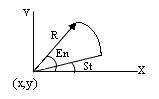
Для использования графических возможностей Турбо Паскаля необходимо в блоке описания uses подключить графический модуль **Graph**. Модуль содержит набор графических функций и процедур, основные из которых рассмотрены ниже.

## Графические процедуры

**Arc (x,y: integer; St,En,R: word);** {рисование дуги, где **x,y** - коорд. центра дуги, **R** - радиус, **St, En** - нач. и конечный углы. }



**ClearDevice**; {очистка экрана}

**Circle (x,y: integer; R: word);** {окружность радиуса **R**

cцентром в точке (**x,y)** }

**CloseGraph**; {закрытие графики}

**InitGraph (gd,gm: integer; Path: string);** {инициализация графики, где

**gd** - графический драйвер,

**gm** - графический режим,

**Path** - путь размещения файла

с расширением. *bgi*. }

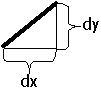
**Line (x1,y1,x2,y2: integer);** {рисование линии, проходящей через две точки:

с координатами (**x1,y1) и (x2,y2)** }

**LineRel (dx,dy: integer);** {рисование линии: **dx, dy** - приращения

координат к координатам текущей

позиции курсора}



**LineTo (x,y: integer);** {рисование линии, проходящую через текущую

позицию курсора и точку с координатами (**x,y**) }

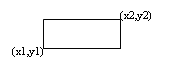
**Moveto (x,y: integer);** {задание координат текущей точки}

**OutText (T: string);** {вывод текста в текущую точку}

**OutTextXY (x,y: integer; T: string);** {вывод текста в точку с координатами **х, у**}

**Rectangle (x1,y1,x2,y2: integer);** {рисование прямоугольника: (**x1,y1) и (x2,y2) -** координаты противоположных вершин}

**PutPixel (x,y: integer; c: word);** {выводит точку с координатами (**х, у)** цветом **с**}



## Графические функции

**GetMaxX** - определяет максимальный размер экрана по оси x.

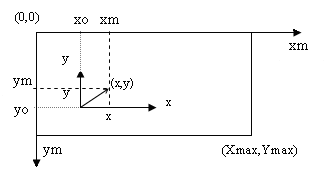
**GetMaxY** - определяет максимальный размер экрана по оси y.

**GetX** - координата текущей позиции курсора по оси x.

**GetY** - координата текущей позиции курсора по оси y.

***Машинные координаты.***

Начало машинной системы координат, направление осей, а также максимальные значения координат монитора показаны на рисунке



На рисунке приведена также машинная (xm, ym) и физическая (x,y) системы координат. Для изображения на экране точки с физическими координатами (x,y) необходимо определить ее машинные координаты (xm, ym). Расчетные формулы имеют следующий вид (попробуйте самостоятельно получить эти формулы):

xm=x0+x\*Mx,

ym=y0-y\*My,

где Mx, My-масштабы соответственно по осям x и y, которые показывают число пикселей в одной физической единице,

x, y - физические координаты точки,

xm, ym - машинные координаты точки,

x0, y0 - машинные координаты начала физической системы координат.

**Пример.** Нарисовать через весь экран горизонтальную и вертикальную линии, пересекающиеся в центре монитора.

Этапы разработки программы сведены в таблицу.

Таблица

|  |  |  |
| --- | --- | --- |
| N | *Этапы программирования* | *Выполнение* |
| 1. | *Постановка задачи* | Нарисовать через весь экран горизонтальную и вертикальную линии, пересекающиеся в центре монитора. |
| 2. | *Математическое описание* | *Изобразим вид экрана с указанием координат*  *требуемых линий* |
| 3. | *Разработка структограммы* |  |
| 4. | *Написание программы* | **Program P5;**  **Uses graph; {**подключение граф. модуля}  **Var gd,gm: integer;**  **Begin**  **gd: =detect; {**определение граф. драйвера**}**  **InitGraph (gd,gm,’ ‘); {**инициализация  графики**}**  **Line (0, round (GetMaxY/2), GetMaxX,**  **Round (GetMaxY/2));** {гориз. лин. }  **Line (round (GetMaxX/2), 0,Round (GetMaxX/2), GetMaxY);**  {вертик. лин. }  **Readln; {**пустой ввод**}**  **CloseGraph;** {закрытие графики}  **End.** |
| 5. | *Отладка и получение результатов* | *Выполнить самостоятельно* |

**Пример.** Написать программу построения графика функции y=x2 для x∈ [-1; 1].

Этапы разработки программы сведены в таблицу.

Таблица

|  |  |  |
| --- | --- | --- |
| N | *Этапы программирования* | *Выполнение* |
| 1. | *Постановка задачи* | Построить график функции y=x2 для x∈ [-1; 1]. |
| 2. | *Математическое описание* | *Изобразим вид экрана, который мы хотели бы получить после выполнения программы.* |
| 3. | *Разработка структограммы* |  |
| 4. | *Написание программы* | **Program P6;**  **Uses graph;**  **Var gd,gm: integer;**  **x,y: real;**  **x0,y0,xm,ym,Mx,My: integer;**  **begin**  **gd: =detect;**  **InitGraph (gd,gm,’ ‘);**  **Mx: =300; x0: =320; My: =440; y0: =460;**  **{**выбраны для монитора 640×480 пикс. **}**  **For xm: =20 to 620 do**  **Begin**  **x: = (xm-x0) /Mx;**  **y: =sqr (x);**  **ym: =round (y0-y\*My);**  **PutPixel (xm,ym,1);**  **End;**  **Readln;**  **CloseGraph;**  **End.** |
| 5. | *Отладка и получение результатов* | *Выполнить самостоятельно* |

**Модуль Graph**

ТР обладает достаточно разнообразными средствами для работы со стандартным VGA экраном (возможно также использование и и других типов видеоадаптеров).

VGA адаптер имеет разрешение 640х480 пиксел (точка (0,0) в левом верхнем углу экрана), 16 цветов.

Перед началом работы с графикой необходимо ее инициализировать, а по окончании - "закрыть". Все графические процедуры и функции находятся в модуле Graph, поэтому также необходимо его подключение.

**Общая структура графической программы:**

Uses crt, graph;

var Gd, Gm: Integer;

begin

Gd: = Detect;

InitGraph (Gd, Gm, 'c: \bp\bgi');

...

{Здесь построение изображения}

...

ReadKey;

CloseGraph;

end.

Путь c: \bp\bgi указывает расположение файла egavga. bgi (драйвер графического адаптера). На разный компьютерах этот путь может быть разным. Если файл egavga. bgi поместить в каталог с программой, то путь можно не указывать.

**Основные графические процедуры и функции:**

**Построение фигур**

PutPixel (x,y,c) - отображает на экране точку с координатами (x,y) и цветом c

Line (x1,y1,x2,y2) - рисует линию с началом в точке (x1,y1) и концом - (x2,y2)

Rectangle (x1,y1,x2,y2) - рисует контур прямоугольника с диагональю (x1,y1) - (x2,y2)

Bar (x1,y1,x2,y2) - рисует закрашенный прямоугольник с диагональю (x1,y1) - (x2,y2)

Circle (x,y,r) - рисует окружность с центром (x,y) и радиусом r

Ellipse (x,y,ba,ea,xr,yr) - рисует дугу эллипса с центром в (x,y), горизонтальным и вертикальным радиусом xr и yr, и начальным и конечным углом ba и ea

FillEllipse (x,y,xr,yr) - рисует закрашенный эллипс с центром в (x,y), горизонтальным и вертикальным радиусом xr и yr

**Определение цветов и стилей**

GetPixel (x,y) - возвращает цвет пиксела с координатами (x,y)

SetColor (c) - устанавливает цвет изображения (для линий)

SetFillStyle (p,c) - устанавливает текущий стиль p и цвет c (для "закрашенных" фигур)

FloodFill (x,y,b) - закрашивает замкнутую область с внутренней точкой (x,y) и цветом контура b

**Работа с текстом**

OutTextXY (x,y,st) - выводит на экран строку st с позиции (x,y)

SetTextStyle (f,d,s) - устанавливает шрифт f, его направление d и размер

**Сообщения об ошибках**

**function GraphErrorMsg (ErrorCode: integer): String;** Возвращает строку сообщения об ошибке для заданного кода **ErrorCode.**

**function GraphResult: integer;** Возвращает код ошибки для последней графической операции.

**Определение адаптера, инициализация и переход между текстом и графикой**

**procedure DetectGraph (var GraphDriver, GraphMode: integer);** Проверяет аппаратуру и определяет какой графический драйвер и в каком режиме используется (тип адаптера - в **GraphDriver**, режим - **GraphMode**).

**function GetDriverName: string;** oзвращает строку с именем текущего драйвера.

**procedure InitGraph (var GraphDriver: integer; var GraphMode: integer; PathToDriver: String);** Инициализирует графическую систему и устанавливает устройство в графический режим. **PathToDriver** - полный путь к драйверу. bgi, обычно **'c: \tp\bgi'**. Если путь опущен (''), то драйвер должен находится в текущем каталоге.

**function RegisterBGIfont (Font: pointer): integer;** Регистрирует шрифт BGI для графической системы. Зарегистрированный фонт может использоваться в вызове **SetTextStyle**. Он может быть загружен с диска в кучу или преобразован в. obj файл (с помощью binobj. exe) и связан в файл. exe. В случае ошибки функция возвращает отрицательное значение.

**function RegisterBGIdriver (Driver: pointer): integer;** Регистрирует драйвер BGI для графической системы. Зарегистрированный драйвер будет использоваться процедурой **InitGraph**. Он может быть загружен с диска в кучу или преобразован в. obj файл (с помощью binobj. exe) и связан в файл. exe. В случае ошибки функция возвращает отрицательное значение.

**function InstallUserDriver (DriverFileName: string; AutoDetectPtr: pointer): integer;** Устанавливает пользовательский драйвер устройства в BGI таблицу драйверов устройств.

**function InstallUserFont (FontFileName: string): integer;** Устанавливает новый шрифт, который не встроен в BGI систему.

**procedure SetGraphBufSize (BufSize: word);** Позволяет изменить размер буфера для функций заполнения (закрасок). Буфер с размером BufSize байт размещается в куче при обращении к InitGraph. По умолчанию размер 4К, что достаточно для построения многоугольника с 650 линиями.

**function GetMaxMode: integer;** Возвращает номер максимального режима текущего загруженного драйвера.

**procedure GetModeRange (GraphDriver: integer; var LoMode, HiMode: integer);** Возвращает минимальный и максимальный графические режимы для данного драйвера.

**function GetModeName (GraphMode: integer): string;** Возвращает строку с именем указанного графического режима.

**procedure SetGraphMode (Mode: integer);** Переводит систему в указанный графический режим и очищает экран.

**function GetGraphMode: integer;** Возвращает текущий графический режим.

**procedure GraphDefaults;** Устанавливает текущий указатель (CP) в исходную позицию (т.е. точку 0,0) и сбрасывает все устанавливаемые пользователем режимы в исходное состояние, в т. ч. окна, палитру, основной цвет и цвет фона, стиль линий, шрифт, выравнивание текста, размер символов и стиль выводимого текста. Система должна находиться в графическом режиме.

**procedure RestoreCrtMode;** Восстанавливает текстовый видеорежим, который был до инициализации графики. Не освобождает буфер и не обнуляет установленные переменные.

**procedure CloseGraph;** Закрывает графическую систему. Восстанавливает текстовый видеорежим, который был до инициализации графики. Освобождает буфер и сбрасывает все графические переменные.

**Координаты графического курсора**

**function GetX: integer;** Возвращает координату X текущей позиции (CP).

**function GetY: integer;** Возвращает координату Y текущей позиции (СР).

**function GetMaxX: integer;** Возврашает максимальный Х (разрешение по горизонтали) для текущего графического драйвера и режима.

**function GetMaxY: integer;** Возвращает максимальный Y (разрешение по вертикали) для текущего графического драйвера и режима.

## Отличия графики Турбо-Паскаля и Турбо-Си

**Турбо-Паскаль** не различает отличий в регистрах (маленькие или большие буквы) при написании констант, типов, процедур и функций модуля **Graph**. В **Турбо-Си** все константы, типы, переменные должны быть в верхнем регистре (заглавными буквами), а названия процедур и функций - в нижнем (прописными буквами). Содержание библиотек графики при этом абсолютно одинаково.

**НЕДОСТАТКИ BGI-ГРАФИКИ**

Несмотря на обширные возможности для деловой графики, графические средства с точки зрения профессионального разработчика оставляют желать лучшего. Нельзя работать с произвольными спрайтами, нет скроллинга экрана, сохранения / вывода всего изображения, масштабирования картинки, функции заполнения и работы с битовыми образами работают медленно, невозможно использование файлов других графических форматов, отсутствуют трехмерные изображения и т.д. Все это делает затруднительным написание графических редакторов, игрушек, систем инженерной и математической графики. Для реализации вышеуказанных возможностей необходимо напрямую управлять графическим адаптером, используя прямой доступ в видеопамять и служебные функции **BIOS**.