**Graphics mode Initialization #**

Graphics options are used in c-programming to draw different graphical shapes. First of all we have to call the **initgraph()** function that will initialize the graphics mode on the computer.

Call to function initgraph() is done as

**initgraph**(&gdriver, &gmode, “path\_to\_driver”);

initgraph() initializes the graphics system by loading a graphics driver from disk (or validating a registered driver) then putting the system into graphics mode. initgraph() also resets all graphics settings (color, palette, current position, viewport, etc.) to their defaults, then resets graph result to 0.

**gdriver:**

It is declared as integer variable that specifies the graphics driver to be used.

**gmode:**

It is also declared as integer variable which specifies the initial graphics mode (unless gdriver = DETECT). If gdriver = DETECT, initgraph sets gmode to the highest resolution available for the detected driver.

**path\_to\_driver:**

Specifies the directory path where initgraph() looks for graphics drivers (e.g. egavga.bgi in turbo c++)

1. If the driver is not there, initgraph() looks in the current directory.

2. If path\_to\_driver is null, the driver files must be in the current directory.

After a call to initgraph, gdriver is set to the current graphics driver, and gmode is set to the current graphics mode.

Gdriver =DETECT auto-detects the attached video adapter at run time and pick the corresponding driver. If we tell initgraph to auto-detect, it calls detectgraph to select a graphics driver and mode.

Normally, initgraph() loads a graphics driver by allocating memory for the driver, then loading the appropriate .BGI file from disk

Graphics functions in C #

There are so many built in graphics functions defined in C library. The basic graphics functions are described as below. Graphics functions are text mode graphics functions as well as graphical mode functions

**Text mode graphics functions:** The text mode graphics functions place the text in certain area of the screen. Those functions are included in the header file <conio.h>. Some of those text mode graphics functions are:

window(): It sets particular area on the output screen and displays the output text on that area.

The syntax of calling this window function is

**window(10,10, 80,25);**

where co-ordinate (10,10) represents the upper left corner of displaying window and (80,25) is the lower right corner of the display window.

**clrscr():** It is used to clear the screen and locates the curser at the beginning of the screen.

**gotoxy(x,y):** it moves the curser at specified co-ordinates (x,y) position.

**cputs(string):**it writes a string given to function to the user defined window screen.

**putch(char) :** it writes a character given to function to the user defined area of window.

**Graphical mode functions:**

There are so many graphical functions in C graphics library. All the graphics mode functions are in <graphics.h> file. So we must include the graphics.h file to use those functions. To run a c program in graphical mode the most important functions are as below.

**initgraph()**: it is one of the function that is used to initialize the computer in graphics mode. Its syntax is as

**initgraph(&gdriver, &gmode, "graphics driver path");**

**closegraph():** It is the graphical function that is used to close the graphics mode initialized by the initgraph() function.

**graphresult():** it is a graphical function that returns the value 0 if the graphics is detected correctly and the driver is correctly initialized to correct graphics mode other wise it returns some error code than 0.

**grapherrormsg(errorcode):** This function returns the message string corresponding to the errorcode returned by the graphresult() function.

**cleardevice():** This function is used to clear the screen in graphical mode as clrscr() in text mode.

After correctly initialized in graphics mode, we can use the different graphical functions

available in the c- library in graphics.h. Following are some basic graphical functions for

drawing geometrical objects.

**setcolor(color):** It is a function that is used to set the color of the drawing object with given color. The color is indicated by the integer from 0 to 15 ( 16 color)

**moveto(x, y):** it is used to move the cursor in display screen at specified co-ordinates by the value (x,y).

**outtext(" text string "):** prints the text string in the current position of screen.

**outtextxy(x, y, "string"):** prints the string from the co-ordinate (x,y) position.

**lineto(x, y):** draws the line from current position to (x,y) position.

**putpixel(x, y, color):**displays a pixel(point) at (x,y) with given color.

**line(a, b, c, d):** draws line from (a,b) to (c,d).

**circle(x, y, r):** prints circle with center (x,y) and radius r.

**rectangle(a, b, c, d):** prints rectangle where (a,b) is upper left coordinate and (c,d) is lower right co-ordinates of rectangle.

program to draw a line #

*/\*program to draw a line\*/*

#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

int **main**()

{

*/\* request auto detection \*/*

    int gdriver = DETECT, gmode, errorcode;

*/\* initialize graphics mode \*/*

**initgraph**(&gdriver, &gmode, "");

*/\* read result of initialization \*/*

    errorcode = **graphresult**();

    if (errorcode != grOk) */\* an error occurred \*/*

    {

**printf**("Graphics error: %s\n", **grapherrormsg**(errorcode));

**printf**("Press any key to halt:");

**getch**();

**exit**(1); */\* return with error code \*/*

    }

*/\* draw a line \*/*

**line**(0, 0, **getmaxx**(), **getmaxy**());

*/\* clean up \*/*

**getch**();

**closegraph**();

    return 0;

}

Program to draw a circle #

*/\*Program to draw a circle\*/*

#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

int **main**()

{

*/\* request auto detection \*/*

    Int x,y;

    int gdriver = DETECT, gmode, errorcode;

*/\* initialize graphics mode \*/*

**initgraph**(&gdriver, &gmode, "");

*/\* read result of initialization \*/*

    errorcode = **graphresult**();

    if (errorcode != grOk) */\* an error occurred \*/*

    {

**printf**("Graphics error: %s\n", **grapherrormsg**(errorcode));

**printf**("Press any key to halt:");

**getch**();

**exit**(1); */\* return with error code \*/*

    }

*/\* draw a line \*/*

**circle**(x,y,100);

*/\* clean up \*/*

**getch**();

**closegraph**();

    return 0;

}

A program to draw concentric circles #

#include <graphics.h>

int **main**()

{

    int gd = DETECT, gm;

    int x = 320, y = 240, radius;

**initgraph**(&gd, &gm, "");

    for ( radius = 25; radius <= 125 ; radius = radius + 20)

**circle**(x, y, radius);

**getch**();

**closegraph**();

    return 0;

}