CG Experiment 11

Yash Sarang D6AD/47

Aim: Implement fractal generation method - Koch curve.

Theory:

Fractals - They are very complex pictures generated by a computer from a single formula. They are created using iterations. That means 1 formula is repeatedly used with slightly different values over & over again, taking into account the result from previous iteration.

Koch Curve - The Koch snowflake (also known as Koch star, Koch curve is a mathematical curve and one of the earliest practical curves to have been constructible from elementary geometry.

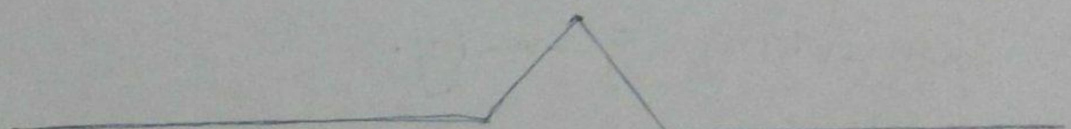
A Koch curve is a fractal generated by a replacement rule. This rule is, at each step, to replace the middle 131/3 of each line segment with two sides of a right angle triangle having sides of length equal to the replaced segment.

This quantity increases without bound. Hence, the Koch curve has infinite length. However, the curve still bounds a finite area.

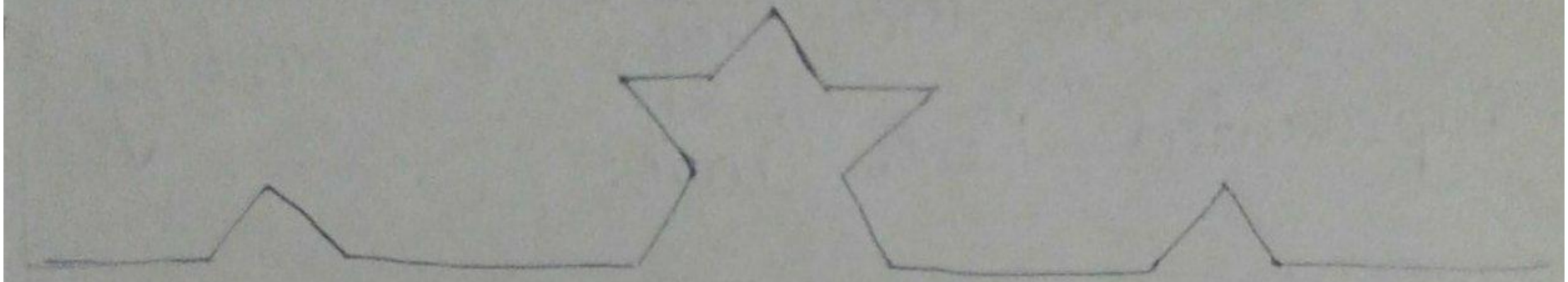
Construction -

The Koch curve can be constructed by starting with an equilateral triangle. Divide the line segment into 3 segments of equal length

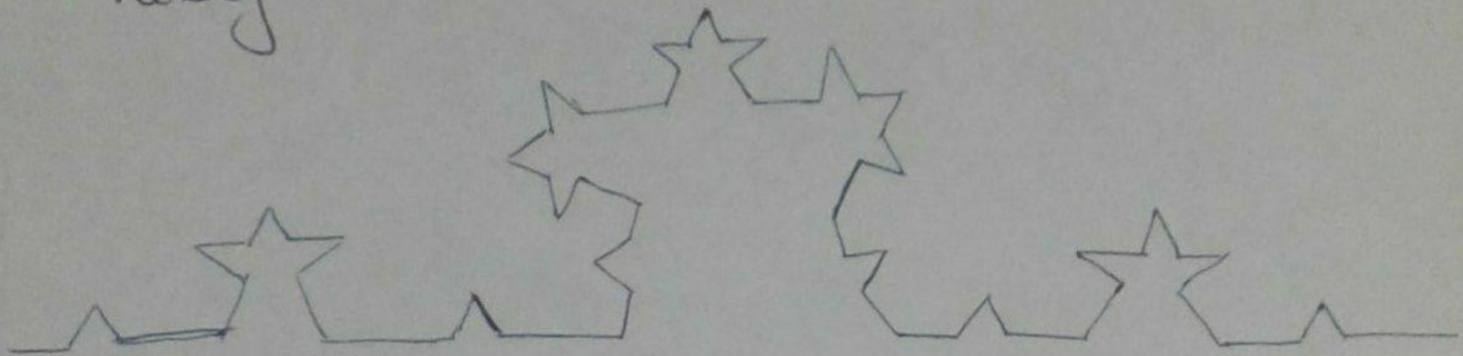
- Draw an equilateral triangle that has the middle segment from step 1 as its base and points outward.



- Remove the line segment that is the base of the triangle from step 2.



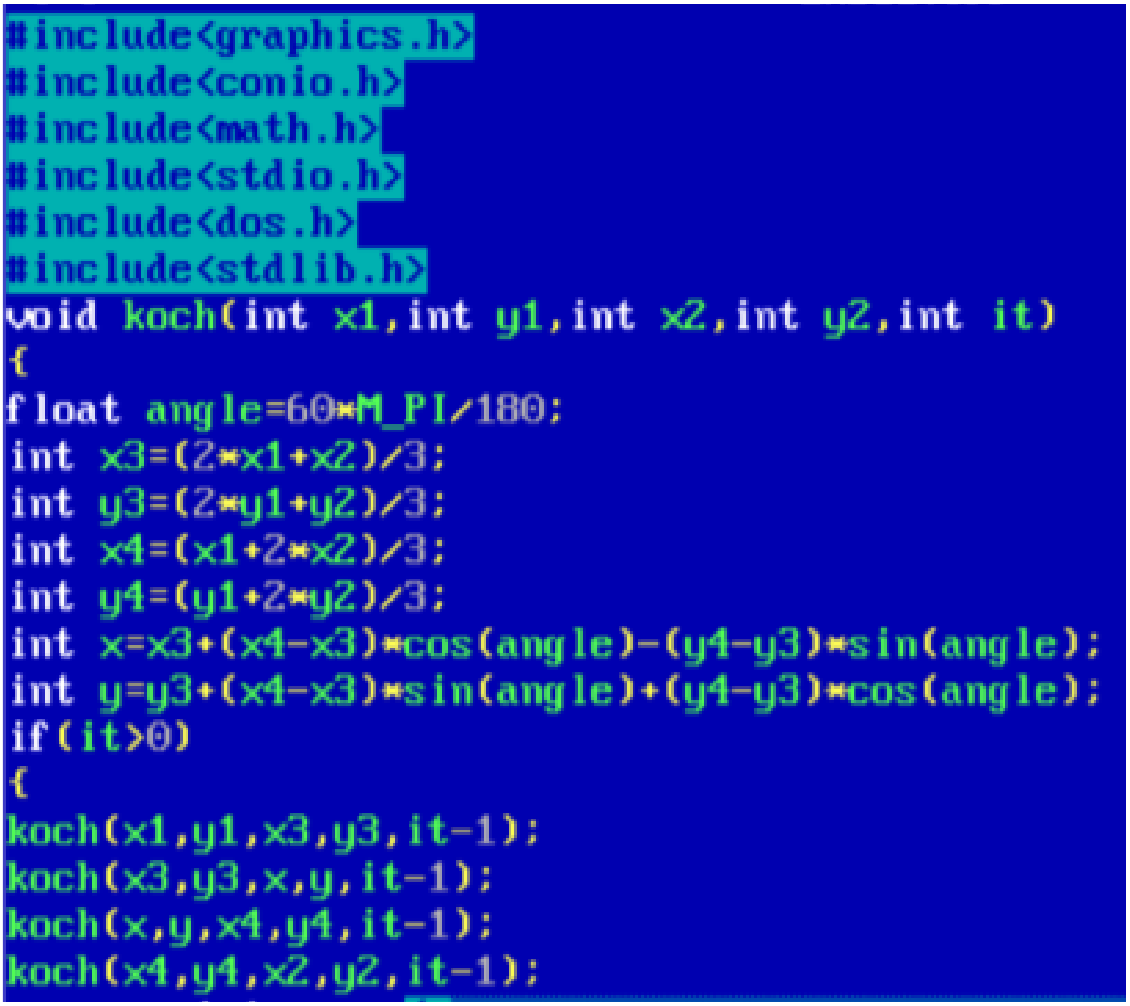
- After one iteration of this process,. the resulting shape is the outline of a hexagram.



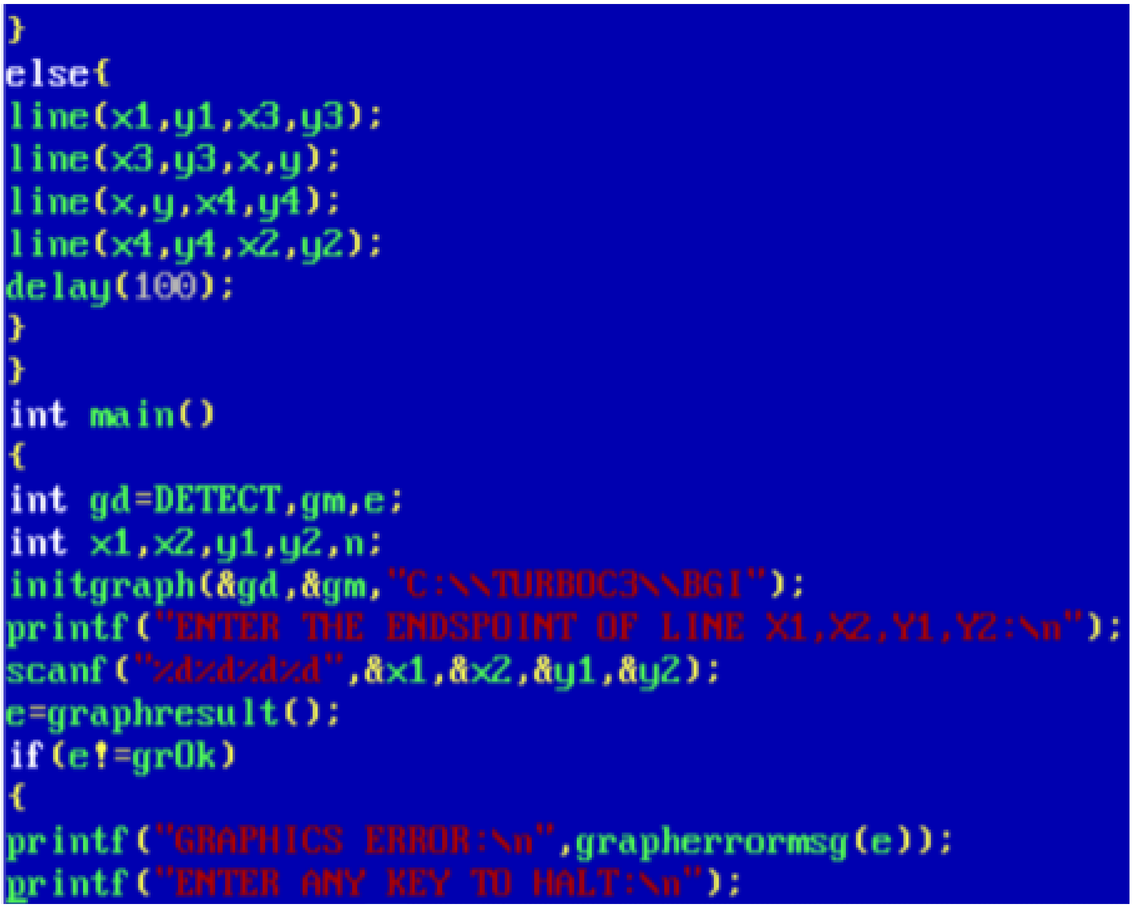
The Koch curve is a limit approach as the above steps are followed over & over ~~agia~~. again.

In short, three Koch curves make a Koch snowflake.

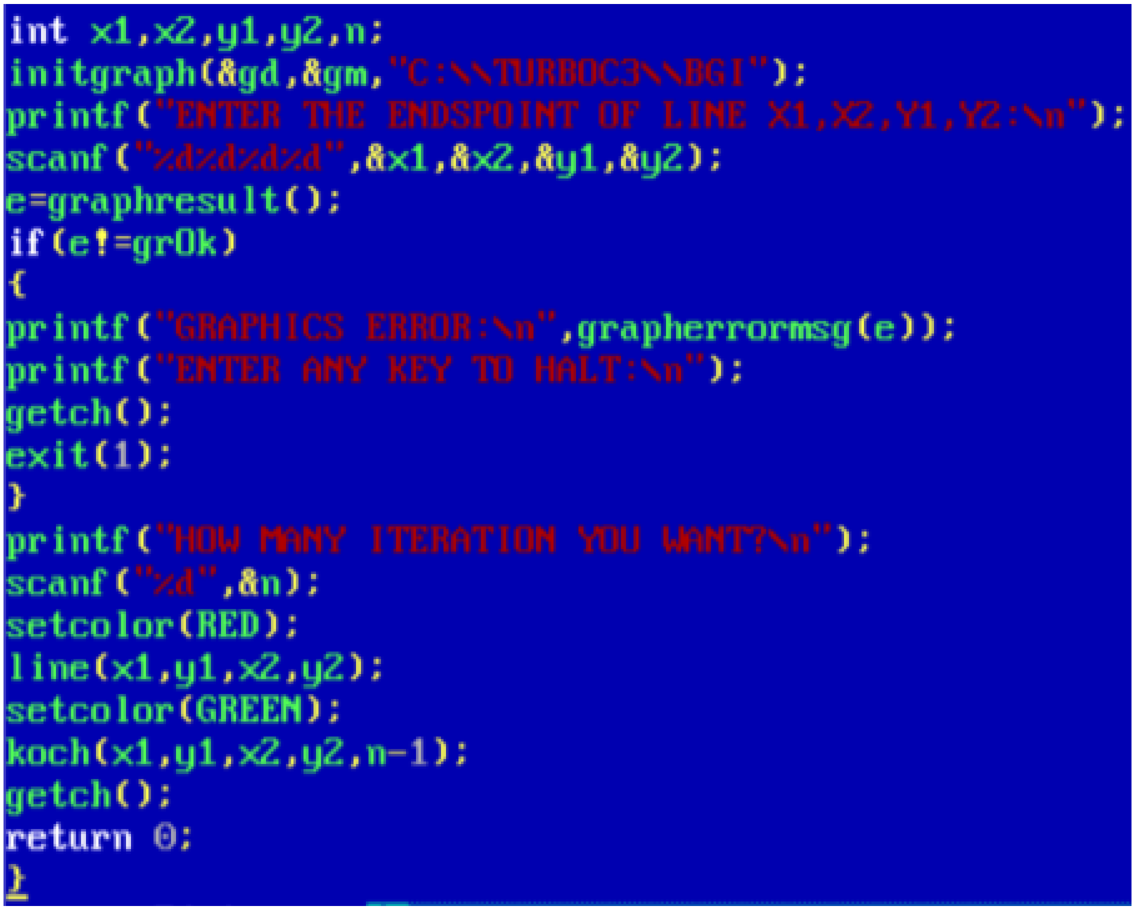
Code:-



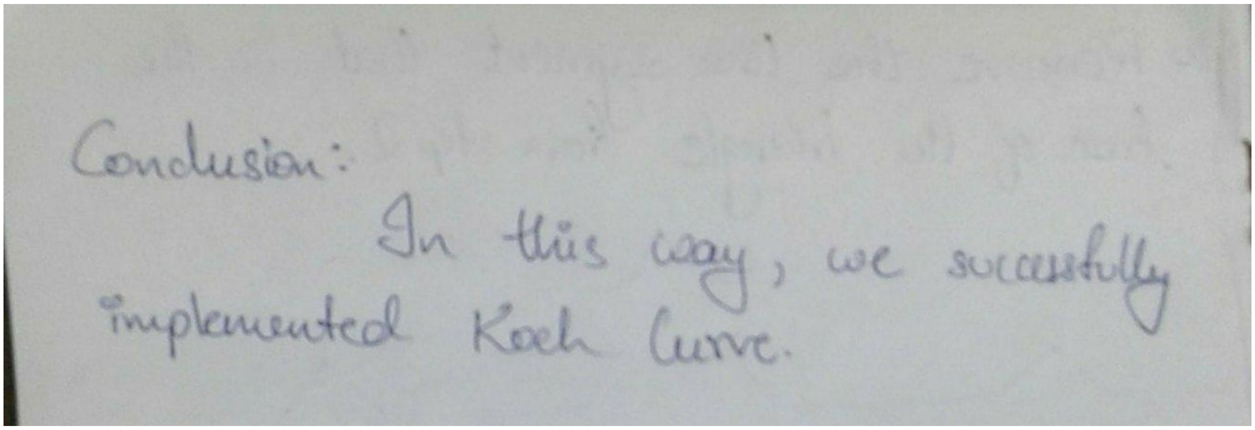
```
#include<graphics.h>
#include<conio.h>
#include<math.h>
#include<stdio.h>
#include<dos.h>
#include<stdlib.h>
void koch(int x1,int y1,int x2,int y2,int it)
{
float angle=60*M_PI/180;
int x3=(2*x1+x2)/3;
int y3=(2*y1+y2)/3;
int x4=(x1+2*x2)/3;
int y4=(y1+2*y2)/3;
int x=x3+(x4-x3)*cos(angle)-(y4-y3)*sin(angle);
int y=y3+(x4-x3)*sin(angle)+(y4-y3)*cos(angle);
if(it>0)
{
koch(x1,y1,x3,y3,it-1);
koch(x3,y3,x,y,it-1);
koch(x,y,x4,y4,it-1);
koch(x4,y4,x2,y2,it-1);
```



```
}
else{
line(x1,y1,x3,y3);
line(x3,y3,x,y);
line(x,y,x4,y4);
line(x4,y4,x2,y2);
delay(100);
}
}
int main()
{
int gd=DETECT,gm,e;
int x1,x2,y1,y2,n;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
printf("ENTER THE ENDSPOINT OF LINE X1,X2,Y1,Y2:\n");
scanf("%d%d%d%d",&x1,&x2,&y1,&y2);
e=graphresult();
if(e!=grOk)
{
printf("GRAPHICS ERROR:\n",grapherrormsg(e));
printf("ENTER ANY KEY TO HALT:\n");
```



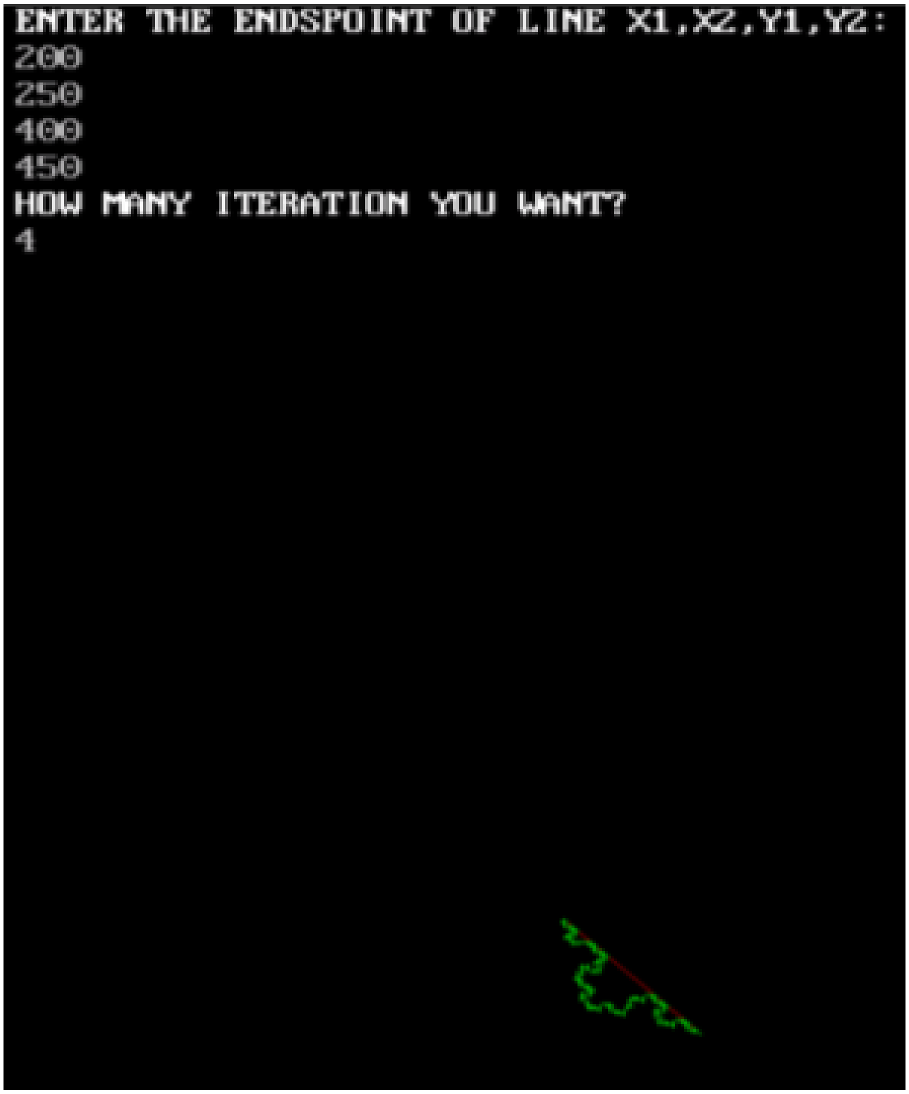
```
int x1,x2,y1,y2,n;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
printf("ENTER THE ENDSPOINT OF LINE X1,X2,Y1,Y2:\n");
scanf("%d%d%d%d",&x1,&x2,&y1,&y2);
e=graphresult();
if(e!=grOk)
{
printf("GRAPHICS ERROR:\n",grapherrormsg(e));
printf("ENTER ANY KEY TO HALT:\n");
getch();
exit(1);
}
printf("HOW MANY ITERATION YOU WANT?\n");
scanf("%d",&n);
setcolor(RED);
line(x1,y1,x2,y2);
setcolor(GREEN);
koch(x1,y1,x2,y2,n-1);
getch();
return 0;
}
```



Conclusion:

In this way, we successfully implemented Koch Curve.

Output :-



```
ENTER THE ENDSPOINT OF LINE X1,X2,Y1,Y2:
200
250
400
450
HOW MANY ITERATION YOU WANT?
4
```
