**Program 8:**

**Write a program to implement the Cohen-Hodgeman polygon clipping algorithm. Make provision to specify the input polygon and window for clipping.**

// algorithm for polygon clipping

#include<iostream>

#include<GL/glut.h>

using namespace std;

int poly\_size, poly\_points[20][2], org\_poly\_size, org\_poly\_points[20][2], clipper\_size, clipper\_points[20][2];

const int MAX\_POINTS = 20;

// Returns x-value of point of intersection of two

// lines

void drawPoly(int p[][2], int n) {

glBegin(GL\_POLYGON);

for (int i = 0; i < n; i++)

glVertex2f(p[i][0], p[i][1]);

glEnd();

}

int x\_intersect(int x1, int y1, int x2, int y2,

int x3, int y3, int x4, int y4)

{

int num = (x1 \* y2 - y1 \* x2) \* (x3 - x4) -

(x1 - x2) \* (x3 \* y4 - y3 \* x4);

int den = (x1 - x2) \* (y3 - y4) - (y1 - y2) \* (x3 - x4);

return num / den;

}

// Returns y-value of point of intersectipn of

// two lines

int y\_intersect(int x1, int y1, int x2, int y2,

int x3, int y3, int x4, int y4)

{

int num = (x1 \* y2 - y1 \* x2) \* (y3 - y4) -

(y1 - y2) \* (x3 \* y4 - y3 \* x4);

int den = (x1 - x2) \* (y3 - y4) - (y1 - y2) \* (x3 - x4);

return num / den;

}

// This functions clips all the edges w.r.t one clip

// edge of clipping area

void clip(int poly\_points[][2], int& poly\_size,

int x1, int y1, int x2, int y2)

{

int new\_points[MAX\_POINTS][2], new\_poly\_size = 0;

// (ix,iy),(kx,ky) are the co-ordinate values of

// the points

for (int i = 0; i < poly\_size; i++)

{

// i and k form a line in polygon

int k = (i + 1) % poly\_size;

int ix = poly\_points[i][0], iy = poly\_points[i][1];

int kx = poly\_points[k][0], ky = poly\_points[k][1];

// Calculating position of first point

// w.r.t. clipper line

int i\_pos = (x2 - x1) \* (iy - y1) - (y2 - y1) \* (ix - x1);

// Calculating position of second point

// w.r.t. clipper line

int k\_pos = (x2 - x1) \* (ky - y1) - (y2 - y1) \* (kx - x1);

// Case 1 : When both points are inside

if (i\_pos >= 0 && k\_pos >= 0)

{

//Only second point is added

new\_points[new\_poly\_size][0] = kx;

new\_points[new\_poly\_size][1] = ky;

new\_poly\_size++;

}

// Case 2: When only first point is outside

else if (i\_pos < 0 && k\_pos >= 0)

{

// Point of intersection with edge

// and the second point is added

new\_points[new\_poly\_size][0] = x\_intersect(x1,

y1, x2, y2, ix, iy, kx, ky);

new\_points[new\_poly\_size][1] = y\_intersect(x1,

y1, x2, y2, ix, iy, kx, ky);

new\_poly\_size++;

new\_points[new\_poly\_size][0] = kx;

new\_points[new\_poly\_size][1] = ky;

new\_poly\_size++;

}

// Case 3: When only second point is outside

else if (i\_pos >= 0 && k\_pos < 0)

{

//Only point of intersection with edge is added

new\_points[new\_poly\_size][0] = x\_intersect(x1,

y1, x2, y2, ix, iy, kx, ky);

new\_points[new\_poly\_size][1] = y\_intersect(x1,

y1, x2, y2, ix, iy, kx, ky);

new\_poly\_size++;

}

// Case 4: When both points are outside

else

{

//No points are added

}

}

// Copying new points into original array

// and changing the no. of vertices

poly\_size = new\_poly\_size;

for (int i = 0; i < poly\_size; i++)

{

poly\_points[i][0] = new\_points[i][0];

poly\_points[i][1] = new\_points[i][1];

}

}

void init() {

glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.0, 500.0, 0.0, 500.0, 0.0, 500.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

}

// Implements Sutherland�Hodgman algorithm

void display()

{

init();

glColor3f(1.0f, 0.0f, 0.0f);

drawPoly(clipper\_points, clipper\_size);

glColor3f(0.0f, 1.0f, 0.0f);

drawPoly(org\_poly\_points, org\_poly\_size);

//i and k are two consecutive indexes

for (int i = 0; i < clipper\_size; i++)

{

int k = (i + 1) % clipper\_size;

// We pass the current array of vertices, it's size

// and the end points of the selected clipper line

clip(poly\_points, poly\_size, clipper\_points[i][0],

clipper\_points[i][1], clipper\_points[k][0],

clipper\_points[k][1]);

}

glColor3f(0.0f, 0.0f, 1.0f);

drawPoly(poly\_points, poly\_size);

glFlush();

}

//Driver code

int main(int argc, char\* argv[])

{

printf("Enter no. of vertices: \n");

scanf\_s("%d", &poly\_size);

org\_poly\_size = poly\_size;

for (int i = 0; i < poly\_size; i++)

{

printf("Polygon Vertex:\n");

scanf\_s("%d%d", &poly\_points[i][0], &poly\_points[i][1]);

org\_poly\_points[i][0] = poly\_points[i][0];

org\_poly\_points[i][1] = poly\_points[i][1];

}

printf("Enter no. of vertices of clipping window:");

scanf\_s("%d", &clipper\_size);

for (int i = 0; i < clipper\_size; i++)

{

printf("Clip Vertex:\n");

scanf\_s("%d%d", &clipper\_points[i][0], &clipper\_points[i][1]);

}

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(400, 400);

glutInitWindowPosition(100, 100);

glutCreateWindow("Polygon Clipping!");

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**OUTPUT:**

