experiment 7

COMPUTER GRAPICS AND MULTIMEDIA

# Aim

Write a C program to demonstrate Cohen Sutherland line clipping algorithm.

**Syeda Reeha Quasar**

**14114802719**

**3C7**

# **EXPERIMENT - 7**

**AIM:**

Write a C program to demonstrate Cohen Sutherland line clipping algorithm.

# **THEORY:**

The Cohen–Sutherland algorithm is a computer-graphics algorithm used for line clipping. The algorithm divides a two-dimensional space into 9 regions and then efficiently determines the lines and portions of lines that are visible in the central region of interest (the viewport).

In the Cohen Sutherland line clipping algorithm, first of all, it is detected whether line lies inside the screen or it is outside the screen. All lines come under any one of the following categories:

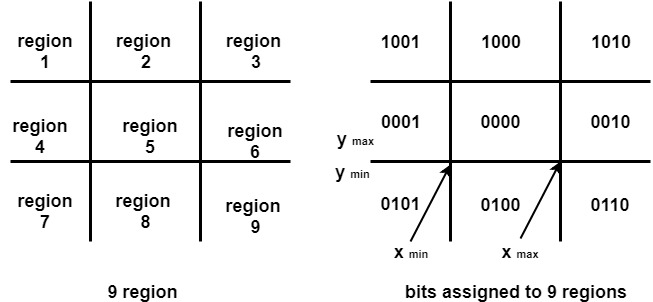
1. Visible
2. Not Visible
3. Clipping Case
4. **Visible:** If a line lies within the window, i.e., both endpoints of the line lies within the window. A line is visible and will be displayed as it is.
5. **Not Visible:** If a line lies outside the window it will be invisible and rejected. Such lines will not display. If any one of the following inequalities is satisfied, then the line is considered invisible.

Let A (x1,y2) and B (x2,y2) are endpoints of line.

xmin,xmax are coordinates of the window.

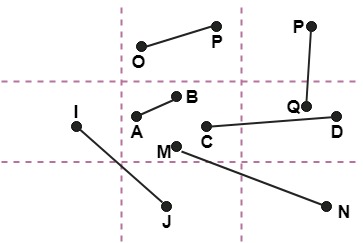
ymin,ymax are also coordinates of the window.  
          x1>xmax  
          x2>xmax  
          y1>ymax  
          y2>ymax  
          x1<xmin  
          x2<xmin  
          y1<ymin  
          y2<ymin

1. **Clipping Case:** If the line is neither visible case nor invisible case. It is considered to be clipped case. First of all, the category of a line is found based on nine regions given below. All nine regions are assigned codes. Each code is of 4 bits. If both endpoints of the line have end bits zero, then the line is considered to be visible.



The center area is having the code, 0000, i.e., region 5 is considered a rectangle window.

**Following figure show lines of various types**



Line AB is the visible case  
Line OP is an invisible case  
Line PQ is an invisible line  
Line IJ are clipping candidates  
Line MN are clipping candidate  
Line CD are clipping candidate

**Advantage of Cohen Sutherland Line Clipping:**

1. **It calculates end-points very quickly and rejects and accepts lines quickly.**
2. **It can clip pictures much large than screen size.**

**Algorithm of Cohen Sutherland Line Clipping:**

**Step1:** Calculate positions of both endpoints of the line.

**Step2:** Perform OR operation on both of these end-points.

**Step3:** If the OR operation gives 0000  
       Then  
                line is considered to be visible  
       else  
          Perform AND operation on both endpoints  
      If And ≠ 0000  
          then the line is invisible  
        else  
      And=0000  
    Line is considered the clipped case.

**Step4:** If a line is clipped case, find an intersection with boundaries of the window  
                m=(y2-y1 )(x2-x1)

**(a)** If bit 1 is "1" line intersects with left boundary of rectangle window  
                y3=y1+m(x-X1)  
                where X = Xwmin  
                where Xwminis the minimum value of X co-ordinate of window

**(b)** If bit 2 is "1" line intersect with right boundary  
                y3=y1+m(X-X1)  
                where X = Xwmax  
                where X more is maximum value of X co-ordinate of the window

**(c)** If bit 3 is "1" line intersects with bottom boundary  
                X3=X1+(y-y1)/m  
                      where y = ywmin  
                ywmin is the minimum value of Y co-ordinate of the window

**(d)** If bit 4 is "1" line intersects with the top boundary  
                X3=X1+(y-y1)/m  
                      where y = ywmax  
                ywmax is the maximum value of Y co-ordinate of the window

# **SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<graphics.h>

#include<dos.h>

typedef struct coordinate

{

int x,y;

char code[4];

}PT;

void drawwindow();

void drawline(PT p1,PT p2);

PT setcode(PT p);

int visibility(PT p1,PT p2);

PT resetendpt(PT p1,PT p2);

int main()

{

PT p1,p2,p3,p4,ptemp;

printf("\nEnter x1 and y1\n");

scanf("%d %d",&p1.x,&p1.y);

printf("\nEnter x2 and y2\n");

scanf("%d %d",&p2.x,&p2.y);

initwindow(800, 800);

drawwindow();

delay(500);

drawline(p1,p2);

delay(500);

cleardevice();

delay(500);

p1=setcode(p1);

p2=setcode(p2);

int v=visibility(p1,p2);

delay(500);

switch(v)

{

case 0: drawwindow();

delay(500);

drawline(p1,p2);

break;

case 1: drawwindow();

delay(500);

break;

case 2: p3=resetendpt(p1,p2);

p4=resetendpt(p2,p1);

drawwindow();

delay(500);

drawline(p3,p4);

break;

}

delay(5000);

getch();

closegraph();

return 0;

}

void drawwindow()

{

line(150,100,450,100);

line(450,100,450,350);

line(450,350,150,350);

line(150,350,150,100);

}

void drawline(PT p1,PT p2)

{

line(p1.x,p1.y,p2.x,p2.y);

}

PT setcode(PT p) //for setting the 4 bit code

{

PT ptemp;

if(p.y<100)

ptemp.code[0]='1'; //Top

else

ptemp.code[0]='0';

if(p.y>350)

ptemp.code[1]='1'; //Bottom

else

ptemp.code[1]='0';

if(p.x>450)

ptemp.code[2]='1'; //Right

else

ptemp.code[2]='0';

if(p.x<150)

ptemp.code[3]='1'; //Left

else

ptemp.code[3]='0';

ptemp.x=p.x;

ptemp.y=p.y;

return(ptemp);

}

int visibility(PT p1,PT p2)

{

int i,flag=0;

for(i=0;i<4;i++)

{

if((p1.code[i]!='0') || (p2.code[i]!='0'))

flag=1;

}

if(flag==0)

return(0);

for(i=0;i<4;i++)

{

if((p1.code[i]==p2.code[i]) && (p1.code[i]=='1'))

flag='0';

}

if(flag==0)

return(1);

return(2);

}

PT resetendpt(PT p1,PT p2)

{

PT temp;

int x,y,i;

float m,k;

if(p1.code[3]=='1')

x=150;

if(p1.code[2]=='1')

x=450;

if((p1.code[3]=='1') || (p1.code[2]=='1'))

{

m=(float)(p2.y-p1.y)/(p2.x-p1.x);

k=(p1.y+(m\*(x-p1.x)));

temp.y=k;

temp.x=x;

for(i=0;i<4;i++)

temp.code[i]=p1.code[i];

if(temp.y<=350 && temp.y>=100)

return (temp);

}

if(p1.code[0]=='1')

y=100;

if(p1.code[1]=='1')

y=350;

if((p1.code[0]=='1') || (p1.code[1]=='1'))

{

m=(float)(p2.y-p1.y)/(p2.x-p1.x);

k=(float)p1.x+(float)(y-p1.y)/m;

temp.x=k;

temp.y=y;

for(i=0;i<4;i++)

temp.code[i]=p1.code[i];

return(temp);

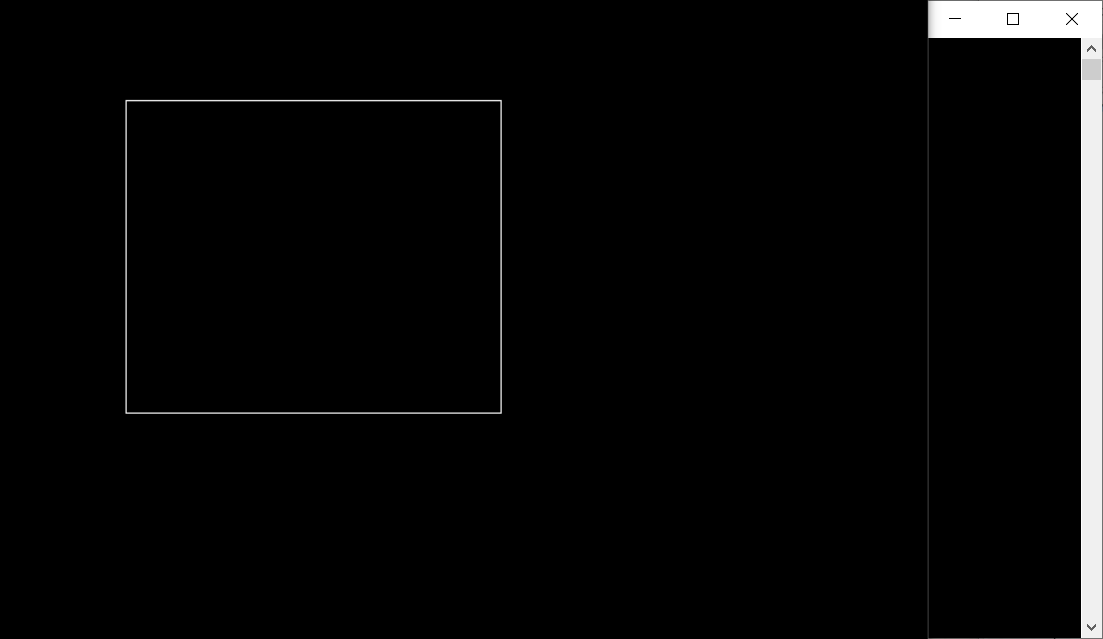
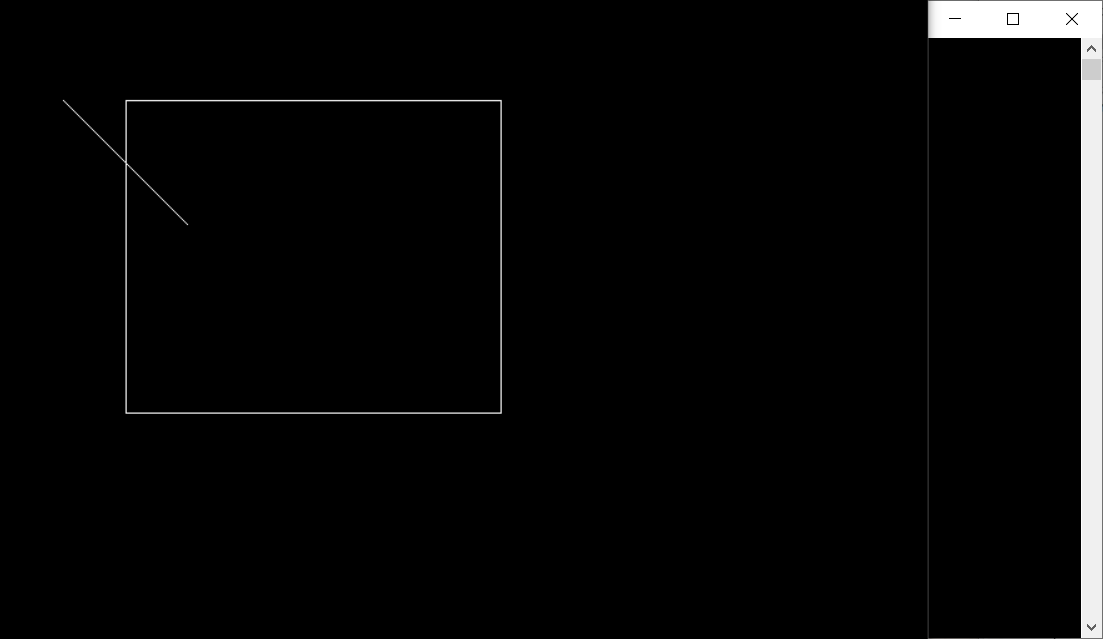
}

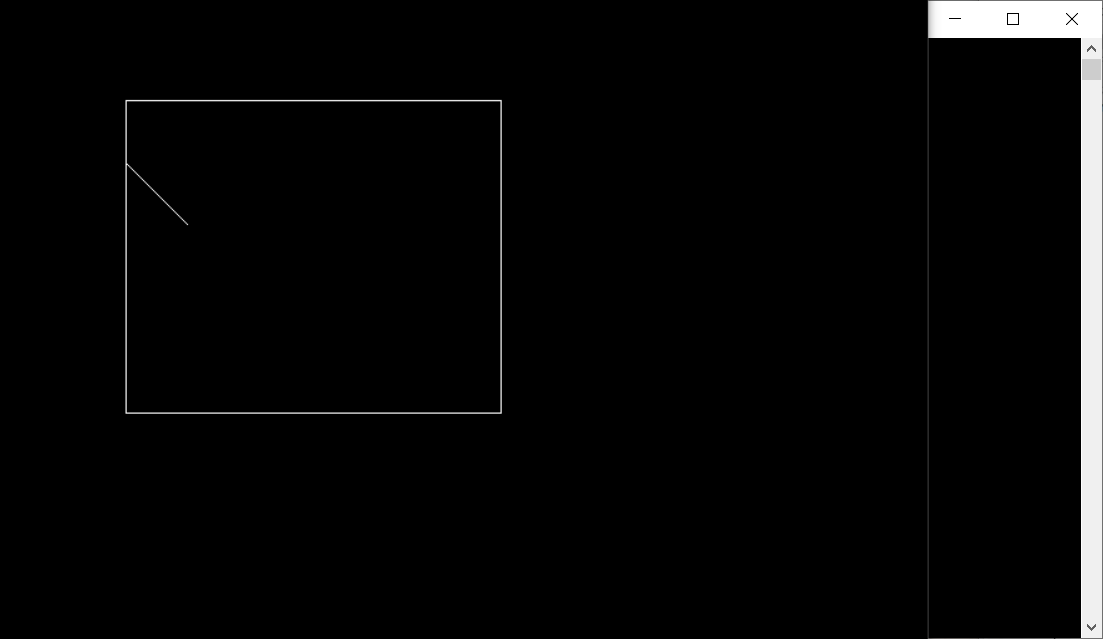
else

return(p1);

}

# **OUTPUT:**





**VIVA Voice**

**Q1. what is advantage of Cohen Sutherland line clipping algorithm?**

Ans.

1. It calculates end-points very quickly and rejects and accepts lines quickly.

2. It can clip pictures much large than screen size.

**Q2. Use the cohen sutherland line clipping algorithm to clip line P1 (70,20) and P2(100,10) against a window lower left corner(50,10) and upper right corner(80,40).**

Ans.

A line with starting point P & end point Q as the mid point

M1=( (-10+50)/2    (20+10)/2 )

M1=(20,15)

For line PM1:

|  |  |  |  |
| --- | --- | --- | --- |
| Starting point | End point | Mid point | Location |
| (-10,20) | (50,10) | (20,15) | Inside |
| (-10,20) | (20,15) | (5,17.5) | Inside |
| (-10,20) | (5,17) | (2.5,18.5) | Inside |
| (-10,20) | (2,18) | (4,19) | Inside |
| (-10,20) | (4,19 | (3,19.5) | Inside |

For lineM1Q:

|  |  |  |  |
| --- | --- | --- | --- |
| Starting point | End point | Mid point | Location |
| (20,15) | (50,10) | (35,12.5) | Inside |
| (20,15) | (35,12) | (27.5,13.5) | Inside |
| (20,15) | (27,13 | (23.5,14) | Inside |
| (20,15) | (23,14) | (21.5,14.5) | Inside |
| (20,15) | (21,14) | (20.5,14.5) | Inside |

**Q3. What is the time and space complexity of Cohen Sutherland line clipping algorithm?**

Ans.

1. Worst case time complexity: Θ(n)
2. Average case time complexity: Θ(n)
3. Best case time complexity: Θ(n)
4. Space complexity: Θ(1)

**Q4. What do you understand by interior and exterior clipping?**

Ans.

Clipping means Identifying portions of a scene that are inside (or outside) a specified region Examples Multiple viewport on a device Deciding how much of a games world the player can see.

In other words, Clipping is the process of removing the graphics parts either inside or outside the given region.

Interior clipping removes the parts outside the given window and exterior clipping removes the parts inside the given window.

**Q5. What is the difference between a window and a viewport?**

Ans.

|  |  |
| --- | --- |
| Window Port | Viewport |
| Window port is the coordinate area specially selected for the display. | Viewport is the display area of viewport in which the window is perfectly mapped. |
| Region Created according to World Coordinates. | Region Created according to Device Coordinates. |
| It is a region selected form the real world. It is a graphically control thing and composed of visual areas along with some of its program controlled with help of window decoration. | It is the region in computer graphics which is a polygon viewing region. |
| A window port can be defined with the help of a GWINDOW statement. | A viewport is defined by the GPORT command. |

**Q6. What are the different types of clipping algorithms?**

Ans.

There are five primitive types clipping, such as point, line, polygon or are, curve and text clipping. Classical line clipping algorithms includes Cohen–Sutherland algorithm, Midpoint Subdivision algorithm, Liang Bearsky and Nicholl-Lee-Nicholl algorithm.