```
#include <stdio.h>
#include <GL/glut.h>
#include<stdbool.h>
double xmin=50,ymin=50, xmax=100,ymax=100; // Window boundaries
double xvmin=200,yvmin=200,xvmax=300,yvmax=300; // Viewport boundaries
//bit codes for the right, left, top, & bottom
const int TOP = 8;
const int BOTTOM = 4;
const int RIGHT= 2;
const int LEFT = 1;
//used to compute bit codes of a intersecting point
int ComputeOutCode (double x, double y);
//Cohen-Sutherland clipping algorithm clips a line from
//P0 = (x0, y0) to P1 = (x1, y1) against a rectangle with
//diagonal from (xmin, ymin) to (xmax, ymax).
void CohenSutherlandLineClipAndDraw (double x0, double y0,double x1, double y1)
{
//Outcodes for P0, P1, and whatever point lies outside the clip rectangle
int outcode0, outcode1, outcodeOut;
bool accept = false, done = false;
//compute outcodes
outcode0 = ComputeOutCode (x0, y0);
outcode1 = ComputeOutCode (x1, y1);
do
{
if ((outcode0 |outcode1)==0) //logical or is 0 Trivially accept & exit
{
accept = true;
done = true;
}
else if (outcode0 & outcode1) //logical and is not 0. Trivially reject and exit
```

```
done = true;
else
{
//failed both tests, so calculate the line segment to clip
//from an outside point to an intersection with clip edge
double x, y;
//At least one endpoint is outside the clip rectangle; pick it.
outcodeOut = outcode0? outcode0: outcode1;
float slope=(y1-y0)/(x1-x0);
//Now find the intersection point;
//use formulas y = y0 + slope * (x - x0), x = x0 + (1/slope)* (y - y0)
if (outcodeOut & TOP) //point is above the clip rectangle
{
x = x0 + (1/slope) * (ymax - y0);
y = ymax;
}
else if (outcodeOut & BOTTOM) //point is below the clip rectangle
{
x = x0 + (1/slope) * (ymin - y0);
y = ymin;
}
else if (outcodeOut & RIGHT) //point is to the right of clip rectangle
{
y = y0 + slope * (xmax - x0);
x = xmax;
} else //point is to the left of clip rectangle
{
y = y0 + slope* (xmin - x0);
x = xmin;
}
```

```
//Now we move outside point to intersection point to clip
//and get ready for next pass.
if (outcodeOut == outcode0)
{
x0 = x;
y0 = y;
outcode0 = ComputeOutCode (x0, y0);
}
else
{
x1 = x;
y1 = y;
outcode1 = ComputeOutCode (x1, y1);
}
}
}
while (!done);
if (accept)
{
// Window to viewport mappings
double sx=(xvmax-xvmin)/(xmax-xmin); // Scale parameters
double sy=(yvmax-yvmin)/(ymax-ymin);
double vx0=xvmin+(x0-xmin)*sx;
double vy0=yvmin+(y0-ymin)*sy;
double vx1=xvmin+(x1-xmin)*sx;
double vy1=yvmin+(y1-ymin)*sy;
//draw a red colored viewport
glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINE_LOOP);
glVertex2f(xvmin, yvmin);
```

```
glVertex2f(xvmax, yvmin);
glVertex2f(xvmax, yvmax);
glVertex2f(xvmin, yvmax);
glEnd();
glColor3f(0.0,0.0,1.0); // draw blue colored clipped line
glBegin(GL_LINES);
glVertex2d (vx0, vy0);
glVertex2d (vx1, vy1);
glEnd();
}
}
//Compute the bit code for a point (x, y) using the clip rectangle
//bounded diagonally by (xmin, ymin), and (xmax, ymax)
int ComputeOutCode (double x, double y)
{
int code = 0;
if (y > ymax) //above the clip window
code |= TOP;
else if (y < ymin) //below the clip window
code |= BOTTOM;
if (x > xmax) //to the right of clip window
code |= RIGHT;
else if (x < xmin) //to the left of clip window
code |= LEFT;
return code;
}
void display()
{
double x0=60,y0=20,x1=80,y1=120;
glClear(GL_COLOR_BUFFER_BIT);
```

```
//draw the line with red color
glColor3f(1.0,0.0,0.0);
//bres(120,20,340,250);
glBegin(GL_LINES);
glVertex2d (x0, y0);
glVertex2d (x1, y1);
glEnd();
//draw a blue colored window
glColor3f(0.0, 0.0, 1.0);
glBegin(GL_LINE_LOOP);
glVertex2f(xmin, ymin);
glVertex2f(xmax, ymin);
glVertex2f(xmax, ymax);
glVertex2f(xmin, ymax);
glEnd();
CohenSutherlandLineClipAndDraw(x0,y0,x1,y1);
glFlush();
}
void myinit()
{
glClearColor(1.0,1.0,1.0,1.0);
glColor3f(1.0,0.0,0.0);
glPointSize(1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0,499.0,0.0,499.0);
}
void main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
```

```
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
glutCreateWindow("Cohen Suderland Line Clipping Algorithm");
glutDisplayFunc(display);
myinit();
glutMainLoop();
}
```