МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА КОМП’ЮТЕРНИХ ТЕХНОЛОГІЙ

**ЗВІТ**

лабораторної роботи №2

з дисципліни «Об’єктно орієнтоване програмування»

на тему «Реалізація алгоритмів у функціональному стилі»

**ВИКОНАВ:**

Студент 1-го курсу

Групи ФЕП-14

Іванський Остап Юрійович

**ПЕРЕВІРИВ:**

Асистент кафедри РКТ

Левуш Павло Назарович

Львів 2025

**МЕТА**: Реалізувати алгоритми у функціональному стилі мовою С++

**ПОСИЛАННЯ**: https://github.com/Ostap-Ivanskyi/OOP/tree/main/Lab3

**ЗАВДАННЯ:**

Реалізувати альтернативний метод перевірки приналежності точки трикутнику за допомогою методу векторного добутку.

Додати перевірку, чи є трикутник виродженим (тобто якщо його площа дорівнює нулю).

Модифікувати код так, щоб програма працювала з довільною кількістю введених точок і виводила результат для кожної з них.

Розширити програму можливістю введення координат трикутника та точки користувачем.

Додати обробку випадків, коли точка лежить на межі трикутника.

Власну реалізацію імплементувати у вигляді трьох файлів:   
файл реалізації, файл заголовків та файл із точкою входу у програму.

**ХІД РОБОТИ**

1.У файлі Triangle.h оголошуємо структури Triangle з полями a,b,c – вершини трикутника, Point з полями x, y – координати точки та методи роботи з ними:

* **area()** — обчислює площу трикутника за формулою Герона.
* **containsByArea(const Point& p)** — перевіряє, чи належить точка трикутнику, порівнюючи площі.
* **crossProduct(...)** — обчислює векторний добуток для трьох точок.
* **isDegenerate()** — перевіряє, чи є трикутник виродженим (площа ≈ 0).
* **containsByCrossProduct(const Point& p)** — перевіряє приналежність точки за допомогою векторного добутку.
* **isOnEdge(const Point& p)** — визначає, чи лежить точка на ребрі трикутника.
* **distance(p1, p2)** — обчислює відстань між двома точками.

**Triangle.h**

#ifndef TRIANGLE\_H  
#define TRIANGLE\_H  
  
struct Point {  
 double x, y;  
};  
  
struct Triangle {  
 Point a, b, c;  
  
 // Area method functions  
 double area() const;  
 bool containsByArea(const Point& p) const;  
  
 // Cross product method functions  
 double crossProduct(const Point& p1, const Point& p2, const Point& p3) const;  
 bool isDegenerate() const;  
 bool containsByCrossProduct(const Point& p) const;  
 bool isOnEdge(const Point& p) const;  
};  
  
// Helper functions  
double distance(const Point& p1, const Point& p2);  
  
  
#endif // TRIANGLE\_H

2.У файлі Triangle.cpp виконуємо реалізацію методів.

**distance(p1, p2)**  
Обчислює відстань між двома точками за формулою відстані між двома точками на площині.

double distance(const Point& p1, const Point& p2) {  
 return sqrt(pow(p2.x - p1.x, 2) + pow(p2.y - p1.y, 2));  
}

**area(t)**  
Обчислює площу трикутника за формулою Герона.

double Triangle::area() const {  
 double sideA = distance(a, b);  
 double sideB = distance(b, c);  
 double sideC = distance(c, a);  
 double s = (sideA + sideB + sideC) / 2.0;  
 return sqrt(s \* (s - sideA) \* (s - sideB) \* (s - sideC));  
}

**Triangle::containsByArea(p)**  
Перевіряє, чи належить точка p трикутнику, порівнюючи площі. Якщо сума площ цих трикутників ≈ площі основного трикутника, то точка всередині.

bool Triangle::containsByArea(const Point& p) const {  
 Triangle t1 = {a, b, p};  
 Triangle t2 = {b, c, p};  
 Triangle t3 = {c, a, p};  
 double s\_main = area();  
 double s\_sum = t1.area() + t2.area() + t3.area();  
 return std::fabs(s\_main - s\_sum) < 1e-9;  
}

**Triangle::crossProduct(p1, p2, p3)**  
Обчислює векторний добуток векторів (p2 - p1) і (p3 - p1).

double Triangle::crossProduct(const Point& p1, const Point& p2, const Point& p3) const {  
 return (p2.x - p1.x) \* (p3.y - p1.y) - (p2.y - p1.y) \* (p3.x - p1.x);  
}

**Triangle::isDegenerate()**Перевіряє, чи є трикутник виродженим (усі точки на одній прямій).  
Якщо векторний добуток (b - a) × (c - a) ≈ 0, то трикутник вироджений.

bool Triangle::isDegenerate() const {  
 return std::fabs(crossProduct(a, b, c)) < 1e-9;  
}

**Triangle::containsByCrossProduct(p)**  
Перевіряє, чи належить точка трикутнику, використовуючи векторні добутки.

1. Обчислює векторні добутки для кожної сторони:
   * cp1 = (b - a) × (p - a)
   * cp2 = (c - b) × (p - b)
   * cp3 = (a - c) × (p - c)
2. Якщо всі добутки одного знака (> 0 або < 0), точка всередині.
3. Якщо хоча б один ≈ 0, перевіряє isOnEdge(p).
4. bool Triangle::containsByCrossProduct(const Point& p) const {  
    double cp1 = crossProduct(a, b, p);  
    double cp2 = crossProduct(b, c, p);  
    double cp3 = crossProduct(c, a, p);  
     
    if (std::fabs(cp1) < 1e-9 || std::fabs(cp2) < 1e-9 || std::fabs(cp3) < 1e-9) {  
    return isOnEdge(p);  
    }  
     
    bool sameSign = (cp1 > 0 && cp2 > 0 && cp3 > 0) || (cp1 < 0 && cp2 < 0 && cp3 < 0);  
    return sameSign;  
   }

**Triangle::isOnEdge(p)**  
Перевіряє, чи лежить точка на ребрі трикутника.

1. Перевіряє, чи точка лежить на прямій між двома вершинами за допомогою crossProduct ≈ 0.
2. Перевіряє, чи координати точки знаходяться між вершинами ребра.
3. bool Triangle::isOnEdge(const Point& p) const {  
    return std::fabs(crossProduct(a, b, p)) < 1e-9 &&  
    ((p.x - a.x) \* (p.x - b.x) <= 0) && ((p.y - a.y) \* (p.y - b.y) <= 0) ||  
    std::fabs(crossProduct(b, c, p)) < 1e-9 &&  
    ((p.x - b.x) \* (p.x - c.x) <= 0) && ((p.y - b.y) \* (p.y - c.y) <= 0) ||  
    std::fabs(crossProduct(c, a, p)) < 1e-9 &&  
    ((p.x - c.x) \* (p.x - a.x) <= 0) && ((p.y - c.y) \* (p.y - a.y) <= 0);  
   }

3. У main.cpp реалізовано меню з можливість вводити точки та виводити результат на екран.

**Main.cpp**

#include "triangle.h"  
#include <iostream>  
#include <vector>  
  
int main() {  
 Triangle triangle;  
 std::cout << "Введіть координати трикутника (x1 y1 x2 y2 x3 y3): ";  
 std::cin >> triangle.a.x >> triangle.a.y  
 >> triangle.b.x >> triangle.b.y  
 >> triangle.c.x >> triangle.c.y;  
  
 if (triangle.isDegenerate()) {  
 std::cout << "Трикутник є виродженим (площа = 0)." << std::endl;  
 return 1;  
 }  
  
 int numPoints;  
 std::cout << "Введіть кількість точок для перевірки: ";  
 std::cin >> numPoints;  
  
 std::vector<Point> points(numPoints);  
 for (int i = 0; i < numPoints; ++i) {  
 std::cout << "Введіть координати точки " << i + 1 << " (x y): ";  
 std::cin >> points[i].x >> points[i].y;  
 }  
  
 std::cout << "\nРезультати перевірки:\n";  
 for (const auto& p : points) {  
 bool onEdge = triangle.isOnEdge(p);  
 bool insideArea = triangle.containsByArea(p);  
 bool insideCross = triangle.containsByCrossProduct(p);  
  
 std::cout << "Точка (" << p.x << ", " << p.y << "):\n";  
 std::cout << " Метод за площею Герона: " << (insideArea ? "всередині" : "зовні") << "\n";  
 std::cout << " Метод векторного добутку: " << (insideCross ? "всередині" : "зовні") << "\n";  
  
 if (onEdge) {  
 std::cout << " Лежить на межі трикутника\n";  
 }  
 std::cout << std::endl;  
 }  
  
 return 0;  
}

**ВИСНОВОК**

Під час виконання лабораторної роботи попрактикувався у розв’язанні алгоритмів за допомогою функціонального стилі.