Вашего участия в формировании траектории движения объекта. Попробуем создать программы, позволяющие управлять движением объектов. На листинге 3.6 приведена программа, управляющая движением точки по полю экрана.

Listing 3.6

```
10 REM CONTROL
15 PRINT AT 0,5; "CONTROL OF THE MOTION"
20 PLOT 0,0: DRAW 0, 165:DRAW 255,0: DRAW 0,-165:
   DRAW -254,0
30 LET X=127:LET Y=82: OVER 1:PLOT X,Y
40 LET X1=X:LET Y1=Y
50 LET DX=0:LET DY=0
60 LET I$=INKEY$ : IF I$="" THEN GO TO 60
70 IF CODE I$>96 THEN LET I$=CHR$ (CODE I$-32)
80 IF I$="P" THEN OVER 1
90 IF I$="L" THEN OVER 0
100 IF I$="S" THEN GO TO 180
110 IF (I$="U" OR I$="J" OR I$="M") AND X<255 THEN
    LET DX=1
120 IF (I$="T" OR I$="Y" OR I$="U") AND Y<165 THEN
    LET DY=1
130 IF (I$="T" OR I$="G" OR I$="B") AND X>0 THEN
    LET DX=-1
140 IF (I$="B" OR I$="N" OR I$="M") AND Y>0 THEN
    LET DY=-1
150 LET X=X+DX:LET Y=Y+DY
160 PLOT X,Y:PLOT X1,Y1
170 GO TO 40
180 STOP
```

После запуска программы на экране рисуется рамка, окаймляющая графическое поле, и изображается точка в центре экрана. Для управления движением точки служат клавиши, расположенные вокруг символа H. Управление осуществляется по восьми направлениям:



При нажатии и удержании одной из восьми клавиш управления точка перемещается с шагом в один пиксель в соответствующем направлении.

При нажатии клавиши P точка перемещается в режиме мультипликации, а при нажатии клавиши L — в режиме прорисовки траектории движения. Сочетая два этих режима, можно создавать различные изображения на экране. Выход из программы осуществляется нажатием клавиши S. Пример работы с программой Вы видите на рис. 3.6.

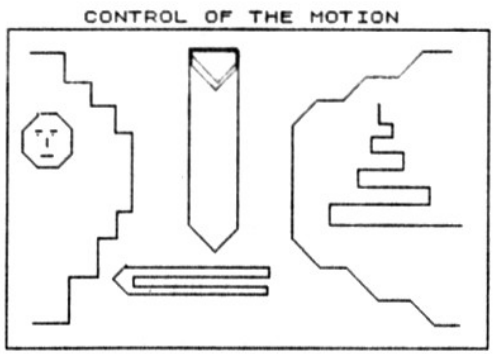


Рис. 3.6

На листинге 3.7 представлена программа управления движением графического символа, которую можно рассматривать как пример простой видеоигры.

Listing 3.7

```
10  REM CONTROL
20  DATA 0,0,254,255,254,0,0,0,  0,0,0,16,0,0,0,0
30  LET M=USR "A": FOR I=0 TO 7: READ V:
    POKE M+I,V:NEXT I
35  LET M=USR "B": FOR I=0 TO 7: READ V:
    POKE M+I,V:NEXT I
40  PLOT 3,3: DRAW 0, 169: DRAW 249,0: DRAW 0,-169:
    DRAW -248,0
45  PRINT AT 1,1; "CONTROL OF THE MOTION"
50  FOR I=2 TO 20
60  IF I=11 THEN PRINT AT I,30; PAPER 2;" ": GO TO 80
70  PRINT AT I,23;PAPER 2;" "
80  NEXT I
90  LET Y=3+INT (RND *17)
100 PRINT AT Y,1;INK 1;"A": PAUSE 0:PRINT AT Y,1;"B":
    LET M=Y:LET N=1
```

```
110 LET I$=INKEY$
120 IF CODE I$>96 THEN LET I$=CHR$ (CODE I$-32)
125 PAUSE 1:PRINT AT M,N;"B"
130 IF I$="R" AND M>2 THEN LET M=M-1
140 IF I$="F" AND M<20 THEN LET M=M+1
150 LET N=N+1
160 PRINT AT M,N;INK 1;"A"
170 IF N=23 AND M<>11 OR N=30 AND M<>11 THEN
    CLS :GO TO 40
180 IF N=30 AND M=11 THEN BEEP 0.25,0: BEEP 0.5,4:
    GO TO 200
190 GO TO 110
200 STOP
```

После запуска программы на экране рисуются окаймляющая рамка, препятствие в правой части экрана, мишень в крайней правой позиции 11-й строки и пуля в первой позиции строки Y, где Y формируется случайным образом в интервале от 3 до 19.

Ваша задача — поразить мишень через отверстие в препятствии.

Траекторию движения пули можно корректировать двумя управляющими клавишами — R (вверх) и F (вниз). При попадании в препятствие Вам предоставляется право нового выстрела. Программа работает до тех пор, пока не будет поражена мишень. Проанализируйте текст программы и найдите, где реализуется каждая из подзадач. Один из моментов работы программы Вы видите на рис. 3.7.

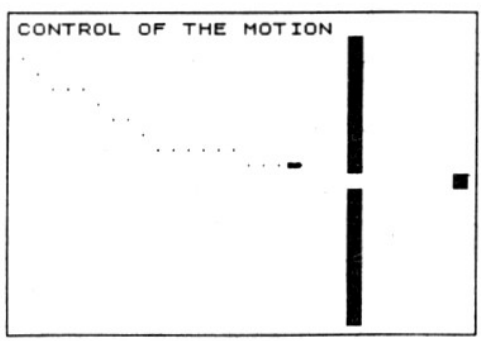


Рис. 3.7

"Бегущий" цвет

Эффект движения может быть получен не только с помощью перемещения объекта по полю экрана, но и другими способами, одним из которых является создание "бегущего" цвета. Если построить непрерывный ряд однотипных элементов таких, что любые смежные элементы имеют разный цвет и поочередно меняют его между собой, то создается впечатление движения элементов в этом ряду.

При работе программы, приведенной на листинге 3.8, на экране строятся два прямоугольника из символов с чередованием желтых и фиолетовых цветов (рис. 3.8). Применение оператора FLASH 1 включает режим мерцания, который создает эффект "бегущего" цвета. Выход из программы осуществляется нажатием любой клавиши.

Listing 3.8

```
10 REM RUNNING COLOUR
15 PRINT AT 0,8; "RUNNING COLOUR"
20 INK 3:PAPER 6:FLASH 1
30 FOR I=0 TO 1
40 FOR J=0 TO 5
50 PRINT AT 2,2*J+I; "<": PRINT AT 2,2*J+I+17;">"
60 PRINT AT 21,2*J+1-I; ">": PRINT AT 21,2*J+1-I+17;"<"
70 NEXT J
80 FOR J=1 TO 10
90 PRINT AT 2*J+I, 0;"V": PRINT AT 2*J+I,12;"↑"
100 PRINT AT 2*J+1-I,16;"↑": PRINT AT 2*J+1-I,28;"V"
110 NEXT J
120 INK 6:PAPER 3
130 NEXT I
140 IF INKEY$ ="" THEN GO TO 140
150 INK 0:PAPER 7:FLASH 0
160 STOP
```

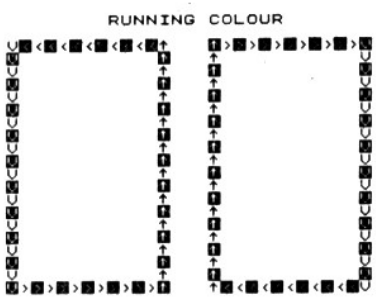


Рис. 3.8

Глава 4

ГРАФИКА ТРЕХМЕРНОГО ПРОСТРАНСТВА

В предыдущих главах мы говорили о графических построениях, лежащих на плоскости, но существует и возможность создавать изображения в трехмерном пространстве — строить графики функций, рисовать объемные тела и т.д.

Для этого выберем систему координат (X,Y,Z), где оси X и Y располагаются как и для случая двух измерений, а ось Z — перпендикулярна плоскости экрана, причем положительное направление этой оси может быть в плоскость экрана (рис. 4.а) или из плоскости экрана (рис. 4.б).

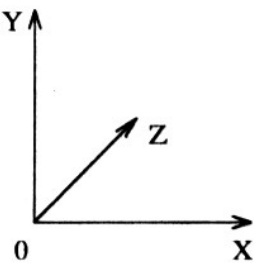


Рис. 4а

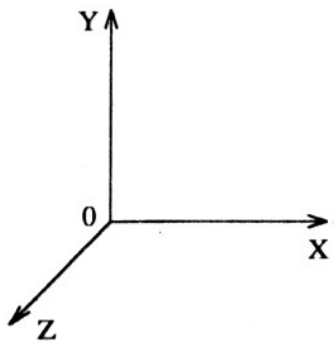


Рис. 4б

На самом деле изображение на экране дисплея всегда является двухмерным, и речь идет лишь об иллюзии создания объемных изображений.

Эффект объемности достигается тем, что точки, расположенные на воображаемой оси Z вычисляются через действительные координаты на осях X и Y.

Точка "а", лежащая на оси Z и имеющая в трехмерном пространстве координаты (0,0,z) (рис. 4.в), определяется через действительные координаты X_Z и Y_Z (рис. 4г) следующим образом:

$$X_z = z \cdot \cos \alpha ,$$

$$Y_z = z \cdot \sin \alpha ;$$

где α - угол, который образует ось Z с осью X.

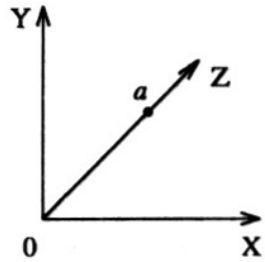


Рис. 4в

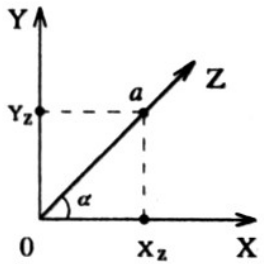


Рис. 4г

Угол α может быть любым. Для дальнейшего рассмотрения примем, например, α = 45°, тогда:

$$sin\ \alpha = cos\ \alpha \approx 0.707\ ,$$

и, следовательно,

$$X_z = 0.707 \cdot z\ ,$$
$$Y_z = 0.707 \cdot z\ .$$

В общем случае точка трехмерного пространства (рис. 4.д) с координатами (x,y,z) может быть построена как точка на плоскости (рис. 4.е) с координатами:

$$X_p = x_a + 0.707 \cdot z\ ,$$
$$Y_p = y_a + 0.707 \cdot z\ .$$

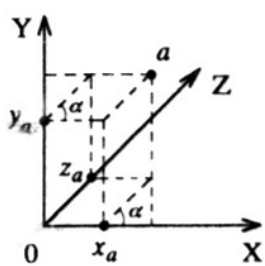


Рис. 4д

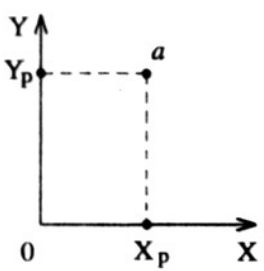


Рис. 4е

Графики функций

Рассмотрим несколько программ, использующих прием преобразования координат для построения графиков функций в трехмерном пространстве.

На листинге 4.1 приведена программа построения параболы, лежащей в плоскости XZ (рис. 4.1).

Listing 4.1

```
10 REM PARABOLA
15 PRINT AT 1,4; "PARABOLA IN THE XZ-PLANE"
20 PLOT 60,50:DRAW 130,0: PRINT AT 15,25;"X"
30 PLOT 60,50:DRAW 0,80: PRINT AT 4,7;"Y"
40 PLOT 60,50:DRAW 90,90: PRINT AT 3,20;"Z"
50 FOR X=-50 TO 50 STEP 0.1
60 LET Z=0.05*X*X
70 PLOT 60+X+.707*Z,50+.707*Z
80 NEXT X
90 STOP
```

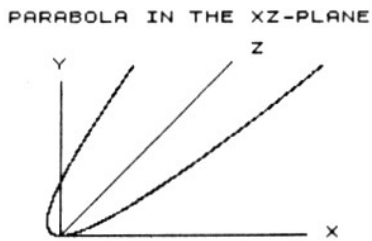


Рис. 4.1

Парабола задается уравнением:

$$z = 0.05 \cdot x^2$$

и строится по точкам в интервале изменения аргумента от -50 до +50 с шагом 0.1.

Эквивалентные точки параболы на графическом поле дисплея имеют координаты, вычисляемые по формулам:

$$X = 60 + x + 0.707 \cdot z\ ,$$

$$Y = 50 + 0 + 0.707 \cdot z\ ;$$

где (60,50) — координаты центра декартовой системы на плоскости экрана.

На листинге 4.2 представлена программа построения синусоиды, лежащей в плоскости YZ (рис. 4.2).

Listing 4.2

```
10 REM SINE
15 PRINT AT 0,6; "SINE IN THE YZ-PLANE"
20 PLOT 90,65:DRAW 100,0: PRINT AT 13,26;"X"
30 PLOT 90,65:DRAW 0,80: PRINT AT 2,11;"Y"
40 PLOT 40,15:DRAW 125,125: PRINT AT 3,21;"Z"
50 LET S=PI /100
60 FOR F=-2*PI TO 3*PI STEP S
70 LET Z=10*F: LET Y=45*SIN F
80 PLOT 90+0.707*Z, 65+Y+0.707*Z
90 NEXT F
100 STOP
```

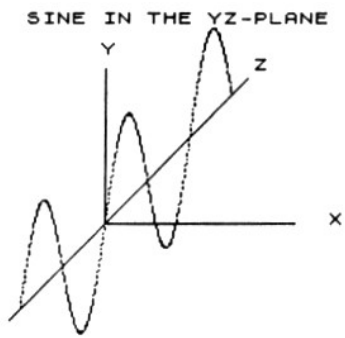


Рис. 4.2

Синусоида задается уравнением:

$$y = 45 \cdot \sin \varphi ,$$

$$z = 10 \cdot \varphi$$

и строится по точкам на 2.5 периодах. Координаты эквивалентных точек вычисляются по формулам:

$$X = 90 + 0 + 0.707 \cdot z ,$$

$$Y = 65 + y + 0.707 \cdot z .$$

Изображенный на рис. 4.3 параболоид строится с помощью программы, приведенной на листинге 4.3.

Listing 4.3

```
10 REM PARABOLOID
15 PRINT AT 0,10;"PARABOLOID"
20 FOR Z=-5 TO 4 STEP 0.15
30 LET S=0.707*Z: LET ZC=Z*Z
40 FOR X=-5 TO 5 STEP 0.15
50 LET Y=0.6*(ZC+X*X)
60 PLOT 127+6*(X+S),6*(Y+S)
70 NEXT X:NEXT Z
80 STOP
```

Рис. 4.3

Параболоид задается уравнением:

$$y = 0.6 \cdot (x^2 + z^2)$$

и строится с помощью двух вложенных циклов. Внешний цикл имеет управляющую переменную Z, изменяемую в диапазоне от -5 до +4 с шагом 0.15, а внутренний цикл — управляющую переменную X, изменяемую от -5 до +5 с шагом 0.15.

Координаты эквивалентных точек определяются следующим образом:

$X = 127 + x + 0.707 \cdot z$,

$Y = 0 + 0.6 \cdot (x^2 + z^2) + 0.707 \cdot z$.

Коэффициент 6 в 60 строке программы введен для получения большей наглядности изображения. Вы можете вынести этот коэффициент во входной параметр и строить масштабированное изображение параболоида.

Рассмотрим теперь некоторые способы построения объемных тел без использования техники преобразования координат.

Пирамиды

На листинге 4.4 приведена программа, с помощью которой можно строить пирамиду по заданным параметрам и вычислять площади ее основания и боковой поверхности, а также объем.

Listing 4.4

```
10  REM PYRAMID
15  PRINT AT 0,5;"PYRAMID"
20  INPUT "N /3-20/="; N: PRINT AT 0,22;"N="; N:
    INPUT "R /15-120/="; R: PRINT AT 1,22;"R="; R:
    INPUT "H /10-110/="; H: PRINT AT 2,22;"H="; H
30  LET X=120:LET Y= 60+R/2: LET S=2*PI /N
40  FOR I=0 TO 2*PI +0.01 STEP S
50  LET XT=120+R*SIN I
60  LET YT= 60+R/2*COS I
70  PLOT X,Y:DRAW XT-X,YT-Y
80  DRAW 120-XT,60+H-YT
90  LET X=XT:LET Y=YT
100 NEXT I
110 INPUT "S,V (Y/N)?";I$
120 IF I$<>"Y" AND I$<>"y" THEN GO TO 200
130 LET SO= 0.5*N*R*R*SIN (2*PI /N)
140 LET Z=R*COS (PI /N): LET F=SQR (Z*Z+H*H)
150 LET SB=N*R*SIN (PI /N)*F
160 LET V=1/3*SO*H
165 LET SO=INT SO: LET SB=INT SB: LET V =INT V
170 PRINT AT 4,22;"SO=";SO
180 PRINT AT 5,22;"SB=";SB
190 PRINT AT 6,22;"V =";V
200 STOP
```

Программа имеет следующие входные параметры:

- *N — число сторон правильного многоугольника, лежащего в основании пирамиды (N = 3-20). При N > 20 графическое изображение пирамиды практически неотличимо от конуса;*
- *R — радиус окружности, описанной вокруг N-угольника основания пирамиды (R = 15-120);*
- *H — высота пирамиды (H = 10-110).*

Для получения трехмерного изображения пирамиды вершины ее основания располагаются на эллипсе с полуосями R и R/2 вдоль координатных осей X и Y соответственно.

Каждая вершина основания соединяется с соседней вершиной и с точкой вершины пирамиды, располагающейся на расстоянии H над центром основания.

После построения пирамиды выдается запрос "S, V (Y/N)?" о вычислении ее параметров, и при утвердительном ответе вычисляются площадь основания, площадь боковой поверхности и объем пирамиды.

Пример работы программы Вы видите на рис. 4.4.

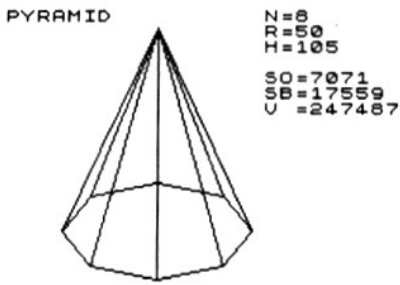


Рис. 4.4

Приведенная выше программа является демонстрационной и имеет ряд ограничений — прозрачность граней пирамиды (нет удаления невидимых линий), наличие правильного, а не произвольного N-угольника в основании. Для устранения этих ограничений потребовалось бы применение одного из методов удаления невидимых линий, усложнилось бы задание входных параметров и труднее было бы вычислить площади и объем. Все это не входит в задачу данной книги.

Хотя в нашем случае можно добиться эффекта удаления невидимых линий внесением следующих изменений в программу:

```
30  LET X=120+R:LET Y=60: LET S=2*PI/N
40  FOR I=PI/2 TO 3/2*PI+0.01 STEP S
```

На рис. 4.4а приведен пример построения пирамиды с учетом внесения этих изменений в программу.

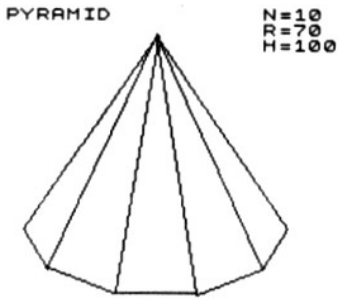


Рис. 4.4а

Конусы

На листинге 4.5 приведена программа построения конуса по заданным параметрам — радиусу основания R и высоте конуса H, с возможностью вычисления площадей основания, боковой поверхности и объема конуса.

Listing 4.5

```
10  REM CONE
15  PRINT AT 0,7;"CONE"
20  INPUT "R /15-120/="; R: PRINT AT 0,22;"R="; R:
    INPUT "H /10-110/="; H: PRINT AT 1,22;"H="; H
30  LET X=120+R:LET Y= 60: LET S=2*PI /60: IF R<40 THEN
    LET S=S*2
40  FOR I=PI/2 TO 3/2*PI STEP S
50  LET XT=120+R*SIN I
60  LET YT= 60+R/2*COS I
70  PLOT X,Y:DRAW XT-X,YT-Y
80  DRAW 120-XT,60+H-YT
90  LET X=XT:LET Y=YT
100 NEXT I
110 INPUT "S,V (Y/N)?";I$
120 IF I$<>"Y" AND I$<>"y" THEN GO TO 200
130 LET SO=INT (PI *R*R)
140 LET L=SQR (R*R+H*H)
150 LET SB=INT (PI *R*L)
160 LET V=INT (1/3*SO*H)
```

```
170 PRINT AT 3,22;"SO=";SO
180 PRINT AT 4,22;"SB=";SB
190 PRINT AT 5,22;"V =";V
200 STOP
```

Конус строится "непрозрачным", т.е. с удалением невидимой его части, что реализуется в 40 строке программы изменением управляющей переменной цикла I в пределах от PI/2 до 3/2*PI. Основание конуса рисуется с помощью эллипса с полуосями R и R/2, который строится по 60 точкам при R > 40 и по 30 точкам при R < 40. Каждая из этих точек соединяется с соседней точкой основания конуса и с вершиной конуса, формируя одну из образующих конуса.

Изменение числа точек, по которым строится основание, связано с тем, что при постоянном числе точек, например 60, с уменьшением радиуса основания образующие конуса сливаются друг с другом. Вы можете вывести формулу для вычисления количества точек основания в зависимости от радиуса основания для получения наглядности изображения конуса. После построения конуса запрашивается необходимость вычисления площадей и объема. При утвердительном ответе вычисляются и выдаются на экран соответствующие параметры (с округлением до целого). На рисунке 4.5 Вы видите конус, построенный с помощью данной программы.

Рис. 4.5

Конус можно построить также с помощью программы, приведенной на листинге 4.6.

Listing 4.6

```
10  REM CONE
15  PRINT AT 0,7;"CONE"
20  INPUT "R /10-100/=";R: PRINT AT 0,22;"R="; R:
    INPUT "H /20-130/=";H: PRINT AT 1,22;"H="; H
```

```
30  FOR I=0 TO H/5
40  LET X=125-5*R/H*I: LET Y=175-5*I
50  PLOT X,Y: DRAW 10*R/H*I,0,1.5
60  NEXT I
70  DRAW -R,H:DRAW -R,-H
80  STOP
```

Изображение конуса формируется следующим образом: оператор вида DRAW X,Y,A (50 строка программы) рисует ряд дуг определенной длины, а затем оператор вида DRAW X,Y (70 строка) рисует две крайние образующие конуса. Такое построение является не совсем точным, так как дуга оператора DRAW X,Y,A неэквивалентна дуге соответствующего эллипса.

На рис. 4.6 приведен пример построения конуса с помощью данной программы.

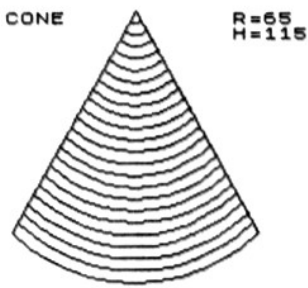


Рис. 4.6

Цилиндры

На листинге 4.7 представлена программа построения цилиндра по заданному радиусу основания R, высоте H и вычисления его выходных параметров — площади основания, площади боковой поверхности и объема.

Listing 4.7

```
10  REM CYLINDER
15  PRINT "CYLINDER"
20  INPUT "R /10-70/="; R: PRINT AT 0,25;"R="; R:
    INPUT "H /10-105/=";H: PRINT AT 1,25;"H="; H
25  LET S=6/R
30  FOR A=-PI/2 TO PI/2 STEP S
40  LET X=127-R*SIN A: LET Y= 35-R/2*COS A
50  PLOT X,Y:DRAW 0,H
```

```
60  NEXT A
70  FOR A=-PI TO PI STEP 0.02
80  LET X=127-R*SIN A: LET Y= 35+H-R/2*COS A
90  PLOT X,Y: IF A>=-PI /2 AND A<=PI /2 THEN PLOT X,Y-H
100 NEXT A
110 INPUT "S,V (Y/N)?";B$
120 IF B$<>"Y" AND B$<>"y" THEN GO TO 190
130 LET SO=INT (PI *R*R)
140 LET SB=INT (2*PI *R*H)
150 LET V =SO*INT H
160 PRINT AT 19,24;"SO=";SO
170 PRINT AT 20,24;"SB=";SB
180 PRINT AT 21,22;"V=";V
190 STOP
```

Видимая боковая поверхность цилиндра строится с помощью образующих, концы которых лежат на эллипсе верхнего основания и полуэллипсе нижнего основания. Обратите внимание на 25 строку программы, в которой определяется шаг приращения S для цикла построения боковой поверхности цилиндра. Шаг вычисляется по эмпирической формуле так, что с уменьшением радиуса основания цилиндра шаг растет; в противном случае образующие боковой поверхности будут сливаться между собой при малых значениях R. По желанию пользователя можно вычислить выходные параметры цилиндра.

На рис. 4.7а, 4.7б приведены примеры построения цилиндров.

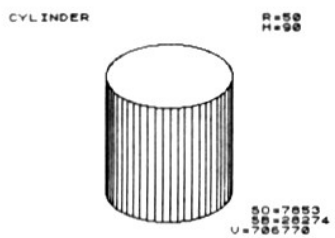


Рис. 4.7а

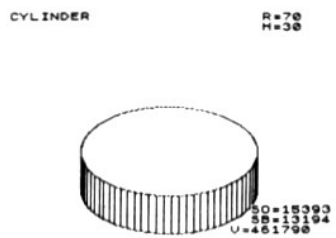


Рис. 4.7б

Глава 5

ЗВУКОВЫЕ ЭФФЕКТЫ

Звуковые эффекты широко применяются в компьютерных играх, усиливая впечатление от происходящих на экране событий. Они могут также использоваться для воспроизведения и создания мелодий музыкальных произведений.

Воспроизведение звуков

Звук воспроизводится оператором

BEEP T,P

где:

- *T — длительность звука (с);*
- *P — высота звука.*

Высота звука задается в полутонах (если P — целое число) от ноты "До" первой (основной) октавы: при P < 0 ноты ниже "До", а при P > 0 — выше "До" первой октавы (рис. 5а).

Рис. 5.а

С помощью простой программы:

```
10  FOR P=-50 TO 50
20  BEEP 0.25,P
30  NEXT P
```

воспроизводится практически весь диапазон тонов, которые можно получить на SPECTRUM'е. В программах допускается использовать и интервалы меньше, чем полутон, например:

P = 2.75, P = 1.8 и т.д.

Примеры

На листинге 5.1 приведена программа, воспроизводящая мелодию песни В. Дорохина "Примерь счастливое лицо", в которой помимо операторов BEEP используются операторы PAUSE, позволяющие выдерживать необходимые паузы для точного воспроизведения мелодий: PAUSE 50 соответствует одной секунде (целая пауза), PAUSE 25 — полсекунде (половинная пауза) и т.д.

Listing 5.1

```
10 REM "TRY ON THE HAPPY FACE" BY V.DOROHIN
15 PRINT "'TRY ON THE HAPPY FACE'":PRINT
20 FOR N=1 TO 3
25 PRINT N;" COUPLET"
30 BEEP 0.12, 0: BEEP 0.25, 5: BEEP 0.12, 8: BEEP 0.12, 5:
   PAUSE 6 : BEEP 0.12, 0: PAUSE 12 : BEEP 0.25, 5:
   BEEP 0.12, 8:  BEEP 0.12, 5: PAUSE 12 : BEEP 0.12, 0:
   BEEP 0.25, 5: BEEP 0.12, 8: BEEP 0.12, 5: PAUSE 6 :
   BEEP 0.12, 3: PAUSE 12: BEEP 0.5, 5: PAUSE 12:
   BEEP 0.12, 0
40 BEEP 0.25, 5: BEEP 0.12, 8:  BEEP 0.12, 5: PAUSE 6 :
   BEEP 0.12, 0: PAUSE 12 : BEEP 0.25, 5: BEEP 0.12, 8:
   BEEP 0.12,10:  PAUSE 12: BEEP 0.12, 8:  BEEP 0.25,10:
   BEEP 0.12, 8: BEEP 0.12, 5: PAUSE 6 : BEEP 0.12, 3:
   PAUSE 12 : BEEP 0.5, 1: PAUSE 12 : BEEP 0.12, 5
50 BEEP 0.12, 7: BEEP 0.12, 8: PAUSE 6 : BEEP 0.12, 10:
   PAUSE 6 :  BEEP 0.12, 8: PAUSE 12 : BEEP 0.25, 7:
   BEEP 0.12, 8:  BEEP 0.12, 5: PAUSE 12 : BEEP 0.12, 5:
   BEEP 0.12, 7:  BEEP 0.12, 8: PAUSE 6 : BEEP 0.12, 10:
   PAUSE 6 : BEEP 0.12, 8: PAUSE 6 : BEEP 0.62, 5:
   PAUSE 12 : BEEP 0.12, 5
60 BEEP 0.12, 7: BEEP 0.12, 8: PAUSE 6 : BEEP 0.12,10:
   PAUSE 6 : BEEP 0.12, 8: PAUSE 12 : BEEP 0.25, 7:
   BEEP 0.12, 8: BEEP 0.12, 5: PAUSE 12 : BEEP 0.12, 5:
   BEEP 0.12, 5: BEEP 0.12, 4: PAUSE 6 :  BEEP 0.12, 4:
   PAUSE 6 : BEEP 0.37, 4: BEEP 0.5, 7: PAUSE 12 :
   BEEP 0.25, 12: BEEP 0.25, 12: BEEP 0.25, 12: BEEP 0.25, 10
70 BEEP 0.12,13:  BEEP 0.37,13: PAUSE 25 : BEEP 0.25,10:
   BEEP 0.25,10: BEEP 0.25,10: BEEP 0.37, 8: BEEP 0.5 ,12:
   PAUSE 25 : BEEP 0.25, 8: BEEP 0.25, 8: BEEP 0.12, 8:
```

```
      BEEP 0.37, 7: BEEP 0.12,10: BEEP 0.37,10: PAUSE 25 :
      BEEP 0.25, 7: BEEP 0.25, 7: BEEP 0.12, 7: BEEP 0.37, 5:
      BEEP 1.0, 8
80    PAUSE 25:NEXT N
90    STOP
```

Нотная запись мелодии представлена на рис. 5.1.

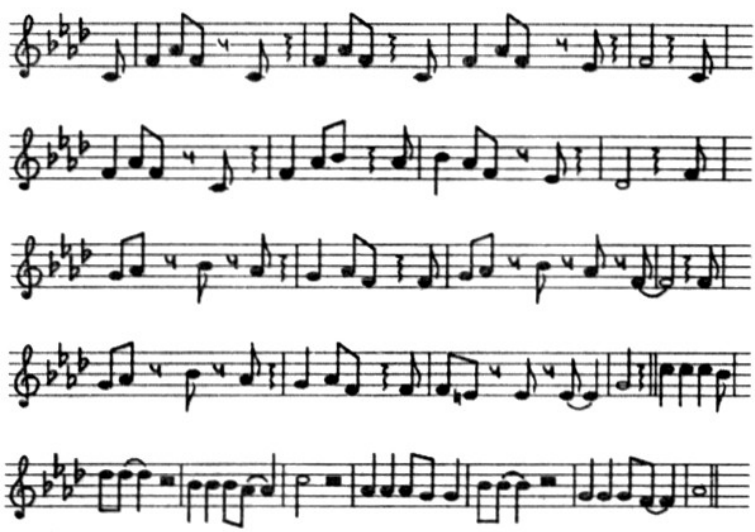


Рис. 5.1

Как видите, для точного воспроизведения мелодии требуются только внимательность и аккуратность.

Внести дополнительные возможности в воспроизведение мелодии можно, например, с помощью программы, приведенной на листинге 5.2.

Listing 5.2

```
10  REM "THE LONELY ACCORDION" BY B.MOCROUSOV
15  PRINT "'THE LONELY ACCORDION'":PRINT
20  INPUT "SET THE TEMPO T=";T: LET T=60/(4*T)
30  DATA T,4, T,7, T,11, T,7, T,4, T,3, T,6, T,9, T,12, 3*T,11,
    T,4, T,6, T,7, T,4, T,0, T,-1, T,3, T,6, 2*T,7, T,9, T,11, T,12,
    T,12, T,7, T,6, T,6, T,9, T,11, 3*T,11, T,4, T,7, T,9, T,9, T,7,
    T,6, T,4, T,6, 2*T,7, T,9, T,11, T,12, T,12, T,7, T,6, T,6, T,9,
    T,11, 3*T,11, T,4, T,7, T,9, T,9, T,7, T,6, T,4, T,3, 2*T,4
```

```
40 DIM T(57):DIM P(57)
50 FOR I=1 TO 57: READ T(I),P(I): NEXT I
60 FOR N=1 TO 4
65 PRINT N;" COUPLET"
70 FOR I=1 TO 57: BEEP T(I),P(I): NEXT I
80 PAUSE 25: NEXT N
90 STOP
```

Программа воспроизводит мелодию песни Б.Мокроусова "Одинокая гармонь" (нотная запись приведена на рис. 5.2) с возможностью задания темпа исполнения. Попробуйте изменять параметр T в пределах от 20 до 180 (T = 60 — нормальный темп).

Обратите внимание на то, что параметры для операторов BEEP задаются с помощью оператора DATA и считываются в циклическом режиме в массивы T(57) и P(57). Этот прием позволяет сократить длину программы, но он применим только для плавных мелодий, где практически отсутствуют паузы, в противном случае произойдет искажение мелодии.

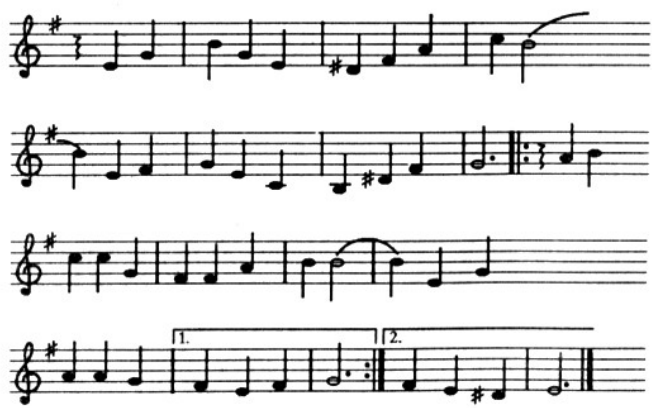


Рис. 5.2

Напишите программу, которая позволит варьировать не только темп исполнения, но и ее тональность с помощью некоторой переменной P, являющейся входным параметром программы.

Например,

```
BEEP T,0+P.
```

При P = 0 исполнится "До" первой октавы, при P = 12 — "До" второй октавы, а при P = 2 — "Ре" первой октавы (тональность ре-минор).

Вы можете превратить свой компьютер в музыкальный инструмент и исполнять или сочинять мелодии, например, с помощью программы, приведенной на листинге 5.3.

Listing 5.3

```
10  REM INSTRUMENT
15  PRINT AT 1,7; "MUSICAL INSTRUMENT"
20  FOR I=0 TO 6
30  PLOT 16+I*32,143: DRAW 0,-108:DRAW 4,-4:
    DRAW 24,0 :DRAW 4 ,4: DRAW 0,108
40  NEXT I
45  PLOT 16,143:DRAW 224,0
50  FOR I=0 TO 4
60  LET X=5+I*4: IF I>1 THEN LET X=X+4
70  FOR J=4 TO 10
80  PRINT AT J,X;PAPER 0;"  "
90  NEXT J:NEXT I
100 DATA 0,1,3,7,7,3,1,0, 0,128,192,224,224,192,128,0
110 FOR I=1 TO 2
120 LET M=USR "A": IF I=2 THEN LET M=USR "B"
130 FOR J=0 TO 7: READ V:POKE M+J,V
140 NEXT J:NEXT I
150 DATA 83,0,15,3, 69,1,9,5, 68,0,15,7, 82,1,9,9, 70,0,15,11,
    71,0,15,15, 89,1,9,17, 72,0,15,19, 85,1,9,21, 74,0,15,23,
    73,1,9,25, 75,0,15,27
160 DIM C(12):DIM K(12): DIM M(12):DIM N(12)
170 FOR I=1 TO 12: READ C(I),K(I),M(I),N(I): NEXT I
180 IF INKEY$ ="" THEN GO TO 180
190 LET I$=INKEY$ : LET F=CODE I$
200 IF CODE I$>92 THEN LET F=CODE I$-32: LET I$=CHR$ F
210 FOR I=1 TO 12
220 IF C(I)=F THEN GO TO 260
230 NEXT I
240 IF I$="Z" THEN GO TO 290
250 GO TO 180
260 IF K(I)=1 THEN GO TO 280
270 PRINT AT M(I),N(I);"AB": BEEP 0.6,I-1:
    PRINT AT M(I),N(I);"  ": GO TO 180
280 PRINT AT M(I),N(I); INK 7;PAPER 0;"AB": BEEP 0.6,I-1:
    PRINT AT M(I),N(I); PAPER 0;"  ":GO TO 180
290 STOP
```

После запуска программы на экране дисплея рисуется клавиатура первой октавы, после чего Вы можете приступать к исполнению мелодии с помощью клавиатуры компьютера. Белым клавишам соответствуют следующие символы клавиатуры: S, D, F, G, H, J, K, а черным клавишам — E, R, Y, U, I. При нажатии любой из этих клавиш формируется соответствующий звук с индикацией выбранной клавиши (смотри рис. 5.3). Выход из программы осуществляется нажатием символа Z.

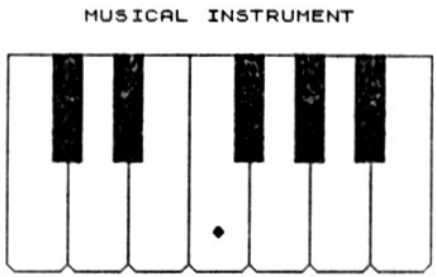


Рис. 5.3

Операторы в 10 - 170 строках программы являются подготовительными. В 20 - 40 строках формируется изображение белых клавиш инструмента, а в 50 - 90 строках — черных клавиш.

Оператор DATA в 100 строке содержит данные для графического эквивалента индикатора выбранной клавиши инструмента, а в 110 - 140 строках происходит переопределение символов A, B для графического режима, который применяется в 270 и 280 строках для формирования изображения индикатора на экране.

Оператор DATA в 150 строке содержит 12 групп данных для каждой из 12 клавиш инструмента. Каждая группа данных, определяющая одну клавишу, содержит четыре вида информации, которая считывается в 170 строке в массивы C(12), K(12), M(12), N(12). В массив C(12) заносятся соответствующие коды символов клавиатуры компьютера, в массив K(12) — признаки цвета клавиш инструмента (0 — белая клавиша, 1 — черная клавиша), массивы M(12) и N(12) содержат атрибуты для операторов PRINT в 270 и 280 строках, выводящих на экран изображение индикатора выбранной клавиши.

Процесс исполнения мелодии реализуется в 180 - 280 строках программы. Значение высоты звука каждой клавиши инструмента формируется с помощью управляющей переменной цикла I и используется в операторах BEEP в 270 и 280 строках программы.

Вы можете написать свою программу, где будет задействовано большее число клавиш инструмента.

Глава 6

КОМПЬЮТЕРНЫЕ ИГРЫ

Игры сопровождают человека всю жизнь, давая выход жажде соперничества и чувству азарта, развивая логическое мышление и умение принять верное решение, совершенствуя реакцию и глазомер, тренируя память. Игра включает элементы творчества и импровизации и может стать антистрессовым тренажером — проигрывая символические очки, мы получаем прививку от более болезненных проигрышей в жизни, выигрывая — мы становимся победителями независимо от того, какое место занимаем в иерархии среди знакомых, в школе и на работе.

Огромное число известных человеку игр "переложено" на компьютер. Хорошая компьютерная графика и большое быстродействие позволяют создавать принципиально новые игры, способствующие приобретению профессиональных навыков — машинописи, управления автомобилями и самолетами. В конце концов, игровые программы просто увлекательны.

Многие из вас уже знакомы с компьютерными играми, использующими сложную графику, имеющими высокое быстродействие. Но, поиграв, часто наступает момент, когда хочется что-то изменить в известной игре или самому создать новую игру. А для этого надо знать, как устроены игры. Практически все распространяемые компьютерные игры написаны в машинных кодах, что позволяет до минимума свести объем занимаемой ими памяти и получить максимальное быстродействие, но делает их практически недоступными для изучения.

В данном разделе книги приведены программы двух простых видеоигр, написанные на бейсике. Это позволит Вам полностью разобраться в них, при желании — изменить, а также создать новые несложные игры. В связи с тем, что быстродействие бейсик-программ невелико, приходится ограничиваться минимумом "изобразительных" возможностей компьютера, уделяя внимание чисто алгоритмической части динамических игр. Это ограничение в меньшей степени распространяется на логические игры (крестики-нолики, морской бой, домино и т.п.).

"Скачки"

Игра "Скачки" имитирует конный забег с верховыми наездниками. Программа приведена на листинге 6.1.

Listing 6.1

```
10  REM RACES
11  DATA 16,26,27,255,255,230,102,102,  16,24,28,30,31,16,16,16,
    0,16,16,16,16,16,16,0
12  FOR I=1 TO 3
```

```
 13  IF I=1 THEN LET M=USR "A"
 14  IF I=2 THEN LET M=USR "B"
 15  IF I=3 THEN LET M=USR "C"
 16  FOR J=0 TO 7
 17  READ V:POKE M+J,V
 18  NEXT J:NEXT I
 20  INK 2:BORDER 1
 30  PRINT AT 1,11;PAPER 6; "R A C E S"
 40  INPUT "N(2-15)=";N: DIM K(N)
 50  FOR I=1 TO N
 60  LET K(I)=2
 70  PRINT AT 3+I, 0;I
 80  PRINT AT 3+I,31;"C"
 90  NEXT I
100 INPUT "MAKE STAKES";I$: PAUSE 25
105 LET F=RND : IF INKEY$ ="" THEN GO TO 105
110 FOR I=1 TO 10000
120 BEEP 0.03,14:BEEP 0.03,16: BEEP 0.03,18
130 FOR J=1 TO N
140 PRINT AT 3+J,K(J); PAPER 6;" "
150 LET K(J)=K(J)+INT (RND *2)
160 IF K(J)>=31 THEN LET K(J)=31:LET W=J:
    PRINT AT 3+J,K(J); PAPER 6;"B":GO TO 190
170 PRINT AT 3+J,K(J); PAPER 6;"A"
180 NEXT J:NEXT I
190 PRINT AT 20,9;PAPER 6; "WON ";W;" HORSE"
200 BEEP 0.12,12:BEEP 0.12,9: BEEP 0.12,19: BEEP 0.12,7:
    BEEP 0.12,12: BEEP 0.12,9: BEEP 0.12,19: BEEP 0.12,7:
    BEEP 0.5 ,12
210 INPUT "NEW HEAT (Y/N)?";A$
220 IF A$="Y" OR A$="y" THEN CLS :GO TO 20
230 INK 0:PAPER 7:BORDER 7: STOP
```

Число лошадей, участвующих в забеге, задается входным параметром N (от 2 до 15). Лошади выстраиваются на стартовой черте в левой части экрана и играющим предлагается сделать ставки на ту или иную лошадь ("MAKE STAKES").

После того, как ставки сделаны, можно начинать забег — двойное нажатие (с интервалом не менее полсекунды) клавиши "Enter". Во время забега производится звуковая имитация перестукивания копыт лошадей.

Лошадь, первой достигнувшая финишной черты, объявляется победителем и исполняется торжественный марш.

После этого программа делает запрос о новом забеге ("NEW HEAT (Y/N)?"), и при утвердительном ответе организуется очередной забег.

Процесс забега организуется с помощью функции RND, когда очередная позиция каждой лошади вычисляется как предыдущая плюс INT(RND*2) в 150 строке. Поскольку функция RND выдает случайные числа в интервале (0,1), то лошадь либо остается на прежней позиции, либо перемещается на одну позицию вправо, а старое изображение "забивается" символом пробела. Процесс повторяется до тех пор, пока какая-нибудь лошадь не достигнет финишной черты (31 позиции соответствующей строки).

Между нажатиями клавиши "Enter" перед началом забега производится "прокрутка" случайных чисел для улучшения характеристики случайности (105 строка программы), в противном случае при каждом новом запуске игры с тем же числом участников будут побеждать одни и те же лошади в каждом соответствующем забеге.

С помощью 11 - 18 строк программы задаются графические эквиваленты изображений лошади со всадником (A), флажка победителя (B), и финишной черты (C). При вводе текста программы не забудьте набрать символы A, B и C в графическом режиме. На рис. 6.1 Вы видите фрагмент игры "Скачки" (рисунок воспроизводится в черно-белом варианте).

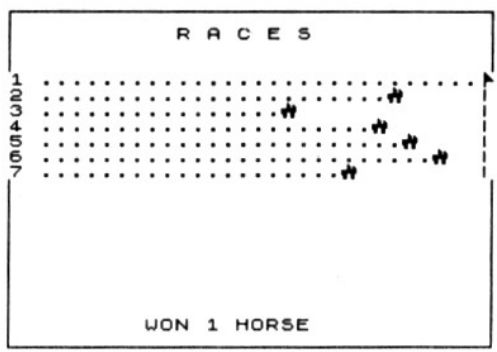


Рис. 6.1

"Стрельба по движущейся мишени"

Данная игра относится к классу "стрелялок" и представляет собой стрельбу по движущейся мишени с возможностью задания вида траектории движения мишени, скорости ее перемещения, позиции стрелка и начальной точки движения мишени, а также ряда других параметров.

Но прежде чем описывать игру в целом, рассмотрим подробно этапы ее создания.

Известно, что любая программа реализует определенную последовательность действий, и в каждый конкретный момент времени может выполняться только одна подзадача.

Очень важной проблемой является создание одномоментности действия, когда создается впечатление, что различные объекты на экране перемещаются одновременно и независимо друг от друга. Рассмотрим один из способов решения этой проблемы.

На листинге 6.2 приведена программа перемещения мишени.

Listing 6.2

```
300 REM TARGET
310 LET R=0: LET C=10+INT (RND *5)
320 PRINT AT R,C;"O"
330 PRINT AT R,C;" ": LET R=R+1
340 IF R=22 THEN GO TO 310
350 IF R>15 THEN LET C=C+1: GO TO 390
360 IF R>10 THEN LET C=C-1: GO TO 390
370 IF R> 5 THEN LET C=C+1: GO TO 390
380 LET C=C-1
390 PRINT AT R,C;"O"
400 PAUSE 5:GO TO 330
```

После запуска программы в верхней строке экрана появляется мишень (символ "O"), которая опускается по зигзагообразной траектории до нижней строки. Затем мишень появляется в новой позиции верхней строки и процесс повторяется. Изменение позиции начальной точки движения мишени осуществляется с помощью функции RND в 310 строке программы. Программа перемещения пули приведена на листинге 6.3.

Listing 6.3

```
200 REM BULLET
210 LET M=10+INT (RND *8)
220 PRINT AT M,0;">"
230 IF INKEY$ ="" THEN GO TO 230
240 LET N=0
250 PRINT AT M,N;" ": LET N=N+1
260 IF N=32 THEN GO TO 210
270 PRINT AT M,N;">"
280 PAUSE 1:GO TO 250
```

После запуска программы в крайней левой позиции строки M экрана появляется изображение пули (символ ">"). После нажатия любой клавиши пуля перемещается по горизонтальной траектории до правой границы экрана.

Затем пуля появляется в крайней левой позиции новой строки экрана, номер которой вычисляется с помощью функции RND в 210 строке программы. Нажатием любой клавиши производится следующий выстрел и т.д.

Следующая программа, приведенная на листинге 6.4, позволяет осуществить стрельбу по движущейся мишени.

Listing 6.4

```
10 REM MAIN PROGRAM
20 LET BULLET=230: LET TARGET=300: LET M=15:LET N=0:
   PRINT AT M,N;">"
30 GOSUB BULLET: GOSUB TARGET
40 GO TO 30
200 REM BULLET
210 LET M=10+INT (RND *8)
220 PRINT AT M,0;">": LET BULLET=230:RETURN
230 IF INKEY$ ="" THEN RETURN
240 LET N=0: LET BULLET=250:RETURN
250 PRINT AT M,N;".": LET N=N+1
260 IF N=32 OR M<R THEN LET BULLET=210:CLS : RETURN
270 PRINT AT M,N;">"
280 RETURN
300 REM TARGET
310 LET R=0: LET C=10+INT (RND *5)
320 PRINT AT R,C;"O": LET TARGET=330:RETURN
330 PRINT AT R,C;".": LET R=R+1
340 IF R=22 THEN LET TARGET=310:CLS : PRINT AT M,0;">":
   RETURN
350 IF R>15 THEN LET C=C+1: GO TO 390
360 IF R>10 THEN LET C=C-1: GO TO 390
370 IF R>5 THEN LET C=C+1: GO TO 390
380 LET C=C-1
390 PRINT AT R,C;"O":PAUSE 2
400 IF R=M AND C=N THEN PRINT AT R,C;"#": BEEP 1,1:
   STOP
410 RETURN
```

В данной программе процессы движения пули и мишени реализованы с помощью подпрограмм BULLET и TARGET соответственно.

Главная (вызывающая) программа производит начальные установки (задание первых точек входа в подпрограммы, изображение первой позиции стрелка на экране) и организует цикл поочередного обращения к подпрограммам.

После запуска программы на экране появляются движущаяся мишень, которая оставляет за собой трассирующий след, и неподвижное изображение пули в нулевой позиции 15 строки. Нажатием любой клавиши производится выстрел, который также оставляет за собой трассу. Если пуля не попала в мишень, выбирается новая позиция стрелка и цикл повторяется. Отвечать выстрелом можно не на каждую мишень. При попадании в мишень на экране возникает взрыв (символ "#") в месте поражения мишени, раздается звуковой сигнал и программа завершает свою работу.

Рассмотрим подробнее организацию этой игровой программы. В двух предыдущих программах Вы видели, что пуля и мишень перемещались быстро, для уменьшения скорости движения мишени даже был применен оператор PAUSE 5. Высокая скорость перемещения обусловлена в этих случаях простотой алгоритма самой задачи. По мере усложнения алгоритма все больше времени требуется на анализ ситуации и соответствующую реакцию, что снижает динамику движения объектов на экране компьютера.

В нашем случае динамику движения определяют подпрограммы, поскольку главная программа организует только цикл обращения к ним. Для повышения скорости выполнения подпрограмм применен метод перемещаемой входной точки при обращении к подпрограммам.

В каждой определенной игровой ситуации подпрограмма выполняет ряд конкретных действий (одну подзадачу), поэтому она написана как последовательность подзадач, обращение к каждой из которых выполняется с помощью переменного указателя (идентификатора).

После выполнения подзадачи указатель настраивается на следующую подзадачу и происходит возврат в вызывающую программу. Например, подпрограмма BULLET имеет три точки входа — 210, 230, 250:

- *210 — формируется позиция стрелка перед выстрелом и выводится на экран. Указатель принимает значение 230, и осуществляется выход из подпрограммы;*
- *230 — если не производился выстрел, то осуществляется выход из подпрограммы. В противном случае вычисляется позиция первой точки трассы пули, указатель принимает значение 250, и осуществляется выход из подпрограммы;*
- *250 — на экран выводится точка трассы полета пули и вычисляется следующая (возможная) позиция пули. Если в этой позиции пуля выйдет за правую границу экрана ($N = 32$) или движущаяся мишень уже пересечет линию огня ($M < R$), то указатель принимает значение 210, очищается экран и осуществляется выход из подпрограммы. В противном случае на экране изображается пуля в этой позиции, и осуществляется выход из подпрограммы.*

Проанализируйте подпрограмму TARGET и выделите соответсвующие подзадачи. Программа, приведенная на листинге 6.5, является дальнейшим развитием предыдущей игровой программы. Она имеет ряд входным параметров, позволяющих Вам организовать игру различной степени сложности.

Listing 6.5

```
10  REM GAME BULLET-TARGET
20  INPUT "BULLET'S POSITION="; M;"+";DM
25  IF M+DM>20 THEN GO TO 20
30  INPUT "TARGET'S POSITION="; C;"+";DC
35  IF C+DC>31 OR C<10 THEN GO TO 30
40  INPUT "NBULLET=";NB; " KWON=";KW
45  IF KW>NB THEN GO TO 40
50  INPUT "SPEED=";SP; " NUM.OF TRASSE=";NT
55  IF SP>20 OR NT>3 THEN GO TO 50
60  INPUT "TRASSE ON/OFF"; " (1/0)=";T: IF T<>0 AND T<>1
    THEN GO TO 60
61  DATA 0,0,60,62,60,0,0,0, 24,36,66,153,153,66,36,24,
    0,0,0,16,0,0,0,0, 0,130,16,0,84,0,16,130
62  FOR I=0 TO 3
63  FOR J=0 TO 7
64  READ V:POKE 65368+I*8+J,V
65  NEXT J:NEXT I
70  LET HSC=0
75  CLS : FOR I=4 TO 9:
    PRINT AT I,9; "BULLET-TARGET":NEXT I
80  PRINT AT 12,5; "BULLET'S POSITION="; M;"+";DM
85  PRINT AT 13,5; "TARGET'S POSITION="; C;"+";DC
90  PRINT AT 14,5; "NBULLET=";NB;" KWON=";KW
95  PRINT AT 15,5; "SPEED=";SP;" NT=";NT
100 PRINT AT 16,5; "TRASSE ON/OFF=";T
110 LET I$=CHR$ 18+CHR$ 1+ "PRESS ENTER TO START"+
    CHR$ 18+CHR$ 0: INPUT (I$+" ");LINE A$
115 CLS :BORDER 2
120 LET EXPLOSION =500: LET STATE =600:
    LET VICTORY =700
130 LET SCORE=0:LET LEVEL=1: LET NBT=NB : LET SPT=SP
140 CLS
144 IF SPT>20 THEN LET SPT=20
145 LET BULLET=230: LET TARGET=300: LET WON=0:
    LET END=0: LET SCT=0:LET MT=M: LET N=0:
    PRINT AT MT,N;"A"
146 LET EXPL=0
```

```
150 GOSUB STATE
160 GOSUB BULLET: BEEP 0.005,9: GOSUB BULLET:
    GOSUB TARGET
170 IF NOT END AND NOT WON AND NOT EXPL THEN
    GO TO 160
180 IF WON THEN LET LEVEL=LEVEL+1: LET NBT=NBT+NB:
    LET SPT=SPT+1: GOSUB VICTORY: GO TO 140
190 IF EXPL THEN LET BULLET=200: GO TO 146
191 IF END THEN FOR I=1 TO 10: BEEP 0.1,15:NEXT I
192 INPUT "NEW GAME (Y/N)?";H$
193 IF H$="Y" OR H$="y" THEN GO TO 75
195 CLS :BORDER 7:STOP
200 REM BULLET
210 LET MT=M+INT (RND *(DM+1))
220 PRINT AT MT,0;"A": LET BULLET=230:RETURN
230 IF INKEY$ ="" THEN RETURN
240 LET N=0: LET BULLET=250:RETURN
250 PRINT AT MT,N;" "
251 IF T=1 THEN PRINT AT MT,N;"C"
252 LET N=N+1
260 IF N<32 AND MT>=R THEN GO TO 270
261 LET NBT=NBT-1: LET BULLET=210: GOSUB STATE
262 IF NBT=0 THEN LET END=1: GO TO 190
263 IF T=1 THEN CLS : GOSUB STATE
264 RETURN
270 PRINT AT MT,N;"A"
280 RETURN
300 REM TARGET
310 LET R=1: LET CT=C+INT (RND *(DC+1))
320 PRINT AT R,CT;"B": LET TARGET=330:RETURN
330 PRINT AT R,CT;" "
331 IF T=1 THEN PRINT AT R,CT;"C"
332 LET R=R+1
340 IF R<21 AND MT>=R THEN GO TO 350
341 LET TARGET=310
342 IF T=1 THEN CLS : GOSUB STATE : PRINT AT MT,0;"A"
343 RETURN
350 IF NT=1 THEN GO TO 410
360 IF NT=2 THEN GO TO 420
370 IF R>15 THEN LET CT=CT+1: GO TO 410
380 IF R>10 THEN LET CT=CT-1: GO TO 410
390 IF R> 5 THEN LET CT=CT+1: GO TO 410
400 LET CT=CT-1
```

```
410 PRINT AT R,CT;"B": BEEP 0.005,-3: PAUSE 10/SPT:
    GO TO 440
420 IF R>10 THEN LET CT=CT+1: GO TO 410
430 GO TO 400
440 IF R=MT AND CT=N THEN PRINT AT R,CT;"D":
    LET EXPL=1: GOSUB EXPLOSION
450 RETURN
500 REM EXPLOSION
510 BEEP .05,-17:BEEP .05,-15: BEEP 1.5,-11
520 LET SCORE=SCORE+1: LET SCT=SCT+1: GOSUB STATE
530 IF SCT>=KW THEN LET WON=1
540 RETURN
600 REM STATE
610 IF SCORE>HSC THEN LET HSC=SCORE
620 PRINT AT 0,0;PAPER 6; "SCORE=";SCORE
630 PRINT AT 0,20;PAPER 6; "HI-SCORE=";HSC
640 PRINT AT 21,0;PAPER 6; "NBULLET=";NBT
650 PRINT AT 21,13;PAPER 6; "SPEED=";SPT
660 PRINT AT 21,24;PAPER 6; "LEVEL=";LEVEL
670 RETURN
700 REM VICTORY
710 DATA 0.06,18, 0.06,19, 0.06,21, 0.15,27, 0.06,21, 0.20,27
720 RESTORE VICTORY: FOR I=1 TO 6: READ L,F:
    BEEP L,F:NEXT I
730 RETURN
```

Рассмотрим входные параметры программы.

1. "BULLET'S POSITION=" M+DM — определяет возможную позицию стрелка в диапазоне от M до M+DM, вычисляемую с помощью функции RND; при DM = 0 стрелок всегда будет находиться на строке M экрана дисплея.

2. "TARGET'S POSITION=" C+DC — определяет возможную точку начала движения мишени в диапазоне от C до C+DC позиции первой строки экрана; при DC = 0 мишень будет перемещаться по фиксированной траектории.

3. "NBULLET=" NB —определяет серию из NB выстрелов.

4. "KWON=" KW — определяет критерий победы, т.е. при KW попаданий из NB выстрелов к оставшемуся числу выстрелов прибавляется еще NB выстрелов, а скорость движения мишени увеличивается — происходит переход на следующий уровень.

5. "SPEED=" SP — определяет начальную скорость движения мишени в диапазоне от 1 до 20.

6. "NUM. OF TRASSE=" NT — определяет номер трассы движения мишени; имеется три вида трассы:

- *мишень перемещается вертикально сверху вниз;*
- *в верхней половине экрана мишень приближается к стрелку, в нижней — удаляется от него;*
- *мишень движется по зигзагообразной траектории.*

7. "TRASSE ON/OFF=" T — определяет режим индикации трассировки движения мишени и пули: 1— индикация производится, 0 — нет.

После задания входных параметров они выводятся на экран. Игра запускается нажатием клавиши "Enter".

В программу игры введены три подпрограммы — EXPLOSION (поражение, взрыв), STATE (счет) и VICTORY (победа) — для реакции на соответсвующие игровые ситуации.

Обратите внимание на двойное обращение к подпрограмме BULLET в 160 строке программы. Это сделано для создания большей скорости перемещения пули. В противном случае можно даже теоретически не попасть в мишень, если стрелок находится в верхних позициях, а мишень вылетает из правых (дальних от стрелка) позиций.

Все выводимые на экран изображения определены в графическом режиме в 61 - 65 строках программы с помощью графических эквивалентов символов A(>), B(O), C(.), D(#). Проанализируйте текст этой программы, рассмотрите реакции на возникающие игровые ситуации.

На рис. 6.5 Вы видите один из игровых моментов.

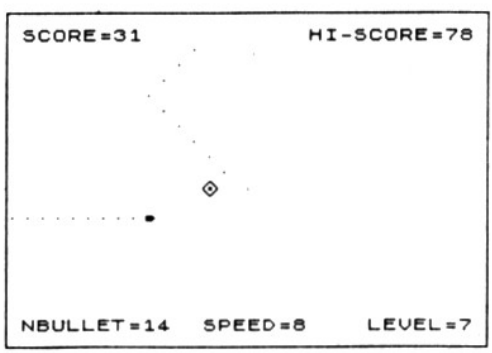


Рис. 6.5

Напишите игровую программу на основе программы изображения прыгающей лягушки (листинг 3.5).

Игра может состоять, например, в следующем. На кочке в левой части экрана сидит лягушка. В правой части экрана появляется вторая кочка, причем ее позиция меняется каждый раз случайным образом. Между этими двумя кочками — болото. Ваша задача — определить длину прыжка лягушки для того, чтобы она перепрыгнула на вторую кочку (длина прыжка в символьных позициях — входной параметр программы). Если Вы неправильно оценили длину прыжка. то лягушка с жалобным кваканьем упадет в болото. Можно устроить соревнование между несколькими игроками на рекордный счет удачных прыжков при заданном числе попыток. Вы можете придумать и другие варианты.

Глава 7

ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ

Эта глава книги может быть полезна старшеклассникам и всем любителям математики при решении некоторых задач: поиска экстремумов функций, решения систем уравнений, вычисления дифференциалов и интегралов и других.

Вычисление простейших функций

В бейсике SPECTRUM'а есть ряд встроенных функций:

- *прямые тригонометрические SIN, COS, TAN;*
- *обратные тригонометрические ASN, ACS, ATN;*
- *экспоненциальная EXP;*
- *натуральный логарифм LN.*

На практике часто возникает задача вычисления недостающих тригонометрических функций (у многих персональных компьютеров имеются только встроенные SIN, COS и ATN), прямых и обратных гиперболических функций, логарифма числа по произвольному основанию и т.д.

При наличии встроенных тригонометрических функций SIN, COS и ATN недостающие функции вычисляются по формулам:

$$tg\ x = sin\ x/cos\ x\ ,$$

$$ctg\ x = cos\ x/sin\ x\ ,$$

$$arcsin\ x = arctg(x/\sqrt{1 - x^2})\ ,$$

$$arccos\ x = \pi/2 - arctg(x/\sqrt{1 - x^2})\ ,$$

$$arcctg\ x = \pi/2 - arctg\ x\ .$$

На основе встроенной функции EXP вычисляются прямые гиперболические функции по формулам:

$$sh\ x = (e^x - e^{-x})\ /\ 2\ ,$$

$$ch\ x = (e^x + e^{-x})\ /\ 2\ ,$$

$$th\ x = (e^x - e^{-x})\ /\ (e^x + e^{-x})\ ,$$

$$cth\ x = (e^x + e^{-x})\ /\ (e^x - e^{-x})\ .$$

На основе встроенной функции LN вычисляются обратные гиперболические функции по формулам:

$$Arsh\ x = ln(x + \sqrt{x^2 + 1}),$$

$$Arch\ x = ln(x + \sqrt{x^2 - 1}),$$

$$Arth\ x = ln((1 + x) / (1 - x)) / 2,$$

$$Arcth\ x = ln((x + 1) / (x - 1)) / 2$$

и логарифмические функции по формуле:

$$log_a x = ln\ x / ln\ a.$$

При вычислении значений всех этих функций необходимо следить за областью их определения.

На листинге 7.1 приведена программа, позволяющая в диалоговом режиме вычислять значения простейших функций по заданному аргументу с осуществлением контроля области их определения. Если значение аргумента выходит за область определения, то вместо значения функции выдается сообщение "no exist" ("не существует").

Listing 7.1

```
10  REM FUNCTIONS
15  PRINT "FUNCTIONS"
20  LET I$="no exist"
30  INPUT "T,G,L=";A$
40  IF CODE A$>96 THEN LET A$=CHR$ (CODE A$-32)
50  IF A$="T" THEN GO TO 90
60  IF A$="G" THEN GO TO 240
70  IF A$="L" THEN GO TO 380
80  GO TO 30
90  INPUT "S,R=";C$
100 IF C$="R" OR C$="r" THEN GO TO 210
110 IF C$<>"S" AND C$<>"s" THEN GO TO 90
115 INPUT "X=";X;" D,R=";B$
120 IF CODE B$>96 THEN LET B$=CHR$ (CODE B$-32)
130 IF B$<>"D" AND B$<>"R" THEN GO TO 90
140 IF B$="D" THEN LET DEG=X: LET RAD=PI *X/180:
    GO TO 150
145 LET RAD=X:LET DEG=180*X/PI
150 PRINT : PRINT "deg";X;"=";DEG: PRINT "rad";X;"=";
    RAD: IF B$="D" THEN LET X=RAD
160 PRINT "sin";X;"=";SIN X: PRINT "cos";X;"=";COS X
170 IF ABS (COS X)=0 THEN PRINT "tg ";X;"=";I$: GO TO 190
180 PRINT "tg ";X;"=";TAN X
190 IF ABS (SIN X)=0 THEN PRINT "ctg";X;"=";I$: GO TO 410
200 PRINT "ctg";X;"="; COS X/SIN X:GO TO 410
210 INPUT "X=";X
```

```
215 PRINT : IF ABS X>1 THEN PRINT "arcsin";X;"=";I$:
    PRINT "arccos";X;"=";I$: GO TO 230
220 PRINT "arcsin";X;"=";ASN X: PRINT "arccos";X;"=";ACS X
230 PRINT "arctg ";X;"=";ATN X: PRINT "arcctg";X;"=";
    PI /2-ATN X:GO TO 410
240 INPUT "S,R=";C$;" X=";X
250 IF C$="R" OR C$="r" THEN GO TO 310
260 IF C$<>"S" AND C$<>"s" THEN GO TO 240
270 LET SH=(EXP X-EXP (-X))/2:
    LET CH=(EXP X+EXP (-X))/2: LET TH=SH/CH
280 PRINT : PRINT "sh ";X;"=";SH: PRINT "ch ";X;"=";CH:
    PRINT "th ";X;"=";TH
290 IF ABS X=0 THEN PRINT "cth";X;"=";I$: GO TO 410
300 PRINT "cth";X;"=";1/TH: GO TO 410
310 LET ASH=LN (X+SQR (X*X+1)): PRINT :
    PRINT "Arsh ";X;"=";ASH
320 IF X<1 THEN PRINT "Arch ";X;"=";I$: GO TO 340
330 LET ACH=LN (X+SQR (X*X-1)): PRINT "Arch ";X;"=";ACH
340 IF ABS X>=1 THEN PRINT "Arth ";X;"=";I$: GO TO 360
350 LET T=0.5*LN ((1+X)/(1-X)): PRINT "Arth ";X;"=";T
360 IF ABS X<=1 THEN PRINT "Arcth";X;"=";I$: GO TO 410
370 LET C=0.5*LN ((X+1)/(X-1)): PRINT "Arcth";X;"=";C:
    GO TO 410
380 INPUT "X=";X;" <a>=";A
390 PRINT : IF X<=0 OR A<=0 OR A=1 THEN
    PRINT "log<";A;">";X; "=";I$:GO TO 410
400 LET L=LN X/LN A: PRINT "log<";A;">";X;"=";L
410 INPUT "NEW FUNC. (Y/N)?";D$
420 IF D$="Y" OR D$="y" THEN GO TO 30
430 STOP
```

После запуска программа запрашивает группу функций:

- *T — тригонометрические;*
- *G — гиперболические;*
- *L — логарифмические.*

Для T и G запрашивается вид функций:

- *S — прямые;*
- *R — обратные.*

После задания вида функций запрашивается аргумент X, а для прямых тригонометрических функций дополнительно запрашивается способ задания аргумента:

- *D — в градусах;*
- *R — в радианах.*

Для группы L запрашивается аргумент X и основание логарифма <a>.

После вывода на экран соответствующих значений функций запрашивается необходимость продолжения вычислений "NEW FUNC. (Y/N)?"; при вводе символа "Y" программа продолжает свою работу, а при нажатии "Enter" программа завершается.

На рис. 7.1 Вы видите пример работы программы.

```
FUNCTIONS

deg64=64
rad64=1.1170107
sin1.1170107=0.89879405
cos1.1170107=0.43837115
tg 1.1170107=2.0503038
ctg1.1170107=0.48773259

arcsin1.75=no exist
arccos1.75=no exist
arctg 1.75=1.0516502
arcctg1.75=0.51914611

sh 2.65=7.0416937
ch 2.65=7.1123449
th 2.65=0.9900664
cth2.65=1.0100333

log<6>12.783=1.4221307
```

Рис. 7.1

Перепишите приведенную выше программу так, чтобы можно было вычислять любую функцию отдельно. Для этого можно вывести на экран меню всех функций и сделать входным параметром номер необходимой функции из этого меню.

Разложение числа на простые множители

Разложение целого числа X на простые множители производится последовательным делением X на 2 и ряд простых чисел XS (XS = 3, 5, 7, 11, 13, 17, 19, ...) по формуле

$$X_i = X_{i-1}/XS$$

до тех пор, пока соблюдается условие

$$\sqrt{X_i} + 1 \le XS .$$

Программа, реализующая этот алгоритм, приведена на листинге 7.2.

Listing 7.2

```
10  REM PRIME MULTIPLIERS
15  PRINT "DECOMPOSITION TO THE","PRIME ";
    "MULTIPLIERS": PRINT
20  INPUT "X=";X
30  PRINT X;"=";:GO TO 60
40  PRINT "2*";:LET X=X/2
50  IF X-1=0 THEN GO TO 70
60  IF X/2-INT (X/2)=0 THEN GO TO 40
70  LET XS=3
80  LET XF=SQR (X)+1
90  IF XS>=XF THEN GO TO 160
100 IF X/XS-INT (X/XS)=0 THEN GO TO 120
110 LET XS=XS+2:GO TO 90
120 IF X/XS*XS-X=0 THEN GO TO 140
130 GO TO 110
140 PRINT XS;"*";
150 LET X=X/XS:GO TO 80
160 PRINT X
170 INPUT "NEW NUMBER?(Y/N)";N$
180 IF N$="Y" OR N$="y" THEN GO TO 20
190 STOP
```

Программа вводит число X и выполняет разложение его на простые множители. После этого запрашивается необходимость разложения нового числа "NEW NUMBER? (Y/N)"; при вводе символа "Y" программа продолжает работу с новым числом, при нажатии "Enter" — завершает работу.

Пример работы программы Вы видите на рис. 7.2.

```
DECOMPOSITION TO THE
PRIME MULTIPLIERS

48=2*2*2*2*3
235=5*47
642=2*3*107
28126=2*7*7*7*41
86963=19*23*199
231=3*7*11
169=13*13
173=173
```

Рис. 7.2

Определение наибольшего общего делителя

Наибольший общий делитель двух чисел A и B находится с помощью соотношения:

$$A_i = A_{i-2} \cdot int(A_{i-2} / A_{i-1}) \cdot A_{i-1} ,$$

где: $i = 2, 3, 4, \ldots$;
$A_0 = \max (|A|, |B|)$;
$A_1 = \min (|A|, |B|)$.

Если A_i станет равным нулю, то наибольший общий делитель MCD будет равен A_{i-1}.

Программа, реализующая этот алгоритм, приведена на листинге 7.3. После нахождения наибольшего общего делителя двух чисел выдается запрос об окончании работы программы "THAT'S ALL? (Y/N)"; при вводе символа "Y" программа завершает работу, при нажатии "Enter" запрашивается новая пара чисел.

Пример работы программы Вы видите на рис. 7.3.

Listing 7.3

```
10  REM DIVIDER
20  PRINT "MAX COMMON DIVIDER": PRINT
30  INPUT "A=";A;" B=";B
40  PRINT "A=";A;" B=";B;
50  LET M=INT (A/B):LET N=B
60  LET B=A-B*M:LET A=N
70  IF B<>0 THEN GO TO 50
80  PRINT " MCD=";A
90  INPUT "THAT'S ALL? ";I$
100 IF I$<>"Y" AND I$<>"y" THEN GO TO 30
110 STOP
```

```
MAX COMMON DIVIDER

A=625 B=155 MCD=5
A=342 B=12 MCD=6
A=248 B=24 MCD=8
A=514 B=8 MCD=2
A=25242 B=126 MCD=42
A=78966 B=248 MCD=2
```

Рис. 7.3

Вычисление факториалов

Факториалом называется целочисленная функция вида:

$$F = N! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot N$$

На листинге 7.4 представлена простая программа вычисления таблицы факториалов от 1! до N!, где N является входным параметром. На рис. 7.4 Вы видите результаты работы программы при N = 20.

Listing 7.4

```
10 REM FACTORIALS
20 PRINT "TABLE OF FACTORIALS" ," N   FACTORIAL"
30 INPUT "NMAX=";N
40 LET P=1
50 FOR I=1 TO N
60 LET P=P*I
70 IF I<10 THEN PRINT " ";
80 PRINT I;"   ";P
90 NEXT I
100 STOP
```

```
TABLE OF FACTORIALS
 N   FACTORIAL
 1   1
 2   2
 3   6
 4   24
 5   120
 6   720
 7   5040
 8   40320
 9   362880
10   3628800
11   39916800
12   4.790016E+8
13   6.2270208E+9
14   8.7178291E+10
15   1.3076744E+12
16   2.0922279E+13
17   3.5568743E+14
18   6.4023737E+15
19   1.216451E+17
20   2.432902E+18
```

Рис. 7.4

Из приведенного примера видно, что уже 12! вычисляется приблизительно, поскольку целое десятичное число может иметь не более 10 цифр.

А 33! не вычисляется совсем, так как наибольшее число, которое воспринимает SPECTRUM, равно 10^{38}.

На листинге 7.5 представлена программа, позволяющая вычислять точные значения факториалов вплоть до 150 цифр результата. Это достигается накоплением десятичных цифр результата в массиве C(150).

На рис. 7.5 приведен результат работы программы при N = 20.

Listing 7.5

```
10  REM FACTORIALS
20  PRINT " EXACT TABLE OF ";
    "FACTORIALS"," N FACTORIAL"
30  INPUT "NMAX=";N
40  DIM C(150)
50  LET R=0:LET F=0: LET C(1)=1
60  FOR I=1 TO N
70  LET T=I
80  FOR J=1 TO 150
90  LET C(J)=C(J)*T+R: LET R=INT (C(J)/10):
    LET C(J)=C(J)-10*R
100 IF R=0 AND J>=F THEN LET F=J:LET K=1: GO TO 120
110 NEXT J
120 IF I<10 THEN PRINT " ";
130 PRINT I;" ";
140 FOR L=F TO 1 STEP -1
150 LET D$=STR$ (C(L))
160 PRINT D$;: IF F-L+2=29*K THEN LET K=K+1:PRINT " ";
170 NEXT L
180 PRINT
190 NEXT I
200 STOP
```

```
 EXACT TABLE OF FACTORIALS
 N FACTORIAL
 1 1
 2 2
 3 6
 4 24
 5 120
 6 720
 7 5040
 8 40320
 9 362880
10 3628800
11 39916600
12 479001600
13 6227020800
14 87178291200
15 1307674368000
16 20922789888000
17 355687428096000
18 6402373705728000
19 121645100408832000
20 2432902008176640000
```

Рис. 7.5

Вычисление степенных полиномов

Степенной полином (многочлен) имеет вид:

$$P(x) = a_n x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_1 \cdot x + a_0$$

Значение полинома в заданной точке x можно вычислить непосредственно по приведенной формуле, но при этом потребуется n сложений и $n \cdot (n+1)/2$ умножений, если каждая степень x получается путем последовательного умножения на x.

Поэтому широко применяется схема Горнера, позволяющая представить степенной полином в виде:

$$P(x) = (\ldots(a_n \cdot x + a_{n-1}) \cdot x + a_{n-2}) + \ldots a_1) \cdot x + a_0$$

В этом случае для вычисления значений полинома в заданной точке требуется n сложений и n умножений.

Программа, приведенная на листинге 7.6, реализует вычисление степенных полиномов по схеме Горнера. После запуска программы необходимо ввести степень полинома N; задать в циклическом режиме коэффициенты полинома A(I), которые выводятся на экран, а затем указать точку X, в которой будет производиться вычисление полинома. После соответствующих вычислений результат выдается на экран и запрашивается необходимость вычисления полинома в новой точке "NEW X (Y/N)?"; для продолжения работы программы надо ввести символ "Y", для окончания — нажать "Enter".

Пример работы программы Вы видите на рис. 7.6.

Listing 7.6

```
10  REM POWER POLYNOMIAL
15  PRINT "CALCULATION OF THE",
    "POWER POLYNOMIAL":PRINT
20  INPUT "N=";N:DIM A(N): PRINT "N=";N
30  FOR I=N TO 1 STEP -1
40  INPUT ("A("+STR$ I+")=");A(I): PRINT "A(";I;")=";A(I);" ";
50  NEXT I
55  INPUT "A(0)=";A0: PRINT "A(0)=";A0
60  INPUT "X=";X: PRINT "X=";X;
70  LET Y=0
80  FOR I=N TO 1 STEP -1
90  LET Y=(Y+A(I))*X
100 NEXT I
110 LET Y=Y+A0
120 PRINT " P(";X;")=";Y
130 INPUT "NEW X (Y/N)?";I$
140 IF I$="Y" OR I$="y" THEN GO TO 60
150 STOP
```

```
CALCULATION OF THE
POWER POLYNOMIAL

N=3
A(3)=0.3
A(2)=2.7
A(1)=-5.2
A(0)=6.4
X=1   P(1)=4.2
X=2   P(2)=9.2
X=3   P(3)=23.2
X=4   P(4)=48
X=5   P(5)=85.4
X=6   P(6)=137.2
X=7   P(7)=205.2
X=8   P(8)=291.2
```

Рис. 7.6

Решение систем линейных уравнений

Система из n линейных уравнений имеет вид:

$$a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n = b_1 ,$$
$$a_{21} \cdot x_1 + a_{22} \cdot x_2 + \ldots + a_{2n} \cdot x_n = b_2 ,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots ,\ldots$$
$$a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + \ldots + a_{nn} \cdot x_n = b_n .$$

Рассмотрим один из наиболее известных и широко применяемых методов решения систем линейных уравнений — ***метод исключения,*** или ***метод Гаусса***, который основан на приведении матрицы коэффициентов a_{ij} к треугольному виду.

Согласно этому методу сначала проводится прямой ход исключения переменных путем преобразования коэффициентов уравнений по формулам:

$$a_{ji} = -a_{ji}/a_{ii} ;$$
$$a_{jk} = a_{jk} + a_{ji} \cdot a_{jk} ,$$
$$b_j = b_j + a_{ji} \quad b_i ;$$

где: $i = 1, 2, \ldots, n\text{-}1$;
$j = i+1, i+2, \ldots, n$;
$k = i+1, i+2, \ldots, n$.

После этих преобразований получаем:

$$x_n = b_n / a_{nn} .$$

Затем организуется обратный ход (последовательное нахождение), проводя вычисления по формулам:

$$h = b_i ,$$
$$h = h - x_j \cdot a_{ij} ,$$
$$x_i = h / a_{ii} ;$$

где: $i = n\text{-}1, n\text{-}2, \ldots, 2, 1$;
$j = i\text{+}1, i\text{+}2, \ldots n$.

Из приведенных выше преобразований видно, что уравнения в системе надо располагать таким образом, чтобы на главной диагонали a_{ii} (при i = 1, 2, ..., n) не было нулевых элементов.

На листинге 7.7 приведена программа, реализующая метод Гаусса с построчным вводом коэффициентов уравнений и выводом на экран коэффициентов и результатов решения системы линейных уравнений. На рис. 7.7 Вы видите пример работы программы.

Listing 7.7

```
10  REM LINEAR EQUATIONS
15  PRINT "SOLUTION OF THE SYSTEM",
    "LINEAR EQUATIONS":PRINT
20  INPUT "N=";N:PRINT "N=";N
30  DIM A(N,N):DIM B(N): DIM X(N)
40  FOR I=1 TO N:PRINT : PRINT "THE ";I;" EQUATION:"
50  FOR J=1 TO N:
    INPUT ("A("+STR$ I+","+ STR$ J+")=");A(I,J):
    PRINT "A(";I;",";J;")="; A(I,J);" ";:NEXT J
60  INPUT ("B("+STR$ I+")=");B(I):
    PRINT " B(";I;")=";B(I): NEXT I
70  FOR I=1 TO N-1: FOR J=I+1 TO N
80  LET A(J,I)=-A(J,I)/A(I,I)
90  FOR K=I+1 TO N
100 LET A(J,K)=A(J,K)+A(J,I)* A(I,K)
110 NEXT K
120 LET B(J)=B(J)+A(J,I)*B(I)
130 NEXT J:NEXT I
140 LET X(N)=B(N)/A(N,N)
150 FOR I=N-1 TO 1 STEP -1
160 LET H=B(I)
170 FOR J=I+1 TO N
180 LET H=H-X(J)*A(I,J)
190 NEXT J
200 LET X(I)=H/A(I,I)
210 NEXT I:PRINT
```

```
220 PRINT "SOLUTION OF THE SYSTEM"
230 FOR I=1 TO N
240 PRINT "X(";I;")=";X(I)
250 NEXT I
260 STOP
```

```
SOLUTION OF THE SYSTEM
LINEAR EQUATIONS

N=3

THE 1 EQUATION:
A(1,1)=3  A(1,2)=5.2  A(1,3)=-4.8
 B(1)=5.6

THE 2 EQUATION:
A(2,1)=7  A(2,2)=-6.4  A(2,3)=1.2
 B(2)=9.5

THE 3 EQUATION:
A(3,1)=9  A(3,2)=-2.4  A(3,3)=7.1
 B(3)=2.7

SOLUTION OF THE SYSTEM
X(1)=1.037908
X(2)=-0.56004184
X(3)=-1.1246862
```

Рис. 7.7

Решение нелинейных уравнений

Решение нелинейного уравнения вида

$$F(x) = 0$$

заключается в нахождении одного или всех корней на отрезке [a,b] изменения аргумента x.

Задача нахождения корней уравнения часто встречается в самых разнообразных областях науки и техники, причем в общем случае функции могут не иметь аналитических формул для своих корней. Поэтому применяются приближенные методы нахождения корней, состоящие обычно из двух этапов:

- *отыскание приближенного значения корня;*
- *уточнение приближенного значения до некоторой заданной степени точности.*

Часто приближенное значение корня бывает известно из физических соображений или его можно определить с помощью простой табуляции функции, т.е. вычисления значений функции в ряде точек с определенным шагом изменения аргумента.

Для уточнения приближенного значения корня существует ряд методов — метод простых итераций, метод деления отрезка пополам, метод хорд, метод секущих и т.д.

Мы рассмотрим ***метод Ньютона***, который еще называют ***методом касательных***. Метод основан на замене $F(x)$ в точке начального приближения x_0 касательной, пересечение которой с осью абсцисс дает первое приближение x_1. Затем x_0 принимает значение x_1 и процесс, реализуемый формулой:

$$x_{n+1} = x_n - F(x_n) / F'(x_n),$$

повторяется до тех пор, пока выполняется условие:

$$|x_{n-1} - x_n| \geq \varepsilon,$$

где ε - заданная погрешность вычисления корня x.

На листинге 7.8 приведена программа решения нелинейного уравнения

$$F(x) = x - cos(x + 2.1) - 0.58 = 0$$

методом Ньютона. Отношение $F(x)/F'(x)$ вычисляется подпрограммой в 110 - 130 строках.

Входными параметрами являются погрешность вычисления корня E и точка начального приближения X0. Уравнение может иметь несколько корней, и поэтому после отыскания корня делается запрос о новой точке начального приближения "NEW X0 (Y/N)?". При положительном ответе (ввод символа "Y") программа продолжает свою работу с новой точкой X0, при отрицательном (нажатие "Enter") — завершается. Результат работы программы приведен на рис. 7.8.

Для решения любого другого уравнения достаточно переписать подпрограмму вычисления отношения $F(x)/F'(x)$.

Listing 7.8

```
10  PRINT "SOLUTION OF THE",, "UNLINEAR EQUATION."
15  PRINT "NEWTON'S METHOD",,,,
20  INPUT "E=";E:PRINT "E=";E
30  INPUT "X0=";X: PRINT "X0=";X
40  GOSUB 110
50  LET X=X-F
60  IF ABS (F)>E THEN GO TO 40
70  PRINT "F(X)=0 IF X=";X
80  INPUT "NEW X0 (Y/N)?";I$
90  IF I$="Y" OR I$="y" THEN GO TO 30
100 STOP
110 REM CALCULATION OF THE F(X)/F'(X)
120 LET F=(X-COS (X+2.1)-0.58)/ (1+SIN (X+2.1))
130 RETURN
```

```
SOLUTION OF THE
UNLINEAR EQUATION.
NEWTON'S METHOD

E=.001
X0=3.55
F(X)=0 IF X=.040563762
```

Рис. 7.8

На листинге 7.9 приведена программа решения методом Ньютона нелинейного уравнения

$$F(x) = -18 \cdot ln(x + 24.5) \cdot 0.39 \cdot x^2 \cdot 8.6 \cdot x + 0.5 = 0$$

с применением табуляции для отыскания приближенных значений корней.

Функция $F(x)$ определяется в 20 строке программы, а ее производная $D(x)$ — в 30 строке с помощью операторов DEF FN. Для табуляции функции необходимо задать левую (A) и правую (B) границы изменений аргумента, а также число отрезков M, на которые разбивается диапазон изменения аргумента.

После табуляции, результаты которой выводятся на экран, можно задавать точки начального приближения для второй части программы, реализующей метод Ньютона. Результаты работы программы приведены на рис. 7.9.

Переопределив функции $F(x)$ и $D(x)$, Вы можете решить любое другое нелинейное уравнение.

Listing 7.9

```
10 REM NEWTON'S METHOD WITH TABULATION OF THE F(X)
20 DEF FN F(X)=-18*LN (X+24.5) -0.39*X*X-8.6*X+0.5
30 DEF FN D(X)=-18/(X+24.5) -0.78*X-8.6
40 PRINT "TABULATION F(X)"
50 INPUT "A=";A:PRINT "A=";A;
60 INPUT "B=";B: PRINT " B=";B;
70 INPUT "M=";M: PRINT " M=";M:PRINT
80 LET C=(B-A)/M
90 FOR X=A TO B+0.1*C STEP C
100 PRINT "X="; INT (X*100)/100; " F(X)=";
    INT (FN F(X)*100)/100
110 NEXT X:PRINT
115 PRINT "NEWTON'S METHOD"
120 INPUT "E=";E: PRINT "E=";E:PRINT
130 INPUT "X0=";X: PRINT "XO=";X
140 LET X1=X-FN F(X)/FN D(X)
150 IF ABS (X1-X)>E THEN LET X=X1:GO TO 140
```

```
160 PRINT "X=";X1;" F(X)="; FN F(X)
170 INPUT "NEW X0 (Y/N)?";I$
180 IF I$="Y" OR I$="y" THEN GO TO 130
190 STOP
```

```
TABULATION F(X)
A=-20.43 B=-10.17 M=9

X=-20.43   F(X)=-11.85
X=-19.29   F(X)=-8.44
X=-18.15   F(X)=-5.16
X=-17.01   F(X)=-2.31
X=-15.87   F(X)=-.04
X=-14.73   F(X)=1.53
X=-13.59   F(X)=2.33
X=-12.45   F(X)=2.31
X=-11.31   F(X)=1.44
X=-10.17   F(X)=-0.3

NEWTON'S METHOD
E=.001

X0=-16
X=-15.84831 F(X)=-3.2782555E-6
X0=-11
X=-10.329315 F(X)=-1.9073486E-6
```

Рис. 7.9

Поиск экстремумов функций

Поиск экстремумов (максимумов или минимумов) функции $y = f(x)$ является часто встречаемой задачей. В общем случае функция может иметь несколько экстремумов, поэтому задача их поиска сводится к локализации и уточнению значений x и $f(x)$ в точке экстремума.

Поскольку максимуму функции $f(x)$ соответствует минимум функции $-f(x)$ (зеркальное отображение относительно оси абсцисс), то мы будем подразумевать под экстремумом максисум $f(x)$, а для поиска минимума достаточно сменить знак у $f(x)$ и воспользоваться методами поиска максимума.

Для локализации экстремумов можно применить обычную табуляцию, как это было сделано при решении нелинейных уравнений методом Ньютона.

В дальнейшем будем считать, что на отрезке [a,b] функция является унимодальной, т.е. имеет один экстремум-максимум (рис. 7а).

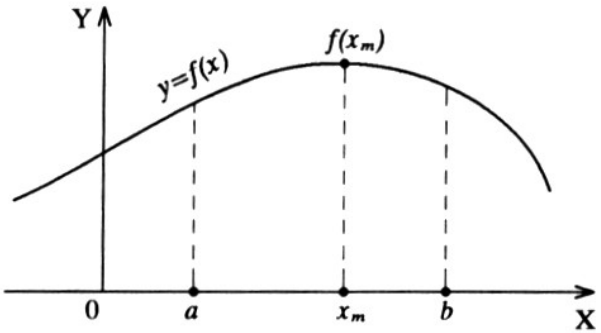


Рис. 7а

Существует целый ряд методов поиска максимумов функции — ***метод равномерного поиска, метод поразрядного приближения, метод деления пополам*** и т.д.

Мы рассмотрим ***метод золотого сечения***, который основан на делении отрезка [a,b] по правилу золотого сечения — коэффициент деления

$$K = 1/2 \cdot (\sqrt{5} \cdot 1) .$$

Рассмотрим алгоритм реализации метода при заданной погрешности ε вычисления x_m:

1. Вычисляем $c = a + (1 \cdot k) \cdot (b \cdot a)$ и $f(c)$.
2. Вычисляем $d = a + k \cdot (b \cdot a)$ и $f(d)$.
3. Проверяем условие $|d \cdot c| < \varepsilon$:
   - если условие выполняется, то вычисляем
     $$x_m = (c + d) / 2 \quad и \quad f(x_m) ,$$
     выдаем их значения и завершаем вычисления;
   - если условие не выполняется, то переходим к п.4.
4. Проверяем условие $f(c) < f(d)$:
   - если условие выполняется, то принимаем
     $$a = c, c = d, f(c) = f(d)$$
     и переходим к п.2;
   - если условие не выполняется, то принимаем
     $$b = d, d = c, f(d) = f(c)$$
     и переходим к п.1 и п.3.

На листинге 7.10 приведена программа реализации метода золотого сечения для функции

$$y = 0.3 \cdot x^3 \cdot 4 \cdot x^2 + 6 \cdot x + 3.2 .$$

Входными параметрами программы являются:

- *A — нижняя граница изменения x;*
- *B — верхняя граница изменения x;*
- *E — погрешность вычисления.*

На рис. 7.10 приведен пример результата работы программы. Для исследования какой-либо другой функции достаточно определить ее в 20 строке программы.

Listing 7.10

```
10  REM MAXIMUM F(X)
15  PRINT "THE SEARCH MAXIMUM F(X)", "BY THE ";
    "METHOD OF", "GOLDEN SECTION":PRINT
20  DEF FN F(X)=((0.3*X-4)*X+ 6)*X+3.2
30  INPUT "A=";A:PRINT "A=";A: INPUT "B=";B:PRINT "B=";B
40  INPUT "E=";E:PRINT "E=";E
50  LET K=(SQR 5-1)/2
60  GOSUB 160: GOSUB 190
70  IF ABS (D-C)<E THEN GO TO 130
80  IF L>M THEN GO TO 110
90  LET A=C:LET C=D:LET L=M
100 GOSUB 190:GO TO 70
110 LET B=D:LET D=C:LET M=L
120 GOSUB 160:GO TO 70
130 LET X=(C+D)/2
140 PRINT "XMAX=";X,, "F(XMAX)=";FN F(X)
150 STOP
160 LET C=A+(1-K)*(B-A)
170 LET L=FN F(C)
180 RETURN
190 LET D=A+K*(B-A)
200 LET M=FN F(D)
210 RETURN
```

```
THE SEARCH MAXIMUM F(X)
BY THE METHOD OF
GOLDEN SECTION

A=-5
B=7
E=.0001
XMAX=0.82701078
F(XMAX)=5.5959668
```

Рис. 7.10

Численное дифференцирование функций

Дифференцированием называют операцию отыскания производной функции. Физический смысл производной функции $y=f(x)$ заключается в том, что она выражает скорость изменения функции в точке x, т.е. скорость протекания процесса, описываемого зависимостью $y=f(x)$.

Если функция $y=f(x)$ определена в точке x и в некоторой окрестности этой точки (рис. 7.б), и если существует предел отношения приращения функции Δy к приращению аргумента Δx при условии, что $\Delta x \to 0$, то функция $y=f(x)$ называется дифференцируемой в точке x, а этот предел называется значением производной функции $y=f(x)$ в точке x:

$$y' = f'(x) = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$$

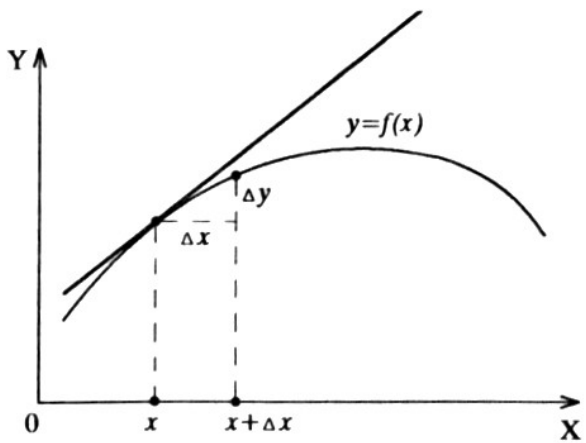


Рис. 7б

Геометрически производную можно интерпретировать как тангенс угла наклона касательной к графику зависимости $y=f(x)$ в точке x относительно оси X.

Существует несколько методов численного дифференцирования функций. На листинге 7.11 приведена программа, вычисляющая производную функции

$$y = 2 \cdot e^{-x^2+3}$$

в точке x с помощью пяти равномерно расположенных (с шагом h) узлов.

Точка x является центральным узлом, по два узла расположены слева и справа от нее.

Производная в точке x вычисляется по формуле:

$$y'(x) = (y(x - 2 \cdot h) - 8 \cdot y(x - h) + 8 \cdot y(x + h) - y(x + 2 \cdot h)) / (12 \cdot h).$$

Listing 7.11

```
10 REM DIFFERENTIATION
15 PRINT "DIFFERENTIATION",, "OF THE FUNCTION":PRINT
20 DEF FN F(X)=2*EXP (-X*X+3)
30 INPUT "H=";H:PRINT "H=";H
40 INPUT "X=";X:PRINT : PRINT "X=";X;
50 LET XT=X-2*H: LET S=FN F(XT)
60 LET XT=X-H: LET S=S-8*FN F(XT)
70 LET XT=X+H: LET S=S+8*FN F(XT)
80 LET XT=X+2*H: LET S=S-FN F(XT)
90 LET DIF=S/(12*H)
100 PRINT " dY/dX=";DIF
110 INPUT "NEW X (Y/N)?";K$
120 IF K$="Y" OR K$="y" THEN GO TO 40
130 STOP
```

Входными параметрами программы являются шаг расположения узлов H и точка X, в которой вычисляется производная. После вычисления производной в заданной точке запрашивается новая точка "NEW X (Y/N)?". При положительном ответе (ввод символа "Y") программа продолжит работу, при отрицательном (нажатие "Enter") — завершится.

Пример работы программы представлен на рис. 7.11. Для дифференцирования любой другой функции ее необходимо определить в 20 строке программы.

```
DIFFERENTIATION
OF THE FUNCTION
H=.001
X=1.1   dY/dX=-26.353597
X=1.2   dY/dX=-22.842343
X=1.3   dY/dX=-19.272114
X=1.4   dY/dX=-15.843617
X=1.5   dY/dX=-12.702003
X=1.6   dY/dX=-9.9373285
X=1.7   dY/dX=-7.5906925
X=1.8   dY/dX=-5.6637215
X=1.9   dY/dX=-4.1294683
```

Рис. 7.11

Вычисление определенных интегралов

Определенный интеграл функции $f(x)$ на отрезке [a,b]

$$I = \int_a^b f(x)\,dx$$

можно интерпретировать как площадь фигуры, ограниченной ординатами $f(a)$ и $f(b)$, осью абсцисс x и графиком функции $y = f(x)$ (рис. 7.в).

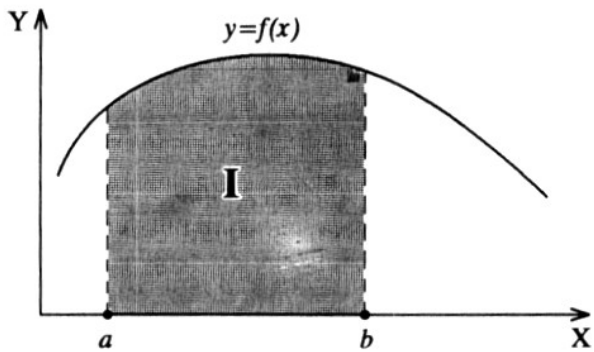


Рис. 7в

Первообразной называется функция, удовлетворяющая условию

$$F'(x) = f(x).$$

Обыкновенный определенный интеграл, у которого известна его первообразная $F(x)$, вычисляется по формуле Ньютона-Лейбница:

$$I = F(b) - F(a).$$

Поэтому достаточно вычислить значения функции $F(x)$. В том случае, если нахождение $F(x)$ сложно или невозможно, применяется численное интегрирование, которое заключается в интерполяции $f(x)$ на отрезке [a,b] соответствующим полиномом, для которого определенный интеграл находится по формулам численного интегрирования. Как правило, отрезок [a,b] разбивается на m частей, к каждой из которых применяется соответствующая простая формула.

Одним из наиболее распространенных методов численного интегрирования является метод Симпсона (метод парабол), заключающийся в интерполяции $f(x)$ полиномом второго порядка на каждом из m отрезков.

На листинге 7.12 приведена программа, реализующая метод Симпсона для функции, которая определена в 20 строке.

Listing 7.12

```
10 REM SIMPSON'S METHOD
15 PRINT "CALCULATION OF THE INTEGRAL",
   "BY THE METHOD OF SIMPSON"
20 DEF FN F(X)=SQR ((7.2*X+1) *X+1.5)
30 PRINT : INPUT "A=";A:PRINT "A=";A: INPUT "B="; B:
   PRINT "B=";B: INPUT "M=";M:PRINT "M=";M
40 LET H=(B-A)/M/2:LET N=0
50 LET X=A:LET I=FN F(A)
60 LET X=X+H: LET I=I+4*FN F(X)
70 LET N=N+2: IF N=2*M THEN GO TO 90
80 LET X=X+H: LET I=I+2*FN F(X):GO TO 60
90 LET I=(I+FN F(B))*H/3
100 PRINT "INTEGRAL=";I
110 INPUT "NEW A,B,M (Y/N)?";I$
120 IF I$="Y" OR I$="y" THEN GO TO 30
130 STOP
```

Входными параметрами программы являются нижний (A) и верхний (B) пределы интегрирования, а также число интервалов интегрирования M.

Алгоритм метода Симпсона реализован в 40 - 90 строках программы. Рассмотрите этот алгоритм, проанализируйте зависимость точности интегрирования от числа M.

Пример работы программы представлен на рис. 7.12. Определив любую другую функцию в 20 строке программы, Вы можете вычислить ее интеграл на соответствующем отрезке.

```
CALCULATION OF THE INTEGRAL
BY THE METHOD OF SIMPSON

A=1
B=5
M=10
INTEGRAL=33.364063

A=4
B=12
M=8
INTEGRAL=173.5172
```

Рис. 7.12

На листинге 7.13 приведена программа, позволяющая вычислить определенный интеграл с заданной точностью, когда погрешность вычисления не превышает заданной величины ε.

Интеграл вычисляется методом Гаусса, который обеспечивает повышенную точность.

Listing 7.13

```
10  REM METHOD OF GAUSSE
15  PRINT "CALCULATION OF THE INTEGRAL", "BY THE";
    "METHOD OF GAUSSE", "WITH DEFINITE ACCURACY"
20  DEF FN F(X)=SQR ((7.2*X+1) *X+1.5)
30  PRINT : INPUT "A=";R:PRINT "A=";R: INPUT "B=";
    B:PRINT "B=";B: INPUT "E=";P:PRINT "E=";P
40  LET P=P*60: LET M=1:LET K=0
50  LET M=M*2:LET A=R
60  LET T=SQR 0.6:LET I=0: LET H=(B-A)/M
70  FOR J=1 TO M
80  LET W=A+H
90  LET C=(W+A)/2: LET D=(W-A)/2
100 LET E=D*5/9: LET L=D*8/9:LET D=D*T
110 LET I=I+E*FN F(C-D)
120 LET I=I+L*FN F(C)
130 LET I=I+E*FN F(C+D)
140 LET A=W
150 NEXT J
160 LET L=K:LET K=I
170 IF ABS (I-L)>P THEN GO TO 50
180 PRINT "INTEGRAL=";I
190 INPUT "NEW A,B,M (Y/N)?";I$
200 IF I$="Y" OR I$="y" THEN GO TO 30
210 STOP
```

Входными параметрами программы являются нижний (A) и верхний (B) пределы интегрирования, а также погрешность вычисления ε. Подынтегральная функция определяется в 20 строке оператором DEF FN.

```
CALCULATION OF THE INTEGRAL
BY THE METHOD OF GAUSSE
WITH DEFINITE ACCURACY

A=3
B=6
E=.001
INTEGRAL=36.968923

A=2
B=12
E=.01
INTEGRAL=190.173
```

Рис. 7.13

В этой главе были рассмотрены некоторые наиболее простые численные методы решения ряда задач. Если Вас заинтересовал предмет данной главы, Вы всегда можете найти дополнительную информацию в специальной литературе и написать или применить уже имеющиеся программы реализации других численных методов.

Послесловие

Вот Вы и перевернули последнюю страницу книги. Автор считает свою задачу выполненной, если Вы проанализировали и применили хотя бы некоторые из приведенных программ, если узнали что-то новое из области компьютерной графики и приемов программирования на бейсике, и тем более, если написали собственные программы, дополняющие и развивающие идеи данной книги.

Автор желает Вам успехов в овладении искусством программирования, достижения новых возможностей в общении с Вашим компьютером.

Оглавление

ВНИМАНИЕ!

Позвонив по телефону: *235-59-85*

Вы можете приобрести:

Книги издательства "ВА Принт" : описания игровых программ для компьютера ZX Spectrum в 8-ми томах; учебные пособия по программированию в широком ассортименте, а также новый многотомник для пользователей IBM-совместимых компьютеров "**PC GAME**", и другую литературу.

Вывески, рекламные таблички для офисов, банков и других служебных помещений, а также, полиграфические работы высокого качества.

Программное обеспечение для ***IBM***-совместимых компьютеров, в том числе ***новые игровые программы*** высокого качества на любых носителях по каталогу или на заказ.

Мы можем дать вашему компьютеру голос ! **Мультимедиа** аппаратура высокого качества: ***"ASTEROID"*** ; ***"VOYAGER"*** в состав которых входят: CD-ROM, компакт-диски, активные колонки, микрофон, 16-ти битная звуковая карта. Для профессионалов ***Roland***-совместимые звуковые карты.

Лицензия ЛР № 061721 от 23 октября 1992 г. Издательский код 6Б8(03)
Подписано в печать 6.10.94 г. Формат 60x88 1/16. Бумага офсетная .
Печать офсетная. Усл. печ. л. 7,5. Тираж 15000 экз. Зак. 6961.

ТОО "ВА ПРИНТ"
127322, Москва, ул. Яблочкова, д. 37, к."В", кв. 210.

Отпечатано с оригинал-макета в филиале Государственного ордена Октябрьской Революции, ордена Трудового Красного Знамени Московского предприятия "Первая Образцовая типография" Комитета Российской Федерации по печати.
113114, Москва, Шлюзовая наб., 10.

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК