Графические возможности

Turbo Pascal

E. Starosvetskiy

Стандартное состояние компьютера после его включения, а также к моменту запуска программы из среды TP соответствует работе экрана в текстовом режиме, поэтому любая программа, использующая графические средства компьютера, должна определённым образом инициировать графический режим работы дисплейного адаптера.

После завершения работы программы ПК возвращается в текстовый режим.

Графические координаты

Любое изображение формируется из достаточно простых геометрических фигур. Это точки, линии окружности и тому подобное.

Положение объекта, его форма задаются координатами его точек. Следовательно, для того чтобы запрограммировать графический вывод, надо научится задавать координаты графических объектов. В качестве графических координат используются порядковые номера пикселов.

от англ. pixel – объединение слов «рисунок» (picture) и «элемент» (element)

Разрешение монитора задаётся в виде:
rx * ry, где rx – кол-во пикселов на экране по горизонтали, ry – по вертикали.
Допустимый диапазон значений изменения графических координат составляет:
[0, rx-1] – для x и **[0, ry-1]** для y. Точкой отсчёта является левый верхний угол экрана.
X – отсчитывается слева направо, Y – сверху вниз.

Состав модуля GRAPH

GRAPH.TPU - библиотека содержащая процедуры и функции для поддержки графического режима.

В графическом режиме экран представляет собой совокупность точек, каждая из которых может быть окрашена в один из 16 цветов. Координаты точек возрастают слева направо и сверху вниз.

Для того, чтобы компилятор «узнавал» названия процедур и функций содержащихся в библиотеке **GRAPH.TPU**, мы должны после заголовка программы разместить строчку следующего вида:

```
uses Graph;
```

Для включения графического режима используется процедура:

```
InitGraph(Gd, Gm: integer; Path: string);
```

InitGraph(Gd, Gm: integer; Path: string);

- У процедуры InitGraph три параметра
- В качестве первых двух параметров должны стоять имена целых (integer) переменных
- Первый параметр Gd является кодом графического адаптера (т. е. электронной схемы, управляющей выводом информации на экран). В используемых нами компьютерах используется адаптер VGA (и компилятор сам "узнаёт" слово VGA и заменит его на нужное целое число). Каждый графический адаптер позволяет использовать несколько графических режимов, отличающихся количеством цветов и разрешающей способностью
- Второй из параметров Gm предназначен для того, чтобы указать, какой из режимов следует включить
- Третий параметр Path является строкой (string), содержащей путь к файлу, который называется egavga.bgi. В этом файле содержится драйвер, необходимый для работы мониторов EGA и VGA в графическом режиме. Этот файл находится в текущем каталоге

GraphResult: integer;

- Эта функция, при обращении к ней, возвращает специальный код (целое число), в зависимости от того, как прошло выполнение любой графической процедуры или функции.
- Код 0 (grOk) - успешное выполнение. Дальнейшая работа программы возможна лишь тогда, когда код функции GraphResult равен grOk.

CloseGraph;

- Выключение графического режима.

Шаблон графической программы

```
program Ex_1;
uses Graph;
var Gd, Gm: integer;
begin
   Gd := VGA; { графический адаптер VGA }
   Gm := VGAhi; { графический режим VGAhi (640x480)x16 }
   Initgraph(Gd, Gm, ''); { включить графический режим Драйвер в текущем каталоге }

   if GraphResult = grOk then begin { если режим включился успешно }
      { сюда следует поместить основные конструкции (операторы) }
      Readln;
      CloseGraph; { выключение графического режима }
   end;
end.
```

Line(x1, y1, x2, y2: integer);

Рисует на экране отрезок, соединяющий точки (x1, y1) и (x2, y2).

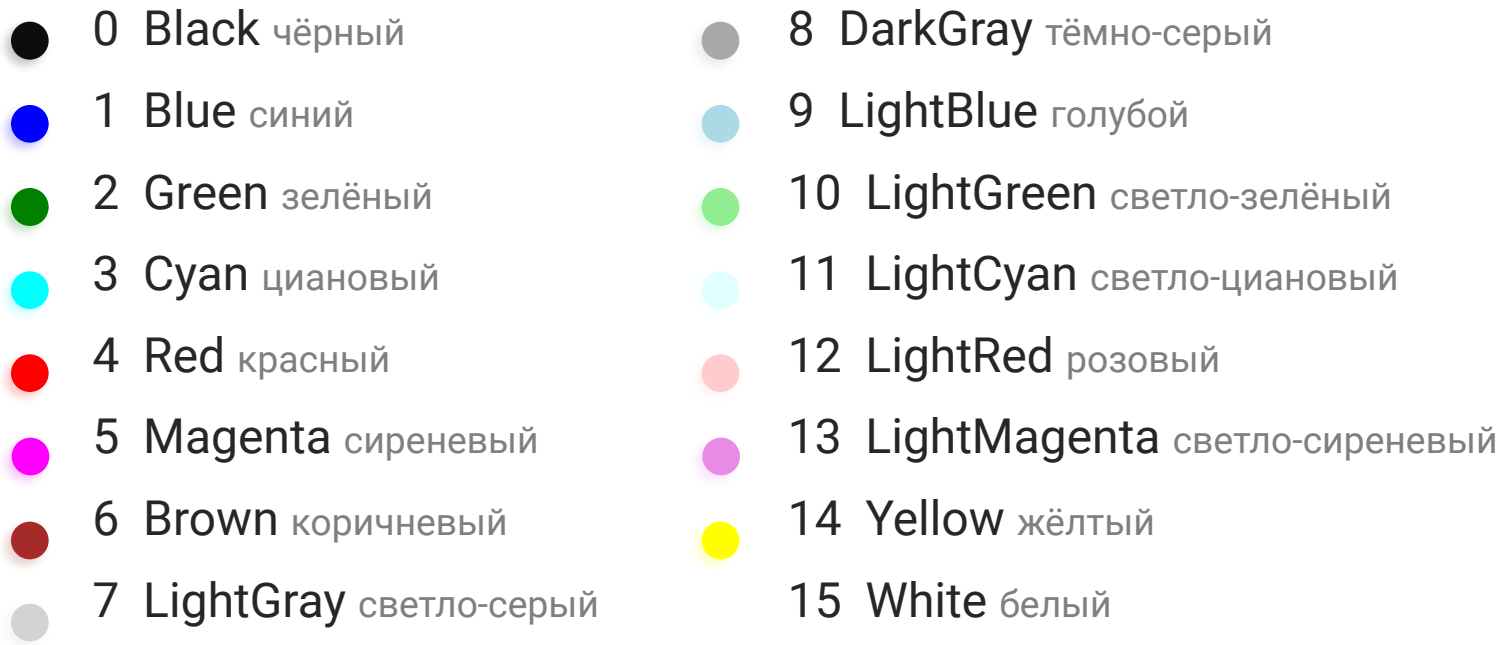
```
program Ex_1; { нарисовать треугольник }
uses Graph;
var Gd, Gm: integer;
begin
   Gd := VGA; { графический адаптер VGA }
   Gm := VGAhi; { графический режим VGAhi (640x480)x16 }
   Initgraph(Gd, Gm, '');

   if GraphResult = grOk then begin
      line(120, 210, 520, 210); { основание }
      line(120, 210, 320, 10); { левая сторона }
      line(320, 10, 520, 210); { правая сторона }
   end;
end.
```

Текущий цвет рисования

При выполнении описанной выше программы мы получим изображение треугольника из белых линий. Обладая цветным монитором VGA, мы можем создавать рисунки, используя различные цвета и оттенки. В любом случае рисование происходит так называемым «текущим цветом рисования». По умолчанию, текущий цвет рисования белый.

SetColor(Color: word);

Для изменения текущего цвета рисования, т. е. для задания цвета рисования прямых, окружностей, точек и пр. используется процедура SetColor. В качестве единственного её параметра Color нужно указать целое число, являющееся кодом цвета.


0 Black чёрный
1 Blue синий
2 Green зелёный
3 Cyan цианový
4 Red красный
5 Magenta сиреневый
6 Brown коричневый
7 LightGray светло-серый
8 DarkGray тёмно-серый
9 LightBlue голубой
10 LightGreen светло-зелёный
11 LightCyan светло-цианový
12 LightRed розовый
13 LightMagenta светло-сиреневый
14 Yellow жёлтый
15 White белый

```
program Ex_2; { цветной треугольник }
uses Graph;
var Gd, Gm: integer;
begin
   Gd := VGA; { графический адаптер VGA }
   Gm := VGAhi; { графический режим VGAhi (640x480)x16 }
   Initgraph(Gd, Gm, '');

   if GraphResult = grOk then begin
      setcolor(lightmagenta);
      line(120, 210, 520, 210); { основание }
      setcolor(lightcyan);
      line(120, 210, 320, 10); { левая сторона }
      setcolor(green);
      line(320, 10, 520, 210); { правая сторона }
   end;
end.
```

SetColor(Color: word);

Возвращает текущий цвет рисования.

GetBkColor: word;

Возвращает текущий цвет фона.

SetBkColor(Color: word);

Данная процедура устанавливает текущий цвет фона. По умолчанию текущий цвет фона - чёрный.

ClearDevice;

Очищает графический экран, закрашивая его в текущий цвет фона. Устанавливает указатель текущей позиции в точку с координатами (0, 0).

PutPixel(x, y: integer; Pixel: word);

Рисует точку с координатами (x, y) цветом Pixel.

GetPixel(x, y: integer): word;

Возвращает цвет точки с координатами (x, y).

Rectangle(x1, y1, x2, y2: integer);

Строит контур прямоугольника из линий текущего цвета.

Circle(x, y: integer; r: word);

Используется для рисования окружностей с тремя целочисленными параметрами:
x, y - координаты центра окружности; r - радиус.

Arc(x, y: integer; StA, EndA, Radius: word);

Нарисует дугу окружности текущим цветом.

x, y - координаты центра окружности данной дуги;

Sta - начальный угол; (в градусах)

EndA - конечный угол; (в градусах)

Radius - радиус дуги.

Ellipse(x, y: integer; stA, endA: word; xr, yr: word);

Рисует дугу эллипса текущим цветом.

x, y - координаты центра эллипса;

stA - начальный угол;

endA - конечный угол;

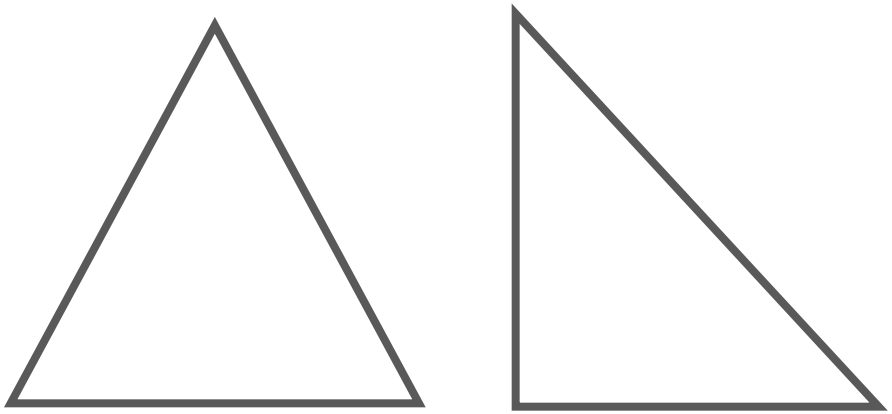
xr, yr - горизонтальная и вертикальная полуоси эллипса.

SetLineStyle(Ln,P,T);

- Рисование линий, окружностей и их элементов, контуров прямоугольников осуществляется линиями.
- Процедура SetLineStyle позволяет изменять параметры этих линий. Параметры этой процедуры - целые числа.
- Ln - стиль линии: 0 = сплошная; 1 = пунктирная; 2 = штрихпунктирная; 3 = штриховая; 4 = заданная пользователем.
- T - толщина линии: 1 = нормальная; 3 = толстая.
- P - шаблон: 0 - во всех случаях для Ln от 0 до 3;
- Для Ln = 4 устанавливается шаблон линии в виде двухбайтового числа, каждый бит которого равен 1, если его надо высветить и 0 - в противном случае. Например линия в виде 1111111100000000 в шестнадцатеричной форме следует записать $AA00.

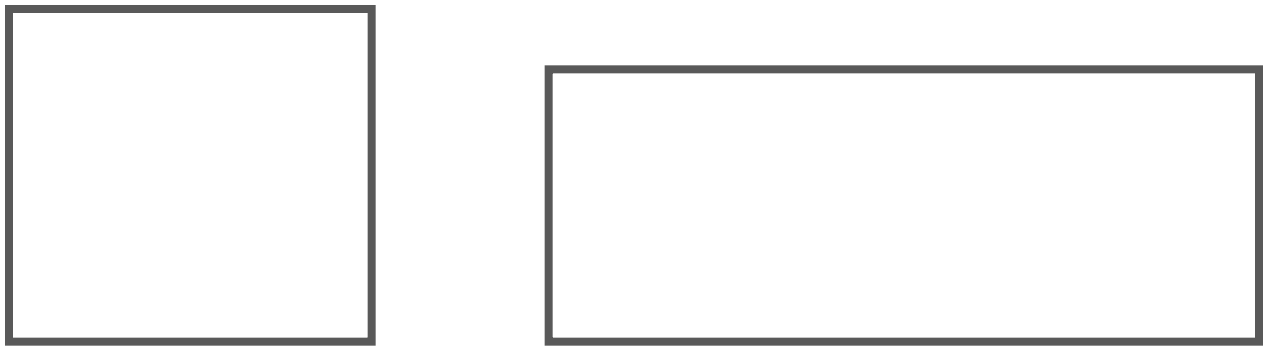
Задания для самостоятельной работы

Треугольники



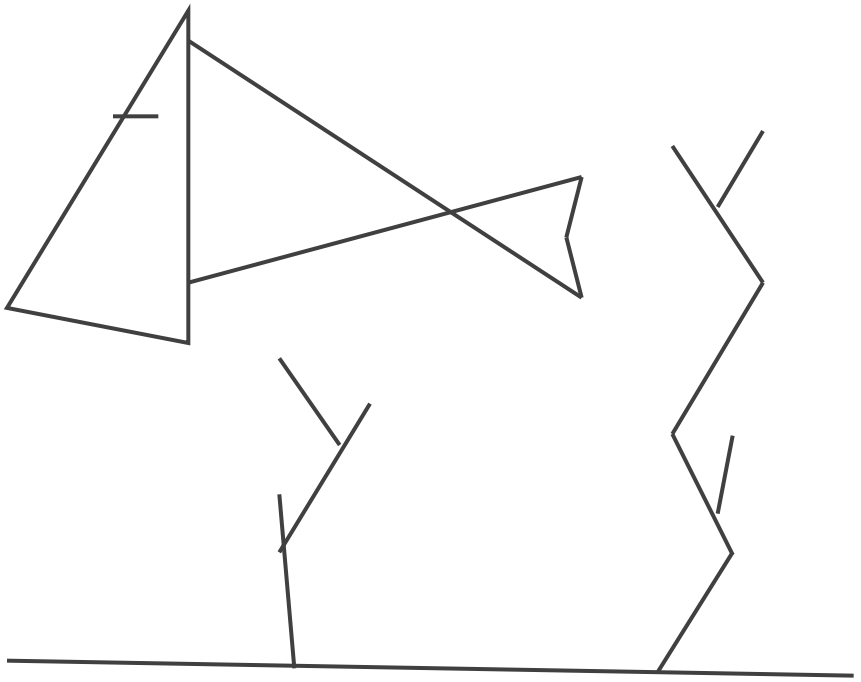
Задания для самостоятельной работы

Квадрат, прямоугольник

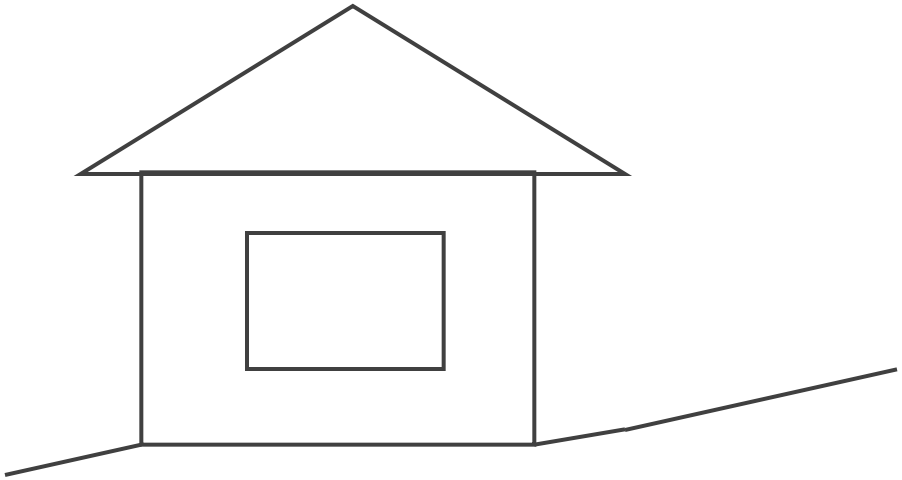


Задания для самостоятельной работы

Аквариум



Дом



OutTextXY(X, Y: Integer; TextString: String);

Выводит текст в заданное место экрана.

SetTextStyle(Font, Direction, CharSize: Word);

Устанавливает текущий шрифт, направление (горизонтальное или вертикальное) и размер текста.

SetViewPort(X1, Y1, X2, Y2: Integer; ClipOn: Boolean);

Устанавливает прямоугольное окно на графическом экране. Параметр ClipOn определяет «отсечку» элементов изображения, не умещающихся в окне.

ClearViewPort;

Очищает выделенное графическое окно, закрашивает его в цвет фона.

Текущая позиция.
Абсолютные координаты.

Текущая позиция - это координаты на экране воображаемого «указателя координат». Как правило, изменение положения «текущей позиции» происходит «незаметно», при использовании графических процедур и функций.

Кроме процедур и функций для управления графическим режимом существует несколько процедур и функций, в которых текущая позиция изменяется при указании абсолютных (непосредственных) координат экрана.

Перемещение указателя координат. Относительные координаты.

Кроме процедур, которые перемещают текущий указатель, выполняя конкретное действие (создавая изображение линии или фигуры), существуют специальные процедуры для изменения положения текущего указателя координат.

MoveTo(x, y: integer);

Перемещает текущий указатель координат в точку окна с координатами (x, y).
Точка на экране не высвечивается.

MoveRel(Dx, Dy: integer);

Перемещает указатель координат «позицию» на заданное расстояние (Dx, Dy) по отношению к его предыдущему положению.
Точка на экране не высвечивается.

Процедуры MoveTo и MoveRel выполняют одно и то же действие - перемещение текущего указателя координат. Но аргументами MoveTo - являются абсолютные координаты экрана, а аргументами MoveRel - так-называемые относительные координаты.

Смещение по отношению к предыдущему положению называют относительными координатами, или координатами по отношению к предыдущему положению.

MoveTo(x, y: integer);

Перемещает текущий указатель координат в точку окна с координатами (x, y).
Точка на экране не высвечивается.

MoveRel(Dx, Dy: integer);

Перемещает указатель координат «позицию» на заданное расстояние (Dx, Dy) по отношению к его предыдущему положению.
Точка на экране не высвечивается.

LineTo(x, y: integer);

Проводит линию текущего цвета из текущей позиции в точку с координатами (x, y). Текущая позиция «указатель» устанавливается в точке (x, y).

LineRel(Dx, Dy: integer);

Проводит линию текущего цвета из текущей позиции в точку, сдвинутую относительно исходной позиции на величины (Dx, Dy). Текущая позиция «указатель» перемещается относительно предыдущего на величину (Dx, Dy).

Определение координат текущей позиции

GetX: integer;

Возвращает координату X текущей позиции на экране.

GetY: integer;

Возвращает координату Y текущей позиции на экране.

GetMaxX и GetMaxY

Возвращает значения максимальных координат экрана в текущем режиме работы, соответственно, по горизонтали и вертикали.

Пример 1

Программа демонстрирует возможности изображения линий в графическом режиме.

Меняем стили рисования линий.

```
program Lines;
uses Graph, crt; { подключение к программе библиотек Crt и Graph }
var
   Key: Char;
   LineStyle: Word; { номер стиля рисования линии }
   Style: String; { название стиля }
   Gd, Gm: Integer; { тип и режим работы графического драйвера }
begin
   Gd := VGA; { графический адаптер VGA }
   Gm := VGAhi; { графический режим VGAhi (640x480)x16 }
   Initgraph(Gd, Gm, '');

   if GraphResult = grOk then begin
      SetBkColor(LightGray);
      SetColor(Red);
      {цвет фона и цвет рисования }
      OutTextXY(120, 100, 'Рисуем линию от точки (200, 200) к точке (400, 280)');
      Line(200, 200, 400, 280);
      Key := ReadKey; { приостановление исполнения программы }
      ClearViewPort; { очистка окна }
...
```

```
    OutTextXY(240, 80, 'Рисуем ломанную');
    Rectangle(110, 120, 520, 400);{ рисование рамки }
    MoveTo(Round(GetMaxX/2), Round(GetMaxY/2));{ указатель в центре окна }
    Repeat { цикл прерывается нажатием любой клавиши }
       LineTo(Random(GetMaxX-250)+120, Random(GetMaxY-210)+120);
  Delay(1000);
    until KeyPressed;
    Key := ReadKey;
    ClearViewPort;
    OutTextXY(190, 80, 'Меняем стили рисования линий');

    For LineStyle := 0 to 3 do begin
        SetLineStyle(LineStyle, 0, 1);
   Case LineStyle of
        0: Style := 'Сплошная';
        1: Style := 'Точечная';
        2: Style := 'Штрихпунктирная';
        3: Style := 'Пунктирная'
        end;
...
```

```
        Line(120, 150+LineStyle*50, 430, 150+LineStyle*50);
        OutTextXY(450, 145+LineStyle*50, Style);
      end;
      Key := ReadKey;
      ClearViewPort; { очистка окна }

      OutTextXY(180, 80, 'Меняем толщину рисования линий');
      SetLineStyle(0, 0, 1); { толщина 1 пиксел }
      Line(140, 200, 430, 200);
      OutTextXY(450, 195, 'Нормальная');
      SetLineStyle(0, 0, 3); { толщина 3 пиксела }
      Line(140, 250, 430, 250);
      OutTextXY(450, 245, 'Тройная');
      ReadLn;
      CloseGraph; { закрытие графического режима }
   end;
end.
```

Пример 2

Программа демонстрирует возможности изображения символов в графическом режиме (требует наличия в текущем каталоге файлов шрифтов *.chr).

```
program Symbols;
uses Graph, crt; { подключение к программе библиотек crt и Graph }
var
   Key: Char;
   Font: String; { названия шрифтов }
   Size, MyFont : Word;
   Gd, Gm: Integer; { тип и режим работы графического драйвера }
begin
   Gd := VGA; { графический адаптер VGA }
   Gm := VGAhi; { графический режим VGAhi (640x480)x16 }
   Initgraph(Gd, Gm, '');

   if GraphResult = grOk  then begin
      SetTextStyle(DefaultFont, HorizDir, 2);
      OutTextXY(140, 80, 'Меняем размер символов');
      OutTextXY(220, 100, 'и цвет фона');

      For Size := 0 to 13  do begin { Size - цвет фона и размер символов }
          SetBkColor(Size);  { изменение цвета фона }
          Rectangle(135, 425, 470, 450);  { рисование рамки }
      SetTextStyle(DefaultFont, HorizDir, 1);
      OutTextXY(150, 435, 'Для продолжения нажмите любую клавишу !');
      SetTextStyle(DefaultFont, HorizDir, Size);
      OutTextXY(250-Size*15, 200, 'HELLO');
      Key := ReadKey; ClearViewPort;
      end;
ReadLn;
```

```
SetBkColor(LightGray); SetColor(Red);  { цвет фона и цвет рисования }
SetTextStyle(DefaultFont, HorizDir, 2);  { установка шрифта, направления и размера символов }
OutTextXY(70, 100, 'Располагаем строку горизонтально');
SetTextStyle(DefaultFont, VertDir, 2);
OutTextXY(310, 150, 'и вертикально');
Key := ReadKey; ClearViewPort;
SetTextStyle(DefaultFont, HorizDir, 2);
{ установка шрифта, направления и размера символов }
OutTextXY(220, 30, 'Меняем шрифты');

For MyFont := 0 to 9 do begin { цикл по номерам шрифтов }
    Case MyFont of
     0: Font := '0 - Точечный (Default)';
     1: Font := '1 - Утроенный (Triplex)';
     2: Font := '2 - Уменьшенный (Small)';
     3: Font := '3 - Прямой (SansSerif)';
     4: Font := '4 - Готический (Gothic)';
     5: Font := '5 - Рукописный';
     6: Font := '6 - Курьер';
     7: Font := '7 - Красивый (Таймс Italic)';
     8: Font := '8 - Таймс Roman';
     9: Font := '9 - Курьер увеличенный';
    end;
```

```
        SetTextStyle(MyFont, HorizDir, 2);
    OutTextXY(40, 70+MyFont*35, 'abcdfxyz   0123456789'); { вывод текста }
    SetTextStyle(DefaultFont, HorizDir, 1);
    OutTextXY(410, 80+MyFont*35, Font){ вывод названия шрифта }
     end;
     OutTextXY(380, 60, 'N шрифта Описание');
     ReadLn;
  end;
  CloseGraph; { закрытие графического режима }
end.
```

Рисование заполненных фигур

Параметры рисования заполненных фигур задаются процедурой SetFillStyle. С её помощью устанавливаются текущий орнамент рисования и текущий цвет рисования.

SetFillStyle(p: word; c: word);

Задаёт стандартный орнамент и цвет заполнения фигур.
p - номер стандартного орнамента;
c - цвет заполнения фигур (номер цвета в палитре).

Константы орнамента заполнения

0 EmptyFill	Заполнение цветом фона
1 SolidFill	Однородное заполнение цветом заполнения
2 LineFill	Заполнение ----------
3 LtSlachFill	Заполнение ///
4 SlachFill	Заполнение /// толстыми линиями
5 BkSlachFill	Заполнение \\\ толстыми линиями
6 LtBkSlachFill	Заполнение \\\
7 HathFill	Заполнение клеткой
8 XHathFill	Заполнение косой клеткой
9 InterleaveFill	Заполнение частой клеткой
10 WideDotFill	Заполнение редкими точками
11 CloseDotFill	Заполнение частыми точками

Контур заполненных фигур

Обратите внимание!

Цвет заполнения фигур ничего общего с цветом рисования линий (который устанавливается процедурой SetColor) не имеет.

Цветом, установленным процедурой SetColor рисуются контуры этих фигур. Стиль линии контура (толщина и прерывистость) устанавливается процедурой SetLineStyle.

Bar(x1, y1, x2, y2: integer);

Строит прямоугольник, закрашенный текущим орнаментом и цветом заполнения.
x1, y1, x2, y2 - координаты левого верхнего и правого нижнего углов прямоугольника.
Координаты текущей позиции не изменяются.

Bar3D(x1, y1, x2, y2: integer; Depth: Word; Top: Boolean);

Строит параллелепипед, закрашенный текущим орнаментом и цветом заполнения.

x1, y1, x2, y2 - координаты левого верхнего и правого нижнего углов передней грани.
Depth - ширина боковой грани (отсчитывается по горизонтали).
Top - признак включения верхней грани (если true - верхняя грань вычерчивается, если false - верхняя грань не отображается).
Координаты текущей позиции совпадают с координатами правого, дальнего, верхнего угла параллелепипеда.
При Depth = 0, координаты теку щей позиции совпадают с координатами левого верхнего угла.

FillEllipse(x, y: integer; xr, yr: word);

Строит эллипс, закрашенный текущим орнаментом и цветом заполнения.
x, y - координаты центра эллипса.
xr, yr - горизонтальная и вертикальная полуоси эллипса.
Координаты текущей позиции (x, y).

FloodFill(x, y: integer; Border: Word);

Закрашивает область, ограниченную непрерывной линией, текущим орнаментом и цветом заполнения.
x, y - координаты любой точки, внутри закрашиваемой области;
Border - цвет линии, до которой производится закрашивание.

PieSlice(x, y: integer; stA, endA, r: word);

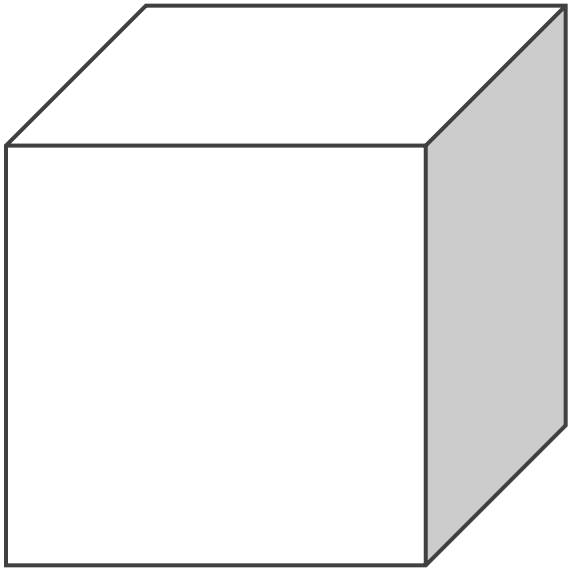
Строит сектор круга, закрашенный текущим орнаментом и цветом заполнения.
x, y - координаты центра сектора круга;
stA - начальный угол; (в градусах)
endA - конечный угол; (в градусах)
r - радиус сектора.
Координаты текущей позиции (x, y).

Sector(x, y: integer; stA, endA, xr, yr: integer);

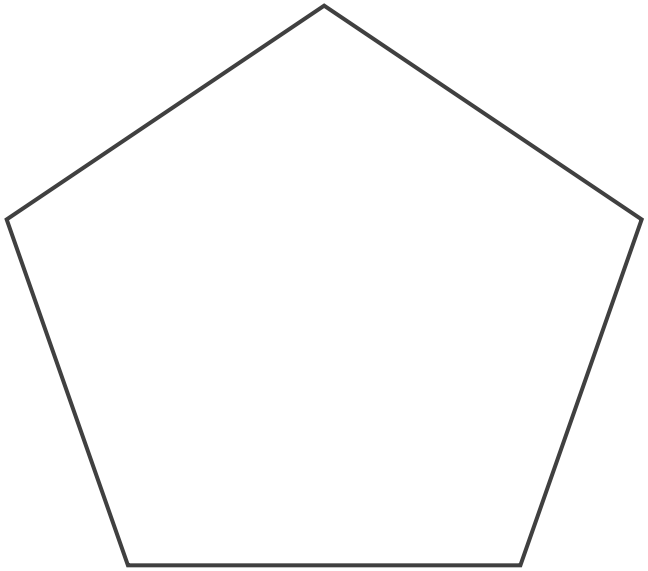
Строит сектор эллипса, закрашенный текущим орнаментом и цветом заполнения.
x, y - координаты центра эллипса;
stA - начальный угол; (в градусах)
endA - конечный угол; (в градусах)
xr, yr - горизонтальная и вертикальная полуоси эллипса.
Координаты текущей позиции (x, y).

Задания для самостоятельной работы

Куб



Пятигранник



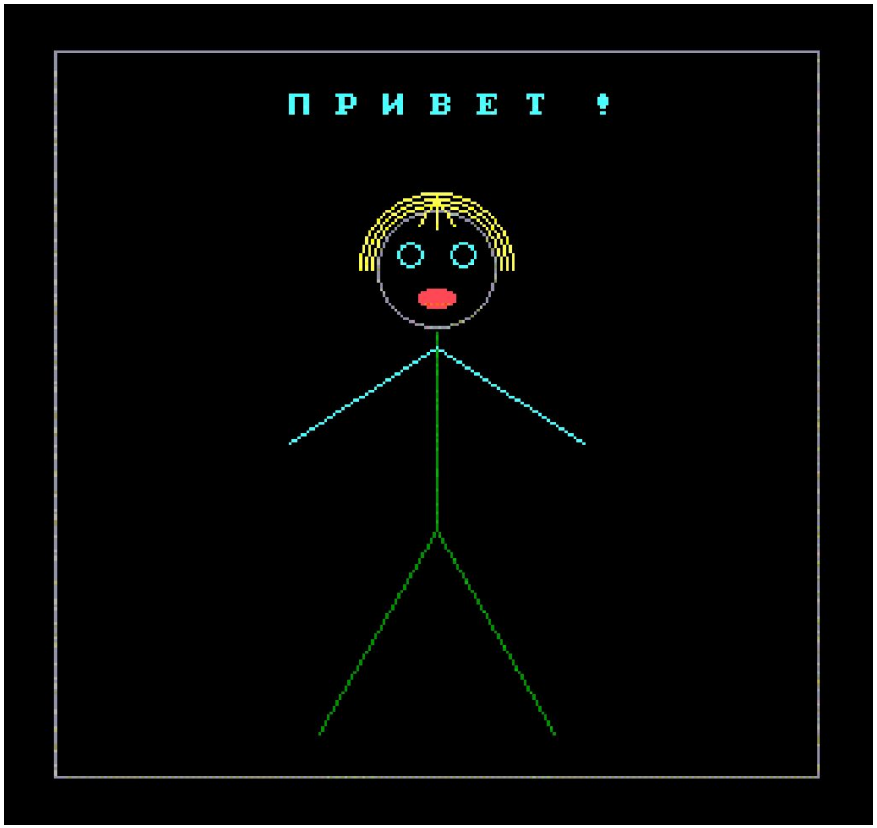
Пример 3

Программа рисует закрашенный прямоугольник, меняя случайным образом цвет, тип штриховки и высоту тона звукового сопровождения.

```
program MusicColor;
uses crt, Graph;
var
   Gd, Gm: Integer;  { тип и режим работы графического драйвера }
begin
   Gd := VGA;  { графический адаптер VGA }
   Gm := VGAhi;  { графический режим VGAhi (640x480)x16 }
   Initgraph(Gd, Gm, '');

   if GraphResult = grOk  then begin
      SetColor(White);  { установка белого цвета рамки }
      Rectangle(130, 130, 460, 370);  { рисование рамки }
      Randomize;  { инициализация датчика случайных чисел }

      Repeat { цикл прерывается нажатием любой клавиши }
         Sound(Random(2000));  { изменение высоты звука }
         Delay(Random(1000));  { задержка }
     SetFillStyle(Random(4), Random(16));  { смена типа штриховки и цвета }
     Bar(140, 140, 450, 360);  { рисование закрашенного прямоугольника }
      until KeyPressed;

      NoSound;  { отмена звука }
      CloseGraph; ReadLn;  { закрытие графического режима }
   end;
end.
```

Пример 4

Программа рисует человечка, делающего утреннюю зарядку.



```
program Animation;
uses crt, Graph;
const { вертикальные и горизонтальные координаты положения рук }
   vert: Array[1..3] of Integer = (190, 157, 120);
   Horizont: Array[1..3] of Integer = (200, 190, 200);
var Gd, Gm,  i, j: Integer;
begin
   Gd := VGA; { графический адаптер VGA }
   Gm := VGAhi; { графический режим VGAhi (640x480)x16 }
   Initgraph(Gd, Gm, '');
if GraphResult = grOk  then begin
   SetColor(LightGray);  { установка светлосерого цвета для рамки}
   Rectangle(20, 20, 480, 400);  { рисование рамки }
   SetColor(LightCyan);  { установка яркоголубого цвета для текста }
   OutTextXY(200, 40, 'П Р И В Е Т !');
   SetColor(LightGray); Circle (250, 130, 20);  { голова }
   SetColor(Yellow); Arc(250, 130, 0, 180, 26);  { волосы }
   Arc(250, 130, 0, 180, 24); Arc(250, 130, 0, 180, 22);
   Line(250, 105, 244, 115); Line(250, 105, 250, 116);  { чубчик }
   Line(250, 105, 256, 115);
   SetColor(LightCyan); Circle(241, 125, 4);  { левый глаз }
   Circle(259, 125, 4);  { правый глаз }
   SetColor(LightRed);
   SetFillStyle(SolidFill, LightRed);
   FillEllipse(250, 140, 6, 3);  { рот }
```

```
   Setcolor(Green);
   Line(250, 152, 250, 220);  { туловище }
   Line(250, 220, 210, 290);  { левая нога }
   Line(250, 220, 290, 290);  { правая нога }
   repeat { цикл прерывается нажатием любой клавиши }
      { Последовательный вывод трех положений рук: вниз, на уровне плеч, вверх }
      for i := 1 to 3 do begin
         SetColor(LightCyan); Sound(200*i);
         Line(250, 157, Horizont[i], Vert[i]);  { левая рука }
         Line(250, 157, 500-Horizont[i], Vert[i]);  { правая рука }
         Delay(300); { задержка }
{ смена цвета на черный для повторного рисования рук в том же положении ("стирания" их с экрана) }
         SetColor(Black);
         Line(250, 157, Horizont[i], Vert[i]);  { левая рука }
         Line(250, 157, 500-Horizont[i], Vert[i]);  { правая рука }
      end
   until Keypressed;
   SetColor(LightCyan);
   Line(250, 157, Horizont[3], Vert[3]);  { левая рука поднята }
   Line(250, 157, 500-Horizont[3], Vert[3]);  { правая рука поднята}
   for i := 1 to 10 do begin{ звуковая трель }
       Sound(1000);  Delay(50); Sound(1500); Delay(50);
   end;
   NoSound; { выключение звука }
   CloseGraph;
end;
end.
```

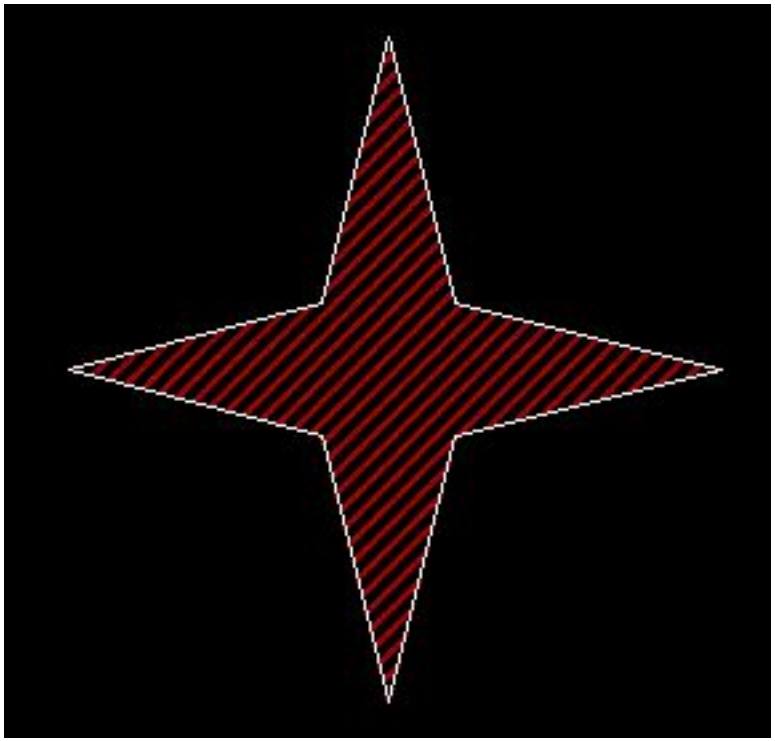
Пример 5

Движение прямоугольника по диагонали.

```
program sdf1;
uses graph, crt;
var driver, mode, x, y: integer;
begin
   driver := detect;
   initgraph(driver, mode, 'c:\tp\bgi');
   setcolor(5);
   repeat
      for x := 1 to getmaxy-120 do begin
          SetFillStyle(1, 10);
          y := x+8;
          bar(x, y, x+15, y+20);
          delay(20);
          clrscr;
      end;
      delay(60);
   until keypressed;
   closegraph;
end.
```

Пример 6

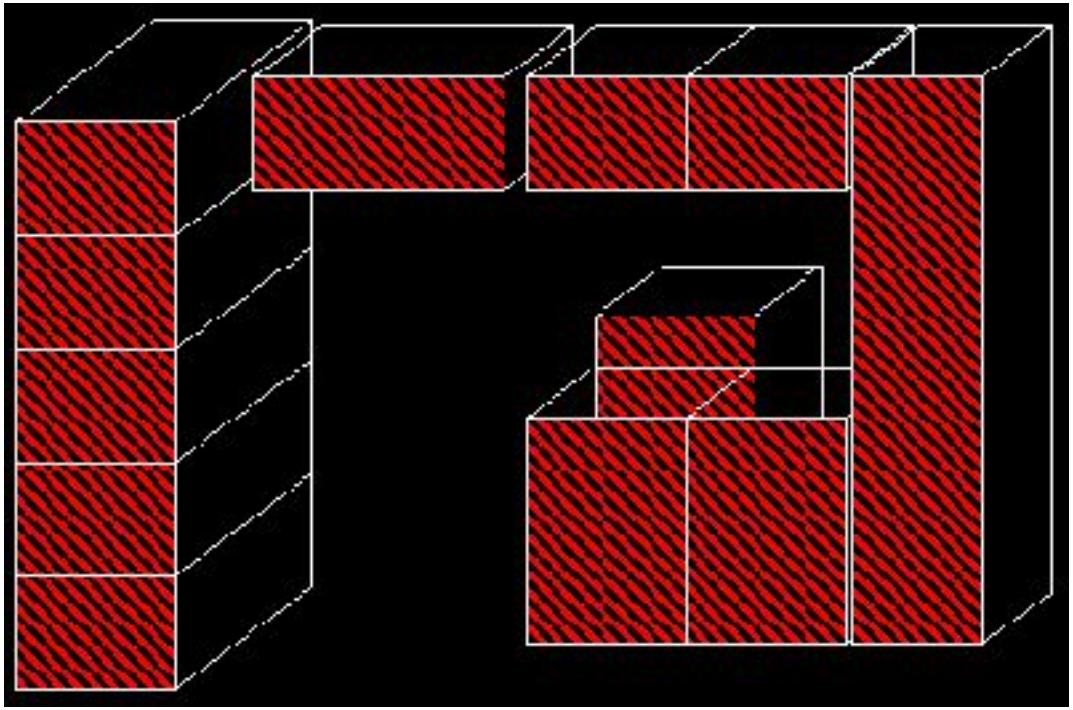
Программа рисует на экране звезду и закрашивает её, используя 12 типов штриховки.



```
program Star;
uses crt, Graph;
const { массив координат вершин многоугольника (звёзды) }
   TopsStar: Array[1..18] of Integer = (300, 125, 325, 225, 425, 250, 325, 275, 300, 375, 275, 275,
180, 250, 275, 225, 300, 125);
var i, j, Gd, Gm: Integer;
begin
   Gd := VGA; { графический адаптер VGA }
   Gm := VGAhi; { графический режим VGAhi (640x480)x16 }
   Initgraph(Gd, Gm, '');

   if GraphResult = grOk  then begin
      { установка шрифта, направления и размера символов }
      SetTextStyle(DefaultFont, HorizDir, 2);
      OutTextXY(220, 60, 'S T A R ');
      SetTextStyle(DefaultFont, VertDir, 2);
      OutTextXY(140, 150, 'S T A R ');
      SetTextStyle(DefaultFont, VertDir, 2);
      OutTextXY(500, 150, 'S T A R ');
      i := 0;

      repeat
         j := i mod 12; { j - остаток от деления i на 12 }
    SetFillStyle(j, Random(13)); { штриховка и фон }
    FillPoly(9, TopsStar); { рисование и штриховка звёзды }
    Inc(i); { увеличение i на 1 }
    Delay(500);
      until KeyPressed; { завершение цикла нажатием любой клавиши }

      CloseGraph; end;
end.
```

Пример 7

Программа демонстрирует возможности изображения объёмных предметов и столбиковых диаграмм.

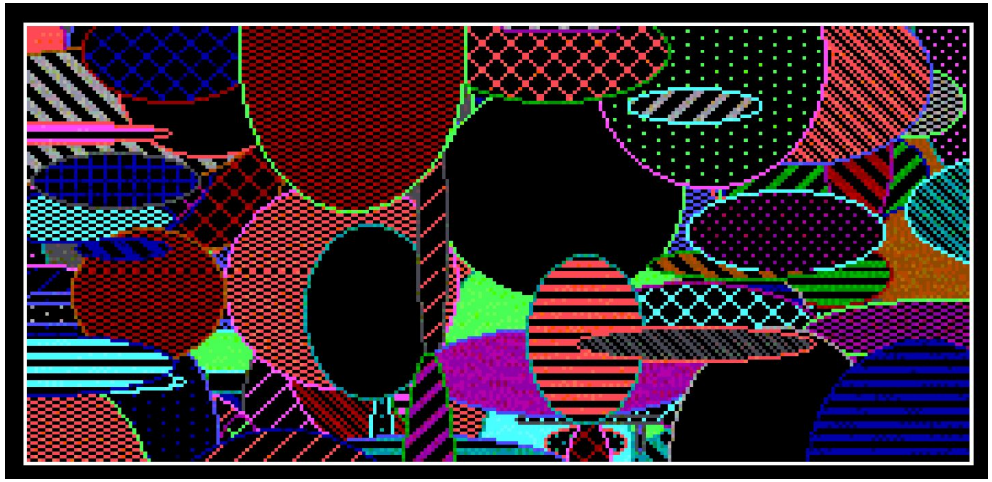


```
program Design;
uses
   Graph, Crt; { подключение к программе библиотек Crt и Graph }
const
   Height: Array[1..8] of Integer = (40, 150, 90, 240, 190, 120, 50, 90);
   { массив высот столбиков диаграммы }
var
   Color: Word; {код цвета}
   Key: Char;
   i, x, y, y1, h: Integer;
   Gd, Gm: Integer; { тип и режим работы графического драйвера }
begin
   Gd := VGA; { графический адаптер VGA }
   Gm := VGAhi; { графический режим VGAhi (640x480)x16 }
   Initgraph(Gd,Gm,'');

   if GraphResult = grOk then begin
      y := 120; h := 50; y1 := 140;
      SetTextStyle(DefaultFont, HorizDir, 2); { шрифт, направление, размер }
      OutTextXY(160, 20, 'Конструируем интерьер');
      SetFillStyle(5, LightRed); { тип штриховки и цвет (ярко красный) }

      for i := 4 downto 1 do begin { рисование параллелепипедов заданного размера }
          Bar3D(75, y1+i*h, 145, y1+(i+1)*h, 60, TopOff); Delay(200);
      end;
```

```
    Bar3D(75 , y1 , 145, y1+h , 60, TopOn); Delay(200);
    Bar3D(180, y , 290, y+h , 30, TopOn); Delay(200);
    Bar3D(330, 225 , 400, y+4*h , 30, TopOn); Delay(200);
    Bar3D(300, y+3*h, 370, y+5*h , 30, TopOn); Delay(200);
    Bar3D(370, y+3*h, 440, y+5*h , 30, TopOn); Delay(200);
    Bar3D(300, y , 370, y+h , 30, TopOn); Delay(200);
    Bar3D(370, y , 440, y+h , 30, TopOn); Delay(200);
    Bar3D(442, y , 500, y+5*h , 30, TopOn); Delay(200);
    Rectangle(135, 425, 470, 450); { рисование рамки для сообщения }
    SetTextStyle(DefaultFont, HorizDir, 1);
    OutTextXY(150, 435, 'Для продолжения нажмите любую клавишу');
    Key := ReadKey; ClearViewPort; { очистка окна }
    SetTextStyle(DefaultFont, HorizDir, 2);
    OutTextXY(100, 20, 'Рисуем столбиковую диаграмму');
    x := 50; Randomize; { инициализация датчика случайных чисел }

    for i := 1 to 8 do begin { цикл по столбикам диаграммы }
     Color := Random(12)+1; { задание кода цвета (кроме черного) }
     SetFillStyle(i, Color); { задание типа штриховки и цвета }
     SetColor(Color);
     { рисование столбика }
     Bar3D(x, 350-Height[i], x+50, 380, 20, TopOn);
     x := x+70; { изменение координаты x };
     Delay(200) { задержка }
    end;

    Key := ReadKey; CloseGraph; { закрытие графического режима }
  end;
end.
```

Пример 8

Программа демонстрирует работу с пикселами, случайными эллипсами и секторами.



```
program RandomFigures;
uses Graph, crt;
var
   Key: char;
   GD, GM: integer;
   Radius, MaxX, MaxY, Ugol: word;
begin
   Gd:=VGA; { графический адаптер VGA }
   Gm:=VGAhi; { графический режим VGAhi (640x480)x16 }
   Initgraph(Gd,Gm,'');

   if GraphResult = grOk then begin
      SetTextStyle(DefaultFont, HorizDir, 2);
      { установка шрифта, направления и размера символов }
      OutTextXY(160, 50, 'Рисуем звездное небо');
      Rectangle(110, 90, 520, 380); { рисование рамки }
      randomize; { инициализация датчика случайных чисел }

      repeat { цикл прерывается нажатием любой клавиши }
     { вывод пикселя в области, ограниченной рамкой }
         PutPixel(Random(GetMaxX-250)+120, Random(GetMaxY-210)+100, Random(15));
         Delay(50) { задержка }
      until KeyPressed;

   Key := ReadKey; ClearDevice; { очистка графического экрана }
```

```
    SetColor(White); { цвет рисования }
    OutTextXY(140, 30, 'Рисуем случайные эллипсы');
    Rectangle(100, 70, 560, 420); { рисование рамки }
    MaxX := GetMaxX; MaxY := GetMaxY; Radius := MaxY div 10;
    SetLineStyle(0, 0, 1); { толщина и стиль линии }
    SetViewPort(101, 71, 559, 419, ClipOn); { установка окна внутри рамки }
    Randomize; { инициализация датчика случайных чисел }

    repeat { цикл прерывается нажатием любой клавиши }
        SetBkColor(Black); { цвет фона } SetColor(Random(13)+1); { цвет рисования }
        SetFillStyle(Random(12), Random(13)+1); { образец и цвет штриховки }
        FillEllipse(Random(MaxX), Random(MaxY), { координаты центра эллипса }
        Random(Radius), Random(Radius)); { полуоси эллипса }
    until KeyPressed;

    Key := ReadKey;
    ClearDevice; { очистка графического экрана }
    SetColor(White); SetViewPort(1, 1, GetMaxX, GetMaxY, ClipOn);
    OutTextXY(140, 20, 'Рисуем случайные секторы');
    Rectangle(90, 60, 570, 420); { рисование рамки }
    SetViewPort(92, 62, 569, 419, ClipOn); { установка окна внутри рамки }
    repeat { цикл прерывается нажатием любой клавиши }
        SetFillStyle(Random(12), Random(13)+1); { изменение штриховки и цвета }
        Ugol := Random(360); { угол сектора }
        Sector(Random(MaxX-200), Random(MaxY-180), Random(Ugol), Ugol,
        Random(Radius*2), Random(Radius*2)); { рисование сектора }
    until KeyPressed;
    ClearViewPort; { очистка окна } CloseGraph; { закрытие графического режима }
  end;
end.
```

Задания для самостоятельного выполнения

1. Нарисуйте круговую диаграмму, состоящую из 10 заполненных секторов, используя различные орнаменты и цвета заполнения.
2. Нарисуйте заполненные различным орнаментом и цветом заполнения треугольник, трапецию и звезду.
3. Нарисуйте свои инициалы в виде заполненных многоугольников.
4. Изобразите горизонтальную последовательность состоящую из 16 различных заполненных эллипсов.
5. Нарисуйте разноцветную мишень.