**Program 1: Write a program in C++ to implement DDA line drawing algorithm.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

class line1

{

public:

void init1()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"c://tc//bgi");

}

void close()

{

closegraph();

}

void dda(float x1,float x2,float y1,float y2);

};

void line1:: dda(float x1,float x2,float y1,float y2)

{

float k=0;

float steps,xinc,yinc,dx,dy;

dx=x2-x1;

dy=y2-y1;

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

steps=abs(dx);

else

steps=abs(dy);

xinc=dx/steps;

yinc=dy/steps;

putpixel(x1,y1,WHITE);

do

{

x1=x1+xinc;

y1=y1+yinc;

k=k+1;

putpixel(x1,y1,WHITE);

}

while(k!=steps);

}

void main()

{

line1 l1;

clrscr();

l1.init1();

l1.dda(500,200,100,400);

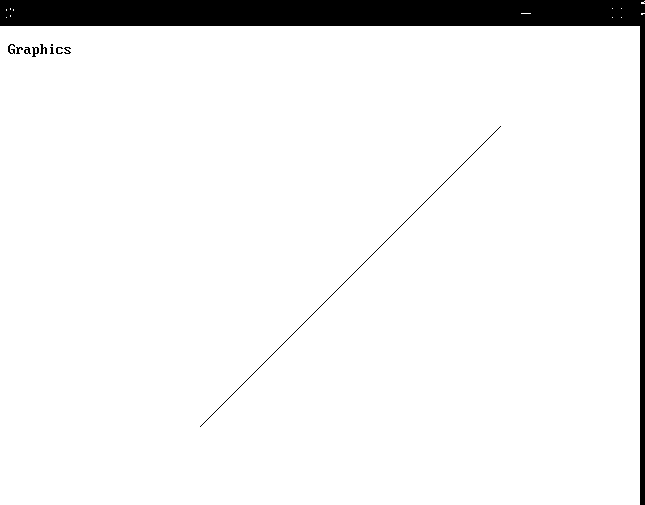
cout<<"\n Graphics";

getch();

l1.close();

}

**OUTPUT:**



**Program 2: Write a program in C++ for Bresenham’s line drawing algorithm.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

class line1

{

public:

void init1()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"c://tc//bgi");

}

void close()

{

closegraph();

}

void bres(int x1,int y1,int x2,int y2,int dx,int dy);

};

void line1::bres(int x1,int y1,int x2,int y2,int dx,int dy)

{

clrscr();

int p=0;

p=2\*dy-dx;

do

{

if(p<0)

{

if(x1<x2)

x1++ ;

else

x1--;

}

else

{

if(x1<x2)

x1++;

else

x1--;

p=2\*(dy-dx);

if(y1<y2)

y1++;

else

y2--;

}

putpixel(x1,y1,15);

}

while(x1!=x2);

}

void main()

{

line1 d;

float x1,x2,y1,y2;

clrscr();

d.init1();

int dx,dy;

cout<<"\nEnter the endpoints of a line.";

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

putpixel(x1,y1,15);

dx=abs(x2-x1);

dy=abs(y2-y1);

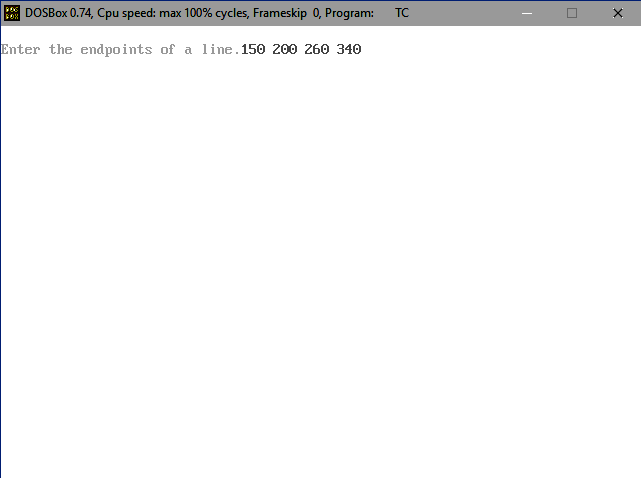
d.bres(x1,y1,x2,y2,dx,dy);

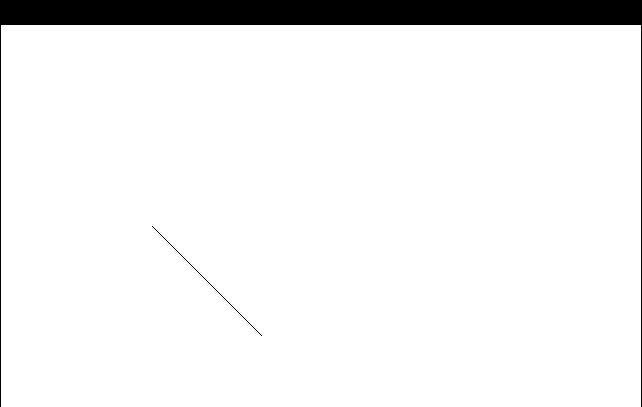
getch();

d.close();

}

**OUTPUT:**

****



**Program 3: Write a program in C++ to Draw a Circle using Mid-Point Algorithm.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void circlemidpoint(int,int,int);

void drawcircle(int,int,int,int);

void main()

{

int xc,yc,r;

int gd=DETECT,gm;

initgraph(&gd,&gm,"c://tc//bgi");

cout<<"Enter center co-ordinates of circle";

cin>>xc>>yc;

cout<<"Enter radius of circle";

cin>>r;

circlemidpoint(xc,yc,r);

getch();

}

void circlemidpoint(int xc,int yc,int r)

{

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

while(x<y)

{

x++;

if(p<0)

p+=2\*x+1;

else

{ y--;

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

}

drawcircle(xc,yc,x,y);

delay(50);

}

}

void drawcircle(int xc,int yc,int x, int y)

{

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

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

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

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

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

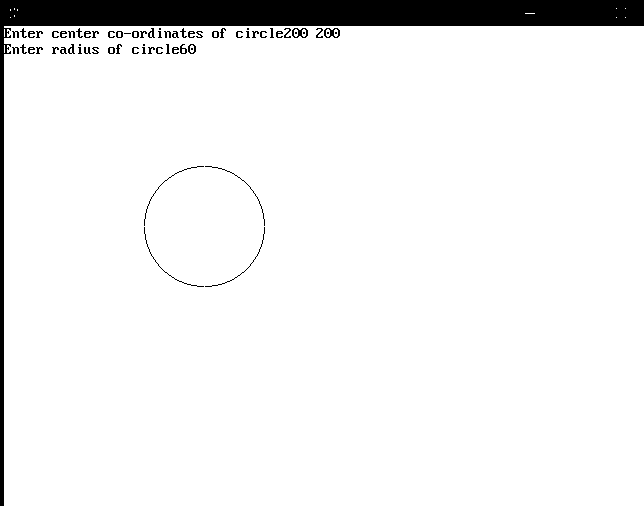
putpixel(xc-y,yc+x,RED);

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

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

}

**OUTPUT:**



**Program 4: Write a program in C++ to Draw an Ellipse using Mid- Point Algorithm.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void ellipsemidpoint(float,float,float,float);

void drawellipse(float,float,float,float);

void main()

{

int xc,yc,rx,ry;

int gd=DETECT,gm;

initgraph(&gd,&gm,"c://tc//bgi");

cout<<"Enter center co-ordinates of ellipse";

cin>>xc>>yc;

cout<<"Enter X-radius";

cin>>rx;

cout<<"Enter Y-radius";

cin>>ry;

ellipsemidpoint(xc,yc,rx,ry);

getch();

}

void ellipsemidpoint(float xc,float yc,float rx,float ry)

{

float rxsq=rx\*rx;

float rysq=ry\*ry;

float x=0,y=ry,p;

float px=0,py=2\*rxsq\*y;

drawellipse(xc,yc,x,y);

p=rysq-(rxsq\*ry)+(0.25\*rxsq);

while(px<py)

{

x++;

px+=2\*rysq;

if(p<0)

p+=rysq+px;

else

{ y--;

py-=2\*rxsq;

p+=rysq+px-py;

}

drawellipse(xc,yc,x,y);

}

p=rysq\*(x+0.5)\*(x+0.5)+rxsq\*(y-1)-rxsq\*rysq;

while(y>0)

{

y--;

py-=2\*rysq;

if(p>0)

p+=rxsq-py;

else

{

x++;

px+=2\*rysq;

p+=rxsq-py+px;

}

drawellipse(xc,yc,x,y);

delay(30);

}

}

void drawellipse(float xc,float yc,float x,float y)

{

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

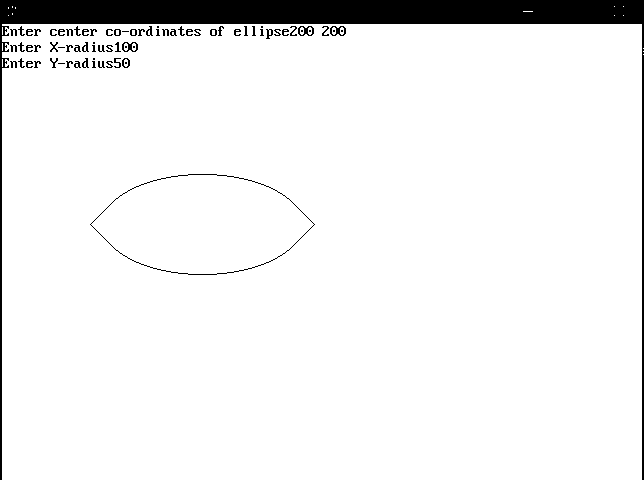
putpixel(xc-x,yc+y,RED);

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

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

}

**OUTPUT:**

****

**Program 5: Write a program in C++ to check point in polygon is inside or outside.**

#include<graphics.h>

#include<iostream.h>

#include<conio.h>

#include<math.h>

#include<stdlib.h>

class chk

{

private:

float xmin,ymin,xmax,ymax,m1[10],xy1[10][2];

int count,count1;

public:

void init1()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"c://tc//bgi");

}

void close()

{

closegraph();

}

void xymin(float x[][2],int n);

void draw(float x[][2],int n);

void test(float x[][2],int n);

chk(float x[][2],int n);

};

chk::chk(float x[][2],int n)

{

int i;

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

{

cin>>x[i][0]>>x[i][1];

}

x[n][0]=x[0][0];

x[n][1]=x[0][1];

}

void chk::xymin(float x[][2],int n)

{

int i;

xmin=x[0][0];

ymin=x[0][1];

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

{

if(xmin>x[i][0])

xmin=x[i][0];

if(ymin>x[i][1])

ymin=x[i][1];

}

xmin-=20;

ymin+=10;

cout<<"Enter the point to be tested:";

cin>>xmax>>ymax;

}

void chk::draw(float a[][2],int n)

{

int i;

moveto(a[0][0],a[0][1]);

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

lineto(a[i][0],a[i][1]);

line(xmin,ymin,xmax,ymax);

}

void chk::test(float a[][2],int n)

{

int i;

double tmp,integer;

float m=0,xy[10][1];

count=0;

if(xmax-xmin==0)

m=100000;

else

m=(ymax-ymin)/(xmax-xmin);

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

{

if(a[i+1][0]-a[i][1]==0)

m1[i]=100000;

else

(a[i+1][1]-a[i][1])/(a[i+1][0]-a[i][0]);

}

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

{

xy[i][0]=a[i][1]-ymin-(m1[i]\*a[i][0])+(m\*xmin);

if(m==m1[i])

{

xy[i][0]=10000;

xy[i][1]=10000;

}

else

{

xy[i][0]=xy[i][0]/(m-m1[i]);

xy[i][1]=m\*(xy[i][0]-xmin)+ymin;

}

tmp=modf(xy[i][0],&integer);

if(tmp>0.5)

xy[i][0]=ceil(xy[i][0]);

else

xy[i][0]=floor(xy[i][0]);

tmp=modf(xy[i][1],&integer);

if(tmp>0.5)

xy[i][1]=ceil(xy[i][1]);

else

xy[i][1]=floor(xy[i][1]);

tmp=modf(xy[i][0],&integer);

if(xy[i][0]&&xy[i][0]<=a[i+1][0]<=a[i][0]&&xy[i][0]>=a[i+1][0])

{

if(xy[i][1]&&xy[i][1]<=a[i+1][1]<=a[i][1]&&xy[i][1]>=a[i+1][1])

{

if(xy[i][0]<=xmax)

{

xy1[count][0]=xy[i][0];

xy1[count][0]=xy[i][1];

count++;

}

}

}

}

if(int(count)%2==0)

cout<<"\nThe point is outside.";

else

cout<<"\nThe point is inside.";

}

void main()

{

int i,n;

float a[10][2];

clrscr();

cout<<"\nEnter the number of vertices:";

cin>>n;

cout<<"\nEnter the vertices:";

chk d(a,n);

d.init1();

d.xymin(a,n);

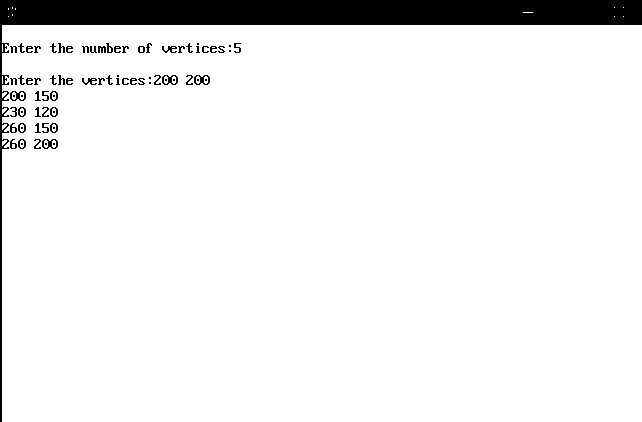
d.draw(a,n);

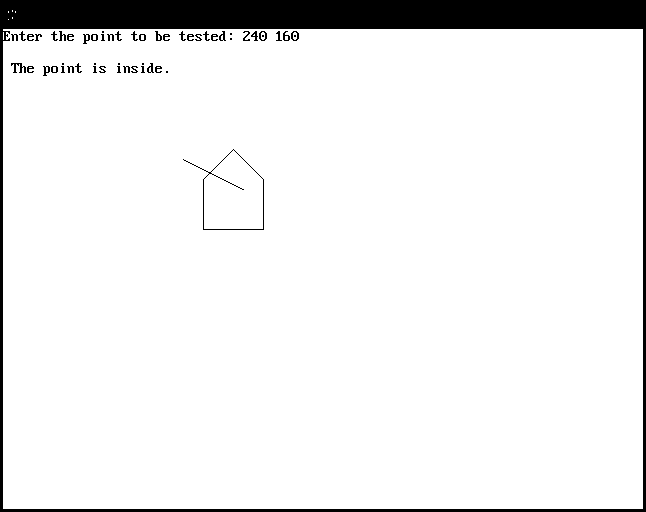
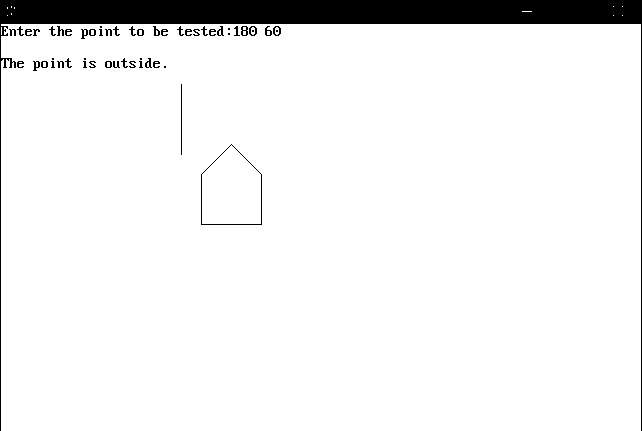
d.test(a,n);

getch();

}

**OUTPUT:**

****

****

**Program 6: Write a program in C++ for filling a Polygon using scan fill.**

#include<graphics.h>

#include<conio.h>

#include<iostream.h>

class ply

{

private:

int i,j,k,n;

int x[10],y[10];

public:

void userinput();

void graph();

void plotinit();

void scanline();

void Fill();

int ser();

};

void ply::userinput()

{

clrscr();

cout<<"\nEnter the number of vertices:";

cin>>n;

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

{

cout<<"\n Enter the "<<i+1<<" coordinate:";

cin>>x[i]>>y[i];

}

}

void ply::graph()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"c://tc//bgi");

}

void ply::plotinit()

{

setcolor(WHITE);

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

line(x[i],y[i],x[i+1],y[i+1]);

line(x[i],y[i],x[0],y[0]);

}

void ply::scanline()

{

int clr;

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

{

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

{

clr=getpixel(j,i);

if(clr==15)

Fill();

}

}

}

void ply::Fill()

{

int cl;

cl=getpixel(j+1,i);

while(cl!=15&&ser()%2!=0)

{

putpixel(j,i,WHITE);

if(cl!=15)

j++;

cl=getpixel(j,i);

}

}

int ply::ser()

{

int r,s,k=0;

for(r=j+1;r<getmaxx();r++)

{

s=getpixel(r,i);

if(s==15)

k++;

}

return(k);

}

void main()

{

ply pl;

pl.userinput();

pl.graph();

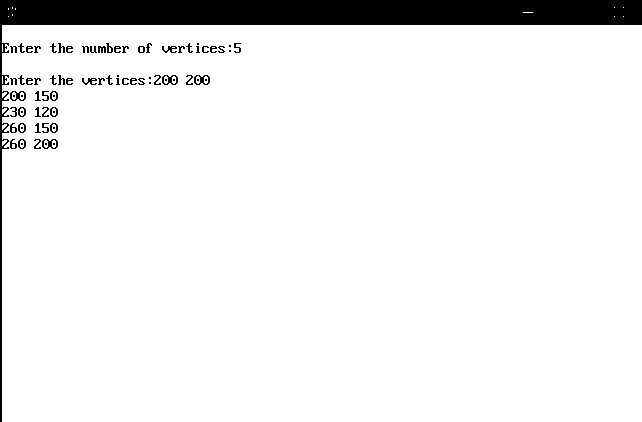
pl.plotinit();

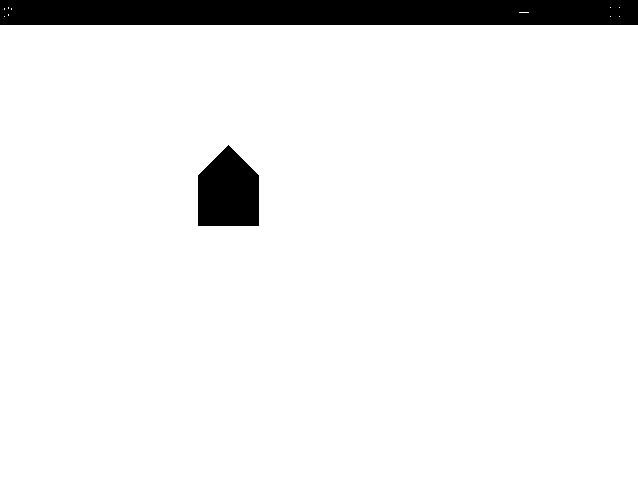
pl.scanline();

getch();

}

**OUTPUT:**

****

****

**Program 7: Write a program in C++ to demonstrate Translation transformation.**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<math.h>

#include<stdlib.h>

class trans

{

private:

float x[20],y[20],xm,ym,ref[2][2],shx,shy;

int i,j,k,n;

float sx,sy,tx,ty,ang;

int gd,gm;

float xtmp[20],ytmp[20];

public:

void init1()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"c://tc//bgi");

}

void close()

{

closegraph();

}

void takept();

void mapgraph();

void plotinit();

void translate();

void plotfinal();

};

int ch;

void trans::takept()

{

cout<<"\n Enter the number of vertices:-\t";

cin>>n;

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

{

cout<<"\n Enter the "<<i+1<<" coordinates\n";

cin>>x[i]>>y[i] ;

}

cout<<"\n Enter the coordinates for translation tx & ty:\t";

cin>>tx>>ty;

}

void trans::mapgraph()

{

xm=getmaxx()/2;

ym=getmaxy()/2;

line(xm,0,xm,2\*ym);

line(0,ym,2\*xm,ym);

}

void trans::plotinit()

{

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

line(x[i]+xm,(-y[i]+ym),x[i+1]+xm,(-y[i+1]+ym));

line(x[i]+xm,(-y[i]+ym),x[0]+xm,(-y[0]+ym));

}

void trans::translate()

{

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

{

xtmp[i]=x[i]+tx;

ytmp[i]=y[i]+ty;

}

}

void trans::plotfinal()

{

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

line(xtmp[i]+xm,(-ytmp[i]+ym),xtmp[i+1]+xm,(-ytmp[i+1]+ym));

line(xtmp[i]+xm,(-ytmp[i]+ym),xtmp[0]+xm,(-ytmp[0]+ym));

}

void main()

{

clrscr();

trans t1;

t1.takept();

t1.init1();

t1.mapgraph();

t1.plotinit();

t1.translate();

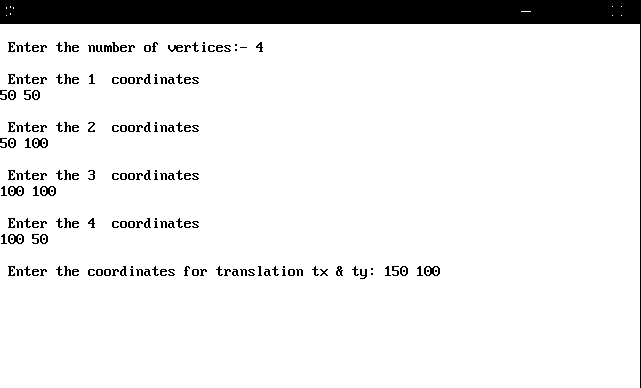
t1.plotfinal();

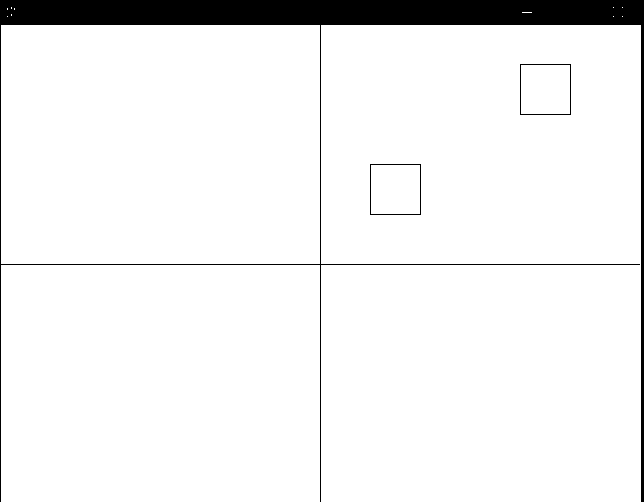
getch();

t1.close();

}

**OUTPUT:**

****

****

**Program 8: Write a program in C++ to demonstrate Scaling transformation.**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<math.h>

#include<stdlib.h>

class trans

{

private:

float x[20],y[20],xm,ym,ref[2][2],shx,shy;

int i,j,k,n;

float sx,sy,tx,ty,ang;

int gd,gm;

float xtmp[20],ytmp[20];

public:

void init1()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"c://tc//bgi");

}

void close()

{

closegraph();

}

void takept();

void mapgraph();

void plotinit();

void translate();

void scale();

void plotfinal();

};

int ch;

void trans::takept()

{

cout<<"\n Enter the number of vertices:-\t";

cin>>n;

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

{

cout<<"\n Enter the "<<i+1<<" coordinates\n";

cin>>x[i]>>y[i] ;

}

cout<<"\n Enter the coordinates for scalling sx & sy:\t";

cin>>sx>>sy;

}

void trans::mapgraph()

{

xm=getmaxx()/2;

ym=getmaxy()/2;

line(xm,0,xm,2\*ym);

line(0,ym,2\*xm,ym);

}

void trans::plotinit()

{

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

line(x[i]+xm,(-y[i]+ym),x[i+1]+xm,(-y[i+1]+ym));

line(x[i]+xm,(-y[i]+ym),x[0]+xm,(-y[0]+ym));

}

void trans::translate()

{

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

{

xtmp[i]=x[i]+tx;

ytmp[i]=y[i]+ty;

}

}

void trans::plotfinal()

{

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

line(xtmp[i]+xm,(-ytmp[i]+ym),xtmp[i+1]+xm,(-ytmp[i+1]+ym));

line(xtmp[i]+xm,(-ytmp[i]+ym),xtmp[0]+xm,(-ytmp[0]+ym));

}

void trans::scale()

{

float s[2][2],mxy[7][2],rxy[7][2];

s[0][0]=sx;

s[0][1]=0;

s[1][0]=0;

s[1][1]=sy;

tx=-x[0];

ty=-y[0];

translate();

k=0;

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

{

j=0;

mxy[i][j]=xtmp[k];

mxy[i][j+1]=ytmp[k];

k++;

}

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

{

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

{

rxy[i][j]=0;

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

{

rxy[i][j]=rxy[i][j]+mxy[i][k]\*s[k][j];

}

}

}

j=0;

k=0;

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

{

j=0;

x[k]=rxy[i][j];

y[k]=rxy[i][j+1];

k++;

}

tx=-tx;

ty=-ty;

translate();

}

void main()

{

clrscr();

trans t1;

t1.takept();

t1.init1();

t1.mapgraph();

t1.plotinit();

t1.scale();

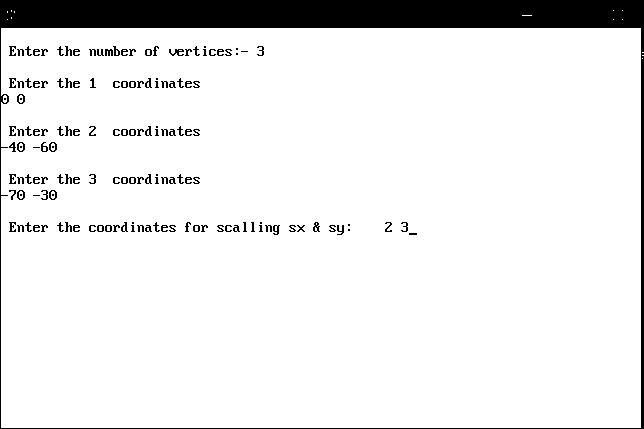
t1.plotfinal();

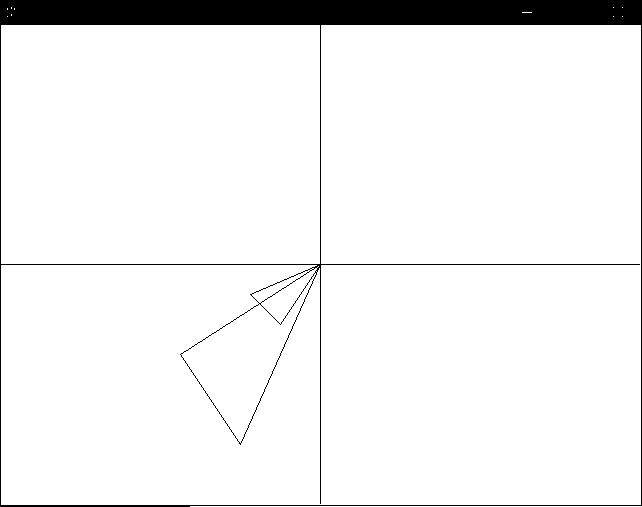
getch();

t1.close();

}

**OUTPUT:**

****

****

**Program 9: Write a program in C++ demonstrate Rotation transformation.**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<math.h>

class Rotate

{

public:

void init1()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"c://tc//bgi");

}

void close()

{

closegraph();

}

void dorotate();

};

void Rotate::dorotate()

{

float deg;

int x1=200,y1=200,x2=250,y2=250,x3=180,y3=270;

cout<<"Enter the angle";

cin>>deg;

deg=deg\*3.14/180;

int x,y;

x=x1;

y=y1;

x1=x\*cos(deg)-y\*sin(deg);

y1=x\*sin(deg)+y\*cos(deg);

x=x2;

y=y2;

x2=x\*cos(deg)-y\*sin(deg);

y2=x\*sin(deg)+y\*cos(deg);

x=x3;

y=y3;

x3=x\*cos(deg)-y\*sin(deg);

y3=x\*sin(deg)+y\*cos(deg);

line(x1,y1,x2,y2);

line(x2,y2,x3,y3);

line(x3,y3,x1,y1);

}

void main()

{

int x1=200,y1=200,x2=250,y2=250,x3=180,y3=270;

Rotate r;

clrscr();

r.init1();

line(x1,y1,x2,y2);

line(x2,y2,x3,y3);

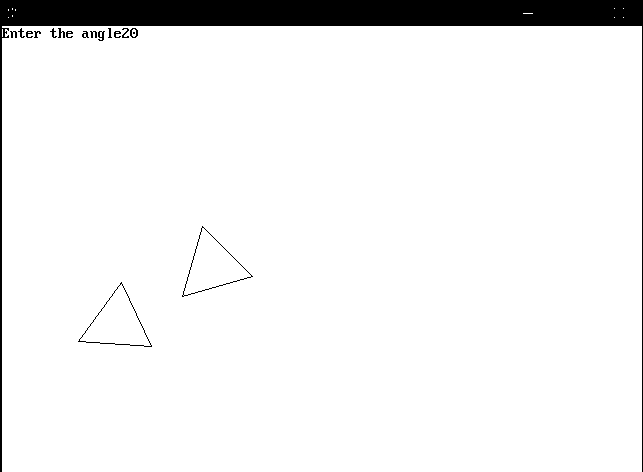
line(x3,y3,x1,y1);

r.dorotate();

getch();

}

**OUTPUT:**

****

**Program 10: Write a program in C++ demonstrate Reflection transformation.**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

class Reflect

{

private:

int i,Xmid,Ymid,X1,X2,X3,Y1,Y2,Y3,T1,T2,T3,option;

public:

void init1()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"c://tc//bgi");

}

void close()

{

closegraph();

}

void doreflect();

};

void Reflect:: doreflect()

{

cout<<"Enter the co-ordinate of Triangle:";

cin>>X1>>X2>>Y1>>Y2>>X3>>Y3;

cout<<"\nReflection About:\n";

cout<<"\nX-axis...1\nY-axis...2\nOrigin...3\nX=Y...4\nX=-Y...5";

cout<<"\nEnter your choice:";

cin>>option;

cleardevice();

Xmid=getmaxx()/2;

Ymid=getmaxy()/2;

line(5,Ymid,getmaxx()-5,Ymid);

line(Xmid+3,5,Xmid+3,getmaxy()-5);

line(X1+Xmid,Ymid-Y1,X2+Xmid,Ymid-Y2);

line(X2+Xmid,Ymid-Y2,X3+Xmid,Ymid-Y3);

line(X3+Xmid,Ymid-Y3,X1+Xmid,Ymid-Y1);

setcolor(255);

switch(option)

{

case 1:

outtextxy(50,10,"\nReflection along X-axis:");

Y1=-Y1;

Y2=-Y2;

Y3=-Y3;

break;

case 2:

outtextxy(50,10,"\nReflection along Y-axis:");

X1=-X1;

X2=-X2;

X3=-X3;

break;

case 3:

outtextxy(50,10,"\nReflection along Origin:");

X1=-X1;

X2=-X2;

X3=-X3;

Y1=-Y1;

Y2=-Y2;

Y3=-Y3;

break;

case 4:

outtextxy(50,10,"\nReflection along X=Y:");

T1=X1;

T2=X2;

T3=X3;

X1=Y1;

X2=Y2;

X3=Y3;

Y1=T1;

Y2=T2;

Y3=T3;

break;

case 5:

outtextxy(50,10,"\nReflection along X=-Y:");

T1=X1;

T2=X2;

T3=X3;

X1=-Y1;

X2=-Y2;

X3=-Y3;

Y1=-T1;

Y2=-T2;

Y3=-T3;

break;

}

line(X1+Xmid,Ymid-Y1,X2+Xmid,Ymid-Y2);

line(X2+Xmid,Ymid-Y2,X3+Xmid,Ymid-Y3);

line(X3+Xmid,Ymid-Y3,X1+Xmid,Ymid-Y1);

}

void main()

{

Reflect r;

clrscr();

r.init1();

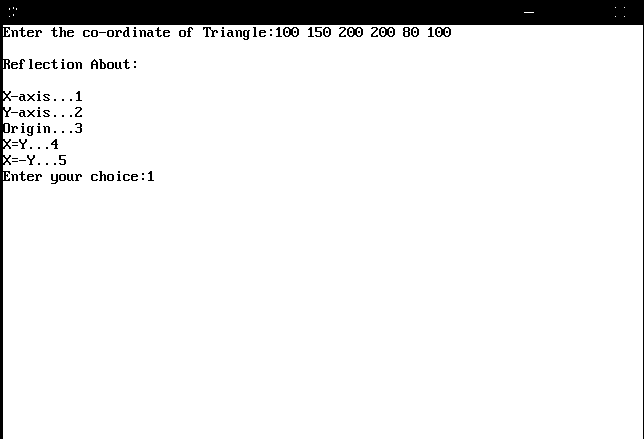
r.doreflect();

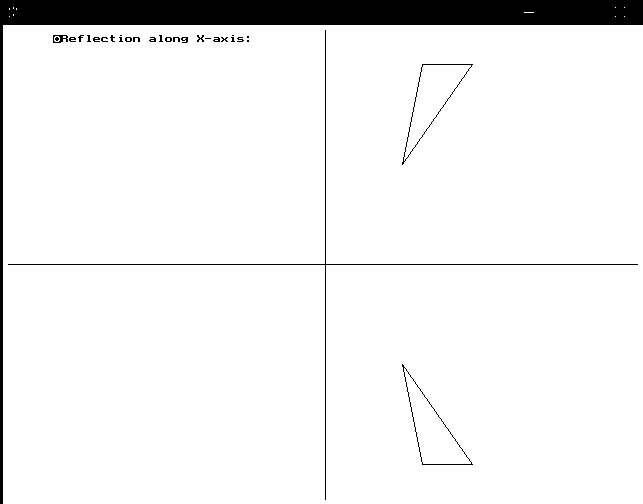
getch();

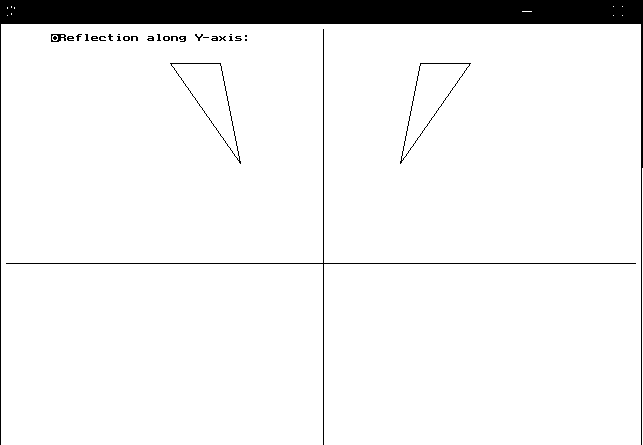
r.close();

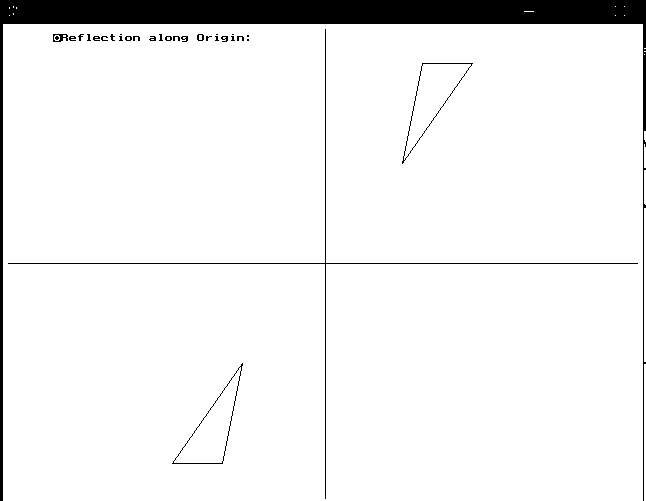
}

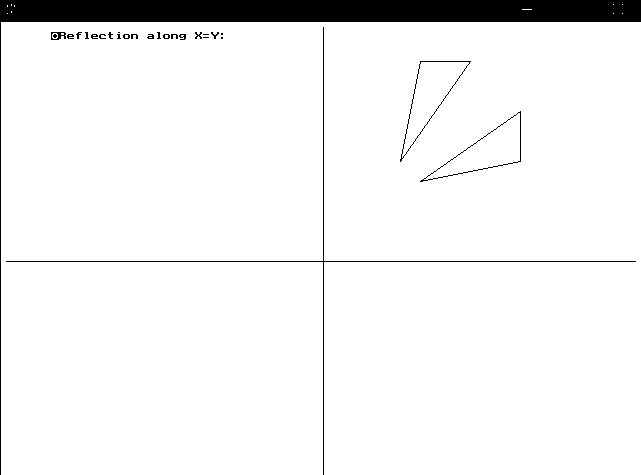
**OUTPUT:**

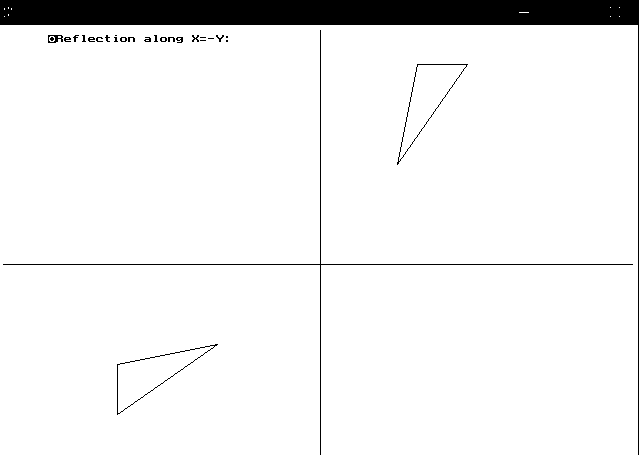
****

****

****

****

****

****

**Program 11: Write a program in C++ for polygon clipping consider appropriate window size.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

enum Area{LEFT,RIGHT,TOP,BOTTOM}id;

typedef struct

{

double x;

double y;

}

points;

points OutVertex[10];

points Vertex[10];

int max=0;

int n;

void SuthHodClip(int,int,int,int);

int Inside(int,points);

int Inside(int clipbound,points S)

{

int pos=0;

switch(id)

{

case LEFT:

if(S.x>clipbound)

pos=1;

break;

case RIGHT:

if(S.x<clipbound)

pos=1;

break;

case TOP:

if(S.y>clipbound)

pos=1;

break;

case BOTTOM:

if(S.y<clipbound)

pos=1;

break;

}

return pos;

}

points Intersect(int clipbound,points S,points P)

{

points TEMP;

double calc;

switch(id)

{

case LEFT:

case RIGHT:

TEMP.x=clipbound;

TEMP.y=S.y+(P.y-S.y)\*(clipbound-S.x)/(P.x-S.x);

break;

case BOTTOM:

case TOP:

TEMP.y=clipbound;

TEMP.x=S.x+(P.x-S.x)\*(clipbound-S.y)/(P.y-S.y);

break;

}

return TEMP;

}

void Clip(int xmin,Area id1)

{

int i;

points Temp;

points S,P;

int pt1,pt2;

id=id1;

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

{

S=Vertex[i];

if(i==(n-1))

P=Vertex[0];

else

P=Vertex[i+1];

pt1=Inside(xmin,S);

pt2=Inside(xmin,P);

if(pt1==1 && pt2==0)

{

Temp=Intersect(xmin,S,P);

OutVertex[max++]=Temp;

}

if(pt1==1 && pt2==1)

OutVertex[max++]=P;

if(pt1==0 && pt2==1)

{

Temp=Intersect(xmin,S,P);

OutVertex[max++]=Temp;

OutVertex[max++]=P;

}

}

n=max;

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

Vertex[i]=OutVertex[i];

max=0;

}

void SuthHodClip(int xmin,int xmax,int ymin,int ymax)

{

Clip(xmin,LEFT);

Clip(xmax,RIGHT);

Clip(ymin,TOP);

Clip(ymax,BOTTOM);

}

void main()

{

int xmin,xmax,ymin,ymax,y;

int gd=DETECT,gm;

sin(1);

clrscr();

cout<<"Enter the co-ordinates of the clipping windows:";

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

cout<<"Enter the number of verticesin the polygon:";

cin>>n;

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

{

cout<<"\nEnter co-ordinates of vertex:"<<i+1;

cin>>Vertex[i].x>>Vertex[i].y;

}

initgraph(&gd,&gm,"c://tc//bgi");

rectangle(xmin,ymin,xmax,ymax);

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

line(Vertex[i].x,Vertex[i].y,Vertex[i+1].x,Vertex[i+1].y);

line(Vertex[n-1].x,Vertex[n-1].y,Vertex[0].x,Vertex[0].y);

getch();

SuthHodClip(xmin,xmax,ymin,ymax);

getch();

cleardevice();

rectangle(xmin,ymin,xmax,ymax);

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

line(OutVertex[i].x,OutVertex[i].y,OutVertex[i+1].x,OutVertex[i+1].y);

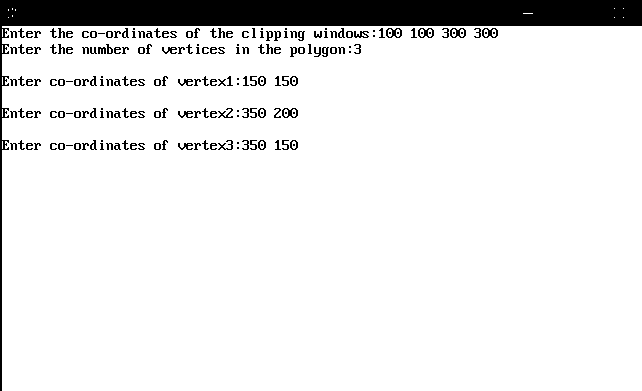
line(OutVertex[n-1].x,OutVertex[n-1].y,OutVertex[0].x,OutVertex[0].y);

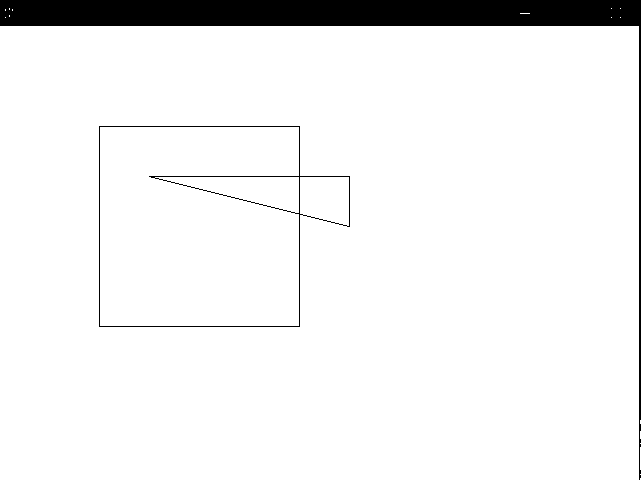
getch();

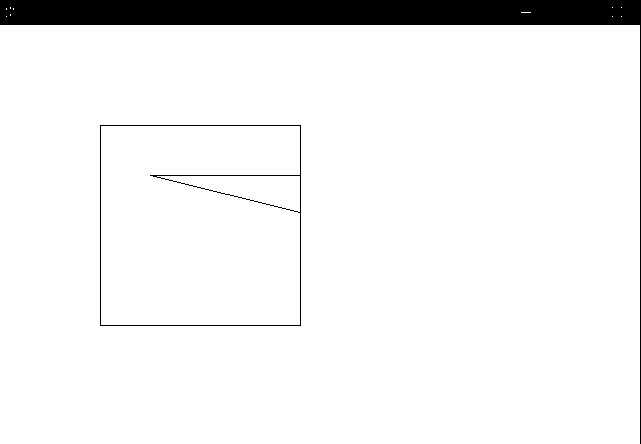
closegraph();

}

**OUTPUT:**





****

**Program 12: Write a program in C++ for cohen-sutherland line clipping algorithm.**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

typedef unsigned int outcode;

enum {TOP=0x1,BOTTOM=0x2,RIGHT=0x4,LEFT=0x8};

int calcode(x,y,xwmin,ywmin,xwmax,ywmax)

float x,y,xwmin,ywmin,xwmax,ywmax;

{

int code=0;

if(y>ywmax)

code|=TOP;

else if(y<ywmin)

code|=BOTTOM;

else if(x>xwmax)

code|=RIGHT;

else if(x<xwmin)

code|=LEFT;

return(code);

}

void lineclip(x0,y0,x1,y1,xwmin,ywmin,xwmax,ywmax)

float x0,y0,x1,y1,xwmin,ywmin,xwmax,ywmax;

{

int gd,gm;

outcode code0,code1,codeout;

int accept=0,done=0;

code0=calcode(x0,y0,xwmin,ywmin,xwmax,ywmax);

code1=calcode(x1,y1,xwmin,ywmin,xwmax,ywmax);

do

{

if(!(code0|code1))

{

accept=1;done=1;

}

else

if(code0&code1)done=1;

else

{

float x,y;

codeout=code0?code0:code1;

if(codeout&TOP)

{

x=x0+(x1-x0)\*(ywmax-y0)/(y1-y0);

y=ywmax;

}

else

if(codeout&BOTTOM)

{

x=x0+(x1-x0)\*(ywmin-y0)/(y1-y0);

y=ywmin;

}

else

if(codeout&RIGHT)

{

y=y0+(y1-y0)\*(xwmax-x0)/(x1-x0);

x=xwmax;

}

else

{

y=y0+(y1-y0)\*(xwmin-x0)/(x1-x0);

x=xwmin;

}

if(codeout==code0)

{

x0=x;y0=y;

code0=calcode(x0,y0,xwmin,ywmin,xwmax,ywmax);

}

else

{

x1=x;y1=y;

code1=calcode(x1,y1,xwmin,ywmin,xwmax,ywmax);

}

}

}

while(done==0);

if(accept)

line(x0,y0,x1,y1);

rectangle(xwmin,ywmin,xwmax,ywmax);

}

void main()

{

float x2,y2,x1,y1,xwmin,ywmin,xwmax,ywmax;

int gd=DETECT,gm;

clrscr();

initgraph(&gd,&gm,"c://tc//bgi");

printf("\n\n\tEnter the co-ordinates of line:");

printf("\n\n\tX1 Y1:");

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

printf("\n\n\tX2 Y2:");

scanf("%f%f",&x2,&y2);

printf("\n\n\tEnter the co-ordinates of window:\n");

printf("\n\txwmin,ywmin:");

scanf("%f%f",&xwmin,&ywmin);

printf("\n\txwmax,ywmax");

scanf("%f%f",&xwmax,&ywmax);

clrscr();

line(x1,y1,x2,y2);

rectangle(xwmin,ywmin,xwmax,ywmax);

getch();

clrscr();

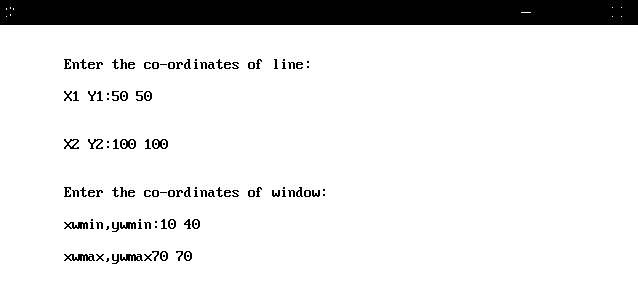
lineclip(x1,y1,x2,y2,xwmin,ywmin,xwmax,ywmax);

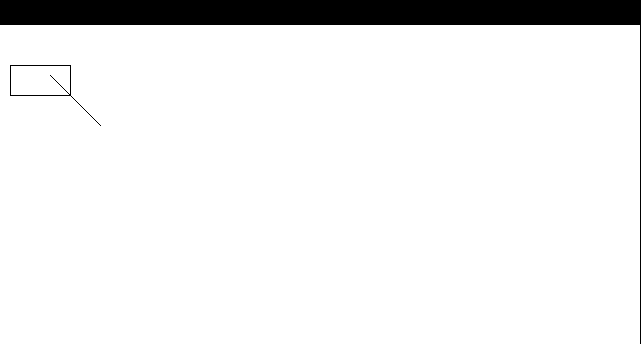
getch();

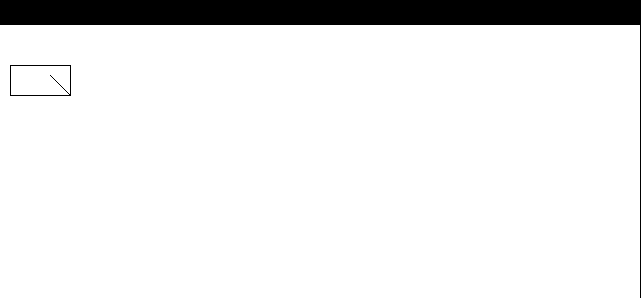
closegraph();

}

**OUTPUT:**

****

****

****

**Program 13: Write a program in C++ for Bezier curve.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

class Bezier

{

public:

void bezier(int x[4],int y[4]);

};

void Bezier::bezier(int x[4],int y[4])

{

int gd=DETECT,gm;

int i;

double t;

initgraph(&gd,&gm,"c://tc//bgi");

for(t=0.0;t<10;t+=0.0005)

{

double xt=pow(1-t,3)\*x[0]+3\*t\*pow(1-t,2)\*x[1]+3\*pow(t,2)\*(1-t)\*x[2]+pow(t,3)\*x[3];

double yt=pow(1-t,3)\*y[0]+3\*t\*pow(1-t,2)\*y[1]+3\*pow(t,2)\*(1-t)\*y[2]+pow(t,3)\*y[3];

putpixel(xt,yt,WHITE);

}

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

putpixel(x[i],y[i],YELLOW);

getch();

closegraph();

return;

}

void main()

{

Bezier b;

clrscr();

int x[4],y[4];

int i;

cout<<"Enter the x and yco-ordinates of the four control point:";

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

{

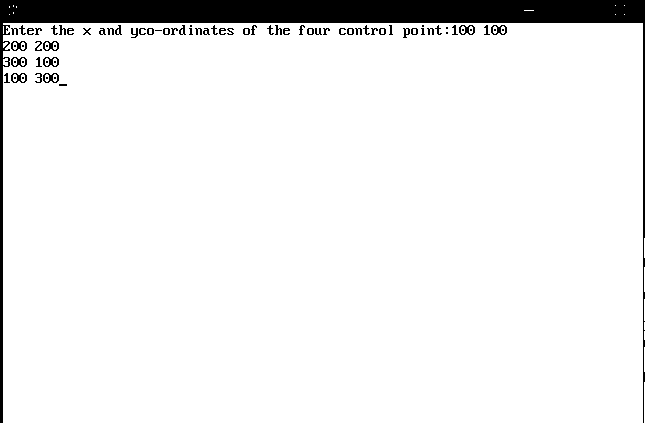
cin>>x[i]>>y[i];

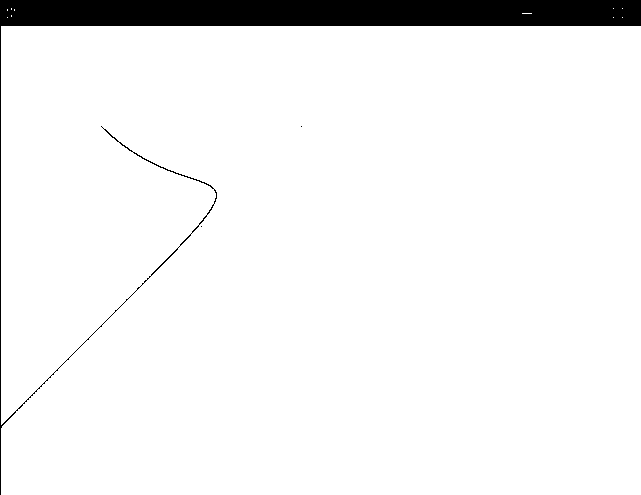
}

b.bezier(x,y);

}

**OUTPUT:**

****

****

**Program 14: Write a program in c++ for Gorum and Shading algorithm.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

classsde

{

private:

floatsa,sb,xa,xb;

public:

void use();

};

voidsde::use()

{

int poly1[10]={140,100,140,400,200,450,200,150,140,100};

int poly2[10]={200,150,200,450,300,475,300,175,200,150};

int poly3[10]={300,175,400,175,400,475,300,475,300,175};

int poly4[10]={400,175,500,150,500,450,400,475,400,175};

int poly5[10]={500,150,560,100,560,400,500,450,500,150};

intgd=DETECT,gm;

float m1,m2,m3,m4,m5;

float x,y,y1,shade;

clrscr();

initgraph(&gd,&gm,"c:/tc3/bgi");

drawpoly(5,poly1);

drawpoly(5,poly2);

drawpoly(5,poly3);

drawpoly(5,poly4);

drawpoly(5,poly5);

m1=50.0/60.0;

m2=25.0/100.0;

intinten[16]={1,4,2,2,8,6,5,3,9,12,10,7,13,14,15,15};

int k;

m3=0;

m4=-m2;

m5=-m1;

xa=140;

xb=560;

intch;

cout<<"1:LEFT LIGHT SOURCE \n 2:RIGHT LIGHT SOURCE \n ENTER YOUR CHOICE \n";

cin>>ch;

switch(ch)

{

case1:sa=15;sb=0;break;

case2:sa=0;sb=15;break;

}

for(inti=xa;i<=200;i++)

{

x=i;

y1=100;

y=m1\*(x-140)+100;

shade=sa+(i-xa)\*((sb-sa)/(xb-xa));

k=shade;

k=k%16;

setcolor(inten[k]);

line(x,y,x,y+300);

}

for(i=201;i<=300;i++)

{

x=i;

y1=150;

y=m2\*(x-201)+150;

shade=sa+(i-xa)\*((sb-sa)/(xb-xa));

k=shade;

k=k%16;

setcolor(inten[k]);

line(x,y,x,y+300);

}

for(i=301;i<=400;i++)

{

x=i;

y1=175;

y=m3\*(x-301)+175;

shade=sa+(i-xa)\*((sb+sa)/(xb-xa));

k=shade;

k=k%16;

setcolor(inten[k]);

line(x,y,x,y+300);

}

for(i=401;i<=500;i++)

{

x=i;

y1=175;

y=m4\*(x-401)+175;

shade=sa+(i-xa)\*((sb-sa)/(xb-xa));

k=shade;

k=k%16;

setcolor(inten[k]);

line(x,y,x,y+300);

}

for(i=560;i>=501;i--)

{

x=i;

y1=150;

y=m5\*(x-501)+150;

shade=sa+(i-xa)\*((sb-sa)/(xb-xa));

k=shade;

k=k%16;

setcolor(inten[k]);

line(x,y,x,y+300);

}

}

main()

{

sde s1;

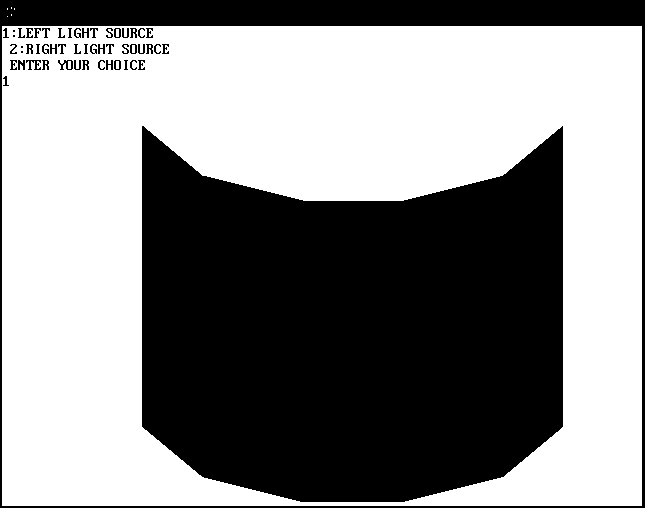
s1.use();

getch();

closegraph();

}

**OUTPUT:**



**Program 15: Write a program in c++ for graphics function for color model.**

#include<iostream.h>

#include<conio.h>

#include<dos.h>

#include<stdlib.h>

#include<graphics.h>

#include<math.h>

void main()

{

int gd=DETECT,gm;

float r=0,g=0,b=0;

float y=0,i=0,q=0;

float c=0,m=0,y1=0;

initgraph(&gd,&gm,"c:\\tc3\\bgi");

cout<<"(r,g,b) keys for incrementing R,G,B values respectively..";

cout<<"(shift+r,g,b) keys for decrementing R,G,B values respectively..";

cout<<"press esc to exit";

setcolor(15);

while(1)

{

gotoxy(18,10);

cout<<"R,G,B";

c=1.0-r;

m=1.0-g;

y1=1.0-y;

gotoxy(47,10);

cout<<"C,M,Y";

y=0.299\*r+0.587\*g+0.144\*b;

i=0.596\*r-0.275\*g-0.3218\*b;

q=0.212\*r-0.528\*g+0.311\*b;

gotoxy(18,23);

cout<<"Y,I,Q";

switch(getch())

{

case 'r':

r++;

break;

case 'g':

g++;

break;

case 'b':

b++;

break;

case 'R':

r--;

break;

case 'G':

g--;

break;

case 'B':

b--;

break;

case 27:

closegraph();

exit(0);

}

if (r>255)

r=0;

if (g>255)

g=0;

if (b>255)

b=0;

setrgbpalette(1,r,g,b);

setfillstyle(1,1);

bar(50,50,270,250);

rectangle(50,50,270,250);

setrgbpalette(2,c,m,y1);

setfillstyle(1,2);

bar(275,50,495,250);

rectangle(275,50,495,250);

setrgbpalette(3,y,i,q);

setfillstyle(1,3);

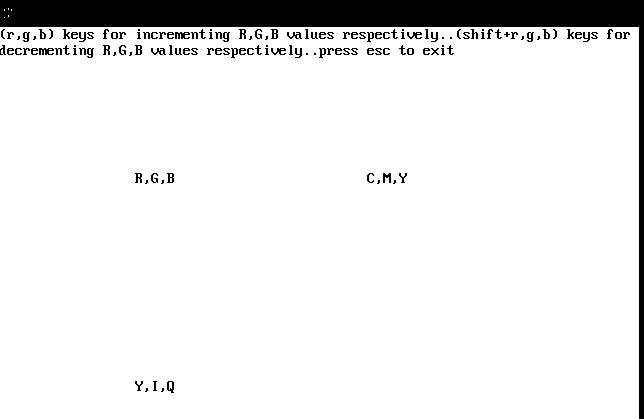
bar(50,255,270,455);

rectangle(50,255,270,455);

}

}

**OUTPUT:**

****

