Звіт по лабораторній роботі №5

З предмету Комп`ютерна графіка

Студента групи ІПС-31

Олійника Микита

**Тема**: побудова опуклої оболонки методом QuickHull (Швидкобол)

**Алгоритм**:

1. Знайдемо найлівішу точку для даного набору точок
2. Від вертикальної відрізку, що проходить через найлівішу точку, та всього набору точок, запустимо рекурсивну процедуру побудови. У ній:
   1. Знайдемо найвіддаленішу точку від поточного відрізка, додамо її до відповіді
   2. Розіб`ємо поточну множину точок на три підмножини трикутником, що отримується поточним відрізком та знайденою точкой
   3. Підмножина точок у середені трикутника відкидається так як вони не будуть входити до опуклої оболонки
   4. Від двох підмножин за межами трикутника та відповідним їм сторін трикутника знов запустимо дві рекурсивні процедури

**Код**:

#include <iostream>  
#include <fstream>  
#include <vector>  
  
using namespace std;  
  
struct Point  
{  
 int x, y;  
 Point(int a, int b) : x(a), y(b) {}  
 bool operator<(const Point &p) const  
 {  
 return x < p.x || (x == p.x && y < p.y); // для пошуку найлівішої точки  
 }  
};  
  
int area(const Point &a, const Point &b, const Point &c) // орієнтована точка прямокутника  
{  
 return (b.x - a.x)\*(c.y - a.y) - (b.y - a.y)\*(c.x - a.x);  
}  
  
void f(Point l, Point r, const vector < Point > &candidates, vector < Point > &convex\_hull)  
{  
 int best = 0;  
 vector < Point > actual\_candidates; // множина точок що не були усередені трикутника з попереднього етапу  
 for (Point candidate: candidates)  
 if (area(l, r, candidate) >= 0)  
 {  
 actual\_candidates.push\_back(candidate);  
 if (area(l, r, candidate) >= area(l, r, actual\_candidates[best])) // шукаємо найвіддалену точку як таку що утворює з поточним відрізком найбільшу площу  
 best = actual\_candidates.size() - 1;  
 }  
 if (!actual\_candidates.empty())  
 {  
 Point chosen = actual\_candidates[best];  
 actual\_candidates.erase(actual\_candidates.begin() + best);  
 f(l, chosen, actual\_candidates, convex\_hull);  
 convex\_hull.push\_back(chosen); // додаємо найвіддаленішу точку до відповіді  
 f(chosen, r, actual\_candidates, convex\_hull);  
 }  
}  
  
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  
 }  
 Point l = in[start];  
 in.erase(in.begin() + start);  
 vector < Point > convex\_hull = {l};  
 Point r(l.x, l.y - 1); // друга точка що задає вертикальний відрізок що проходить через найлівішу точку  
  
 f(l, r, in, convex\_hull); // запускаємо рекурсивну процедуру  
  
 cout << convex\_hull.size() << endl;  
 for (Point p: convex\_hull)  
 cout << p.x << ' ' << p.y << endl;  
 return 0;  
}