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**:

**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 клас Triangle з полями a,b,c – вершини трикутника, та методи роботи з ними:

crossProduct() – обчислення векторного добутку.

isDegenerate() – перевірка трикутника на виродженість.

containsPoint() – перевірка чи точка належить трикутнику.

isOnEdge() – перевірка чи точка ледить на межі трикутника.

**Triangle.h**

#ifndef TRIANGLE\_H  
#define TRIANGLE\_H  
  
struct Point {  
 double x, y;  
};  
  
class Triangle {  
public:  
 Triangle(const Point& a, const Point& b, const Point& c);  
 bool isDegenerate() const;  
 bool containsPoint(const Point& p) const;  
 bool isOnEdge(const Point& p) const;  
 double crossProduct(const Point& p1, const Point& p2, const Point& p3) const;  
  
private:  
 Point a, b, c  
};  
  
#endif // TRIANGLE\_H

2.У файлі Triangle.cpp виконуємо реалізацію методів.

crossProduct() – метод отримує 3 точки трикутника.

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);  
}

isDegenerate() – обчислює векторний добуток 3 точок трикутника і порівнює результат чи він наближається до 0, якщо так це означає що всі точки лежать на одній прямій і площа трикутника = 0 (трикутник вироджений)

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

containsPoint() – 1.Обчилюємо векторний добуток кожної пари ребер та точки ‘p’, якщо хоча б один з добутків =0 значить точка лежить на межі і ми перевіряємо через метод isOnEdge(). 2.Якщо всі три результати мають однаковий знак, точка знаходиться всередині трикутника.

bool Triangle::containsPoint(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;  
}

isOnEdge() – 1.Перевіряє, чи точка p знаходиться на одній з трьох сторін трикутника (AB, BC, CA). 2.Якщо crossProduct між p і якимось ребром ≈ 0, це означає, що точка лежить на прямій, що містить ребро. 3.Додаткова перевірка: а.) (p.x - a.x) \* (p.x - b.x) <= 0 перевіряє, що p знаходиться між a та b (не виходить за межі відрізка). б.)Аналогічно для y.

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**

int main() {  
 Point a, b, c;  
 std::cout << "Введіть координати трикутника (x1 y1 x2 y2 x3 y3): ";  
 std::cin >> a.x >> a.y >> b.x >> b.y >> c.x >> c.y;  
  
 Triangle triangle(a, b, c);  
  
 if (triangle.isDegenerate()) {  
 std::cout << "Трикутник є виродженим (площа = 0)." << std::endl;  
 return 1;  
 }  
  
 int numPoints;  
 std::cout << "Введіть кількість точок: ";  
 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;  
 }  
  
 for (const auto& p : points) {  
 if (triangle.containsPoint(p)) {  
 if (triangle.isOnEdge(p))  
 std::cout << "Точка (" << p.x << ", " << p.y << ") лежить на межі трикутника." << std::endl;  
 else  
 std::cout << "Точка (" << p.x << ", " << p.y << ") всередині трикутника." << std::endl;  
 } else {  
 std::cout << "Точка (" << p.x << ", " << p.y << ") поза трикутником." << std::endl;  
 }  
 }  
  
 return 0;  
}

**ВИСНОВОК**

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