МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Вычислительная техника»

Дисциплина «Геометрическое моделирование»

**Лабораторная работа №4.**

**Вычислительная геометрия**

**3 вариант**

Выполнил:

студент группы ИВТАПбд-31.

Вершинин Д. В.

Проверил:

Войт Н. Н.

Ульяновск, 2018

**Техническое задание:** Методом трассировки луча определить находится ли точка с заданными координатами внутри полигона с заданными вершинами.

**Ход работы:** Один из стандартных методов определения принадлежности точки произвольному простому многоугольнику заключается в следующем. Выпустим луч из данной точки в произвольном направлении (например, в положительном направлении горизонтальной оси), и посчитаем сколько раз луч пересекает рёбра многоугольника. Для этого достаточно пройтись в цикле по рёбрам многоугольника и определить, пересекает ли луч каждое ребро. Если число пересечений нечётно, то объявляется, что точка лежит внутри многоугольника, если чётно — то снаружи.

Для реализации алгоритма необходимо реализовать функцию, определяющую будут ли пересекаться отрезки с определенными координатами. Воспользуемся ориентированной площадью треугольника, то есть площадью треугольника, взятой со знаком плюс или минус, в зависимости от типа поворота, образуемого вершинами этого треугольника. Действительно, чтобы отрезки AB и CD пересекались, необходимо и достаточно, чтобы точки A и B находились по разные стороны прямой CD, и, аналогично, точки C и D — по разные стороны прямой AB. Проверить это можно, вычисляя ориентированные площади соответствующих треугольников и сравнивая их знаки. Если знаки различны, то отрезки будут пересекаться.

Для реализации основного алгоритма будем считывать координаты вершин, и запоминать стороны многоугольника, образованные этими вершинами. Затем считаем с консоли координаты точки, которую необходимо проверить и формируем луч. Луч можно представить, как отрезок с длиной намного превышающей размеры многоугольника.

**Исходный код алгоритма:**

#include <iostream>

#include<vector>

#include <algorithm>

using namespace std;

struct pt {

int x, y;

};

struct edge {

pt a, b;

};

int area(pt a, pt b, pt c) {

return (b.x - a.x) \* (c.y - a.y) - (b.y - a.y) \* (c.x - a.x);

}

bool intersect\_1(int a, int b, int c, int d) {

if (a > b) swap(a, b);

if (c > d) swap(c, d);

return max(a, c) <= min(b, d);

}

bool intersect(pt a, pt b, pt c, pt d) {

return intersect\_1(a.x, b.x, c.x, d.x)

&& intersect\_1(a.y, b.y, c.y, d.y)

&& area(a, b, c) \* area(a, b, d) <= 0

&& area(c, d, a) \* area(c, d, b) <= 0;

}

int main() {

int n;

cin >> n;

vector<edge> ed;

pt a, b;

cin >> a.x >> a.y;

for (int i = 0; i < n - 1; i++) {

cin >> b.x >> b.y;

edge edge;

edge.a = a;

edge.b = b;

ed.push\_back(edge);

a = b;

}

edge edge;

edge.a = ed[n - 2].b;

edge.b = ed[0].a;

ed.push\_back(edge);

pt point\_b;

cin >> point\_b.x >> point\_b.y;

pt point\_end;

point\_end.x = point\_b.x + 100000;

point\_end.y = point\_b.y;

int counter = 0;

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

counter += intersect(ed[i].a, ed[i].b, point\_b, point\_end);

}

cout << counter << endl;

if (counter % 2 == 0) {

cout << "Outside";

}

else {

cout << "Inside";

}

}