Program 2:

#include<stdio.h>

#include<graphics.h>

void drawline(int x0, int y0, int x1, int y1)

{

int dx, dy, p, x, y;

dx=x1-x0;

dy=y1-y0;

x=x0;

y=y0;

p=2\*dy-dx;

while(x<x1)

{

if(p>=0)

{

putpixel(x,y,7);

y=y+1;

p=p+2\*dy-2\*dx;

}

else

{

putpixel(x,y,7);

p=p+2\*dy;}

x=x+1;

}

}

int main()

{

int gdriver=DETECT, gmode, error, x0, y0, x1, y1;

initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");

printf("Enter co-ordinates of first point: ");

scanf("%d%d", &x0, &y0);

printf("Enter co-ordinates of second point: ");

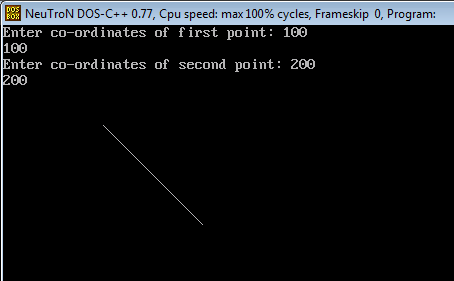
scanf("%d%d", &x1, &y1);

drawline(x0, y0, x1, y1);

return 0;

}

Output:



Program 3:

// C Implementation for Boundary Filling Algorithm

#include <graphics.h>

// Function for 4 connected Pixels

void boundaryFill4(int x, int y, int fill\_color,int boundary\_color)

{

if(getpixel(x, y) != boundary\_color &&

getpixel(x, y) != fill\_color)

{

putpixel(x, y, fill\_color);

boundaryFill4(x + 1, y, fill\_color, boundary\_color);

boundaryFill4(x, y + 1, fill\_color, boundary\_color);

boundaryFill4(x - 1, y, fill\_color, boundary\_color);

boundaryFill4(x, y - 1, fill\_color, boundary\_color);

}

}

//driver code

int main()

{

// gm is Graphics mode which is

// a computer display mode that

// generates image using pixels.

// DETECT is a macro defined in

// "graphics.h" header file

int gd = DETECT, gm;

// initgraph initializes the

// graphics system by loading a

// graphics driver from disk

initgraph(&gd, &gm, "");

int x = 250, y = 200, radius = 50;

// circle function

circle(x, y, radius);

// Function calling

boundaryFill4(x, y, 6, 15);

delay(10000);

getch();

// closegraph function closes the

// graphics mode and deallocates

// all memory allocated by

// graphics system .

closegraph();

return 0;

}

Output:

