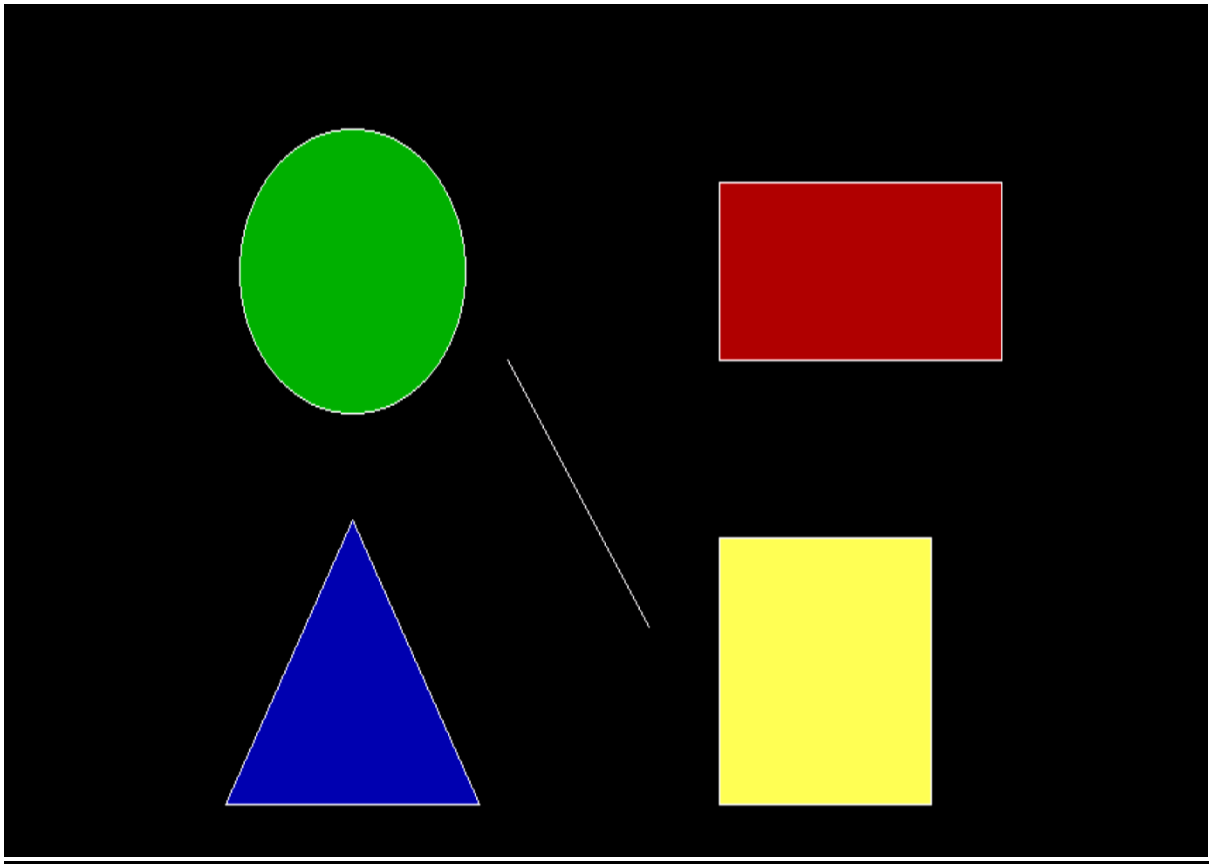
```
/*
1. Write a C++ program for drawing graphics
   primitives and color it.
*/
```

CODE:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdio.h>

int main()
{
    int gdriver = DETECT, gmode;
    initgraph(&gdriver, &gmode,
              "c:\\turboc3\\bgi");

    //LINE
    line(250,200,350,350);

    //CIRCLE
    setfillstyle(SOLID_FILL,GREEN);
    circle(140,150,80);
    floodfill(141,150,WHITE);

    //RECTANGLE
    setfillstyle(SOLID_FILL,RED);
    rectangle(400,100,600,200);
    floodfill(401,110,WHITE);

    //TRIANGLE
    setfillstyle(SOLID_FILL,BLUE);
    line(140,290,50,450);
    line(140,290,230,450);
    line(50,450,230,450);
    floodfill(141,300,WHITE);
```

```
    //SQUARE
    setfillstyle(SOLID_FILL,YELLOW);
    rectangle(400,300,550,450);
    floodfill(401,310,WHITE);

    getch();
    closegraph();
    return 0;
}
```

OUTPUT:

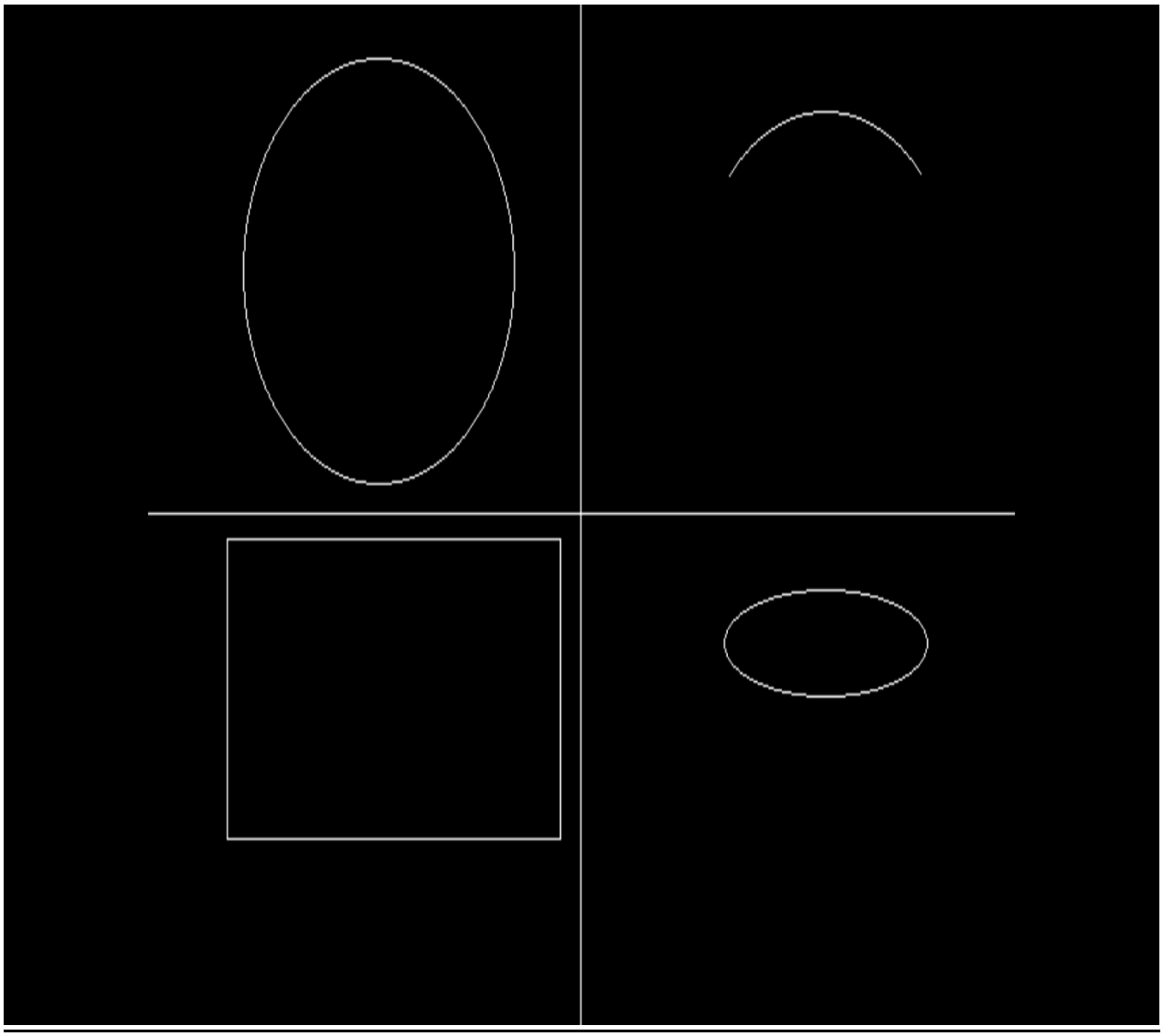


```
/*
2. Write a C++ program to divide screen into
   four region and draw circle,rectangle,arc
   and ellipse.
*/
```

CODE:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdio.h>

int main()
{
    int gdriver = DETECT, gmode;
    int xmax,ymax;
    initgraph(&gdriver, &gmode,
              "c:\\turboc3\\bgi");
    xmax = getmaxx();
    ymax = getmaxy();
    line(xmax/2,0,xmax/2,ymax);
    line(0,ymax/2,xmax,ymax/2);
    circle(170,125,100);
    rectangle(58,251,304,392);
    arc(500,150,45,135,100);
    ellipse(500,300,0,360,75,25);
    getch();
    closegraph();
    return 0;
}
```

OUTPUT:



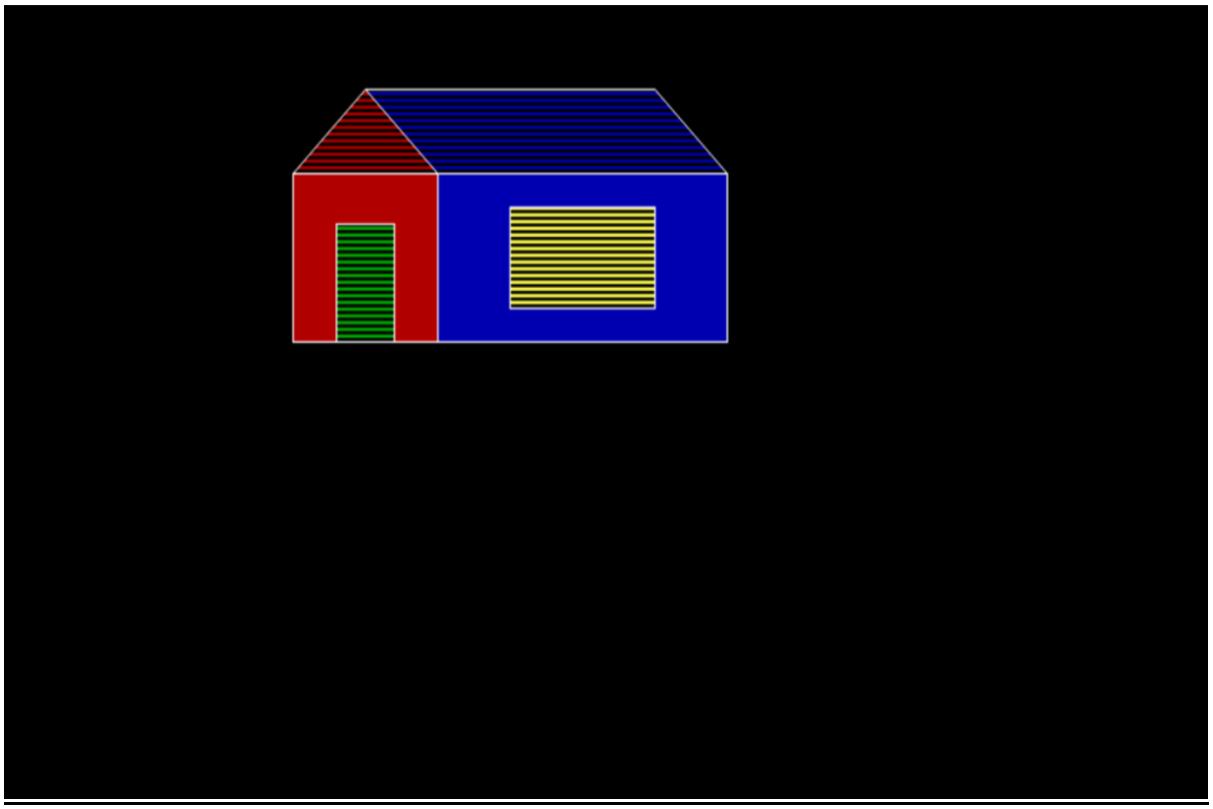
```
/*
3. Write a C++ program for drawing a simple
object.
*/
```

CODE:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
int gdriver=DETECT,gmode;
initgraph(&gdriver,&gmode,"C:\\turboc3\\bgi");

line(100,100,150,50); // two lines forming the
line(150,50,200,100); // triangle shape

line(150,50,350,50); // lines forming the roof
line(350,50,400,100);

setfillstyle(SOLID_FILL,RED);
rectangle(100,100,200,200); // first rectangle
floodfill(101,199,WHITE);

setfillstyle(SOLID_FILL,BLUE);
rectangle(200,100,400,200); // second rectangle
floodfill(201,199,WHITE);

setfillstyle(LINE_FILL,GREEN);
rectangle(130,130,170,200); // door
floodfill(131,169,WHITE);

setfillstyle(LINE_FILL,YELLOW);
rectangle(250,120,350,180); // window
floodfill(251,179,WHITE);
```

```
setfillstyle(LINE_FILL,RED); // filling the
floodfill(149,99,WHITE);     // colour in the
                             // triangle shape


setfillstyle(LINE_FILL,BLUE); // filling the
floodfill(201,99,WHITE);      // colour in the
                              // roof


getch();
closegraph();
}
```

OUTPUT:



```
/*
4. Write a c++ program for drawing a line using
DDA and Bresahnams Line Drawing Algorithm
*/
```

CODE:

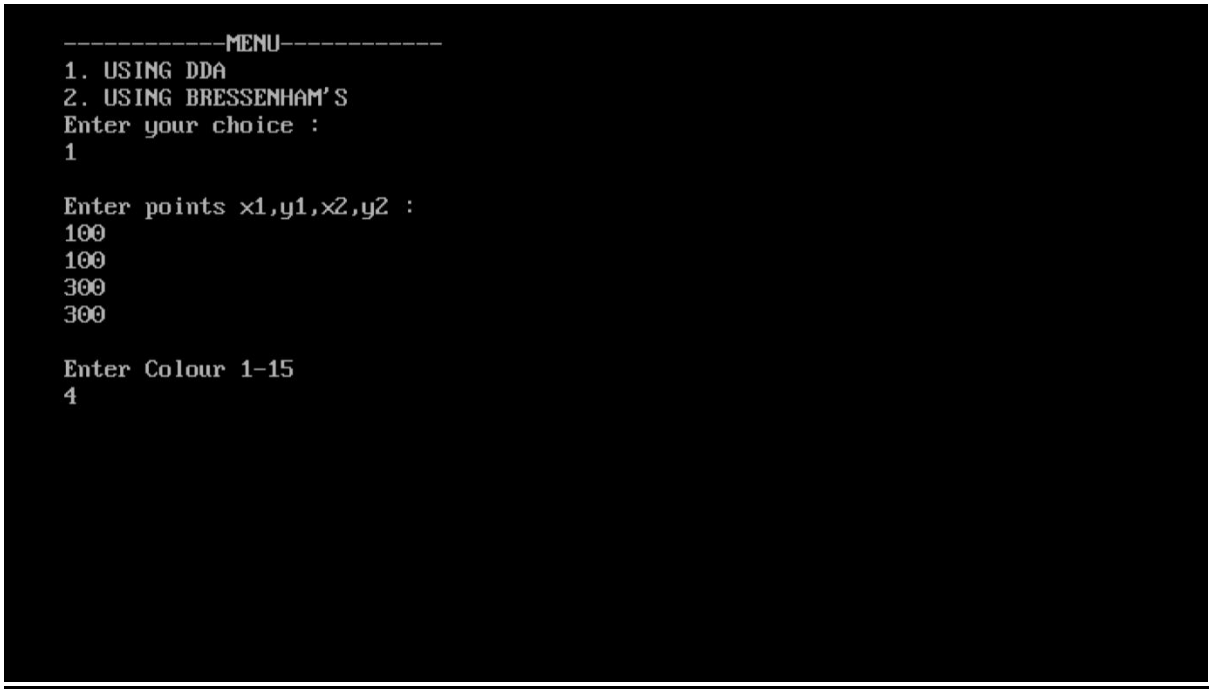
```
#include<iostream.h>
#include<graphics.h>
#include<math.h>


int sign(int x)
{
    if(x<0)
        return -1;
    else if(x>0)
        return 1;
    else
        return 0;
}

void bline(int x1,int y1,int x2,int y2,int col)
{
    int dx,dy,e,x,y,i=1;
    dx=x2-x1;
    dy=y2-y1;
    x=x1;
    y=y1;
    e=2*dy-dx;
    while(i<=dx)
    {
        while(e>=0)
        {
            y++;
            e=e-2*dx;
```

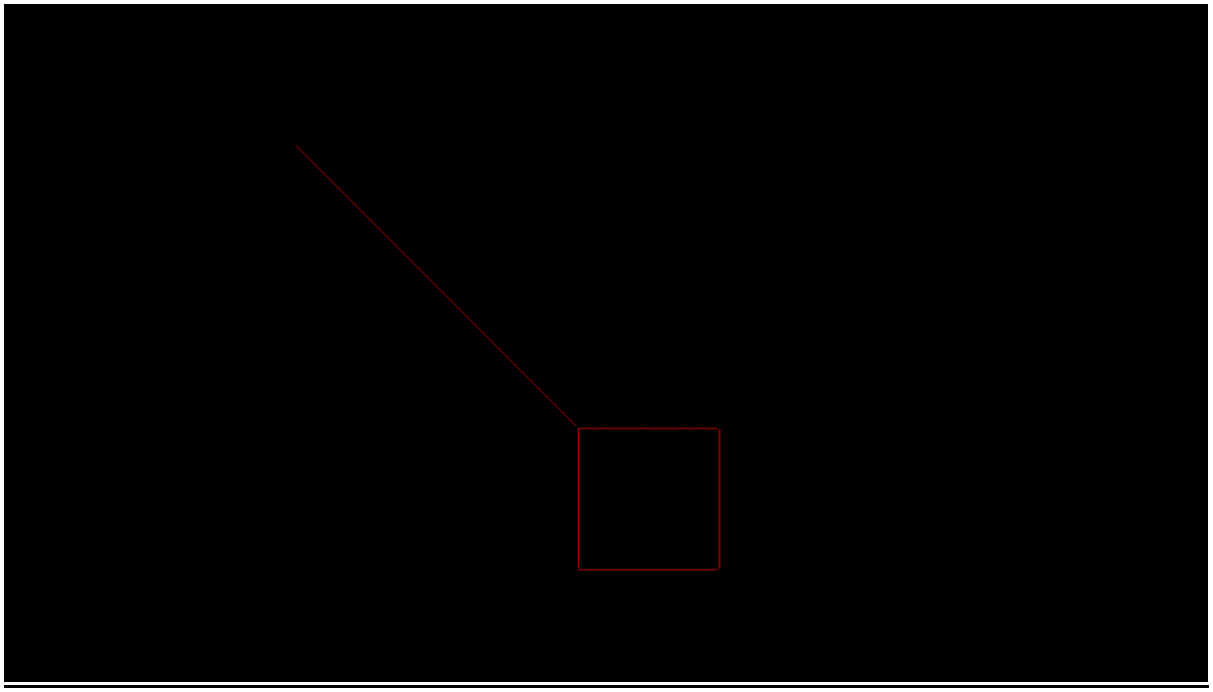
```
            }
            x++;
            e=e+2*dy;
            putpixel(x,y,col);
            i++;
        }
}

void ddaline(int x1,int y1,int x2,int y2,int
col)
{
    int x,y,len,i;
    float dx,dy;
    if(x1==x2 && y1==y2)
        putpixel(x1,y1,col);
    else
    {
        dx=abs(x2-x1);
        dy=abs(y2-y1);
        if(dx&gt;dy)
        len=dx;
        else
            len=dy;
        dx=(x2-x1)/len;
        dy=(y2-y1)/len;
        x=x1+0.5*sign(dx);
        y=y1+0.5*sign(dy);
        i=1;
        while(i<len)
        {
            putpixel(x,y,col);
            x=x+dx;
            y=y+dy;
            i++;
        }
    }
}
```

```
int main()
{
    int ch,col,x1,x2,y1,y2;
    cout<<"\n------------MENU------------\n"
    cout<<"1.USING DDA\n";
    cout<<"2.Using Bresahnams\n";

    cout<<"\nEnter your choice : \n";
    cin>>ch;
    cout<<"\nEnter points x1,y1,x2,y2 : \n";
    cin>>x1>>y1>>x2>>y2;
    cout<<"\nEnter colour 1-15 : \n";
    cin>>col;
    if(col>15||col<1)
        col=1;
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"c:\\turboc3\\bgi");

    switch(ch)
    {
        case 1:
              ddaline(x1,y1,x2,y2,col);
              ddaline(300,300,400,300,col);
              ddaline(300,300,300,400,col);
              ddaline(300,400,400,400,col);
              ddaline(400,400,400,300,col);
              break;

        case 2:
              bline(x1,y1,x2,y2,col);
              ddaline(300,300,400,300,col);
              ddaline(300,300,300,400,col);
              ddaline(300,400,400,400,col);
              ddaline(400,400,400,300,col);
              break;

        default :
             cout<<"\nEnter valid choice :\n";
```

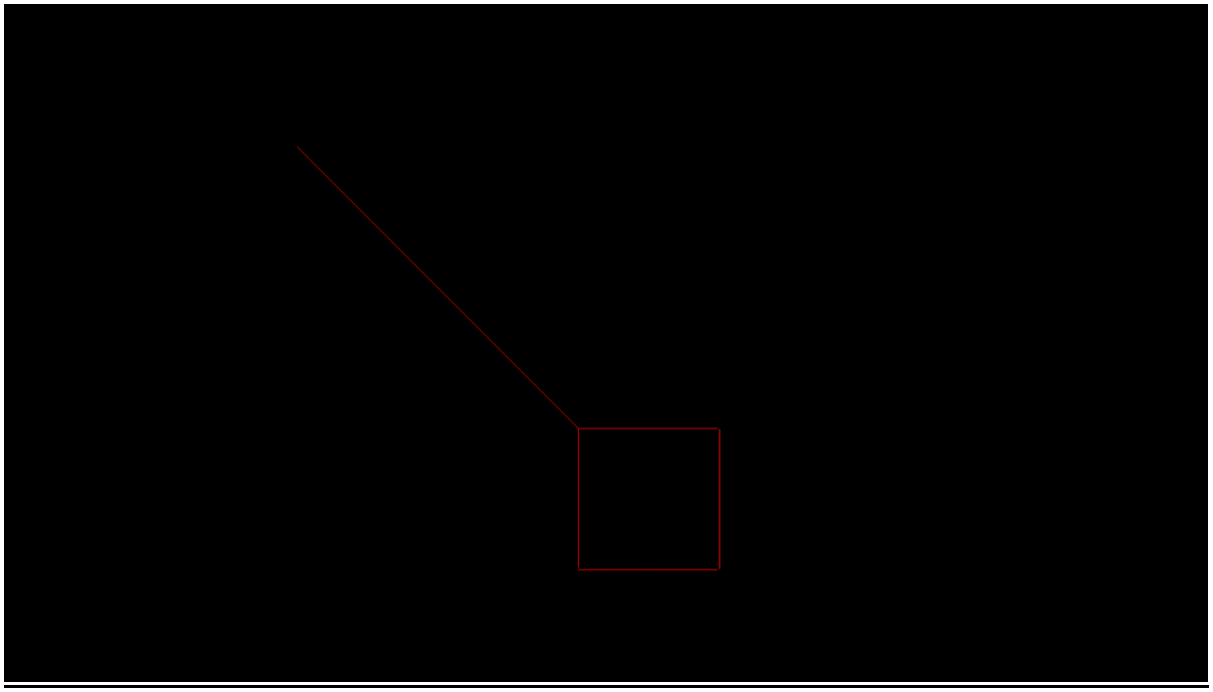
```
    }
    getch();
    closegraph();
    return 0;
}
```

OUTPUT:





```
------------MENU------------
1. USING DDA
2. USING BRESSENHAM'S
Enter your choice :
2

Enter points x1,y1,x2,y2 :
100
100
300
300

Enter Colour 1-15
4
```



```
/*
5. A) Write a C++ program for drawing a
following pattern(diamond in rectangle)
*/
```

CODE:

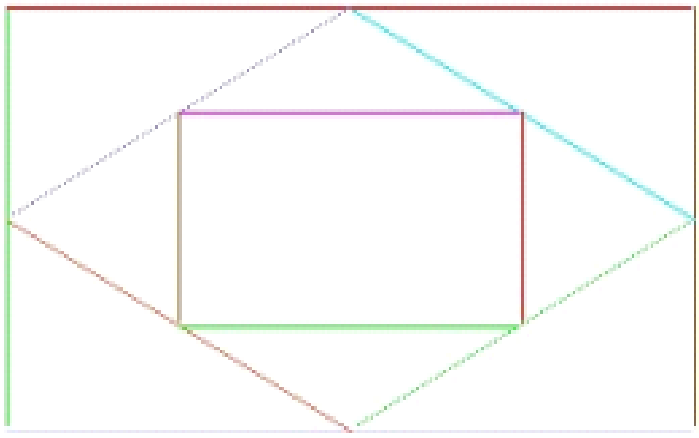
```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>

int sign(int x)
{
```

```
    if(x<0)
        return -1;
    else if(x>0)
        return 1;
    else
        return 0;
}

void bline(int x1,int y1,int x2,int y2,int col)
{
    int dx,dy,e,x,y,i=1;
    dx=x2-x1;
    dy=y2-y1;
    x=x1;
    y=y1;
    e=2*dy-dx;
    while(i<=dx)
    {
        while(e>=0)
        {
            y++;
            e=e-2*dx;
        }
        x++;
        e=e+2*dy;
        putpixel(x,y,col);
        i++;
    }
}

void ddaline(int x1,int y1,int x2,int y2,int
col)
{
    int x,y,len,i;
    float dx,dy;
    if(x1==x2 && y1==y2)
    putpixel(x1,y1,col);
    else
```

```
    {
        dx=x2-x1;
        dy=y2-y1;
        if(dx>dy)
            len=dx;
        else
        len=dy;
        dx=(x2-x1)/len;
        dy=(y2-y1)/len;
        x=x1+0.5*sign(dx);
        y=y1+0.5*sign(dy);
        i=1;
        while(i<len)
        {
            putpixel(x,y,col);
            x=x+dx;
            y=y+dy;
            i++;
        }
    }
}

int main()
{
    int ch,col,x1,x2,y1,y2;
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"c:\\turboc3\\bgi");
    setbkcolor(WHITE);

    ddaline(50,50,50,200,2); //left vert
    ddaline(50,50,350,50,4); //up horizontal
    ddaline(350,50,350,200,6); //right vert
    ddaline(50,200,350,200,7); //down
horizontal
    ddaline(200,50,50,125,9); //diamond up left
    bline(50,125,200,200,12); //diamond
left,down
    ddaline(350,125,200,200,14);//diamond
```

```
down,right
    bline(200,50,350,125,3); //diamond right,up
    ddaline(275,87,275,163,4);//in right
    ddaline(125,87,275,87,5);//in up
    ddaline(125,87,125,163,6);//in left
    ddaline(125,163,275,163,2);//in down

    getch();
    closegraph();
    return 0;
}
```

OUTPUT:



```
/*
5. B) Write a c++ program inscribed and
circumscribed circles in triangle
```

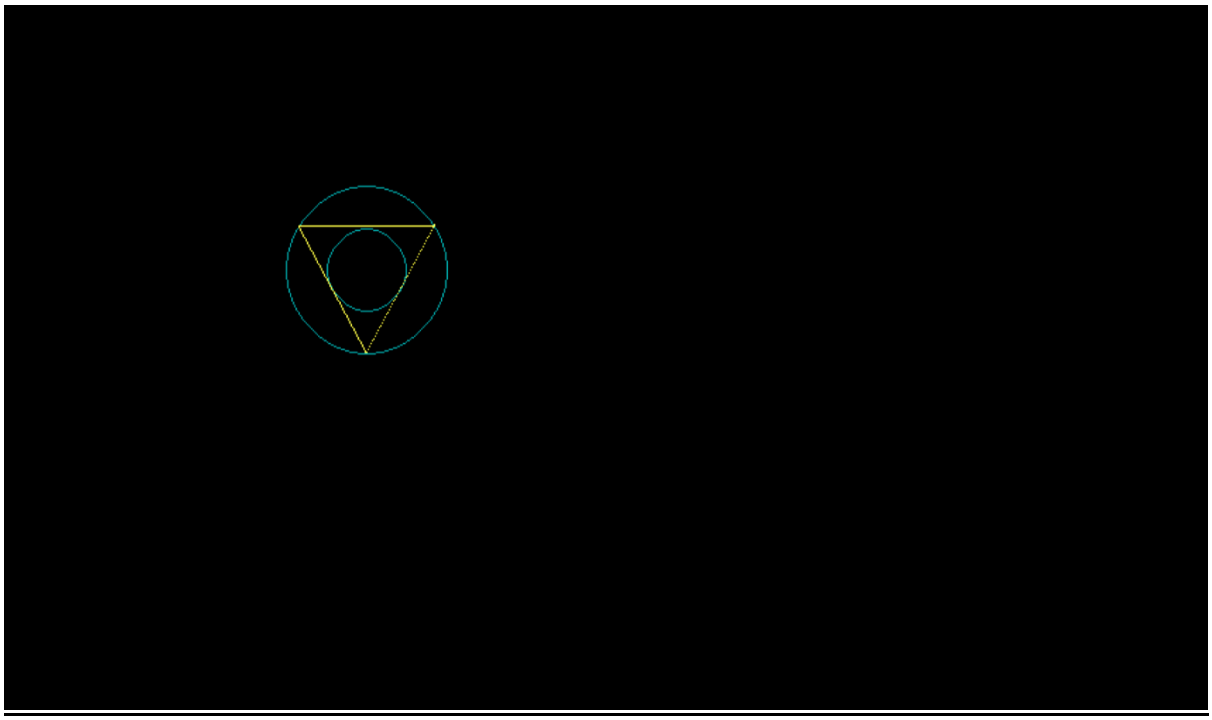
CODE:

```
#include<iostream.H>
#include<graphics.h>
#include<stdio.h>

void ddaAlg(int x1,int y1,int x2,int y2)
{
    int dx=x2-x1;
    int dy=y2-y1;
    int steps=dx>dy?dx:dy;
    float xInc=dx/(float)steps;
    float yInc=dy/(float)steps;
    float x=x1;
    float y=y1;
    for(int i=0;i<=steps;i++)
    {
        putpixel(x,y,14);
        x+=xInc;
        y+=yInc;
    }
}

void display(int xc,int yc,int x,int y)
{
    putpixel(xc+x, yc+y, 3);
    putpixel(xc-x, yc+y, 3);
    putpixel(xc+x, yc-y, 3);
    putpixel(xc-x, yc-y, 3);
    putpixel(xc+y, yc+x, 3);
    putpixel(xc-y, yc+x, 3);
```

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

void CircleB(int x1,int y1,int r)
{
    int x=0,y=r;
    int d=3-2*r;
    display(x1,y1,x,y);
    while(y>=x)
    {
        x++;
        if(d>0)
        {
            y--;
            d=d+4*(x-y)+10;
        }
        else
        {
            d=d+4*x+6;
        }
        display(x1,y1,x,y);
    }
}

int main()
{
    int gd=DETECT, gm;
    initgraph(&gd,&gm,"c:\\turboc3\\bgi");

    CircleB(150,180,57);
    CircleB(150,180,57/2);
    ddaAlg(102,150,198,150);
    ddaAlg(102,150,150,236);
    ddaAlg(150,236,198,150);

    getch();
    closegraph();
```

```
    return 0;
}
```

OUTPUT:



```
/* Write C++ program to draw a concave
polygon and fill it with desired color
using scan fill algorithm.
*/
```

CODE:

```
#include <conio.h>
#include <iostream>
#include <graphics.h>
#include <stdlib.h>
using namespace std;

class point
{
    public:
    int x,y;
};

class poly
{
    private:
        point p[20];
        int inter[20],x,y;
        int v,xmin,ymin,xmax,ymax;
    public:
        int c;
        void read();
        void calcs();
        void display();
        void ints(float);
        void sort(int);
};
```

```
void poly::read()
{
    int i;
    cout<<"\n\t SCAN_FILL ALGORITHM";
    cout<<"\n Enter the no of vertices
of polygon:";
    cin>>v;
    if(v>2)
    {
        for(i=0;i<v; i++)
        {
            cout<<"\nEnter the co-
ordinate no.- "<<i+1<<" : ";
            cout<<"\n\tx"<<(i+1)<<"=";
            cin>>p[i].x;
            cout<<"\n\ty"<<(i+1)<<"=";
            cin>>p[i].y;
        }
        p[i].x=p[0].x;
        p[i].y=p[0].y;
        xmin=xmax=p[0].x;
        ymin=ymax=p[0].y;
    }
    else
        cout<<"\n Enter valid no. of
vertices.";
}

void poly::calcs()
{ //MAX,MIN
    for(int i=0;i<v;i++)
    {
        if(xmin>p[i].x)
```

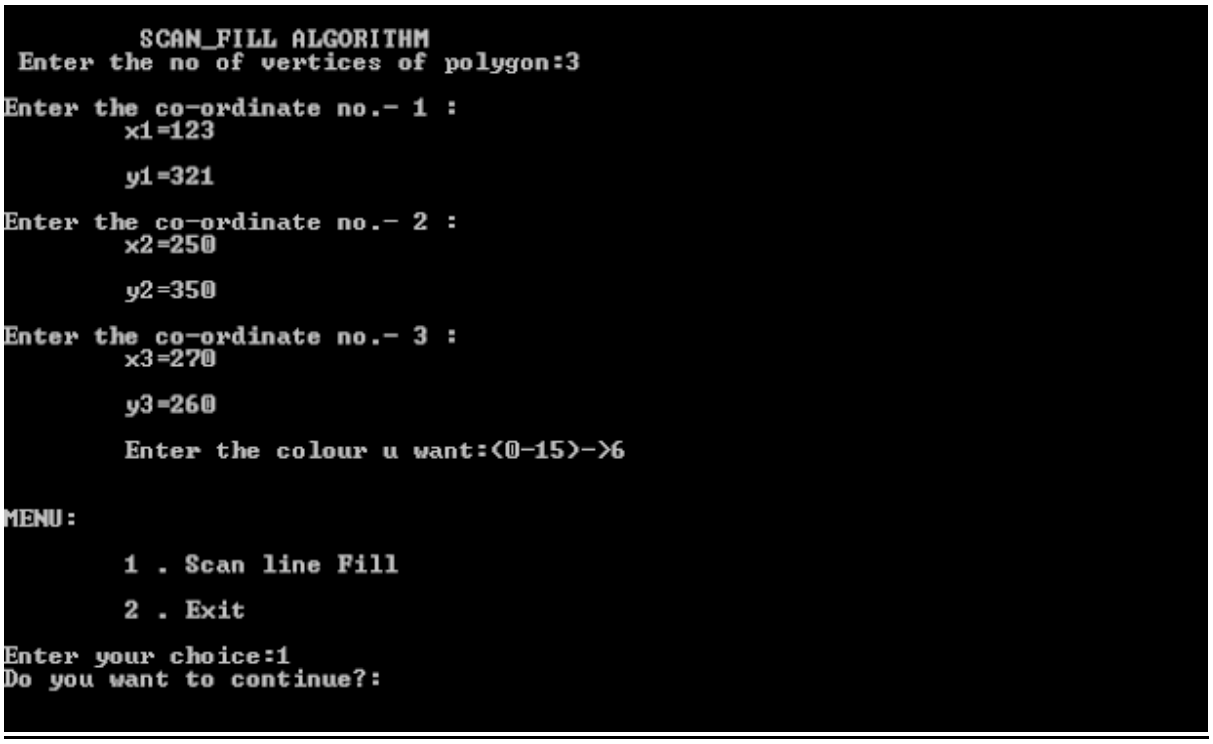
```
            xmin=p[i].x;
            if(xmax<p[i].x)
            xmax=p[i].x;
            if(ymin>p[i].y)
            ymin=p[i].y;
            if(ymax<p[i].y)
            ymax=p[i].y;
        }
}

void poly::display()
{
    int ch1;
    char ch='y';
    float s,s2;
    do
    {
        cout<<"\n\nMENU:";
        cout<<"\n\n\t1 . Scan line Fill
";
        cout<<"\n\n\t2 . Exit ";
        cout<<"\n\nEnter your choice:";
        cin>>ch1;
        switch(ch1)
        {
            case 1:
                s=ymin+0.01;
                delay(100);
                cleardevice();
                while(s<=ymax)
                {
                    ints(s);
                    sort(s);
                    s++;
```

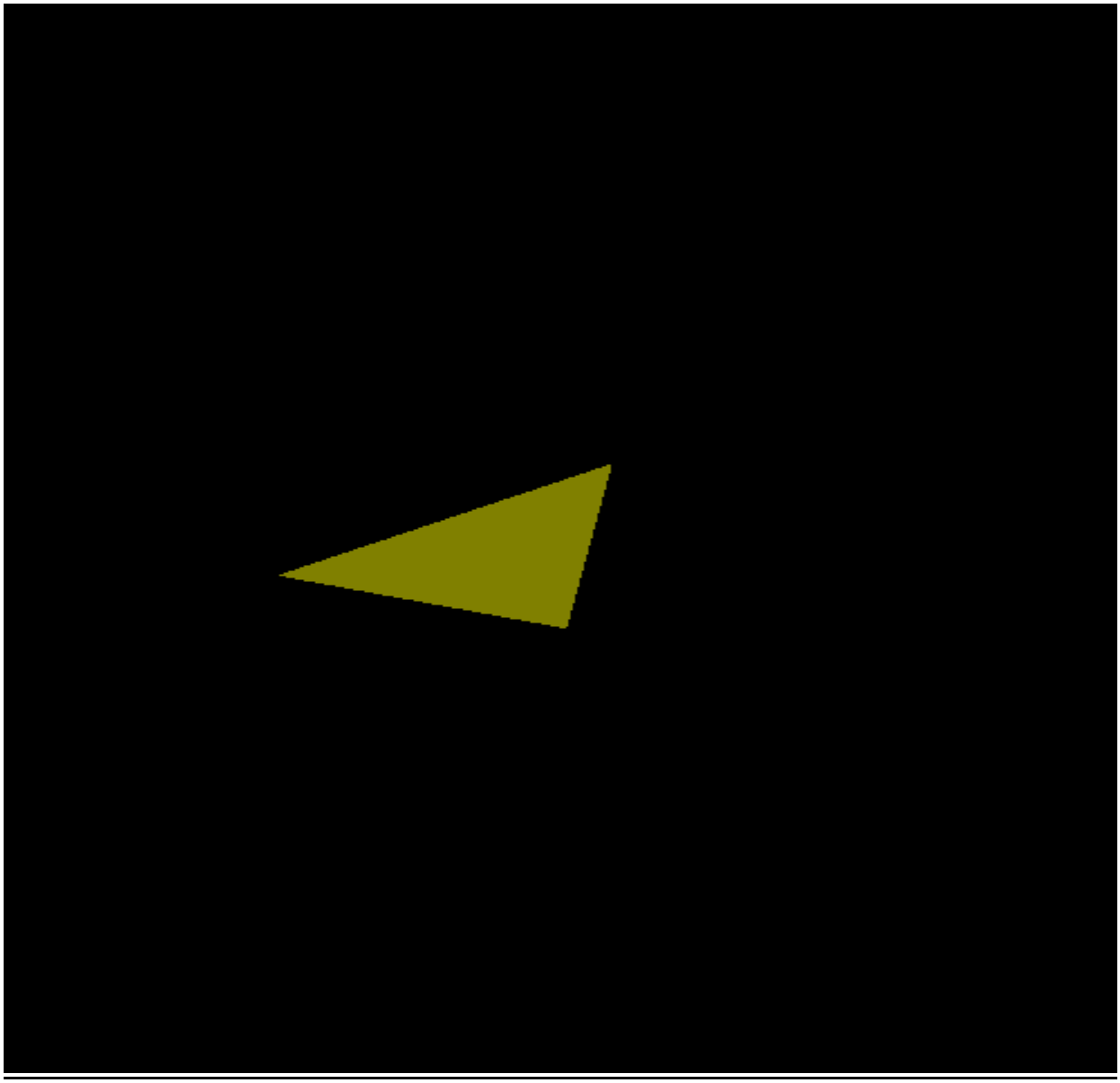
```
                }
                break;
            case 2:
                exit(0);
        }

        cout<<"Do you want to
continue?: ";
        cin>>ch;
    }while(ch=='y' || ch=='Y');
}

void poly::ints(float z)
{
    int x1,x2,y1,y2,temp;
    c=0;
    for(int i=0;i<v;i++)
    {
        x1=p[i].x;
        y1=p[i].y;
        x2=p[i+1].x;
        y2=p[i+1].y;
        if(y2<y1)
        {
            temp=x1;
            x1=x2;
            x2=temp;
            temp=y1;
            y1=y2;
            y2=temp;
        }
        if(z<=y2&&z>=y1)
        {
            if((y1-y2)==0)
```

```
            x=x1;
            else
            {
                x=((x2-x1)*(z-y1))/(y2-
y1);
                x=x+x1;
            }
            if(x<=xmax && x>=xmin)
            inter[c++]=x;
        }
    }
}

void poly::sort(int z)
{
    int temp,j,i;

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

line(p[i].x,p[i].y,p[i+1].x,p[i+1].y);
        }
        delay(100);
        for(i=0; i<c;i+=2)
        {
            delay(100);

line(inter[i],z,inter[i+1],z);
        }
}

int main()
{
    int cl;
```

```
    initwindow(500,600);
    cleardevice();
    poly x;
    x.read();
    x.calcs();
    cleardevice();
    cout<<"\n\tEnter the colour u
want:(0-15)->"; //Selecting colour
    cin>>cl;
    setcolor(cl);
    x.display();
    closegraph(); //CLOSE OF GRAPH
    getch();
    return 0;
}
```

OUTPUT:





```
/*7. Write C++ program to implement
Cohen Southerland line clipping
algorithm.
*/
```

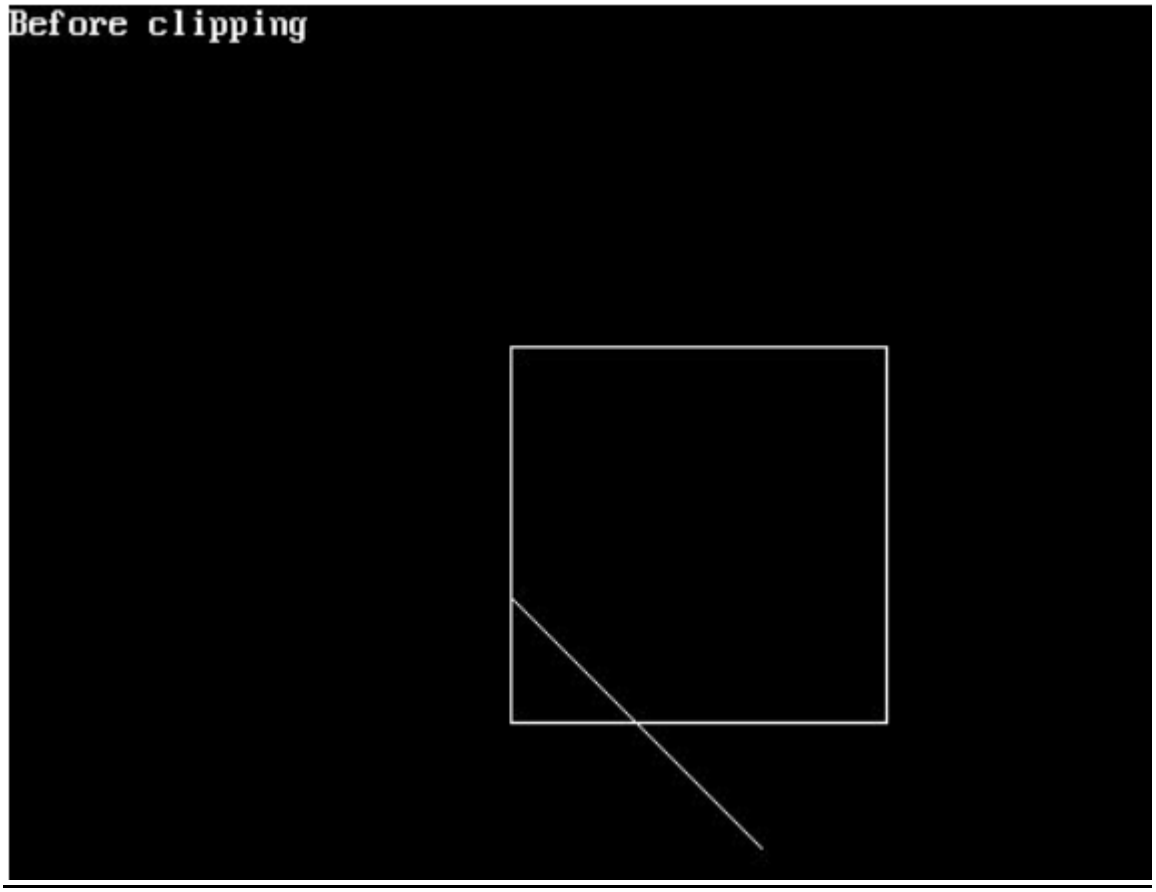
CODE:

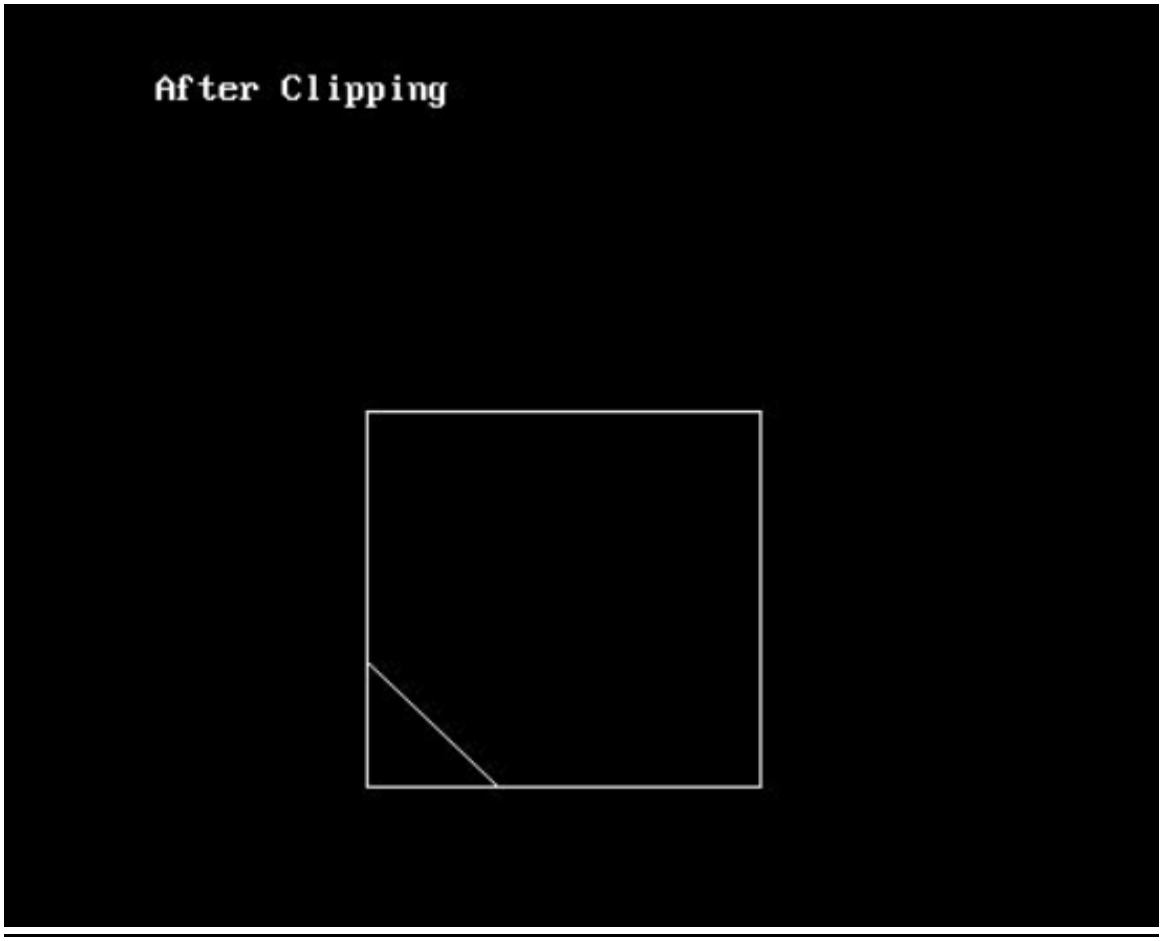
```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void Window()
{
    line (200,200,350,200);
    line(350,200,350,350);
    line(200,200,200,350);
    line(200,350,350,350);
}
void Code(char c[4],float x,float y)
{         c[0]=(x<200)?'1':'0';
     c[1]=(x>350)?'1':'0';
     c[2]=(y<200)?'1':'0';
     c[3]=(y>350)?'1':'0';
}
void Clipping  (char c[],char d[],float
&x,float &y,float m)
{
    int flag=1,i=0;
    for (i=0;i<4;i++)
    {
        if(c[i]!='0' && d[i]!='0')
        {
            flag=0;
```

```
            break;
        }
        if(flag)
        {
            if(c[0]!='0')
            {
                y=m*(200-x)+y;
                x=200;
            }
            else if(c[1]!='0')
            {
                y=m*(350-x)+y;
                x=350;
            }
            else if(c[2]!='0')
            {
                x=((200-y)/m)+x;
                y=200;
            }
            else if(c[3]!='0')
            {
                x=((350-y)/m)+x;
                y=350;
            }
        }
        if (flag==0)
            cout<<"Line lying outside";
    }
}
void main()
{
int gdriver = DETECT, gmode, errorcode;
float x1,y1,x2,y2;
float m;
```

```
char c[4],d[4];
clrscr();
initgraph(&gdriver, &gmode,
"//Turboc3//bgi");
cout<<"Enter coordinates";
cin>>x1>>y1>>x2>>y2;
cout<<"Before clipping";
Window();
line(x1,y1,x2,y2);
getch();
cleardevice();
m=float((y2-y1)/(x2-x1));
Code(c,x1,y1);
Code(d,x2,y2) ;
Clipping(c,d,x1,y1,m);
Clipping(d,c,x2,y2,m);
cout<<"After Clipping";
Window();
line(x1,y1,x2,y2);
getch();
closegraph();
}
```

OUTPUT:




After Clipping