Name : Priyanka Patil

Email Id: patilpriyanka223@gamil.com

Mobile no: 8788308292

---------------------------------------------------------------------------------------------------------------------**Program 1: Check if the given point lies inside or outside a polygon?**

**#include** <iostream>

**using** **namespace** std; // Define Infinite (Using INT\_MAX caused overflow problems)

**#define** INF 10000

**struct** Point

{

**double** x;

**double** y;

};

// Given three colinear points p, q, r, the function checks if

// point q lies on line segment 'pr'

**bool** **onSegment**(Point p, Point q, Point r)

{

**if** (q.x <= max(p.x, r.x) && q.x >= min(p.x, r.x) &&

q.y <= max(p.y, r.y) && q.y >= min(p.y, r.y))

**return** **true**;

**return** **false**;

}

// To find orientation of ordered triplet (p, q, r).

// The function returns following values

// 0 --> p, q and r are colinear

// 1 --> Clockwise

// 2 --> Counterclockwise

**int** **orientation**(Point p, Point q, Point r)

{

**int** val = (q.y - p.y) \* (r.x - q.x) -

(q.x - p.x) \* (r.y - q.y);

**if** (val == 0) **return** 0; // colinear

**return** (val > 0)? 1: 2; // clock or counterclock wise

}

// The function that returns true if line segment 'p1q1'

// and 'p2q2' intersect.

**bool** **doIntersect**(Point p1, Point q1, Point p2, Point q2)

{

// Find the four orientations needed for general and

// special cases

**int** o1 = orientation(p1, q1, p2);

**int** o2 = orientation(p1, q1, q2);

**int** o3 = orientation(p2, q2, p1);

**int** o4 = orientation(p2, q2, q1);

// General case

**if** (o1 != o2 && o3 != o4)

**return** **true**;

// Special Cases

// p1, q1 and p2 are colinear and p2 lies on segment p1q1

**if** (o1 == 0 && onSegment(p1, p2, q1)) **return** **true**;

// p1, q1 and p2 are colinear and q2 lies on segment p1q1

**if** (o2 == 0 && onSegment(p1, q2, q1)) **return** **true**;

// p2, q2 and p1 are colinear and p1 lies on segment p2q2

**if** (o3 == 0 && onSegment(p2, p1, q2)) **return** **true**;

// p2, q2 and q1 are colinear and q1 lies on segment p2q2

**if** (o4 == 0 && onSegment(p2, q1, q2)) **return** **true**;

**return** **false**; // Doesn't fall in any of the above cases

}

// Returns true if the point p lies inside the polygon[] with n vertices

**bool** **isInside**(Point polygon[], **int** n, Point p)

{

// There must be at least 3 vertices in polygon[]

**if** (n < 3) **return** **false**;

// Create a point for line segment from p to infinite

Point extreme = {INF, p.y};

// Count intersections of the above line with sides of polygon

**int** count = 0, i = 0;

**do**

{

**int** next = (i+1)%n;

// Check if the line segment from 'p' to 'extreme' intersects

// with the line segment from 'polygon[i]' to 'polygon[next]'

**if** (doIntersect(polygon[i], polygon[next], p, extreme))

{

// If the point 'p' is colinear with line segment 'i-next',

// then check if it lies on segment. If it lies, return true,

// otherwise false

**if** (orientation(polygon[i], p, polygon[next]) == 0)

**return** onSegment(polygon[i], p, polygon[next]);

count++;

}

i = next;

} **while** (i != 0);

// Return true if count is odd, false otherwise

**return** count&1; // Same as (count%2 == 1)

}

// Driver program to test above functions

**int** **main**()

{

**int** n;

cout<<"Enter number of point of polygon";

cin>>n;

**struct** Point polygon1[n];

**struct** Point p;

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

{

cout<<"Enter "<<i+1<<"th x coordinate"<<**endl**;

cin>>polygon1[i].x;

cout<<"Enter "<<i+1<<"th y coordinate"<<**endl**;

cin>>polygon1[i].y;

}

cout<<"Enter the point p"<<**endl**;

cout<<"Enter the X coordinate"<<**endl**;

cin>>p.x;

cout<<"Enter the Y coordinate"<<**endl**;

cin>>p.y;

isInside(polygon1, n, p)? cout << "True \n": cout << "False\n";

**return** 0;

}