PART- B

# Program to implement DDA Line generation algorithm for all 4 quadrants of a plane (x and y axis should meet at centre(320,240) i.e., Origin O(0,0) of user coordinate system).

#include<iostream.h>

#include<math.h>

#include<graphics.h>

#include<dos.h>

#include<conio.h>

void main()

{

int gd,gm;

detectgraph(&gd,&gm);

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

int x1,y1,x2,y2,clr,dx,dy;

float xinc,yinc,steps,k,x,y;

cleardevice();

cout<<"\nEnter the starting point of the line: ";

cin>>x1>>y1;

cout<<"Enter the ending point of the line: ";

cin>>x2>>y2;

if(x1==x2&&y1==y2)

{

cout<<"Both end points are same, line cannot be drawn.";

}

else

{

cout<<"Enter color: ";

cin>>clr;

dx=x2-x1;

dy=y2-y1;

if(abs(dx)>abs(dy))

steps=abs(dx);

else

steps=abs(dy);

xinc=dx/steps;

yinc=dy/steps;

x=x1;

y=y1;

line(0,240,640,240);

line(320,0,320,480);

for(k=0;k<steps;k++)

{

x+=xinc;

y+=yinc;

putpixel(320+float(x+0.5),240-float(y+0.5),clr);

delay(1);

}

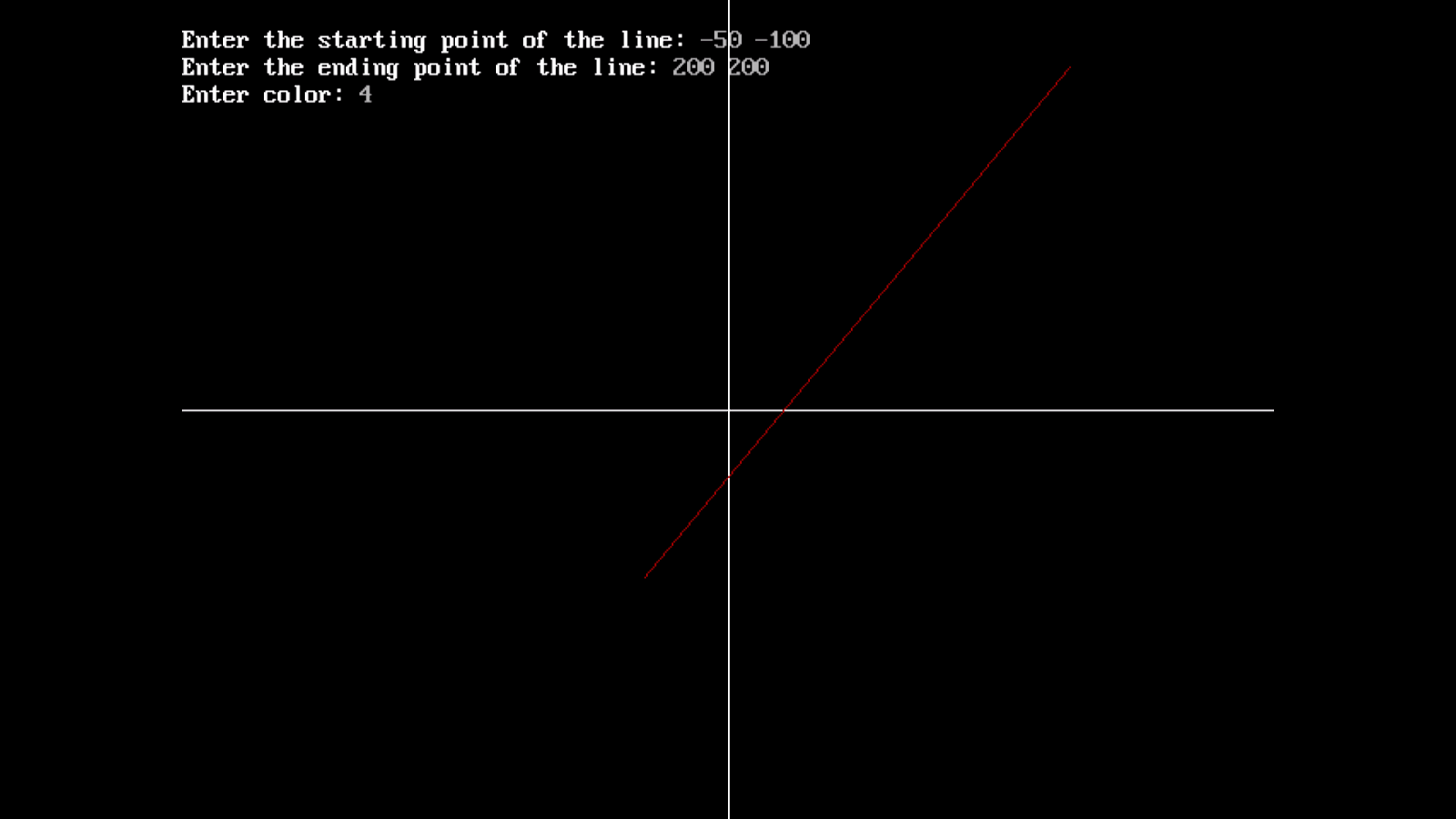
}

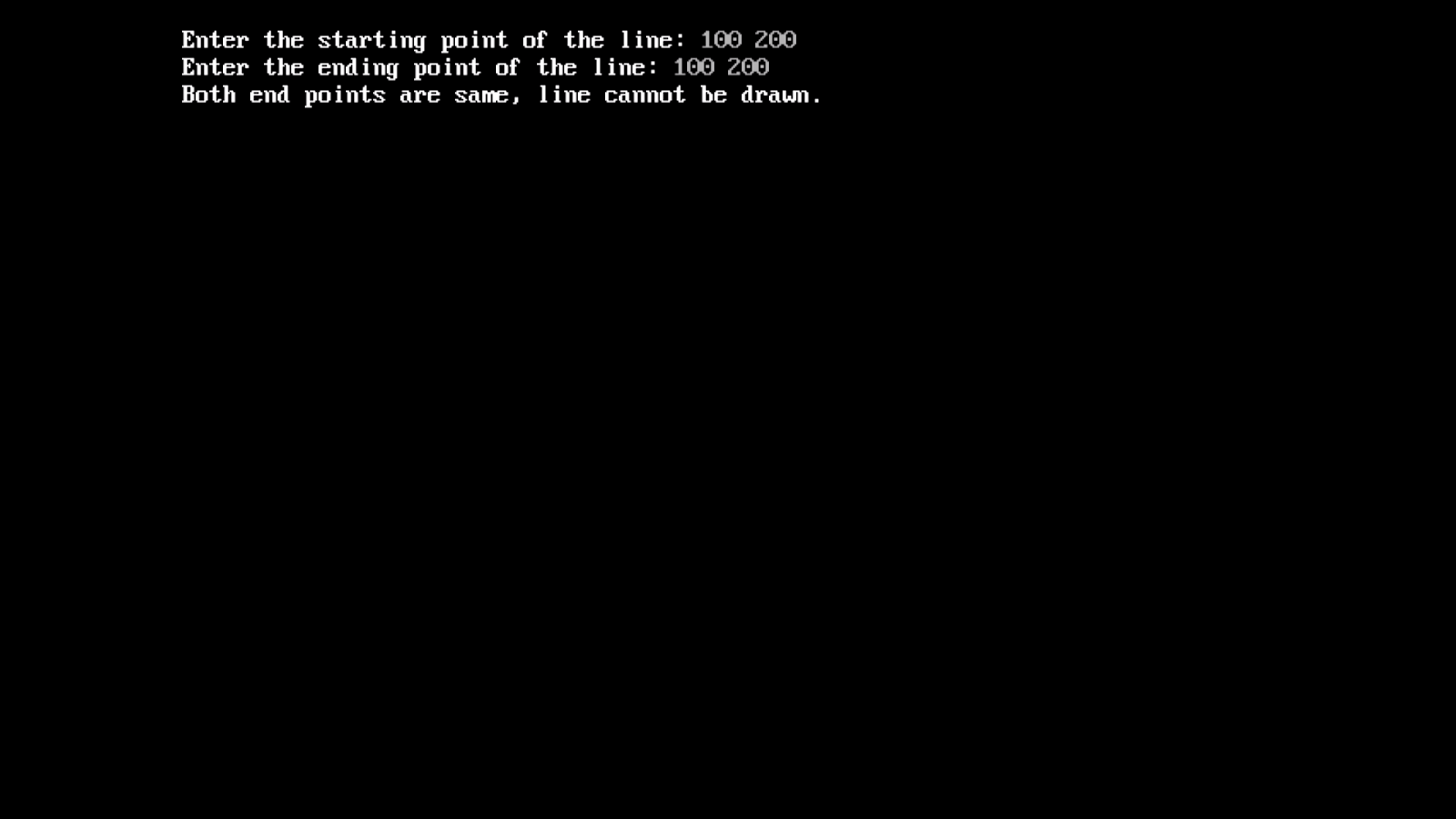
getch();

closegraph();

}

**Output:**





# Program to draw a line using Bresenham’s line generation algorithm with slope m such that 0<m

#include<graphics.h>

#include<iostream.h>

#include<conio.h>

#include<process.h>

#include<math.h>

void main()

{

int gd,gm;

int x0,y0, clr, x, y, x1, y1, p, dx,dy,xend, yend;

float m;

detectgraph(&gd,&gm);

initgraph(&gd, &gm, "c:\\turboc3\\bgi");

cleardevice();

cout<<"Enter the starting point: ";

cin>>x0>>y0;

cout<<"Enter the ending point : ";

cin>>x1>>y1;

cout<<"Enter the colour of the line :";

cin>>clr;

if(x0==x1 && y0==y1)

{

cout<<"Both end points are same line cannot be drawn..";

getch();

exit(0);

}

dx=abs(x1-x0);

dy=abs(y1-y0);

line(0,240,640,240);

line(320,0,320,480);

if(m>=0&&m<=1)

{

p=2\*dy-dx;

if(x0>x1)

{

x=x1;

y=y1;

xend=x0;

}

else

{

x=x0;

y=y0;

xend=x1;

}

putpixel(320+x,240-y,clr);

while(x<xend)

{

x++;

if(p<0)

{

putpixel(320+x,240-y,clr);

p=p+2\*dy;

}

else

{

y++;

putpixel(320+x,240-y,clr);

p=p+2\*dy-dx;

}

}

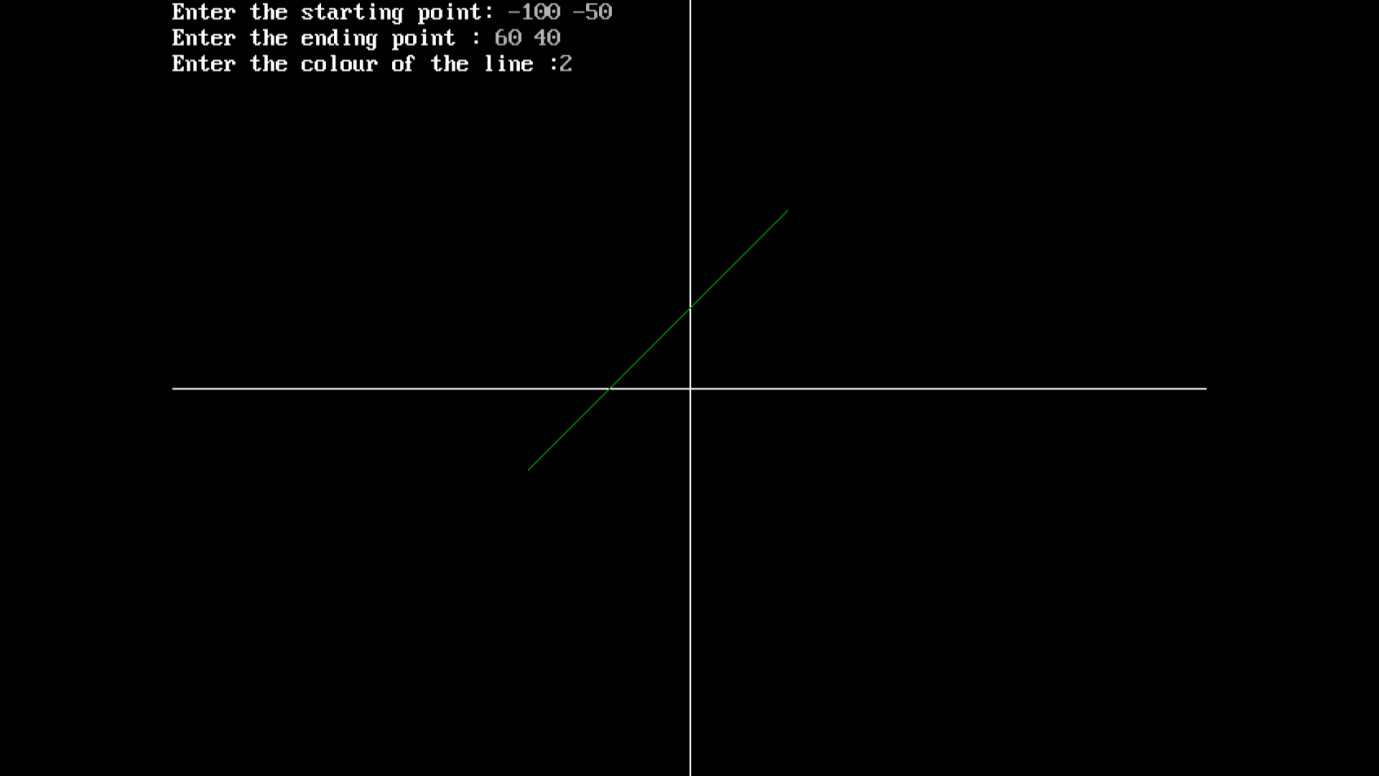
}

getch();

closegraph();

}

Output:





# 3. Program to implement midpoint circle generation algorithm.

#include<graphics.h>

#include<iostream.h>

#include<conio.h>

void circlepts(int mc,int nc,int m,int n,int clr)

{

putpixel(mc+m,nc+n,clr);

putpixel(mc-m,nc+n,clr);

putpixel(mc+m,nc-n,clr);

putpixel(mc-m,nc-n,clr);

putpixel(mc+n,nc+m,clr);

putpixel(mc-n,nc+m,clr);

putpixel(mc+n,nc-m,clr);

putpixel(mc-n,nc-m,clr);

}

void main()

{

int gd=DETECT,gm;

int x,y,xc,yc,r,clr,p;

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

cleardevice();

cout<<"Enter the center of circle:";

cin>>xc>>yc;

cout<<"Enter Radius of circle:";

cin>>r;

cout<<"Enter color:";

cin>>clr;

x=0;

y=r;

p=1-r;

do

{

circlepts(xc,yc,x,y,clr);

x++;

if(p<0) p=p+2\*x+1;

else

{

y--;

p=p+2\*(x-y)+1;

}

circlepts(xc,yc,x,y,clr);

}

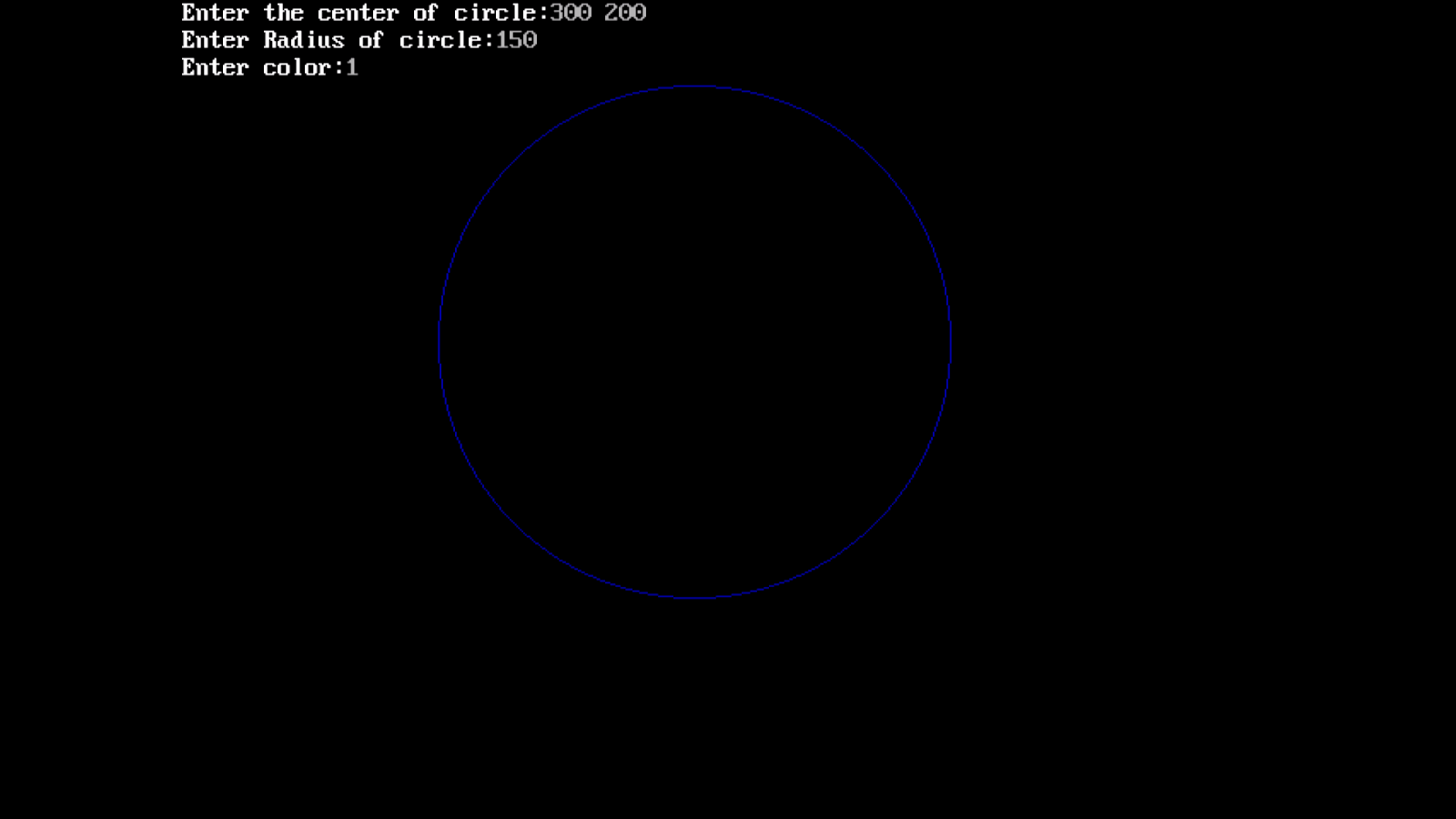
while(x<y);

getch();

closegraph();

}

**Output:**



# Program to transform object ( Tringle or rectangle or Polygon) from window to view port.

//Program to transform an object from window to view port

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<iostream.h>

//Function for window to viewport transformation

void windowToViewport(int winx[], int winy[], int n, int xwmin, int ywmin, int xwmax, int ywmax, int xvmin, int yvmin, int xvmax, int yvmax)

{

//Point on viewport

int i, viewx[10], viewy[10];

//Scaling factor for x cordinate and y cordinate

float sx, sy;

//Calculating sx and sy

sx=(float)(xvmax - xvmin) / (xwmax - xwmin);

sy=(float)(yvmax - yvmin) / (ywmax - ywmin);

//Calcualting the points of an object on viewport

for (i=0;i<n;i++)

{

viewx[i]= xvmin + (float) ((winx[i] - xwmin) \*sx);

viewy[i]= yvmin + (float) ((winy[i] - ywmin) \*sy);

}

//Draw viewport and plot an object inside viewport

setcolor(BLUE);

rectangle(xvmin, yvmin, xvmax, yvmax);

for(i=0;i<n-1;i++)

line(viewx[i], viewy[i], viewx[i+1], viewy[i+1]);

line(viewx[i], viewy[i], viewx[0], viewy[0]);

}

void main()

{

int gd= DETECT, gm;

//Boundary values for windows

int xwmax, ywmax, xwmin, ywmin;

//Boundary values for viewport

int xvmax, yvmax, xvmin, yvmin;

//Object inside the window

int winx[10],winy[10], n, i;

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

cout<<"Enter the window cordinates(xmin,ymin,xmax,ymax): \n";

cin>>xwmin>>ywmin>>xwmax>>ywmax;

cout<<"\nEnter the viewport cordinates(xmin,ymin,xmax,ymax):\n ";

cin>>xvmin>>yvmin>>xvmax>>yvmax;

cout<<"Enter the number of vertices: ";

cin>>n;

cout<<"\nEnter values of vertices: \n";

for(i=0;i<n;i++)

cin>>winx[i]>>winy[i];

cleardevice();

//Draw window and plot an object

setcolor(RED);

rectangle(xwmin, ywmin, xwmax,ywmax);

for (i=0;i<n-1;i++)

line(winx[i],winy[i],winx[i+1], winy[i+1]);

line(winx[i],winy[i],winx[0], winy[0]);

getch();

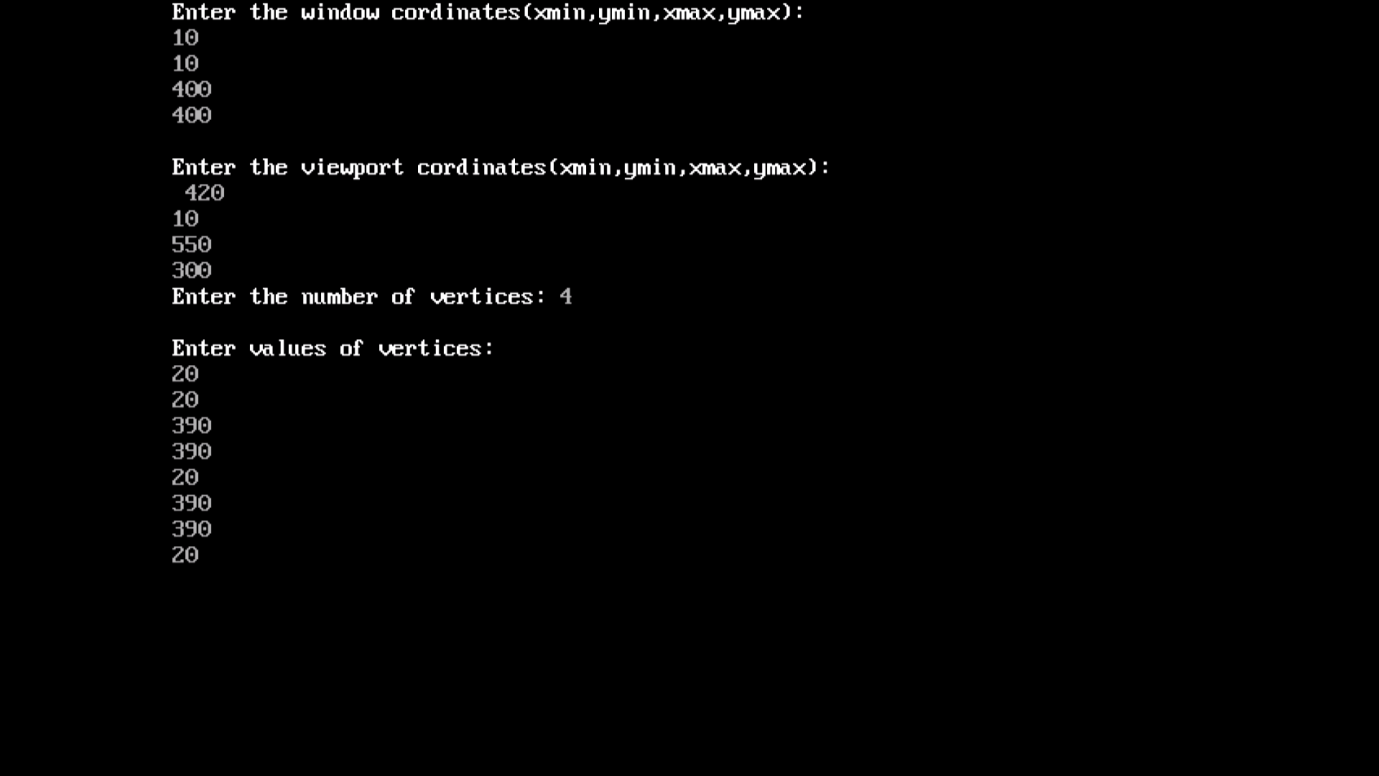
//call the method to transform the object from window into viewport

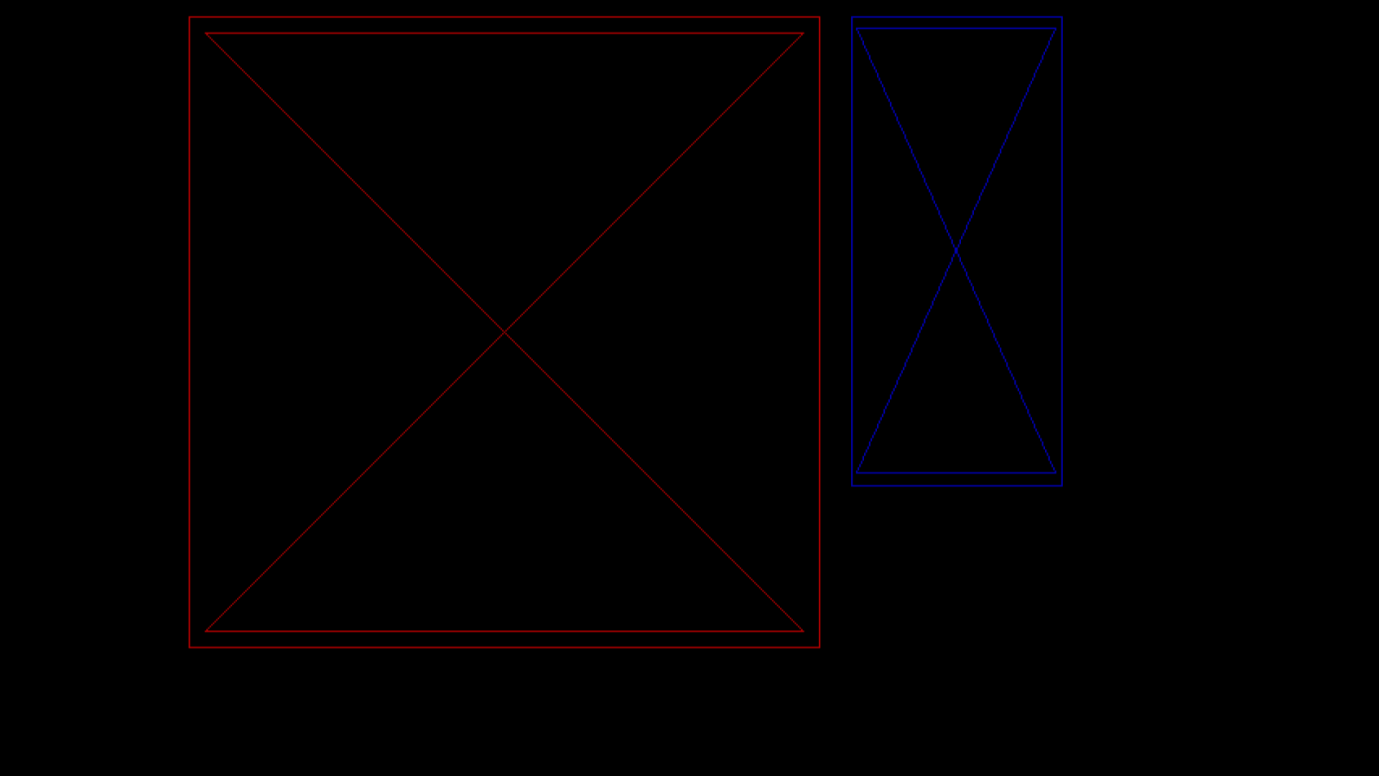
windowToViewport(winx, winy, n, xwmin,ywmin, xwmax,ywmax, xvmin,yvmin, xvmax,yvmax);

getch();

closegraph();

}





# Program to draw a thick circle using moving pen method at a given centre.

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<stdio.h>

void pen(int m,int n, int c)

{

putpixel(m,n,c);

putpixel(m+1,n,c);

putpixel(m-1,n,c);

putpixel(m,n+1,c);

putpixel(m+1,n+1,c);

putpixel(m+1,n-1,c);

putpixel(m-1,n+1,c);

putpixel(m-1,n-1,c);

}

void main()

{

int gd=DETECT, gm;

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

cleardevice();

int x,y, r, xc, yc, clr, p;

cout<<"Enter the center of the circle: ";

cin>>xc>>yc;

cout<<"Enter the radius: ";

cin>>r;

cout<<"Enter the color of the circle: ";

cin>>clr;

p=1-r,x=0,y=r;

do{

pen(xc+x,yc+y,clr);

pen(xc+x,yc-y,clr);

pen(xc-x,yc+y,clr);

pen(xc-x,yc-y,clr);

pen(xc+y,yc+x,clr);

pen(xc+y,yc-x,clr);

pen(xc-y,yc+x,clr);

pen(xc-y,yc-x,clr);

x++;

if(p<0)

p=p+2\*x+1;

else

{

y--;

p=p+2\*x+1-2\*y;

}

}

while(x<y);

getch();

closegraph();

}

Output:

