Line Clipping:

It is performed by using the line clipping algorithm. The line clipping algorithms are:

1. Cohen Sutherland Line Clipping Algorithm
2. Midpoint Subdivision Line Clipping Algorithm
3. Liang-Barsky Line Clipping Algorithm

Cohen Sutherland Line Clipping Algorithm:

In the 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

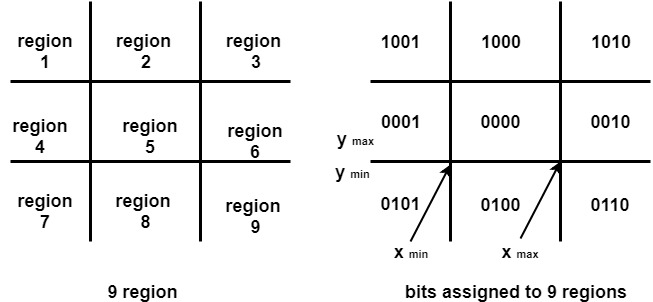
**1. Visible:** If a line lies within the window, i.e., both endpoints of the line lies within the window. A line is visible and will be displayed as it is.

**2. 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.

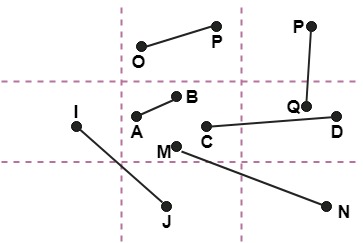
ymin,ymax are also coordinates of the window.  
          x1>xmax  
          x2>xmax  
          y1>ymax  
          y2>ymax  
          x1<xmin  
          x2<xmin  
          y1<ymin  
          y2<ymin

**3. 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

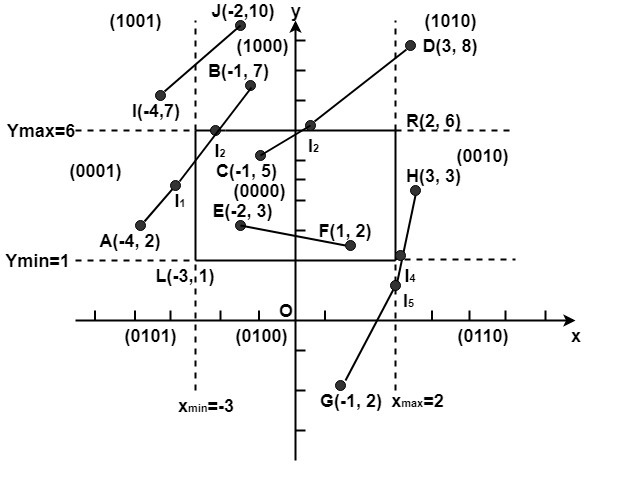
**(b)** If bit 2 is "1" line intersect with right boundary  
                y3=y1+m(X-X1)  
                where X = Xwmax  
                where X more is maximum value of X co-ordinate of the window

**(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

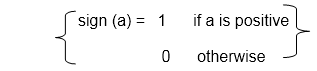
Example of Cohen-Sutherland Line Clipping Algorithm:

Let R be the rectangular window whose lower left-hand corner is at L (-3, 1) and upper right-hand corner is at R (2, 6). Find the region codes for the endpoints in fig:



The region code for point (x, y) is set according to the scheme  
Bit 1 = sign (y-ymax)=sign (y-6)         Bit 3 = sign (x-xmax)= sign (x-2)  
Bit 2 = sign (ymin-y)=sign(1-y)         Bit 4 = sign (xmin-x)=sign(-3-x)

Here



So

                A (-4, 2)→ 0001         F (1, 2)→ 0000  
                B (-1, 7) → 1000         G (1, -2) →0100  
                C (-1, 5)→ 0000         H (3, 3) → 0100  
                D (3, 8) → 1010         I (-4, 7) → 1001  
                E (-2, 3) → 0000         J (-2, 10) → 1000

We place the line segments in their appropriate categories by testing the region codes found in the problem.

**Category1 (visible):** EF since the region code for both endpoints is 0000.

**Category2 (not visible):** IJ since (1001) AND (1000) =1000 (which is not 0000).

**Category 3 (candidate for clipping):** AB since (0001) AND (1000) = 0000, CD since (0000) AND (1010) =0000, and GH. since (0100) AND (0010) =0000.

The candidates for clipping are AB, CD, and GH.

In clipping AB, the code for A is 0001. To push the 1 to 0, we clip against the boundary line xmin=-3. The resulting intersection point is I1 (-3,3Line Clipping). We clip (do not display) AI1 and I1 B. The code for I1is 1001. The clipping category for I1 B is 3 since (0000) AND (1000) is (0000). Now B is outside the window (i.e., its code is 1000), so we push the 1 to a 0 by clipping against the line ymax=6. The resulting intersection is l2 (-1Line Clipping,6). Thus I2 B is clipped. The code for I2 is 0000. The remaining segment I1 I2 is displayed since both endpoints lie in the window (i.e., their codes are 0000).

For clipping CD, we start with D since it is outside the window. Its code is 1010. We push the first 1 to a 0 by clipping against the line ymax=6. The resulting intersection I3 is (Line Clipping,6),and its code is 0000. Thus I3 D is clipped and the remaining segment CI3 has both endpoints coded 0000 and so it is displayed.

For clipping GH, we can start with either G or H since both are outside the window. The code for G is 0100, and we push the 1 to a 0 by clipping against the line ymin=1.The resulting intersection point is I4 (2Line Clipping,1) and its code is 0010. We clip GI4 and work on I4 H. Segment I4 H is not displaying since (0010) AND (0010) =0010.

Program to perform Line Clipping using Cohen Sutherland Algorithm:

1. #include <iostream.h>
2. #include <conio.h>
3. #include <graphics.h>
4. #include <dos.h>
5. **class** data
6. {
7. **int** gd, gmode, x, y, xmin,ymin,ymax,xmax;
8. **int** a1,a2;
9. **float** x1, y1,x2,y2,x3,y3;
10. **int** xs, ys, xe, ye;
11. **float** maxx,maxy;
12. **public**:
13. **void** getdata ();
14. **void** find ();
15. **void** clip ();
16. **void** display (**float**, **float**,**float**,**float**);
17. **void** checkonof (**int**);
18. **void** showbit (**int**);
19. };
20. **void** data :: getdata ()
21. {
22. cout<<"Enter the minimum and maximum coordinate of window (x, y) ";
23. cin >>xmin>>ymin>>xmax>>ymax;
24. cout<<"Enter the end points of the line to be clipped";
25. cin >>xs>>ys>>xe>>ye;
26. display (xs, ys, xe,ye);
27. }
28. **void** data :: display (**float**, xs, **float**, ys,**float** xe, **float** ye)
29. {
30. **int** gd=DETECT;
31. initgraph (&gd,&gmode, "");
32. maxx=getmaxx();
33. maxy=getmaxy();
34. line (maxx/2,0,maxx/2,maxy);
35. line (0, maxy/2,maxx,maxy/2);
36. rectangle (maxx/2+xmin,maxy/2-ymax,maxx/2+xmax,maxy/2-ymin);
37. line (maxx/2+xs,maxy/2-ys,maxx/2+xe,maxy/2-ye);
38. getch();
39. }
40. **void** data :: find ()
41. {
42. a1=0;
43. a2=0;
44. **if** ((ys-ymax)>0)
45. a1+=8;
46. **if** ((ymin-ys)>0)
47. a1+=4;
48. **if** ((xs-xmax)>0)
49. a1+=2;
50. **if** ((xmin-xs)>0)
51. a1+=1;
52. **if** ((ye-ymax)>0)
53. a2+=8;
54. **if** ((ymin-ye)>0)
55. a2+=4;
56. **if** ((xe-xmax)>0)
57. a2+=2;
58. **if** ((xmin-xe)>0)
59. a2+=1;
60. cout<<"\nThe area code of Ist point is ";
61. showbit (a1);
62. getch ();
63. cout <<"\nThe area code of 2nd point is ";
64. showbit (a2);
65. getch ();
66. }
67. **void** data :: showbit (**int** n)
68. {
69. **int** i,k, and;
70. **for** (i=3;i>=0;i--)
71. {
72. and =1<<i;
73. k = n?
74. k ==0?cout<<"0": cout<<"1\"";
75. }
76. }
77. **void** data ::clip()
78. {
79. **int** j=a1&a2;
80. **if** (j==0)
81. {
82. cout<<"\nLine is perfect candidate for clipping";
83. **if** (a1==0)
84. {
85. **else**
86. {
87. checkonof(a1);
88. x2=x1;y2=y1;
89. }
90. **if** (a2=0)
91. {
92. x3=xe; y3=ye;
93. }
94. **else**
95. {
96. checkonof (a2);
97. x3=x1; y3=y1;
98. }
99. xs=x2; ys=y2;xe=x3;ye=y3;
100. cout << endl;
101. display (xs,ys,xe,ye);
102. cout<<"Line after clipping";
103. getch ()
104. }
105. **else** **if** ((a1==0) && (a2=0))
106. {
107. cout <<"\n Line is in the visible region";
108. getch ();
109. }
110. }
111. **void** data :: checkonof (**int** i)
112. {
113. **int** j, k,l,m;
114. 1=i&1;
115. x1=0;y1=0;
116. **if** (1==1)
117. {
118. x1=xmin;
119. y1=ys+ ((x1-xs)/ (xe-xs))\*(ye-ys);
120. }
121. j=i&8;
122. **if** (j>0)
123. {
124. y1=ymax;
125. x1=xs+(y1-ys)/(ye-ys))\*(xe-xs);
126. }
127. k=i & 4;
128. **if** (k==1)
129. {
130. y1=ymin;
131. x1=xs+((y1-ys)/(ye-ys))\*(xe-xs);
132. }
133. m= i&2;
134. **if** (m==1)
135. {
136. x1=xmax;
137. y1=ys+ ((x1-xs)/ (xe-xs))\*(ye-ys);
138. }
139. main ()
140. {
141. data s;
142. clrscr();
143. s.getdata();
144. s.find();
145. getch();
146. closegraph ();
147. **return** ();
148. }

**Output:**

