**PMC-403**

**List of Practical**

1. Write a program to draw a line using DDA line generation algorithm

2. Write a program to draw a line using Bresenham’s line generation algorithm

3. Write a program to draw a circle using midpoint circle generation algorithm

4. Write a program to draw a circle using Bresenham’s circle generation algorithm

5. Write a program to implement boundary fill algorithm to fill a triangle.

6. Write a program to implement flood fill algorithm to fill a circle.

7. Write a program to implement Liang-Barsky line clipping algorithm.

8. Write a program to implement 2D reflection of a triangle.

9. Write a program to scale a triangle about origin

10. Write a program to rotate a triangle

11. Write a program to draw an ellipse using midpoint ellipse generation algorithm

12. Write a program to find implement cabinet/Cavalier projection of a unit cube.

13. Write a program to find perspective projection of a unit cube.

14. Write a program to draw cubic Bezier Curve.

Practical assignment:

* 1. **Write a program to draw a line using DDA line generation algorithm.**

#include<graphics.h> #include<stdio.h> #include<conio.h> #include<math.h> int main(void)

{

int gdriver=DETECT,gmode; int x1,x2,y1,y2;

float a,b;

int i=1,dx,dy,steps; float x,y;

initgraph(&gdriver,&gmode," "); printf("enter the values of x1 and y1"); scanf("%d%d",&x1,&y1);

printf("enter the values of x2 and y2"); scanf("%d%d",&x2,&y2);

dx=abs(x2-x1); printf("dx=%d",dx); dy=abs(y2-y1); printf("dy=%d",dy); if(dx>dy)

{

steps=dx;

}

else

{

steps=dy;

}

printf("no.of steps=%d",steps); x=dx/steps;

y=dy/steps; while(i<=steps)

{

a=a+x; b=b+y;

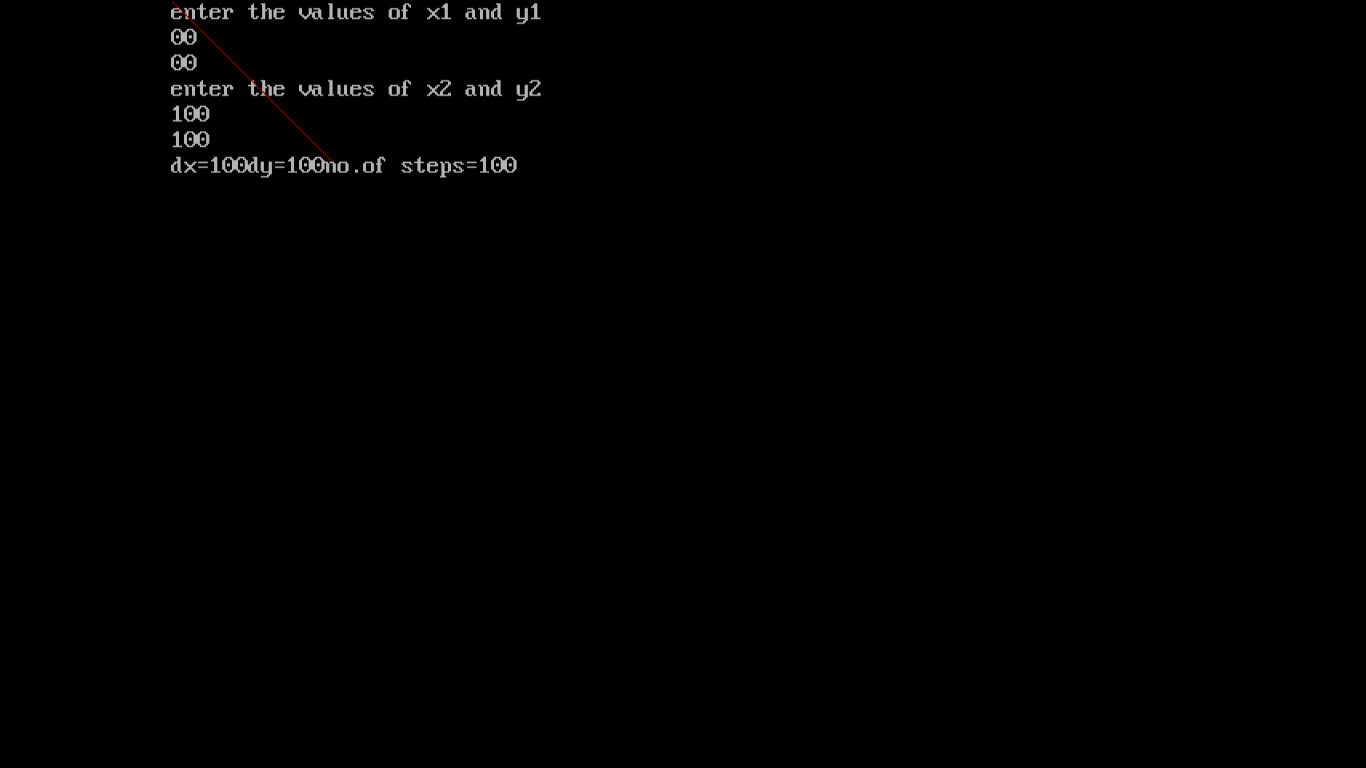
putpixel(a,b,RED); i++;

}

getch(); return(0);

}

**Output**



**Write a program to draw a line using Bresenham’s line generation algorithm.**

#include<graphics.h> #include<stdio.h> #include<math.h> #include<conio.h> int main(void)

{

int gdriver=DETECT,gmode; int x1,x2,y1,y2,dx,dy,p,x,y;

initgraph(&gdriver,&gmode," "); printf("enter the value of x1 and y1"); scanf("%d%d",&x1,&y1);

printf("enter the value of x2 and y2"); scanf("%d%d",&x2,&y2);

dx=x2-x1; dy=y2-y1; p=2\*dy-dx; x=x1; y=y1;

putpixel(x,y,WHITE); while(x<=x2)

{

if(p<0)

{

}

else

{

}

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

x=x+1; y=y+1;

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

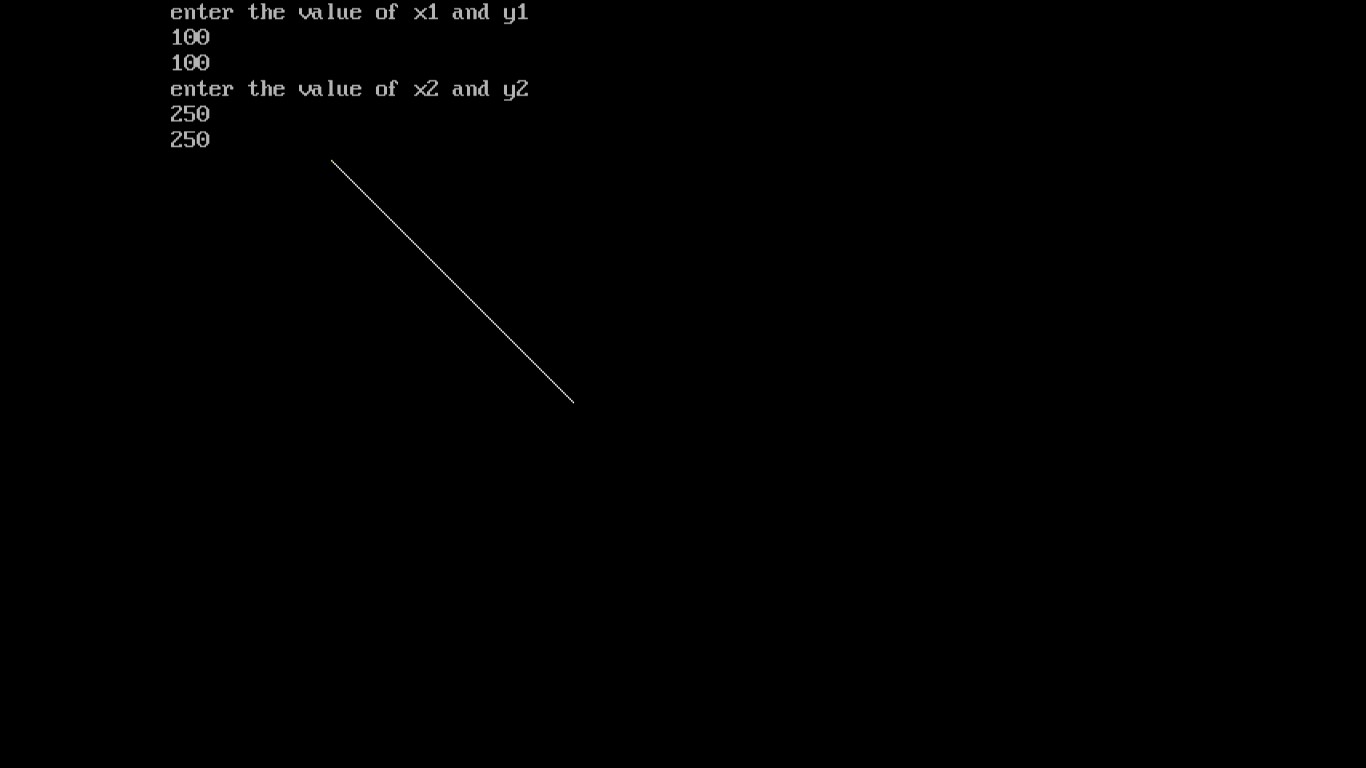
putpixel(x,y,WHITE);

}

getch(); return(0);

}

**Output**



* 1. **Write a program to draw a circle using midpoint circle generation algorithm.**

#include<graphics.h> #include<stdio.h> #include<math.h> #include<conio.h> int main(void)

{

int gdriver=DETECT,gmode; int x,y,r,p,xc,yc; clrscr();

initgraph(&gdriver,&gmode," "); printf("enter the value of x,y and radius"); scanf("%d%d%d",&xc,&yc,&r);

p=1-r; x=0;

y=r; do

{

if(p<0)

{

x=x+1; y=y;

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

}

else

{

x=x+1; y=y-1;

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

}

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

}

while(x<=y); getch(); return(0);

}

**Output:**



* 1. **Write a program to draw a circle using Bresenham’s circle generation algorithm.**

#include<stdio.h> #include<conio.h> #include<graphics.h> #include<math.h>

int main(void)

{

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

int gdriver=DETECT,gmode; initgraph(&gdriver,&gmode," "); printf("enter the center of the circle"); scanf("%d%d",&xc,&yc);

printf("enter the radius of the circle"); scanf("%d",&r);

p=3-2\*r; x=0;

y=r; do

{

if(p<0)

{

x=x+1; y=y; p=p+4\*x+1;

}

else

{

x=x+1; y=y-1;

p=p+4\*x-4\*y+1;

}

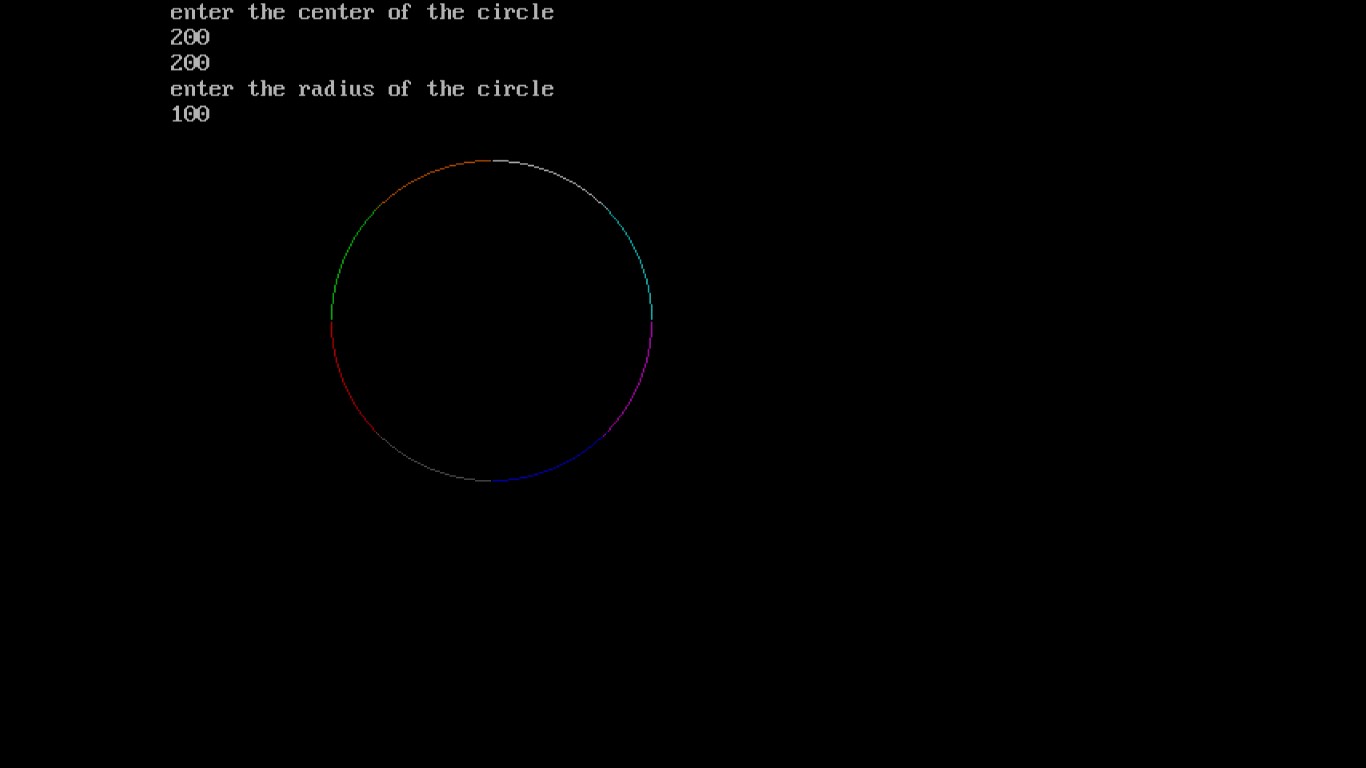
putpixel(xc+x,yc+y,1); putpixel(xc-y,yc-x,2); putpixel(xc+y,yc-x,3); putpixel(xc-y,yc+x,4); putpixel(xc+y,yc+x,5); putpixel(xc-x,yc-y,6); putpixel(xc+x,yc-y,7); putpixel(xc-x,yc+y,8);

}

while(x<=y); getch(); return(0);

}

**Output:**



* 1. **Write a program to scale a triangle about origin.**

#include<stdio.h> #include<conio.h> #include<math.h> #include<graphics.h> int main(void)

{

int gdriver=DETECT,gmode; int t[3][3],x[3][3],w[3][3];

int x1,y1,x2,y2,x3,y3,i,j,k,sx,sy; initgraph(&gdriver,&gmode," "); printf("\n enter the points");

scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3); printf("enter scaling factor in x"); scanf("%d",&sx);

printf("enter scaling factor in y"); scanf("%d",&sy);

printf("before scaling"); line(x1,y1,x2,y2);

line(x2,y2,x3,y3);

line(x1,y1,x3,y3);

t[0][0]=sx;

t[0][1]=0;

t[0][2]=0;

t[1][0]=0;

t[1][1]=sy;

t[1][2]=0;

t[2][0]=0;

t[2][1]=0;

t[2][2]=1;

x[0][0]=x1;

x[1][0]=y1;

x[0][1]=x2;

x[1][1]=y2;

x[0][2]=x3;

x[1][2]=y3;

x[2][0]=1;

x[2][1]=1;

x[2][2]=1;

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{ w[i][j]=0;

for(k=0;k<3;k++)

{

w[i][j]=w[i][j] + t[i][k] \* x[k][j];

}

}

}

printf("after scaling"); line(w[0][0],w[1][0],w[0][1],w[1][1]);

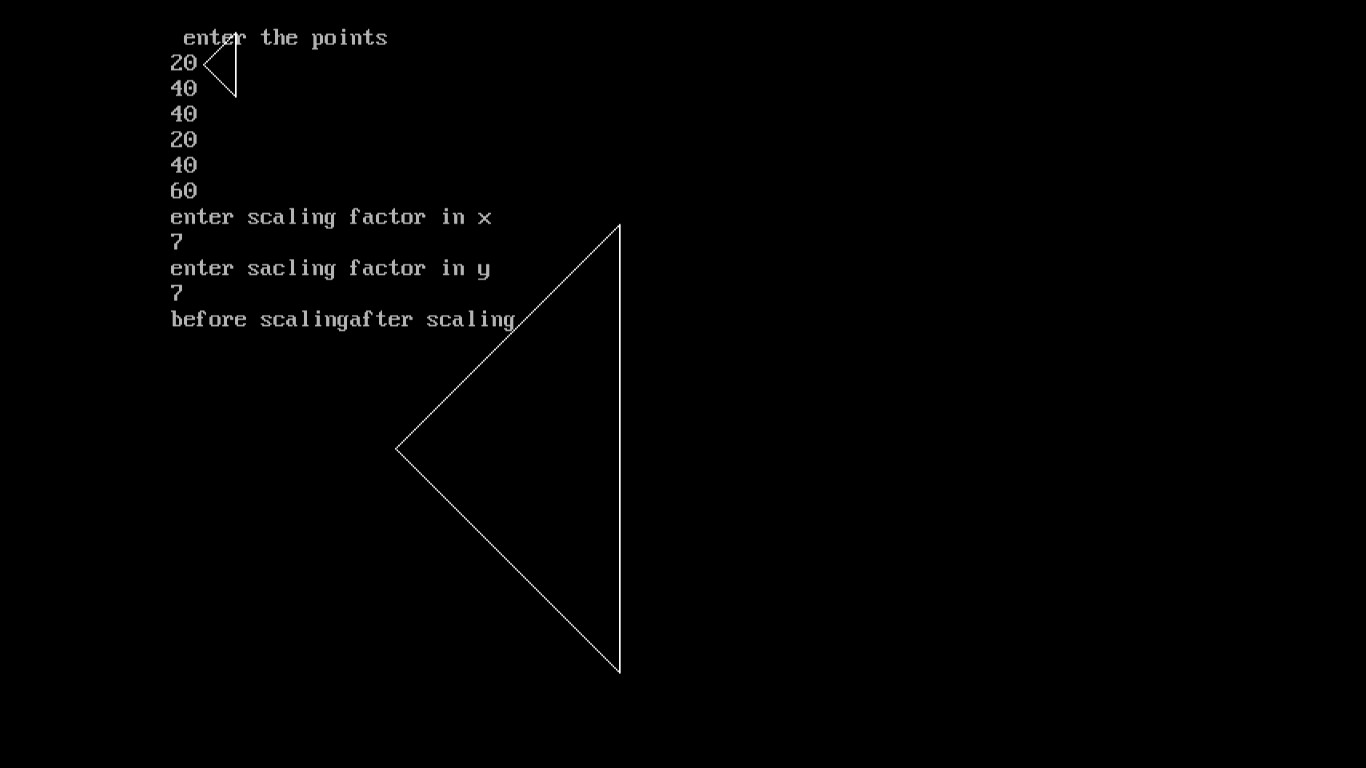
line(w[0][0],w[1][0],w[0][2],w[1][2]);

line(w[0][1],w[1][1],w[0][2],w[1][2]);

getch(); return(0);

}

**Output**



* 1. **Write a program to rotate a triangle.**

#include<stdio.h> #include<conio.h> #include<math.h> #include<graphics.h> #define pi 3.14

int main(void)

{

float t[3][3],r[3][3],o[3][3];

float x1,x2,x3,y1,y2,y3,d,i,j,k; float a;

int gdriver=DETECT,gmode; initgraph(&gdriver,&gmode," ");

printf("\n enter the values of triangle"); scanf("%f%f%f%f%f%f",&x1,&y1,&x2,&y2,&x3,&y3); printf("\n enter the degree of rotation"); scanf("%f",&d);

a=d\*(pi/180); printf("%f",a);

line(x1,y1,x2,y2);

line(x1,y1,x3,y3);

line(x2,y2,x3,y3);

r[0][0]=cos(a);

r[0][1]=sin(a);

r[0][2]=0;

r[1][0]=sin(-a);

r[1][1]=cos(a);

r[1][2]=0;

r[2][0]=0;

r[2][1]=0;

r[2][2]=1;

o[0][0]=x1;

o[0][1]=x2;

o[0][2]=x3;

o[1][0]=y1;

o[1][1]=y2;

o[1][2]=y3;

o[2][0]=1;

o[2][1]=1;

o[2][2]=1;

printf("\n after rotating in clockwise direction"); for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{ t[i][j]=0;

for(k=0;k<3;k++)

{

t[i][j]=t[i][j] + r[i][k] \* o[k][j];

}

printf("%f \t",t[i][j]);

}

printf("\n");

} line(t[0][0],t[1][0],t[0][1],t[1][1]);

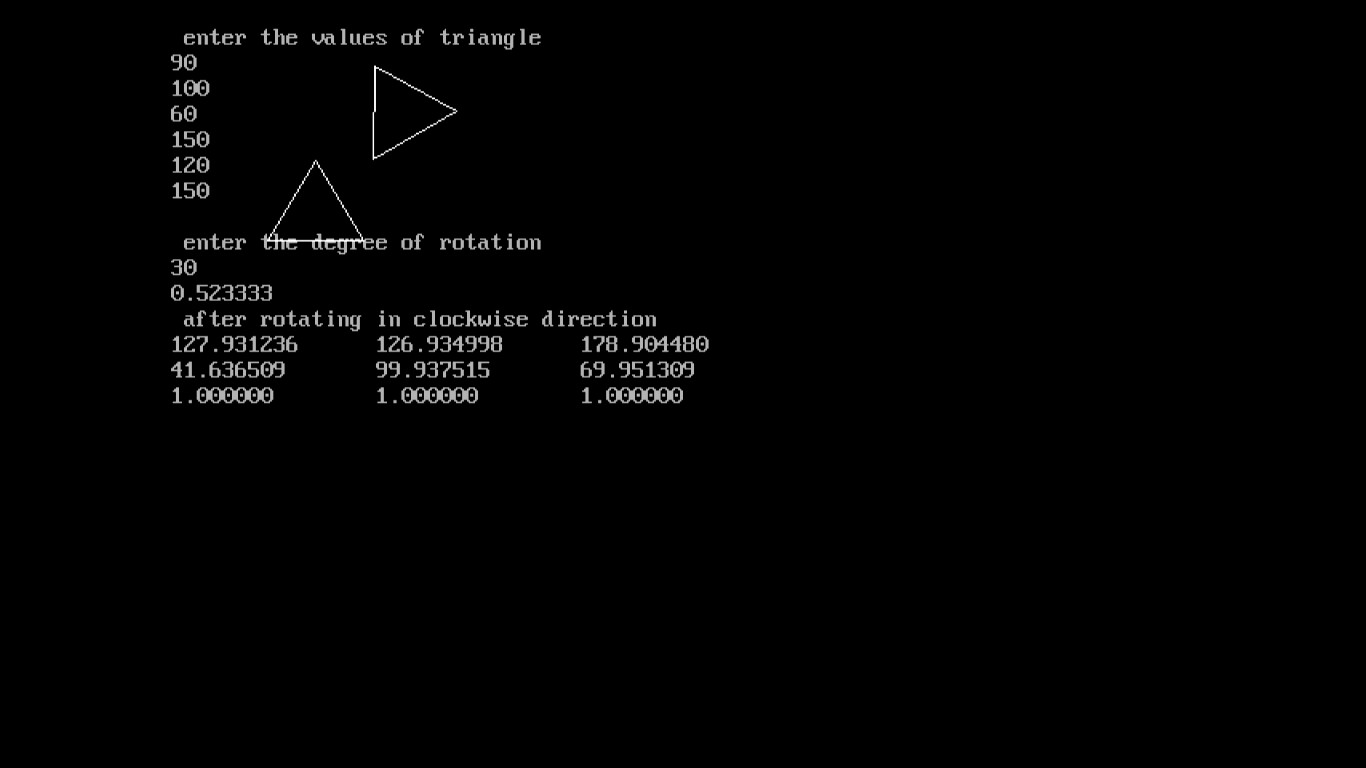
line(t[0][0],t[1][0],t[0][2],t[1][2]);

line(t[0][1],t[1][1],t[0][2],t[1][2]);

getch(); return(0);

}

**Output**



* 1. **Write a program to implement 2D reflection of a triangle.**

#include<stdio.h> #include<conio.h> #include<math.h> #include<graphics.h> int main(void)

{

int a[3][3],b[3][3],c[3][3];

int ch,x1,y1,x2,y2,x3,y3,i,j,k,x,y,p,q; int gdriver=DETECT,gmode; initgraph(&gdriver,&gmode," "); x=getmaxx()/2;

y=getmaxy()/2; p=getmaxx(); q=getmaxy(); line(0,y,p,y);

line(x,0,x,q);

printf("\n enter the points"); scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3); line(x+x1,y-y1,x+x2,y-y2);

line(x+x1,y-y1,x+x3,y-y3);

line(x+x2,y-y2,x+x3,y-y3); b[0][0]=x1;

b[0][1]=x2;

b[0][2]=x3;

b[1][0]=y1;

b[1][1]=y2;

b[1][2]=y3;

b[2][0]=1;

b[2][1]=1;

b[2][2]=1;

printf("\n Press 1:x-axis \n 2:y-axis \n 3:y=x \n 4:y=-x"); printf("\n enter your choice");

scanf("%d",&ch); switch(ch)

{

case 1:

printf("\n reflection about x-axis"); a[0][0]=1;

a[0][1]=0;

a[0][2]=0;

a[1][0]=0;

a[1][1]=-1;

a[1][2]=0;

a[2][0]=0;

a[2][1]=0;

a[2][2]=1;

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{ c[i][j]=0;

for(k=0;k<3;k++)

{

c[i][j]= c[i][j] + a[i][k] \* b[k][j];

}

}

}

line((c[0][0]+x),(y-c[1][0]),(x+c[0][1]),(y-c[1][1]));

line((x+c[0][0]),(y-c[1][0]),(x+c[0][2]),(y-c[1][2]));

line((x+c[0][1]),(y-c[1][1]),(x+c[0][2]),(y-c[1][2]));

break; case 2:

printf("\n reflection about y-axis"); a[0][0]=-1;

a[0][1]=0;

a[0][2]=0;

a[1][0]=0;

a[1][1]=1;

a[1][2]=0;

a[2][0]=0;

a[2][1]=0;

a[2][2]=1;

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{ c[i][j]=0;

for(k=0;k<3;k++)

{

c[i][j]=c[i][j]+a[i][k]\*b[k][j];

}

}

}

line((c[0][0]+x),(y-c[1][0]),(x+c[0][1]),(y-c[1][1]));

line((x+c[0][0]),(y-c[1][0]),(x+c[0][2]),(y-c[1][2]));

line((x+c[0][1]),(y-c[1][1]),(x+c[0][2]),(y-c[1][2]));

break; case 3:

printf("\n reflection about y=x"); a[0][0]=0;

a[0][1]=1;

a[0][2]=0;

a[1][0]=1;

a[1][1]=0;

a[1][2]=0;

a[2][0]=0;

a[2][1]=0;

a[2][2]=1;

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{ c[i][j]=0;

for(k=0;k<3;k++)

{

c[i][j]=c[i][j]+a[i][k]\*b[k][j];

}

}

}

line((c[0][0]+x),(y-c[1][0]),(x+c[0][1]),(y-c[1][1]));

line((x+c[0][0]),(y-c[1][0]),(x+c[0][2]),(y-c[1][2]));

line((x+c[0][1]),(y-c[1][1]),(x+c[0][2]),(y-c[1][2]));

break; case 4:

printf("\n reflection about y=-x"); a[0][0]=0;

a[0][1]=-1;

a[0][2]=0;

a[1][0]=-1;

a[1][1]=0;

a[1][2]=0;

a[2][0]=0;

a[2][1]=0;

a[2][2]=1;

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{ c[i][j]=0;

for(k=0;k<3;k++)

{

c[i][j]=c[i][j]+a[i][k]\*b[k][j];

}

}

}

line((c[0][0]+x),(y-c[1][0]),(x+c[0][1]),(y-c[1][1]));

line((x+c[0][0]),(y-c[1][0]),(x+c[0][2]),(y-c[1][2]));

line((x+c[0][1]),(y-c[1][1]),(x+c[0][2]),(y-c[1][2]));

break;

default:

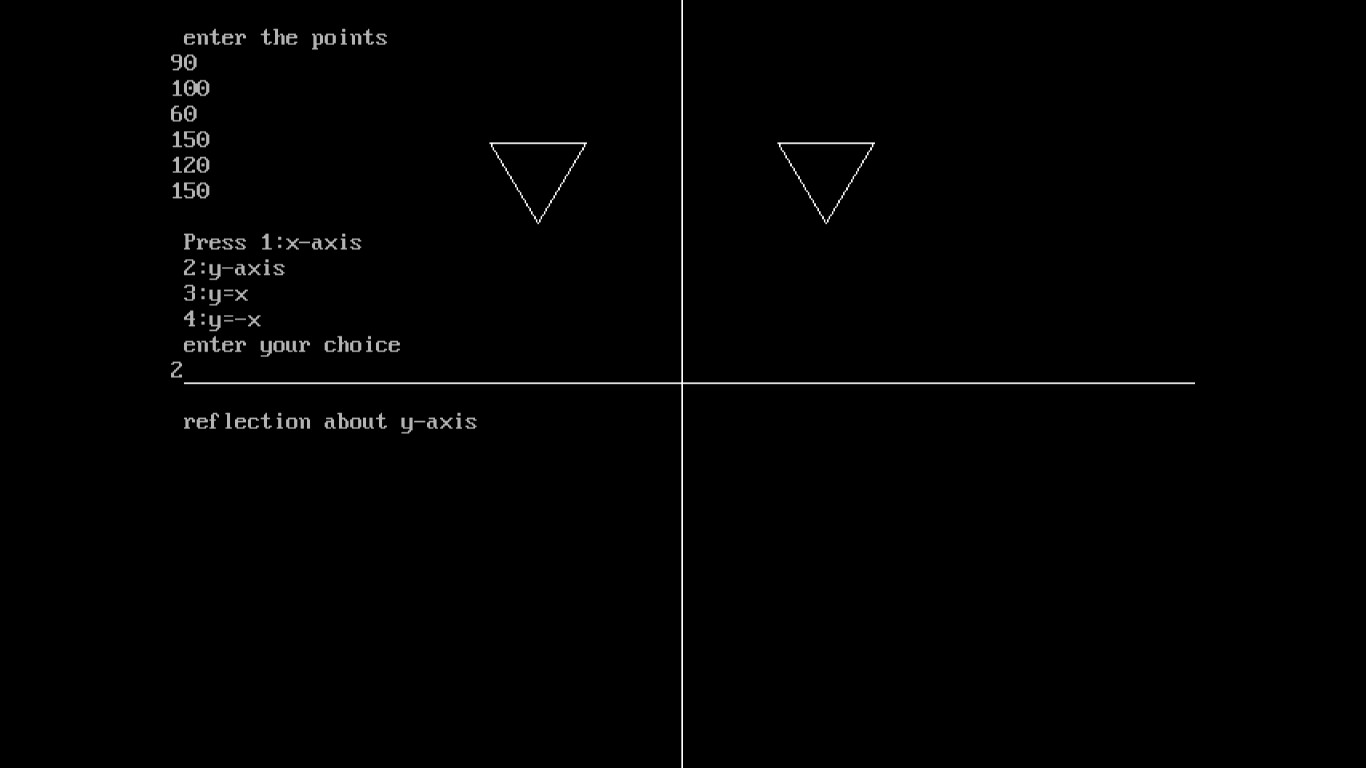
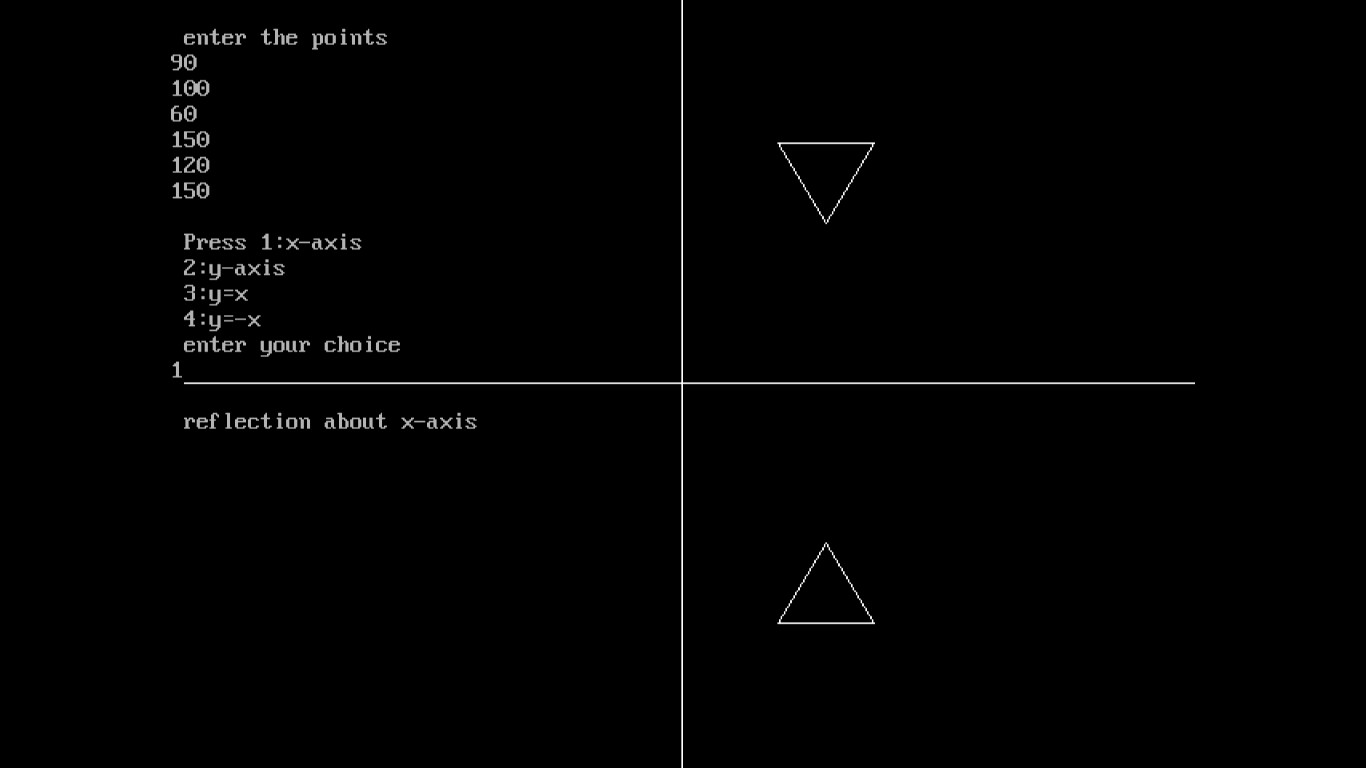
printf("\n wrong choice");

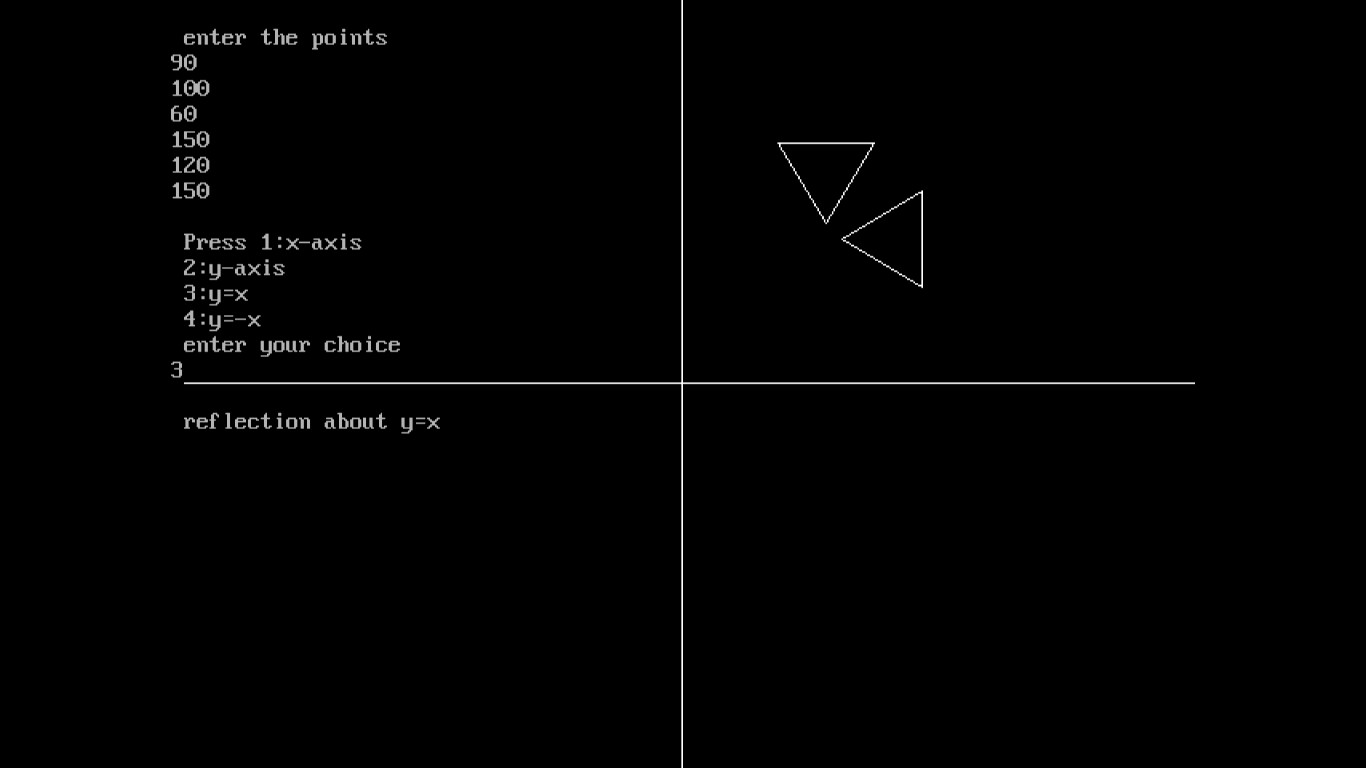
}

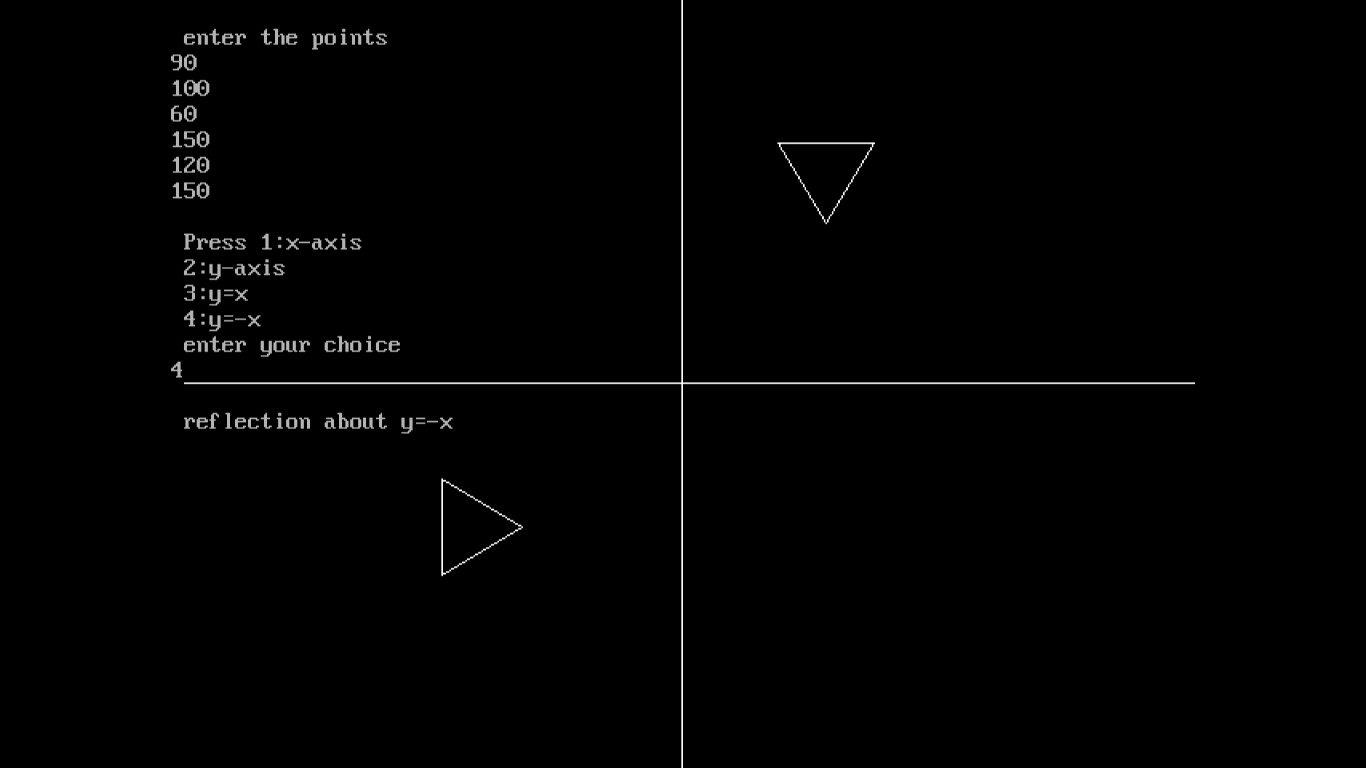
getch(); return(0);

}

**Output**







* 1. **Write a program to draw an ellipse using midpoint ellipse generation algorithm.**

#include<stdio.h> #include<conio.h> #include<math.h> #include<graphics.h> int main(void)

{

int gdriver=DETECT,gmode; float a,b,x,y,p,q,xc,yc;

initgraph(&gdriver,&gmode," ");

printf("\n enter the center points for ellipse"); scanf("%f%f",&xc,&yc);

printf("\n enter the half of major and minor axis"); scanf("%f%f",&a,&b);

x=0;

y=b;

p=(b\*b)+(a\*a)/4-a\*a\*b; while((2\*b\*b\*x)<(2\*a\*a\*y))

{

if(p<0)

{

x=x+1; y=y;

p=p+(2\*b\*b\*x)+b\*b;

}

else

{

x=x+1; y=y-1;

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

}

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

}

q=(b\*b)\*(x+0.5)\*(x+0.5)+((a\*a)\*(y-1)\*(y-1))-(a\*a)\*(b\*b); while(y>=0)

{

if(q<0)

{

x=x+1; y=y-1;

q=q+(2\*b\*b\*x)-(2\*a\*a\*y)+a\*a;

}

else

{

x=x; y=y-1;

q=q+(a\*a)-(2\*a\*a\*y);

}

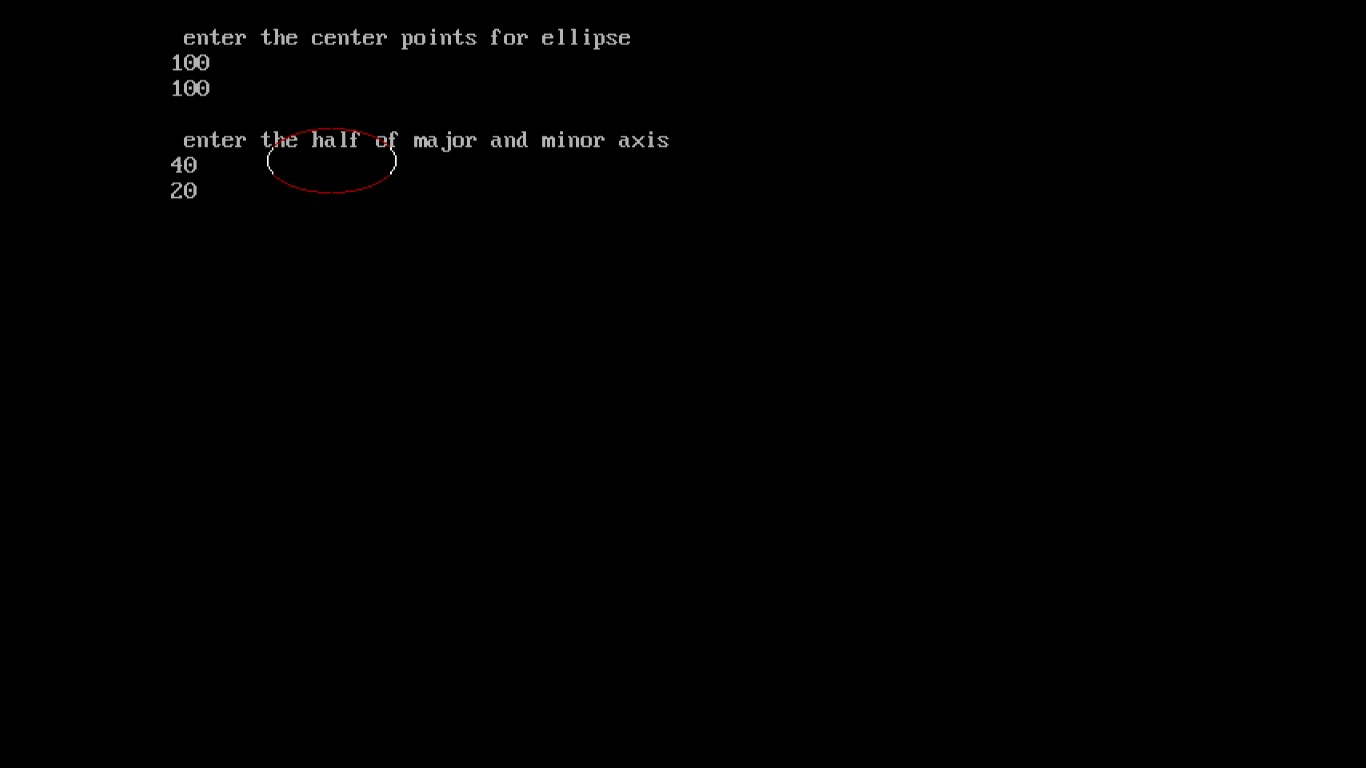
putpixel(xc-x,yc-y,WHITE); putpixel(xc+x,yc-y,WHITE); putpixel(xc+x,yc+y,WHITE); putpixel(xc-x,yc+y,WHITE);

}

getch(); return(0);

}

**Output:**



* 1. **Write a program to implement boundary fill algorithm to fill a triangle.**

#include<stdio.h> #include<conio.h> #include<math.h> #include<graphics.h>

void boundary\_fill(int,int,int,int); void main()

{

int gdriver=DETECT,gmode; int x,y,fill,boundary;

initgraph(&gdriver,&gmode,""); printf("\n enter the values of x and y"); scanf("%d%d",&x,&y);

printf("\n enter the value of fill "); scanf("%d",&fill);

printf("enter the value of boundary"); scanf("%d",&boundary); line(240,60,220,120); line(220,120,260,120); line(240,60,260,120);

boundary\_fill(x,y,fill,boundary); getch();

}

void boundary\_fill(int x,int y,int fill,int boundary)

{

int current=getpixel(x,y); if((current!=boundary)&&(current!=fill))

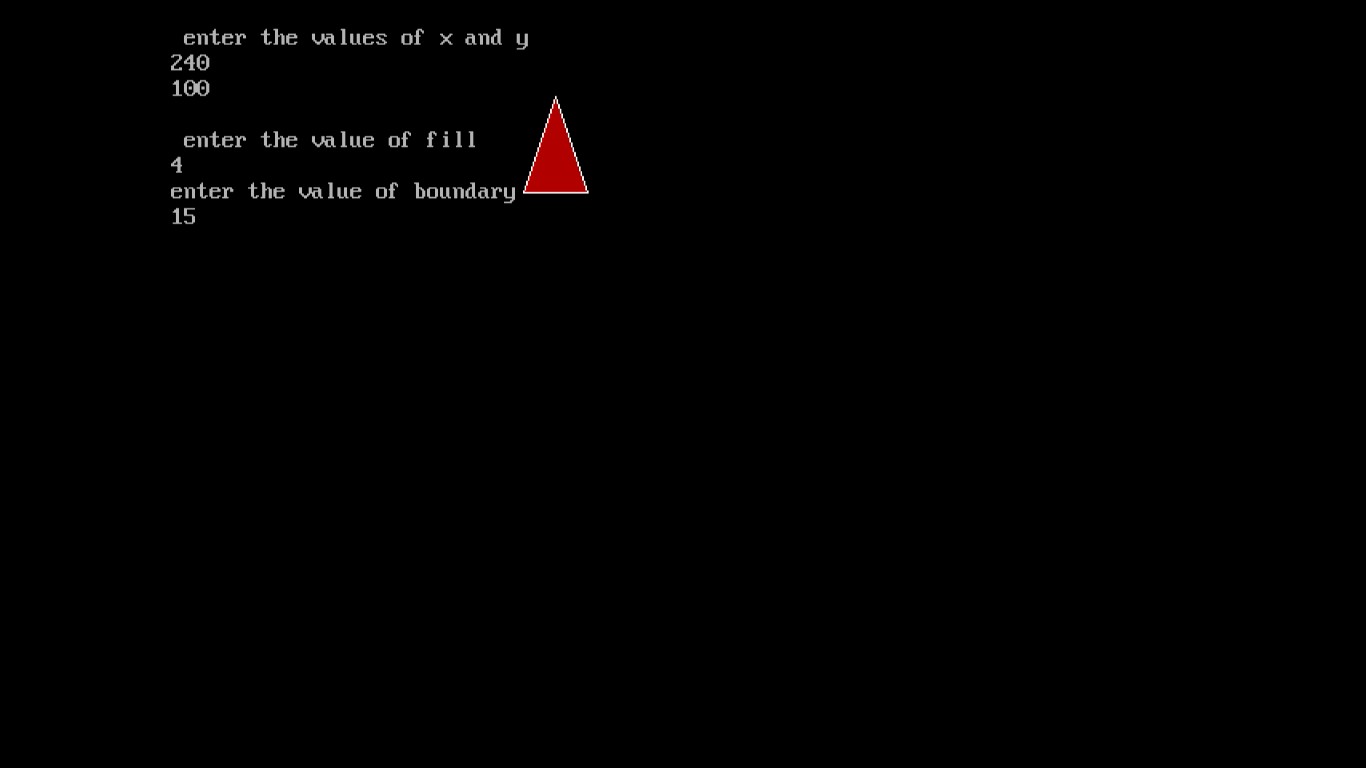
{

putpixel(x,y,fill); boundary\_fill(x+1,y,fill,boundary); boundary\_fill(x-1,y,fill,boundary); boundary\_fill(x,y+1,fill,boundary); boundary\_fill(x,y-1,fill,boundary);

}

}

**Output:**



* 1. **Write a program to implement flood fill algorithm to fill a circle.**

#include<stdio.h> #include<conio.h> #include<math.h> #include<graphics.h>

void flood\_fill(int,int,int,int); void main()

{

int gdriver=DETECT,gmode; int x,y,newcolor,oldcolor;

initgraph(&gdriver,&gmode,""); printf("\n enter the values of x and y"); scanf("%d%d",&x,&y);

printf("\n enter the value of new color "); scanf("%d",&newcolor);

printf("enter the value of old color"); scanf("%d",&oldcolor); circle(200,200,30); flood\_fill(x,y,newcolor,oldcolor); getch();

}

void flood\_fill(int x,int y,int newcolor,int oldcolor)

{

int current=getpixel(x,y); if(current==oldcolor)

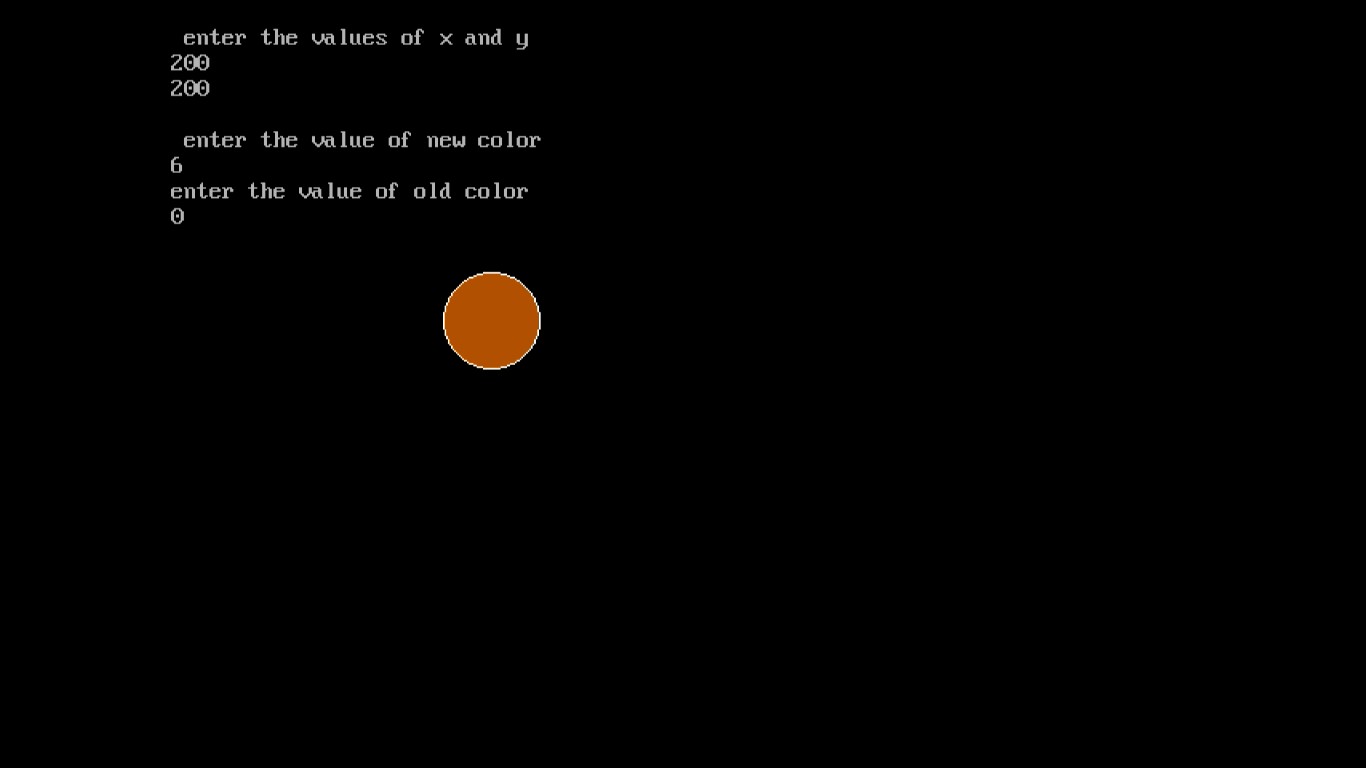
{

putpixel(x,y,newcolor); flood\_fill(x+1,y,newcolor,oldcolor); flood\_fill(x-1,y,newcolor,oldcolor); flood\_fill(x,y+1,newcolor,oldcolor); flood\_fill(x,y-1,newcolor,oldcolor);

}

}

**Output:**



* 1. **Write a program to implement Liang-Barsky line clipping algorithm.**

#include<stdlib.h> #include<conio.h> #include<dos.h> #include<graphics.h> void main()

{

int gdriver=DETECT,gmode; int x1,x2,y1,y2;

int wxmin,wymin,wxmax,wymax; float u1=0.0,u2=1.0;

int p1,q1,p2,q2,p3,q3,p4,q4; float r1,r2,r3,r4;

int x11,y11,x22,y22; clrscr();

initgraph(&gdriver,&gmode,"");

printf("\n enter the windows left xmin , top boundary ymin"); scanf("%d%d",&wxmin,&wymin);

printf("\n enter the windows right xmax , bottom boundary ymax"); scanf("%d%d",&wxmax,&wymax);

printf("\n enter line: x1,y1 coordinates:\n"); scanf("%d%d",&x1,&y1);

printf("\n enter line: x2,y2 coordinates:\n"); scanf("%d%d",&x2,&y2);

printf("\n liang barsky express these 4 inequalities using lpk=qr"); p1=-(x2-x1);

q1=x1-wxmin; p2=(x2-x1);

q2=wxmax-x1; p3=-(y2-y1);

q3=y1-wymin; p4=(y2-y1);

q4=wymax-y1;

printf("\n p1=0 line is parallel to left clipping \n"); printf("\n p2=0 line is parallel to right clipping \n"); printf("\n p3=0 line is parallel to bottom clipping \n"); printf("\n p4=0 line is parallel to top clipping \n");

if(((p1==0.0) && (q1<0.0)) || ((p2==0.0) && (q2<0.0)) || ((p3==0.0)&&(q3<0.0)) || ((p4==0.0)&&(q4<0.0)))

{

printf("\n line is rejected:"); getch(); detectgraph(&gdriver,&gmode); initgraph(&gdriver,&gmode," "); setcolor(RED); rectangle(wxmin,wymax,wxmax,wymin);

setcolor(BLUE); line(x1,y1,x2,y2); getch(); setcolor(WHITE); line(x1,y1,x2,y2); getch();

}

else

{ if(p1!=0.0)

{

r1=(float)q1/p1; if(p1<0) u1=max(r1,u1); else u2=min(r1,u2);

} if(p2!=0.0)

{

r2=(float)q2/p2; if(p2<0) u1=max(r2,u1); else u2=min(r2,u2);

} if(p3!=0.0)

{

r3=(float)q3/p3; if(p3<0) u1=max(r3,u1); else u2=min(r3,u2);

} if(p4!=0.0)

{

r4=(float)q4/p4; if(p4<0) u1=max(r4,u1); else u2=min(r4,u2);

}

if(u1>u2)

printf("\n line rejected:"); else

{

x11=x1+u1\*(x2-x1); y11=y1+u1\*(y2-y1);

x22=x1+u2\*(x2-x1); y22=y1+u2\*(y2-y1);

printf("\n original line coordinates\n"); printf("\n x1=%d y1=%d x2=%d y2=%d\n",x1,y1,x2,y2); printf("\n windows coordinates are:\n");

printf("\n wxmin=%d,wymin=%d,wxmax=%d,wymax=%d",wxmin,wymin,wxmax,wymax); printf("\n new coordinates are ;\n");

printf("\n x1=%d,y1=%d,x2=%d,y2=%d\n",x11,y11,x22,y22); detectgraph(&gdriver,&gmode); initgraph(&gdriver,&gmode,"");

setcolor(2); rectangle(wxmin,wymax,wxmax,wymin); setcolor(1);

line(x1,y1,x2,y2); getch(); setcolor(0); line(x1,y1,x2,y2); setcolor(3);

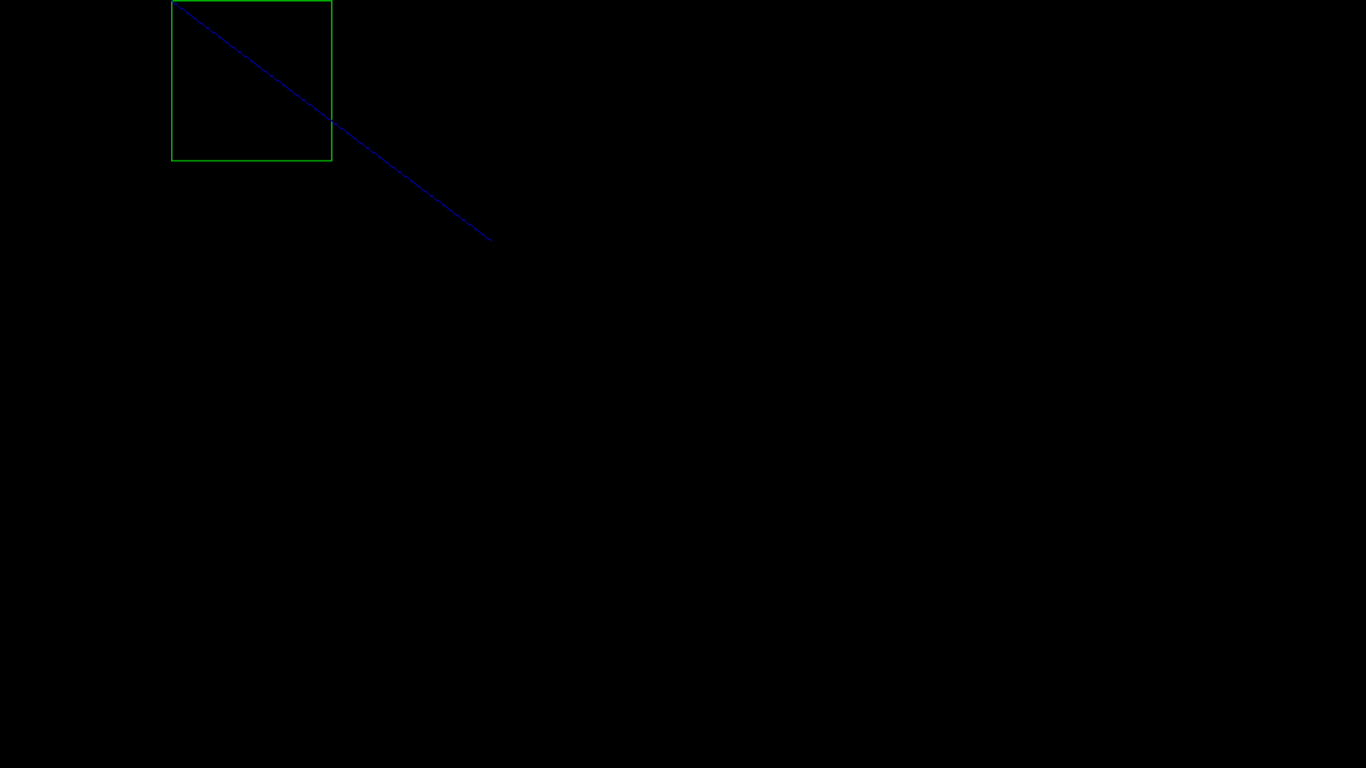
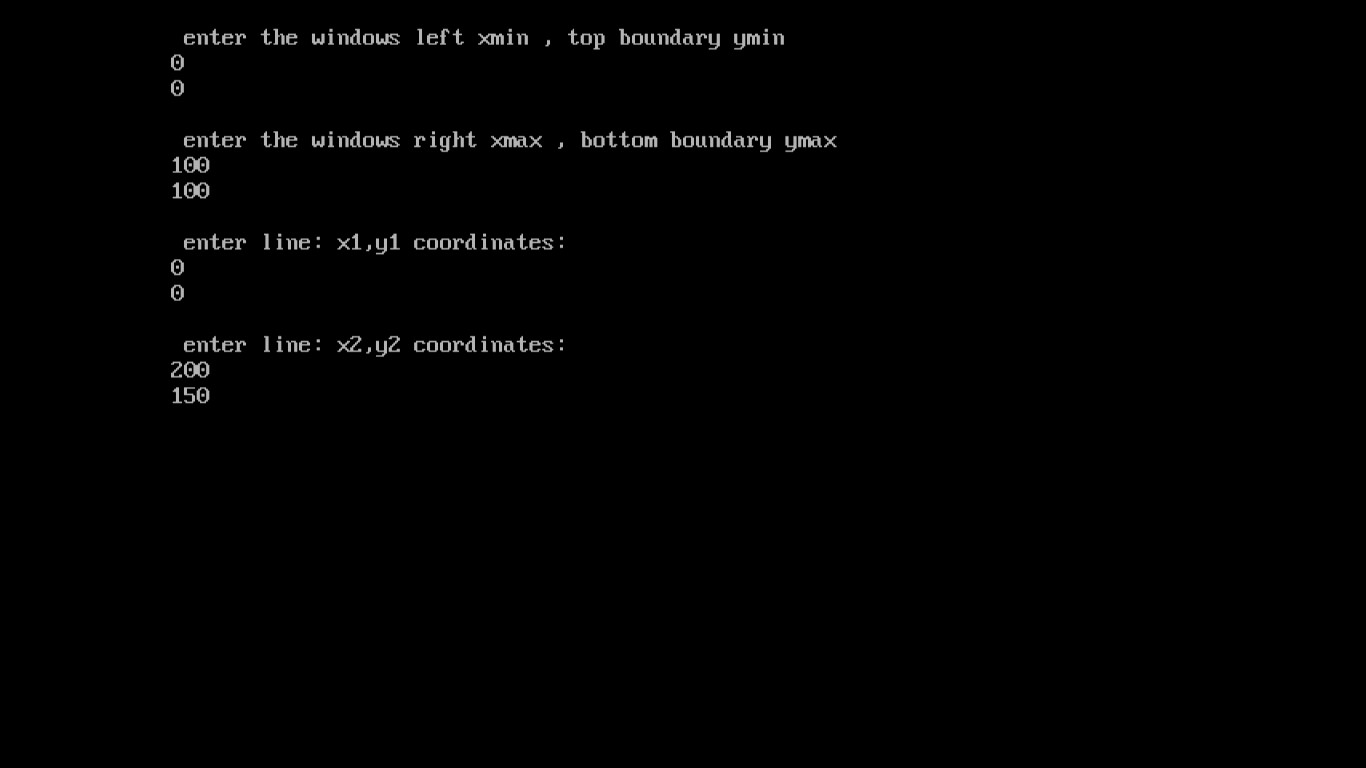
line(x11,y11,x22,y22); getch();

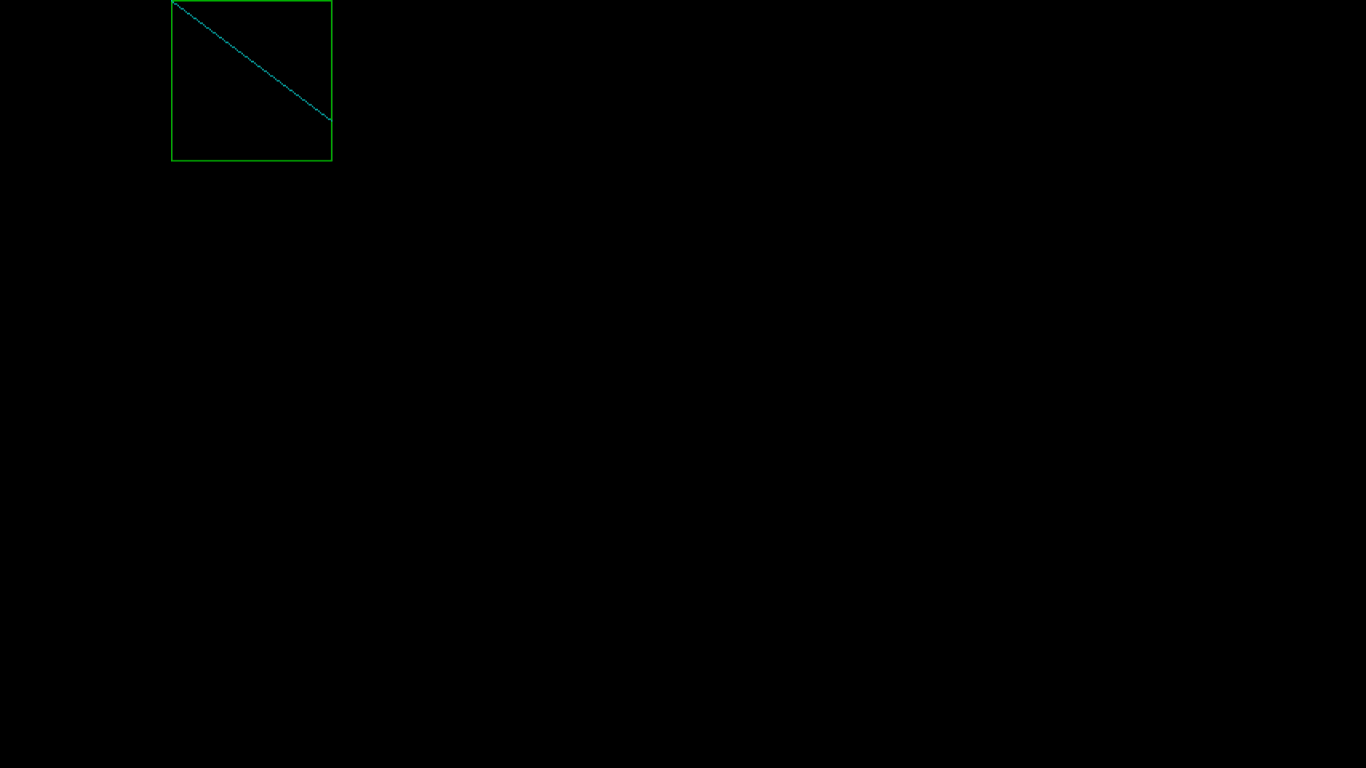
}

}

}

**Output:**





* 1. **Write a program to implement oblique projection.**

#include<stdio.h> #include<conio.h> #include<math.h> #include<graphics.h> void main()

{

int x1,y1,x2,y2;

int gdriver=DETECT,gmode; int ymax,a[4][8];

float par[4][4],b[4][8]; int i,j,k,m,n,p;

double L1,phi;

a[0][0]=100;

a[1][0]=100;

a[2][0]=100;

a[0][1]=200;

a[1][1]=100;

a[2][1]=100;

a[0][2]=200;

a[1][2]=200;

a[2][2]=100;

a[0][3]=100;

a[1][3]=200;

a[2][3]=100;

a[0][4]=100;

a[1][4]=100;

a[2][4]=200;

a[0][5]=200;

a[1][5]=100;

a[2][5]=200;

a[0][6]=200;

a[1][6]=200;

a[2][6]=200;

a[0][7]=100;

a[1][7]=200;

a[2][7]=200;

phi=(double)(3.14\*45.0)/180; L1=0.5;

par[0][0]=1;

par[0][1]=0;

par[0][2]=L1\*cos(phi);

par[0][3]=0;

par[1][0]=0;

par[1][1]=1;

par[1][2]=L1\*sin(phi);

par[1][3]=0;

par[2][0]=0;

par[2][1]=0;

par[2][2]=0;

par[2][3]=0;

par[3][0]=0;

par[3][1]=0;

par[3][2]=0;

par[3][3]=1; m=4;

n=4; p=8;

for(i=0;i<n;i++) for(k=0;k<p;k++) b[i][k]=0; for(i=0;i<m;i++) for(k=0;k<p;k++) for(j=0;j<n;j++)

b[i][k]+=(float)par[i][j]\*a[j][k]; detectgraph(&gdriver,&gmode); initgraph(&gdriver,&gmode,"c:\\tc\\bgi"); ymax=getmaxy();

/\*- front plane display -\*/ for(j=0;j<3;j++)

{

x1=(int) b[0][j];

y1=(int) b[1][j];

x2=(int) b[0][j+1];

y2=(int) b[1][j+1]; line(x1,ymax-y1,x2,ymax-y2);

}

x1=(int) b[0][3];

y1=(int) b[1][3];

x2=(int) b[0][0];

y2=(int) b[1][0]; line(x1,ymax-y1,x2,ymax-y2);

/\*- back plane display -\*/ setcolor(11); for(j=4;j<7;j++)

{

x1=(int) b[0][j];

y1=(int) b[1][j];

x2=(int) b[0][j+1];

y2=(int) b[1][j+1]; line(x1,ymax-y1,x2,ymax-y2);

}

x1=(int) b[0][7];

y1=(int) b[1][7];

x2=(int) b[0][4];

y2=(int) b[1][4]; line(x1,ymax-y1,x2,ymax-y2); setcolor(13); for(i=0;i<4;i++)

{

x1=(int) b[0][i];

y1=(int) b[1][i];

x2=(int) b[0][4+i];

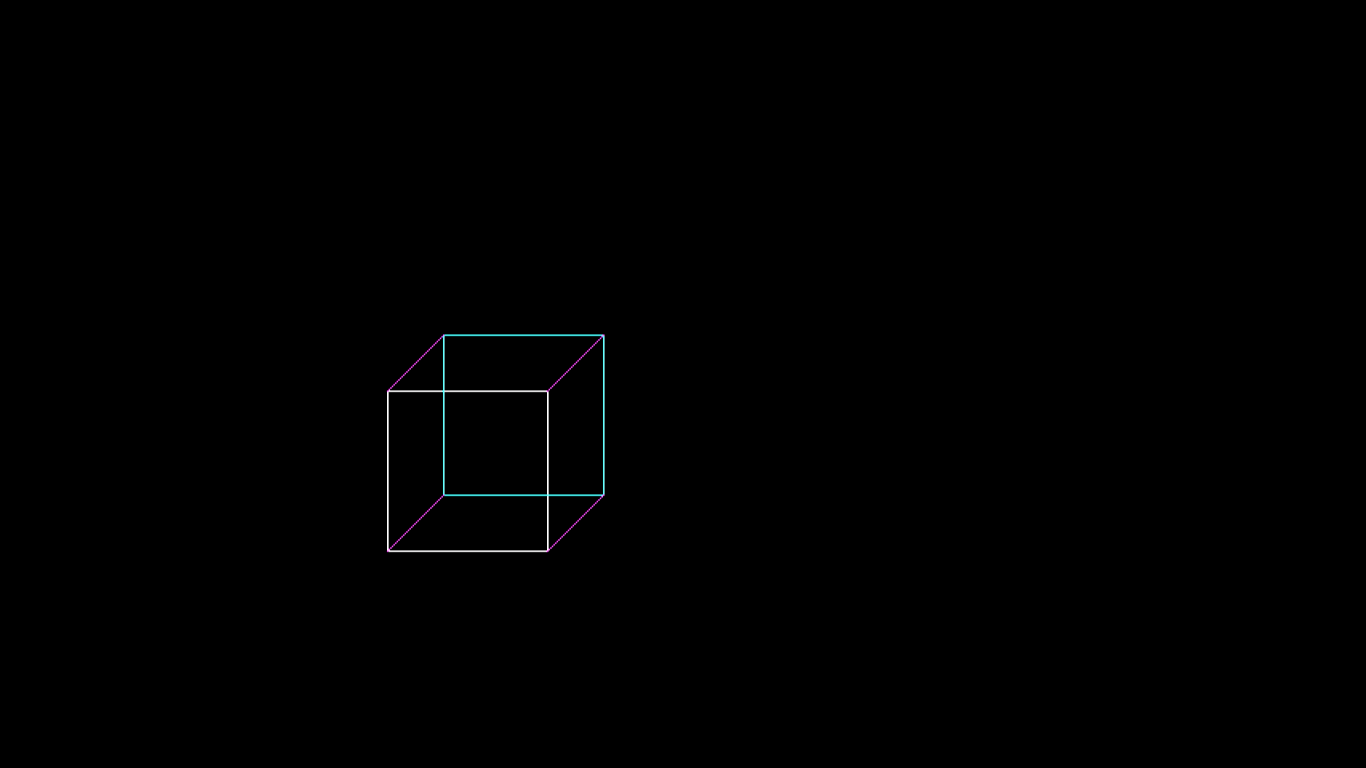
y2=(int) b[1][4+i]; line(x1,ymax-y1,x2,ymax-y2);

}

getch();

}

**Output:**



* 1. **Write a program to implement perspective projection.**

#include<stdio.h> #include<conio.h> #include<graphics.h> void main()

{

int x1,y1,x2,y2;

int gdriver=DETECT,gmode; int ymax,a[4][8];

float par[4][4],b[4][8]; int i,j,k,m,n,p;

int xp,yp,zp,x,y,z;

a[0][0]=100;

a[1][0]=100;

a[2][0]=-100;

a[0][1]=200;

a[1][1]=100;

a[2][1]=-100;

a[0][2]=200;

a[1][2]=200;

a[2][2]=-100;

a[0][3]=100;

a[1][3]=200;

a[2][3]=-100;

a[0][4]=100;

a[1][4]=100;

a[2][4]=-200;

a[0][5]=200;

a[1][5]=100;

a[2][5]=-200;

a[0][6]=200;

a[1][6]=200;

a[2][6]=-200;

a[0][7]=100;

a[1][7]=200;

a[2][7]=-200;

detectgraph(&gdriver,&gmode); initgraph(&gdriver,&gmode,"c:\\tc\\bgi"); ymax=getmaxy();

xp=300; yp=320; zp=100;

for(j=0;j<8;j++)

{ x=a[0][j];

y=a[1][j];

z=a[2][j];

b[0][j]=xp-((float)(x-xp)/(z-zp))\*(zp);

b[1][j]=yp-((float)(y-yp)/(z-zp))\*(zp);

}

/\* front plane display\*/ for(j=0;j<3;j++)

{

x1=(int)b[0][j];

y1=(int)b[1][j];

x2=(int)b[0][j+1];

y2=(int)b[1][j+1]; line(x1,ymax-y1,x2,ymax-y2);

}

x1=(int)b[0][3];

y1=(int)b[1][3];

x2=(int)b[0][0];

y2=(int)b[1][0]; line(x1,ymax-y1,x2,ymax-y2);

/\* back plane display \*/ setcolor(11); for(j=4;j<7;j++)

{

x1=(int)b[0][j];

y1=(int)b[1][j];

x2=(int)b[0][j+1];

y2=(int)b[1][j+1]; line(x1,ymax-y1,x2,ymax-y2);

}

x1=(int)b[0][7];

y1=(int)b[1][7];

x2=(int)b[0][4];

y2=(int)b[1][4]; line(x1,ymax-y1,x2,ymax-y2); setcolor(7); for(i=0;i<4;i++)

{

x1=(int)b[0][i];

y1=(int)b[1][i];

x2=(int)b[0][4+i];

y2=(int)b[1][4+i]; line(x1,ymax-y1,x2,ymax-y2);

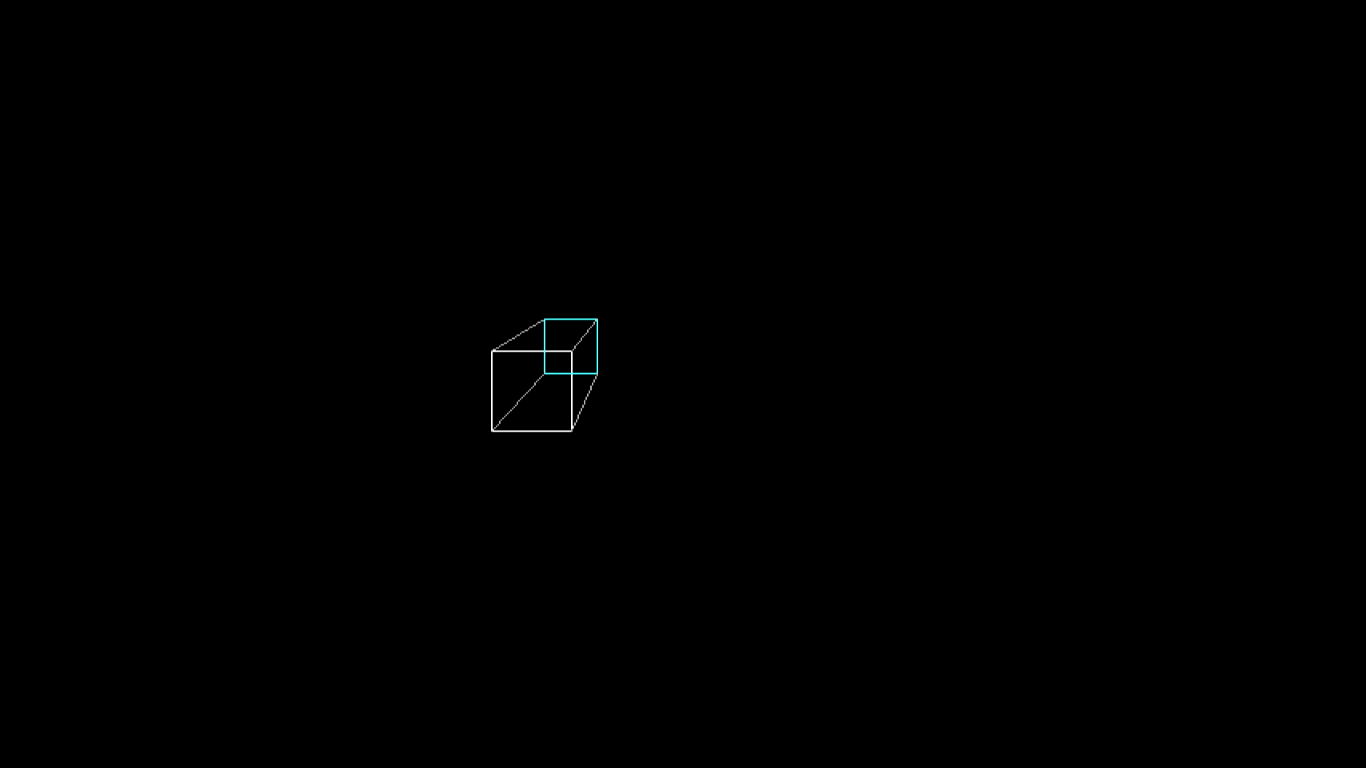
}

getch();

getch();

}

**Output:**



* 1. **Write a program to draw the Bezier curve .**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

int x,y,z;

void main()

{

float u;

int gd,gm,ymax,i,n,c[4][3];

for(i=0;i<4;i++) { c[i][0]=0; c[i][1]=0; }

printf("\n\n Enter four points : \n\n");

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

{

printf("\t X%d Y%d : ",i,i);

scanf("%d %d",&c[i][0],&c[i][1]);

}

c[4][0]=c[0][0];

c[4][1]=c[0][1];

detectgraph(&gd,&gm);

initgraph(&gd,&gm,"e:\\tc\\bgi");

ymax = 480;

setcolor(13);

for(i=0;i<3;i++)

{

line(c[i][0],ymax-c[i][1],c[i+1][0],ymax-c[i+1][1]);

}

setcolor(3);

n=3;

for(i=0;i<=40;i++)

{

u=(float)i/40.0;

bezier(u,n,c);

if(i==0)

{ moveto(x,ymax-y);}

else

{ lineto(x,ymax-y); }

getch();

}

getch();

}

bezier(u,n,p)

float u;int n; int p[4][3];

{

int j;

float v,b;

float blend(int,int,float);

x=0;y=0;z=0;

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

{

b=blend(j,n,u);

x=x+(p[j][0]\*b);

y=y+(p[j][1]\*b);

z=z+(p[j][2]\*b);

}

}

float blend(int j,int n,float u)

{

int k;

float v,blend;

v=C(n,j);

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

{ v\*=u; }

for(k=1;k<=(n-j);k++)

{ v \*= (1-u); }

blend=v;

return(blend);

}

C(int n,int j)

{

int k,a,c;

a=1;

for(k=j+1;k<=n;k++) { a\*=k; }

for(k=1;k<=(n-j);k++) { a=a/k; }

c=a;

return(c);

}