# PART- C

# Program to draw a polygon by accepting vertex coordinates and fill it using Boundary fill or Flood fill based on user’s choice.

#include<iostream.h>

#include<conio.h>

#include<stdio.h>

#include<graphics.h>

#include<dos.h>

void floodfill4(int x,int y, int fill,int old)

{

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

{

delay(5);

putpixel(x,y,fill);

floodfill4(x+1,y,fill,old);

floodfill4(x-1,y,fill,old);

floodfill4(x,y+1,fill,old);

floodfill4(x,y-1,fill,old);

}

}

void boundryfill4(int x,int y, int fill, int boundry)

{

int current;

current = getpixel(x,y);

if(current != fill&&current!=boundry)

{

putpixel(x,y,fill);

boundryfill4(x+1,y,fill,boundry);

boundryfill4(x-1,y,fill,boundry);

boundryfill4(x,y+1,fill,boundry);

boundryfill4(x,y-1,fill,boundry);

}

}

void main()

{

int gd=DETECT,gm;

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

int n,i,x,y,ch,bc,fc,old;

int a[10],b[10];

cleardevice();

cout<<"Enter no vertices of a polygon: ";

cin>>n;

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

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

cin>>a[i]>>b[i];

}

cout<<"Enter seed pixel: ";

cin>>x>>y;

do{

cout<<"1. Flood fill"<<endl;

cout<<"2. Boundry fill"<<endl;

cout<<"3. Exit"<<endl;

cout<<"Enter your choice :";

cin>>ch;

switch(ch)

{

case 1: cleardevice();

cout<<"Enter the border color: ";

cin>>bc;

cout<<"Enter the filling color: ";

cin>>fc;

setcolor(bc);

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

line(a[i],b[i],a[i+1],b[i+1]);

line(a[i],b[i],a[0],b[0]);

old=getpixel(x,y);

floodfill4(x,y,fc,old);

break;

case 2: cleardevice();

cout<<"Enter the border color :";

cin>>bc;

cout<<"Enter the filling color :";

cin>>fc;

setcolor(bc);

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

line(a[i],b[i],a[i+1],b[i+1]);

line(a[i],b[i],a[0],b[0]);

boundryfill4(x,y,fc,bc);

break;

}

}

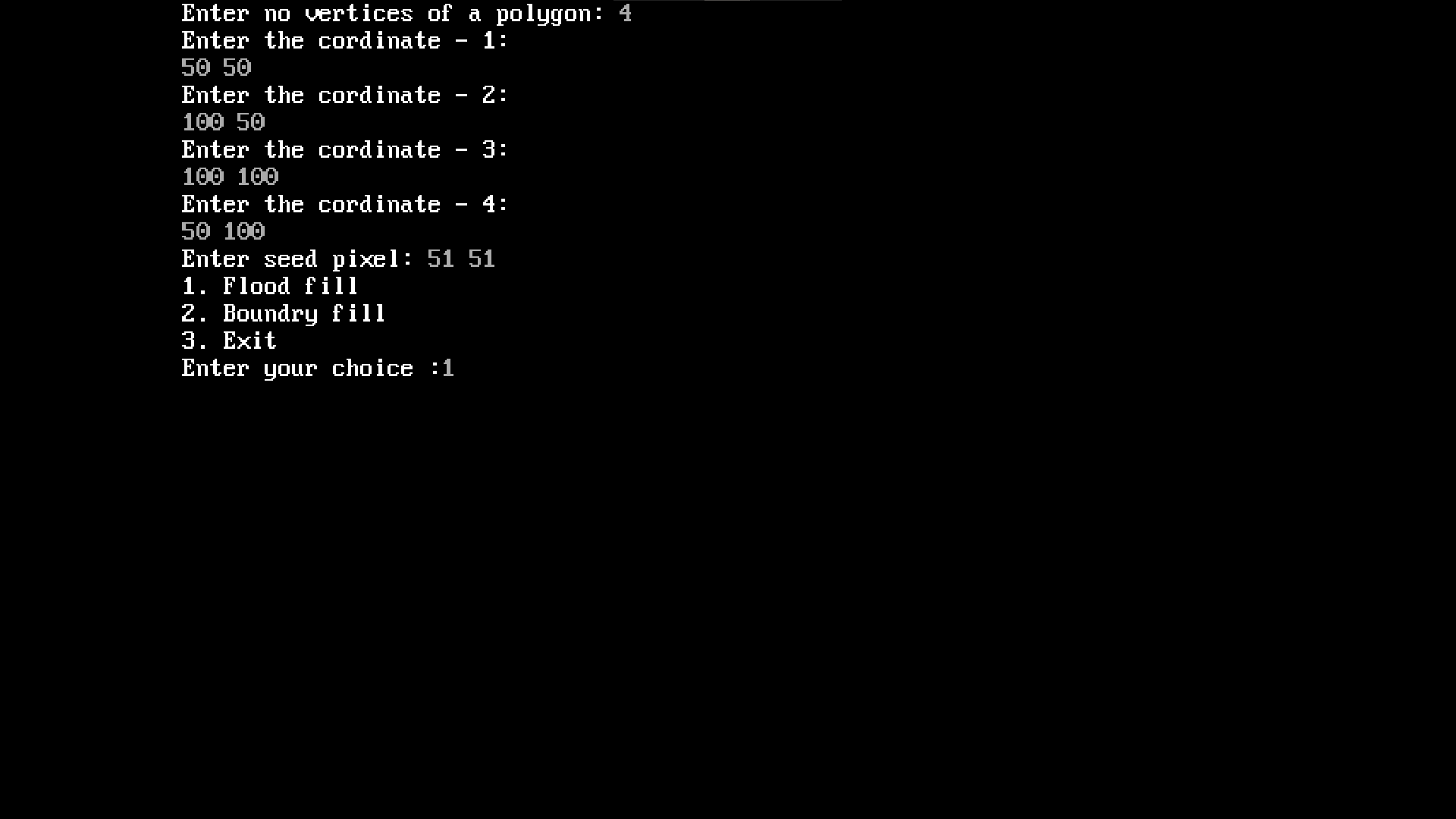
while(ch<3);

getch();

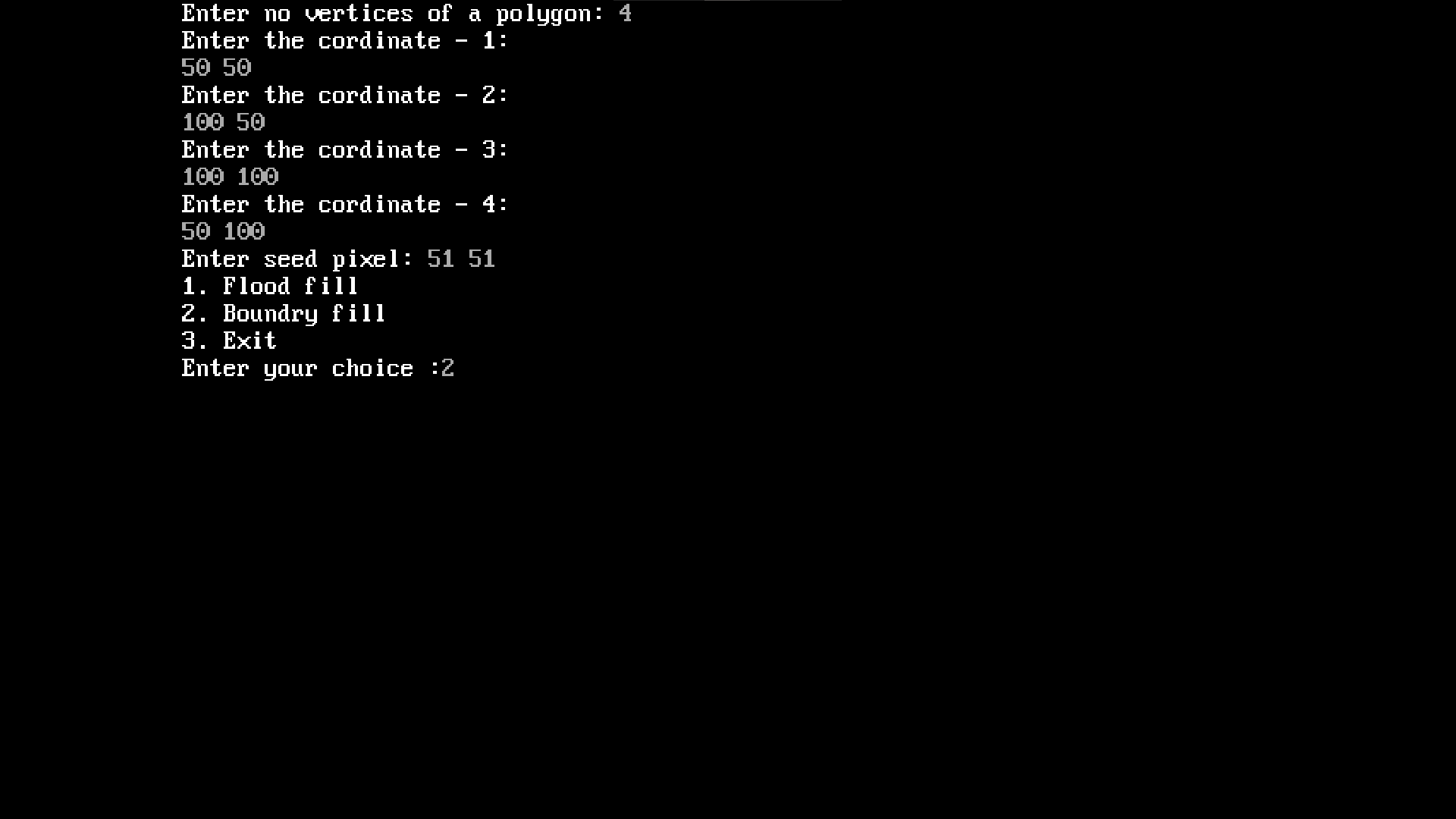
closegraph();

}

Output:









# 2. Program to draw a polygon by accepting vertex coordinates and perform basic transformations (translation, scaling, rotation) about a given point based on user’s choice.

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

#include<stdlib.h>

void menu();

void input();

void output();

void translation();

void rotation();

void scaling();

int i,x,ch=0,temp,tx,ty,fx,fy,r,ang,n;

float sx,sy, a[10],b[10];

void input()

{

printf("enter the number of vertices:" );

scanf("%d",&n);

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

{

printf("enter the coordinates:\n");

scanf("%f%f",&a[i],&b[i]);

}

}

void output()

{

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

line(a[i],b[i],a[i+1],b[i+1]);

line(a[i],b[i],a[0],b[0]);

}

void translation()

{

output();

printf("enter the tranformation vertex tx,ty:\n");

scanf("%d%d",&tx,&ty);

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

a[i]=a[i]+tx;

b[i]=b[i]+ty;

}

delay(10);

output();

getch();

}

void rotation()

{

printf("enter the rotating angle:");

scanf("%d",&ang);

printf("enter the pivot point:\n");

scanf("%d%d",&fx,&fy);

output();

r=(ang\*3.14)/180;

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

{

a[i]=fx+(a[i]-fx)\*cos(r)-(b[i]-fy)\*sin(r);

b[i]=fy+(a[i]-fx)\*sin(r)+(b[i]-fy)\*cos(r);

}

delay(10);

output();

getch();

}

void scaling()

{

printf("enter the scaling factor\n");

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

printf("enter the fixed point:\n");

scanf("%d%d",&fx,&fy);

output();

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

{

a[i]=a[i]\*sx+fy\*(1-sx);

b[i]=b[i]\*sy+fy\*(1-sy);

}

delay(10);

output();

getch();

}

void main(){

int gd=DETECT,gm;

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

do

{

cleardevice();

printf("menu\n");

printf("1.Translation\n");

printf("2.rotation\n");

printf("3.scaling\n");

printf("4.exit\n");

printf("enter the choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:

input();

translation();

break;

case 2:

input();

rotation();

break;

case 3:

input();

scaling();

break;

}

}

while(ch<=3);

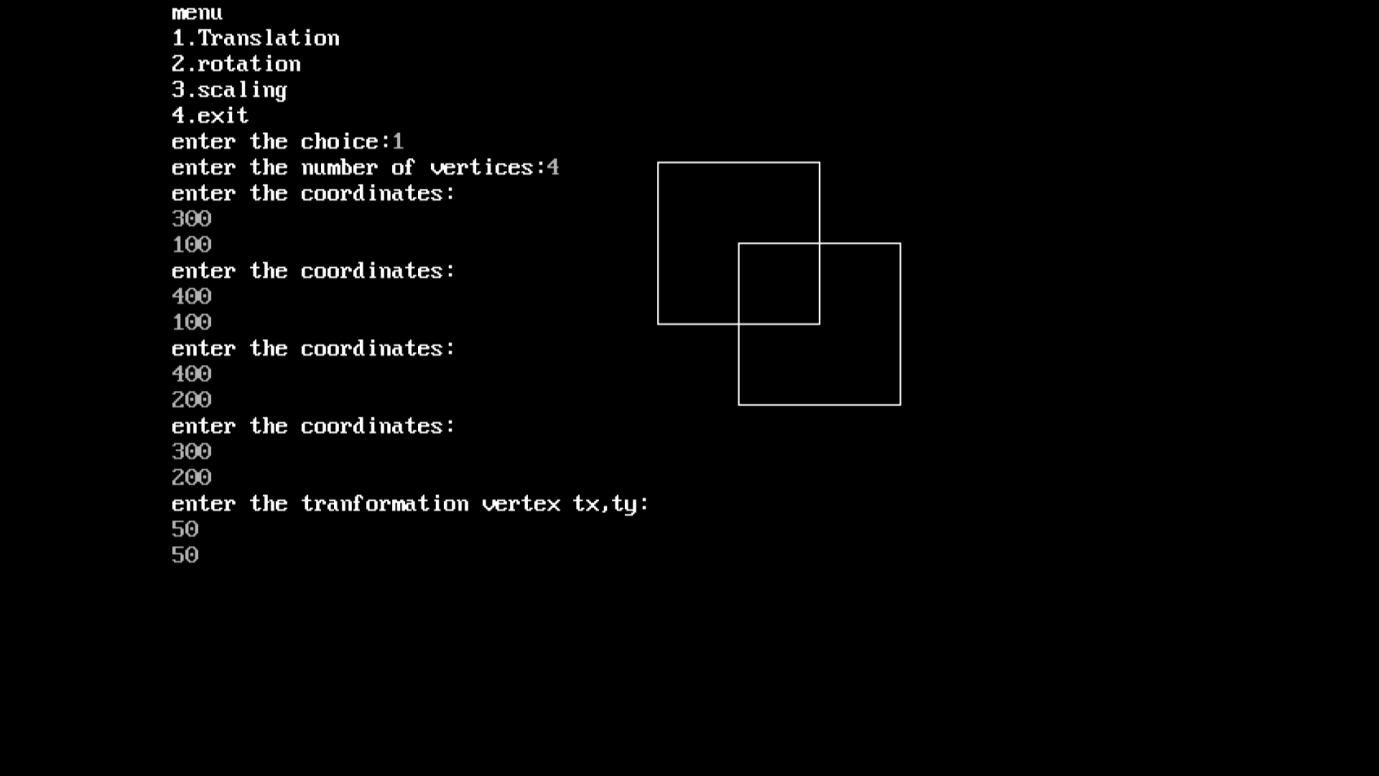
getch();

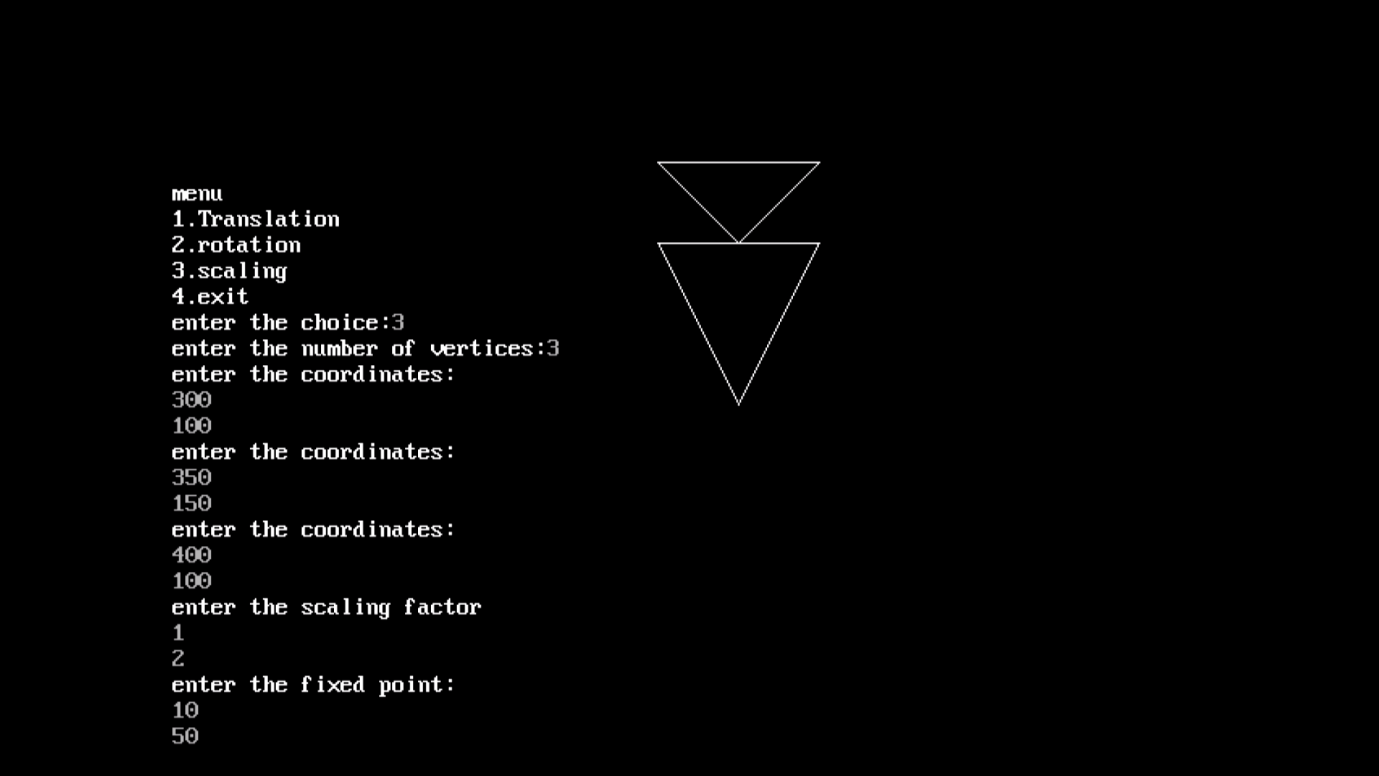
closegraph();

}

Output:







# 3. Program to draw a polygon by accepting vertex coordinates and perform shear and reflections about x and y axes based on user’s choice.

#include<stdio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

#include<stdlib.h>

#include<conio.h>

void input();

void output();

void shearing();

void reflection();

int a[10],b[10],i,ch=0,tx,ty,fx,fy,n,axis,sh;

float sx,sy;

void input()

{

printf("Enter the number of vertices:");

scanf("%d",&n);

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

{

printf("Enter the coordinate: ");

scanf("%d%d",&a[i],&b[i]);

}

}

void output()

{

line(0,240,640,240 );

line(320,0,320,480);

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

line(320+a[i],240-b[i],320+a[i+1],240-b[i+1]);

line(320+a[i],240-b[i],320+a[0],240-b[0]);

}

void shearing()

{

output();

printf("Enter the shear value:");

scanf("%d",&sh);

printf("Enter the fixed points:");

scanf("%d%d",&fx,&fy);

printf("Enter the axis for for shearing. If x-axis then 1,if y-axis then 0:");

scanf("%d",&axis);

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

{

if(axis==1)

a[i]=a[i]+sh\*(b[i]-fy);

else

b[i]=b[i]+sh\*(a[i]-fx);

}

delay(10);

output();

getch();

}

void reflection()

{

output();

printf("Enter the reflection about axis if x-axis then 1 if y-axis then 0: ");

scanf("%d",&axis);

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

{

if(axis==1)

b[i]=(-1)\*b[i];

else

a[i]=(-1)\*a[i];

}

delay(10);

output();

getch();

}

void main()

{

int gd=DETECT,gm;

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

do

{

cleardevice();

printf("Menu\n");

printf("1.Shearing\n");

printf("2.Reflection\n");

printf("3.Exit\n");

printf("Enter your choice: ");

scanf("%d",&ch);

switch(ch)

{

case 1:

input();

shearing();

break;

case 2:

input();

reflection();

break;

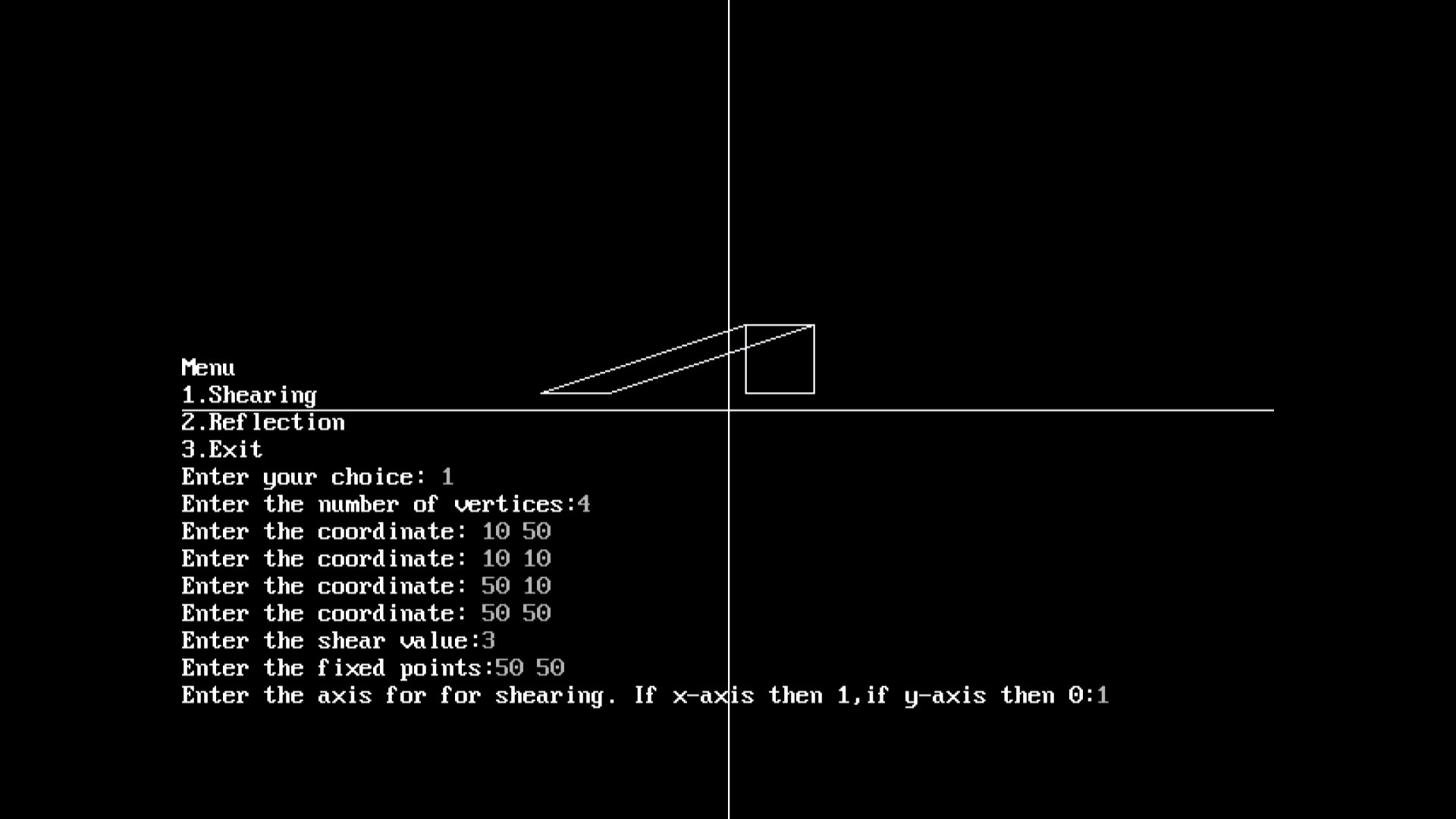
}

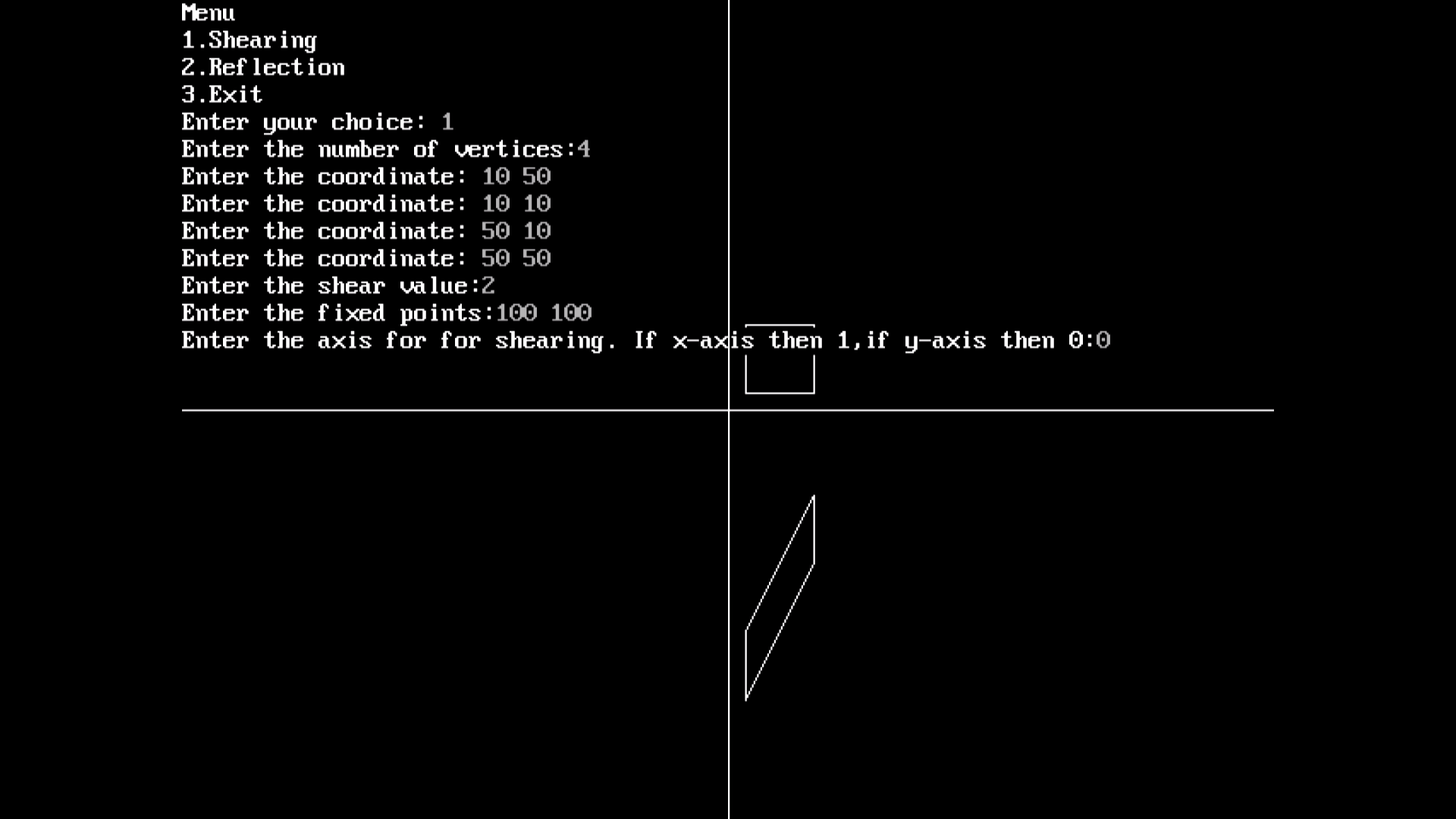
}while(ch<=2);

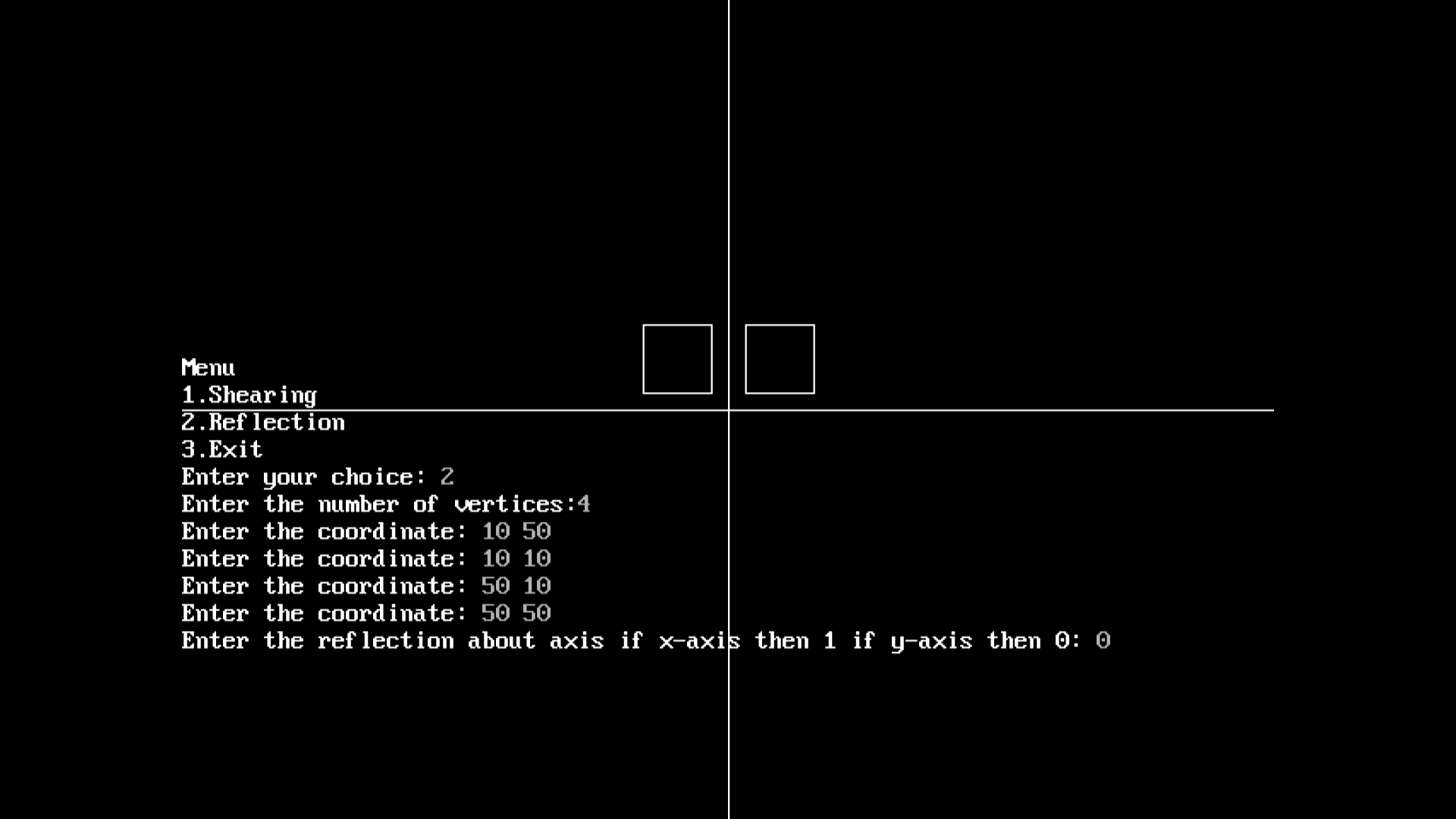
getch();

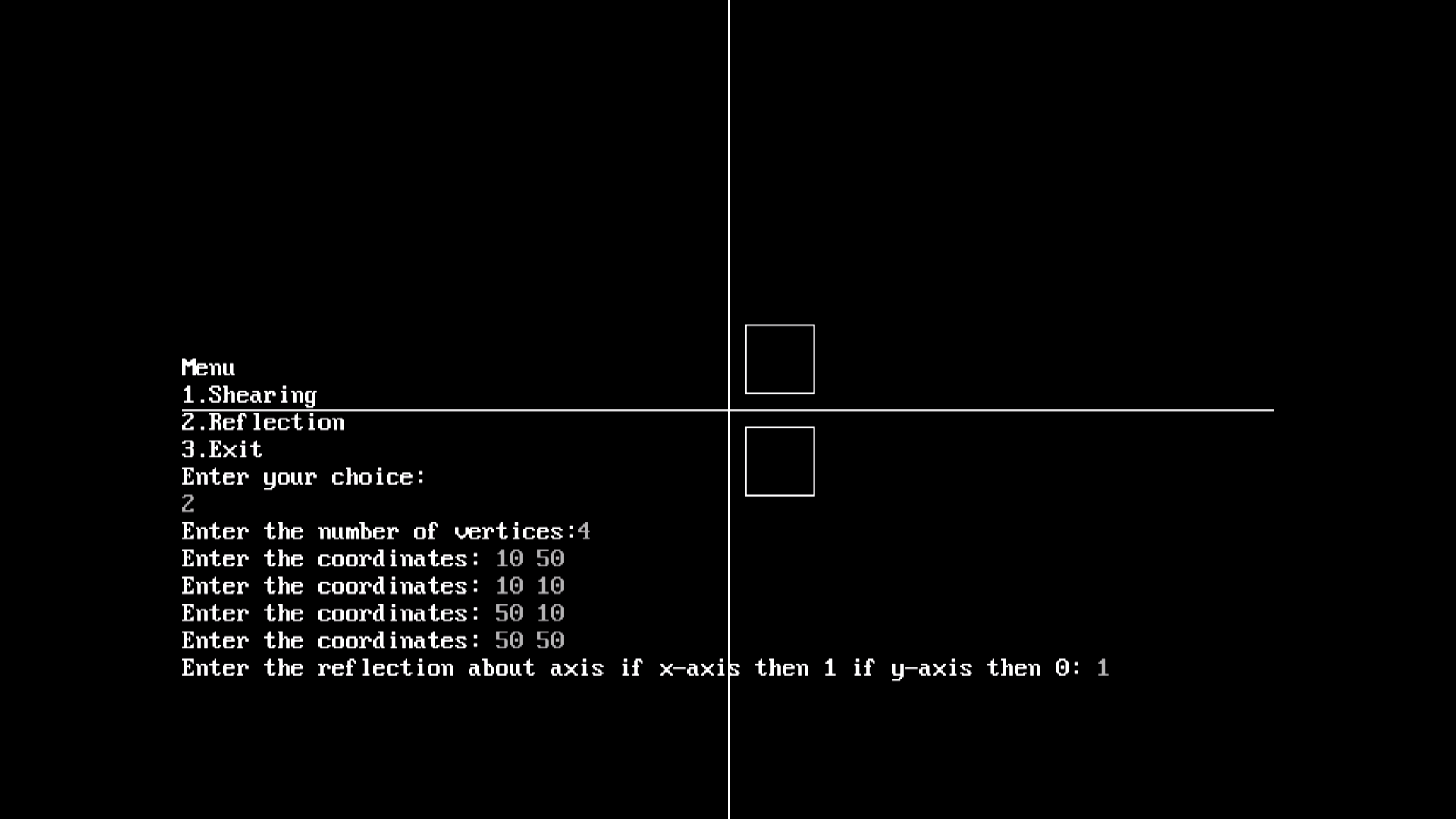
}

Output:









# 4. Program to implement Cohen-Sutherland Line clipping algorithm.

//Program to implement Cohen-Sutherland Line Clipping Algorithm

#include<conio.h>

#include<iostream.h>

#include<graphics.h>

static int LEFT=1,RIGHT=2,BOTTOM=4,TOP=8,x1,y1,xh,yh;

int getcode(int x, int y)

{

int code=0;

//Perform bitwisw OR to get outcode

if(y>yh) code |= TOP;

if(y<y1) code |= BOTTOM;

if(x<x1) code |= LEFT;

if(x>xh) code |= RIGHT;

return code;

}

void main()

{

int gd=DETECT,gm;

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

setcolor(BLUE);

cout<<"Enter bottom left and top right co-ordinates of window: ";

cin>>x1>>y1>>xh>>yh;

rectangle(x1,y1,xh,yh);

int x1,y1,x2,y2;

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

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

line(x1,y1,x2,y2);

getch();

int outcode1=getcode(x1,y1), outcode2=getcode(x2,y2);

int accept = 0; //declare if line is to be drawn

while(1)

{

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

//Both points inside.Accept line

if(outcode1==0 && outcode2==0){

accept=1;

break;

}

//AND of both codes !=0.Line is outside. Reject line

else if((outcode1 & outcode2)!=0){

break;

}

else{

int x,y;

int temp;

//Decide if point1 is inside, if not, calculate intersection

if(outcode1==0)

temp=outcode2;

else

temp=outcode1;

//Line clips top edge

if(temp & TOP)

{

x=x1+(yh-y1)/m;

y=yh;

}

else if(temp & BOTTOM)//Line clips bottom edge

{

x=x1+(y1-y1)/m;

y=y1;

}

else if(temp & LEFT)//Line clips left edge

{

x=x1;

y=y1+m\*(x1-x1);

}

else if(temp & RIGHT)//Line clips right edge

{

x=xh;

y=y1+m\*(xh-x1);

}

//Check which point we had selected earlier as temp, and replace its co-ordinates

if(temp==outcode1)

{

x1=x;

y1=y;

outcode1=getcode(x1,y1);

}

else{

x2=x;

y2=y;

outcode2=getcode(x2,y2);

}

}

}

setcolor(WHITE);

if(accept)

{

cleardevice();

cout<<"After clipping: ";

rectangle(x1,y1,xh,yh);

line(x1,y1,x2,y2);

}

else

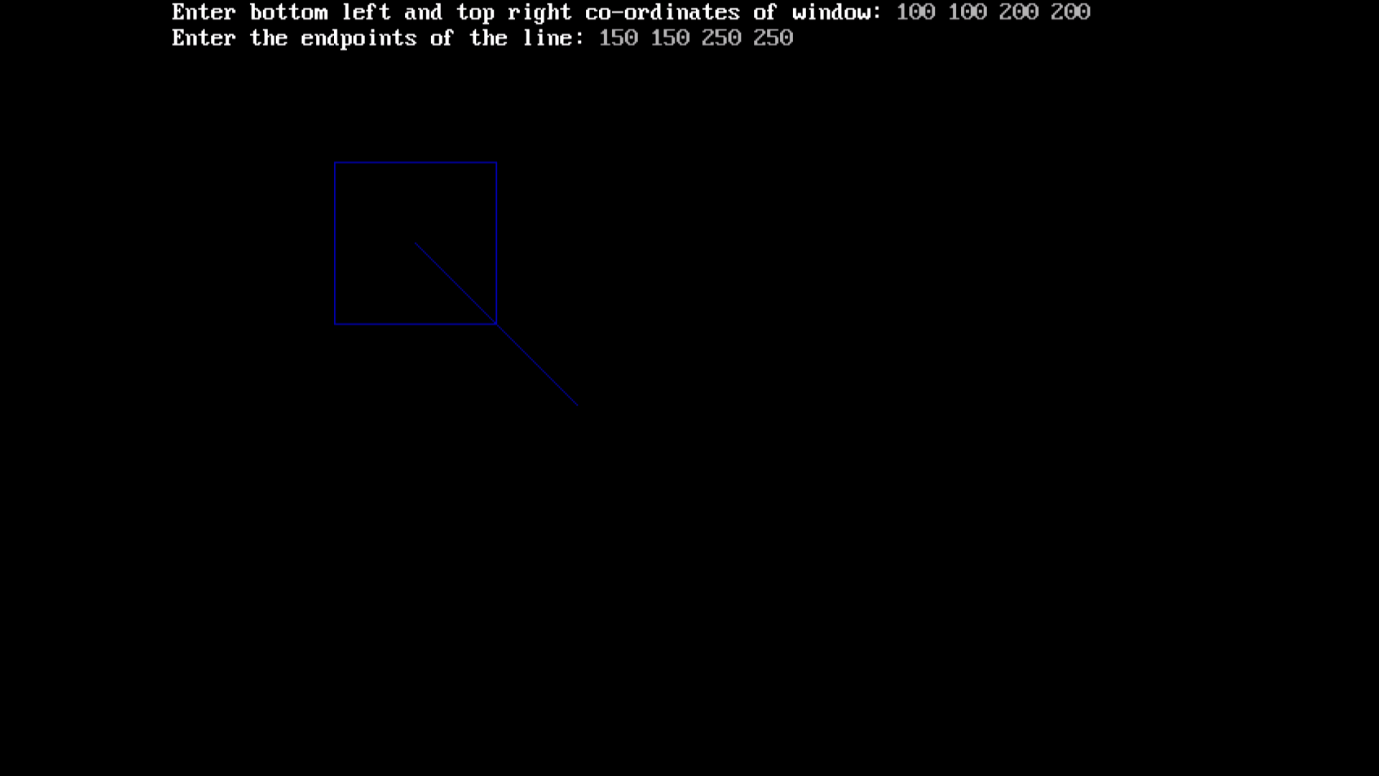
cout<<"Line is out side the clipping window, so clipping is not done,";

getch();

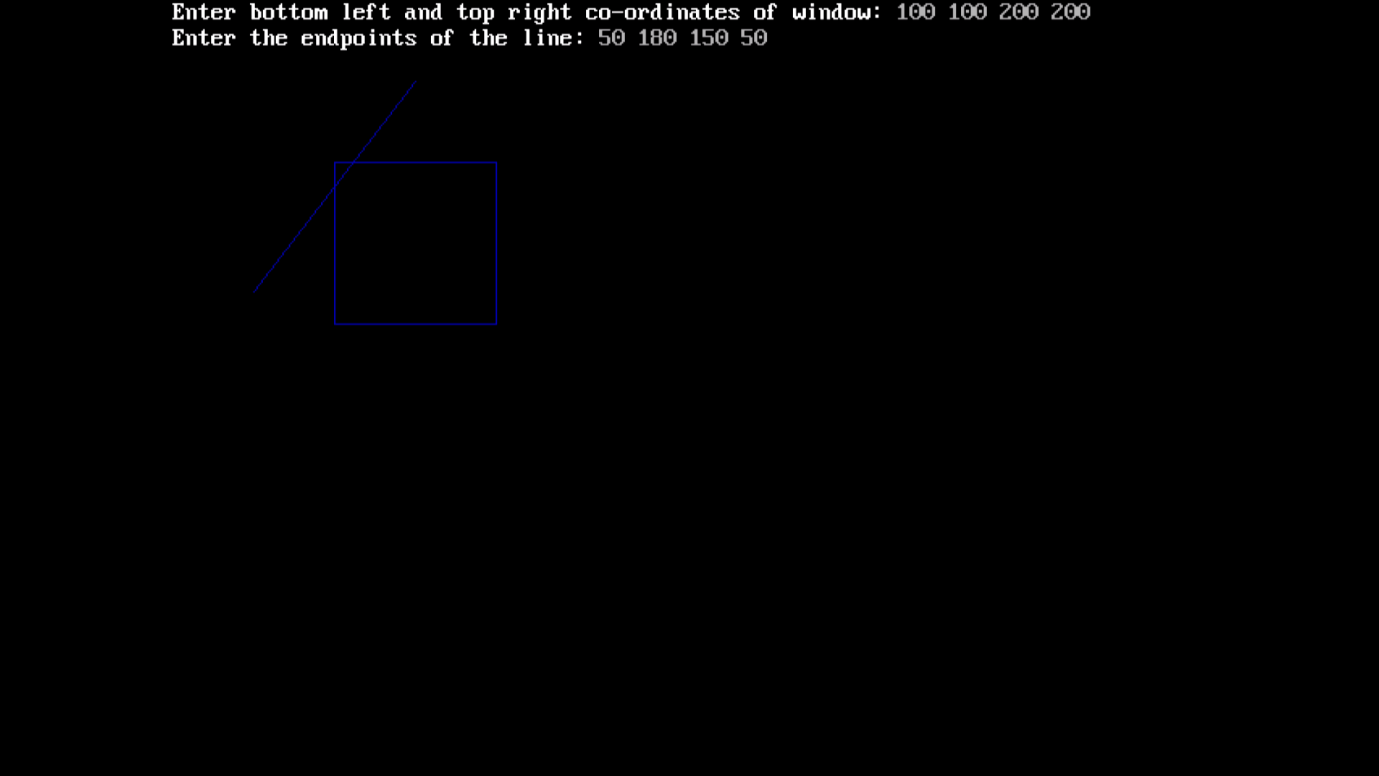
closegraph();

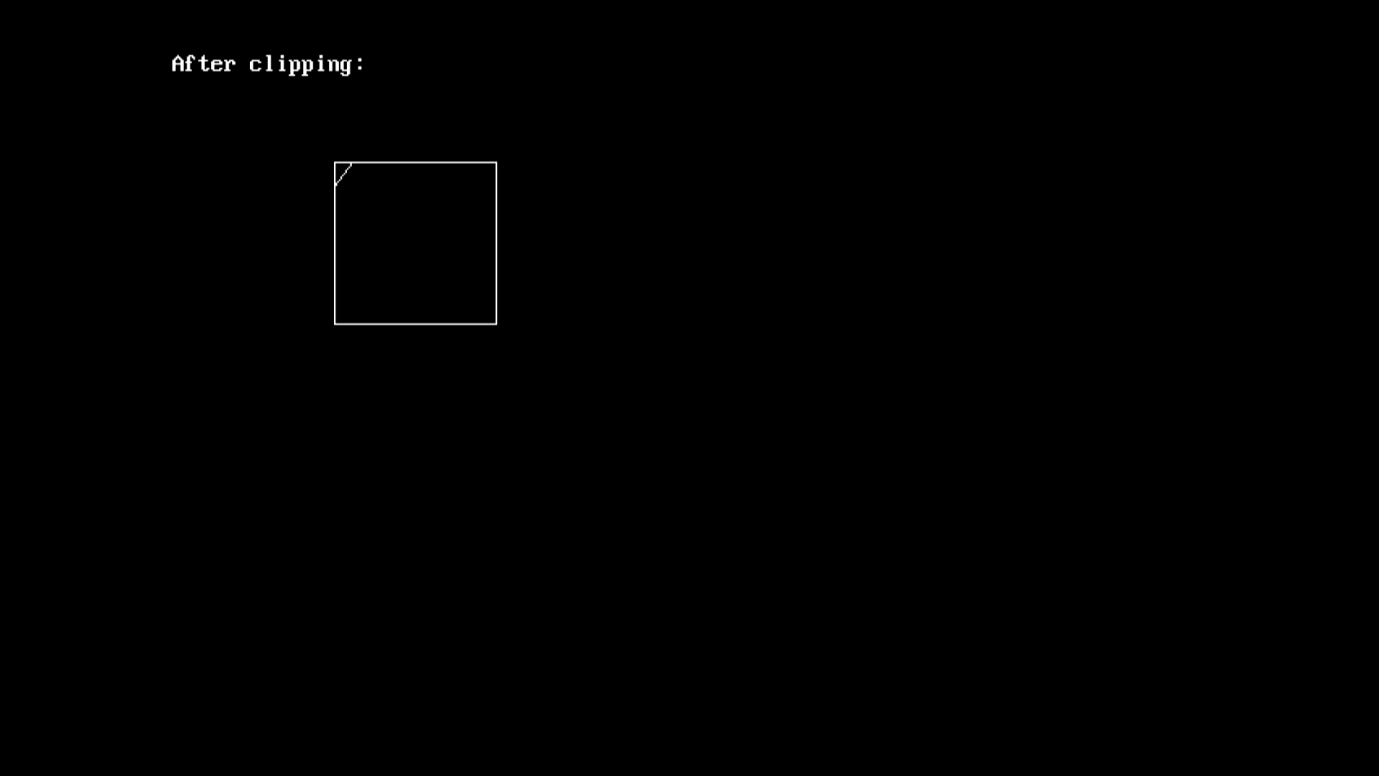
}

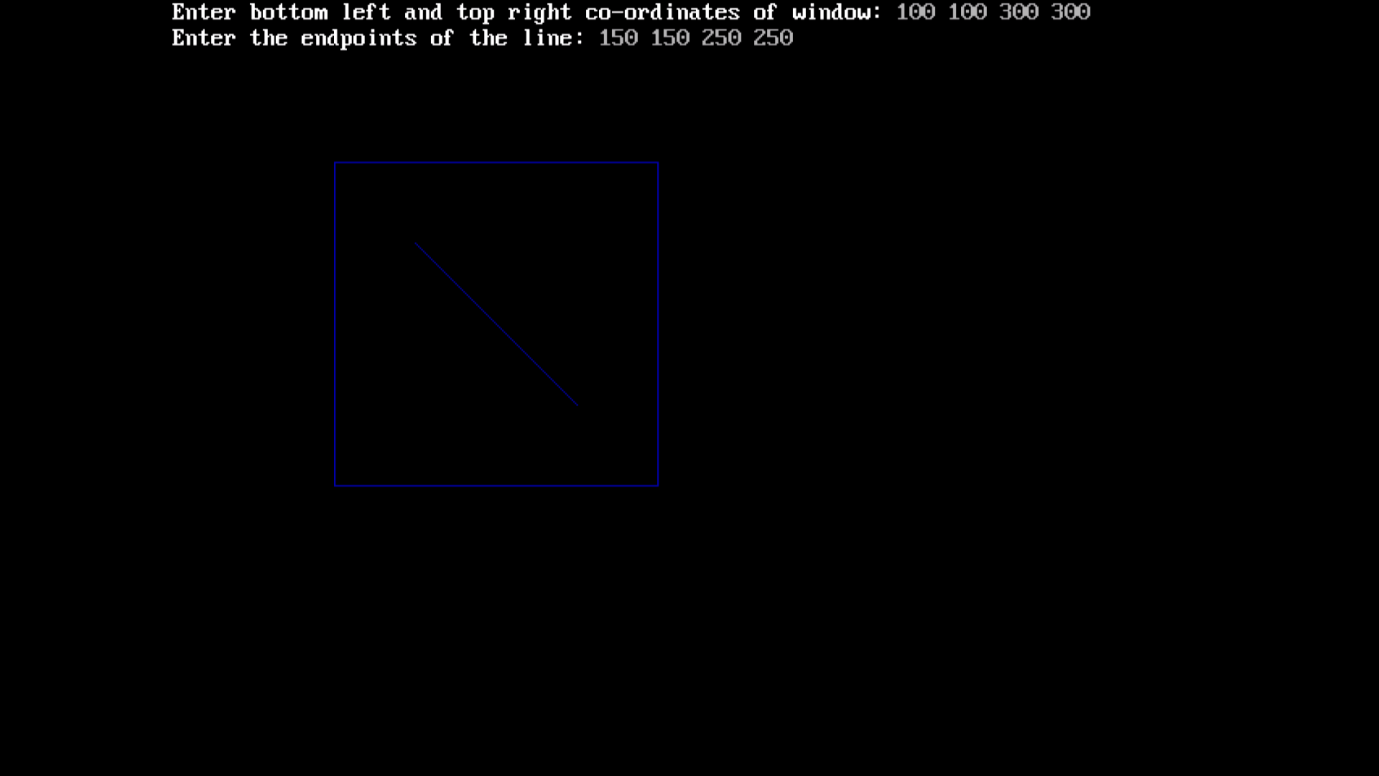
Output:

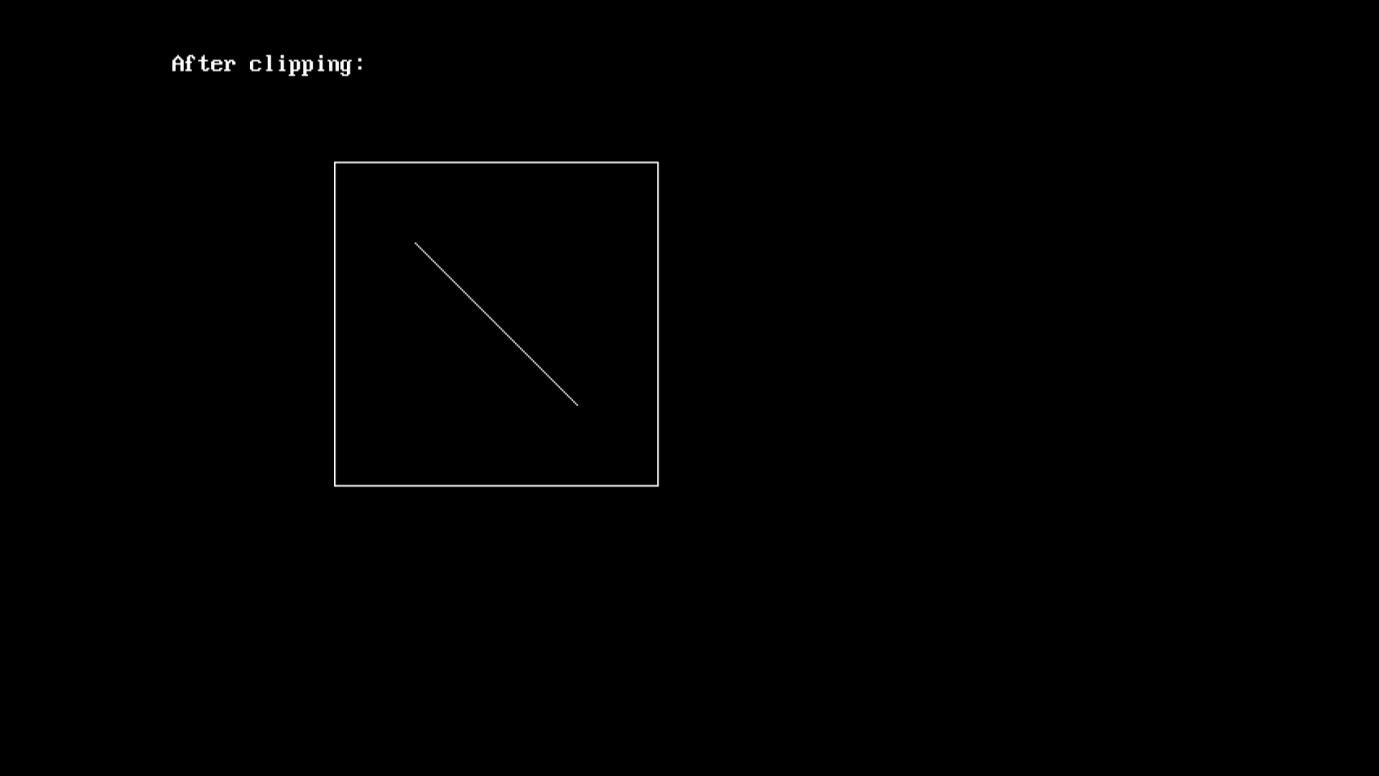


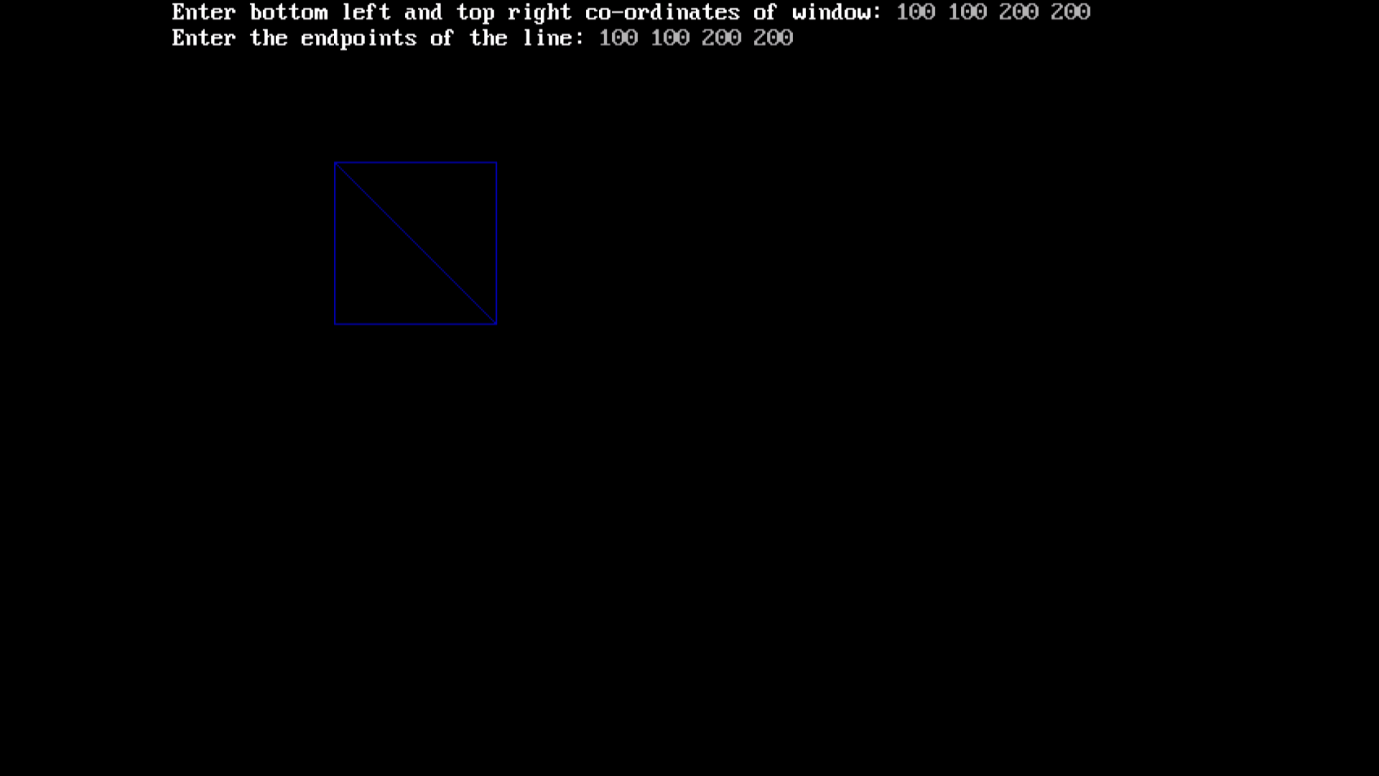


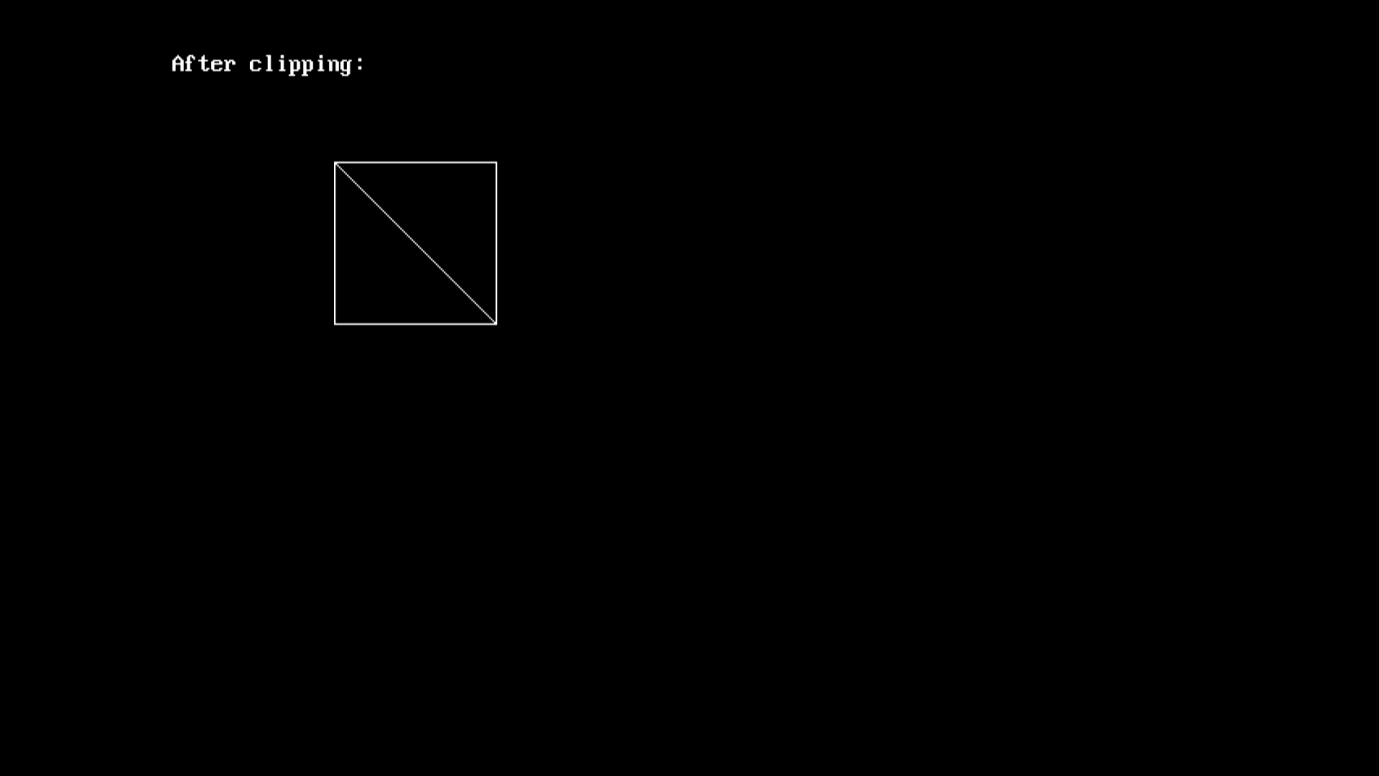


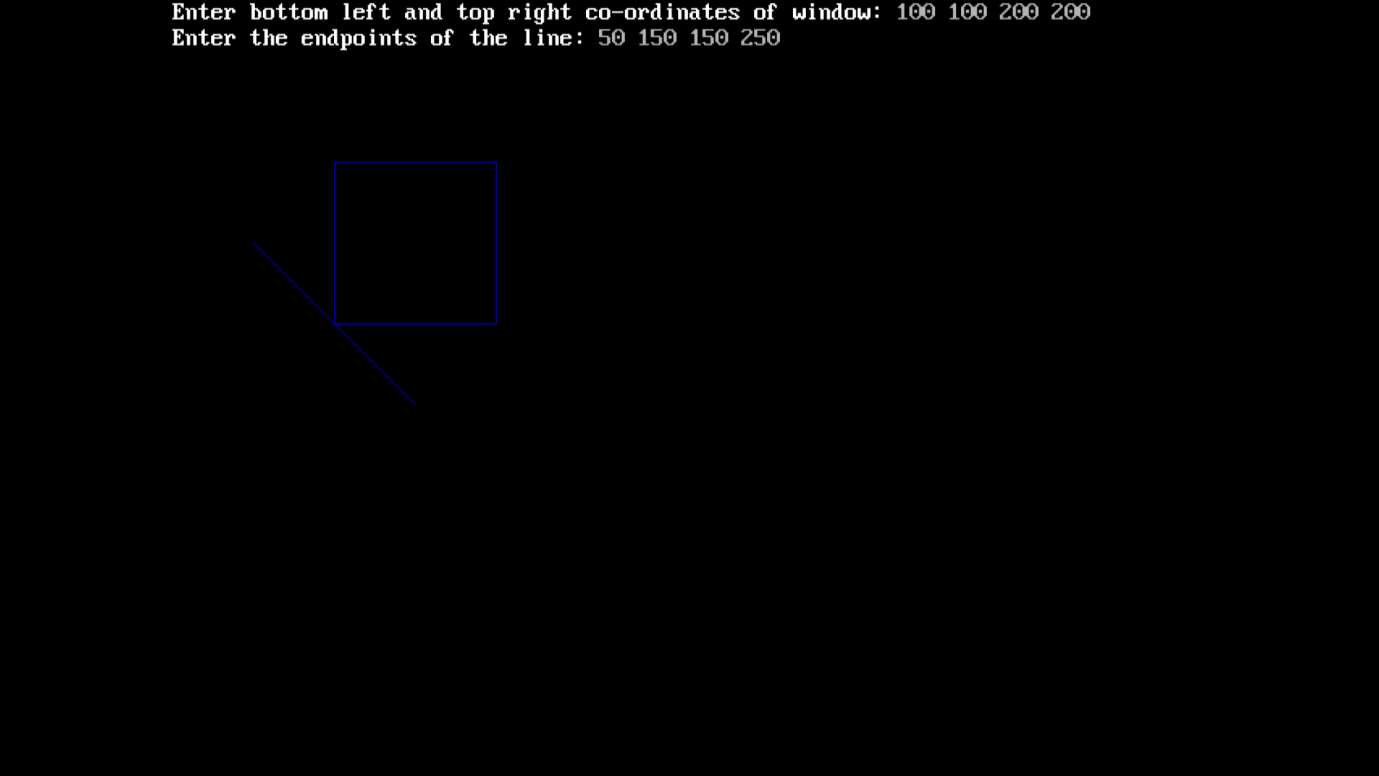




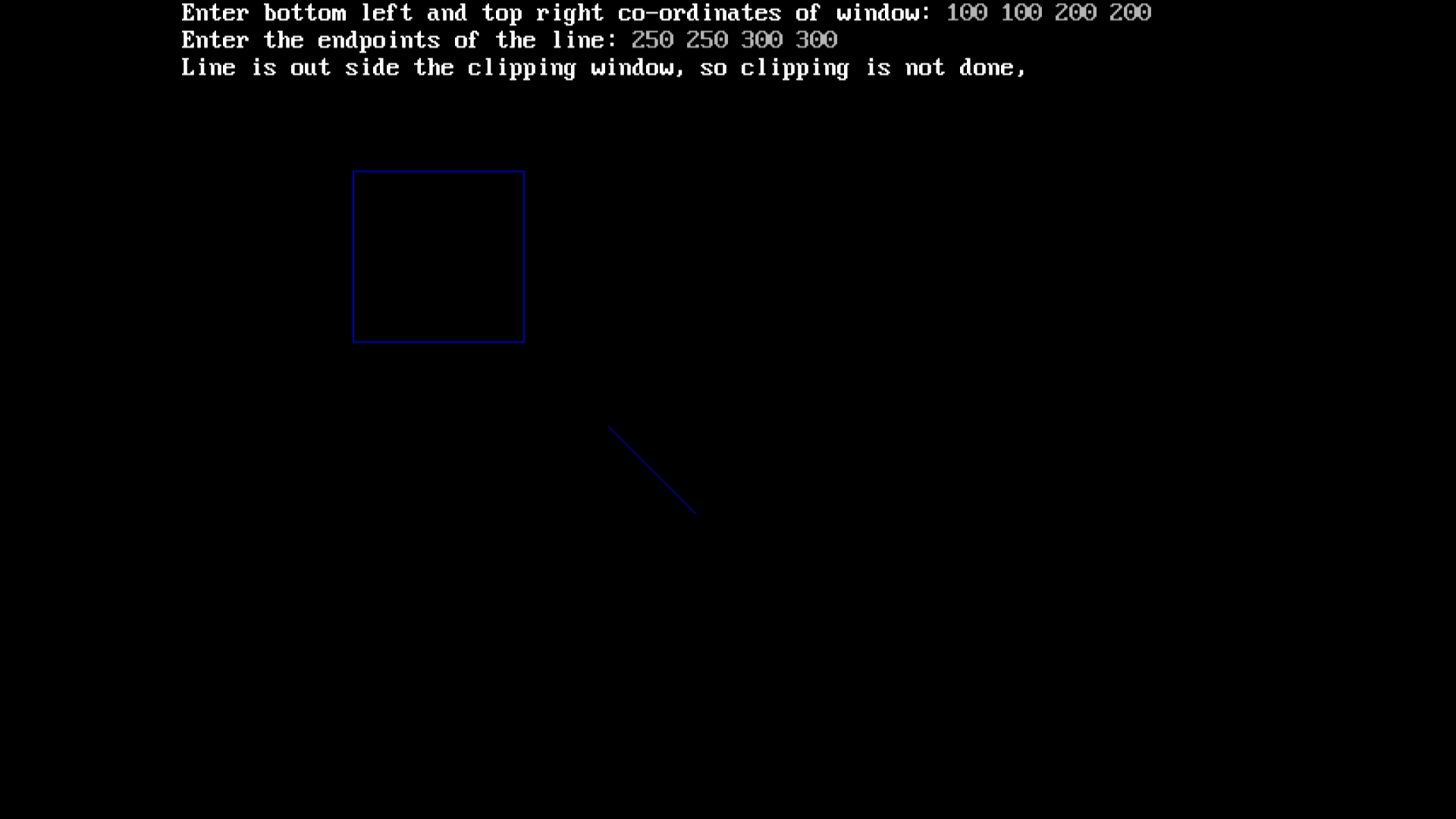












# 5. Program to draw an ellipse using midpoint algorithm.