**Program No:**

**Aim:** Write a program to draw a line using Bresenham’s line drawing algorithm.

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void bsline(int x,int y,int x2,int y2)

{

int dx,dy,p;

dx=x2-x;

dy=y2-y;

p=2\*(dy)-(dx);

while(x <=x2)

{

if(p<0)

{

x=x+1;

y=y;

p=p+2\*(dy);

}

else

{

x=x+1;

y=y+1;

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

}

putpixel(x,y,RED);

delay(10);

}

}

void main()

{

int gd=DETECT,gm;

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

int x1,x2,y1,y2;

cout<<"Enter the x1,y1,x2,y2 values:";

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

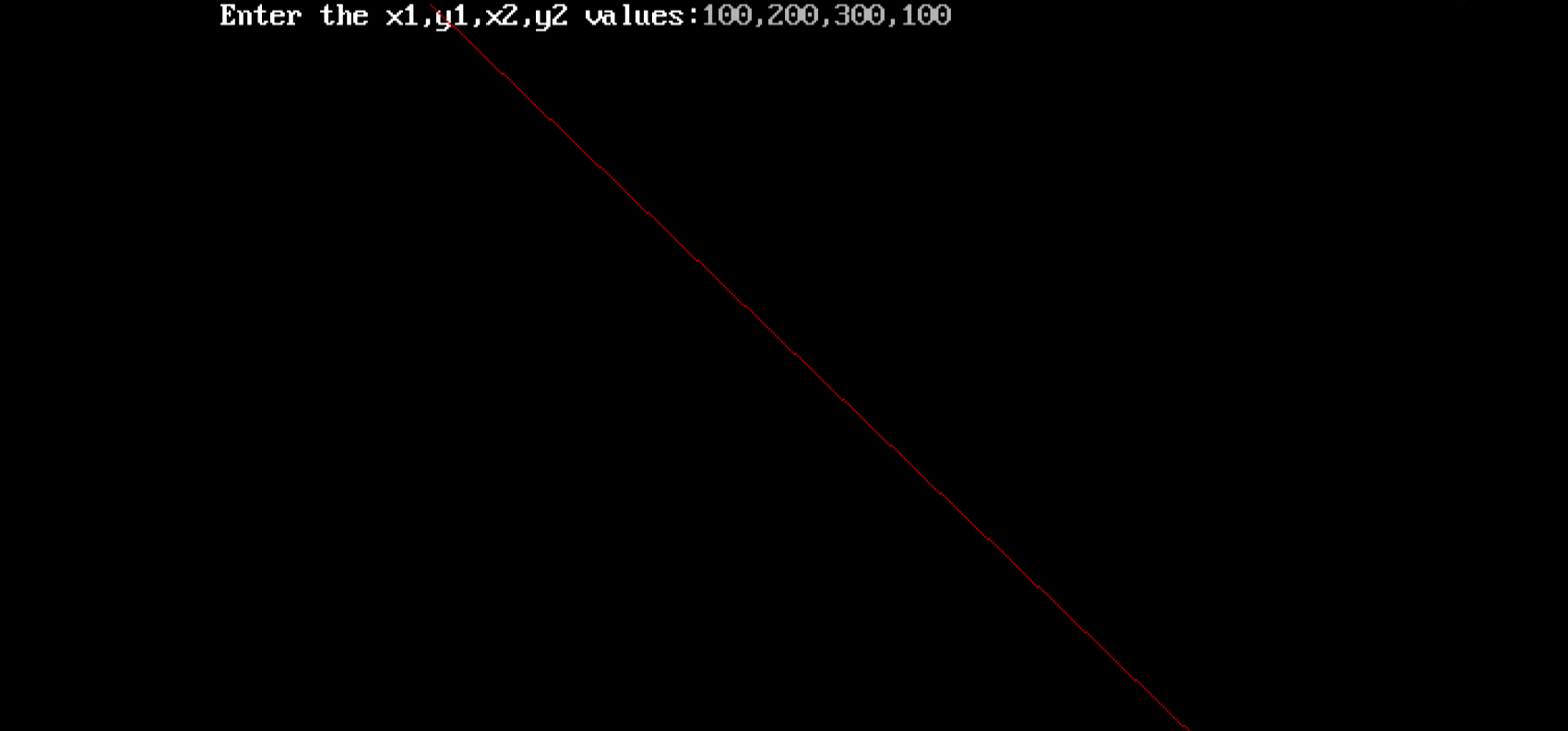
bsline(x1,y1,x2,y2);

getch();

closegraph();

}

**Output:**



**Program No**

**Aim:** Write a program to draw a line using DDA line drawing algorithm.

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

void main()

{

float x,y,x1,y1,x2,y2,dx,dy,step;

int i,gd=DETECT,gm;

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

cout<<"Enter the value of x1 and y1:";

cin>>x1>>y1;

cout<<"Ente the value of x2 and y2:";

cin>>x2>>y2;

dx=abs(x2-x1);

dy=abs(y2-y1);

if(dx>=dy)

step=dx;

else

step=dy;

dx=dx/step;

dy=dy/step;

x=x1;

y=y1;

i=1;

while(i<=step)

{

putpixel(x,y,5);

x=x+dx;

y=y+dy;

i=i+1;

delay(100);

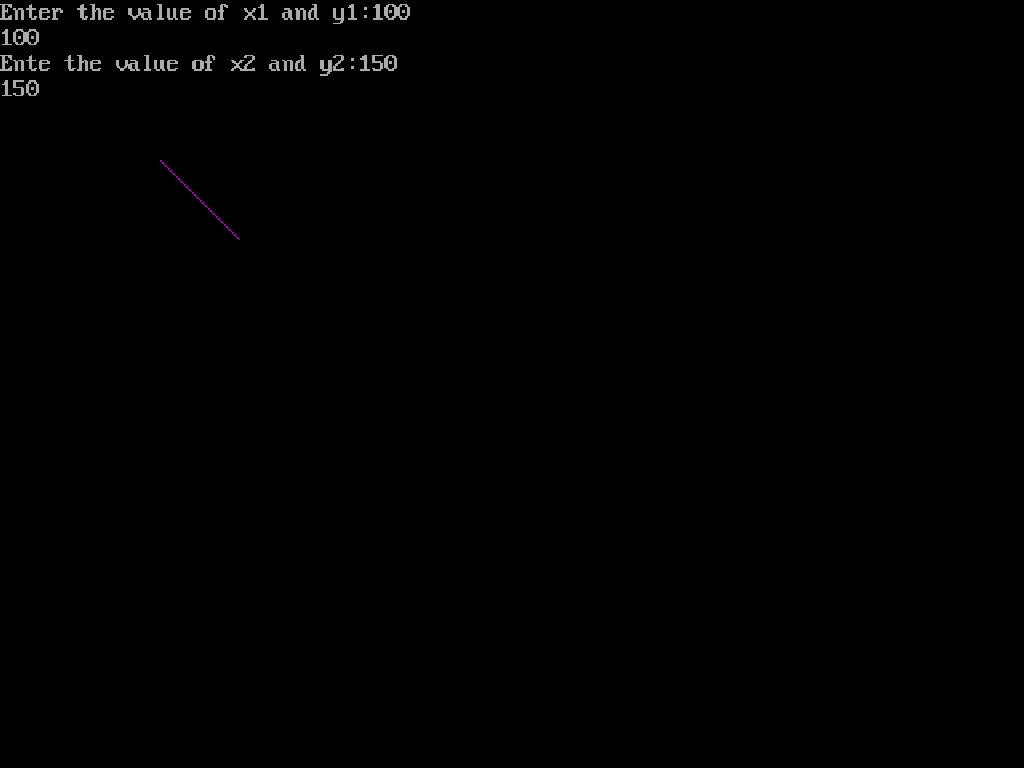
}

closegraph();

getch();

}

**Output:**



**Program No:**

**/\*Aim:** Write a program to draw a circle using ARCDDA algorithm.

**\*/**

#include<graphics.h>

#include<dos.h>

#include<conio.h>

#include<iostream.h>

#include<math.h>

void delay(int milliseconds)

{

int delay = 1000 \* milliseconds / 50;

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

{

asm nop;

}

}

Void main()

{

int gd = DETECT, gm;

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

float x1, y1, x2, y2, x, y, k;

clrscr();

cout<<"Enter the radius of circle ";

cin>>r;

x1 = r \* cos(0);

y1 = r \* sin(0);

x = x1;

y = y1;

i = 0;

do

{

val = pow(2, i);

i++;

} while (val < r);

k = 1 / pow(2, i - 1);

do

{

x2 = x1 + y1 \* k;

y2 = y1 - k \* x2;

putpixel(200 + x2, 200 + y2, 15);

x1 = x2;

y1 = y2;

} while ((y1 - y) < k || (x - x1) > k);

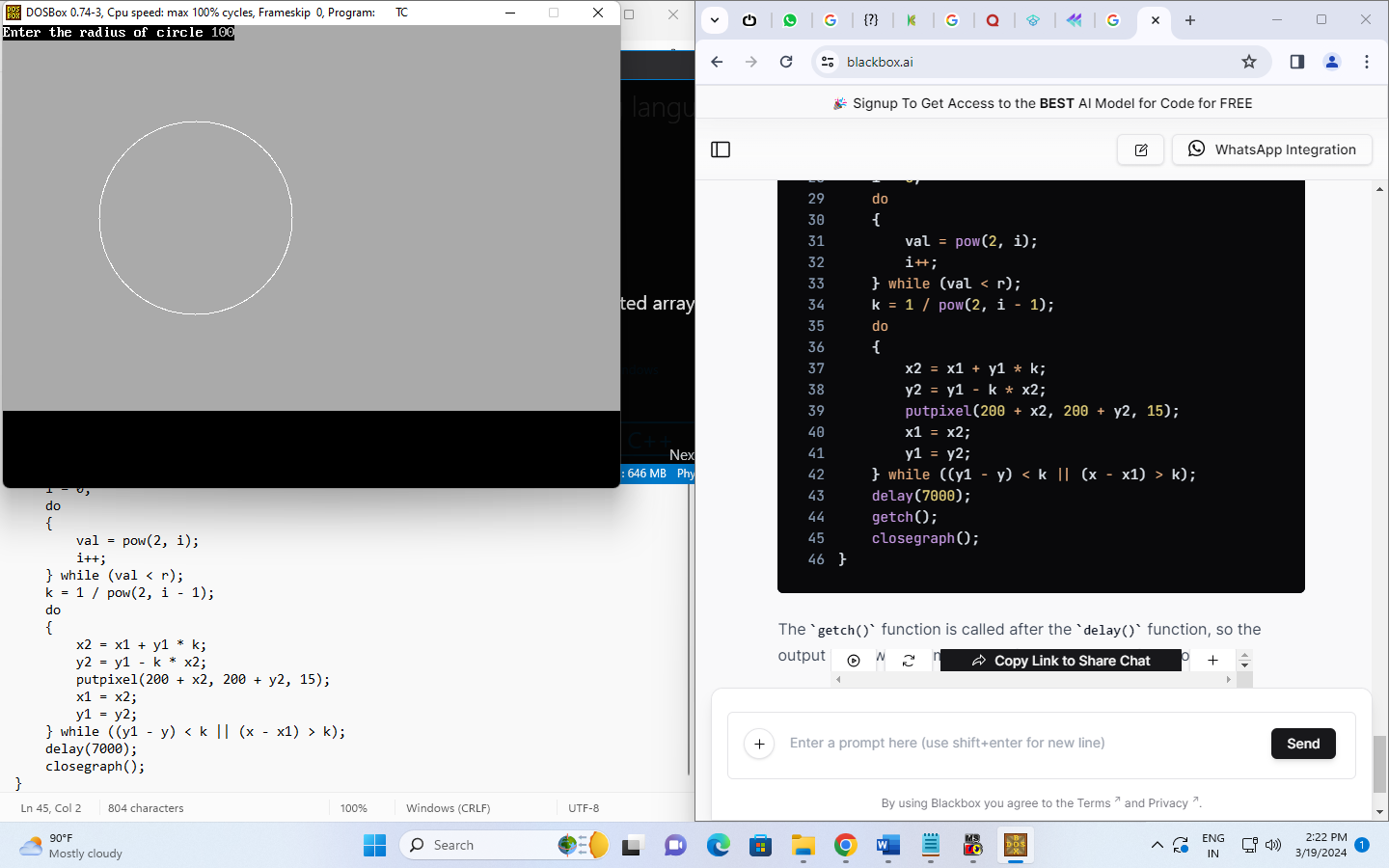
delay(8000);

getch();

closegraph();

}

**Output:**



**Program No:**

**Aim:** Write a program to draw a rectangle.

#include<graphics.h>

#include<iostream.h>

#include<conio.h>

#include<math.h>

int main()

{

int gd=DETECT,gm;

int left=150, top=150;

int right=450,bottom=450;

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

rectangle(left, top, right, bottom);

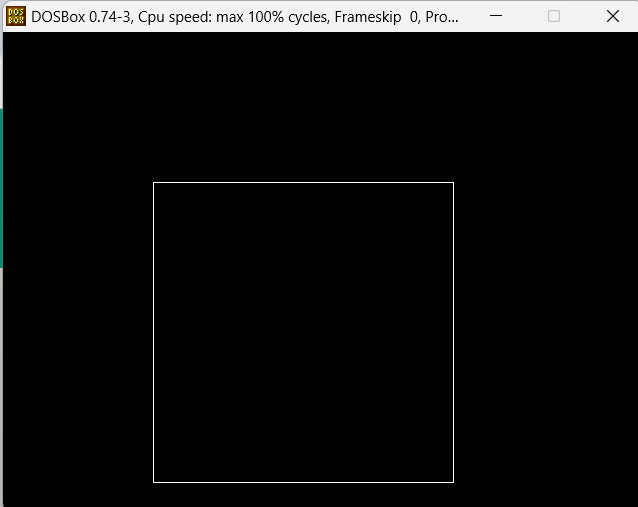
getch();

closegraph();

return 0;

}

**Output:**



**Program No:**

**Aim:** write a program to draw a ellipse using ARCDDA algorithm.

#include<graphics.h>

#include<iostream.h>

#include<conio.h>

int main()

{

int gd = DETECT, gm;

int x = 250, y = 200;

int start\_angle = 0;

int end\_angle = 360;

int x\_rad = 100;

int y\_rad = 50;

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

ellipse(x, y, start\_angle, end\_angle, x\_rad, y\_rad);

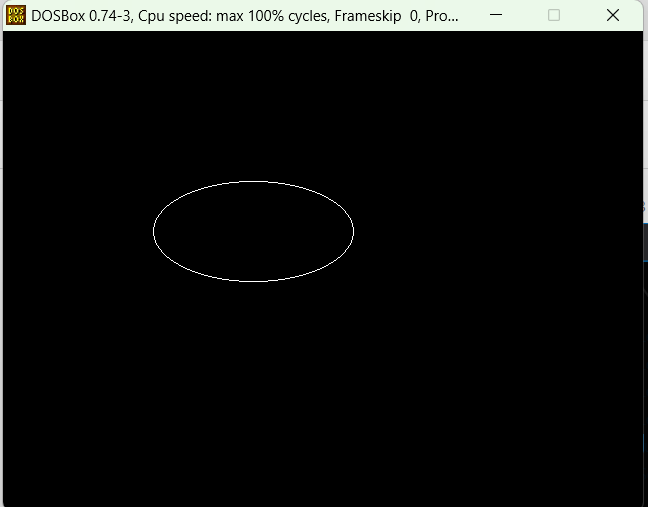
getch();

closegraph();

return 0;

}

**Output:**



**Program No:**

**Aim:** Write a program to rotate an object about origin.

#include<math.h>

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

clrscr();

int gd=DETECT,gm,x1,x2,y1,y2,x4,y4;

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

float angle=0,ang;

cout<<"\nROTATION OF A LINE";

cout<<"\nENTER THE FIRST COORDINATE OF A LINE:";

cin>>x1>>y1;

cout<<"\nEnter the second coordinates of a line:";

cin>>x2>>y2;

line(x1,y1,x2,y2);

cout<<"Enter the angle:";

cin>>ang;

angle=(ang\*3.14)/180;

setcolor(RED);

x4=x2-(((x2-x1)\*cos(angle))-((y2-y1)+sin(angle)));

y4=y2-(((x2-x1)\*sin(angle))+((y2-y1)\*cos(angle)));

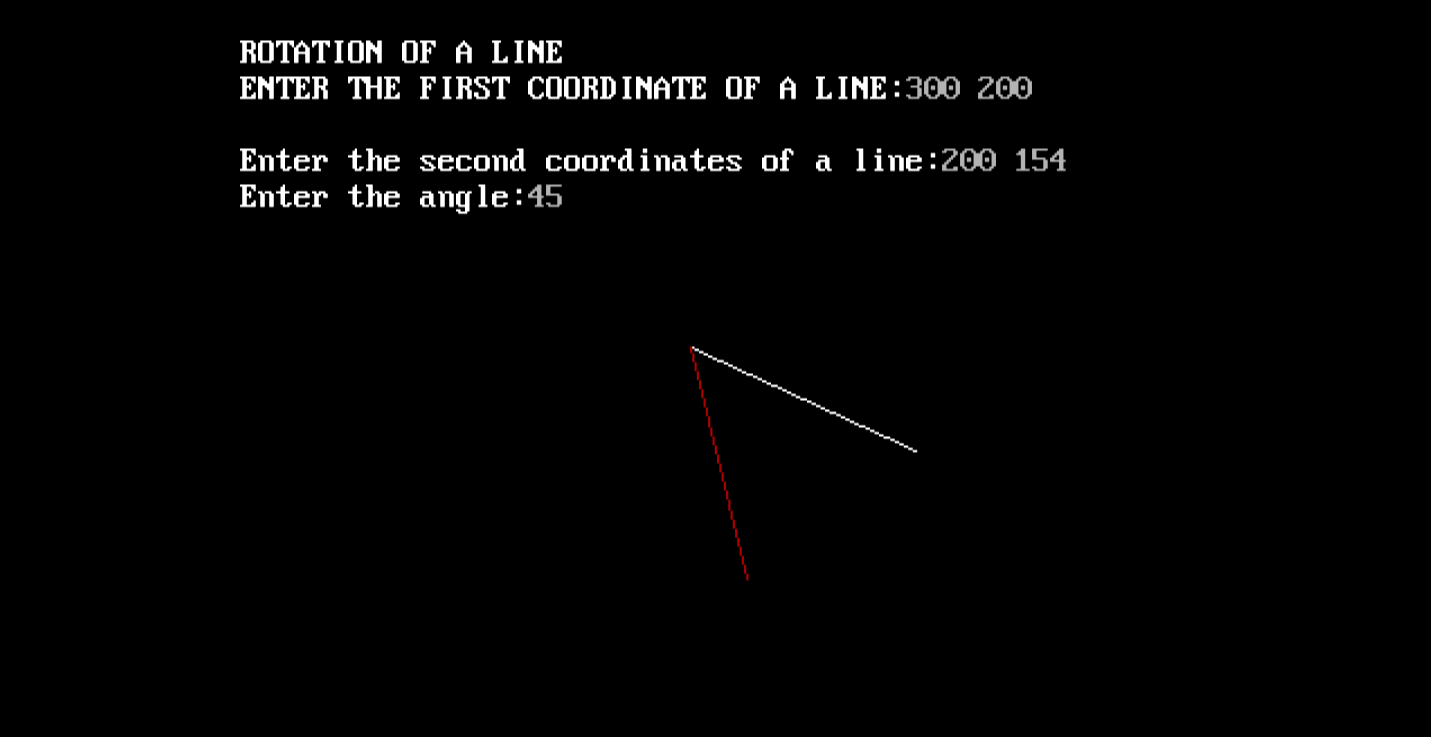
line(x2,y2,x4,y4);

getch();

closegraph();

}

**Output:**



**Program No:**

**Aim:** Write a program to rotate a object about any arbitrary point.

#include<math.h>

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

clrscr();

int gd=DETECT,gm,x1,x2,y1,y2,x4,y4;

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

float angle=0,ang;

cout<<"\nRotation of a Line";

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

cin>>x1>>y1;

cout<<"\nEnter the second coordinate of line:";

cin>>x2>>y2;

line(x1,y1,x2,y2);

cout<<"Enter the angle:";

cin>>ang;

angle=(ang\*3.14)/180;

setcolor(RED);

x4=x2-(((x2-x1)\*cos(angle))-((y2-y1)+sin(angle)));

y4=y2-(((x2-x1)\*sin(angle))+((y2-y1)\*cos(angle)));

line(x2,y2,x4,y4);

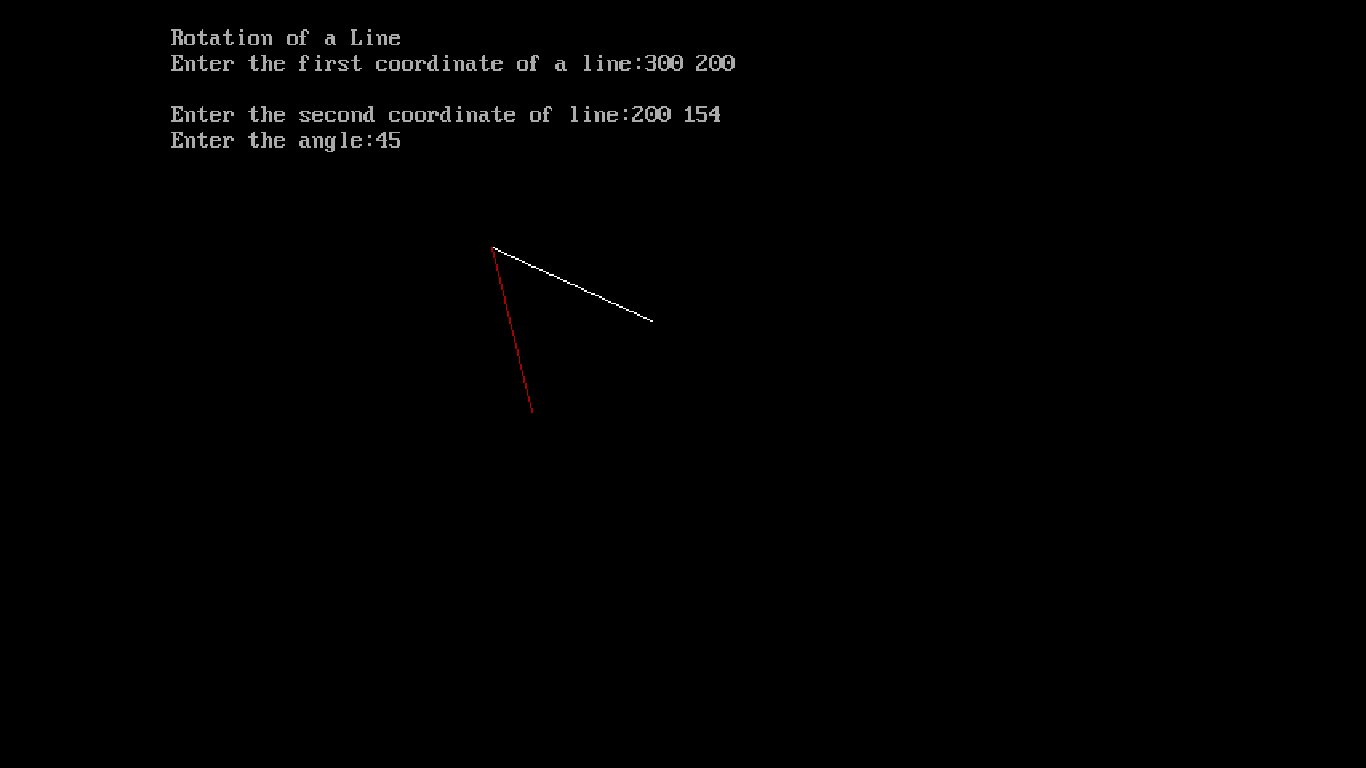
getch();

closegraph();

return 0;

}

**Output:**



**Program No.**

**/\* Aim :** Write a program to implement 2d Scaling transformation.

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int x1,x2,x3,x4,x5,x6,y1,y2,y3,y4,y5,y6,sx,sy;

int gd=DETECT, gm;

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

printf("Enter the values of x1, y1,x2, y2:\n");

scanf("%d%d%d%d",&x1, &y1, &x2, &y2);

outtextxy(200,250,"original:");

rectangle(x1,y1,x2,y2);

printf("enter the valuve of sx,xy:\n");

scanf("%d%d",&sx,&sy);

x3=x1\*sx;

y3=y1\*sy;

x4=x2\*sx;

y4=y2\*sy;

outtextxy(400,250,"increased size:");

rectangle(x3,y3,x4,y4);

printf("enter the values of -sx and -sy for decreasing size:\n");

scanf("%d%d", &sx,&sy);

x5 = x1/sx;

y5 = y1/sy;

x6 = x2/sx;

y6 = y2/sy;

outtextxy(50, 250, "Decreased size:");

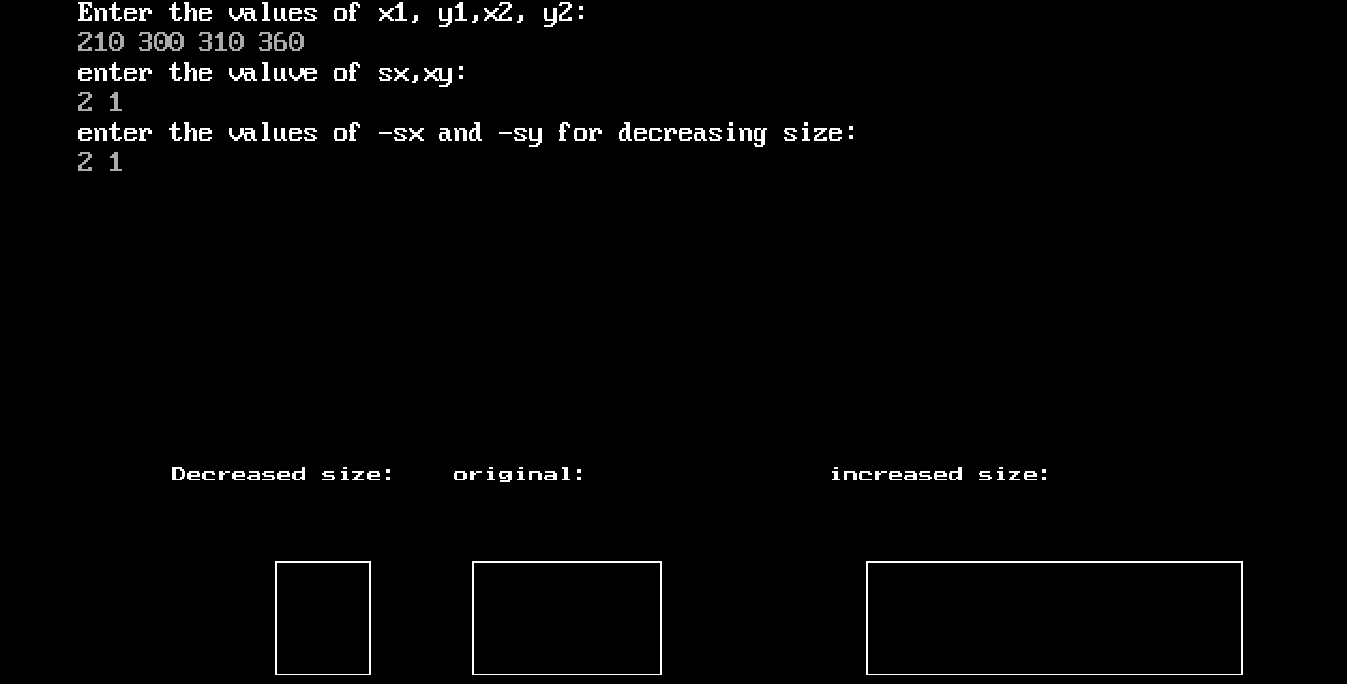
rectangle(x5, y5, x6, y6);

getch();

closegraph();

}

**Output :**



**Program No:**

**Aim:** Write a program to implement 3D Scaling Transformation.

#include <graphics.h>

#include<iostream.h>

#include<conio.h>

int main() {

int gd = DETECT, gm;

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

int x1 = 100, y1 = 100, z1 = 100;

int x2 = 200, y2 = 100, z2 = 100;

int x3 = 200, y3 = 200, z3 = 100;

int x4 = 100, y4 = 200, z4 = 100;

float sx = 2.0, sy = 2.0, sz = 2.0;

int newX1 = x1 \* sx, newY1 = y1 \* sy, newZ1 = z1 \* sz;

int newX2 = x2 \* sx, newY2 = y2 \* sy, newZ2 = z2 \* sz;

int newX3 = x3 \* sx, newY3 = y3 \* sy, newZ3 = z3 \* sz;

int newX4 = x4 \* sx, newY4 = y4 \* sy, newZ4 = z4 \* sz;

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x4, y4);

line(x4, y4, x1, y1);

line(newX1, newY1, newX2, newY2);

line(newX2, newY2, newX3, newY3);

line(newX3, newY3, newX4, newY4);

line(newX4, newY4, newX1, newY1);

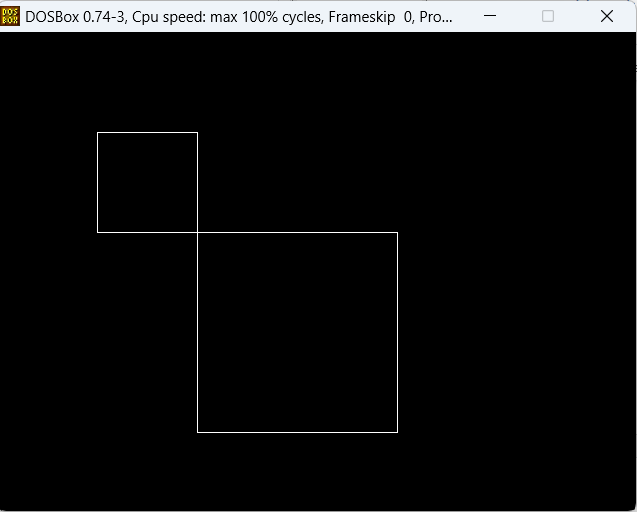
getch();

closegraph();

return 0;

}

**Output:**



**Program No:**

**Aim:** Write a program to implement 2D Translation Transformation.

#include<iostream.h>

#include<graphics.h>

#include<math.h>

#include<conio.h>

int main()

{

int gd=DETECT,gm,s;

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

int x1=200,y1=150,x2=300,y2=250;

int tx=50,ty=50;

cout<<"Rectangle before translation"<<endl;

setcolor(3);

rectangle(x1,y1,x2,y2);

setcolor(4);

cout<<"Rectangle after translation"<<endl;

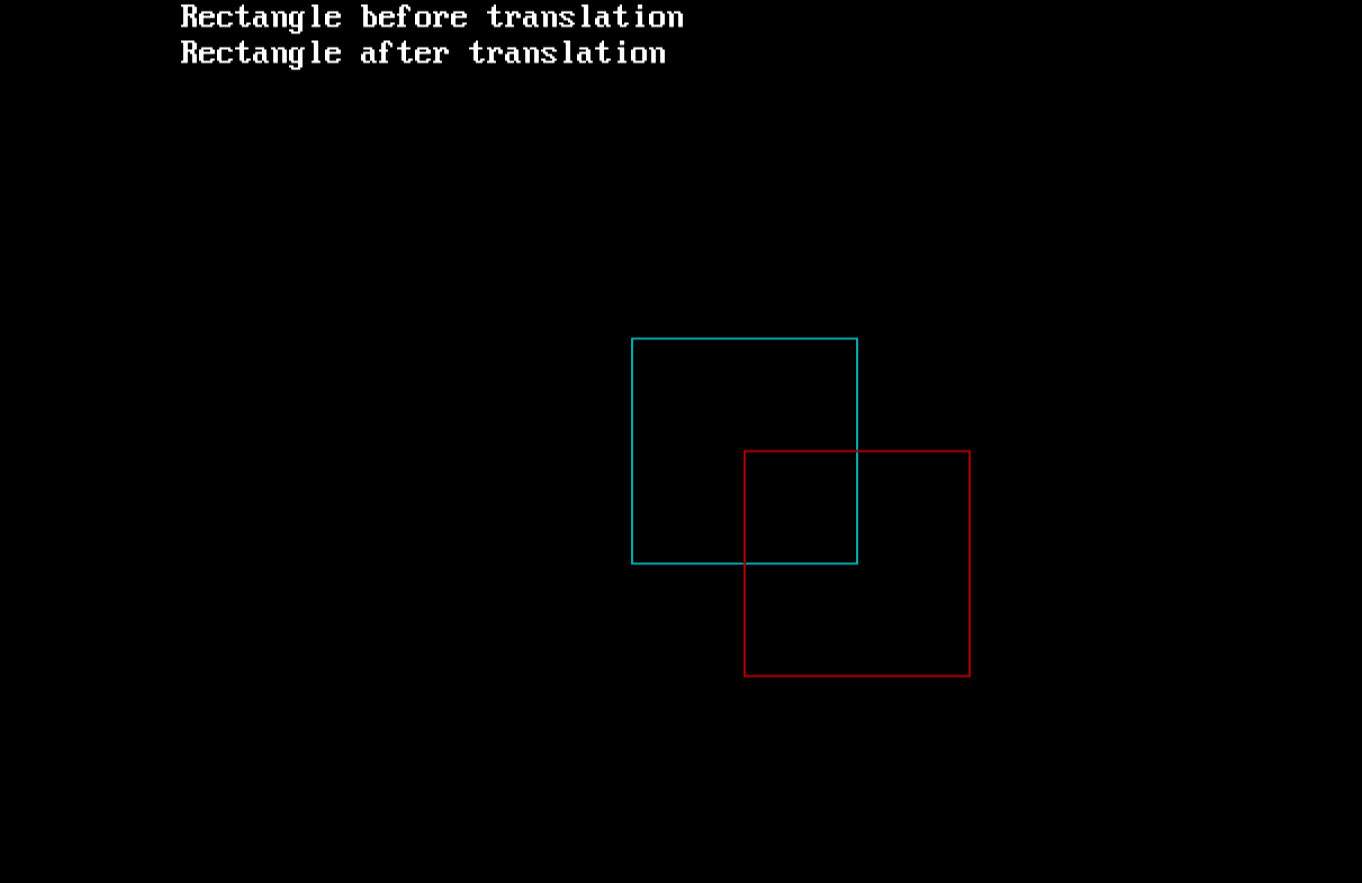
rectangle(x1+tx,y1+ty,x2+tx,y2+ty);

getch();

return 0;

}

**Output:**



**Program No:**

**Aim:** Write a program to implement 3D Translation Transformation.

#include<iostream.h>

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

int maxx,maxy,midx,midy;

void axis()

{

getch();

cleardevice();

line(midx,0,midx,maxy);

line(0,midy,maxx,midy);

}

void main()

{

int x,y,z,o,x1,x2,y1,y2;

int gd=DETECT,gm;

detectgraph(&gd,&gm);

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

//setfillstyle(0,getmaxcolor());

maxx=getmaxx();

maxy=getmaxy();

midx=maxx/2;

midy=maxy/2;

axis();

bar3d(midx+50,midy-100,midx+60,midy-90,10,1);

printf("Enter translation factor");

scanf("%d%d",&x,&y);

//axis();

printf("After translation:");

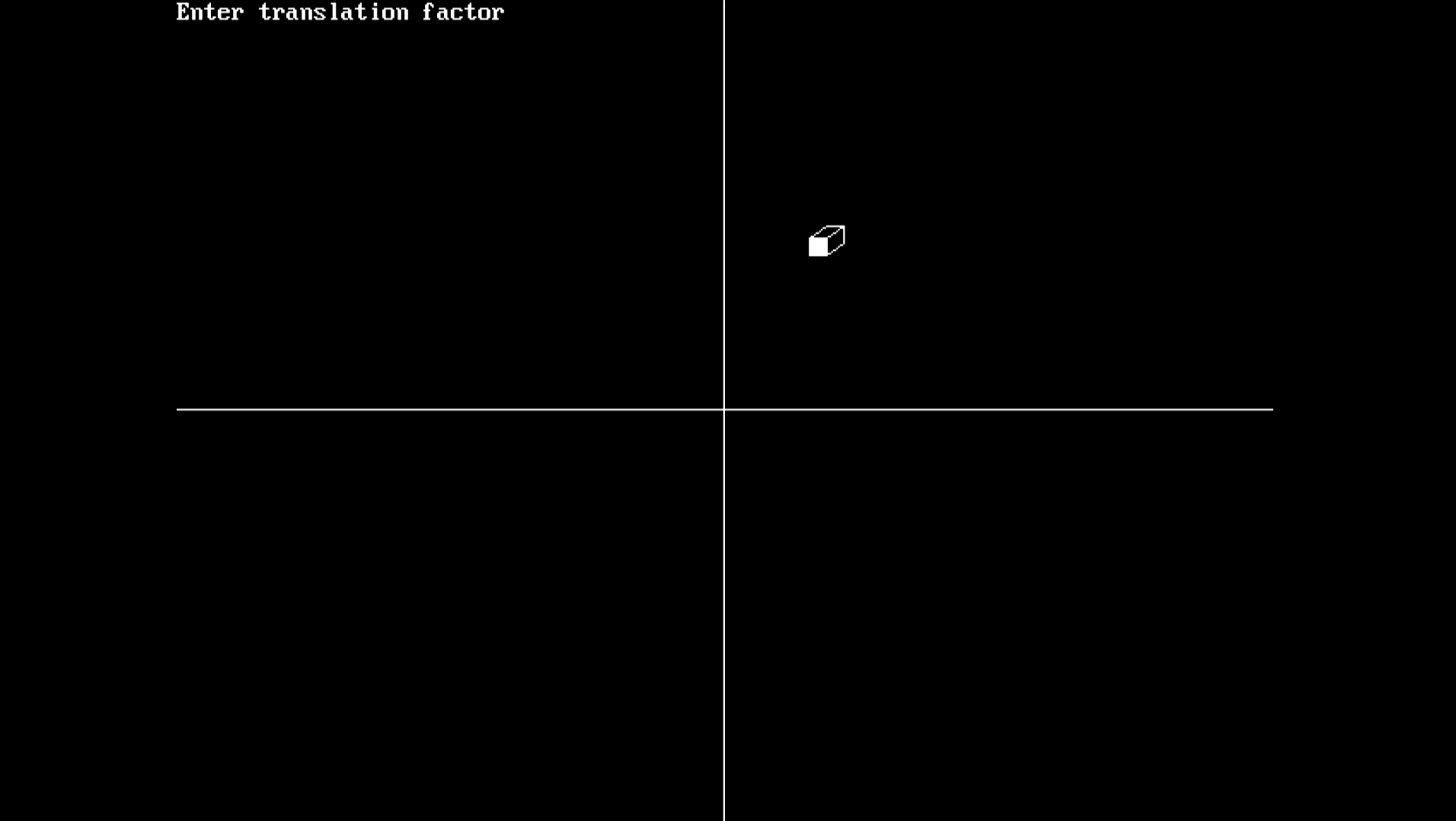
bar3d(midx+x+50,midy-(y+100),midx+x+60,midy-(y+90),10,1);

getch();

closegraph();

}

**Output:**



**Program No.**

**/\* Aim :** Write a program to move an image(ball) on the screen**.**

#include<dos.h>

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

{

int gd=DETECT,gm;

initgraph(&gd,&gm, "C:/TC/BGI"); float x=1,y=0.00000,j=5,count=1; float r=15;

setcolor(14);

line (0,215,650,215)

sleep(1);

for(int k=0;k<=7;k++)

{

for(float i=90;i<270;i+=10)

{

y=cos(((i\*22/7)/180))/j; if(y>0)

y=-y; x+=5;

setcolor(14); setfillstyle(1,14);

circle(x,y\*100+200,r); floodfill(x,y\*100+200,14); delay(100);

setcolor(0); setfillstyle(1,0);

circle(x,y\*100+200,r); floofill(x,y\*100+200,0);

}

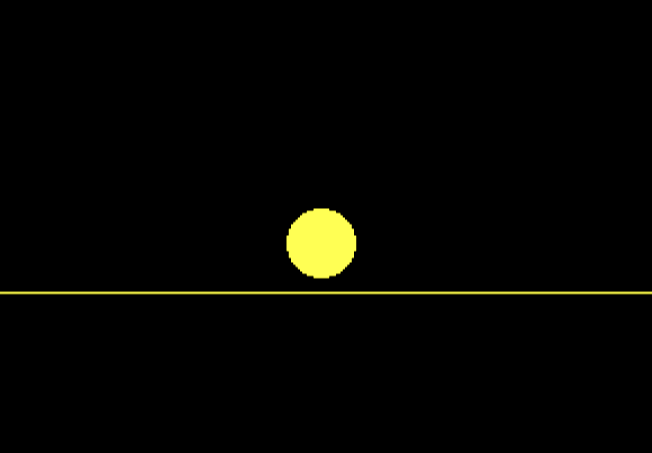
j+=count; count+=1;

}

getch();

}

**OUTPUT:**



**Program No:**

**Aim:** Write a program to generate a Cubic Bezier curve.

#include<iostream.h>

#include<stdlib.h>

#include<graphics.h>

#include<math.h>

#include<conio.h>

void 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<1.0;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()

{

int x[4],y[4];

int i;

cout<<"Enter the x- and y-coordinates of the four control points.";

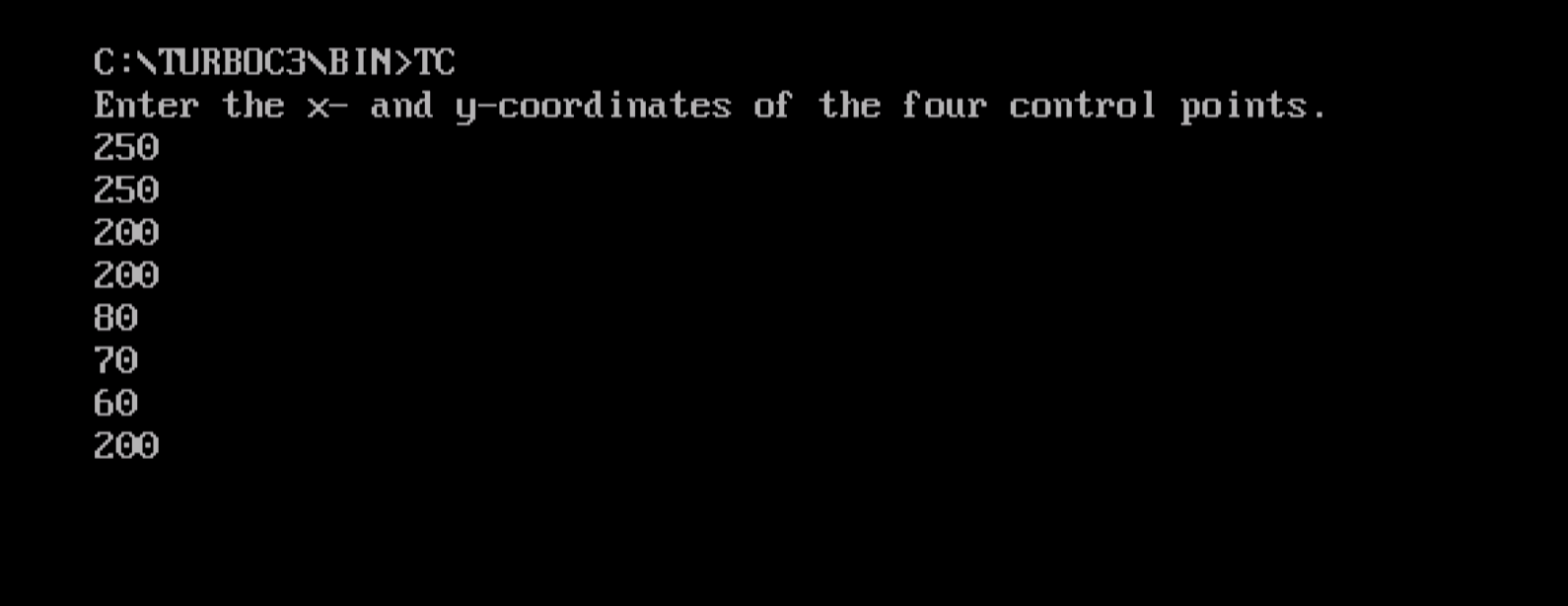
for(i=0;i<4;i++)

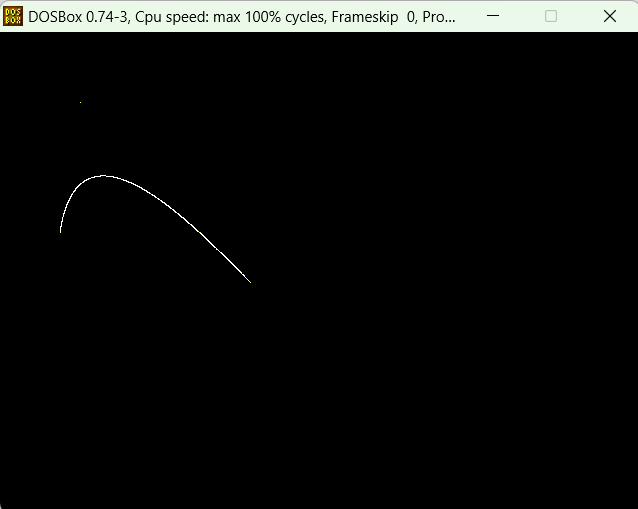
cin>>x[i]>>y[i];

bezier(x,y);

}

**Output:**





**Program No:**

**Aim:** Write a program to draw a polygon using Absolute and Relative commands.

#include <graphics.h>

#include<iostream.h>

#include<conio.h>

int main()

{

int gd = DETECT, gm;

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

int numVertices = 4;

int x[] = {100, 200, 300, 200};

int y[] = {100, 200, 100, 50};

setcolor(YELLOW);

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

{

int next = (i + 1) % numVertices;

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

}

setcolor(RED);

int dx[] = {50, 50, -50, -50};

int dy[] = {0, 50, 0, -50};

int startX = 400, startY = 100;

int currentX = startX, currentY = startY;

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

{

currentX += dx[i];

currentY += dy[i];

int next = (i + 1) % numVertices;

line(currentX, currentY, currentX + dx[next], currentY + dy[next]);

}

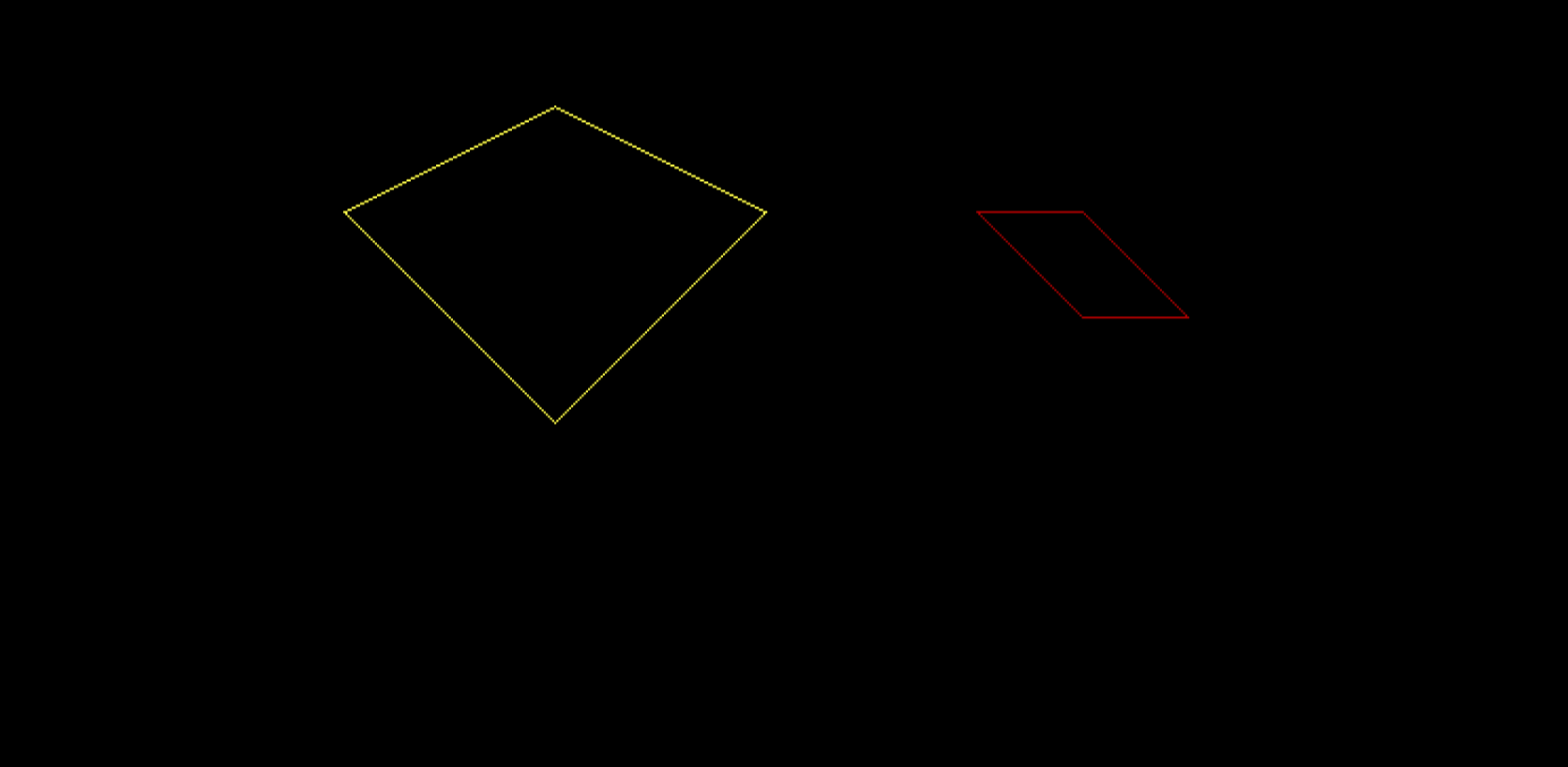
getch();

closegraph();

return 0;

}

**Output:**



**Program No:**

**Aim:** Write a program to clip user defined area of a screen.

#include <graphics.h>

#include<iostream.h>

#include<conio.h>

int main()

{

int gd = DETECT, gm;

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

int screenWidth = getmaxx();

int screenHeight = getmaxy();

int left, top, right, bottom;

cout << "Enter the coordinates of the rectangle (left, top, right, bottom): ";

cin >> left >> top >> right >> bottom;

if (left < 0)

{

left = 0;

}

if (top < 0)

{

top = 0;

}

if (right > screenWidth)

{

right = screenWidth;

}

if (bottom > screenHeight)

{

bottom = screenHeight;

}

setcolor(YELLOW);

rectangle(left, top, right, bottom);

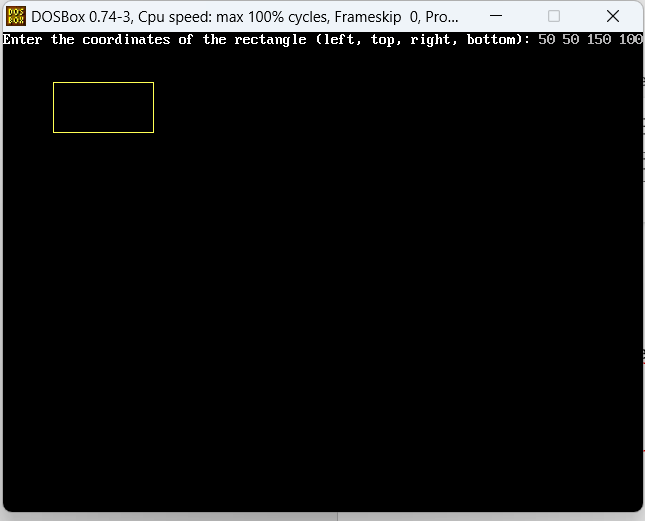
getch();

closegraph();

return 0;

}

**Output:**



**Program No:**

**Aim:** Write a program to clip the line segment.

#include<graphics.h>

#include<conio.h>

#include<iostream.h>

void clipLine(int &x0, int &y0, int &x1, int &y1, int left, int top, int right, int bottom) {

if (x0 < left) {

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

x0 = left;

} else if (x0 > right) {

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

x0 = right;

}

if (x1 < left) {

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

x1 = left;

} else if (x1 > right) {

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

x1 = right;

}

if (y0 < top) {

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

y0 = top;

} else if (y0 > bottom) {

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

y0 = bottom;

}

if (y1 < top) {

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

y1 = top;

} else if (y1 > bottom) {

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

y1 = bottom;

}

line(x0, y0, x1, y1);

}

int main() {

int gd = DETECT, gm;

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

int left = 100, top = 100, right = 300, bottom = 200;

rectangle(left, top, right, bottom);

int x0 = 50, y0 = 150, x1 = 350, y1 = 250;

line(x0, y0, x1, y1);

clipLine(x0, y0, x1, y1, left, top, right, bottom);

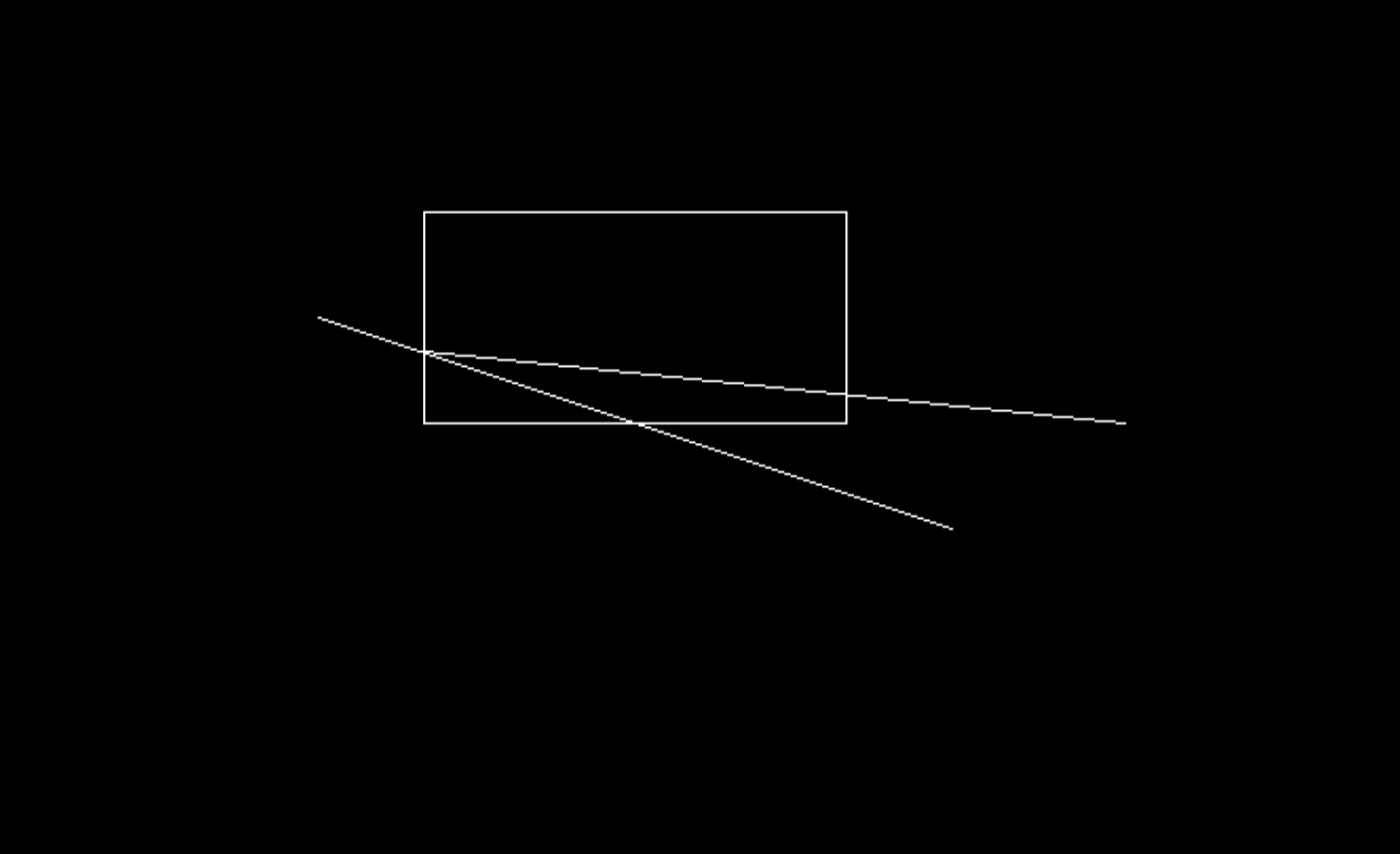
getch();

closegraph();

return 0;

}

**Output:**



**Program No:**

**Aim:** Write a program to clip polygon.

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#define round(a) ((int)(a+0.5))

int k;

float xmin,ymin,xmax,ymax,arr[20],m;

void clipl(float x1,float y1,float x2,float y2)

{

if(x2-x1)

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

else

m=100000;

if(x1 >= xmin && x2 >= xmin)

{

arr[k]=x2;

arr[k+1]=y2;

k+=2;

}

if(x1 < xmin && x2 >= xmin)

{

arr[k]=xmin;

arr[k+1]=y1+m\*(xmin-x1);

arr[k+2]=x2;

arr[k+3]=y2;

k+=4;

}

if(x1 >= xmin && x2 < xmin)

{

arr[k]=xmin;

arr[k+1]=y1+m\*(xmin-x1);

k+=2;

}}

void clipt(float x1,float y1,float x2,float y2)

{

if(y2-y1)

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

else

m=100000;

if(y1 <= ymax && y2 <= ymax)

{

arr[k]=x2;

arr[k+1]=y2;

k+=2;

}

if(y1 > ymax && y2 <= ymax)

{

arr[k]=x1+m\*(ymax-y1);

arr[k+1]=ymax;

arr[k+2]=x2;

arr[k+3]=y2;

k+=4;

}

if(y1 <= ymax && y2 > ymax)

{

arr[k]=x1+m\*(ymax-y1);

arr[k+1]=ymax;

k+=2;

}}

void clipr(float x1,float y1,float x2,float y2)

{

if(x2-x1)

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

else

m=100000;

if(x1 <= xmax && x2 <= xmax)

{

arr[k]=x2;

arr[k+1]=y2;

k+=2;

}

if(x1 > xmax && x2 <= xmax)

{

arr[k]=xmax;

arr[k+1]=y1+m\*(xmax-x1);

arr[k+2]=x2;

arr[k+3]=y2;

k+=4;

}

if(x1 <= xmax && x2 > xmax)

{

arr[k]=xmax;

arr[k+1]=y1+m\*(xmax-x1);

k+=2;

}

}

void clipb(float x1,float y1,float x2,float y2)

{

if(y2-y1)

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

else

m=100000;

if(y1 >= ymin && y2 >= ymin)

{

arr[k]=x2;

arr[k+1]=y2;

k+=2;

}

if(y1 < ymin && y2 >= ymin)

{

arr[k]=x1+m\*(ymin-y1);

arr[k+1]=ymin;

arr[k+2]=x2;

arr[k+3]=y2;

k+=4;

}

if(y1 >= ymin && y2 < ymin)

{

arr[k]=x1+m\*(ymin-y1);

arr[k+1]=ymin;

k+=2;

}

}

void main()

{

int gdriver=DETECT,gmode,n,poly[20];

float xi,yi,xf,yf,polyy[20];

clrscr();

cout<<"Coordinates of rectangular clip window :\nxmin,ymin:";

cin>>xmin>>ymin;

cout<<"xmax,ymax:";

cin>>xmax>>ymax;

cout<<"\n\nPolygon to be clipped:\nNumber of sides :";

cin>>n;

cout<<"Enter the coordinates:";

for(int i=0;i < 2\*n;i++)

cin>>polyy[i];

polyy[i]=polyy[0];

polyy[i+1]=polyy[1];

for(i=0;i < 2\*n+2;i++)

poly[i]=round(polyy[i]);

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

setcolor(RED);

rectangle(xmin,ymax,xmax,ymin);

cout<<"\t\tUNCLIPPED POLYGON";

setcolor(WHITE);

fillpoly(n,poly);

getch();

cleardevice();

k=0;

for(i=0; i<2\*n; i+=2)

clipl(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);

n=k/2;

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

polyy[i]=arr[i];

polyy[i]=polyy[0];

polyy[i+1]=polyy[1];

k=0;

for(i=0;i < 2\*n;i+=2)

clipt(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);

n=k/2;

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

polyy[i]=arr[i];

polyy[i]=polyy[0];

polyy[i+1]=polyy[1];

k=0;

for(i=0;i < 2\*n;i+=2)

clipr(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);

n=k/2;

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

polyy[i]=arr[i];

polyy[i]=polyy[0];

polyy[i+1]=polyy[1];

k=0;

for(i=0;i < 2\*n;i+=2)

clipb(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);

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

poly[i]=round(arr[i]);

if(k)

fillpoly(k/2,poly);

setcolor(RED);

rectangle(xmin,ymax,xmax,ymin);

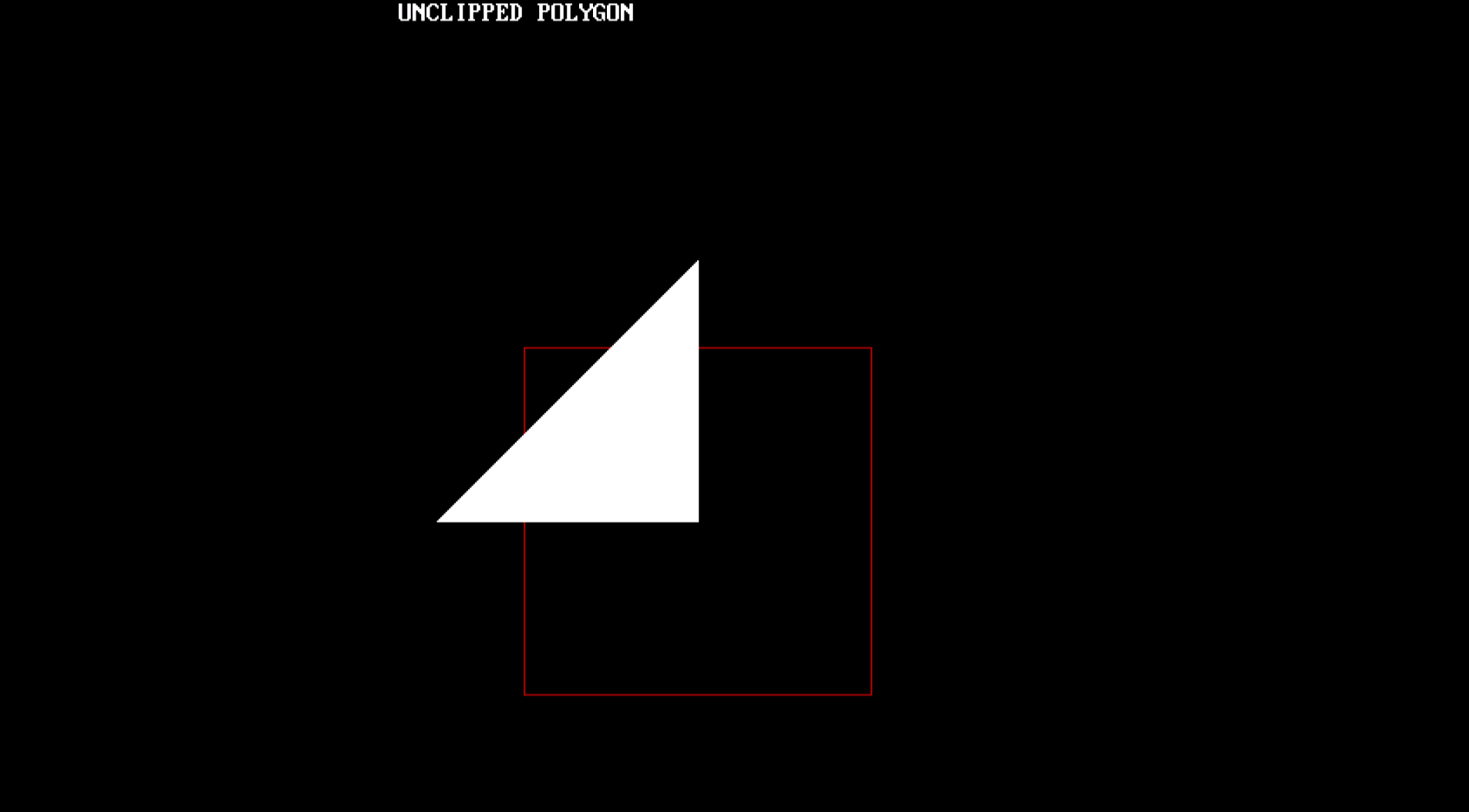
cout<<"\tCLIPPED POLYGON";

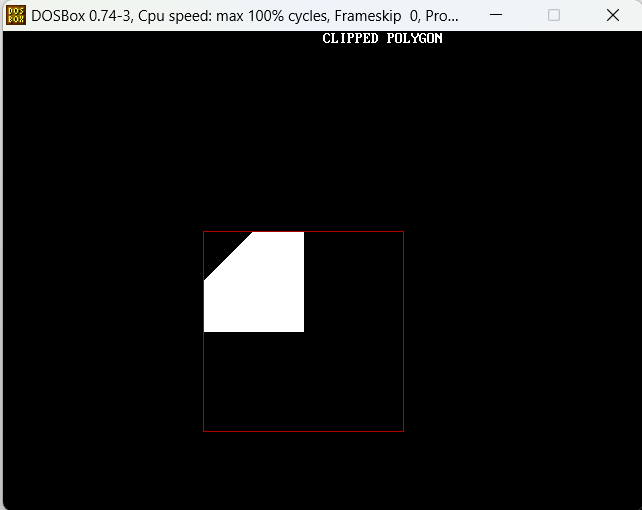
getch();

closegraph();

}

**Output:**





**Program No:**

/\***Aim:** Write a program to demonstrate rotation of point.

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

double x, y, angle, radian;

const double PI = 3.14;

cout << "Enter the x-coordinate of the point: ";

cin >> x;

cout << "Enter the y-coordinate of the point: ";

cin >> y;

cout << "Enter the angle of rotation in degrees: ";

cin >> angle;

radian = angle \* PI / 180.0;

double rotatedX = x \* cos(radian) - y \* sin(radian);

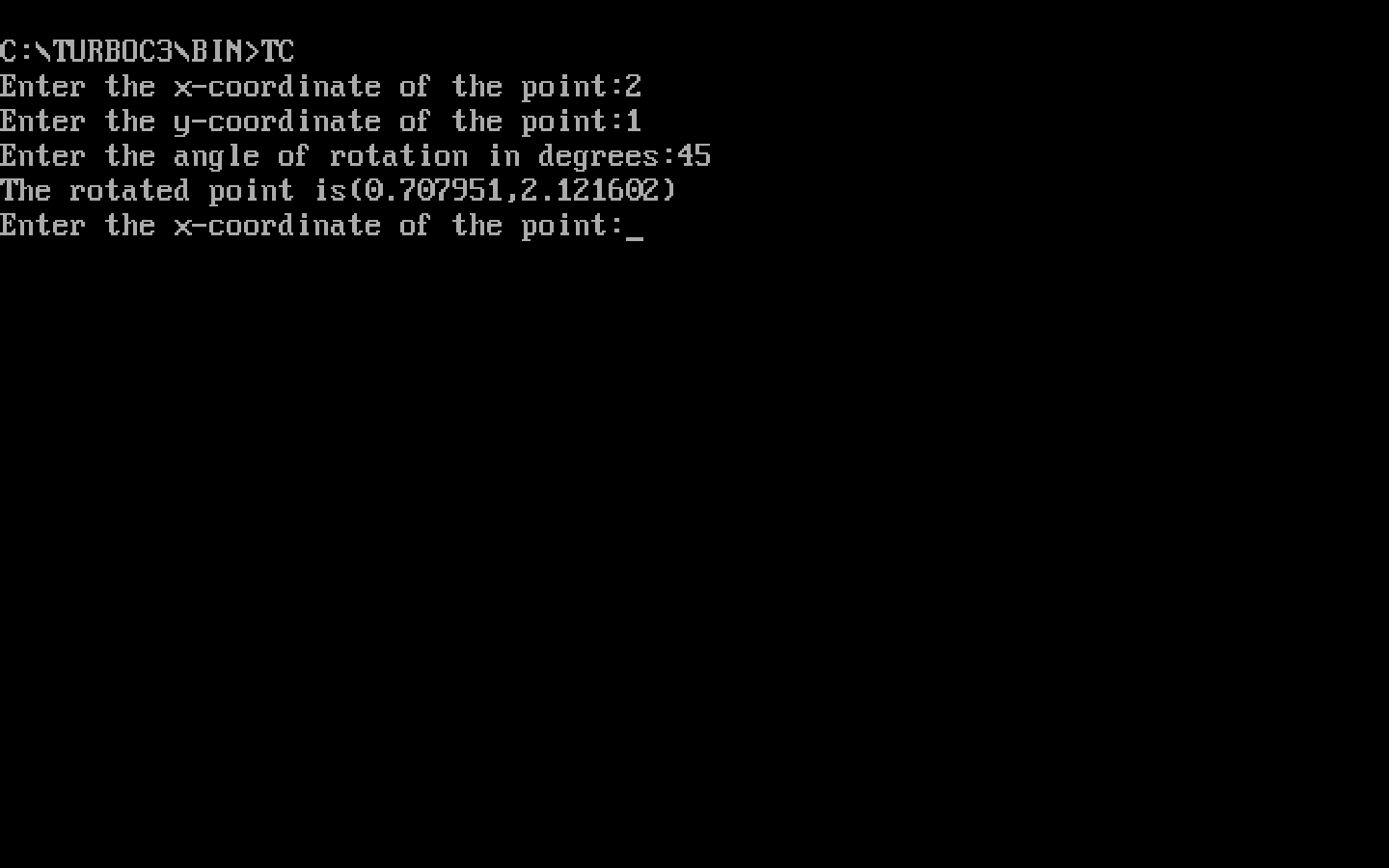
double rotatedY = x \* sin(radian) + y \* cos(radian);

cout << "The rotated point is (" << rotatedX << ", " << rotatedY << ")" << endl;

return 0;

}

**Output:**



**Program No:**

**/\*Aim:** Write a program to fill area by given pattern.

#include <graphics.h>

#include<iostream.h>

#include<conio.h>

int main()

{

int gd = DETECT, gm;

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

setfillstyle(SOLID\_FILL, YELLOW);

int points[] = {100, 100, 200, 100, 150, 200, 100, 150, 100, 100};

fillpoly(5, points);

getch();

closegraph();

return 0;

}

**Output:**

}

**Program No:**

**/\*Aim:** Write a program to fill the closed area using flood fill method.

#include <graphics.h>

#include<iostream.h>

#include<conio.h>

void floodFill(int x, int y, int fillColor, int oldColor)

{

if (getpixel(x, y)==oldColor)

{

putpixel(x, y, fillColor);

floodFill(x + 1, y, fillColor, oldColor);

floodFill(x - 1, y, fillColor, oldColor);

floodFill(x, y + 1, fillColor, oldColor);

floodFill(x, y - 1, fillColor, oldColor);

}

}

int main() {

int gd = DETECT, gm;

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

rectangle(100, 100, 200, 200);

floodFill(150, 150, GREEN, BLACK);

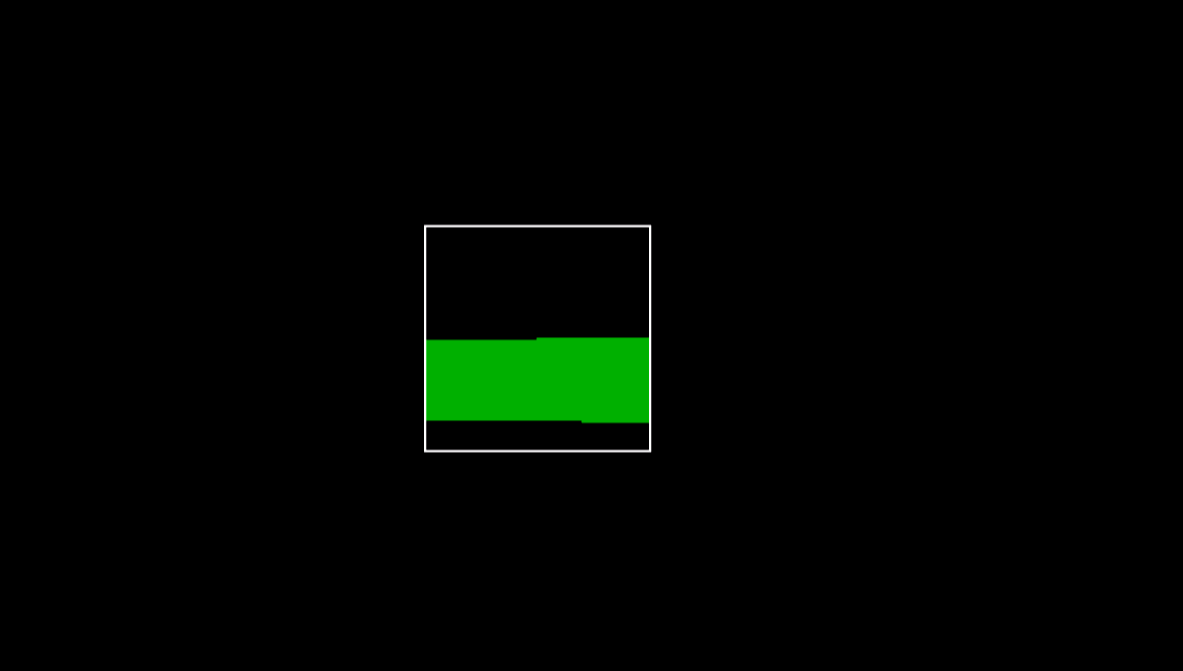
getch();

closegraph();

return 0;

}

**Output:**



**Program No:**

**/\*Aim:** Write a program to fill the closed area using Scan line method.

#include <graphics.h>

#include<iostream.h>

#include<conio.h>

void scanLineFill(int x, int y, int fillColor)

{

int currentColor = getpixel(x, y);

if (currentColor != fillColor)

{

int stackX[1000], stackY[1000];

int top = 0;

stackX[top] = x;

stackY[top] = y;

while (top >= 0)

{

x = stackX[top];

y = stackY[top];

top--;

while (getpixel(x, y) == currentColor)

{

putpixel(x, y, fillColor);

x++;

}

if (getpixel(x, y) != fillColor)

{

int newX = x;

while (getpixel(newX, y) != currentColor && getpixel(newX, y) != fillColor)

{

newX++;

}

top++;

stackX[top] = newX;

stackY[top] = y;

}

if (getpixel(x - 1, y) != fillColor)

{

int newX = x - 1;

while (getpixel(newX, y) != currentColor && getpixel(newX, y) != fillColor) {

newX--;

}

top++;

stackX[top] = newX;

stackY[top] = y;

}

}

}

}

int main() {

int gd = DETECT, gm;

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

rectangle(100, 100, 200, 200);

scanLineFill(150, 150, GREEN);

getch();

closegraph();

return 0;

}

**Output:**

