МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ УПРАВЛІННЯ

ЗВІТ

З ЛАБОРАТОРНОЇ РОБОТИ №7

ЗА ТЕМОЮ “ГЕОМЕТРИЧНІ АЛГОРИТМИ”

Виконав студент

групи КН-221д

Кукуєв Руслан Олександрович

Перевірила

Солонська С.В.

Харків 2022

ЛАБОРАТОРНА РОБОТА №7

Тема: ГЕОМЕТРИЧНІ АЛГОРИТМИ.

Мета роботи: познайомитися з геометричними алгоритмами.

**Завдання:**

Розробити програму, яка читає з клавіатури число N (1 < N < 256) та N пар дійсних чисел — координати точок на площині. Програма виконує один за алгоритмів згідно варіанту. 7.3 Варіанти завдань 1 Точки належать ламаній. Потрібно для кожної нової ланки вказати, направо чи наліво здійснено поворот. 2 Точки належать ламаній. Потрібно для кожної нової ланки вказати, чи перетинає вона будь-яку з попередніх. 3 Точки є координатами багатокутника в порядку обходу. Вивести площину багатокутника та повідомити, по чи проти годинникової стрілки здійснено обхід. 4 Побудувати опуклу оболонку наданих точок алгоритмом Грехема. 5 Побудувати опуклу оболонку наданих точок алгоритмом Джарвіса.

**Варіант №2**

**Lab\_7.cpp**

#include <iostream>

#include "Cross.cpp"

void FindCrossAllSegments(int segmentIndex, Segment \*segments);

int main()

{

srand(time(NULL));

int N;

std::cout << "Enter the number of pairs of points: ";

std::cin >> N;

if (1 >= N || N > 256)

{

std::cout << "Incorrect value: " << N << "\n";

system("pause");

return 0;

}

Point \*points = new Point[N];

Segment \*segments = new Segment[N - 1];

int segmentCounter = 0;

for (int i = 0; i < N; i++)

{

points[i].x = rand() % 100;

points[i].y = rand() % 100;

if (i > 0)

{

segments[segmentCounter].first = points[i - 1];

segments[segmentCounter].second = points[i];

FindCrossAllSegments(segmentCounter, segments);

segmentCounter++;

}

}

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

{

std::cout << "Segment: " << (i + 1) << "\n";

std::cout << (segments[i].haveCross ? "Have cross!" : "Not cross!") << "\n";

}

delete[] points;

delete[] segments;

system("pause");

}

void FindCrossAllSegments(int segmentIndex, Segment \*segments)

{

if (segmentIndex < 0)

{

return;

}

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

{

segments[segmentIndex].haveCross = Cross(segments[i], segments[segmentIndex]);

}

}

**Cross.cpp**

#include <iostream>

struct Point

{

int x, y;

};

struct Segment

{

Point first, second;

bool haveCross = false;

};

bool Cross(Segment s1, Segment s2)

{

double Ua, Ub, numerator\_a, numerator\_b, denominator;

denominator = (s2.second.y - s2.first.y) \* (s1.first.x - s1.second.x) - (s2.second.x - s2.first.x) \* (s1.first.y - s1.second.y);

if (denominator == 0)

{

if ((s1.first.x \* s1.second.y - s1.second.x \* s1.first.y) \* (s2.second.x - s2.first.x) - (s2.first.x \* s2.second.y - s2.first.y \* s2.first.y) \* (s1.second.x - s1.first.x) == 0 && (s1.first.x \* s1.second.y - s1.second.x \* s1.first.y) \* (s2.second.y - s2.first.y) - (s2.first.x \* s2.second.y - s2.first.y \* s2.first.y) \* (s1.second.y - s1.first.y) == 0)

return true;

else

return false;

}

else

{

numerator\_a = (s2.second.x - s1.second.x) \* (s2.second.y - s2.first.y) - (s2.second.x - s2.first.x) \* (s2.second.y - s2.first.y);

numerator\_b = (s1.first.x - s1.second.x) \* (s2.second.y - s2.first.y) - (s2.second.x - s1.second.x) \* (s1.first.y - s1.second.y);

Ua = numerator\_a / denominator;

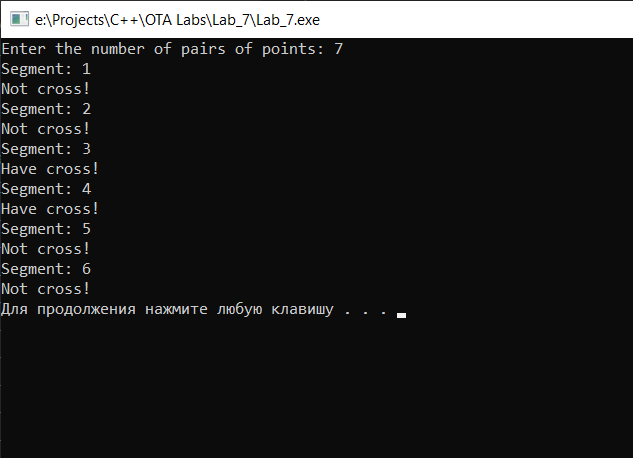
Ub = numerator\_b / denominator;

return (Ua >= 0 && Ua <= 1 && Ub >= 0 && Ub <= 1 ? true : false);

}

}

**Демонстрація роботи:**



**Висновок:** виконавши лабораторну роботу №7, я познайомився з геометричними алгоритмами.