**Program 1 : Write a program to clip a polygon using Sutherland Hodgeman algorithm .**

#include <iostream>

#include <graphics.h>

using namespace std;

int xmin, xmax, ymin, ymax;

class Coordinate

{

public:

    int X;

    int Y;

    Coordinate() {}

    Coordinate(float x, float y)

    {

        X = x;

        Y = y;

    }

};

Coordinate pointOfIntersection(Coordinate s1, Coordinate e1, Coordinate s2, Coordinate e2)

{

    float a1 = e1.Y - s1.Y;

    float b1 = s1.X - e1.X;

    float c1 = a1 \* s1.X + b1 \* s1.Y;

    float a2 = e2.Y - s2.Y;

    float b2 = s2.X - e2.X;

    float c2 = a2 \* s2.X + b2 \* s2.Y;

    float delta = a1 \* b2 - a2 \* b1;

    return Coordinate((b2 \* c1 - b1 \* c2) / delta, (a1 \* c2 - a2 \* c1) / delta);

}

int main()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, "C:\\MinGW\\lib\\libbgi.a");

    int i, j, choice = 1;

    cout << "Enter co-ordinates for window ABCD:\n\n";

    cout << "xmin, xmax, ymin, ymax: ";

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

    rectangle(xmin, ymin, xmax, ymax);

    do

    {

        int n;

        cout << "\n\nEnter number of vertices (max 10): ";

        cin >> n;

        Coordinate V[n];

        cout << "Enter co-ordinates of vertices\n";

        int drawPolygon[n \* 2], newPolygon[100];

        for (i = 0, j = 0; i < n; i++)

        {

            cout << "Enter x" << i + 1 << " and y" << i + 1 << ": ";

            cin >> V[i].X >> V[i].Y;

            drawPolygon[j++] = V[i].X;

            drawPolygon[j++] = V[i].Y;

        }

        drawPolygon[j++] = V[0].X;

        drawPolygon[j] = V[0].Y;

        setcolor(MAGENTA);

        drawpoly(n + 1, drawPolygon);

        Coordinate top\_left(xmin, ymin), top\_right(xmax, ymin),

            bot\_left(xmin, ymax), bot\_right(xmax, ymax);

        int newN;

        cout << endl;

        for (i = 0, j = 0, newN = 0; i < n; i++)

        {

            Coordinate temp;

            if (V[i].X < xmin)

            {

                if (V[(i + 1) % n].X < xmin)

                    cout << "\nOut->Out";

                else

                {

                    temp = pointOfIntersection(V[i], V[(i + 1) % n], top\_left, bot\_left);

                    newN++;

                    newPolygon[j++] = temp.X;

                    newPolygon[j++] = temp.Y;

                    newN++;

                    newPolygon[j++] = V[(i + 1) % n].X;

                    newPolygon[j++] = V[(i + 1) % n].Y;

                    cout << "\nOut->In";

                }

            }

            else

            {

                if (V[(i + 1) % n].X < xmin)

                {

                    temp = pointOfIntersection(V[i], V[(i + 1) % n], top\_left,bot\_left);

                    newN++;

                    newPolygon[j++] = temp.X;

                    newPolygon[j++] = temp.Y;

                    cout << "\nIn->Out";

                }

                else

                {

                    newN++;

                    newPolygon[j++] = V[(i + 1) % n].X;

                    newPolygon[j++] = V[(i + 1) % n].Y;

                    cout << "\nIn->In";

                }

            }

        }

        newPolygon[j++] = newPolygon[0];

        newPolygon[j] = newPolygon[1];

        setcolor(RED);

        cout << "\nRed coordinates: ";

        for (i = 0; i <= j; i++)

            cout << newPolygon[i] << " ";

        drawpoly(newN + 1, newPolygon);

        for (i = 0, j = 0; i < newN; i++)

        {

            V[i].X = newPolygon[j++];

            V[i].Y = newPolygon[j++];

        }

        n = newN;

        cout << endl;

        for (i = 0, j = 0, newN = 0; i < n; i++)

        {

            Coordinate temp;

            if (V[i].X > xmax)

            {

                if (V[(i + 1) % n].X > xmax)

                    cout << "\nOut->Out";

                else

                {

                    temp = pointOfIntersection(V[i], V[(i + 1) % n], top\_right,bot\_right);

                    newN++;

                    newPolygon[j++] = temp.X;

                    newPolygon[j++] = temp.Y;

                    newN++;

                    newPolygon[j++] = V[(i + 1) % n].X;

                    newPolygon[j++] = V[(i + 1) % n].Y;

                    cout << "\nOut->In";

                }

            }

            else

            {

                if (V[(i + 1) % n].X > xmax)

                {

                    temp = pointOfIntersection(V[i], V[(i + 1) % n], top\_right,bot\_right);

                    newN++;

                    newPolygon[j++] = temp.X;

                    newPolygon[j++] = temp.Y;

                    cout << "\nIn->Out";

                }

                else

                {

                    newN++;

                    newPolygon[j++] = V[(i + 1) % n].X;

                    newPolygon[j++] = V[(i + 1) % n].Y;

                    cout << "\nIn->In";

                }

            }

        }

        newPolygon[j++] = newPolygon[0];

        newPolygon[j] = newPolygon[1];

        setcolor(GREEN);

        cout << "\nGreen coordinates: ";

        for (i = 0; i <= j; i++)

            cout << newPolygon[i] << " ";

        drawpoly(newN + 1, newPolygon);

        for (i = 0, j = 0; i < newN; i++)

        {

            V[i].X = newPolygon[j++];

            V[i].Y = newPolygon[j++];

        }

        n = newN;

        cout << endl;

        for (i = 0, j = 0, newN = 0; i < n; i++)

        {

            Coordinate temp;

            if (V[i].Y > ymax)

            {

                if (V[(i + 1) % n].Y > ymax)

                    cout << "\nOut->Out";

                else

                {

                    temp = pointOfIntersection(V[i], V[(i + 1) % n], bot\_left,bot\_right);

                    newN++;

                    newPolygon[j++] = temp.X;

                    newPolygon[j++] = temp.Y;

                    newN++;

                    newPolygon[j++] = V[(i + 1) % n].X;

                    newPolygon[j++] = V[(i + 1) % n].Y;

                    cout << "\nOut->In";

                }

            }

            else

            {

                if (V[(i + 1) % n].Y > ymax)

                {

                    temp = pointOfIntersection(V[i], V[(i + 1) % n], bot\_left, bot\_right);

                    newN++;

                    newPolygon[j++] = temp.X;

                    newPolygon[j++] = temp.Y;

                    cout << "\nIn->Out";

                }

                else

                {

                    newN++;

                    newPolygon[j++] = V[(i + 1) % n].X;

                    newPolygon[j++] = V[(i + 1) % n].Y;

                    cout << "\nIn->In";

                }

            }

        }

        newPolygon[j++] = newPolygon[0];

        newPolygon[j] = newPolygon[1];

        setcolor(BLUE);

        cout << "\nBlue coordinates: ";

        for (i = 0; i <= j; i++)

            cout << newPolygon[i] << " ";

        drawpoly(newN + 1, newPolygon);

        for (i = 0, j = 0; i < newN; i++)

        {

            V[i].X = newPolygon[j++];

            V[i].Y = newPolygon[j++];

        }

        n = newN;

        cout << endl;

        for (i = 0, j = 0, newN = 0; i < n; i++)

        {

            Coordinate temp;

            if (V[i].Y < ymin)

            {

                if (V[(i + 1) % n].Y < ymin)

                    cout << "\nOut->Out";

                else

                {

                    temp = pointOfIntersection(V[i], V[(i + 1) % n], top\_left,

                                               top\_right);

                    newN++;

                    newPolygon[j++] = temp.X;

                    newPolygon[j++] = temp.Y;

                    newN++;

                    newPolygon[j++] = V[(i + 1) % n].X;

                    newPolygon[j++] = V[(i + 1) % n].Y;

                    cout << "\nOut->In";

                }

            }

            else

            {

                if (V[(i + 1) % n].Y < ymin)

                {

                    temp = pointOfIntersection(V[i], V[(i + 1) % n], top\_left,

                                               top\_right);

                    newN++;

                    newPolygon[j++] = temp.X;

                    newPolygon[j++] = temp.Y;

                    cout << "\nIn->Out";

                }

                else

                {

                    newN++;

                    newPolygon[j++] = V[(i + 1) % n].X;

                    newPolygon[j++] = V[(i + 1) % n].Y;

                    cout << "\nIn->In";

                }

            }

        }

        newPolygon[j++] = newPolygon[0];

        newPolygon[j] = newPolygon[1];

        for (i = 0, j = 0; i < newN; i++)

        {

            V[i].X = newPolygon[j++];

            V[i].Y = newPolygon[j++];

        }

        n = newN;

        setcolor(YELLOW);

        drawpoly(newN + 1, newPolygon);

        cout << "\n\nContinue?\t1.Yes\t0.No:\t";

        cin >> choice;

    } while (choice != 0);

    getch();

    closegraph();

}



