# Работа с графикой в Turbo Pascal 7

## 1. Шкура графической программы

Шкура графической программы выглядит следующим образом:

```
Uses Graph;
var
grDriver:integer; {Драйвер}
grMode:integer; {Графический режим}
grPath:string; {Путь к драйверу }
ErrCode:integer; {Результат инициализации графического режима}

Begin
grDriver:=VGA;
grMode:= VGAHi;
grPath:='C:\TP7\BGI';
InitGraph(grDriver, grMode, grPath);
ErrCode:= GraphResult;
If ErrCode <> grOk then
Begin
   Writeln('Error # ', ErrCode); {выводим на экран ошибку}
   Readln; {программа ждёт нажатия <Enter>}
   Halt(1); {Halt(1) – функция закрывает программу}
End;

…… {место точек}
……..…… {здесь должен}
…… {быть код}

Readln; {программа ждёт нажатия клавиши «Enter»}
CloseGraph;
End.
```

**Теперь рассмотрим всё подробно.**
Сначала мы подключаем  модуль GRAPH

*Модуль Graph представляет собой мощную библиотеку графических подпрограмм универсального назначения. Он поддерживает графику CGA, EGA, VGA, Hercules, AT&T 400, MCGA, 3270PC и др.*

Объявляем модуль Граф.
**Uses Graph;**

После объявления модуля Graph нам необходимо объявить переменные, которые пригодятся при инициализации графического режима.

а именно:

**var**
**grDriver:integer;** *{Драйвер}*
**grMode:integer;** *{Графический режим}*
**grPath:string;** *{Путь к драйверу }*

**ErrCode:integer;** *{Результат инициализации графического режима}*

Далее Begin т.е. начало

**Begin**

А теперь поподробнее рассмотрим, что за переменные мы объявили:

В переменную ***grDriver***, указываем графический адаптер.

*VGA является стандартом, который используют большинство производителей видеоадаптеров. По мимо VGA еще существуют много графических адаптеров, например: CGA, MCGA, EGA, EGA64, PC3270 и др.*

Мы будем использовать VGA поэтому в переменную «***grDriver»*** запишем константу VGA или значение этой константы – 9

*Примечание:*
*Для того чтобы Pascal автоматически выбрал драйвер графического устройство нужно вместо VGA написать «**Detect»** т.е. «**grDriver:=Detect;»**. Detect - это константа модуля Graph, равная 0, задающая автоматический выбор драйвера графического устройства .*

**grDriver:=VGA;** *{тут мы могли записать так: grDriver:=Detect; }*

Далее мы видим переменную ***grMode.*** В ней мы указываем графический режим (под словом «режим» – подразумевается разрешение экрана).

Рассмотрим следующие режимы VGA:

| Константа | Значение | Режим |
|---|---|---|
| VGALo | 0 | 640x200 |
| VGAMed | 1 | 640x350 |
| VGAHi | 2 | 640x480 |

Мы будем использовать режим 640x480 поэтому в переменную ***grMode*** указываем константу ***VGAHi,*** или значение этой константы – 2

*Примечание:*
*Если у нас **«grDriver:=Detect;»** то строчку « **grMode:= VGAHi;»** можно не писать т.к.*

*если **«grDriver:=Detect;»** то графический режим тоже определяется автоматически.*

**grMode:= VGAHi;**

В переменную ***grPath*** указываем папку в которой находится наш драйвер т.к. мы используем адаптер VGA, нам нужен драйвер «EGAVGA.BGI», который находится в папке «TP7\BGI», поэтому в переменную **grPath** указываем путь к этому файлу, если TP7 у вас установлен в диск «C» то тогда пишем:

**grPath:='C:\TP7\BGI';** *{путь к драйверу}*

Всё, теперь у нас всё готово для того чтобы инициализировать графический режим.

**InitGraph(grDriver, grMode, grPath);**

Функция ***InitGraph*** как вы уже догадались, инициализирует граф режим.

Переменную ***ErrCode*** сделаем значение равное ***GraphResult*** *m.e.:*

**ErrCode:= GraphResult;**

***GraphResult*** - это функция модуля Graph, которая возвращает результат инициализации графического режима. Константа ***grOK=0.*** Если ***«ErrCode=grOK»*** т.е. 0 то графический режим успешно инициализирован. В случае неуспешной загрузки графического режима в переменную ***ErrCode*** записывается номер ошибки.

| Номер ошибки | Описание |
|---|---|
| 2 | нет графического адаптера |
| 3 | нет драйвера устройства |
| 4 | ошибка в драйвере |
| 5 | нет памяти для загрузки драйвера |
| 10 | недопустимый режим для выбранного драйвера |

Зададим условие:
Если графический режим не запустился то программа об этом сообщает, выводит номер ошибки и закрывается после нажатия клавиши «Enter»

```
If ErrCode <> grOk then  {Если ErrCode не равно 0 то выполняем:}
Begin
   Writeln('Error # ', ErrCode); {выводим на экран ошибку}
   Readln; {программа ждёт нажатия <Enter>}
   Halt(1); {Halt(1) – функция закрывает программу}
End;
```

Если ErrCode не равно 0 то программа продолжает выполнять свои действия дальше.

Сразу после условия которое проверяет успешно ли загрузился графический режим, обычно идёт программный код и процедуры которые что то рисуют.

…… *{место точек}*
…….…… *{здесь должен}*
…… *{быть наш код}*

**Readln;** *{программа ждёт нажатия клавиши «Enter»}*
**CloseGraph;**
***{CloseGraph** – процедура встроенная в модуль Graph, она закрывает графический режим}*
**End.**

## 2. Рисование линий

| Процедура | Описание |
|---|---|
| Line(X1, Y1, X2, Y2);<br>‘(X1,Y1) и (X2, Y2) – целые числа | Рисует линию от точки X1, Y1 до точки X2,Y2. |

Вот код программы которая вырисовывает линию:

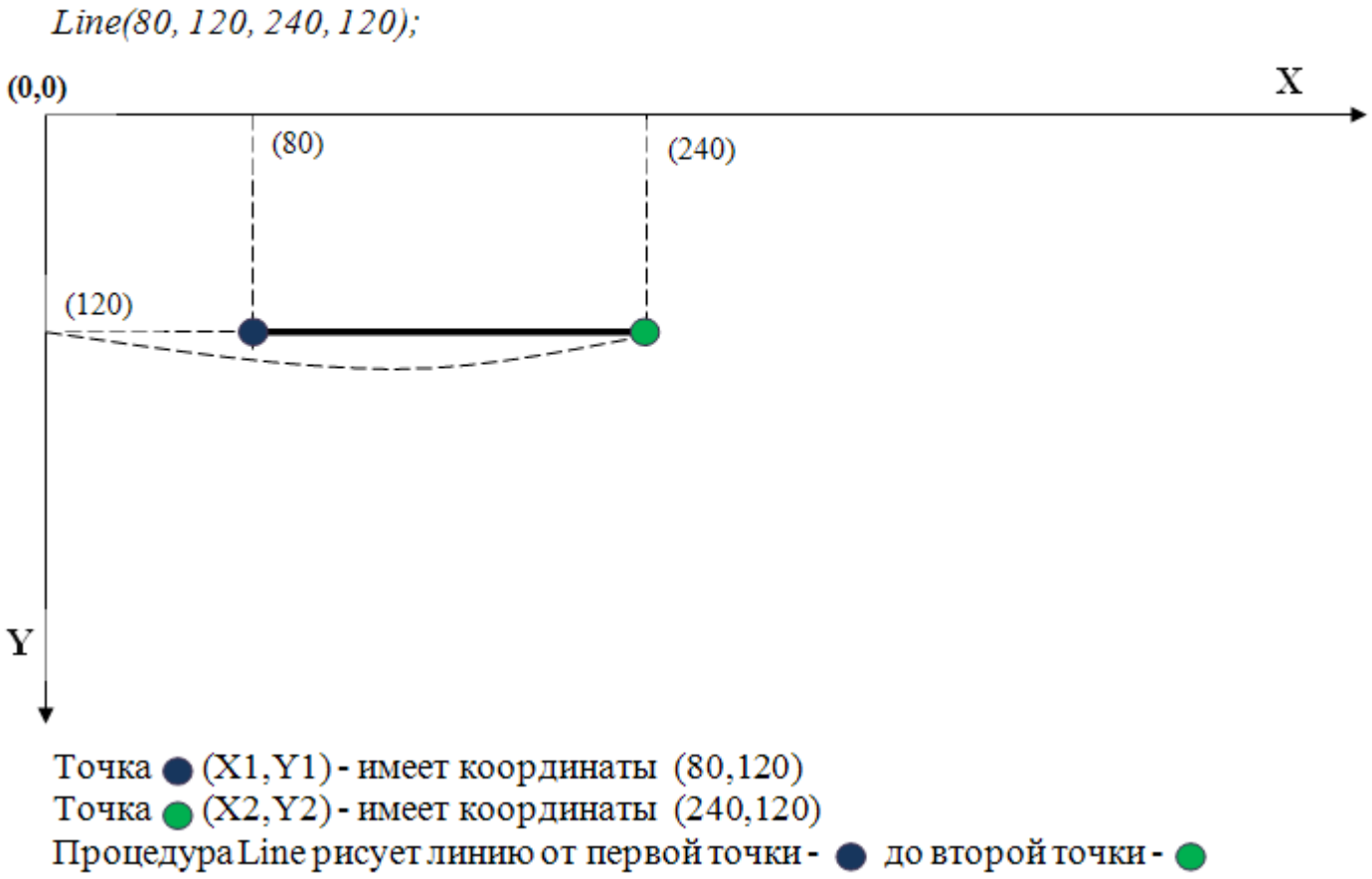
```
Uses Graph;
var
grDriver:integer; {Драйвер}
grMode:integer; {Графический режим}
grPath:string; {Путь к драйверу }
ErrCode:integer; {Результат инициализации  графического режима}

Begin
grDriver:=VGA;
grMode:= VGAHi;
grPath:='C:\TP7\BGI';
InitGraph(grDriver, grMode, grPath);
ErrCode:= GraphResult;
If ErrCode <> grOk then
Begin
   Writeln('Error # ', ErrCode); {выводим на экран ошибку}
   Readln; {программа ждёт нажатия <Enter>}
   Halt(1); {Halt(1) – функция закрывает программу}
End;

LINE(80, 120, 240, 120);

Readln; {программа ждёт нажатия клавиши «Enter»}
CloseGraph;
End.
```

Шкура программы нам уже известна, Line – это процедура рисует линию. 80, 120, 240, 120 -- это координаты, 80, 120 – это начальный X и Y (X1 и Y1), а 240, 120 – это конечный X и Y (X2 и Y2).
Обрати внимание на рисунок ниже и всё поймешь.




| Процедура | Описание |
|---|---|
| MoveTo (X,Y) | Перемещает курсор в точку с координатами X и Y |

| Процедура | Описание |
|---|---|
| LineTo(X,Y) | Рисует линию от текущего положения курсора до указанной точки и при этом курсор перемещается к точке (указанному положению). |

Команда:
LineTo(120,120);

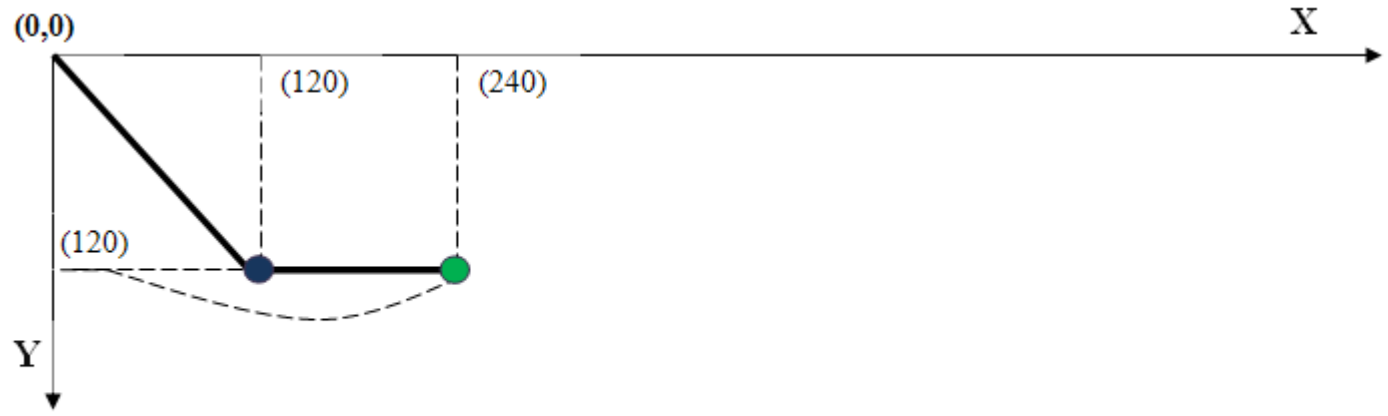
*Аналогична команде:*
*Line(0, 0 120,120); если MoveTo(0,0):*

Т.е. команды
**MoveTo(0,0);**
**LineTo(120,120);**

Выведут на экран следующую линию:


Точка (курсор) ● - имеет координаты (0,0) - **MoveTo(0,0);**
Точка ● - имеет координаты (120,120) - **LineTo(120,120);**

Сначала у нас курсор находится в точке **(0,0),** после вызова **LineTo(120,120);** вырисовывается линия от точки **(0,0)** до точки **(120,120)** и курсор (точка) ● перемещается в точку **(120,120),** а после того как мы вызываем например **LineTo(240, 120);** то вырисовывается линия от точки(текущего положения курсора) **(120,120)** до точки **(240,120)** и соответственно курсор перемещается в точку **(240,120)**
Например, если после команд **MoveTo(0,0); LineTo(120,120);** сразу вызвать команду **LineTo(240, 120);** то нарисуется еще одна линия:




Точка ● - имеет координаты (120,120)
Точка ● - имеет координаты (240,120) - **LineTo(240, 120);**

*Если мы хотим нарисовать линию от точки до точки необязательно использовать процедуру Line, достаточно просто записать так:*

**MoveTo (80, 120);** *{Устанавливаем курсор в точку (80,120)}*
**LineTo(240, 120);** *{Рисуем линию начиная от положения курсора заканчивая координатами (240,120) }*

Команды:
**MoveTo (80, 120); LineTo(240, 120);**
Аналогичны команде:
Line(80,120,240,120)