**Уважаемые студенты групп!**

**Вашему вниманию представлена лекция на тему «ОБРАБОТКА ГРАФИЧЕСКОЙ ИНФОРМАЦИИ».**

**Задание**

1. Прочитать внимательно лекцию.
2. Законспектировать лекцию в рабочую тетрадь не менее 3-5 страницы рукописного текста. В конспекте лекции обязательно должно быть приведены примеры.
3. Решить приведенные в лекции в контрольных вопросах задачи.
4. Дата предоставления фотоотчета лекции до10.04.2023.

С уважением Ганзенко Ирина Владимировна

!!! Если возникнут вопросы обращаться по телефону, 0721134803 (вацап), +79591134803 (телеграмм)

disobuch.ganzenko2020@mail.ru

# ОБРАБОТКА ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

**План**



## 1 Теоретические положения

Любая информация на экране монитора изображается совокупностью близко расположенных друг к другу точек, темные - пикселей. Программист может управлять светимостью и цветом любого пикселя, что позволяет формировать на экране любые изображения, в том числе рисунки, графики, чертежи, символы. Информация может выводиться на экран монитора в одном из двух режимов - графическом или текстовом.

В графическом режиме каждый пиксель имеет свои координаты - положение относительно левого верхнего угла экрана с координатами 0, 0 (см. Рис. 1). Состояние любого пикселя определяется содержанием специальной буферной памяти, которую называют видеопамятью. Такая видеопамять входит в состав аппаратных средств ПЭВМ, называют

видеоадаптером. Существуют различные типы видеоадаптеров, которые отличаются объемом видеопамяти и возможностями ее управления (CGA-адаптер, EGA-адаптер, VGA- адаптер и другие). Так, например, объем видеопамяти CGA-адаптера составляет 16 Кбайт. В более совершенных адаптерах - EGA и VGA - объем видеопамяти увеличен и составляет от 64 до 256 Кбайт.

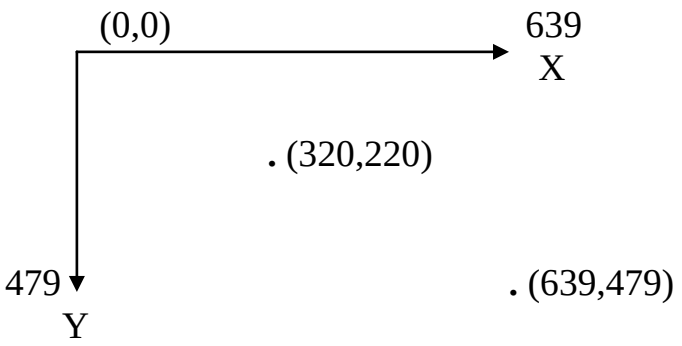



Рисунок1 - координаты графического экрана для CGA-адаптера в режиме высокого разрешения

Содержание видеопамяти связано с пикселями следующим образом: нулевое положение любого разряда видеопамяти означает отсутствие светимости соответствующего пикселя, единичное состояние - наличие его светимости.

Экран ПЭВМ формируется из множества пикселей, цвет и светимость которых может задаваться программно. Состояние каждого пикселя кодируется одним или несколькими разрядами (битами) в видеопамяти. К видеопамяти могут одновременного доступа как программа для установки состояния любого пикселя, так и электронные схемы телевизионного развертки изображения на экране. Эти схемы рассматривают видеопамять как одну очень длинную срока бит, которые сохраняют команды на установление светимости и цвета того или иного, пикселя. Электронные схемы "опрашивают" видеопамять последовательно, бит за битом, в темпе электронного луча, обегает экран.

В текстовом режиме экран монитора рассматривается как совокупность знакомест - частей экрана, служащих для изображения отдельных символов. Каждое знака представляет собой матрицу пикселей и имеет свои координаты - положение относительно левого верхнего угла экрана с координатами 1,1. Графические образы (матрицы) всех символов, которые могут уместиться в то или иное, знака, и хранятся в постоянном запоминающем устройстве (ПЗУ). Каждый символ имеет свой внутренний код, согласно которому электронные схемы генерации текста выбирают из ПЗУ его графический образ. В текстовом режиме программист может управлять отображением графической информации, задавая код символа и координаты знакомест.

## 1.2 Стандартные процедуры и функции управления графическим экраном

Следующие процедуры и функции помещены в стандартный модуль GRAPH.TPU, что входит в комплект поставки TURBO PASCAL. Для того, чтобы они стали доступны компилятору, в программе сразу после оператору PROGRAM необходимо использовать специальный оператор USES GRAPH.

Всего к содержанию модуля входит 73 процедуры и функции, дающие пользователю различные возможности управления экраном в графическом режиме.

Настройка графических процедур на работу с конкретным адаптером достигается при подключении нужного графического драйвера. Драйвер - это специальная программа, которая осуществляет управление теми или этими техническими средствами ПЭВМ. Графический драйвер управляет графическим адаптером. Графические драйверы разработаны фирмой BORLAND почти для всех адаптеров. Обычно они находятся на диске в отдельном подкаталоге BGI в виде файлов с расширением BGI (от англ. BORLAND GRAPHICS INTERFACE - графический интерфейс фирмы BORLAND). Например, CGA.BGI - драйвер для CGA-адаптера, EGA.BGI и VGA.BGI - драйверы для адаптеров EGA и VGA, и тому подобное.

## 1.2 Переход в графический режим и возвращение в текстовый

Стандартный состояние ПЭВМ после ее включения, а также ко времени реализации программы со среды TURBO PASCAL соответствует работе экрана в текстовом режиме, поэтому любая программа, которая использует графические средства компьютеру, должна определенным образом инициировать графический режим работы адаптера. После завершения работы программы, ПЭВМ возвращается в текстовый режим.

**процедура INITGRAPH**

Инициирует графический режим работы адаптера; формат обращения:

INITGRAPH ( "драйвер", "режим", "путь")

здесь:

«Драйвер» - переменная типа INTEGER, которая определяет тип графического драйвера;

"Режим" - переменная типа INTEGER, что определяет режим работы графического адаптера;

"Путь" - выражение типа STRING, который сохраняет путь к файлу драйвера.

Ко времени обращения к процедуре на одном из дисков должен находиться файл, в состав которого входит нужен графический драйвер.

Процедура загружает этот драйвер в ОЗУ и переводит адаптер в графический режим работы. Тип драйвера должен соответствовать типу графического адаптера.

Для указания типа драйвера в модуле являются следующие константы

CONST
DETECT = 0 (* Режим авто определения типа *)
CGA = 1;
MCGA = 2;
EGA = 3;
EGA 64 = 4;
EGA MONO = 5,
IBM 8514 = 6;
HERC MONO = 7;
ATT 400 = 8;
VGA = 9;
PC 3270 = 10;

Большинство адаптеров может работать в различных режимах.

**функция GRAPHRESULT**

Возвращает значение типа INTEGER, в котором закодировано результат последнего обращения к графическим процедурам. Обращение:

**GRAPHRESULT**

Если ошибка найдена, режим экономии 0, в противном случае - отрицательное число, указывает на номер ошибки. При анализе ошибок возможно использование следующих мнемонических констант, определенных в модуле GRAPH:

CONST
GROK = 0 (* Нет ошибок *)
GRINITGRAPH = 1; (* Не инициированный графический режим *)
GRNOTDETECTED = 2; (* Не определен тип драйвера *)
тому подобное.

После обращения к функции GRAPHRESULT признак ошибки сбрасывается, поэтому следующее обращение к ней вернет 0.

**процедура CLOSEGRAPH**

Завершает работу адаптера в графическом режиме и восстанавливает текстовый режим работы экрана. Обращение:

**CLOSEGRAPH**

## 1.3 Координаты окна, страницы

**Функции GETMAXX и GETMAXY**

Возвращают значение типа INTEGER, в которых содержатся максимальные координаты экрана в настоящем режиме работы относительно горизонтали и вертикали. Обращение:

**GETMAXX или GETMAXY**

Функции GETX и GETY

Возвращают значение типа INTEGER, в которых содержатся текущие координаты курсора относительно горизонтали и вертикали.

## 1.4 Процедуры модуля Graph

Здесь приведены (Таблица 1) наиболее интересные и часто используемые процедуры для рисования. Все остальные могут появятся позже, хотя в их использовании и нет необходимости. К неописанному ниже, относятся такие процедуры, как DrawPoly и FillPoly, позволяющие рисовать многоугольники и закрашивать их, процедуры для работы с экранными координатами и другие. Но того, что есть, конечно, хватает.

Очень важно! В языке Паскаль используется нестандартная система координат. Ноль ее расположен в левом верхнем углу, ось Ох направлена вправо, а Оу - вниз.

*Таблица 1. Процедуры модуля Graph*

| **Процедура** | **Формат** | **Действие** |
|---|---|---|
| SetColor | SetColor(a: word); | Устанавливает цвет, которым будет осуществляться рисование |
| SetBkColor | SetBkColor(a: word); | Устанавливает цвет тела |
| SetFillStyle | SetFillStyle(a, b: word);<br>a - стиль закрашивания, b - цвет | Устанавливает стиль и цвет закраски |
| ***FloodFill*** | FloodFill (X, Y:Integer; Border:Word); | Заливает область вокруг точки (X, Y) к линии цвета Border, используя текущий стиль и цвет заливки. |
| SetLineStyle | SetLineStyle(a, b, c: word);<br>a - стиль линии, b- образец построения линии (может устанавливаться пользователем), c-толщина линии | Устанавливает стиль и толщину линии |
| SetTextStyle | SetTextStyle(a, b, c: word); | Устанавливает шрифт, стиль и размер текста |
| SetFillPattern | SetFillPattern(Pattern: FillpatternType; Color: word); | Выбирает шаблон заполнения, определенный |

| | | |
|---|---|---|
| | Pattern - маска | пользователем |
| ClearDivice | ClearDivice; | Очищает экран и устанавливает текущий указатель в начало |
| SetViewPort | SetViewPort(x1, y1, x2, y2: integer, Clip: boolean); | Устанавливает текущее окно для графического вывода |
| ClearViewPort | ClearViewPort | Очищает окно |
| PutPixel | PutPixel(a, b, c: integer); | Рисует точку цветом с у (x, y) |
| Line | Line(x1, y1, x2, y2: integer); | Рисует линию от (x1, y1) до (x2,y2) |
| Rectangle | Rectangle(x1, y1, x2, y2: integer); | Рисует прямоугольник с диагональю от (x1, y1) до (x2, y2) |
| Bar | Bar(x1, y1, x2, y2: integer); | Рисует зарисованный прямоугольник |
| Bar3D | Bar3D(x1, y1, x2, y2, d: integer; a: boolean); | Рисует трехмерную полосу (параллелепипед) |
| Circle | Circle(x, y, r: word); | Рисует окружность радиуса r с центром в точке (x, y) |
| Arc | Arc(x, y, a, b, R: integer);<br>a, b- начальный и конечный углы в градусах | Рисует дугу с начального угла до конечного, используя (x,y) в качестве центра |
| Ellipse | Ellipse(x, y, a, b, Rx, Ry: integer);<br>a, b - начальный и конечный углы в градусах | Рисует эллиптическую дугу от начального угла к конечному, используя (x, y) как центр |
| FillEllipse | FillEllipse(x, y, Rx, Ry: integer);<br>Rx, Ry - вертикальная и горизонтальная оси | Рисует закрашенный эллипс |
| MoveTo | MoveTo(x, y: integer); | Перемещает текущий указатель в (x, y) |
| MoveRel | MoveRel(x, y: integer); | Пересуває поточний покажчик на задану відстань від поточної позиції на x по горизонталі і на y по вертикалі |
| OutText | OutText(text: string); | Выводит текст от текущего указателя |

| | | |
|---|---|---|
| OutTextxy | OutTextxy(x, y: integer; text: string); | Выводит текст из (x, y) |
| Sector | Sector(x, y, a, b, Rx, Ry: integer);<br>a, b - начальный и конечный углы в градусах | Рисует и заполняет сектор эллипса |

Далее приведена цветовая шкала, которая может использоваться в графическом режиме (см. Таблица 2).

*Таблица 1.* ***Процедуры модуля Graph***

| Цвет | Код | Цвет | Код |
|---|---|---|---|
| Black – черный | 0 | DarkGray – темно-серый | 8 |
| Blue – синий | 1 | LightBlue – голубой | 9 |
| Green - зеленый | 2 | LghtGreen – ярко-зеленый | 10 |
| Gyan – бирюзовый | 3 | LightGyan – ярко-березовый | 11 |
| Red – красный | 4 | LightRed – ярко-красный | 12 |
| Magenta – малиновый | 5 | LightMagenta – ярко-малиновый | 13 |
| Brown – коричневый | 6 | Yellow – желтый | 14 |
| LightGray – светло-серый | 7 | White – белый | 15 |

Далее приведены типы штриховки, с помощью штриховки можно заполнить любые фрагменты изображения узором, периодически повторяющимся. Для указания типа штриховки используют следующие заранее определенные константы:

```
CONST:

EMPTYFILL = 0; (*штриховка фоном (узор отсутствует)*)
SOLIDFILL = 1; (*соединительная штриховка*)
LINEFILL = 2; (*штриховка линиями*)
LTSLASHFILL = 3; (*штриховка //////////*)
SLAHSFILL = 4; (*штриховка утолщенными / / / / / / / / / /*)
BKSLASHFILL = 5; (*штриховка \\\\\\\\\\*)
LTBKSLASHFILL = 6; (*штриховка утолщенными \ \ \ \ \ \ \ \ \ \*)
HATCHFILL = 7; (*штриховка + + + + +*)
XHATCHFILL = 8; (*штриховка X X X X X *)
```

INTERLEAVEFILL = 9; (*штриховка в прямоугольную клетку*)
WIDEDOTFILL = 10; (*штриховка жидкими точками*)
CLOSEDOTFILL = 11; (*штриховка частыми точками*)
USERFILL = 12; (*штриховка определяется пользователем*)

## 2 Пример решения задач с использованием графики

**Пример 1.** Составить программу для вывода на экран 3-х отрезков разного цвета, окружности, прямоугольника.

Для решения задачи необходимо задаться расположением указанных элементов. Из расположения определяем координаты (рис. 1).

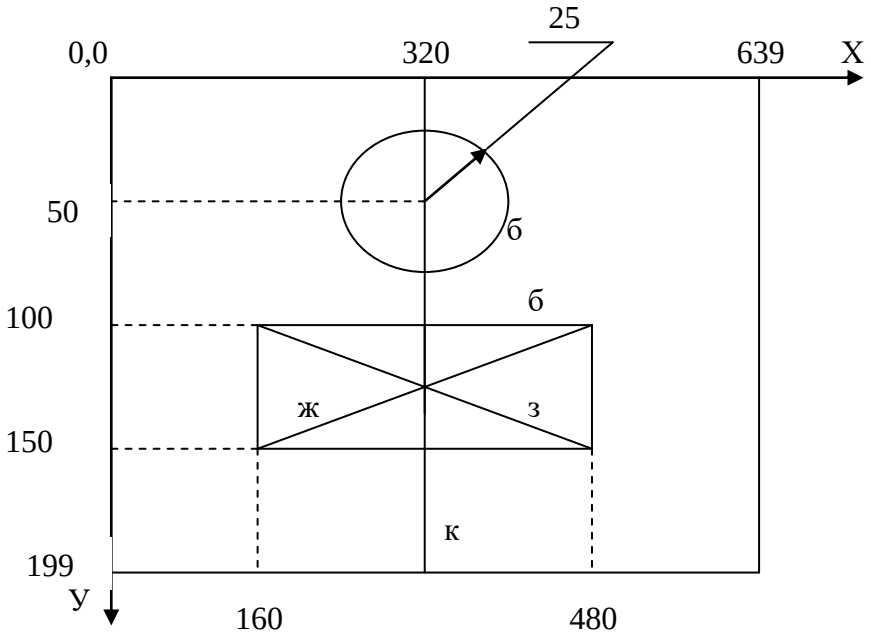
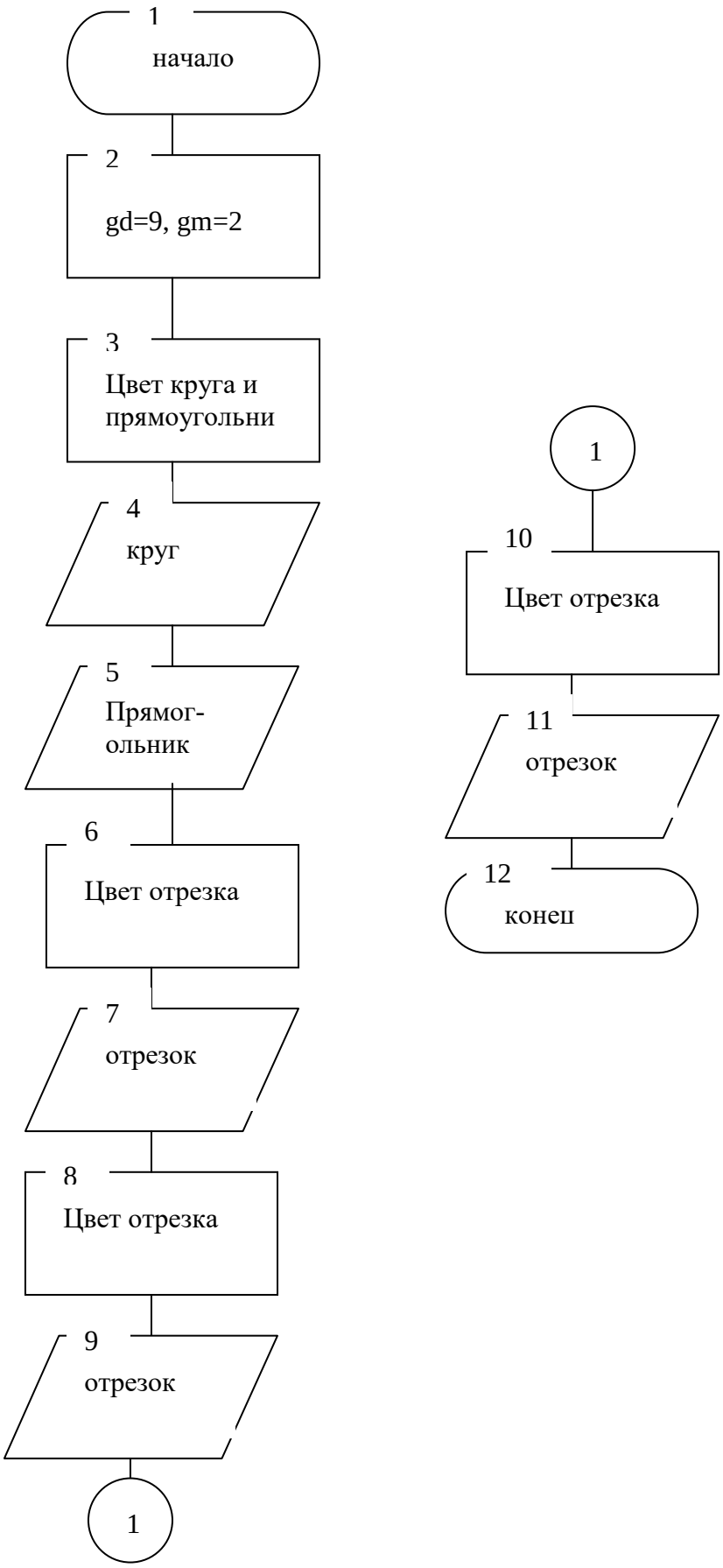



Рисунок 1 - Размещение координат

```
Program picture;
UsesGraph¸ crt;
Var
gd,gr:integer;
begin
clrscr;
gd:=9;                                  // драйвер для монитора VGA
gm=2;                                   // работы монитора 640× 480
InitGraph (gd,gm,’’);                   // инициализация графического режима
Setcolor (15); Circle (320,50,25);      // рисование круга
Rectangle (160,100,480,150);            // рисование прямоугольника
Setcolor (Red);                         // задание цвета отрезка
Line (320,0,320,199);                   // рисование отрезка
Setcolor (Green);                       // задание цвета отрезка
```

```
Line (160,100,480,150);          // рисование отрезка
Setcolor (Yellow);               // задание цвета отрезка
Line (160,150,480,100);          // рисование отрезка
Riadln;                          // задержка для просмотра рисунка на экране
Closegraph; end.                 //закрытие графического режима
```


1
начало
2
gd=9, gm=2
3
Цвет круга и прямоугольни
4
круг
5
Прямог-ольник
6
Цвет отрезка
7
отрезок
8
Цвет отрезка
9
отрезок
1
1
10
Цвет отрезка
11
отрезок
12
конец

**Пример 2.** Демонстрация различных видов закрашивания.

Суть задачи заключается в анализе готовой программы. На экране в соответствии с программой будет 10 концентрических колл разного цвета. На экране вместе с этим, будут концентрические кольца, залитые разным цветом и с разным типом заполнения (рис. 2).

**Идентификаторы программы:**

**gd-графический** драйвер для монитора VGA,
**gm-режим** работы монитора 640 480
**i-количество** окружностей, цвет границы закрашивания,
**n=1-тип** заполнителя,
**i+3-цвет** закрашивания, заполнения, заливки,
**r-радиус** окружности.

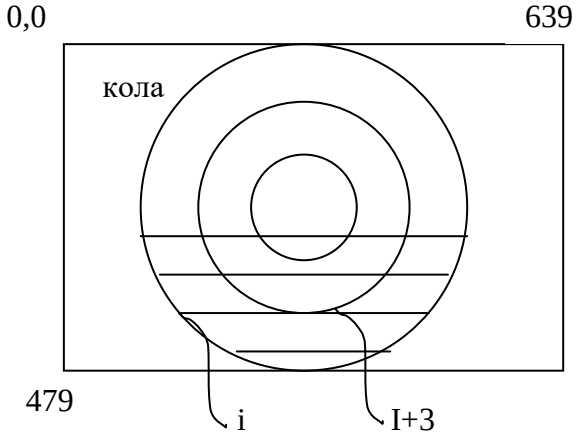
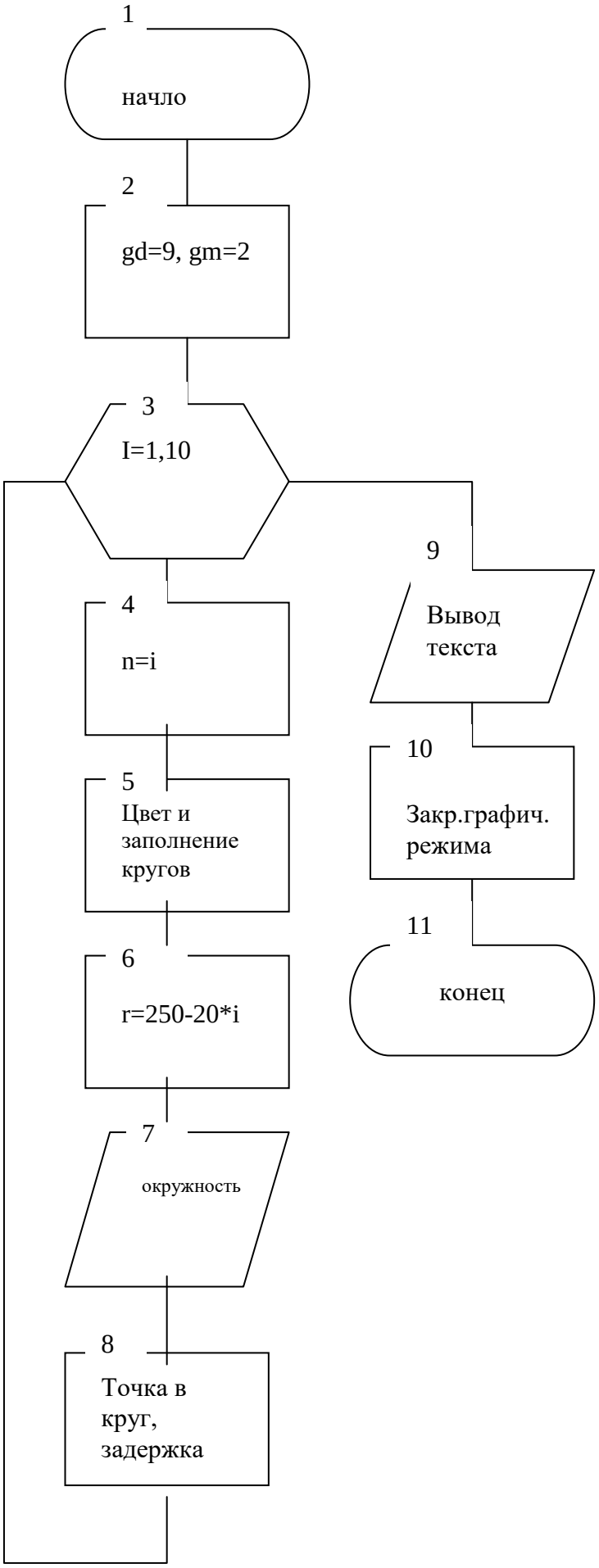



Рисунок 2–Концентрические кольца

```
program  Cinema;
UsesGraph, crt;
var
 gd, gr, i, n, r: integer;
begin
 gd:=9;
 gm:=2;
InitGraph( gd,gm,’’);
  for  i:=1  to  10  do
        begin
         n:=i;
{ цвет границы закрашивания }
         SetColor(i);
{выбор цвета и типа штриховки}
         SetFillStyle(n;i+3);
{расчет радіуса окружности}
         r:=250-20*i;
 {окружность}
         circle(320,240,r);
{ Заливает область вокруг точки 320,240 к линии и , используя текущий стиль
и цвет заливки }
          FloodFill(320,240,i);
{задержка рисования}
          Delay(10200);
         end;
{ Выводит текст 'круги' из точки 20,20}
         OutText,XY(20,20,'кола');
         readln;
```

{за5крытие графического режима}
CloseGraph; end.



## 3 Контрольные вопросы

1. Какая процедура устанавливает графический режим?
2. Как отличается система координат в текстовом и в графическом режиме?
3. Какая процедура закрывает графический режим?
4. Назвать процедуру для задания цвета фона.
5. Какая разница между процедурами SetFillStyle и FloodFill ?
6. Создать программу, которая демонстрирует возможные цвета графического режима, используя генератор случайных чисел .
7. Объяснить следующую строку: GD:= Detect;
8. Из чего состоит графический экран дисплея ?
9. Где начало отсчета точек в графическом режиме?
10. С каким цветом должен совпадать цвет предела в процедуре FloodFill ?
11. Что можно сказать о черном цвете в графике TP?
12. Вывести круг с демонстрацией всех возможностей (изменение цвета и типа границы, цвета и типа заливки, положения, размеров, растяжения.).
13. Вывести сектор с демонстрацией всех возможностей (изменение цвета и типа границы, цвета и типа заливки, положения, размеров, растяжения.).
14. Вывести эллипс с демонстрацией всех возможностей (изменение цвета и типа границы, цвета и типа заливки, положения, размеров, растяжения.).
15. Вывести изображение звездного неба пикселями случайных цветов .
16. Какова процедура рисует линию между двумя заданными точками?
17. Какова процедура рисует линию от текущей точки до заданной точки?
18. Назвать процедуру для вывода прямоугольника?
19. Назвать процедуру для вывода параллелепипеда?
20. Вывести на экран изображение множества разных цветовых пузырьков (случайные размеры и заполнение) .
21. Вывести на экран изображение вложенных прямоугольников.
22. Вывести на экран изображение вертикальных разноцветных линий на весь экран .
23. Вывести на экран изображение горизонтальных линий через интервал.
24. Вывести на экран изображение шахматной доски, с случайными цветами заливки.

25. Вывести на экран изображение решетки (свободные или заданные размеры).
26. Вывести на экран изображение круга, разбитого на секторы и разрисованного в виде зонтика.
27. Вывести на экран изображение светофора (со сменой цветов).
28. Вывести на экран изображение елки из треугольников.
29. Вывести изображение множества дуг, растущих к сектору заданного размера.