МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

**Лабораторная работа №4 (вариант 4)**

по дисциплине: «Геометрическое моделирование».

Выполнил:

студент 3 курса, гр. ИВТАПбд-31

Кондратьев Павел Сергеевич.

Проверил:

Войт Николай Николаевич

г. Ульяновск, 2018

**Задание:** Алгоритм заворачивания подарков (Джарвиса)

Этот алгоритм, предложен Джарвисом в 1973. Он также известен под названием «метод заворачивания подарка», так как этот алгоритм обрабатывает точки выпуклой оболочки одну за другой, как если бы мы оборачивали множество точек листом бумаги. Джарвис обратил внимание на то, что многоугольник, которым является выпуклая оболочка, с одинаковым успехом можно задать упорядоченным множеством, как его ребер, так и его вершин. Если задано множество точек, то довольно трудно быстро определить, является или нет некоторая точка крайней. Однако если даны две точки, то непосредственно можно проверить, является или нет соединяющий их отрезок ребром выпуклой оболочки.

**Выпуклая оболочка**

* Любая экстремальная точка (например, самая левая из самых нижних) всегда принадлежит выпуклой оболочке. Сортируем точки по координатам, выбираем экстремальную точку O.
* Сортируем остальные почки по полярному углу от O. Точка A должна идти раньше точки B, если поворот от вектора OA к вектору OB происходит против часовой стрелки (косое произведение положительно). Если OA и OB коллинеарны, то раньше идёт точка, ближайшая к O.
* Добавляем точку O в оболочку.
* Рассматриваем остальные точки в порядке сортировки по углу. Пока две последние точки оболочки вместе с новой точкой лежат на одной прямой или дают поворот по часовой стрелке (косое произведение неотрицательно), выкидываем последнюю точку из оболочки. Добавляем следующую точку в оболочку.

Трудоемкость этого алгоритма составляет O(hn) или O() в худшем случае, когда в выпуклую оболочку попадут все точки.

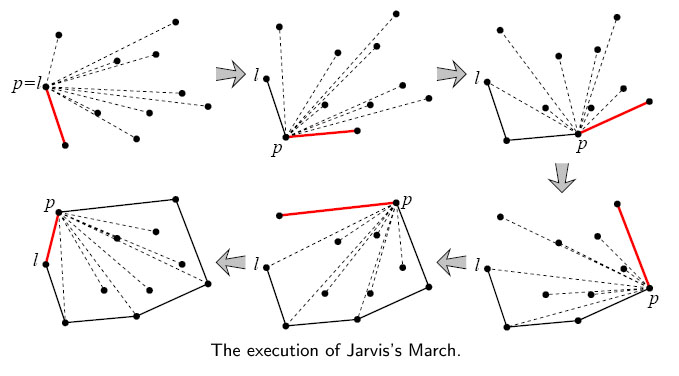


Рис. 1. Построение выпуклой оболочки методом Джарвиса.

**Примеры работы программы:**

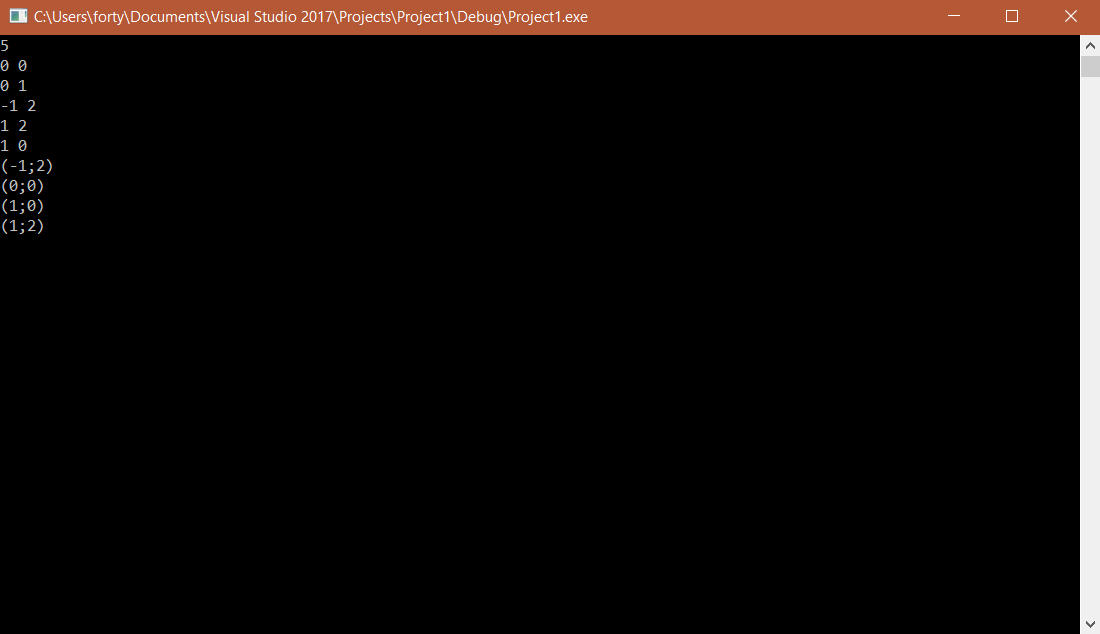


Рис. 2. Пример 1

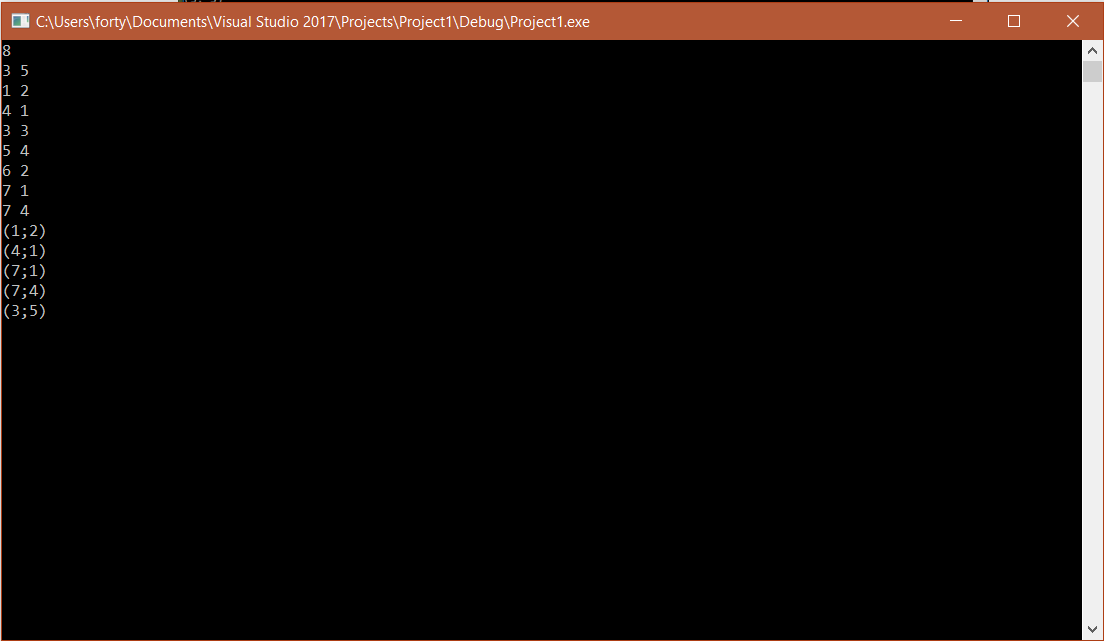


Рис. 3. Пример 2

**Код:**

#include <cstdlib>

#include <cstdio>

#include <algorithm>

#include <vector>

#include <cmath>

using namespace std;

#define sqr(x) ((x) \* (x))

const double inf = 1e100;

struct point {

double x, y;

void read() {

scanf("%lf %lf", &x, &y);

}

bool operator<(const point& r) const {

if (x < r.x) {

return 1;

}

if (x > r.x) {

return 0;

}

return y < r.y;

}

point operator-(point& r) {

point res = { x - r.x, y - r.y };

return res;

}

double slope() {

if (x == 0.0 && y == 0.0) {

return -inf;

}

if (x == 0.0) {

return inf;

}

return y / x;

}

double operator\*(const point& r) {

return x \* r.y - y \* r.x;

}

double dist\_to(point& r) {

return sqrt(sqr(x - r.x) + sqr(y - r.y));

}

};

point O; // left-most lower point

bool BY\_SLOPE(point l, point r) {

double ls = (l - O).slope(), rs = (r - O).slope();

if (ls < rs) {

return 1;

}

if (ls > rs) {

return 0;

}

return l.dist\_to(O) < r.dist\_to(O);

}

// pre: N >= 0, [p, p + N) - points

vector<point> convex\_hull(point \*p, int N) {

//база, если всего 2 точки -> ответ

if (N <= 2) {

return vector<point>(p, p + N);

}

sort(p, p + N);

//стартовая точка

O = p[0];

sort(p + 1, p + N, BY\_SLOPE);

vector<point> hull;

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

if (i < 3) {

hull.push\_back(p[i]);

}

else {

int sz = hull.size();

while (sz >= 2 && (p[i] - hull[sz - 2])\*(hull[sz - 1] - hull[sz - 2]) >= 0) {

hull.pop\_back(), sz--;

}

hull.push\_back(p[i]);

}

}

return hull;

}// post: convex hull in hull, given in ccw order

vector<point> v;

int n;

point P[21100];

int main() {

//колличество пар точек

scanf("%d", &n);

//считывание пар точек

for (int i = 0; i < n; i++) {

P[i].read();

}

//ищем выпуклую оболочку

v = convex\_hull(P, n);

//вывод пар мбо

for (int i = 0; i < v.size(); i++) {

printf("(%.lf;%.lf)\n", v[i].x, v[i].y);

}

return 0;

}