



Модуль Graph



Инициализация и завершение графического режима

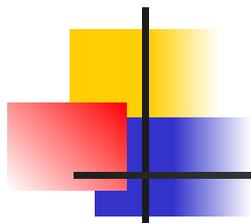
- **InitGraph (driver, mode : integer; path : string);**
 - При **driver:=detect** включается автоопределение режима работы видеоадаптера.
- **CloseGraph;**



Процедуры и функции, управляющие графическим режимом

- **RestoreCrtMode;** - Возвращение в текстовый режим.
- **SetGraphMode(mode:integer);** - Вход в графический режим, и очистка экрана.
- **GraphResult : integer;** - получение кода ошибки.

Система координат



(0,0)

(639,0)

X

Y

(0,479)



Процедуры и функции работы с точками

- **GetMaxX:integer;** - получение максимального значения координаты X
- **GetMaxY:integer;** - получение максимального значения координаты Y
- **GetPixel:(x,y:integer):word;** - получение цвета пикселя с заданными координатами
- **GetX:integer;** - получение координаты X текущей позиции
- **GetY:integer;** - получение координаты Y текущей позиции
- **PutPixel(x,y:integer; C:word)** - рисует точку с заданными координатами и цветом



Перемещение указателя CP (current pointer)

- **Moverel (dx,dy : integer);** - относительное смещение курсора из текущей точки
- **Moveto (x,y : integer);** - передвижение курсора в точку (x,y)

Процедуры и функции управления цветом



- **SetBkColor(C:word);** - задание цвета фона
- **GetBkColor:word;** - получение цвета фона
- **SetColor(c:word);** - задание текущего цвета изображения
- **GetColor:word;** - получение текущего цвета линий и контуров
- **GetMaxColor:word;** - получение максимального кода цвета в палитре

Процедуры работы с линиями

- **Line (X1,Y1,X2,Y2:integer);** - построение линии от одной точки до другой
- **LineRel (dX,dY:integer);** - построение линии на вектор
- **LineTo(X,Y:integer);** - построение линии от текущей точки до заданной

Процедуры работы с линиями

■ **SetLineStyle(LineStyle,Pattern,T:word);**

- задает параметры линии (стиль, шаблон и толщину).

■ Стиль:

0 – сплошная; **1** – пунктирная;

2 – штрихпунктирная; **3** – штриховая;

4 – заданная пользователем (задается шаблон)

■ Толщина линии:

1 – нормальная; **3** – толстая

■ Шаблон: **0**

Процедуры работы с примитивами

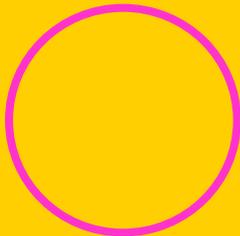
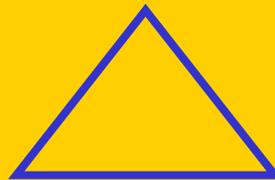
- **Rectangle(X1,Y1,X2,Y2:integer);** - построение прямоугольника
- **Arc(X,Y:integer; StartAngle,EndAngle,Radius:word);** - построение дуги окружности
- **Circle(X,Y:integer;Radius:word);** - построение окружности
- **Ellipse(X,Y:integer; StartAngle,EndAngle,Xradius,YRadius:word);** - построение дуги эллипса



Пример 1.

```
uses crt, graph;  
var driver, mode : integer; path : string;  
begin  
driver:=detect;  
initgraph (driver, mode, 'c:\program\bp\bgi');  
setbkcolor(14); setcolor(2);  
while not keypressed do; {readln;}  
closegraph;  
end.
```

Нарисовать по образцу





Построение многоугольника

DrawPoly(N, A)

где N- количество вершин,

A – массив координат вершин

```

uses Graph, crt;
type k=record
    x1,y1:integer;
end;
var a:array [1..100] of k; i,m,r,n:integer;

Begin
    readln(n);
    m:=detect; Initgraph(m,r,'c:\bp\bgi');
Randomize;
for i:=1 to n do
    begin
        a[i].x1:=random(630);
        a[i].y1:=random(470);
    end;
a[n+1]:=a[1];
i:=n+1;
drawpoly(i,a);
while not keypressed do;
end.

```

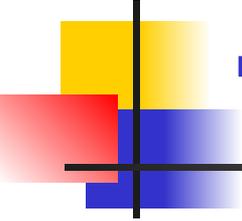
Процедуры построения закрашенных фигур

- **SetFillStyle(Pattern,Color)** - задает орнамент и цвет заполнения фигур

где **Pattern**:

- 0 - Заполнение цветом фона;
- 1 - Однородное заполнение цветом;
- 2 - Заполнение - - -;
- 3 - Заполнение / / /;
- 4 - Заполнение / / / толстыми линиями;
- 5 - Заполнение \ \ \ толстыми линиями;
- 6 - Заполнение \ \ \;
- 7 - Заполнение клеткой;
- 8 - Заполнение косой клеткой;
- 9 - Заполнение частой сеткой;
- 10 - Заполнение редкими точками;
- 11 - Заполнение частыми точками.

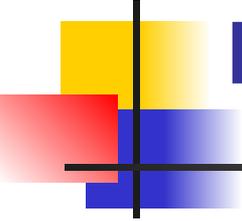
Процедуры построения закрашенных фигур

- 
- **Bar(X1,Y1,X2,Y2)** - Построение закрашенного прямоугольника
 - **FloodFill(X,Y,ColorBk)** - Заполнение замкнутой области из точки(X,Y) до границы с цветом ColorBk
 - **FillEllipse(X,Y,Xradius,Yradius)** - Построение закрашенного эллипса
 - **PieSlice(X,Y,StAngle,EndAngle,Radius)** - Построение закрашенного сектора круга
 - **Sector(X,Y,StAngle,EndAngle,Xradius,Yradius)** - Построение закрашенного сектора эллипса



Процедура очистки графического экрана

- **ClearDevice;**



Процедуры для работы с текстом

- **OutText(text:string);** - выводит текст в текущее положение
- **OutTextXY(x,y: integer;text:string);** - выводит текст в заданное положение
- **SetTextStyle(Font, Direction, Size:word);** - установка стиля текста
- **SetTextJustify(Horis, Vert:word);** - установка выравнивания текста

Движение по кругу

```
a:=0;  
while not keypressed do  
  begin  
    circle(320,240,100);  
    x:=320+trunc(100*sin(a));  
    y:=240+trunc(100*cos(a));  
    fillellipse(x,y,20,20);a:=a+1;  
    delay(20000);  
    cleardevice;  
  end;
```



Условный оператор для выхода из цикла

```
if keypressed then exit;
```

Задания

1. движение стрелки секундомера по (против) часовой стрелки, пока не нажата клавиша;
2. движение по периметру экрана по (против) часовой стрелки, пока не нажата клавиша;
3. горизонтальное движение по синусоиде, пока не нажата клавиша;
4. вертикальное движение по косинусоиде, пока не нажата клавиша;
5. движение со случайным выбором координат, пока не нажата клавиша.
6. светофор
7. елка с мигающими шарами
8. перемещающийся номер телефон

Построение графика функции

■ Создание декартовой системы координат

- Построение осей координат: 2 прямые через точки **(0,240)**, **(639,240)**, **(320,0)**, **(320,479)**;
- Построение стрелок, указывающих направление осей и обозначение осей;
- Построение насечек на осях и обозначение насечек:

Пусть единичный отрезок равен **80 px**.

Построение насечек на оси **OX**

v:=-3; { число под насечкой на оси }

i:=1; { счетчик насечек }

while v<=3 do begin

moveto(80*i,238);{ построение насечки }

lineto(80*i,242);

str(v:2,s);{ преобразование числа в строку }

outtextxy(80*i,244,s);{ вывод числа под насечкой }

v:=v+1;

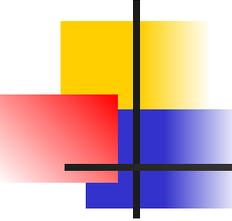
i:=i+1;

end;

Построение графика функции $y=\sin(x)$

- Используем преобразованные координаты
 $x=x+320$, $y=240-y$

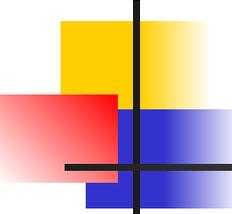
```
x:=-pi; y:=sin(x); {координаты начала}  
moveto(trunc(x*80+320),trunc(240-y*80));  
  {перемещение указателя}  
while x<=pi do begin  
x:=x+0.01;{шаг ломаной}  
y:=sin(x);  
  lineto(trunc(x*80+320),trunc(240-y*80));  
  {построение звена}  
  end;
```



Построение графика функции

Для построения графиков функций необходимо :

- 1) определить саму функцию
- 2) определить координаты начала координат и прорисовать оси координат
- 3) прорисовку графика следует сделать в цикле. Целесообразно использовать цикл с малым шагом (счетчик типа `real`).



Построение графика функции

Для построения графиков функций необходимо :

- 4) y -координатой служит заданная функция со счетчиком в качестве аргумента.
- 5) преобразованные координаты точек графика складываются из координат начала координат «+» сама координата точки (для x -координаты) или «-» сама координата точки (для y -координаты).
- 6) для растяжения графика по осям координат следует применять коэффициенты масштабирования, на которые надо домножить x - и y -координату .

```

uses graph, crt;
var parx, pary, i:real; d, m, x0, y0:integer; ch:char;
function fc(i:real):real;
begin           {задается любая функция}
fc:=sin(i);
end;
begin
readln( xmix, xmax, ymin, ymax);           {задание областей определения и значения}
d:=detect;  initgraph(d, m, ' ');
x0:=round(getmaxx/2);                       {определение координат центра}
y0:=round(getmaxy/2);
line(1, round(getmaxy/2), getmaxx, round(getmaxy/2)); {прорисовка осей}
line(round(getmaxx/2), 1 ,round(getmaxx/2), getmaxy);
parx:=getmaxx/(xmax-xmin)/; pary:=getmaxy/(ymax-ymin)/;
                                                {расчет коэффициентов масштабирования}
i:=xmin;
moveto(trunc(x0+i*parx), trunc(y0-fc(i)*pary));
                                                {перемещение курсора в начало графика}

while i<=xmax do
begin           {основной цикл прорисовки графика}
i:=i+0.01;
lineto(trunc(x0+i * parx),trunc(y0-fc(i) * pary)) ;
end;
readln;
closegraph; end.

```