**EXPERIMENT NO.1(a)**

**AIM-**To draw a line using Simple DDA Algorithmfor positive line slope.

**Source Code:**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

void main()

{

float x1,x2,y1,y2,xinc,yinc,l,dx,dy,x,y;

float temp;

int gd=DETECT,gm;

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

cout<<"Enter the coordinates of first point"<<endl;

cin>>x1>>y1;

cout<<"Enter the coordinates of second point"<<endl;

cin>>x2>>y2;

dx=x2-x1;

dy=y2-y1;

if(dx>=dy)

l=dx;

else

l=dy;

xinc=dx/l;

yinc=dy/l;

x=abs(x1+0.5);

y=abs(y1+0.5);

for(int i=1;i<=l;i++)

{

putpixel(x,y,WHITE);

x=x+xinc;

y=y+yinc;

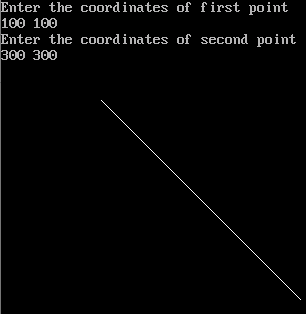
}

getch();

closegraph();

}

**OUTPUT:-**



**EXPERIMENT NO.2(a)**

**AIM-**To draw a line using Simple DDA Algorithmfor negative line slope.

**Source Code:**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

void main()

{

float x1,x2,y1,y2,xinc,yinc,l,dx,dy,x,y;

float temp;

int gd=DETECT,gm;

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

cout<<"Enter the coordinates of first point"<<endl;

cin>>x1>>y1;

cout<<"Enter the coordinates of second point"<<endl;

cin>>x2>>y2;

dx=x2-x1;

dy=y2-y1;

if(dx>=dy)

l=dx;

else

l=dy;

xinc=dx/l;

yinc=dy/l;

x=abs(x1+0.5);

y=abs(y1+0.5);

for(int i=1;i<=l;i++)

{

putpixel(x,y,WHITE);

x=x+xinc;

y=y+yinc;

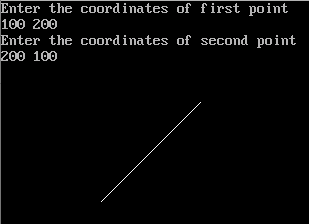
}

getch();

closegraph();

}

**OUTPUT:-**



**EXPERIMENT NO. 1(b)**

**AIM-**To draw a line using Symmetrical DDAAlgorithm for positive line slope.

**Source Code:**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

void main()

{

intgd=DETECT,gm;

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

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

floatx,y,xInc,yInc,length;

cout<<"Ente the co-odinates of first point"<<endl;

cin>>x1>>y1;

cout<<"Ente the co-odinates of second point"<<endl;

cin>>x2>>y2;

dx=x2-x1;

dy=y2-y1;

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

length=abs(dx);

else

length=abs(dy);

float n=log10(length)/log10(2);

cout<<"value of n="<<n;

xInc=dx/(pow(2,n));

yInc=dy/(pow(2,n));

x=x1+0.5;

y=y1+0.5;

putpixel(x,y,WHITE);

for(i=1;i<=length;i++)

{

x=x+xInc;

y=y+yInc;

putpixel(x,y,WHITE);

delay(100);

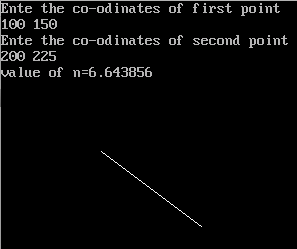
}

getch();

closegraph();

}

**OUTPUT:-**



**EXPERIMENT NO. 1(c)**

**AIM-**To draw a line using Bresenham’s Algorithmfor positive line slope.

**Source Code:**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void main()

{

intgd=DETECT,gm;

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

int x1,x2,y1,y2,dx,dy,x,y,p,ymax,xmax;

cout<<"Ente the co-odinates of first point"<<endl;

cin>>x1>>y1;

cout<<"Ente the co-odinates of second point"<<endl;

cin>>x2>>y2;

dx=x2-x1;

dy=y2-y1;

if(x1>x2)

{

x=x2;

y=y2;

}

else

{

x=x1;

y=y1;

}

p=2\*dy-dx;

ymax=(y1>=y2)?y1:y2;

xmax=(x1>=x2)?x1:x2;

// while(x<=x2 && y<=y2)

while(x<=ymax || y<=xmax)

{

if(p<0)

{

x=x+1;

p=p+2\*dy;

}

else if(p>=0)

{

x=x+1;

y=y+1;

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

}

putpixel(x,y,WHITE);

delay(50);

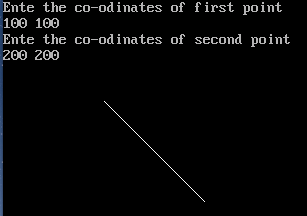
}

getch();

closegraph();

}

**OUTPUT:-**



**EXPERIMENT NO. 2(b)**

**AIM-**To draw a line using Symmetrical DDAAlgorithm for negative line slope.

**Source Code:**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

void main()

{

intgd=DETECT,gm;

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

int x1,x2,y1,y2,dx,dy,x,y,s1,s2,flag,p,c,a,b;

cout<<"Ente the co-odinates of first point"<<endl;

cin>>x1>>y1;

cout<<"Ente the co-odinates of second point"<<endl;

cin>>x2>>y2;

dx=x2-x1;

dy=y2-y1;

x=x1;

y=y1;

s1=(dx==0)?0:((dx>0)?1:-1);

s2=(dy==0)?0:((dy>0)?1:-1);

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

{

flag=1;

c=abs(dx);

dx=abs(dy);

dy=abs(c);

}

else

{ dx=abs(dx);

dy=abs(dy);

flag=0;

}

a=2\*dx;

b=2\*dy;

p=b-dx;

putpixel(x,y,white);

while(x!=x2&&y!=y2)

{

while(p>=0)

{

if(flag==1)

x=x+s1;

else

y=y+s2;

p=p-a;

putpixel(x,y,WHITE);

}

if(flag==1)

y=y+s2;

else

x=x+s1;

p=p+b;

putpixel(x,y,WHITE);

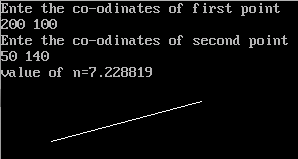
}

getch();

closegraph();

}

**OUTPUT:-**



**EXPERIMENT NO. 2(c)**

**AIM-**To draw a line using Bresenham’s Algorithmfor negative line slope.

**Source Code:**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

void main()

{

intgd=DETECT,gm;

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

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

floatx,y,xInc,yInc,length;

cout<<"Ente the co-odinates of first point"<<endl;

cin>>x1>>y1;

cout<<"Ente the co-odinates of second point"<<endl;

cin>>x2>>y2;

dx=x2-x1;

dy=y2-y1;

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

length=abs(dx);

else

length=abs(dy);

float n=log10(length)/log10(2);

cout<<"value of n="<<n;

xInc=dx/(pow(2,n));

yInc=dy/(pow(2,n));

x=x1+0.5;

y=y1+0.5;

putpixel(x,y,WHITE);

for(i=1;i<=length;i++)

{

x=x+xInc;

y=y+yInc;

putpixel(x,y,WHITE);

delay(100);

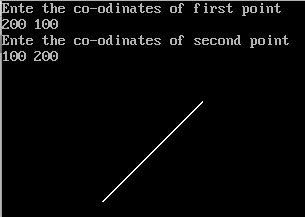
}

getch();

closegraph();

}

**OUTPUT:-**



**PRACTICAL NO 3**

**Aim** To draw a line, triangle and circle using functions of graphics.h header file

#include<graphics.h>

#include<conio.h>

void main()

{

int gdriver=DETECT,gmode;

initgraph(&gdriver,&gmode,"C://TC//BGI");

setbkcolor(CYAN);

setcolor(RED);

line(20,20,50,50);

circle(80,30,20);

rectangle(20,70,70,100);

arc(100,110,50,110,30);

ellipse(150,50,70,430,25,30);

setcolor(WHITE);

outtextxy(10,2 ,"Line:");

outtextxy(60,2 ,"Circle:");

outtextxy(120,5 ,"Ellipse:");

outtextxy(10,60 ,"Rectangle:");

outtextxy(95,70 ,"Arc:");

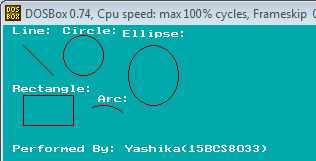
outtextxy(10,120,"Performed By: Yashika(15BCS8033)");

getch();

closegraph();

}

**Output:**



**EXPERIMENT NO 4.**

**AIM: To display simple shapes (Like hut, star, car etc.) using graphics primitives**

**STAR:**

#include<graphics.h>

#include<conio.h>

void main()

{

intgd=DETECT,gm;

initgraph(&gd,&gm,"C:\\TC\\bgi");

line(150,100,100,200);

line(100,200,200,200);

line(200,200,150,100);

line(100,125,200,125);

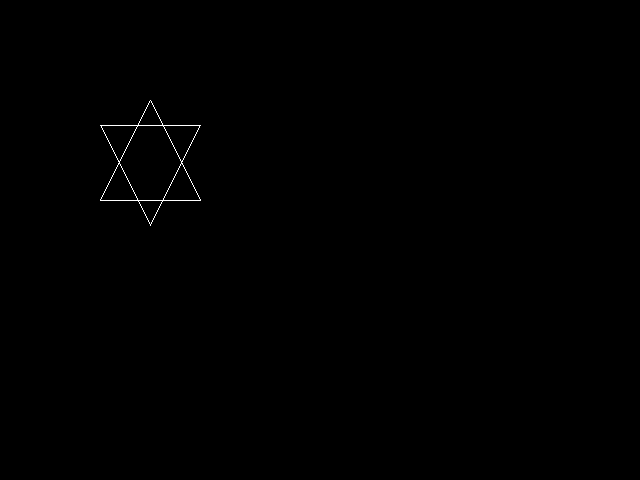
line(100,125,150,225);

line(150,225,200,125);

getch();

closegraph();

}



**CAR:**

#include <stdio.h>

#include <graphics.h>

#include <conio.h>

#include <dos.h>

intmain() {

    intgd = DETECT, gm;

    inti, maxx, midy;

    initgraph(&gd, &gm, "X:\\TC\\BGI");

    maxx = getmaxx();

        midy = getmaxy()/2;

    for(i=0; i < maxx-150; i=i+5) {

        cleardevice();

        setcolor(WHITE);

        line(0, midy + 37, maxx, midy + 37);

        setcolor(YELLOW);

        setfillstyle(SOLID\_FILL, RED);

        line(i, midy + 23, i, midy);

        line(i, midy, 40 + i, midy - 20);

        line(40 + i, midy - 20, 80 + i, midy - 20);

        line(80 + i, midy - 20, 100 + i, midy);

        line(100 + i, midy, 120 + i, midy);

        line(120 + i, midy, 120 + i, midy + 23);

        line(0 + i, midy + 23, 18 + i, midy + 23);

        arc(30 + i, midy + 23, 0, 180, 12);

        line(42 + i, midy + 23, 78 + i, midy + 23);

        arc(90 + i, midy + 23, 0, 180, 12);

        line(102 + i, midy + 23, 120 + i, midy + 23);

        line(28 + i, midy, 43 + i, midy - 15);

        line(43 + i, midy - 15, 57 + i, midy - 15);

        line(57 + i, midy - 15, 57 + i, midy);

        line(57 + i, midy, 28 + i, midy);

        line(62 + i, midy - 15, 77 + i, midy - 15);

        line(77 + i, midy - 15, 92 + i, midy);

        line(92 + i, midy, 62 + i, midy);

        line(62 + i, midy, 62 + i, midy - 15);

        floodfill(5 + i, midy + 22, YELLOW);

        setcolor(BLUE);

        setfillstyle(SOLID\_FILL, DARKGRAY);

        circle(30 + i, midy + 25, 9);

        circle(90 + i, midy + 25, 9);

        floodfill(30 + i, midy + 25, BLUE);

        floodfill(90 + i, midy + 25, BLUE);

          delay(100);

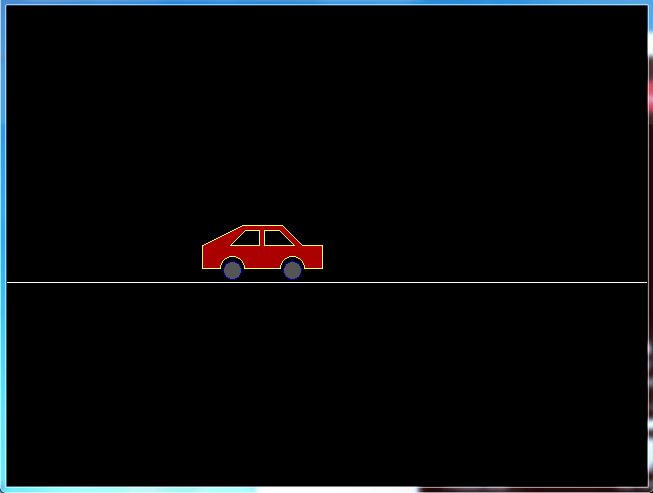
    }

    getch();

    closegraph();

    return0;

}



**PROGRAM NO. 5(a)**

**AIM-** To perform translation transformation on a given triangle.

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

inttx,ty;

intgd=DETECT,gm;

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

line(50,150,200,150);

line(50,150,125,50);

line(125,50,200,150);

cout<<"Enter translation factors"<<endl;

cout<<"Enter tx: ";

cin>>tx;

cout<<"Enter ty: ";

cin>>ty;

line(50+tx,150+ty,200+tx,150+ty);

line(50+tx,150+ty,125+tx,50+ty);

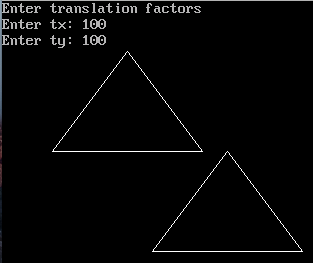
line(125+tx,50+ty,200+tx,150+ty);

getch();

closegraph();

}

**OUTPUT:-**



**PROGRAM NO. 5(b)**

**AIM-** To perform scaling transformation on a given triangle.

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int x;

intgd=DETECT,gm;

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

line(100,200,250,200);

line(100,200,175,100);

line(175,100,250,200);

cout<<"Enter Scaling factors"<<endl;

cin>>x;

line(100-2\*(5\*x),200+(5\*x),250+2\*(5\*x),200+(5\*x));

line(100-2\*(5\*x),200+(5\*x),175,100-2\*(5\*x));

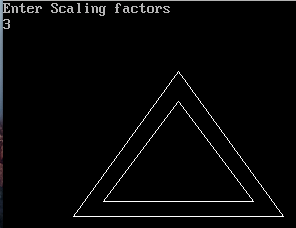
line(175,100-2\*(5\*x),250+2\*(5\*x),200+(5\*x));

getch();

closegraph();

}

**OUTPUT:-**



**PROGRAM NO. 5(c)**

**AIM-** To perform rotation transformation on a given triangle.

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

void main()

{

intgd = DETECT, gmode;

initgraph(&gd, &gmode, "C:\\turboc3\\bgi");

double x1, x2, x3, y1, y2, y3, rot;

cout<<"Enter First X - Coordinate : ";

cin>>x1;

cout<<"Enter First Y - Coordinate : ";

cin>>y1;

cout<<"Enter Second X - Coordinate : ";

cin>>x2;

cout<<"Enter Second Y - Coordinate : ";

cin>>y2;

cout<<"Enter Third X - Coordinate : ";

cin>>x3;

cout<<"Enter Third Y - Coordinate : ";

cin>>y3;

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

cout<<"Enter Rotation Angle in Degrees : ";

cin>>rot;

doublefpx = (x1+x2+x3)/3;

doublefpy = (y1+y2+y3)/3;

double rad = rot\*3.14/180;

double X1 = (x1-fpx)\*cos(rad) - (y1-fpy)\*sin(rad) + fpx;

double Y1 = (x1-fpx)\*sin(rad) + (y1-fpx)\*cos(rad) + fpy;

double X2 = (x2-fpx)\*cos(rad) - (y2-fpy)\*sin(rad) + fpx;

double Y2 = (x2-fpx)\*sin(rad) + (y2-fpx)\*cos(rad) + fpy;

double X3 = (x3-fpx)\*cos(rad) - (y3-fpy)\*sin(rad) + fpx;

double Y3 = (x3-fpx)\*sin(rad) + (y3-fpx)\*cos(rad) + fpy;

setcolor(BLUE);

line(X1, Y1, X2, Y2);

line(X2, Y2, X3, Y3);

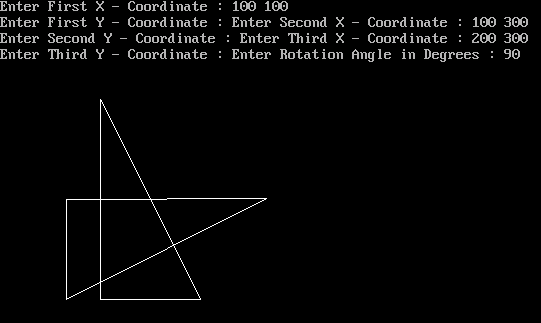
line(X3, Y3, X1, Y1);

getch();

closegraph();

}

**OUTPUT:-**



**PROGRAM NO. 6**

**AIM-** To rotate a given triangle clockwise and anticlockwise about a given point.

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

void main()

{

int gd = DETECT, gmode;

initgraph(&gd, &gmode, "C:\\turboc3\\bgi");

double x1, x2, x3, y1, y2, y3, rot;

cout<<"Enter First X - Coordinate : ";

cin>>x1;

cout<<"Enter First Y - Coordinate : ";

cin>>y1;

cout<<"Enter Second X - Coordinate : ";

cin>>x2;

cout<<"Enter Second Y - Coordinate : ";

cin>>y2;

cout<<"Enter Third X - Coordinate : ";

cin>>x3;

cout<<"Enter Third Y - Coordinate : ";

cin>>y3;

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

cout<<"Enter Rotation Angle in Degrees : ";

cin>>rot;

double fpx;

double fpy;

cout<<"Enter Point of rotation x1 & X2 : ";

cin>>fpx>>fpy;

double rad = rot\*3.14/180;

double X1 = (x1-fpx)\*cos(rad) - (y1-fpy)\*sin(rad) + fpx;

double Y1 = (x1-fpx)\*sin(rad) + (y1-fpx)\*cos(rad) + fpy;

double X2 = (x2-fpx)\*cos(rad) - (y2-fpy)\*sin(rad) + fpx;

double Y2 = (x2-fpx)\*sin(rad) + (y2-fpx)\*cos(rad) + fpy;

double X3 = (x3-fpx)\*cos(rad) - (y3-fpy)\*sin(rad) + fpx;

double Y3 = (x3-fpx)\*sin(rad) + (y3-fpx)\*cos(rad) + fpy;

line(X1, Y1, X2, Y2);

line(X2, Y2, X3, Y3);

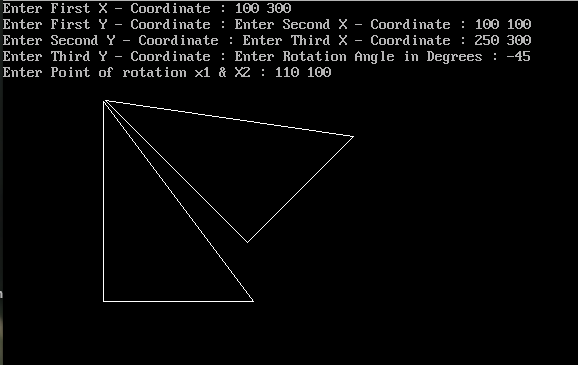
line(X3, Y3, X1, Y1);

getch();

closegraph();

}

**OUTPUT:-**



**PROGRAM NO. 7**

**AIM-** To perform reflection of a point about a line y=mx+c.

#include<graphics.h>

#include<math.h>

#include<iostream.h>

#include<conio.h>

void main()

{

intgd=DETECT,gm;

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

intmidx=getmaxx()/2;

intmidy=getmaxy()/2;

line(midx,0,midx,getmaxy());

line(0,midy,getmaxx(),midy);

rectangle(50,50,100,100);

//setcolor(RED);

rectangle((midx-50)+midx,50,(midx-100)+midx,100);

line(100,midy+50,100,midy+150);

line(100,midy+50,midx-100,midy+150);

line(100,midy+150,midx-100,midy+150);

//setcolor(YELLOW);

line(getmaxx()-100,midy+50,getmaxx()-100,midy+150);

line(getmaxx()-100,midy+50,midx+100,midy+150);

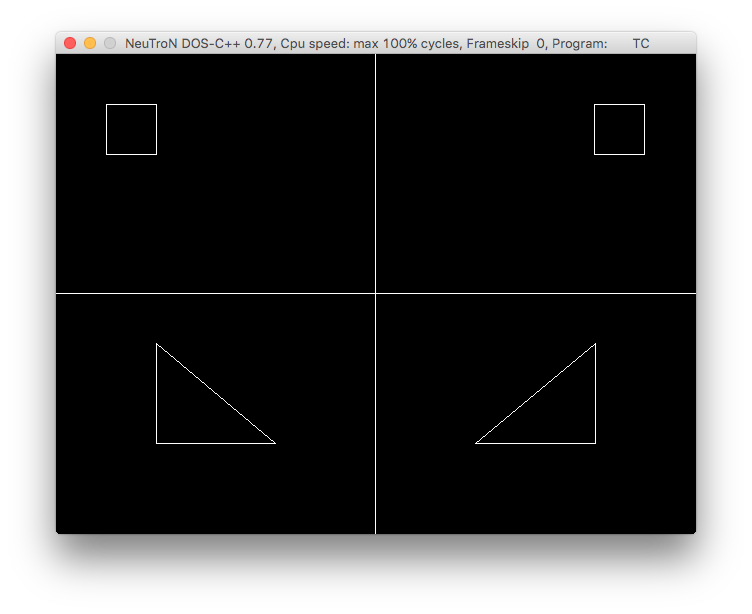
line(getmaxx()-100,midy+150,midx+100,midy+150);

getch();

closegraph();

}

**OUTPUT:-**



**EXPERIMENT NO. 8**

**AIM-** To draw a circle with a given centre and radius using midpoint circle algorithm.

**Source Code:**

#include<conio.h>7

#include<iostream.h>

#include<dos.h>

#include<graphics.h>

void main()

{

intgd=DETECT,gm;

inti,r,x,y,xc,yc;

float d;

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

cout<<"Enter Radius\n";

cin>>r;

cout<<"Enter Center of circle\n";

cin>>xc;

cin>>yc;

d=1.25-r;

x=0;

y=r;

do

{

if(d<0)

{

x=x+1;

d=d+2\*x+1;

}

else

{

x=x+1;

y=y-1;

d=d+2\*x-2\*y+10;

}

putpixel(xc+x,yc+y,5);

putpixel(xc-y,yc-x,5);

putpixel(xc+y,yc-x,6);

putpixel(xc-y,yc+x,6);

putpixel(xc+y,yc+x,2);

putpixel(xc-x,yc-y,3);

putpixel(xc+x,yc-y,4);

putpixel(xc-x,yc+y,4);

delay(100);

}

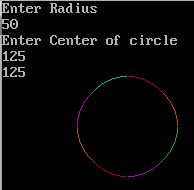
while(x<y);

getch();

closegraph();

}

**OUTPUT:-**



**EXPERIMENT NO. 9**

**AIM-** To draw a circle with a given centre and radius using circle generator algorithm.

**Source Code:**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

#include<dos.h>

voidcircleplot(int,int,int,int);

voidcirclepoint(intxc,intyc,float r)

{

doublex,y;

for(x=0;x<=y;x++)

{

double temp=((r\*r)-(x\*x));

y=sqrt(temp);

circleplot(xc,yc,x,y);

delay(50);

}

}

void main()

{

intgd=DETECT,gm,x,y,r;

clrscr();

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

cout<<"Enter center point(Xc,Yc):\t";

cin>>x>>y;

cout<<"Enter Radius\t";

cin>>r;

circlepoint(x,y,r);

getch();

closegraph();

}

voidcircleplot(intxc,intyc,intx,int y)

{

putpixel(xc+x,yc+y,1);

putpixel(xc+x,yc-y,2);

putpixel(xc-x,yc+y,3);

putpixel(xc-x,yc-y,4);

putpixel(xc+y,yc-x,5);

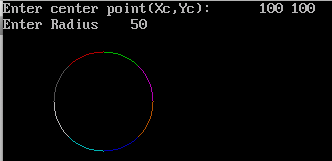
putpixel(xc+y,yc+x,6);

putpixel(xc-y,yc+x,7);

putpixel(xc-y,yc-x,8);

}

**OUTPUT:-**



**PROGRAM NO. 10**

**AIM-** To display 4-bit region code for end point of a line and check whether line iscompletely on the screen or off the screen.

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

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

float x,y;

int x1,x2,y1,y2;

x=getmaxx();

y=getmaxy();

cout<<"Enter the first coordinate of a line:\n";

cin>>x1>>y1;

cout<<"Enter the second coordinate of a line:\n";

cin>>x2>>y2;

// setcolor(BLUE);

line(x1,y1,x2,y2);

{

setcolor(WHITE);

line(x/3,0,x/3,y);

line(2\*x/3,0,2\*x/3,y);

line(0,y/3,x,y/3);

line(0,2\*y/3,x,2\*y/3);

outtextxy(2\*x/3-50,y/3-10,"1000");

outtextxy(x-50,y/3-10,"1010");

outtextxy(x/3-50,2\*y/3-10,"0001");

outtextxy(2\*x/3-50,2\*y/3-10,"0000");

outtextxy(x-50,2\*y/3-10,"0010");

outtextxy(x/3-50,y-10,"0101");

outtextxy(2\*x/3-50,y-10,"0100");

outtextxy(x-50,y-10,"0110");

}

if(x1&x2<x/3&&y1&y2<y/3)

{

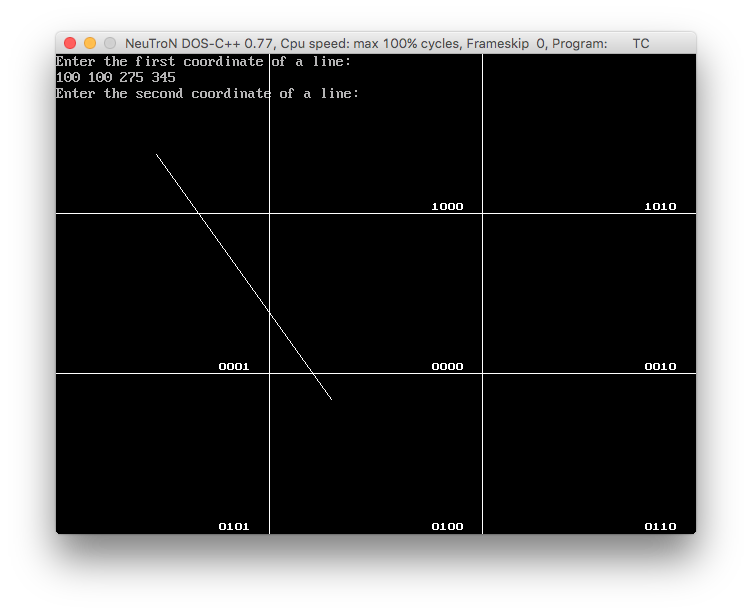
outtextxy(x/3-50,y/3-10,"1001");

}

getch();

}

**OUTPUT:-**



**PROGRAM NO. 11**

**AIM-** To clip a line intersecting at one point withgiven window using Cohen Sutherland LineClipping algorithm.

#include<iostream.h>

#include<conio.h>

#include<math.h>

#include<graphics.h>

#include<process.h>

int pixels[2][4];

float xn1,xn2,yn1,yn2,m;

void cohen(float x1,float y1,float x2,float y2,float xmin,float ymin,float xmax,float ymax)

{

int i,j,fl;

for(i=0;i<2;i++)

for(j=0;j<4;j++)

pixels[i][j]=0;

if(y1>ymax)

pixels[0][0]=1;

if(y1<ymin)

pixels[0][1]=1;

if(x1>xmax)

pixels[0][2]=1;

if(x1<xmin)

pixels[0][3]=1;

if(y2>ymax)

pixels[1][0]=1;

if(y2<ymin)

pixels[1][1]=1;

if(x2>xmax)

pixels[1][2]=1;

if(x2<xmin)

pixels[1][3]=1;

for(j=0;j<4;j++)

{

if(pixels[0][j]==0&&pixels[1][j]==0)

fl=1;

else if(pixels[0][j]==1&&pixels[1][j]==1)

{

fl=3;

break;

}

else

fl=2;

}

switch(fl)

{

case 1:

cout<<"line is visible";

break;

case 3:

cout<<"Line Is Not Visible";

break;

case 2:

cout<<"CLipping Candidate";

m=(y2-y1)/(x2-x1);

xn1=x1;

yn1=y1;

xn2=x2;

yn2=y2;

if(pixels[0][0]==1)

{

xn1=x1+(ymax-y1)/m;

yn1=ymax;

}

if(pixels[0][1]==1)

{

xn1=x1+(ymin-y1)/m;

yn1=ymin;

}

if(pixels[0][2]==1)

{

yn1=y1+(xmax-x1)\*m;

xn1=xmax;

}

if(pixels[0][3]==1)

{

yn1=y1+(xmin-x1)\*m;

xn1=xmin;

}

if(pixels[1][0]==1)

{

xn2=x2+(ymax-y2)/m;

yn2=ymax;

}

if(pixels[1][1]==1)

{

xn2=x2+(ymin-y2)/m;

yn2=ymin;

}

if(pixels[1][2]==1)

{

yn2=y2+(xmax-x2)\*m;

xn2=xmax;

}

if(pixels[1][3]==1)

{

yn2=y2+(xmin-x2)\*m;

xn2=xmin;

}

line(xn1,yn1,xn2,yn2);

cout<<"Line Clipped";

break;

}

}

void main()

{

int gd=DETECT,gm,i,j;

float xmin,ymin,xmax,ymax,x1,y1,x2,y2;

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

clearviewport();

cout<<"Enter xmin,ymin,xmax,ymax \n";

cin>>xmin>>ymin>>xmax>>ymax;

rectangle(xmin,ymin,xmax,ymax);

cout<<"Enter x1,y1,x2,y2 \n";

cin>>x1>>y1>>x2>>y2;

line(x1,y1,x2,y2);

setcolor(YELLOW);

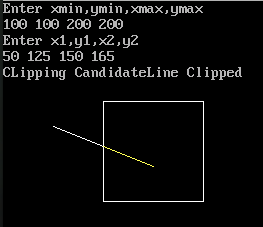
cohen(x1,y1,x2,y2,xmin,ymin,xmax,ymax);

getch();

closegraph();

}

**OUTPUT:-**



**PROGRAM NO. 12**

**AIM-** To clip a line intersecting at Two or morepoints with given window using CohenSutherland Line Clipping algorithm.

#include<iostream.h>

#include<conio.h>

#include<math.h>

#include<graphics.h>

#include<process.h>

int pixels[2][4];

float xn1,xn2,yn1,yn2,m;

void cohen(float x1,float y1,float x2,float y2,float xmin,float ymin,float xmax,float ymax)

{

int i,j,fl;

for(i=0;i<2;i++)

for(j=0;j<4;j++)

pixels[i][j]=0;

if(y1>ymax)

pixels[0][0]=1;

if(y1<ymin)

pixels[0][1]=1;

if(x1>xmax)

pixels[0][2]=1;

if(x1<xmin)

pixels[0][3]=1;

if(y2>ymax)

pixels[1][0]=1;

if(y2<ymin)

pixels[1][1]=1;

if(x2>xmax)

pixels[1][2]=1;

if(x2<xmin)

pixels[1][3]=1;

for(j=0;j<4;j++)

{

if(pixels[0][j]==0&&pixels[1][j]==0)

fl=1;

else if(pixels[0][j]==1&&pixels[1][j]==1)

{

fl=3;

break;

}

else

fl=2;

}

switch(fl)

{

case 1:

cout<<"line is visible";

break;

case 3:

cout<<"Line Is Not Visible";

break;

case 2:

cout<<"CLipping Candidate";

m=(y2-y1)/(x2-x1);

xn1=x1;

yn1=y1;

xn2=x2;

yn2=y2;

if(pixels[0][0]==1)

{

xn1=x1+(ymax-y1)/m;

yn1=ymax;

}

if(pixels[0][1]==1)

{

xn1=x1+(ymin-y1)/m;

yn1=ymin;

}

if(pixels[0][2]==1)

{

yn1=y1+(xmax-x1)\*m;

xn1=xmax;

}

if(pixels[0][3]==1)

{

yn1=y1+(xmin-x1)\*m;

xn1=xmin;

}

if(pixels[1][0]==1)

{

xn2=x2+(ymax-y2)/m;

yn2=ymax;

}

if(pixels[1][1]==1)

{

xn2=x2+(ymin-y2)/m;

yn2=ymin;

}

if(pixels[1][2]==1)

{

yn2=y2+(xmax-x2)\*m;

xn2=xmax;

}

if(pixels[1][3]==1)

{

yn2=y2+(xmin-x2)\*m;

xn2=xmin;

}

line(xn1,yn1,xn2,yn2);

cout<<"Line Clipped";

break;

}

}

void main()

{

int gd=DETECT,gm,i,j;

float xmin,ymin,xmax,ymax,x1,y1,x2,y2;

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

clearviewport();

cout<<"Enter xmin,ymin,xmax,ymax \n";

cin>>xmin>>ymin>>xmax>>ymax;

rectangle(xmin,ymin,xmax,ymax);

cout<<"Enter x1,y1,x2,y2 \n";

cin>>x1>>y1>>x2>>y2;

line(x1,y1,x2,y2);

setcolor(YELLOW);

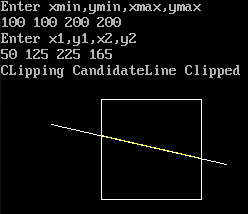
cohen(x1,y1,x2,y2,xmin,ymin,xmax,ymax);

getch();

closegraph();

}

**OUTPUT:-**



**PROGRAM NO. 13**

**AIM-** To display the result of window to viewporttransformation.

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

float sx,sy,Wxmin,Wxmax,Wymin,Wymax,Vxmin,Vxmax,Vymin,Vymax,x1,x2,y1,y2,Xv1,Xv2,Yv1,Yv2;

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

cout<<"Coordinates of window (Xmin,Ymin,Xmax,Ymax): ";

cin>>Wxmin>>Wymin>>Wxmax>>Wymax;

rectangle(Wxmin,Wymin,Wxmax,Wymax);

cout<<"Coordinates of Viewport (Xmin,Ymin,Xmax,Ymax): ";

cin>>Vxmin>>Vymin>>Vxmax>>Vymax;

rectangle(Vxmin,Vymin,Vxmax,Vymax);

cout<<"Coordinates of line(x1,y1,x2,y2):";

cin>>x1>>y1>>x2>>y2;

line(x1,y1,x2,y2);

sx=(Vxmax-Vxmin)/(Wxmax-Wxmin);

sy=(Vymax-Vymin)/(Wymax-Wymin);

Xv1=sx\*(x1-Wxmin)+Vxmin;

Xv2=sx\*(x2-Wxmin)+Vxmin;

Yv1=sy\*(y1-Wymin)+Vymin;

Yv2=sy\*(y2-Wymin)+Vymin;

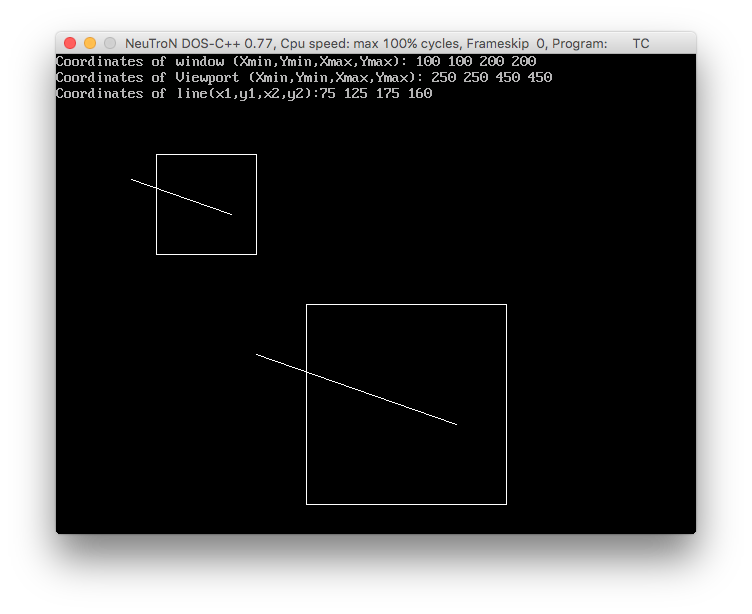
line(Xv1,Yv1,Xv2,Yv2);

getch();

closegraph();

}

**OUTPUT:-**



**PROGRAM NO. 14**

**AIM-** To clip a line intersecting at one point withgiven window using Cohen Sutherland LineClipping algorithm.

#include<iostream.h>

#include<conio.h>

#include<dos.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

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

int x[4],y[4],px,py,i;

cout<<"Enter four control points of bezier curve:";

for(i=0;i<4;i++) cin>>x[i]>>y[i];

double u;

for(u=0.0;u<=1.0;u+=0.001){

px=(1-u)\*(1-u)\*(1-u)\*x[0]+3\*u\*(1-u)\*(1-u)\*x[1]+3\*u\*u\*(1-u)\*x[2]+u\*u\*u\*x[3];

py=(1-u)\*(1-u)\*(1-u)\*y[0]+3\*u\*(1-u)\*(1-u)\*y[1]+3\*u\*u\*(1-u)\*y[2]+u\*u\*u\*y[3];

putpixel(px,py,WHITE);

delay(2);

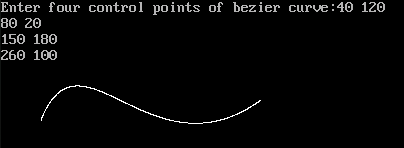
}

getch();

closegraph();

}

**OUTPUT:-**



**PROGRAM NO. 15**

**AIM-** To draw a B-Spline curve.

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#define max 6

typedefstruct

{

intx,y;}

POINT;

voidknotspline(POINT control[],double knot[],intnum,intklast);

double basis(inti,intk,double knot[],double stpos);

void main()

{

intgm,gd=DETECT;

//control points

POINT control[max]={10,180,80,130,250,10,400,700,500,50,550,70};

//knot vector

double knot[]={0,1,2,3,4,5,6,7};

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

//control,knotvector,no of control points,elements in knot

knotspline(control,knot,max,8);

getch();

closegraph();

}

voidknotspline(POINT control[],double knot[],intnum,intklast)

{

int k=4,i;

doublebas,stpos=knot[k-1],endpos=knot[klast-k],slice=(endpos-stpos)/100;

doublex,y,lx,ly;

lx=control[0].x; //first point

ly=control[0].y;

for(;stpos<endpos;stpos+=slice)

{

x=y=0;

for(i=1;i<=num;i++)

{

bas=basis(i-1,k,knot,stpos);

x=x+(control[i-1].x\*bas); //x,y for bspline curves

y=y+(control[i-1].y\*bas);

}

line(lx,ly,x,y);

lx=x; //last point

ly=y;

}

}

double basis(inti,intk,double knot[],double stpos)

{

doubleval; //recursion

if(k==1)

{

if(knot[i]<=stpos&&stpos<knot[i+1])

return(1);

else

return(0);

}

val=((stpos-knot[i])\*basis(i,k-1,knot,stpos))/(knot[i+k-1]-knot[i])+((knot[i+k]-stpos)\*basis(i+1,k-1,knot,stpos))/(knot[i+k]-knot[i+1]);

return(val);

}

**OUTPUT:-**

