**Experiment No:-10 Date:-\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Aim**:- To study and implement Cohen Sutherland line clipping algorithm.

**Theory:-**

The Cohen-Sutherland line clipping algorithm quickly detects and dispenses with two common and trivial cases. To clip a line, we need to consider only its endpoints. If both endpoints of a line lie inside the window, the entire line lies inside the window. It is trivially accepted and needs no clipping. On the other hand, if both endpoints of a line lie entirely to one side of the window, the line must lie entirely outside of the window. It is trivially and needs to be neither clipped nor displayed.

It uses a divide-and-conquer strategy. The line segment's endpoints are tested to see if the line can be trivially accepted or rejected. If the line cannot be trivally accepted or rejected, an intersection of the line with a window edge is determined and the trivial reject/accept test is repeated. This process is continued until the line is accepted.

**Algorithm:-**

To perform the trivial acceptance and rejection tests, we extend the edges of the window to divide the plane of the window into the nine regions. Each end point of the line segment is then assigned the code of the region in which it lies.

1. Given a line segment with endpoint and 2. Compute the 4-bit codes for each endpoint.

If both codes are 0000,(bitwise OR of the codes yields 0000 ) line lies completely inside the window: pass the endpoints to the draw routine.

If both codes have a 1 in the same bit position (bitwise AND of the codes is not 0000), the line lies outside the window. It can be trivially rejected.

3. If a line cannot be trivially accepted or rejected, at least one of the two endpoints must lie outside the window and the line segment crosses a window edge. This line must be clipped at the window edge before being passed to the drawing routine. 4. Examine one of the endpoints, say . Read 's 4-bit code in order: Left- to-Right, Bottom-to-Top. 5. When a set bit (1) is found, compute the intersection I of the corresponding window edge with the line from to . Replace with I and repeat the algorithm.

**Program:-**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>

class clips

{

private:

int x1[10],y1[10],x2[10],y2[10],xmin,ymin,xmax,ymax,n,i,a1[5], b1[5],c1[5],d1[5],a2[5],b2[5],c2[5],d2[5],j,l,m,x,y;

public:

void input();

void call();

};

void clips::input()

{

cout<<"Enter the values of Windows(xmin,ymin,xmax,ymax):\n";

cin>>xmin>>ymin>>xmax>>ymax;

rectangle(xmin,ymin,xmax,ymax);

cout<<"Enter the No: of Lines:\n";

cin>>n;

cout<<"Enter the Line Co-ordinate one by one\n";

for(i=1;i<=n;i++)

{

cin>>x1[i]>>y1[i]>>x2[i]>>y2[i];

line(x1[i],y1[i],x2[i],y2[i]);

}

getch();

}

void clips::call()

{

cout<<"The Clipped Image";

rectangle(xmin,ymin,xmax,ymax);

for(i=1;i<=n;i++)

{

if(x1[i]>xmin)

{

a1[i]=0;

}

else

{

a1[i]=1;

}

if(x1[i]<xmax)

{

b1[i]=0;

}

else

{

b1[i]=1;

}

if(y1[i]>ymin)

{

c1[i]=0;

}

else

{

c1[i]=1;

}

if(y1[i]<ymax)

{

d1[i]=0;

}

else

{

d1[i]=0;

}

if(x2[i]>xmin)

{

a2[i]=0;

}

else

{

a2[i]=1;

}

if(x2[i]<xmax)

{

b2[i]=0;

}

else

{

b2[i]=1;

}

if(y2[i]>ymin)

{

c2[i]=0;

}

else

{

c2[i]=1;

}

if(y2[i]<ymax)

{

d2[i]=0;

}

else

{

d2[i]=1;

}

if(a1[i]==0&&b1[i]==0&&c1[i]==0&&d1[i]==0&&a2[i]==0&&b2[i]==0

&&c2[i]==0&&d2[i]==0)

{

line(x1[i],y1[i],x2[i],y2[i]);

}

else if(((a1[i]&&b1[i]&&c1[i]&&d1[i]==1))&&((a1[i]&&b2[i]&&c2[i]

&&d2[i]==1)))

{

cout<<" ";

}

else

{

j=y2[i]-y1[i];

l=x2[i]-x1[i];

m=j/l;

if(a1[i]==0&&b1[i]==0&&c1[i]==0&&d1[i]==0)

{

if(x2[i]<xmin)

{

x=xmin;

y=y1[i]+(m\*(xmin-x1[i]));

}

else if(x2[i]>xmax)

{

x=xmax;

y=y1[i]+(m\*(xmax-x1[i]));

}

else if(y2[i]<ymin)

{

y=ymin;

x=x1[i]+((ymin-y1[i])/m);

}

else if(y2[i]>ymax)

{

y=ymax;

x=x1[i]+((ymax-y1[i])/m);

}

line(x,y,x1[i],y1[i]);

}

else if(a2[i]==0&&b2[i]==0&&c2[i]==0&&d2[i]==0)

{

if(x1[i]<xmin)

{

x=xmin;

y=y1[i]+(m\*(xmin-x1[i]));

}

else if(x1[i]>xmax)

{

x=xmax;

y=y1[i]+(m\*(xmax-x1[i]));

}

else if(y1[i]<ymin)

{

y=ymin;

x=x1[i]+((ymin-y1[i])/m);

}

else if(y1[i]>ymax)

{

y=ymax;

x=x1[i]+((ymax-y1[i])/m);

}

line(x,y,x2[i],y2[i]);

}

}

}

}

void main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"..\\BGI");

clips clip;

clip.input();

delay(10);

clearviewport();

clip.call();

getch();

closegraph();

}