Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

им. В.И. Ульянова (Ленина)»

кафедра систем автоматизированного проектирования

**Курсовая работа**

**по дисциплине «Программирование»**

**Тема: «Использование массивов для решения геометрических задач»**

Выполнил : Игнатьев Валерий Алексеевич

Группа № 1309

Преподаватель: Калмычков Виталий Анатольевич

Санкт-Петербург

2021

# Задание на курсовую работу

Студент Бобков Владислав Дмитриевич

Группа 1309

Тема работы: Использование массивов для решения геометрических задач

Исходные данные:

Даны N точек на плоскости. Найти среди них точки, являющиеся вершинами двойной стрелки, содержащего наибольшее количество заданных точек.

Содержание пояснительной записки:

Содержание, введение, заключение, список использованных источников.

Предполагаемый объем пояснительной записки:

Не менее 0 страниц.

Дата выдачи задания: 16.11.2021

Дата сдачи реферата: .12.2021

Дата защиты реферата: .12.2021

Студент \_\_\_\_\_\_\_\_\_\_\_\_ Игнатьев В. А.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ Калмычков В.А

# Аннотация

Способ выполнения задачи состоит в следующем: я запускаю пять вложенных форов и перебираю половину стрелки. После чего проверяю правда ли это половина стрелки и через центр симетрии ищу вторую половину и проверяю есть ли она. После чего проверяю находил ли я уже эту стрелку ведь у каждой стрелки 4 половины которые можно найти. После чего проверял сколько точек лежит внутри этой стрелки и записывал ее в массив в котором хранятся все стрелки.

# Summary

The way to accomplish the task is as follows: I run five nested for and iterate over half of the arrow. Then I check if it is true half of the arrow and through the center of symmetry I look for the second half and check if it is there. Then I check if I have already found this arrow, because each arrow has 4 halves that can be found. Then I checked how many points lie inside this arrow.

# 

# Содержание

Оглавление

Введение

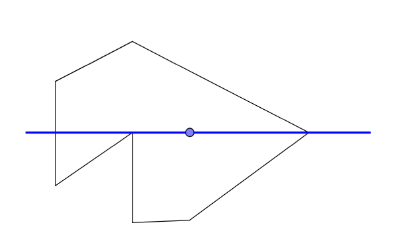
1. Математическая постановка задачи
2. Особенности реализации на компьютере
3. Обработка исходного файла
4. Выполнение задачи
5. Контрольные примеры
6. Описание входных и выходных файлов
7. Текст программы.

# Введение

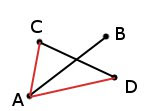
Идет перебор всех возможных полученных фигур, но нам интересны только двойные стрелки. По двум координатам строится точка. После построения фигуры идет перебор заданных точек. Методом трассировки лучей из заданной точки определяется принадлежность точки фигуре. Затем, после перебора всех возможных фигур выбираются стрелки, содержащие наибольшее количество заданных точек.

# Математическая постановка задачи

Метод, согласно которому будет определяться принадлежность точки фигуре заключается в следующем: из точки выпускается луч в произвольном направлении. Если луч пересечет контур фигуры нечетное количество раз, то точка лежит внутри фигуры, четное — снаружи.



Рассмотрим два пересекающихся отрезка AB и CD. Мы можем умножить попарно - векторно вектор разделяющего отрезка на вектора, направленные от начала разделяющего отрезка к обеим точкам проверяемого отрезка. Тогда произведение это не будет положительным, если отрезки пересекаются.



Затем рассмотрим относительно точки B точно так же.

Тогда векторное произведение AC на AD и BC на BD будет неположительным.

А пересечение сопровождается тем, что для первого разные концы второго отрезка лежат по разные стороны от первого и наоборот. И тогда произведение двух векторов, проходящих через один конец первого, первый и второй конец второго соответственно, не может быть положительным. Если это условие соблюдено, отрезки пересекаются.

Так и проходит перебор всех заданных точек.

# Особенности реализации на компьютере

Ввиду того, что программа содержит в себе 6 вложенных циклов, то при большом количестве заданных точек могут возникнуть проблемы с тем, что программа будет компилироваться достаточно долго, поэтому рекомендуется задавать количество точек, меньшее 40, так как при этом количестве заданных точек программа компилировалась больше 10 минут.

# Обработка исходного файла

Я считываю по 2 числа из каждой строки до конца файла если в строке меньше 2 числе я ее пропускаю если больше беру первые 2 и так до конца. Все это выполняется в функции Read\_one\_line().

Код функций:

point read\_one\_line() {

/// -999999 = end of line

point a;

a.x = empty\_error;

a.y = empty\_error;

string s;

getline(fin, s);

int j = 0;

for (int k = 0; k < 2; k++) {

int znak = 1;

double res = 0;

int x = 0;

double power = 10;

int d = 0;

for (int i = j; i < s.size(); i++, j++) {

if (s[i] == '-') {

znak = -1;

x = 1;

}

else if (s[i] == '.') {

d = 1;

x = 1;

}

else if (s[i] <= '9' and s[i] >= '0') {

x = 1;

if (d) {

res += double(s[i] - '0') / power;

power \*= 10;

}

else {

res = res \* 10 + s[i] - '0';

}

}

else if (x) {

break;

}

}

if (x) {

if (k == 0) {

a.x = res \* znak;

}

else a.y = res \* znak;

}

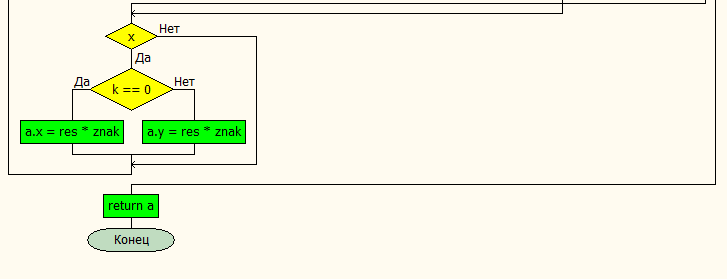
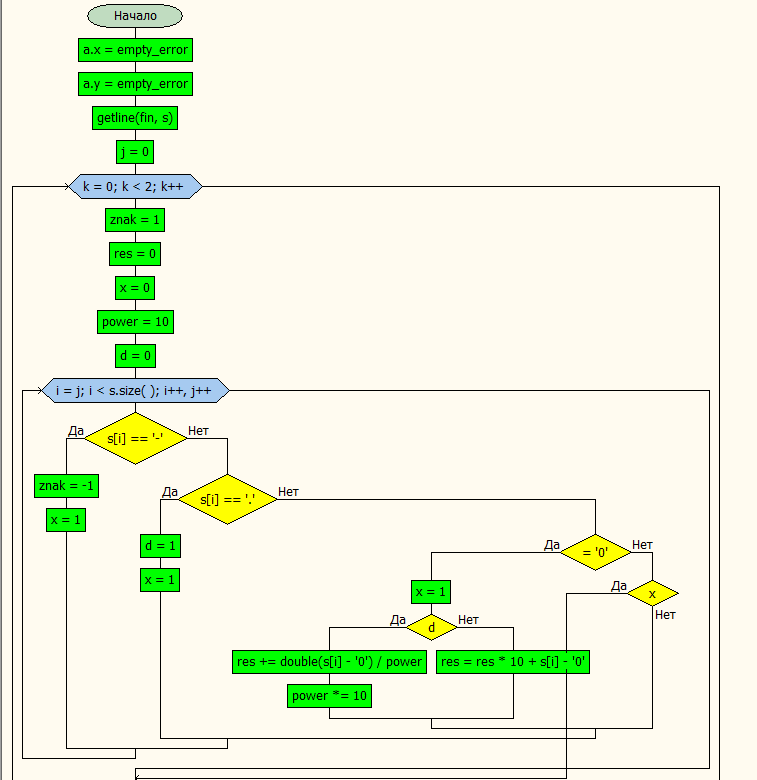
}

return a;

}

Блок-схемы

Read\_one\_line



|  |  |  |
| --- | --- | --- |
| Название функции | Данные, задействованные в функции | Предназначение функции |
| Read\_one\_line() | ifstream fin("input.txt"); | считывание 2 чисел из 1 строки |

# Выполнение задачи

Для начала следует ввести из нового файла массив, который содержит n количество координат точек на плоскости. Ввод происходит построчно, исключая лишние символы.

Функция Read\_one\_line считывает строку и обрабатывает ее таким образом беря только нужную информацию.

Далее в протокольный файл выводятся координаты, подлежащие обработке.

Далее следует сам процесс выполнения задачи. Для начала в пяти вложенных циклах определяем вершины половины двойной стрелки, исключая случаи, когда вершины имеют одинаковые координаты в функции check\_unique. После чего проверяется действительно ли это половина двойной стрелки в first\_half . После ищется центр симметрии (Find\_centre).

И на основе центра симметрии проверяется существование второй половины двойной стрелки в функции second\_exist. И наконец проверяется находили ли мы такую стрелку до этого в функции fwas.

Далее идет вывод в протокольный файл всех различных двойных стрелок с указанным количеством принадлежащих точек и сами точки.

После идет вывод в консоль и в протокольный файл всех двойных стрелок содержащие наибольшее количество точек.

Это все происходит в функции main.

Код функции:

main

int main() {

while (!fin.eof()) {

point a = read\_one\_line();

if(a.x != empty\_error and a.y != empty\_error){

p[n].x = a.x;

p[n++].y = a.y;

}

}

fout << "There are(is) " << n << " points.";

for (int i = 0; i < n; i++) fout << p[i].x << " " << p[i].y << endl;

cout << endl;

int nmax = -1;

for (int i1 = 0; i1 < n; i1++) {

for (int i2 = 0; i2 < n; i2++) {

for (int i3 = 0; i3 < n; i3++) {

for (int i4 = 0; i4 < n; i4++) {

for (int i5 = 0; i5 < n; i5++) {

point first\_cur[5] = { p[i1], p[i2], p[i3], p[i4], p[i5] };

if (!(check\_unique(first\_cur))) continue;

if (!first\_half(first\_cur)) continue;

point cen = find\_centre(p[i1], p[i2], p[i3], p[i5]);

point second\_cur[5];

if (!(second\_exist(first\_cur, cen, second\_cur))) continue;

point ans2[10];

ans2[0] = p[i1];

ans2[1] = p[i2];

ans2[2] = p[i3];

ans2[3] = p[i4];

ans2[4] = p[i5];

ans2[5] = second\_cur[0];

ans2[6] = second\_cur[1];

ans2[7] = second\_cur[2];

ans2[8] = second\_cur[3];

ans2[9] = second\_cur[4];

if (fwas(ans2)) continue;

}

}

}

}

}

if (!nwas) {

fout << "No one figure exist\n";

}

else {

fout << "\nTotal figure(s) is " << nwas << endl;

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

fout << "Another one figure:\n";

for (int j = 0; j < 10; j++) {

fout << was[i][j].x << " " << was[i][j].y << endl;

}

fout << "Points inside it:\n";

int cur = check\_figure(was[i]);

if (!cur) fout << "No one point in it!\n";

else cout << endl << cur << " point(s).\n";

insidecur[i] = cur;

if (cur > nmax) {

nmax = cur;

}

}

fout << "The best figure(s): \n";

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

if (insidecur[i] != nmax) continue;

fout << "Another one figure with " << insidecur[i] << " point(s) inside is:\n";

for (int j = 0; j < 10; j++) {

fout << was[i][j].x << " " << was[i][j].y << endl;

}

fout << "Points inside it:\n";

int cur = check\_figure(was[i]);

if (!cur) fout << "No one point in it!\n";

}

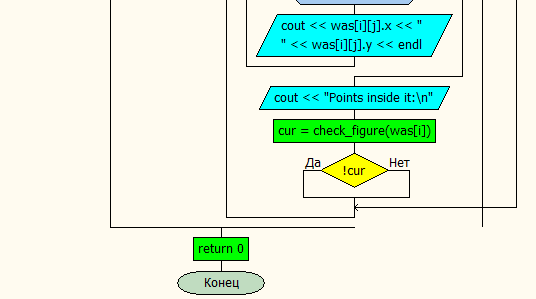
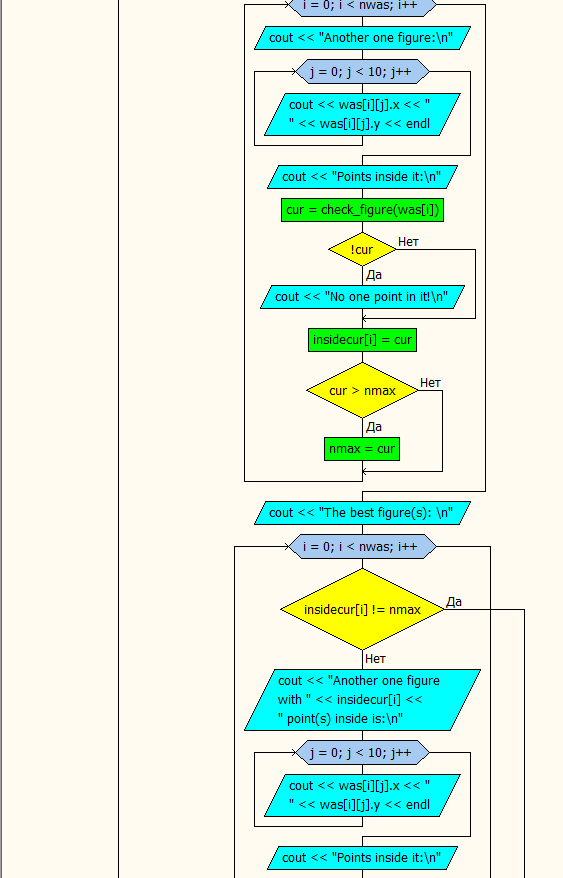
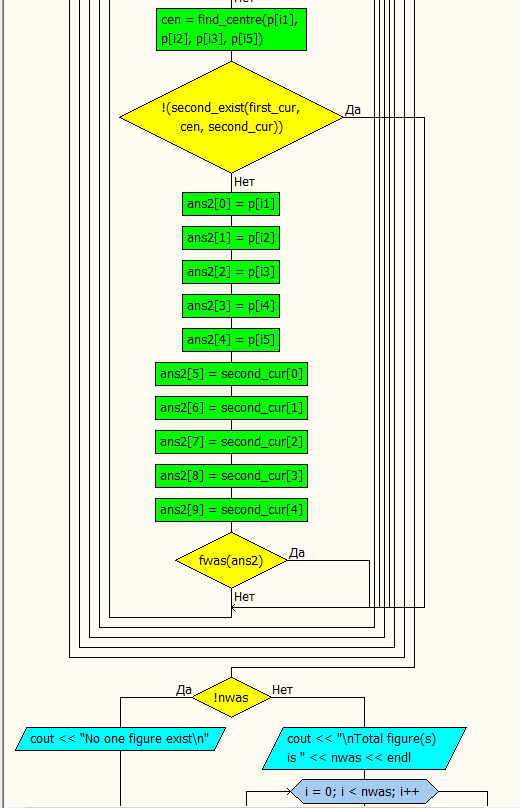
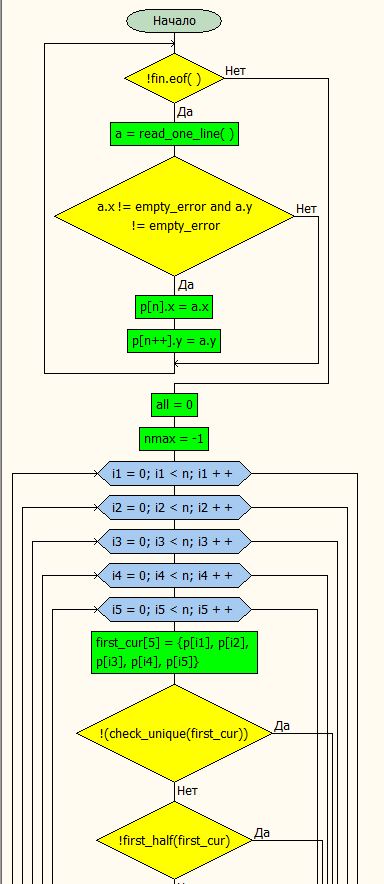
}

return 0;

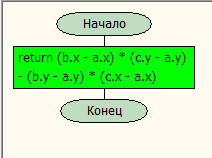
}

Блок-схемы:

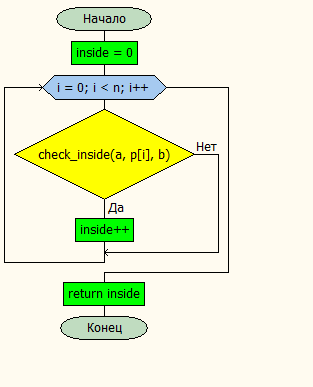
main



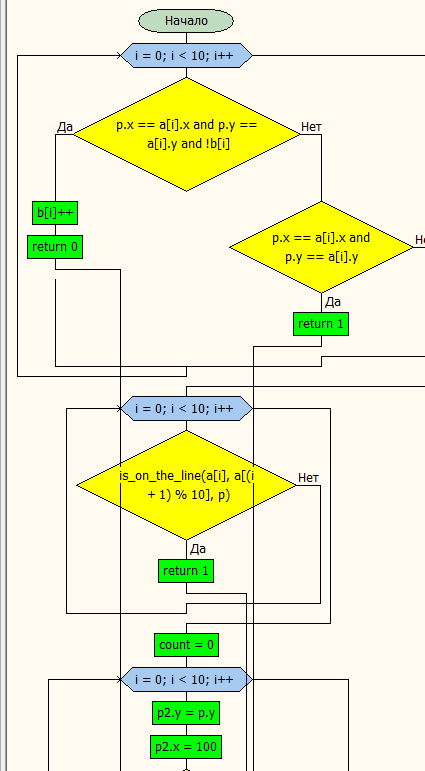
Area

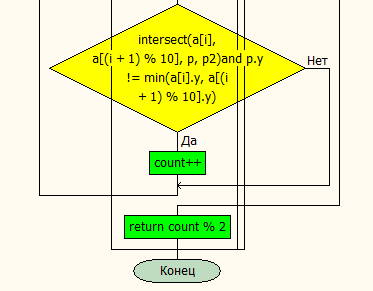


Check\_figure

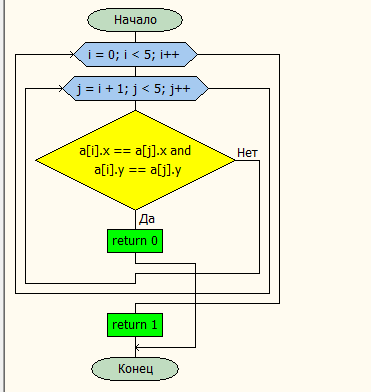


Check\_inside

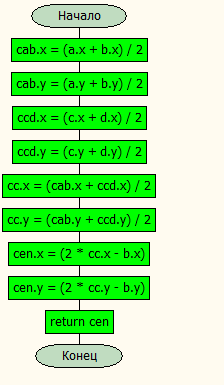




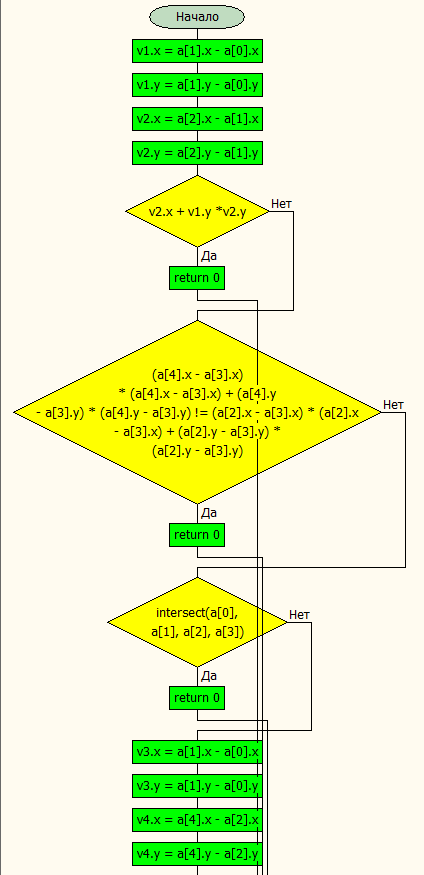
Check\_unique

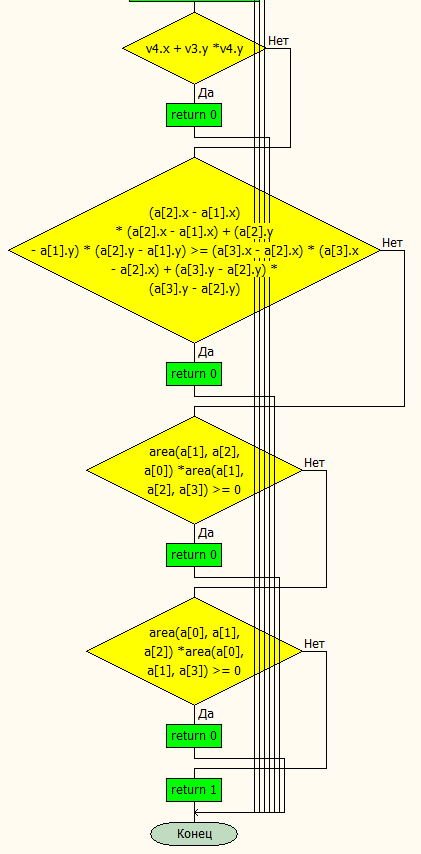


Find\_centre

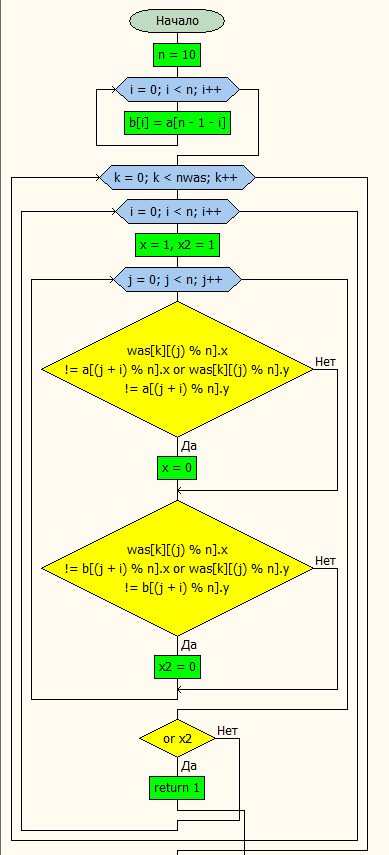


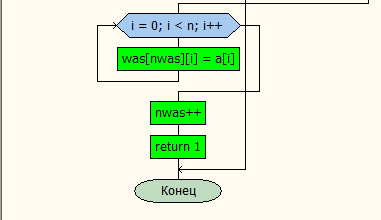
First\_half



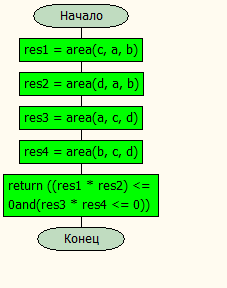


Fwas

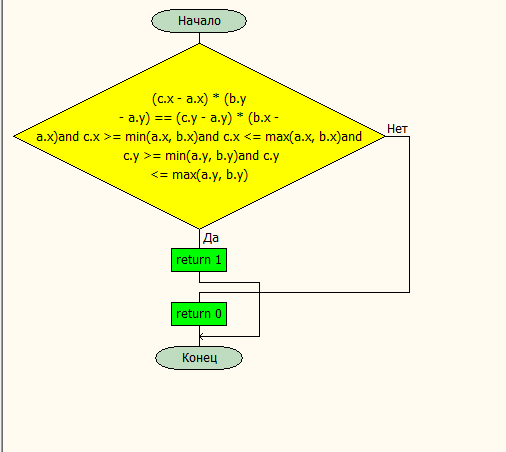




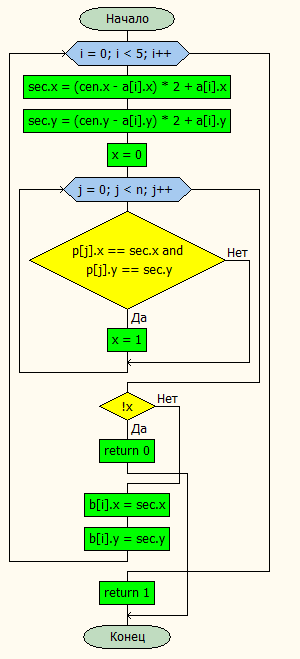
Intersect



Is\_on\_the\_line



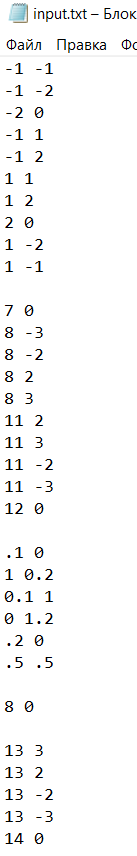
Second\_exist



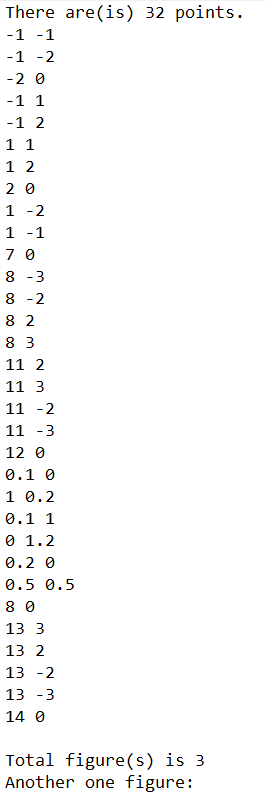
|  |  |  |
| --- | --- | --- |
| Название функции | Данные, задействованные в функции | Предназначение функции |
| main | int A[50][2]; int n; fstream f, protokol, f1,rez; | Главная функция, для перебора точек , ввода и вывода, вызывает функции |
| area | Point a,b,c | Ориентированная площадь треугольника |
| Check\_figure | point a[10] | Нахождение количества и координат точек которые принадлежат фигуре |
| Check\_inside | point a[10], point p, int b[10] | Проверка на принадлежность точки фигуре |
| Find\_centre | point a, point b, point c, point d | Нахождение центра симметрии |
| First\_half | point a[5] | Проверка на правильность первой половины |
| Fwas | point a[10] | Проверка была ли такая фигура уже найдена |
| Intersect | point a, point b, point c, point d | Проверка на пересечение двух отрезков |
| Is\_on\_the\_line | point a, point b, point c | Проверка принадлежит ли точка отрезку |
| Second\_exist | point a[5], point cen, point b[5] | Проверка существуют ли точки которые образуют вторую половину |
| Check\_unique | point a[5] | Проверка на то что все точки первой половины уникальны |

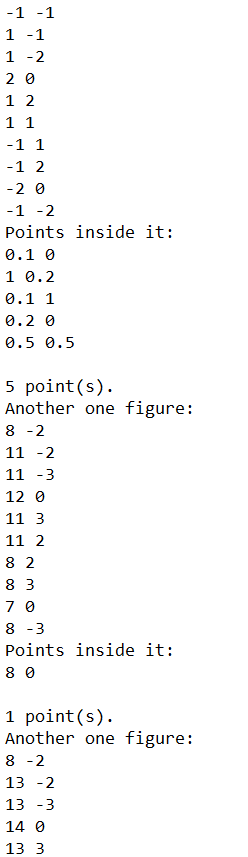
# Контрольные примеры

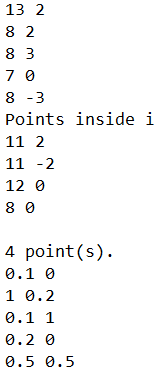
1 Пример



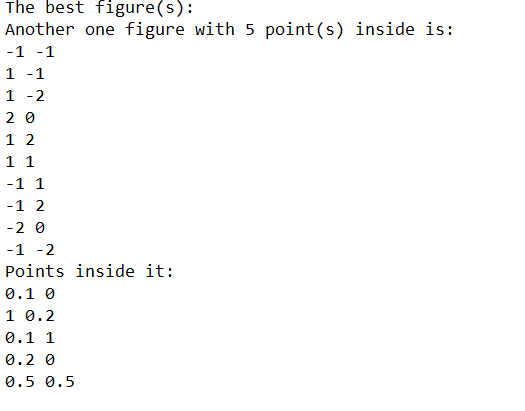
Результирующий файл



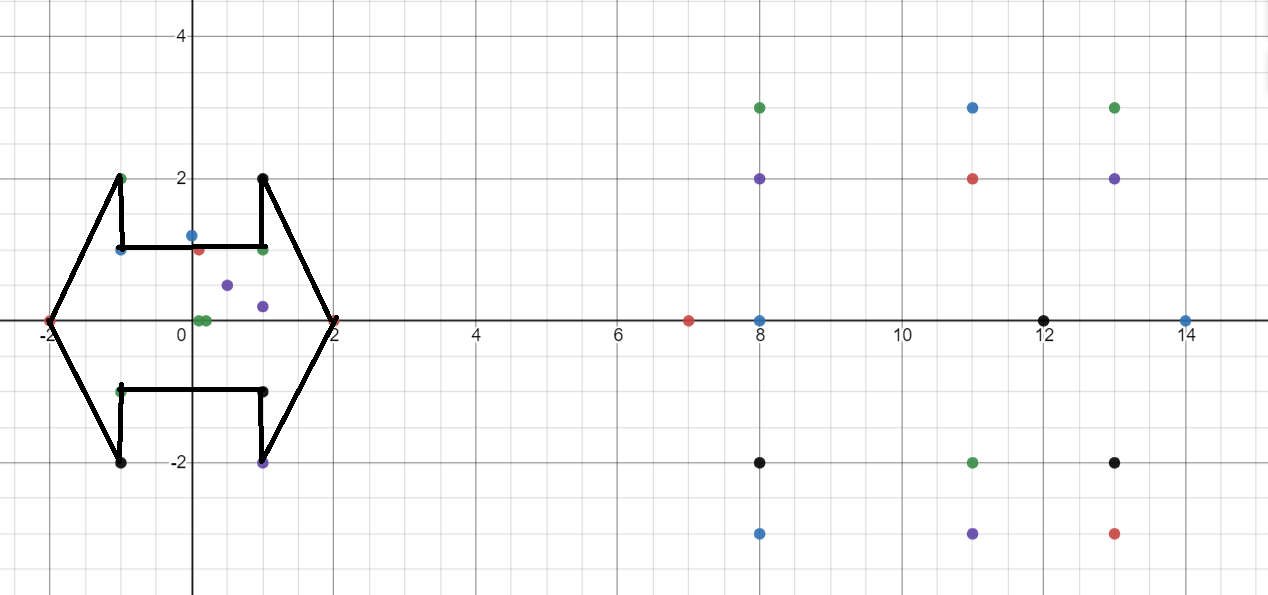




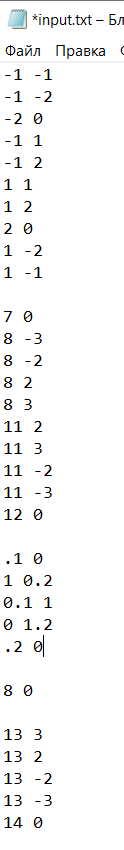
Результирующий файл



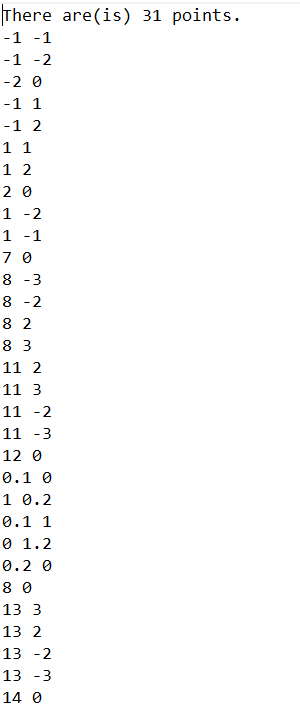
Рисунок

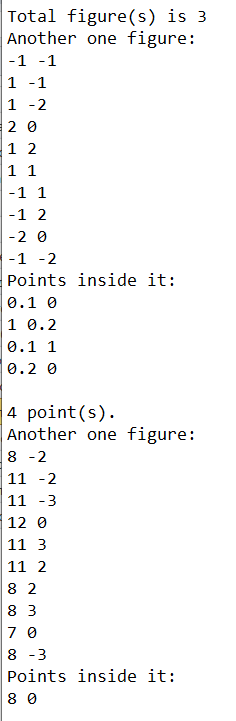


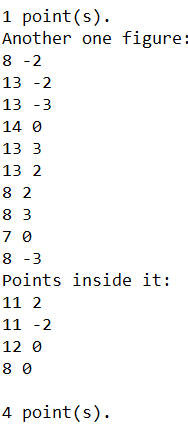
2.



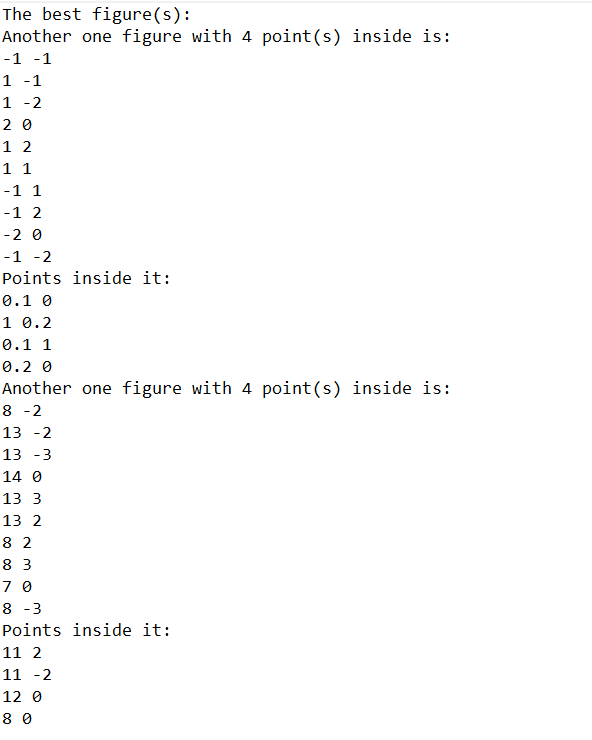
Результирующий файл



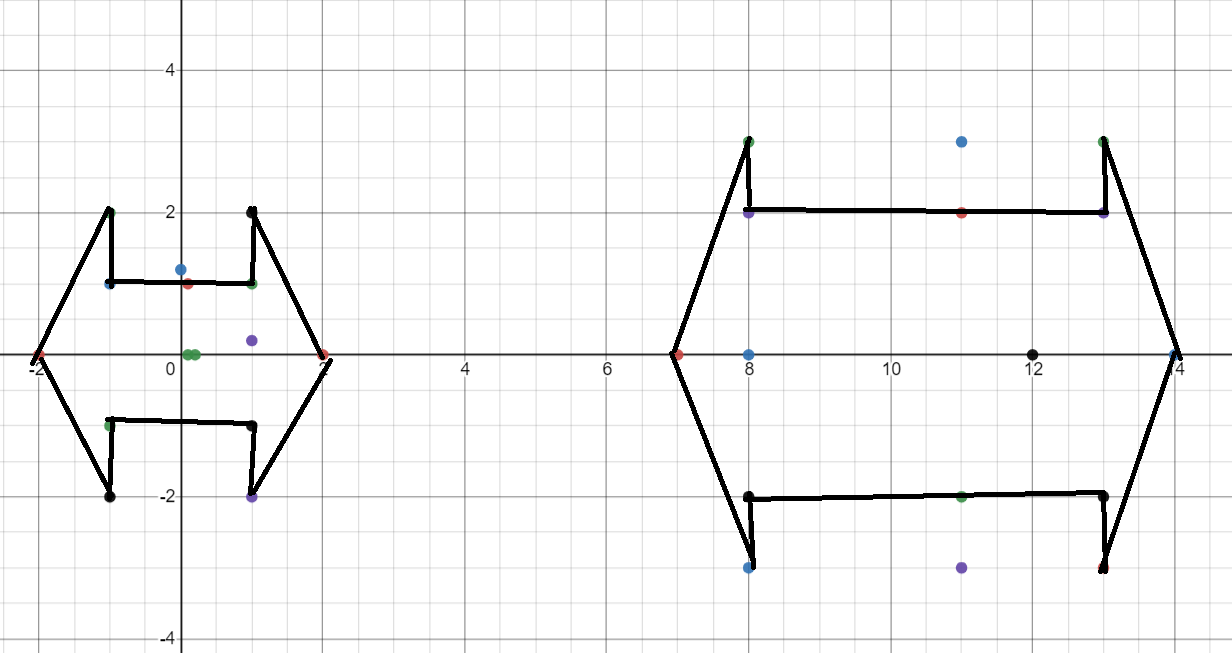




Результирующий файл



Рисунок



# Описание входных и входных файлов

Запускаем программу. На вход поступает файл input.txt. обрабатываться должны строки, имеющие 2 координаты как минимум. Если строка не содержит 2 координат, она пропускается, читается еще одна строка. Допустим, у нас есть текстовый файл со следующим содержанием:

-4 -8

-4 17

15 -15

-7 -24

1

1 -22

-48 29

Как видно, в пятой строчке отсутствует вторая координата, следовательно, эта строчка не будет обработана

На вывод поступает файл output.txt. в этот файл записывается окончательное решение нашей задачи — описание всех стрелок и описание всех стрелок с максимальным содержанием заданных точек или сообщение о том, что таких стрелок нет.

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

ifstream fin("input.txt");

ofstream fout("output.txt");

const int empty\_error = -999999;

struct point {

double x;

double y;

};

point p[100];

point was[100][10];

int insidecur[100];

int nwas = 0;

int check\_figure(point a[10]);

double area(point a, point b, point c);

bool check\_inside(point a[10], point p, int b[10]);

bool is\_on\_the\_line(point a, point b, point c);

bool first\_half(point a[5]);

bool intersect(point a, point b, point c, point d);

bool second\_exist(point a[5], point cen, point b[5]);

bool check\_unique(point a[5]);

bool fwas(point a[10]);

point find\_centre(point a, point b, point c, point d);

point read\_one\_line();

int n = 0;

int main() {

while (!fin.eof()) {

point a = read\_one\_line();

if(a.x != empty\_error and a.y != empty\_error){

p[n].x = a.x;

p[n++].y = a.y;

}

}

fout << "There are(is) " << n << " points.\n";

for (int i = 0; i < n; i++) fout << p[i].x << " " << p[i].y << endl;

cout << endl;

int nmax = -1;

for (int i1 = 0; i1 < n; i1++) {

for (int i2 = 0; i2 < n; i2++) {

for (int i3 = 0; i3 < n; i3++) {

for (int i4 = 0; i4 < n; i4++) {

for (int i5 = 0; i5 < n; i5++) {

point first\_cur[5] = { p[i1], p[i2], p[i3], p[i4], p[i5] };

if (!(check\_unique(first\_cur))) continue;

if (!first\_half(first\_cur)) continue;

point cen = find\_centre(p[i1], p[i2], p[i3], p[i5]);

point second\_cur[5];

if (!(second\_exist(first\_cur, cen, second\_cur))) continue;

point ans2[10];

ans2[0] = p[i1];

ans2[1] = p[i2];

ans2[2] = p[i3];

ans2[3] = p[i4];

ans2[4] = p[i5];

ans2[5] = second\_cur[0];

ans2[6] = second\_cur[1];

ans2[7] = second\_cur[2];

ans2[8] = second\_cur[3];

ans2[9] = second\_cur[4];

if (fwas(ans2)) continue;

}

}

}

}

}

if (!nwas) {

fout << "No one figure exist\n";

}

else {

fout << "\nTotal figure(s) is " << nwas << endl;

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

fout << "Another one figure:\n";

for (int j = 0; j < 10; j++) {

fout << was[i][j].x << " " << was[i][j].y << endl;

}

fout << "Points inside it:\n";

int cur = check\_figure(was[i]);

if (!cur) fout << "No one point in it!\n";

else fout << endl << cur << " point(s).\n";

insidecur[i] = cur;

if (cur > nmax) {

nmax = cur;

}

}

fout << "The best figure(s): \n";

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

if (insidecur[i] != nmax) continue;

fout << "Another one figure with " << insidecur[i] << " point(s) inside is:\n";

for (int j = 0; j < 10; j++) {

fout << was[i][j].x << " " << was[i][j].y << endl;

}

fout << "Points inside it:\n";

int cur = check\_figure(was[i]);

if (!cur) fout << "No one point in it!\n";

}

}

return 0;

}

bool fwas(point a[10]) {

int n = 10;

point b[10];

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

b[i] = a[n - 1 - i];

}

for (int k = 0; k < nwas; k++) {

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

int x = 1, x2 = 1;

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

if (was[k][(j) % n].x != a[(j + i) % n].x or was[k][(j) % n].y != a[(j + i) % n].y) x = 0;

if (was[k][(j) % n].x != b[(j + i) % n].x or was[k][(j) % n].y != b[(j + i) % n].y) x2 = 0;

}

if (x or x2) {

return 1;

}

}

}

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

was[nwas][i] = a[i];

}

nwas++;

return 1;

}

bool check\_unique(point a[5]) {

for (int i = 0; i < 5; i++) {

for (int j = i + 1; j < 5; j++) {

if (a[i].x == a[j].x and a[i].y == a[j].y) return 0;

}

}

return 1;

}

point find\_centre(point a, point b, point c, point d) {

point cab, ccd;

cab.x = (a.x + b.x) / 2;

cab.y = (a.y + b.y) / 2;

ccd.x = (c.x + d.x) / 2;

ccd.y = (c.y + d.y) / 2;

point cc;

cc.x = (cab.x + ccd.x) / 2;

cc.y = (cab.y + ccd.y) / 2;

point cen;

cen.x = (2 \* cc.x - b.x);

cen.y = (2 \* cc.y - b.y);

return cen;

}

bool second\_exist(point a[5], point cen, point b[5]) {

for (int i = 0; i < 5; i++) {

point sec;

sec.x = (cen.x - a[i].x) \* 2 + a[i].x;

sec.y = (cen.y - a[i].y) \* 2 + a[i].y;

//cout << sec.x << " " << sec.y << endl;

int x = 0;

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

if (p[j].x == sec.x and p[j].y == sec.y) x = 1;

}

if (!x) return 0;

b[i].x = sec.x;

b[i].y = sec.y;

}

return 1;

}

bool first\_half(point a[5]) {

point v1, v2;

v1.x = a[1].x - a[0].x;

v1.y = a[1].y - a[0].y;

v2.x = a[2].x - a[1].x;

v2.y = a[2].y - a[1].y;

//12 и 23 перпендикулярны

if (v1.x \* v2.x + v1.y \* v2.y) return 0;

//cout << "Complete 1\n";

// 34 и 45 равны

if ((a[4].x - a[3].x) \* (a[4].x - a[3].x) + (a[4].y - a[3].y) \* (a[4].y - a[3].y) != (a[2].x - a[3].x) \* (a[2].x - a[3].x) + (a[2].y - a[3].y) \* (a[2].y - a[3].y)) return 0;

//cout << "Complete 2\n";

// 12 и 34 не пересекаются

if (intersect(a[0], a[1], a[2], a[3])) return 0;

// 3 и 4 по разные стороны от прямой 1 2

//cout << "Complete 3\n";

point v3, v4;

v3.x = a[1].x - a[0].x;

v3.y = a[1].y - a[0].y;

v4.x = a[4].x - a[2].x;

v4.y = a[4].y - a[2].y;

//12 и 53 перпендикулярны

if (v3.x \* v4.x + v3.y \* v4.y) return 0;

// длина 34 больше 23

//cout << "Complete 4\n";

if ((a[2].x - a[1].x) \* (a[2].x - a[1].x) + (a[2].y - a[1].y) \* (a[2].y - a[1].y) >= (a[3].x - a[2].x) \* (a[3].x - a[2].x) + (a[3].y - a[2].y) \* (a[3].y - a[2].y)) return 0;

//cout << "Complete 5\n";

// 1 b 4 лежат по разные стороны

if (area(a[1], a[2], a[0]) \* area(a[1], a[2], a[3]) >= 0) return 0;

if (area(a[0], a[1], a[2]) \* area(a[0], a[1], a[3]) >= 0) return 0;

return 1;

}

double area(point a, point b, point c) {

return (b.x - a.x) \* (c.y - a.y) - (b.y - a.y) \* (c.x - a.x);

}

bool intersect(point a, point b, point c, point d) {

int res1 = area(c, a, b);

int res2 = area(d, a, b);

int res3 = area(a, c, d);

int res4 = area(b, c, d);

//cout << res1 << " " << res2 << " " << res3 << " " << res4 << endl;

return ((res1 \* res2) <= 0 and (res3 \* res4 <= 0));

}

bool is\_on\_the\_line(point a, point b, point c) {

if ((c.x - a.x) \* (b.y - a.y) == (c.y - a.y) \* (b.x - a.x) and

c.x >= min(a.x, b.x) and c.x <= max(a.x, b.x) and

c.y >= min(a.y, b.y) and c.y <= max(a.y, b.y))

return 1;

return 0;

}

bool check\_inside(point a[10], point p, int b[10]) {

//cout << p.x << " " << p.y << endl;

for (int i = 0; i < 10; i++) {

if (p.x == a[i].x and p.y == a[i].y and !b[i]) {

b[i]++;

return 0;

}

else if (p.x == a[i].x and p.y == a[i].y) {

return 1;

}

}

//cout << "not a point\n";

for (int i = 0; i < 10; i++) {

if (is\_on\_the\_line(a[i], a[(i + 1) % 10], p)) {

//fout << p.x << " " << p.y << " " << "on line" << "\n";

return 1;

}

}

//cout << "not on the line\n";

int count = 0;

for (int i = 0; i < 10; i++) {

point p2;

p2.y = p.y;

p2.x = 100;

//cout << a[i].x << " " << a[i].y << " " << a[(i + 1) % 10].x << " " << a[(i + 1) % 10].y << endl;

//cout << intersect(a[i], a[(i + 1) % 10], p, p2) << endl;

if (intersect(a[i], a[(i + 1) % 10], p, p2) and p.y != min(a[i].y, a[(i + 1) % 10].y)) count++;

}

//cout << count << endl;

//fout << p.x << " " << p.y << " " << count << "\n";

return count % 2;

}

int check\_figure(point a[10]) {

int inside = 0;

int b[10]{};

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

if (check\_inside(a, p[i], b)) {

inside++;

fout << p[i].x << " " << p[i].y << endl;

}

//if (check\_inside(a, p[i])) cout << "One of this point is : " << p[i].x << " " << p[i].y << endl;

}

return inside;

}

point read\_one\_line() {

/// -999999 = end of line

point a;

a.x = empty\_error;

a.y = empty\_error;

string s;

getline(fin, s);

int j = 0;

for (int k = 0; k < 2; k++) {

int znak = 1;

double res = 0;

int x = 0;

double power = 10;

int d = 0;

for (int i = j; i < s.size(); i++, j++) {

if (s[i] == '-') {

znak = -1;

x = 1;

}

else if (s[i] == '.') {

d = 1;

x = 1;

}

else if (s[i] <= '9' and s[i] >= '0') {

x = 1;

if (d) {

res += double(s[i] - '0') / power;

power \*= 10;

}

else {

res = res \* 10 + s[i] - '0';

}

}

else if (x) {

break;

}

}

if (x) {

if (k == 0) {

a.x = res \* znak;

}

else a.y = res \* znak;

}

}

return a;

}

# Заключение

В процессе выполнения данной работы были закреплены навыки и знания, полученные в течение первого семестра. Освоен метод работы с геометрическими фигурами на языке программирования с++.

# Список использованных источников

1. Материалы в Интернете: <https://habr.com/ru/post/301102/>
2. Знания, полученные в течение семестра
3. Тематические форумы по программированию