Department of Computer Science and Engineering

University of Rajshahi

Course Code : CSE3221

Course Title : Computer Graphics

Submitted By:	Submitted To:
Name : Mst.Jannatun Ferdous Student ID/Roll : 1912076147 Session : 2018-19	Name: Abu Raihan Shoyeb Ahmed Siddique Professor Department of CSE University of Rajshahi

Submission Date : 10 July,2023

Submission Deadline : 10 July,2023

1. National Flag Algorithm C++ Code

```
#include<graphics.h>
#include<bits/stdc++.h>
using namespace std;
int main()
{
    cout<<"Enter Coordinate axes: ";
    int flag_x1 ;
    int flag_y1;
    cin>>flag_x1>>flag_y1;
    getchar();
    int times = 20;
    // Redundadnt
    int flag_x2 = flag_x1+10*times;
    int flag_y2 = flag_y1+6*times;

    cout << flag_x2 <<" "<<flag_y2<<endl;
    int gd,gm;
    detectgraph(&gd,&gm);
    initgraph(&gd,&gm,"");
    setcolor(GREEN);
    rectangle(flag_x1,flag_y1,flag_x2,flag_y2);
    setfillstyle(SOLID_FILL,GREEN);
    floodfill((flag_x1+1),(flag_y1+1),GREEN);

    setcolor(RED);
    circle((flag_x1+(flag_x2-flag_x1)*0.45),(flag_y1+flag_y2)/2,(flag_y2-flag_y1)*.4);
    setfillstyle(SOLID_FILL,RED);
    floodfill((flag_x1+(flag_x2-flag_x1)*0.45),(flag_y1+flag_y2)/2,RED);
    getchar();

}
```

Input & Output:

2. Hidden Surface Elimination Algorithm C++ Code

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
void circle()
{
   setcolor(BLUE);
   circle(150,150,30);
   setfillstyle(SOLID_FILL,BLUE);
   floodfill(151,151,BLUE);
}
void rectangle()
{
```

```
    setcolor(GREEN);
    rectangle(80,80,150,130);
    setfillstyle(SOLID_FILL,GREEN);
    floodfill(101,101,GREEN);
}

void triangle()
{
    setcolor(RED);
    line(100,100,200,100);
    line(100,100,150,150);
    line(200,100,150,150);
    setfillstyle(SOLID_FILL,RED);
    floodfill(150,120,RED);
}

int main()
{
    int gd,gm;
    detectgraph(&gd,&gm);
    initgraph(&gd,&gm,"");

    //string str="RCT";
    string str;
    char ch;
    cout <<"Type C for circle,T for triangle and R for rectangle"<<endl;
    for(int i=0; i<3; i++)
    {
        if(i==0)
            cout <<"First one : ";
        else if(i==1)
            cout <<"Second one : ";
        else
            cout <<"Third one : ";

        cin >> ch;
        str+=ch;
    }
    getchar();
```

```
    for(int i=0; i<3; i++)
    {
        if(str[i]=='C')
            circle();
        else if(str[i]=='R')
            rectangle();
        else if(str[i]=='T')
            triangle();
    }
    getchar();
}
```

Input & Output:

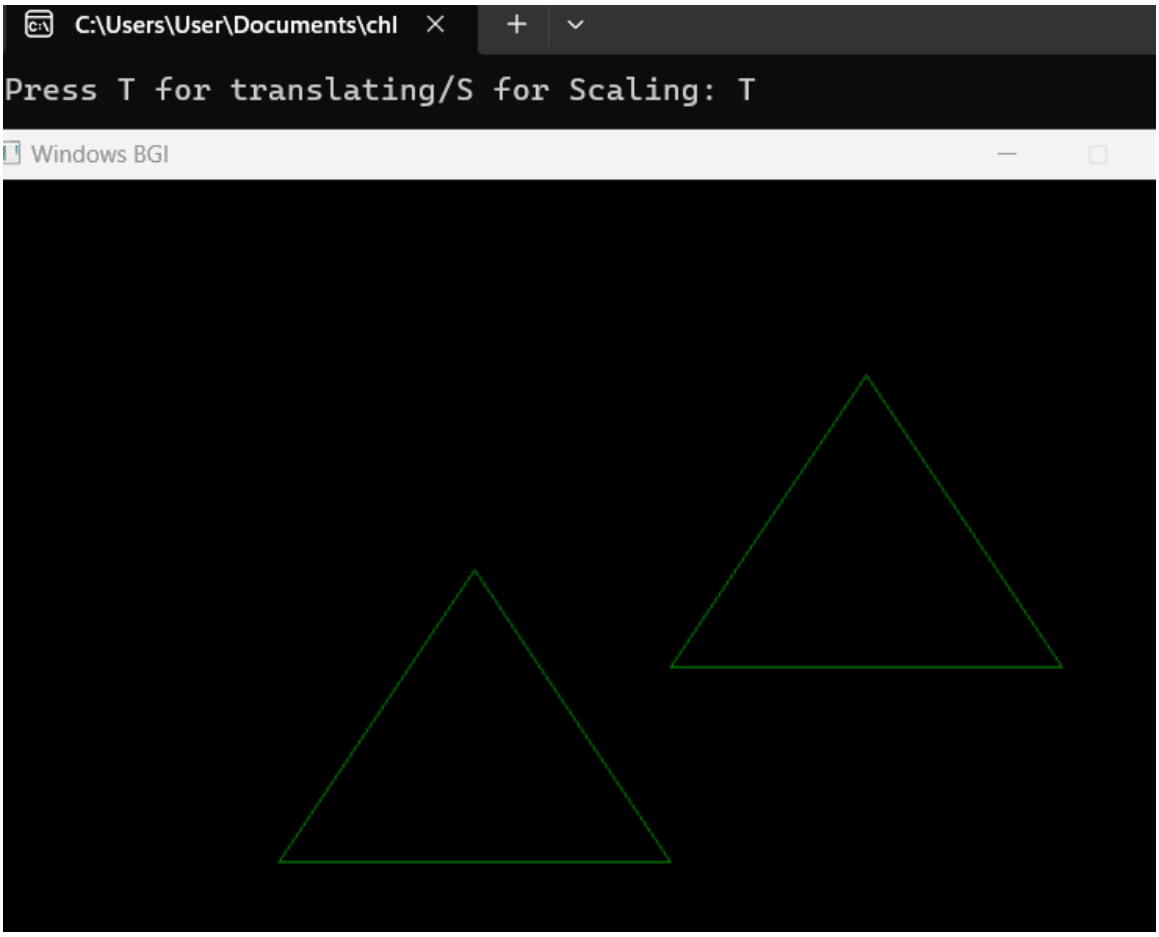
3. Scaling And Translation Algorithm C++ Code

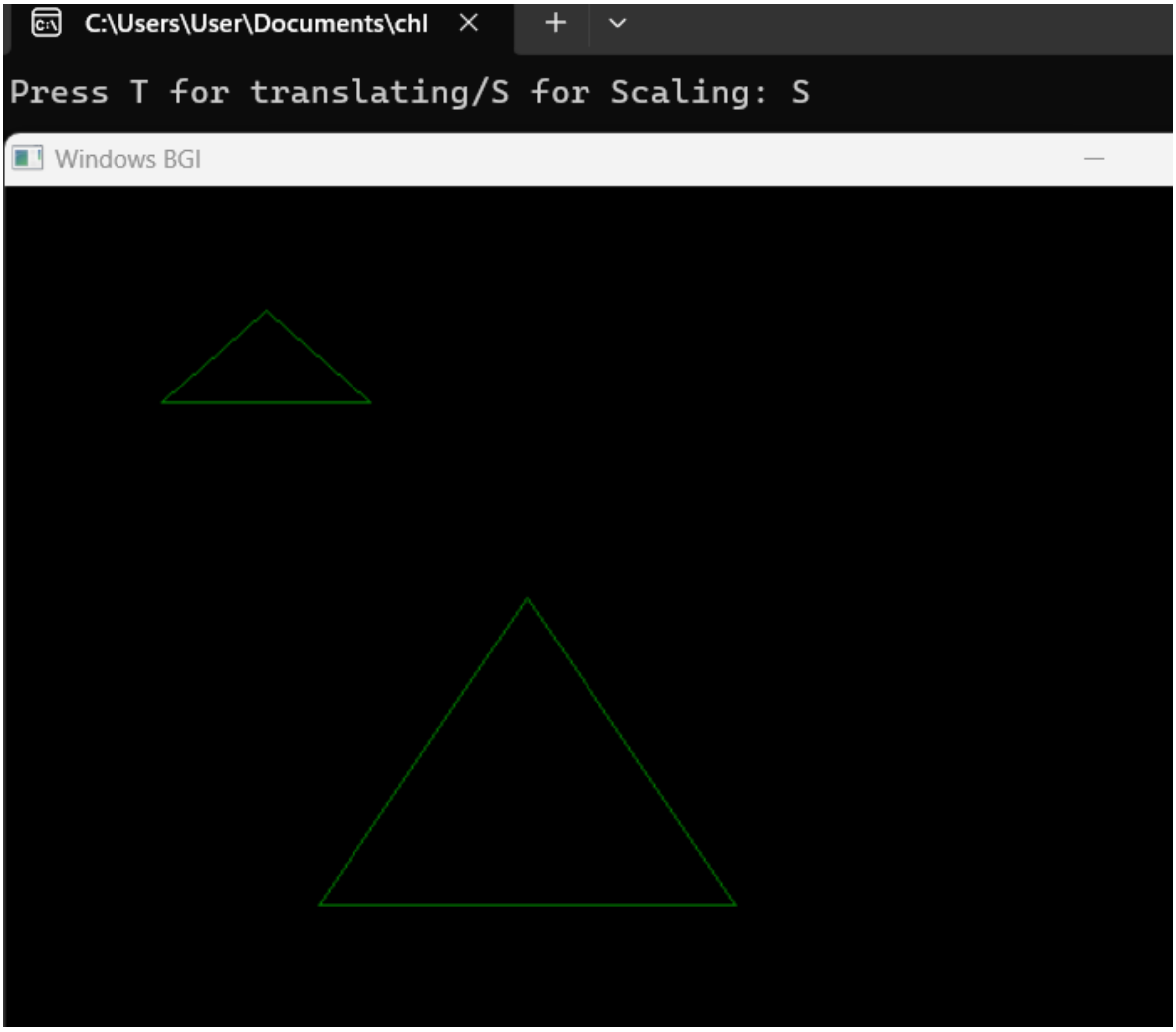
```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
```

```
void translation(int x1,int y1,int x2,int y2,int x3,int y3){
        int tx=200;
        int ty=-100;
        line(x1,y1,x2,y2);
    line(x2,y2,x3,y3);
    line(x3,y3,x1,y1);
        x1+=tx;
        y1+=ty;
    x2+=tx;
    y2+=ty;
    x3+=tx;
    y3+=ty;
    line(x1,y1,x2,y2);
    line(x2,y2,x3,y3);
    line(x3,y3,x1,y1);
 getch();
}
 void scaling(float x1, float y1,float x2,float y2,float x3,float y3){
    float sx,sy;
    sx = 0.5;
    sy = 0.3;
    x1 *= sx;
    x2 *= sx;
    y1 *= sy;
    y2 *= sy;
    x3 *= sx;
    y3 *= sy;
    line(x1,y1,x2,y2);
    line(x2,y2,x3,y3);
    line(x3,y3,x1,y1);
 getch();
}
int main(){
    int gd,gm;
   detectgraph(&gd,&gm);
    initgraph(&gd,&gm,"");
    int x1=250, y1=200, x2=150, y2=350, x3=350, y3=350;
    setcolor(GREEN);
    line(x1,y1,x2,y2);
    line(x2,y2,x3,y3);
```

```
    line(x3,y3,x1,y1);
    cout<<"Press T for translating/S for Scaling: ";
    char ch;
    cin>>ch;
    if(ch=='T')
          translation(x1,y1,x2,y2,x3,y3);
    else if(ch=='S')
        scaling(x1,y1,x2,y2,x3,y3);
}
```

Input & Output:




C:\Users\User\Documents\chl
Press T for translating/S for Scaling: S
Windows BGI

Rotation Algorithm C++ Code

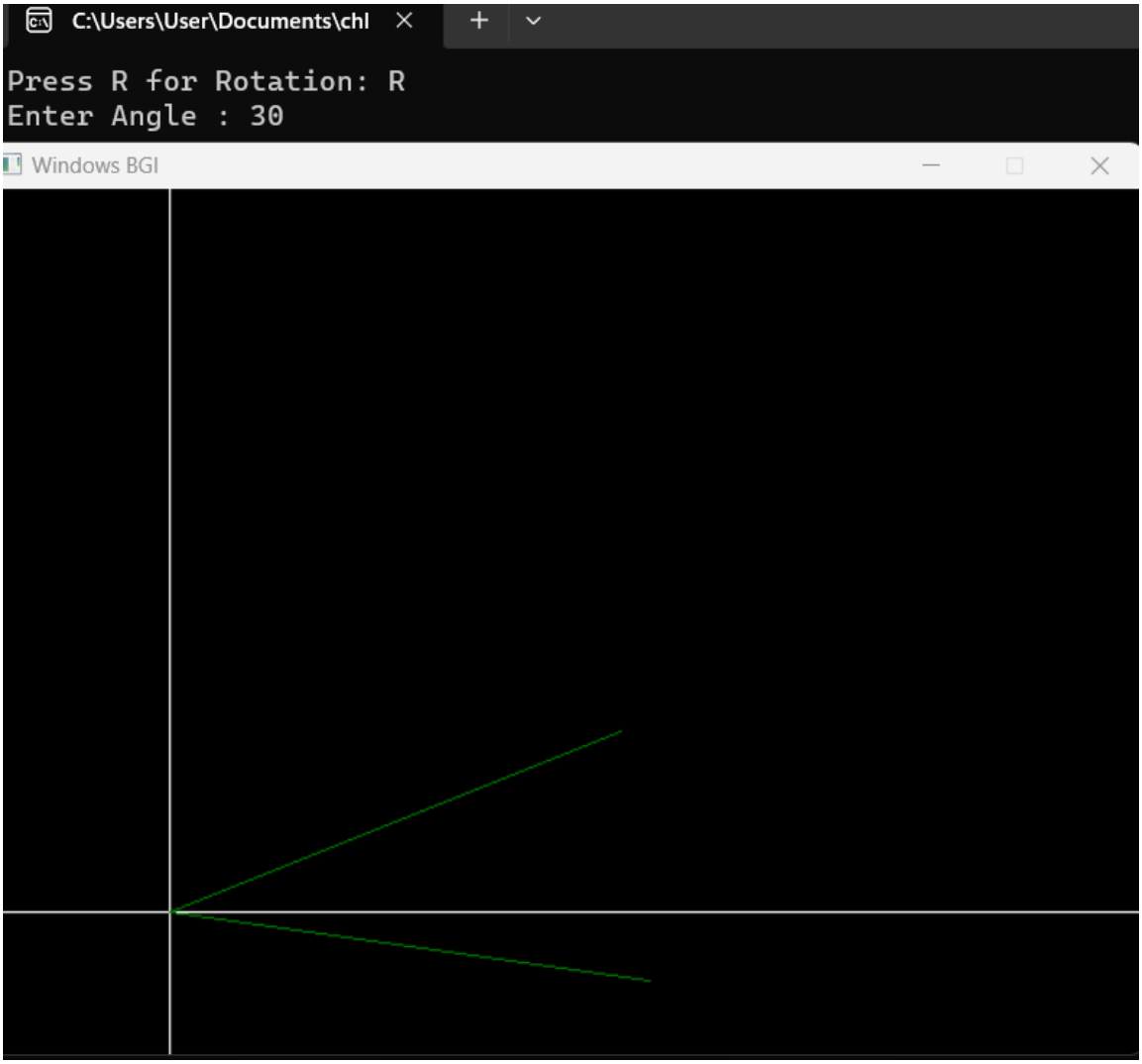
```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
void Rotation(float x1, float y1,float x2,float y2)
{
    cout <<"Enter Angle : ";
    getchar();
    float angle,xr,yr;
    cin >> angle;
    getchar();
    angle = M_PI*angle/180;
    xr = x1+(x2-x1)*cos(angle) - (y2-y1)*sin(angle);
    yr = y1+(x2-x1)*sin(angle) + (y2-y1)*cos(angle);
    setcolor(GREEN);
    line(x1,y1,xr,yr);
}

int main()
{
    int gd,gm;
    detectgraph(&gd,&gm);
    initgraph(&gd,&gm,"");
    line(100, 0, 100, 500);
    line(0, 400, 700, 400);

    int x1=100, y1=400, x2=350, y2=300;
    setcolor(GREEN);
    line(x1,y1,x2,y2);
    cout<<"Press R for Rotation: ";
    char ch;
    cin>>ch;
    if(ch=='R')
        Rotation(x1,y1,x2,y2);
    getchar();

}
```

Input & Output:



4. Bezier Curve Algorithm C++ Code

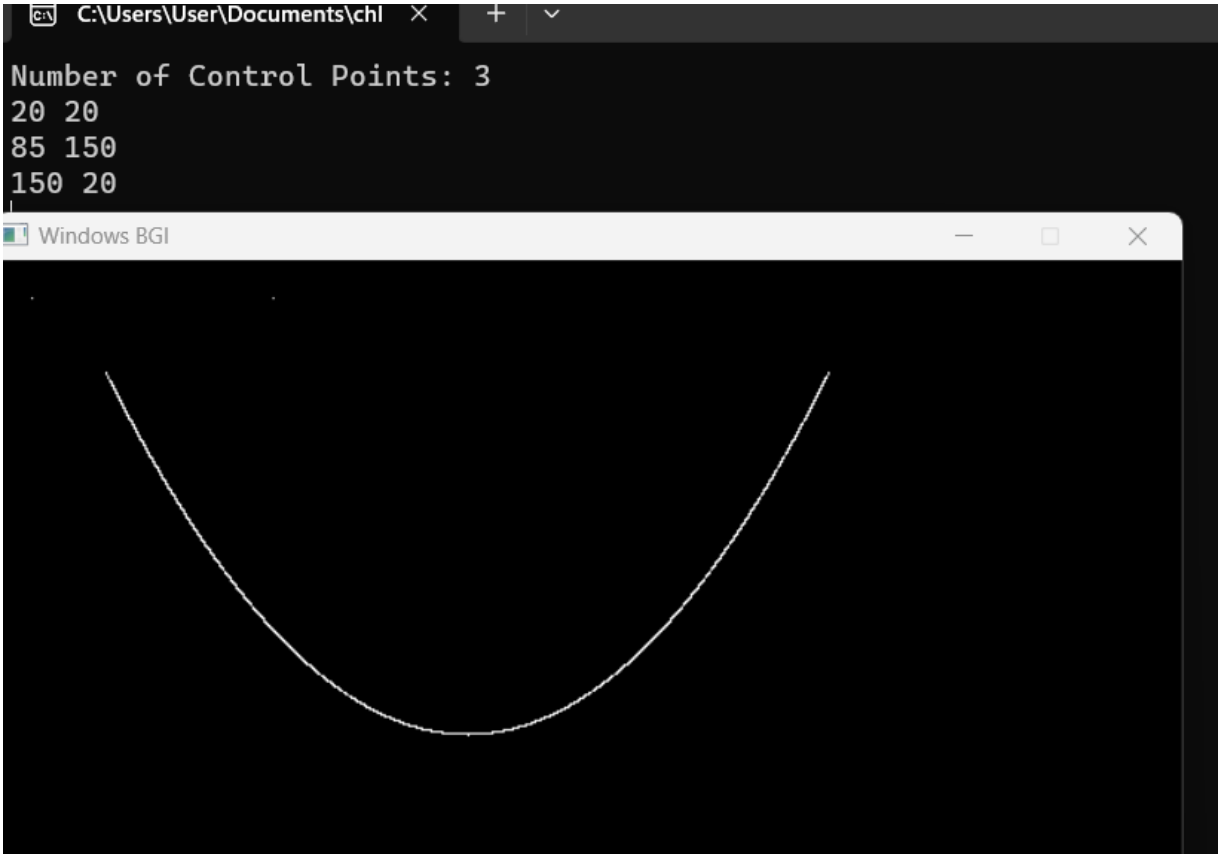
```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
int fact(int n)
{
    if(n>1)
        return n*fact(n-1);
    else if(n>=0)
        return 1;
}
```

```
void bezier(int px[], int py[], int n)
{
    float u,x,y,b;
    int i;
    putpixel(px[0],px[0],WHITE);

    int nfact = fact(n-1);
    for(u=0.0001; u<1; u+=0.0001)
    {
        x = 0;
        y = 0;
        for(i=0; i<n; i++)
        {
            b = fact(n)*pow(u,i)*pow(1-u,n-1-i)/(fact(n-1-i)*fact(i));
            x+=b*px[i];
            y+=b*py[i];
        }
        putpixel(x,y,WHITE);
    }
    putpixel(px[n-1],py[n-1],WHITE);
}
int main()
{
    int gd = DETECT,gm;
    initgraph(&gd,&gm,"");
    int n,i;
    cout<<"Number of Control Points: ";
    cin >> n;
    int px[n],py[n];

    for(i=0; i<n; i++)
    {
        cin >> px[i] >> py[i];
    }
    getchar();
    bezier(px,py,n);
    getchar();
}
```

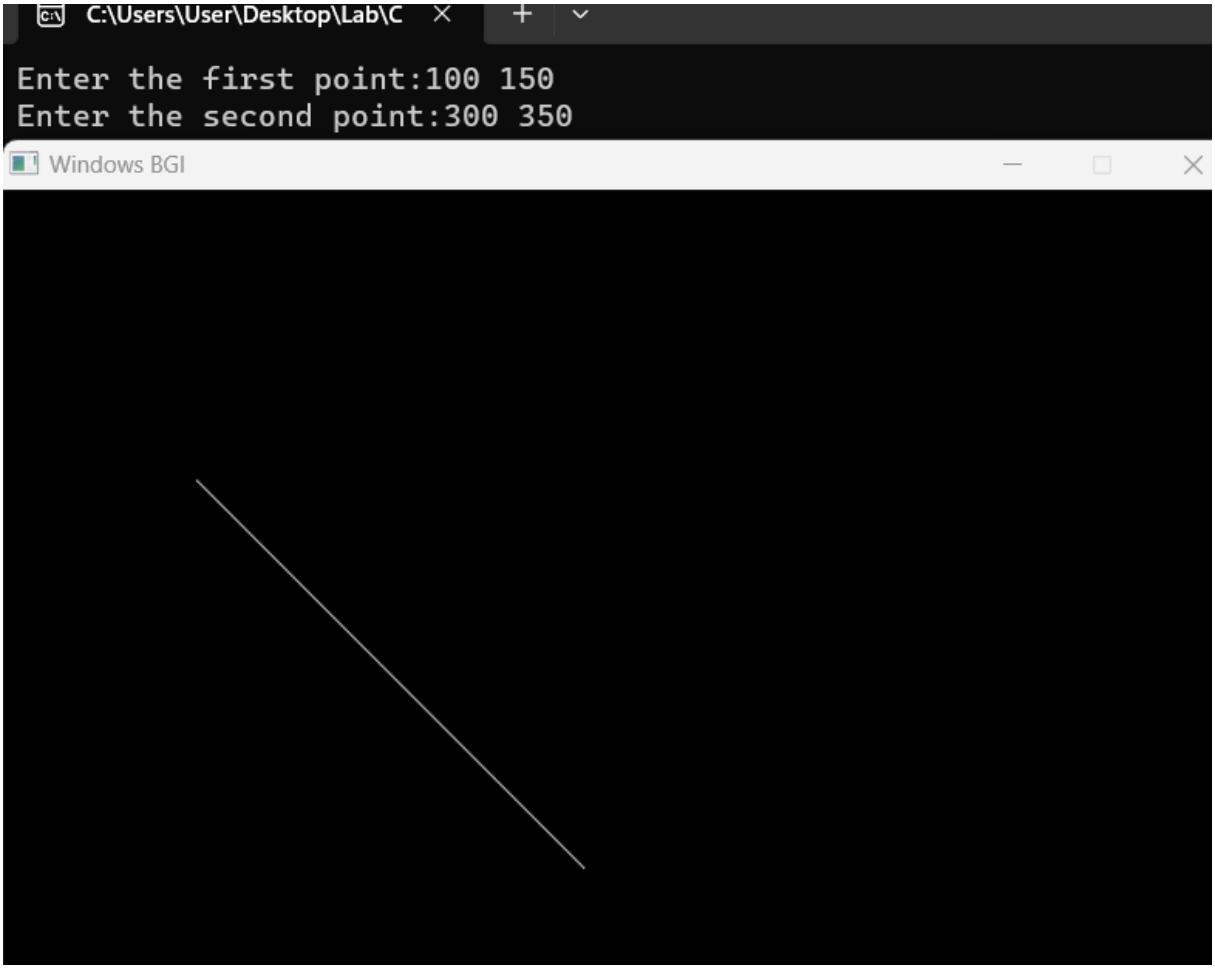
Input & Output:



5. Bezier Curve Algorithm C++ Code

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
int main()
{
    int gd,gm;
    detectgraph(&gd,&gm);
    initgraph(&gd,&gm,"");
    int x1,x2,y1,y2,dx,dy,x,y,p,i;
    cout <<"Enter the first point:";
    cin >> x1 >> y1;
    cout <<"Enter the second point:";
    cin >> x2 >> y2;
    getchar();
    i = 0;
    dx = x2-x1;
    dy = y2-y1;
    p = 2*dy-dx;
    x = x1;
    y = y1;
    while(i<=dx)
    {
        putpixel(x,y,WHITE);
        if(p<0)
        {
            x++;
            p+=(2*dy);
        }
        else
        {
            x++;
            y++;
            p = p+2*dy-2*dx;
        }
        i++;
    }
    getchar();
}
```

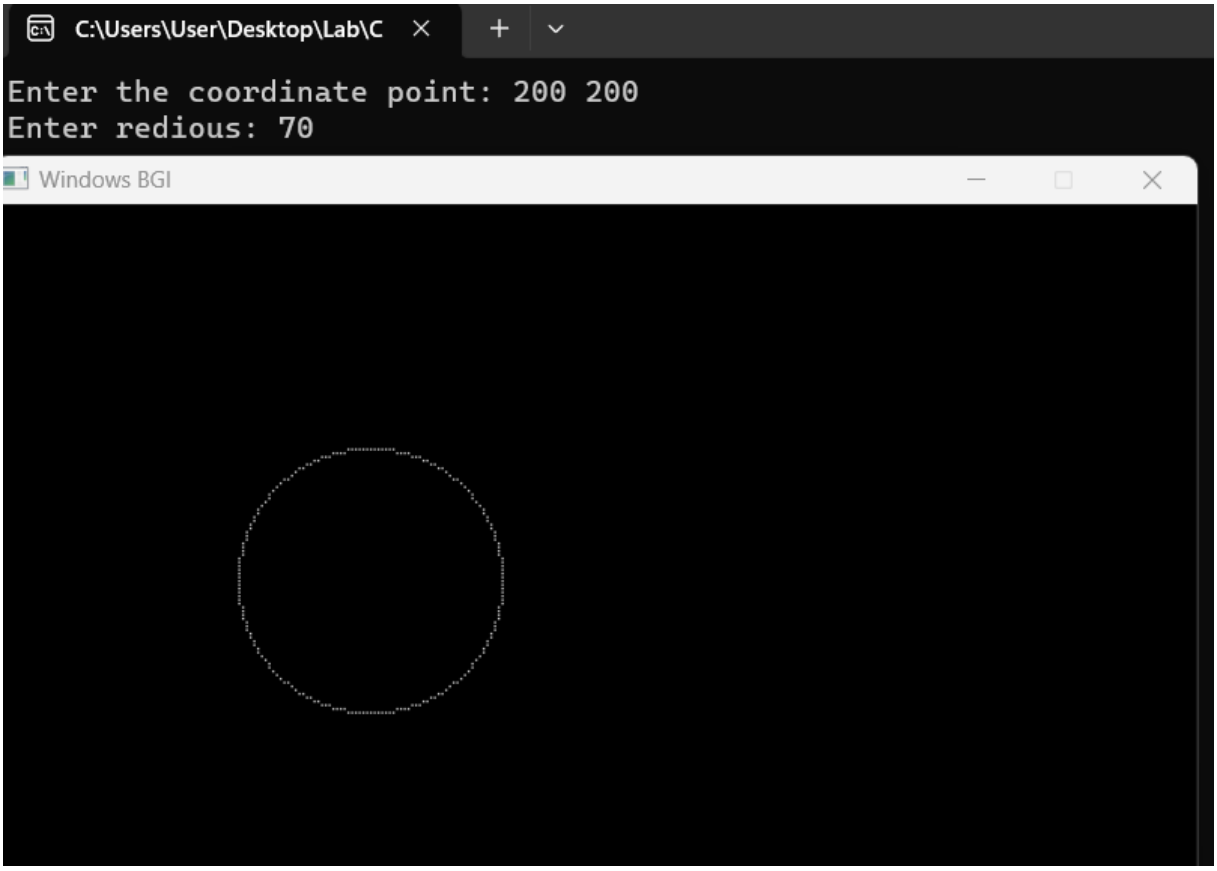
Input & Output:



6. Midpoint Circle Algorithm C++ Code

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
int main()
{
    int gd,gm;
    detectgraph(&gd,&gm);
    initgraph(&gd,&gm,"");
    double x,y,r,p,xp,yp;
    cout <<"Enter the coordinate point: ";
    cin >> xp >> yp;
    cout <<"Enter redious: ";
    cin >> r;
    getchar();
    x=0;
    y=r;
    p=1.25-r;
    while(x<=y)
    {
        putpixel(x+xp,y+yp,WHITE);
        putpixel(-x+xp,y+yp,WHITE);
        putpixel(x+xp,-y+yp,WHITE);
        putpixel(-x+xp,-y+yp,WHITE);
        putpixel(y+yp,x+xp,WHITE);
        putpixel(-y+yp,x+xp,WHITE);
        putpixel(y+yp,-x+xp,WHITE);
        putpixel(-y+yp,-x+xp,WHITE);
        x+=2;
        if(p<0)
            p = p + 2*x +1;
        else
        {
            y-=2;
            p = p - 2*y + 2*x +1;
        }
    }
    getchar();
    closegraph();
}
```

Input & Output:



7. Cohen Sutherland line clipping Algorithm C++ Code

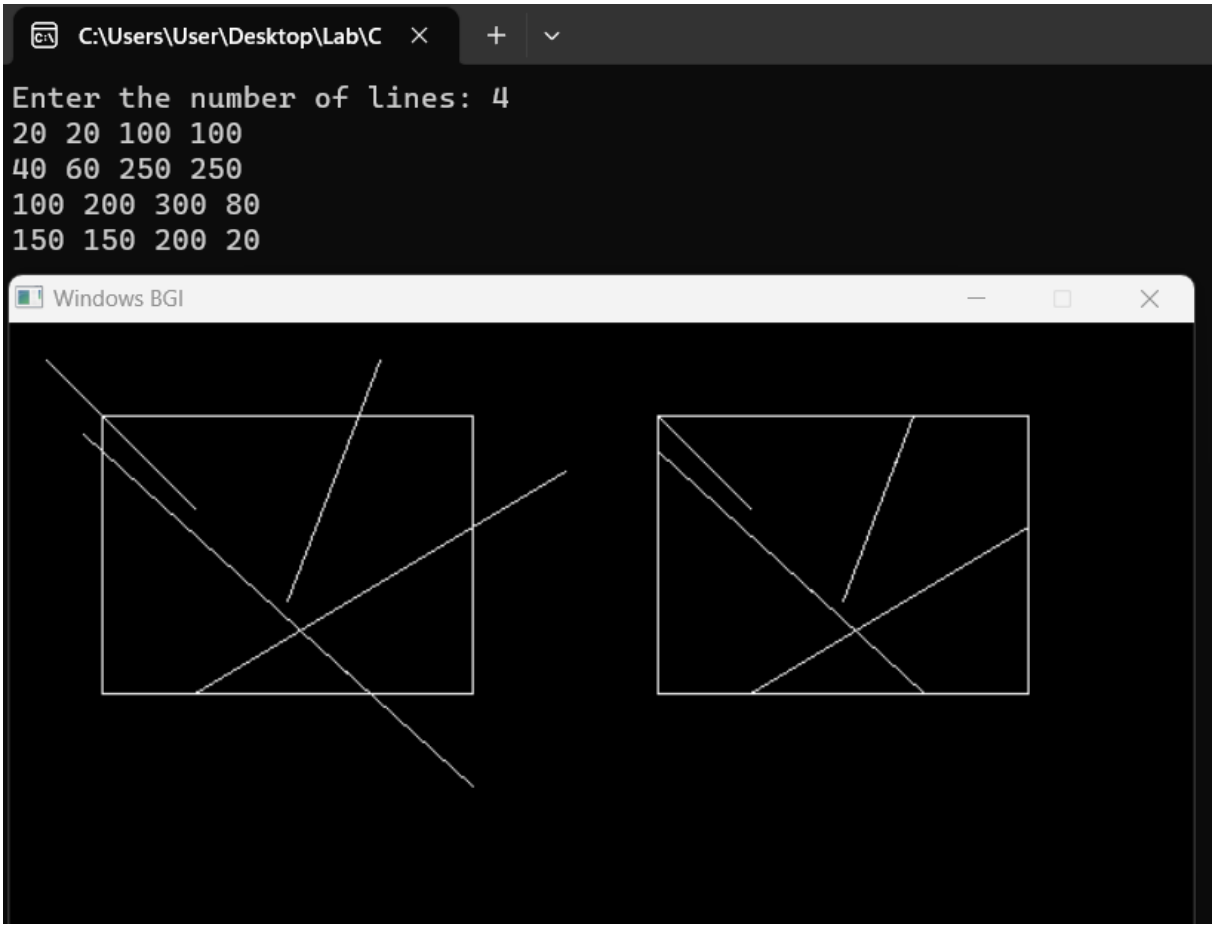
```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
int main()
{
    //freopen("7_Cohen_Sutherland_Line_Clipping_Algorithm.txt","r",stdin);
    int gd,gm;
    detectgraph(&gd,&gm);
    initgraph(&gd,&gm,"");
    double xmin,xmax,ymin,ymax,xdif;
    xmin = 50;
    ymin = 50;
    xmax = 250;
    ymax = 200;
    xdif = 300;
    //cout <<"Enter the coordinates of clipping window: ";
```

```
//cin >> x1 >> y1 >> x2 >> y2;
//getchar();
rectangle(xmin,ymin,xmax,ymax);
rectangle(xmin+xdif,ymin,xmax+xdif,ymax);
int lines;
cout <<"Enter the number of lines: ";
cin >> lines;
while(lines--)
{
   double x1,x2,y1,y2,x,y,m;
   cin >> x1 >> y1 >> x2 >> y2;
   //x1 = 100;y1 =300;x2 = 300;y2 = 80;
   m = (y2-y1)/(x2-x1);
   line(x1,y1,x2,y2);
   if(x1<xmin)
   {
      y1 = y1+(xmin-x1)*m;
      x1 = xmin;
   }
   if(x1>xmax)
   {
      y1 = y1+(xmax-x1)*m;
      x1 = xmax;
   }
   if(y1<ymin)
   {
      x1 = x1+(ymin-y1)/m;
      y1 = ymin;
   }
   if(y1>ymax)
   {
      x1 = x1+(ymax-y1)/m;
      y1 = ymax;
   }

   if(x2<xmin)
   {
      y2 = y2+(xmin-x2)*m;
      x2 = xmin;
   }
```

```
        if(x2>xmax)
        {
            y2 = y2+(xmax-x2)*m;
            x2 = xmax;
        }
        if(y2<ymin)
        {
            x2 = x2+(ymin-y2)/m;
            y2 = ymin;
        }
        if(y2>ymax)
        {
            x2 = x2+(ymax-y2)/m;
            y2 = ymax;
        }
        line(x1+xdif,y1,x2+xdif,y2);
        getchar();
    }
    getchar();
}
```

Input & Output:



8. Sutherland line clipping Algorithm C++ Code

```
#include<graphics.h>
#include<bits/stdc++.h>
using namespace std;
void mainpolygon(double x1,double y1,double x2,double y2)
{
    setcolor(YELLOW);
    line(x1,y1,x2,y2);
    setfillstyle(SOLID_FILL,YELLOW);
}
int main()
{
    int gd, gm;
    detectgraph(&gd,&gm);
    initgraph(&gd, &gm,(char*)"");
    double xmin,xmax,ymin,ymax,xdif;
    xmin = 50;
```

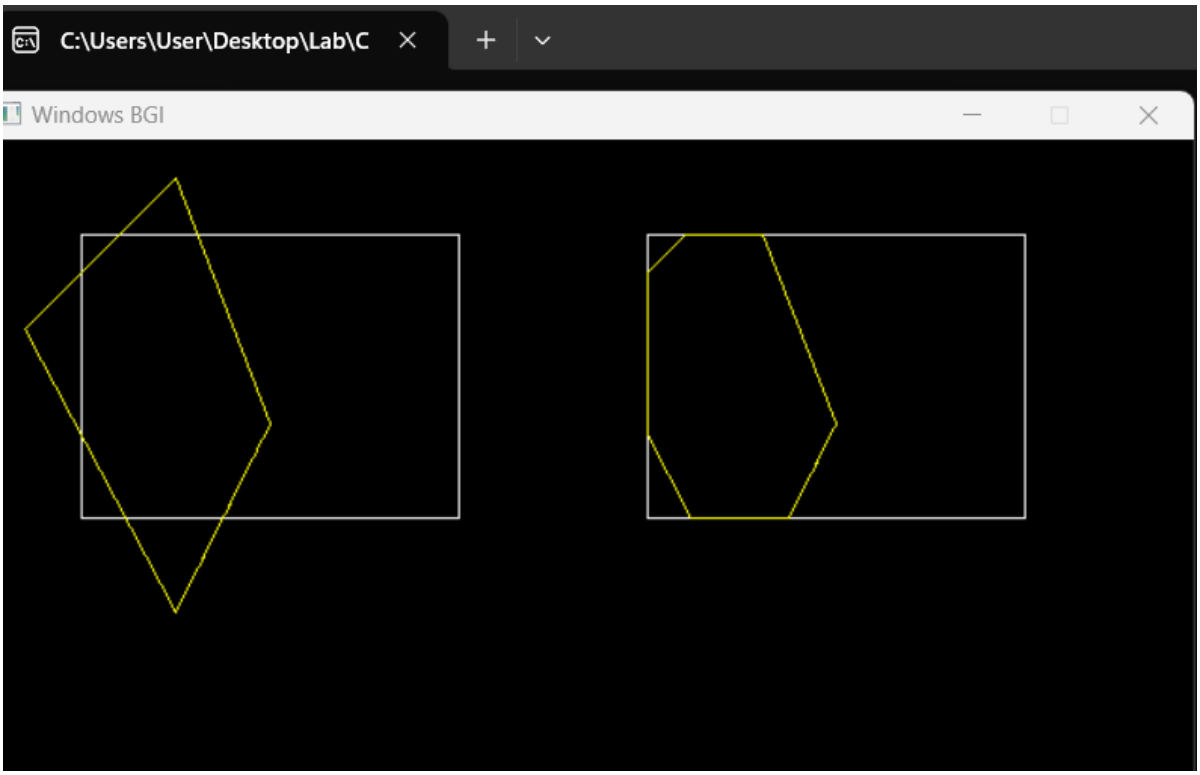
```
ymin = 50;
xmax = 250;
ymax = 200;
xdif = 300;
rectangle(xmin,ymin,xmax,ymax);
rectangle(xmin+xdif,ymin,xmax+xdif,ymax);
//double points [][2] = {{20,100},{100,250},{150,150},{100,20}};
double points[8] = {20,100,100,250,150,150,100,20};
vector<double> newpoint;
for(int i=0; i<8; i+=2)
{
   double x1,y1,x2,y2,m;
   x1 = points[i];
   y1 = points[i+1];
   x2 = points[(i+2)%8];
   y2 = points[(i+3)%8];
   m = (y2-y1)/(x2-x1);
   mainpolygon(x1,y1,x2,y2);
   if(x1<xmin)
   {
      y1 = y1+(xmin-x1)*m;
      x1 = xmin;
      newpoint.push_back(x1);
      newpoint.push_back(y1);
   }
   if(x1>xmax)
   {
      y1 = y1+(xmax-x1)*m;
      x1 = xmax;
      newpoint.push_back(x1);
      newpoint.push_back(y1);
   }
   if(y1<ymin)
   {
      x1 = x1+(ymin-y1)/m;
      y1 = ymin;
      newpoint.push_back(x1);
      newpoint.push_back(y1);
   }
   if(y1>ymax)
```

```
    {
        x1 = x1+(ymax-y1)/m;
        y1 = ymax;
        newpoint.push_back(x1);
        newpoint.push_back(y1);
    }

    if(x2<xmin)
    {
        y2 = y2+(xmin-x2)*m;
        x2 = xmin;
        newpoint.push_back(x2);
        newpoint.push_back(y2);
    }
    if(x2>xmax)
    {
        y2 = y2+(xmax-x2)*m;
        x2 = xmax;
        newpoint.push_back(x2);
        newpoint.push_back(y2);
    }
    if(y2<ymin)
    {
        x2 = x2+(ymin-y2)/m;
        y2 = ymin;
        newpoint.push_back(x2);
        newpoint.push_back(y2);
    }
    if(y2>ymax)
    {
        x2 = x2+(ymax-y2)/m;
        y2 = ymax;
        newpoint.push_back(x2);
        newpoint.push_back(y2);
    }
    setcolor(YELLOW);
    setfillstyle(SOLID_FILL,YELLOW);
    line(x1+xdif,y1,x2+xdif,y2);
}
for(int i=0; i<newpoint.size(); i+=2)
```

```
    {
        double x1,y1,x2,y2;
        x1 = newpoint[i]+xdif;
        y1 = newpoint[i+1];
        x2 = newpoint[(i+2)%newpoint.size()]+xdif;
        y2 = newpoint[(i+3)%newpoint.size()];
        setcolor(YELLOW);
        setfillstyle(SOLID_FILL,YELLOW);
        if(x1==x2||y1==y2)
            line(x1,y1,x2,y2);
    }
    cout << endl;
    getch();
    closegraph();
    return 0;
}
```

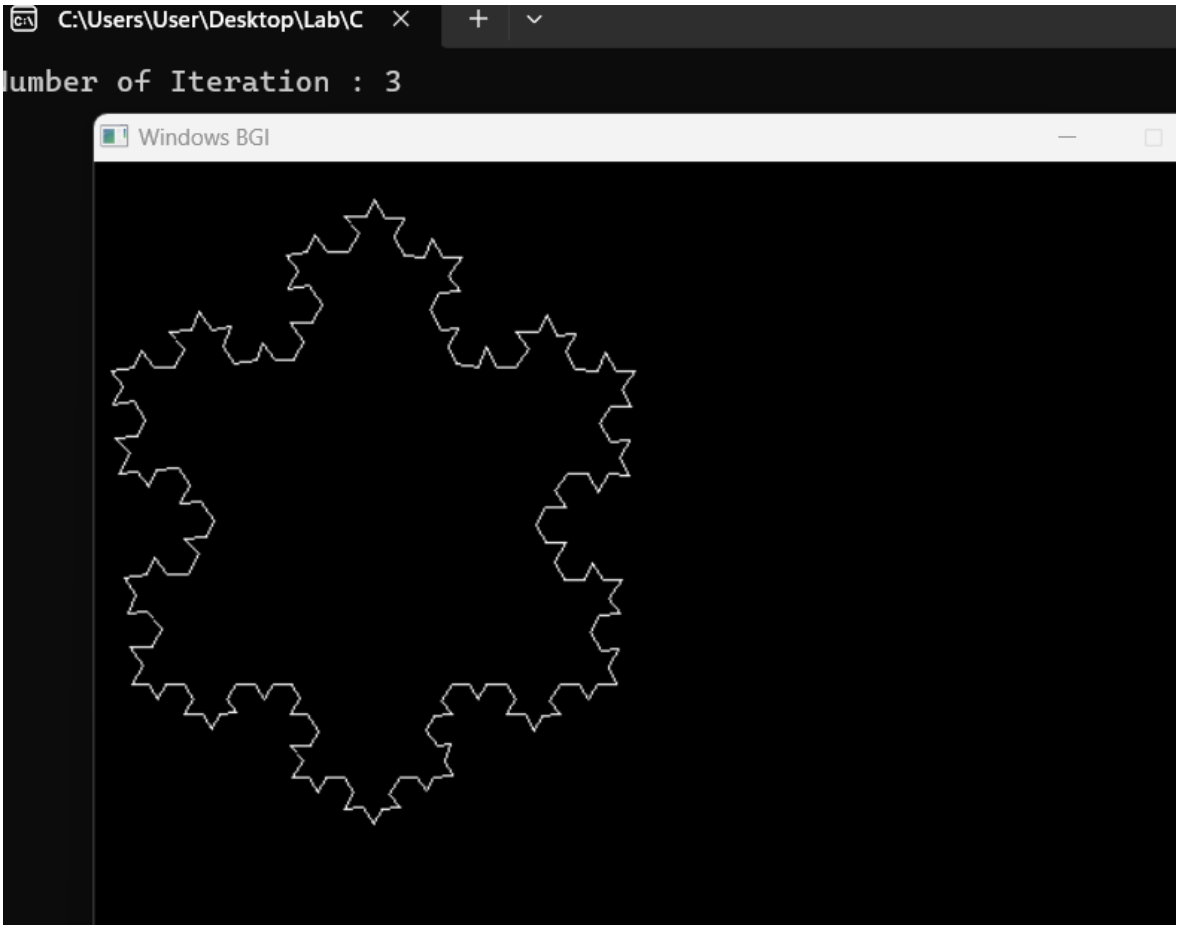
Input & Output:



9. C++ Program to Generate Fractal Patterns by Using Koch Curves

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
void koch(int it,int x1,int y1,int x2,int y2){
    int x3,y3,x4,y4,x5,y5;
    int dx,dy;
    if(it==0){
        line(x1,y1,x2,y2);
    }
    else{
        dx = (x2-x1)/3;
        dy = (y2-y1)/3;
        x3 = x1+dx;
        y3 = y1+dy;
        x4 = 0.5*(x1+x2)+sqrt(3)*(y1-y2)/6;
        y4 = 0.5*(y1+y2)+sqrt(3)*(x2-x1)/6;
        x5 = 2*dx+x1;
        y5 = 2*dy+y1;
        koch(it-1,x1,y1,x3,y3);
        koch(it-1,x3,y3,x4,y4);
        koch(it-1,x4,y4,x5,y5);
        koch(it-1,x5,y5,x2,y2);
    }
}
int main(){
    int it;
    cout <<"Number of Iteration : ";
    cin >> it;
    int gd,gm;
    detectgraph(&gd,&gm);
    initgraph(&gd,&gm,NULL);
    koch(it,150,20,20,280);
    koch(it,280,280,150,20);
    koch(it,20,280,280,280);
    getch();
}
```

Input & Output:


C:\Users\User\Desktop\Lab\C
umber of Iteration : 3
Windows BGI