Polygon filling using scanline fill algorithm(concept of inheritance)

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

static int
LEFT=1,RIGHT=2,BOTTOM=4,TOP=8,xl,yl,xh,yh;

int getcode(int x,int y){
int code = 0;
//Perform Bitwise OR to get outcode
if(y > yh) code |=TOP;
if(y < yl) code |=BOTTOM;
if(x < xl) code |=LEFT;
if(x > xh) code |=RIGHT;
return code;
}

int main()
{
```

```
int gdriver = DETECT,gmode;

cout<<"Enter bottom left and top right
co-ordinates of window: ";
cin>>xl>>yl>>xh>>yh;

int x1,y1,x2,y2;
cout<<"Enter the endpoints of the line: ";
cin>>x1>>y1>>x2>>y2;
initgraph(&gdriver,&gmode,NULL);
setcolor(BLUE);
rectangle(xl,yl,xh,yh);
line(x1,y1,x2,y2);
getch();

int outcode1=getcode(x1,y1),
outcode2=getcode(x2,y2);
int accept = 0; //decides 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+ (yl-y1)/m;
y = yl;
}else if(temp & LEFT){ //Line clips left
edge
x = xl;
y = y1+ m*(xl-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);
}
}
}

cout<<"After clipping:";
if(accept)
cleardevice();
rectangle(xl,yl,xh,yh);
setcolor(WHITE);
line(x1,y1,x2,y2);
delay(5000);
closegraph();
return 0;
}
```

```
Output -----
Scan Fill Algorithm
Enter Number Of Vertices Of Polygon: 4
Enter co-ordinate no. 1 :
```

```
x1=100
y1=100
Enter co-ordinate no. 2 :
x2=200
y2=100
Enter co-ordinate no. 3 :
x3=200
y3=200
Enter co-ordinate no. 4 :
x4=100
y4=200
Enter The Color You Want :(In Range 0 To 15
)->12
```

Explanation

This C++ program implements the Cohen-Sutherland line clipping algorithm, which is a widely-used method for clipping lines against a rectangular clipping window. The algorithm determines which parts of a line segment lie inside a rectangular region and allows for drawing only those parts.

Key Features of the Code:

1.Boundary Codes:

•The program defines boundary codes using bitwise flags for the four edges of the rectangle (left, right, bottom, and top). Each point is assigned a code based on its position relative to the clipping window.

2.User Input:

•The user is prompted to input the coordinates of the clipping window (bottom-left and top-right corners) and the endpoints of the line segment to be clipped.

3.Line Clipping Logic:

•The algorithm calculates the outcodes for the endpoints of the line segment.

•If both endpoints are inside the clipping region (outcode = 0), the line is accepted as it is.

•If both endpoints are outside (AND operation of outcodes ≠ 0), the line is rejected.

•If one endpoint is inside and the other is outside, the program computes the intersection points with the clipping edges, updating the coordinates of the endpoints accordingly.

4.Graphics Initialization:

•The program uses the graphics.h library to visualize the rectangle and the clipped line segment.

5.Output:

•After processing, the program displays the clipped line segment within the defined rectangle.