Звіт по лабораторній роботі №6

З предмету Комп`ютерна графіка

Студента групи ІПС-31

Олійника Микита

**Тема**: побудова опуклої оболонки методом Джарвіса

**Алгоритм**:

1. Шукаємо найлівішу точку для даного набору, додаємо її до опуклої оболонки
2. На кожній наступній ітерації алгоритму шукаємо точку таку що утворює найбільший полярний кут з попереднєю та додаємо її до опуклої оболонки
3. Зупиняємо пошук коли знайдена точка збігається з першою

**Код**:

#include <iostream>  
#include <fstream>  
#include <vector>  
#include <algorithm>  
  
using namespace std;  
  
struct Point  
{  
 int x, y;  
 Point(int a, int b) : x(a), y(b) {}  
 bool operator<(Point p)  
 {  
 return x < p.x || (x == p.x && y < p.y); // для пошуку найлівішої точки  
 }  
};  
  
int area(Point a, Point b, Point c)  
{  
 return (b.x - a.x)\*(c.y - a.y) - (b.y - a.y)\*(c.x - a.x); // орієнтована площа трикутника  
}  
  
bool left(Point a, Point b, Point c)  
{  
 return area(a, b, c) > 0; // предикат «проти годинникової стрілки»  
}  
  
bool collinear(Point a, Point b, Point c)  
{  
 return area(a, b, c) == 0; // перевірка трьох точок на коллінеарність   
}  
  
bool between(Point a, Point b, Point c)  
{  
 return abs(c.x - a.x) <= abs(b.x - a.x) && abs(c.y - a.y) <= abs(b.y - a.y); // чи лежить точка С між точками А та В  
}  
  
int main()  
{  
 ifstream fin("../input.txt");  
  
 int n, start = 0;  
 fin >> n;  
 vector < Point > in;  
  
 for (int i = 0; i < n; ++i)  
 {  
 int x, y;  
 fin >> x >> y;  
 Point p(x, y);  
 in.emplace\_back(p);  
 if (p < in[start])  
 start = i; // шукаємо найлівішу точку  
 }  
  
 vector < bool > selected(in.size(), false);  
 selected[start] = true;  
  
 vector < Point > convex\_hull;  
 convex\_hull.push\_back(in[start]); // додаємо найлівішу точку до опуклої оболонки  
  
 while (true)  
 {  
 int best = -1;  
 Point previous = convex\_hull[convex\_hull.size() - 1]; // попередня знайдена точка опуклої оболонки  
 for (int i = 0; i < in.size(); ++i)  
 {  
 if (!selected[i])  
 {  
 if (best == -1)  
 best = i;  
 else if (left(previous, in[best], in[i])) // шукаємо точку що утворює найбільший полярний кут з попереднью  
 best = i;  
 else if (collinear(previous, in[best], in[i]) && between(previous, in[best], in[i])) // враховуємо точки що лежать на одній прямій  
 best = i;  
 }  
 }  
 if (best == -1 || left(previous, in[best], in[start])) // якщо точок не залишилось або якщо перша точка є кращим кандидатом за знайдену, зупиняємось  
 break;  
 else  
 {  
 selected[best] = true;  
 convex\_hull.push\_back(in[best]);  
 }  
 }  
  
 cout << convex\_hull.size() << endl;  
 for (Point p: convex\_hull)  
 cout << p.x << ' ' << p.y << endl;  
 return 0;  
}