**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

**Курсовая работа**

**по дисциплине «Программирование»**

Тема: Использование массивов для решения

геометрических задач

|  |  |  |
| --- | --- | --- |
| Студент гр. 4353 |  | Кацер М.А. |
| Преподаватель |  | Калмычков В.А. |

Санкт-Петербург

2024

# Задание на курсовую работу

Студент: Кацер Марк Алексеевич

Группа: 4353

Тема работы: использование массивов для решения геометрических задач на

языке с++.

Исходные данные:

Дано N произвольных точек на плоскости. Найти среди них точки, которые являются вершинами (образующими) фигур, обладающих следующими свойствами: с максимальным числом точек в области (внутренней или строго выделенной).

Содержание пояснительной записки:

Задание на курсовую работу, аннотация, содержание, основная часть, исходная формулировка задания, анализ задания, устранение неточностей, математическая постановка задачи, контрольный пример, ограничения, разработка интерфейса пользователя, библиотеки, внутреннее представление данных в программе, функции, представление алгоритма, результаты работы программы, вывод по проделанной работе.

Предполагаемый объём пояснительной записки:

Не менее 28 страниц

Дата выдачи задания: 18.11.2024

Дата сдачи:

Дата защиты:

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. 4353 |  | Кацер М.А. |
| Преподаватель |  | Калмычков В.А. |

# Аннотация

В данной курсовой работе рассмотрено решение геометрической задачи с использованием массивов и чтением из файла на языке программирования C++. Задание заключается в анализе трапеций, их свойств и внутренних точек.

Для хранения данных выбран комбинированный способ. Используется двумерный динамический массив: каждая строка массива отвечает за хранение координат одной точки, а структура программы позволяет эффективно обрабатывать все данные, считываемые из файла.

Чтение файла выполняется последовательно в соответствии с заданными условиями, включая проверку корректности данных и исключение дублирующихся точек. Далее осуществляется перебор всех возможных комбинаций четырёх точек, формирование трапеций, проверка их геометрических свойств (наличие одной пары параллельных сторон и отсутствие пересечений), а также подсчёт количества точек, лежащих внутри каждой трапеции.

Результатом работы является вывод информации о:

* максимальном количестве точек, находящихся внутри какой-либо трапеции;
* всех трапециях, обладающих таким количеством внутренних точек;
* всех возможных комбинациях, где удалось построить трапецию.

В работе приводится описание математического решения задачи, включая проверку свойств трапеции и вычисление внутренних точек, а также алгоритма реализации этого решения на компьютере.

Summary

This coursework addresses the solution of a geometric problem using arrays and file reading in the C++ programming language. The task involves analyzing trapezoids, their properties, and the internal points contained within them.

A combined data storage approach is used. A two-dimensional dynamic array is employed: each row of the array stores the coordinates of a single point, and the program's structure enables efficient processing of all data read from the file.

The file is read sequentially according to the specified conditions, including data validation and the elimination of duplicate points. Subsequently, all possible combinations of four points are examined to form trapezoids, verify their geometric properties (one pair of parallel sides and no intersecting sides), and count the number of points located within each trapezoid.

The result of the program includes the following outputs:

* the maximum number of points contained inside any trapezoid;
* all trapezoids with such a number of internal points;
* all possible combinations where trapezoids could be constructed.

The work provides a description of the mathematical solution to the problem, including verifying trapezoid properties and calculating internal points, as well as the algorithm for implementing this solution on a computer.

Содержание

[Задание на курсовую работу 2](#_Toc184044370)

[Аннотация 3](#_Toc184044371)

[Содержание 4](#_Toc184044372)

[Основная часть 5](#_Toc184044373)

[Исходная формулировка задания 5](#_Toc184044374)

[Анализ задания, устранение неточностей 5](#_Toc184044375)

[Математическая постановка задачи 5](#_Toc184044376)

[Контрольный пример 6](#_Toc184044377)

[Ограничения 6](#_Toc184044378)

[Разработка интерфейса пользователя 6](#_Toc184044379)

[Библиотеки 7](#_Toc184044380)

[Внутреннее представление данных в программе 8](#_Toc184044381)

[Функции 9](#_Toc184044382)

[Представление алгоритма 9](#_Toc184044383)

[Результаты работы программы 22](#_Toc184044384)

[Вывод по проделанной работе 23](#_Toc184044385)

# **ОСНОВНАЯ ЧАСТЬ**

# Исходная формулировка задания

Дано N произвольных точек на плоскости. Найти среди них точки,

являющиеся вершинами (образующими) фигур, обладающих следующими

свойствами: с максимальным числом точек в области (внутренней или строго

выделенной). Задание выполнить для фигуры трапеция.

# Анализ задания, устранение неточностей

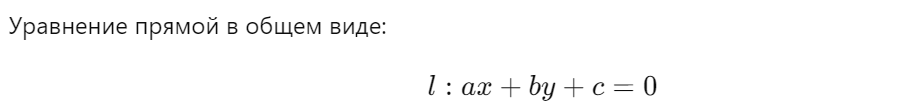
Каждое число в массивах ∈ R.

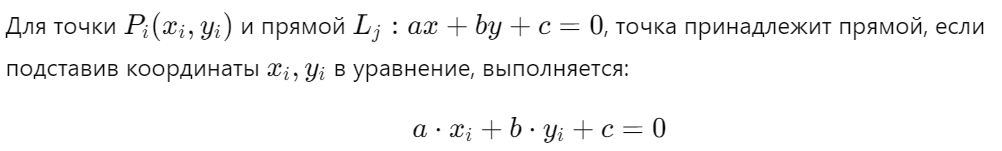
# Математическая постановка задачи

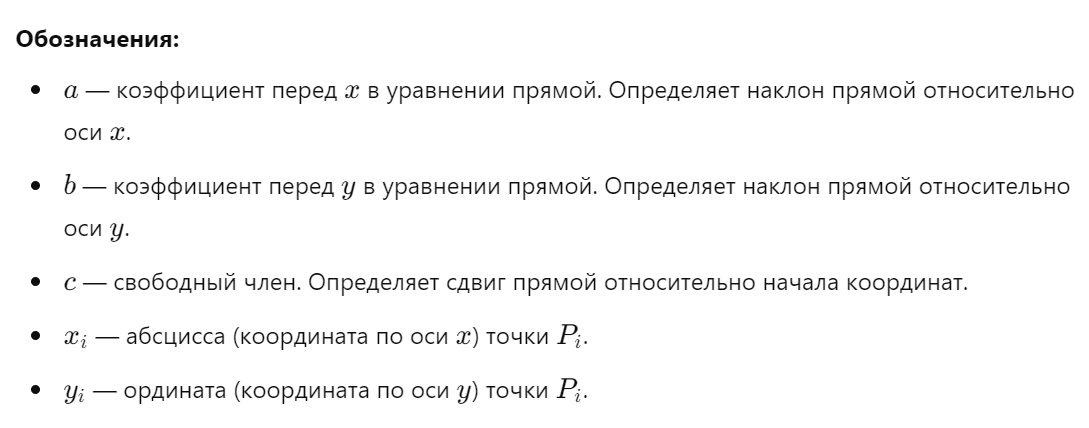
Дано: координаты точек

Найти: найти среди них точки, являющиеся вершинами (образующими) фигур, обладающих следующими свойствами: с максимальным числом точек в области (внутренней или строго выделенной)

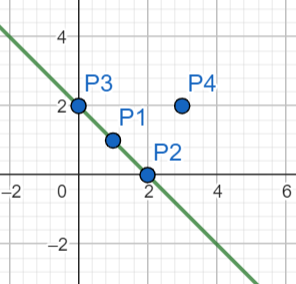
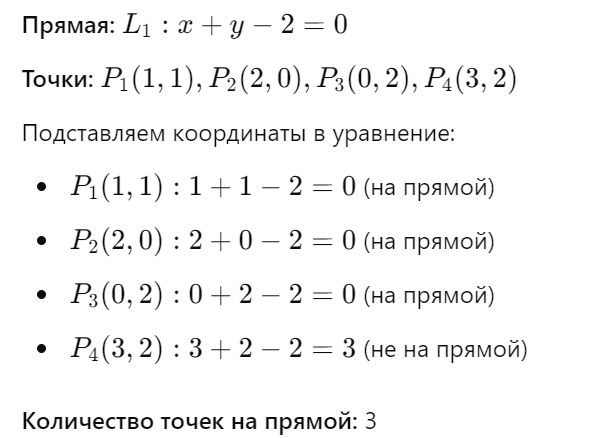
Способ решения:

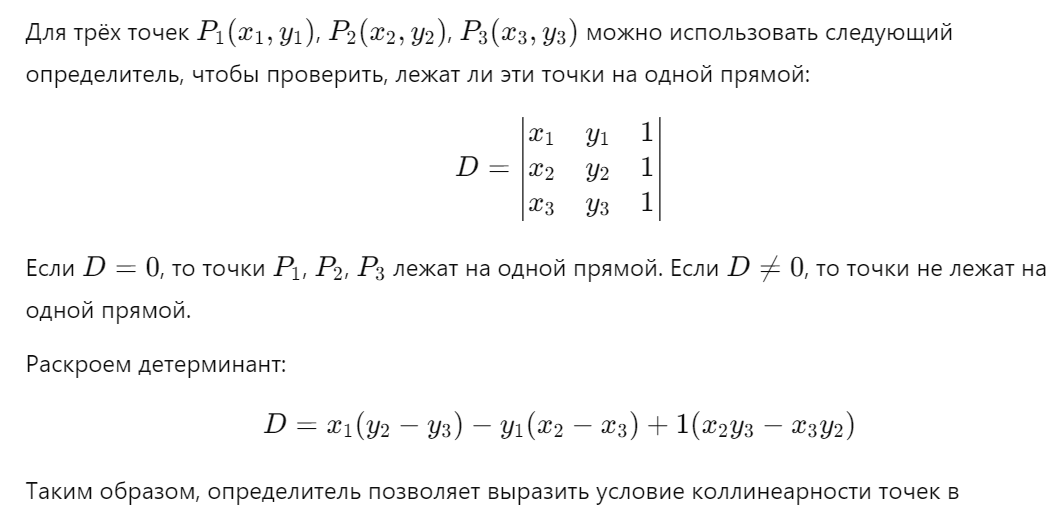


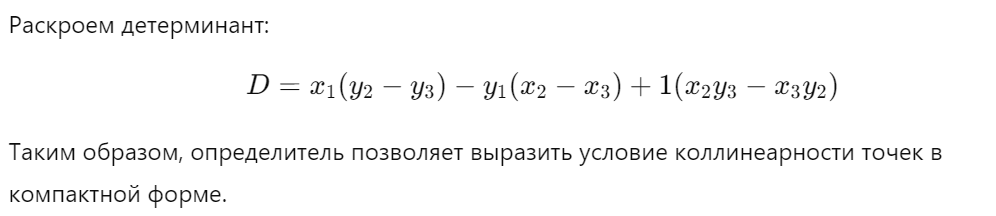


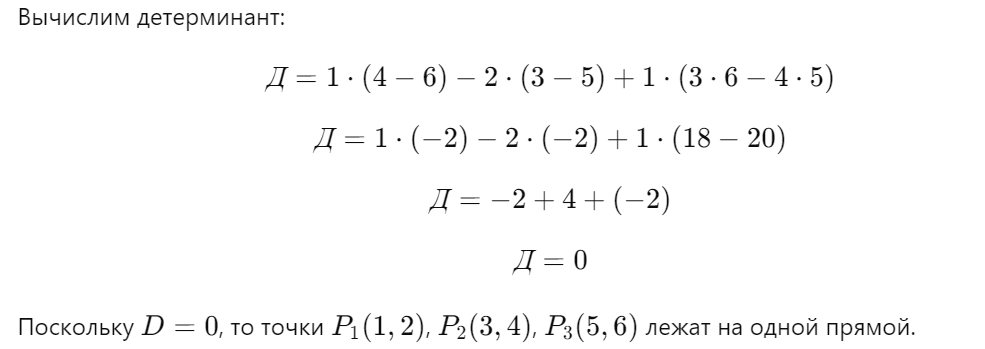


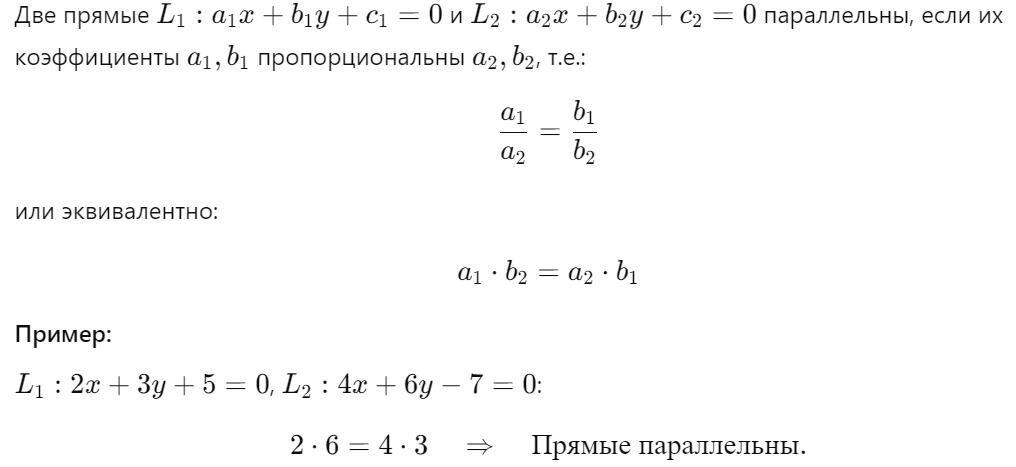


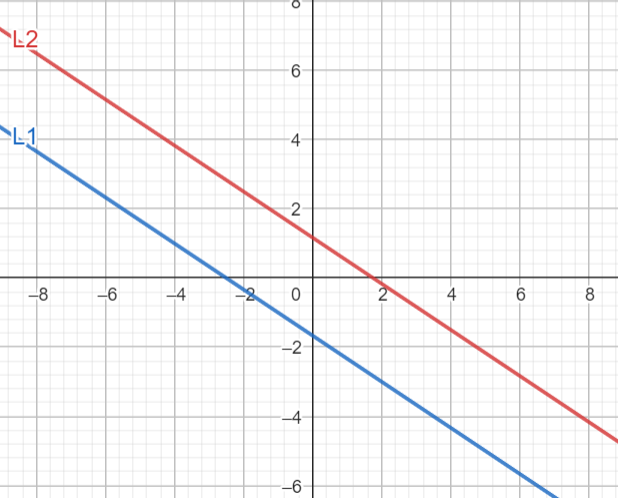


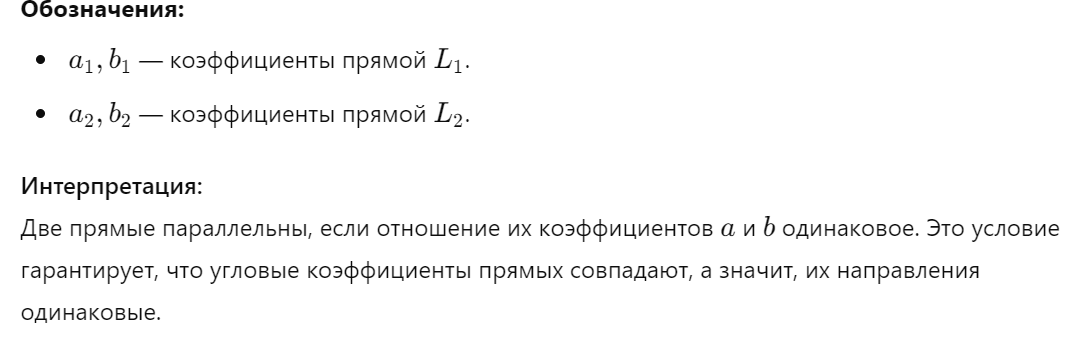


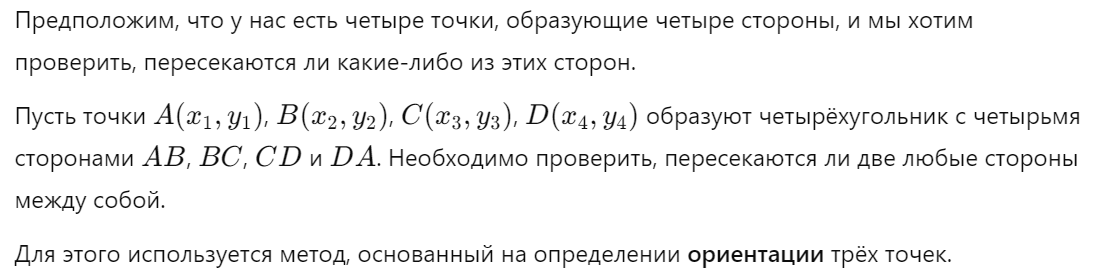


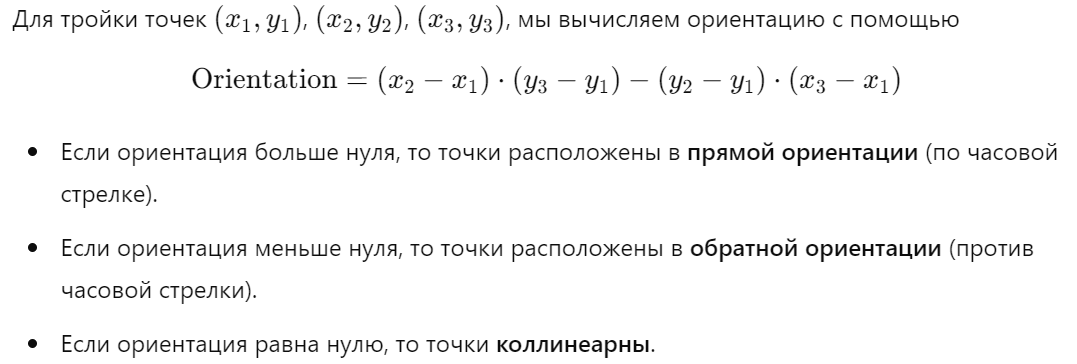


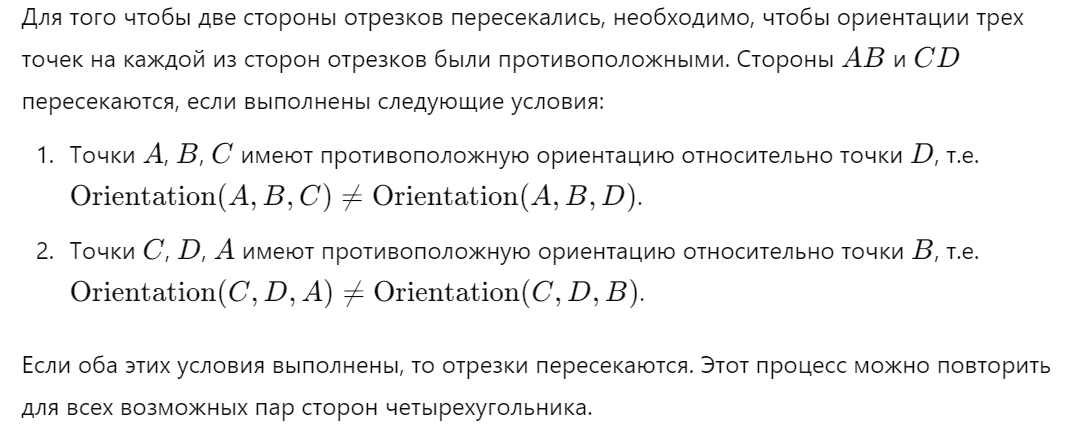


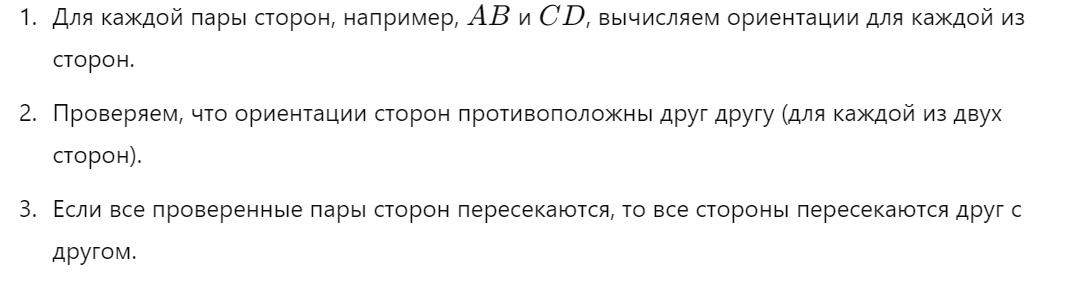


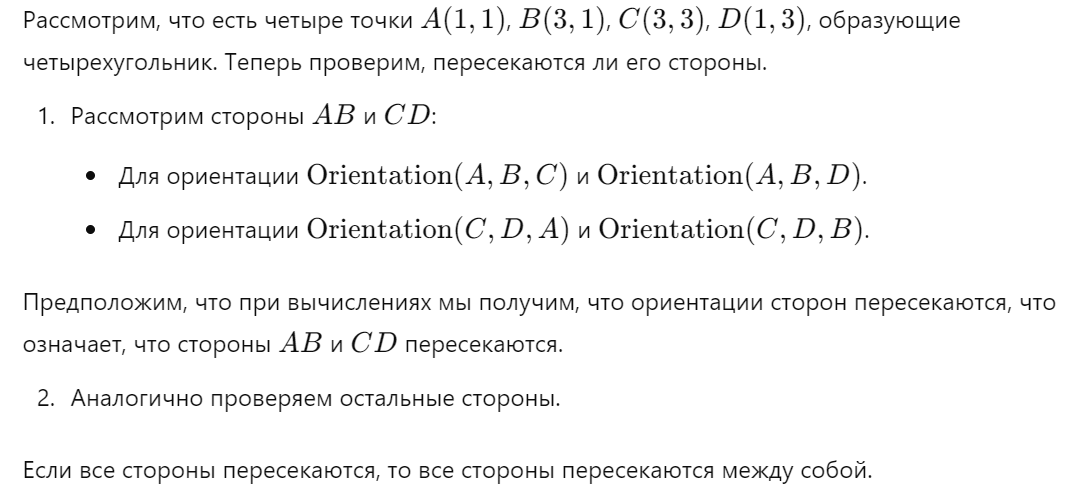


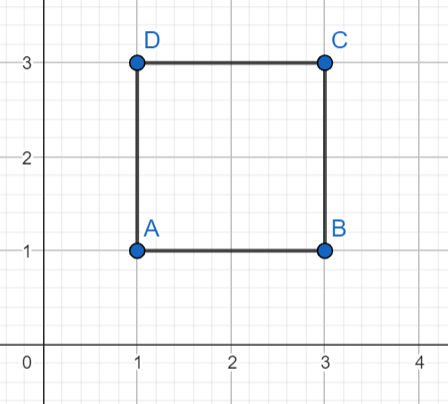






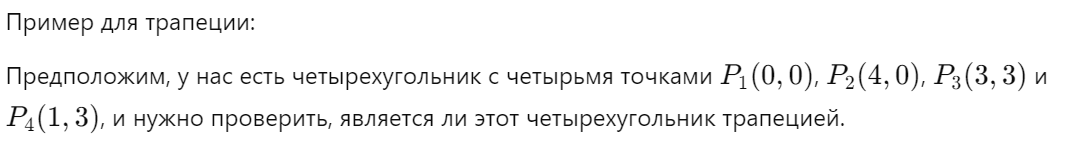


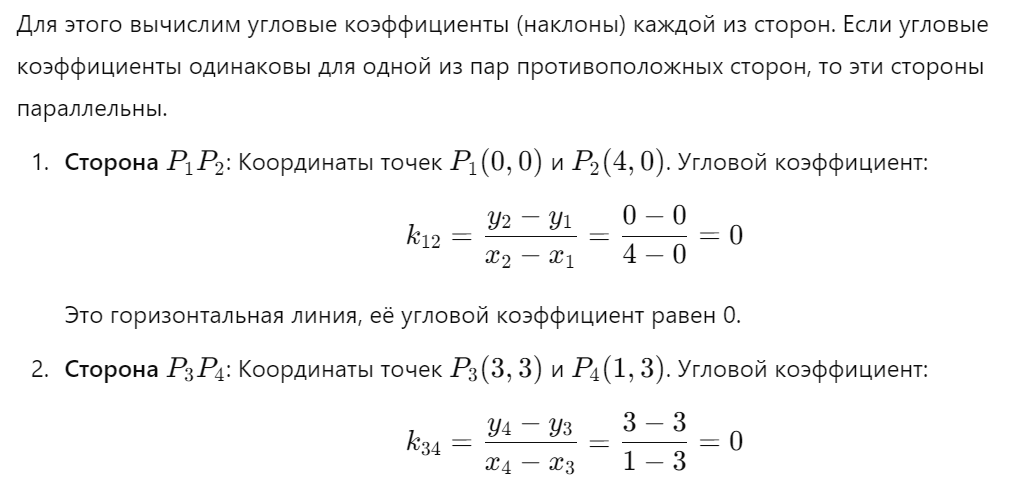


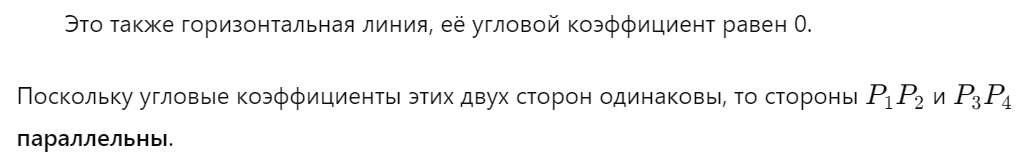


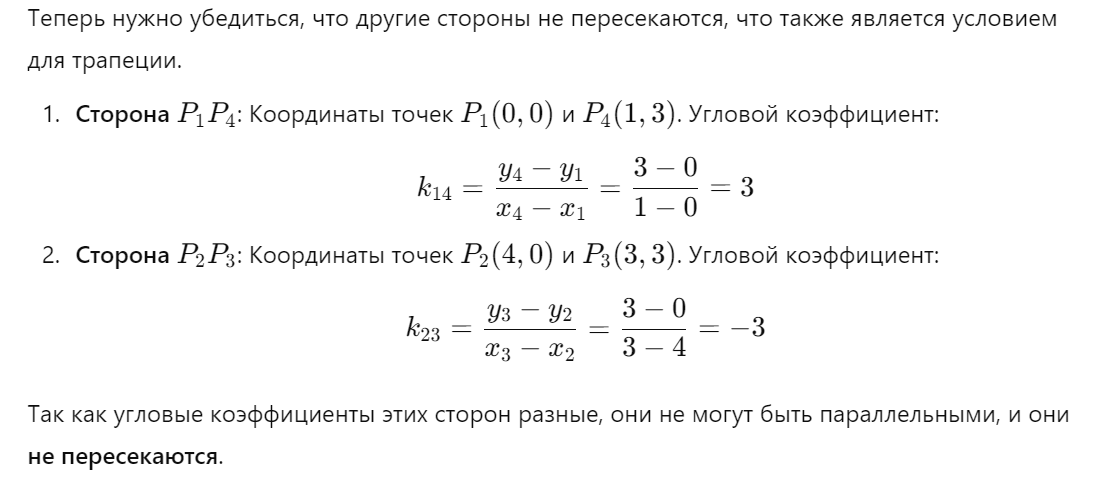
Итак, для того чтобы четырехугольник был трапецией, должны выполняться следующие условия:

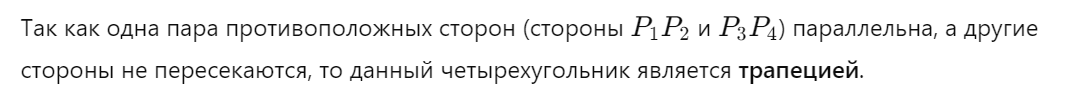
1. **Наличие хотя бы одной пары параллельных сторон**: Трапеция — это четырехугольник, у которого существует одна пара противоположных сторон, которые параллельны. Чтобы проверить, что стороны параллельны, необходимо, чтобы коэффициенты угловых наклонов этих сторон были одинаковыми. Для этого проверяется условие, при котором разность координат для каждой стороны дает одинаковое значение углового наклона.
2. **Проверка пересечения сторон**: Важно, чтобы другие стороны не пересекались, так как это может нарушить структуру трапеции. Для проверки того, что стороны не пересекаются, используется метод ориентации точек и проверки противоположности ориентаций сторон. Для каждой пары сторон проверяется, пересекаются ли они, используя соответствующие геометрические формулы и вычисления.
3. **Учитывая все вышеизложенные условия**, можно точно определить, является ли данный четырехугольник трапецией. Если выполняются условия параллельности одной пары сторон и отсутствие пересечений для других сторон, то четырехугольник можно считать трапецией.

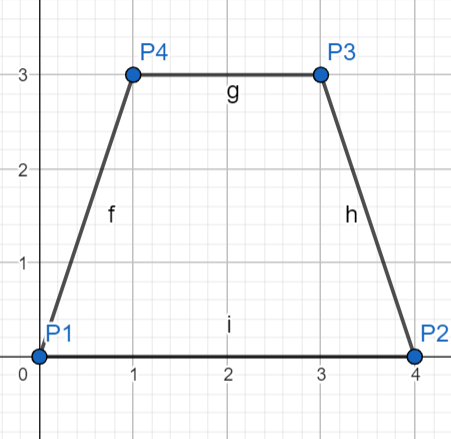




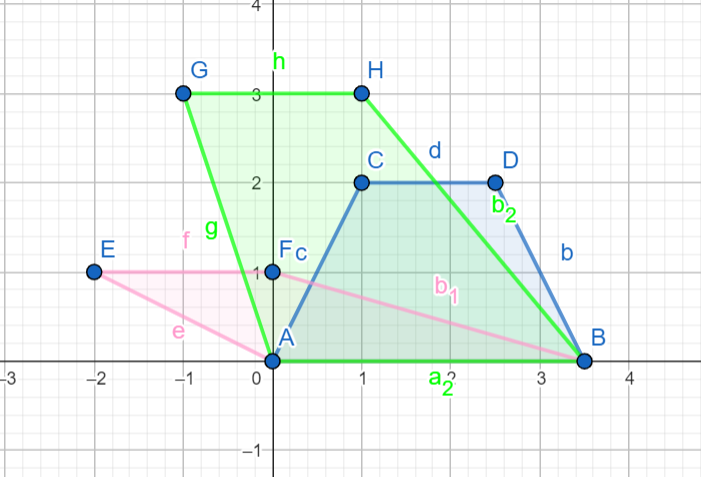
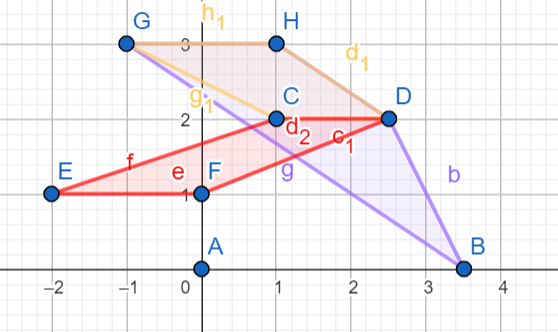








# Контрольный пример

1 трапеция (синяя): 0 точек 4 трапеция (фиолетовая): 1 точка

2 трапеция (розовая): 0 точек 5 трапеция (желтая): 0 точек

3 трапеция (зеленая): 2 точек 6 трапеция (красная): 0 точек

# Ограничения

Программа использует сравнение с учётом погрешности (EPSILON = 1e-10)

# Разработка интерфейса пользователя

O1:Задание: Дано N произвольных точек на плоскости. Найти среди них точки, являющиеся вершинами (образующими) фигур, обладающих следующими

свойствами: с максимальным числом точек в области (внутренней или строго

выделенной). Задание выполнить для фигуры трапеция.

Автор: Смирнова Наталья Евгеньевна Группа: 4353  
Дата: 18.11.24

O2: Сколько точек надо считать

O3: Остались ли данные в файле

O4: Точка считана

O5: Неверные данные, пропуск строки

O6: Что отброшено

O7: Считано меньше точек чем предполагалось

O8: Трапеция составлена из точек

O9: Трапецию составить нельзя

O10: Комбинация точек уже записана

O11: Какая точка вошла в трапецию

O12: Точка не вошла в трапецию

O13: Сколько точек вошло в трапецию

O14: Сколько точек считано

O15: Сколько трапеций можно составить из данного кол-ва точек

O16: Какие точки вошли в трапецию

O17: Какие трапеции имеют максимальное кол-во точек

O18: Сколько трапеций имеют максимум точек

O19: Нет данных для обработки

O20: Данные считанны

O21: Нет трапеций для обработки

O22: Трапеции составлены

O19: Точки внутри трапеций подсчитанны

O20: Максимум точек подсчитан

# Библиотеки

|  |  |  |
| --- | --- | --- |
| Библиотека | Элементы | Назначение |
| fstream | ofstream | Вывод чисел и текста |
| \n | Переход к новой строке |
| ifstream | Ввод |
| string | std::getline | Чтение строк из файла |
| std::stod | Преобразование строк координат в числа |
| substr | Для разбора строк на координаты |
| find |
| cmath | sqrt | Вычисление расстояний между точками |
| pow |
| fabs | Для проверки равенства длин с учетом погрешности |

# Внутреннее представление данных в программе

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| n1, n2 | int | Подсчет количества точек в массиве |
| c | Количество найденных трапеций |
| count\_dots | Число точек внутри одной трапеции |
| md | Максимальное количество точек внутри трапеции |
| nom | Счетчик |
| i, j, h, k | Индексы для перебора точек и составления трапеций |
| ct0, ct1 | Счетчики знаков при проверке принадлежности точки области |
| sign1, sign2, sign3, sign4 | Знаки для определения позиции точки относительно сторон трапеции |
| mcount\_dots | Массив количества точек внутри каждой трапеции |
| t | Массив текущих знаков для одной точки относительно сторон трапеции |
| \*t1, \*t2 | int\* | Упорядоченные индексы точек трапеции |
| trap[i] | Массив индексов точек, составляющих одну трапецию |
| \*\*trap | int\*\* | Массив трапеций, каждая из которых представлена четырьмя индексами точек |
| a1, b, a, b1, c1 | double | Коэффициенты уравнений прямых (линии сторон трапеции) |
| pr1, pr2, pr3, pr4 | Массивы координат концов сторон трапеции |
| a | Координаты одной точки для добавления в массив |
| d | Координаты одной точки для добавления в массив |
| a[i] | double\* | Массив координат одной точки |
| mdint | Массив координат всех точек внутри конкретной трапеции |
| \*\*a | double\*\* | Двумерный массив для хранения координат всех точек |
| \*\*mdots\_in\_trapezium[i] | Массив координат точек, принадлежащих одной трапеции |
| \*\*\*mdots\_in\_trapezium | double\*\*\* | Трехмерный массив координат точек, принадлежащих всем трапециям |
| set\_dots | bool | Проверяет, уникальна ли точка |
| peresek | Определяет, пересекаются ли два отрезка |
| are\_parallel | Проверяет, параллельны ли два отрезка |
| check\_trp | Проверяет, образуют ли четыре точки трапецию |
| m\_ravn | Сравнивает два массива точек |
| check\_app | Проверяет существование трапеции с такой же комбинацией точек |
| in\_area | Проверяет, находится ли точка внутри заданной трапеции |
| EPSILON | Используется для проверки параллельности или близости чисел |
| s | char | Временная строка для пропуска некорректных данных при чтении из файла |

# 

# Функции

|  |  |  |
| --- | --- | --- |
| Имя | Параметры | Назначение |
| set\_dots | double \*\*ar,  int n, int a, b | Проверяет, уникальна ли точка с координатами  (a, b) в массиве точек |
| input\_dots | ifstream &fin,  int &n1, ofstream &log | Считывает точки из файла, проверяет их корректность и записывает в массив |
| peresek | const double pr1[4],  const double pr2[4] | Проверяет пересечение двух отрезков |
| are\_parallel | double pr1[4],  double pr2[4] | Проверяет, являются ли два отрезка параллельными |
| check\_trp | pr1, pr2, pr3, pr4 | Проверяет, можно ли из четырех отрезков составить трапецию |
| bouble\_sort | int \*a | Сортирует массив из четырех индексов для однозначного представления трапеции |
| m\_ravn | int \*a1, int \*a2 | Сравнивает два массива индексов точек |
| check\_app | int \*\*trap, int n,  int value[4] | Проверяет, уникальна ли трапеция среди уже добавленных |
| append\_trap | int \*\*trap, int n,  int value[4] | Добавляет новую трапецию в массив трапеций |
| ABC | const double pr[4],  double &a, &b, &c | Вычисляет коэффициенты уравнения прямой по отрезку |
| znak | double a, b, c,  const double p[2] | Определяет положение точки относительно прямой |
| in\_area | double a[2],  pr1, pr2, pr3, pr4 | Проверяет, находится ли точка внутри трапеции |
| trapezium | int &c, double \*\*a,  int n, ofstream &log | Ищет все возможные трапеции среди заданных точек |
| append\_dot | double \*\*mdint,  double d[2], int n | Добавляет точку в массив точек внутри трапеции |
| dots\_in\_trapezion | double \*\*a, int c,  int \*\*trap, int n,  double \*\*\*mdint,  ofstream &log | Определяет точки, которые находятся внутри каждой трапеции |
| output\_dots\_in\_trapezion |  | Выводит информацию о точках внутри трапеций в лог и файл |
| output\_max\_dots\_in\_trapezion |  | Выводит трапеции, содержащие максимальное число точек |
| main |  | Основная программа |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя | Возвращаемое значение | Параметры | Входной | Выходной | Модифицируемый | Транзитный |
| setpoints | True/False | p, n, a, b | p, n, a, b | - | - | - |
| setlines | True/False | l, n, a, b, c | l, n, a, b, c | - | - | - |
| appoint | new\_points | points, n, a, b | points, n, a, b | new\_points | - | - |
| appline | new\_lines | lines, n, a, b, c | lines, n, a, b, c | new\_lines | - | - |
| inputpoints | - | p, n, fin, log | fin, log | - | fout, n, s | - |
| inputlines | - | l, n, fin, log | fin, log | - | fout, n, s | - |
| create\_section | sec | p, np, ns, log | p, np, log | sec | ns, k | - |
| check\_pararel | True/False | a1, a2, b1, b2 | a1, a2, b1, b2 | True/False | - | - |
| apppws | newpws | pws, n, nline | pws, n, nline | newpws | - | - |
| printpoint | - | p[2], fout | p[2], fout | - | - | - |
| printline | - | l[3], fout | l[3], fout | - | - | - |
| process | pws | sec, c, lin, ns, nl, log | sec, c, lin, ns, nl, log | pws | - | - |
| getmax | maxc | cpl, n | cpl, n | maxc | - | - |
| print\_decision | - | p, l, s, cpl, pws, mcl, np, fout | p, l, s, cpl, pws, mcl, np, fout | - | k, count\_max | - |
| clear\_all | - | points, np, lines, nl, section, ns, cpl, pws | points, np, lines, nl, section, ns, cpl, pws | - | - | - |

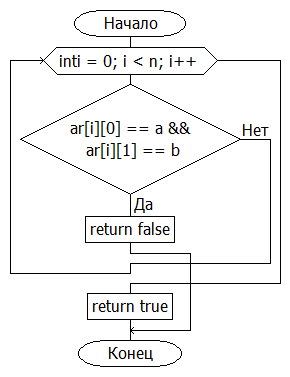
# Представление алгоритма

Краткое пояснение выполняемых действий

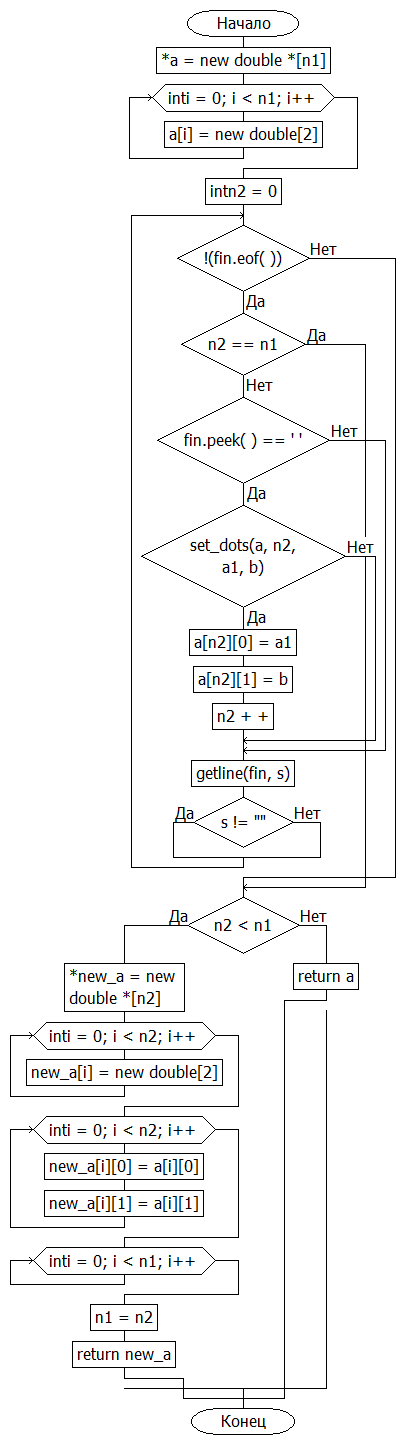
Для решения задачи создаем динамический двумерный массив и заполняем его числами. Обходим строки матрицы. Для каждой строки проверяем все возможные группы из четырех соседних элементов. Если среди них находится возрастающая последовательность, увеличиваем счетчик таких строк и переходим к следующей строке. В конце выводим количество строк с возрастающими последовательностями.

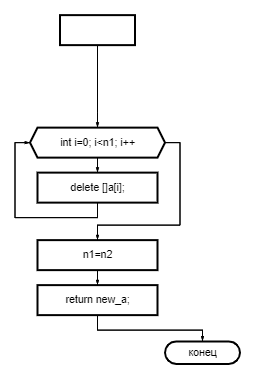
Блоксхема

Для первой функции:

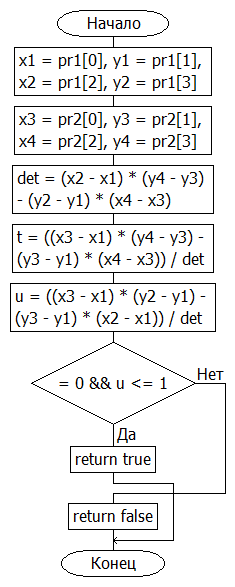


Для второй функции:

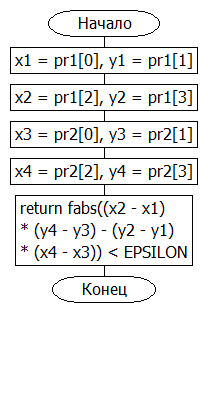




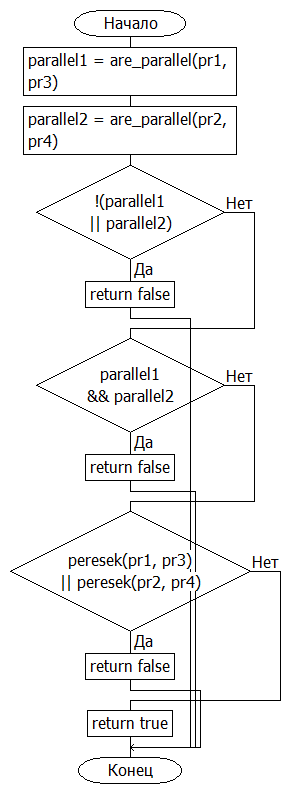
Для третьей функции:



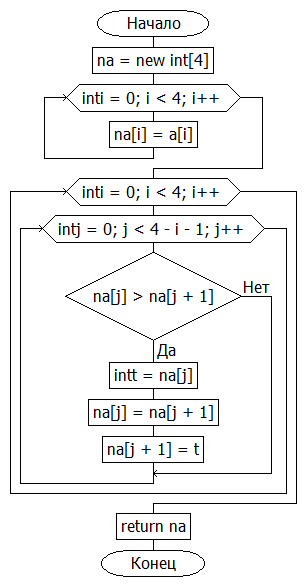
Для четвертой функции:



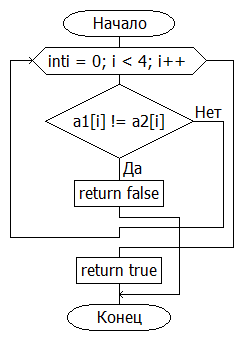
Для пятой функции:



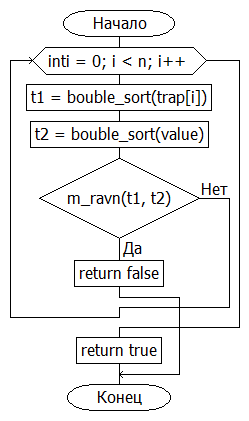
Для шестой функции:



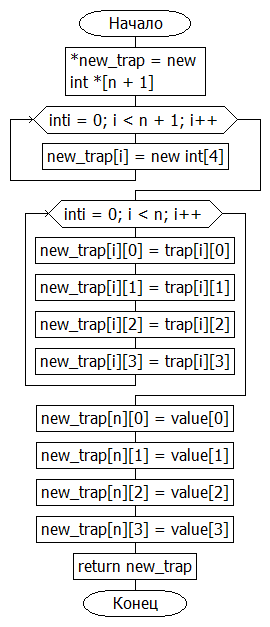
Для седьмой функции:



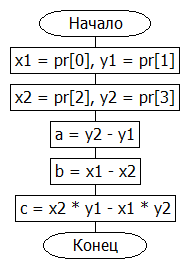
Для восьмой функции:



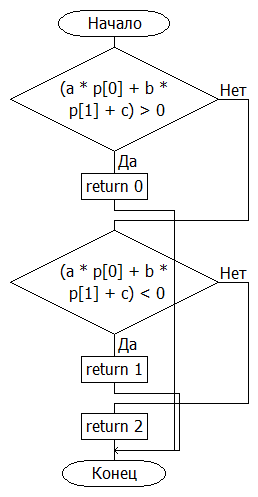
Для девятой функции:



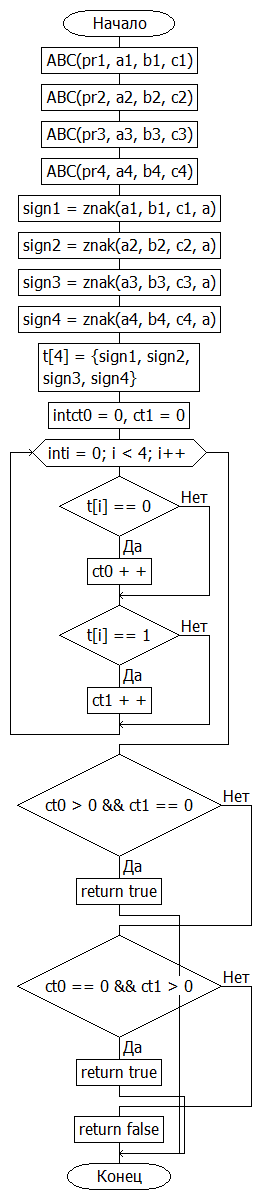
Для десятой функции:



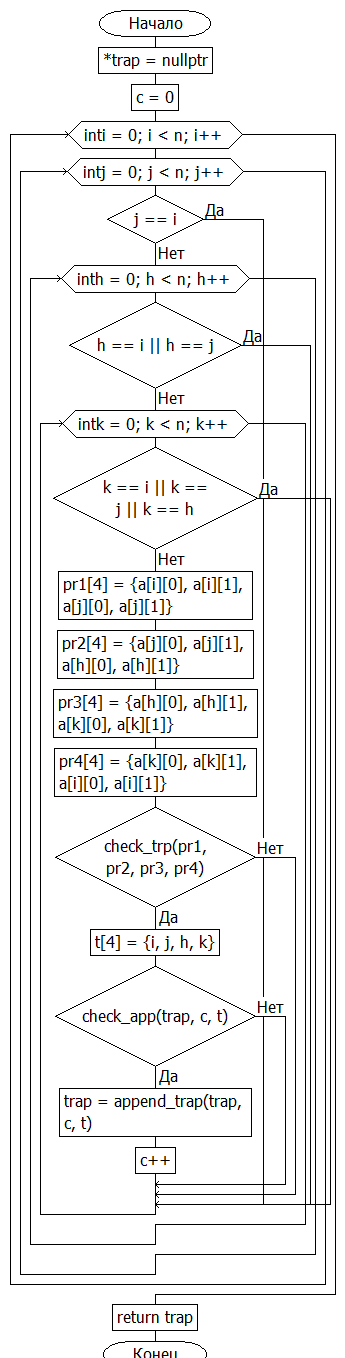
Для одиннадцатой функции:



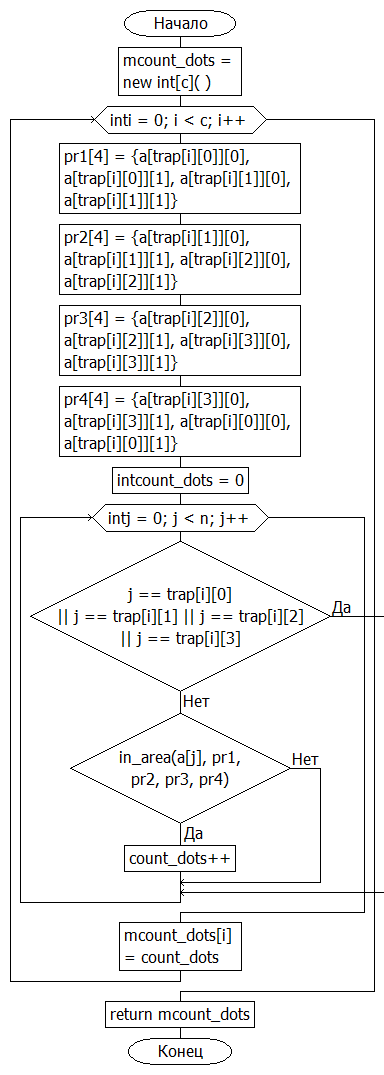
Для двенадцатой функции:



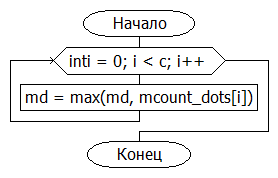
Для тринадцатой функции:



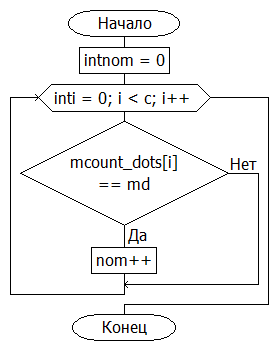
Для четырнадцатой функции:



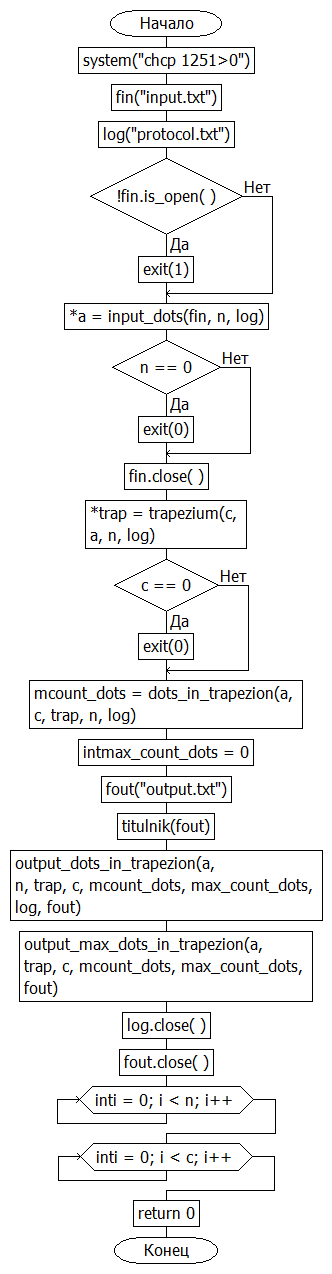
Для пятнадцатой функции:

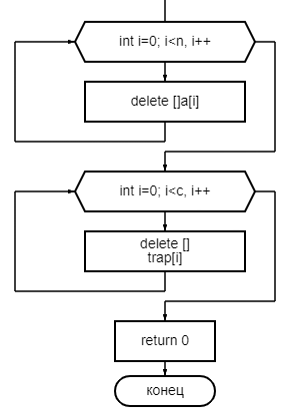


Для шестнадцатой функции:



Для семнадцатой функции:

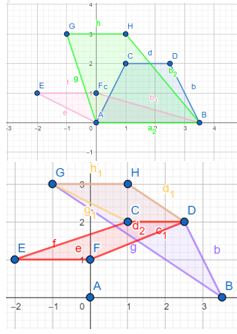
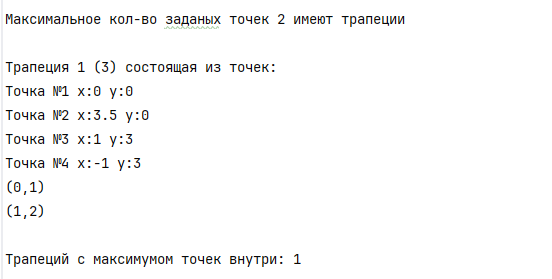




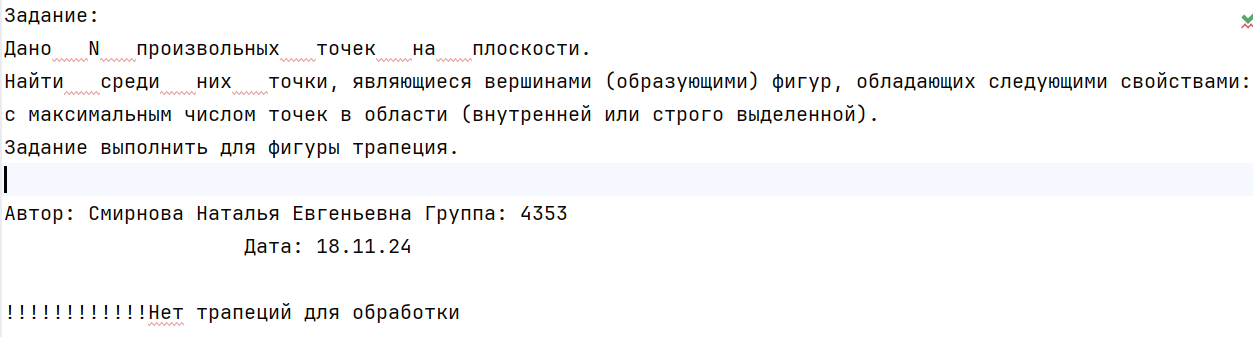
Текст программы#include <iostream>  
#include <fstream>  
#include <string>  
#include <cmath>  
  
using namespace std;  
  
void titulnik(ofstream&fout) {  
 fout << "Задание:" << endl;  
 fout << "Дано N произвольных точек на плоскости. \n"  
 "Найти среди них точки, являющиеся вершинами (образующими) фигур, обладающих следующими свойствами: \n"  
 "с максимальным числом точек в области (внутренней или строго выделенной). \n"  
 "Задание выполнить для фигуры трапеция. \n"<< endl;  
 fout << "Автор: Смирнова Наталья Евгеньевна Группа: 4353" << endl;  
 fout << "\t\t\t\t\tДата: 18.11.24" << endl;  
 fout << endl;  
}  
  
bool set\_dots(double\*\*ar,int n,int a,int b) {  
 for (int i=0;i<n;i++) {  
 if (ar[i][0]==a && ar[i][1]==b) {  
 return false;  
 }  
 }  
 return true;  
}  
  
  
double \*\* input\_dots(ifstream &fin,int &n1,ofstream&log) {  
 fin>>n1; double \*\*a=new double\*[n1];  
 log<<"Надо считать "<<n1<<" точек"<<endl;  
 for(int i=0;i<n1;i++) { a[i]=new double [2];}  
 int n2=0;  
 while (!(fin.eof())) {  
 if (n2==n1) {log << "В файле остались данные, не считываем"<<endl; break;}  
 double a1, b;  
 fin>>a1;  
 if (fin.peek()==' ') {  
 fin>>b;  
 if(set\_dots(a,n2,a1,b)) {  
 log<<"Cчитана точка "<<n2+1<<" ("<<a1<<","<<b<<")"<<endl;  
 a[n2][0]=a1;  
 a[n2][1]=b;  
 n2++;  
 }  
 }  
 else log<<"Неверные данные, пропуск строки"<<endl;  
 string s;  
 getline(fin,s);  
 if (s!="") log<<s<<" отброшено"<<endl;  
 }  
 if (n2<n1) {  
 log<<"Cчитано меньше точек чем предполагалось: "<<n2<<endl;  
 double \*\*new\_a=new double\*[n2];  
 for(int i=0;i<n2;i++) {  
 new\_a[i]=new double [2];  
 }  
 for(int i=0;i<n2;i++) {  
 new\_a[i][0]=a[i][0];  
 new\_a[i][1]=a[i][1];  
 }  
 for (int i=0;i<n1;i++) {  
 delete []a[i];  
 }  
 delete []a;  
 n1=n2;  
 return new\_a;  
 }  
 else {  
 return a;  
 }  
}  
  
bool peresek(const double pr1[4], const double pr2[4]) {  
 double x1 = pr1[0], y1 = pr1[1], x2 = pr1[2], y2 = pr1[3];  
 double x3 = pr2[0], y3 = pr2[1], x4 = pr2[2], y4 = pr2[3];  
 double det = (x2 - x1) \* (y4 - y3) - (y2 - y1) \* (x4 - x3);  
 double t = ((x3 - x1) \* (y4 - y3) - (y3 - y1) \* (x4 - x3)) / det;  
 double u = ((x3 - x1) \* (y2 - y1) - (y3 - y1) \* (x2 - x1)) / det;  
 if (t >= 0 && t <= 1 && u >= 0 && u <= 1) {  
 return true;  
 }  
 return false;  
}  
  
#define EPSILON 1e-10  
  
bool are\_parallel(double pr1[4], double pr2[4]) {  
 double x1 = pr1[0], y1 = pr1[1];  
 double x2 = pr1[2], y2 = pr1[3];  
 double x3 = pr2[0], y3 = pr2[1];  
 double x4 = pr2[2], y4 = pr2[3];  
 return fabs((x2 - x1) \* (y4 - y3) - (y2 - y1) \* (x4 - x3)) < EPSILON;  
}  
  
  
bool check\_trp(double pr1[4], double pr2[4], double pr3[4], double pr4[4]) {  
 bool parallel1 = are\_parallel(pr1, pr3);  
 bool parallel2 = are\_parallel(pr2, pr4);  
 if (!(parallel1 || parallel2)) {  
 return false;  
 }  
 if (parallel1 && parallel2) return false;  
 if (peresek(pr1, pr3) || peresek(pr2, pr4)) {  
 return false;  
 }  
 return true;  
}  
  
int \*bouble\_sort(int \*a) {  
 int \*na=new int[4];  
 for(int i=0;i<4;i++) {  
 na[i]=a[i];  
 }  
 for (int i=0;i<4;i++) {  
 for(int j=0;j<4-i-1;j++) {  
 if(na[j]>na[j+1]) {  
 int t=na[j];  
 na[j]=na[j+1];  
 na[j+1]=t;  
 }  
 }  
 }  
 return na;  
}  
  
bool m\_ravn(int \*a1,int \*a2) {  
 for(int i=0;i<4;i++) {  
 if(a1[i]!=a2[i]) return false;  
 }  
 return true;  
}  
  
bool check\_app(int \*\*trap,int n, int value[4]) {  
 for (int i=0;i<n;i++) {  
 int \*t1=bouble\_sort(trap[i]);  
 int \*t2 = bouble\_sort(value);  
 if (m\_ravn(t1,t2)) {  
 delete []t1;  
 delete []t2;  
 return false;  
 }  
 }  
 return true;  
}  
  
int \*\* append\_trap(int \*\*trap,int n, int value[4]) {  
 int \*\*new\_trap = new int\* [n+1];  
 for (int i=0;i<n+1;i++) {  
 new\_trap[i] =new int [4];  
 }  
 for (int i=0;i<n;i++) {  
 new\_trap[i][0]=trap[i][0];  
 new\_trap[i][1]=trap[i][1];  
 new\_trap[i][2]=trap[i][2];  
 new\_trap[i][3]=trap[i][3];  
 }  
 new\_trap[n][0]=value[0];  
 new\_trap[n][1]=value[1];  
 new\_trap[n][2]=value[2];  
 new\_trap[n][3]=value[3];  
 return new\_trap;  
}  
  
  
void ABC(const double pr[4], double &a, double &b, double &c) {  
 double x1 = pr[0], y1 = pr[1];  
 double x2 = pr[2], y2 = pr[3];  
 a = y2 - y1;  
 b = x1 - x2;  
 c = x2 \* y1 - x1 \* y2;  
}  
  
int znak(double a, double b, double c, const double p[2]) {  
 if ((a \* p[0] + b \* p[1] + c) > 0) return 0;  
 if ((a \* p[0] + b \* p[1] + c) < 0) return 1;  
 return 2;  
}  
  
  
bool in\_area(double a[2], double pr1[4], double pr2[4], double pr3[4], double pr4[4]) {  
 double a1, b1, c1;  
 ABC(pr1, a1, b1, c1);  
 double a2, b2, c2;  
 ABC(pr2, a2, b2, c2);  
 double a3, b3, c3;  
 ABC(pr3, a3, b3, c3);  
 double a4, b4, c4;  
 ABC(pr4, a4, b4, c4);  
  
 int sign1 = znak(a1, b1, c1, a);  
 int sign2 = znak(a2, b2, c2, a);  
 int sign3 = znak(a3, b3, c3, a);  
 int sign4 = znak(a4, b4, c4, a);  
  
 int t[4] ={sign1,sign2,sign3,sign4};  
 int ct0=0, ct1=0;  
 for (int i=0;i<4;i++) {  
 if (t[i]==0) ct0++;  
 if (t[i]==1) ct1++;  
 }  
 if (ct0>0 && ct1==0) return true;  
 if (ct0==0 && ct1>0) return true;  
 return false;  
}  
  
int \*\* trapezium(int &c, double \*\*a,int n,ofstream &log) {  
 int \*\*trap=nullptr;  
 c=0;  
 for (int i=0;i<n;i++) {  
 for(int j=0;j<n;j++) {  
 if (j == i) continue;  
 for (int h=0;h<n;h++) {  
 if (h == i || h == j) continue;  
 for (int k=0;k<n;k++) {  
 if (k == i || k == j || k == h) continue;  
 double pr1 [4]= {a[i][0],a[i][1],a[j][0],a[j][1]};  
 double pr2 [4]= {a[j][0],a[j][1],a[h][0],a[h][1]};  
 double pr3 [4]= {a[h][0],a[h][1],a[k][0],a[k][1]};  
 double pr4 [4]= {a[k][0],a[k][1],a[i][0],a[i][1]};  
 if (check\_trp(pr1,pr2,pr3,pr4)) {  
 int t[4]={i,j,h,k};  
 if (check\_app(trap,c,t) ){  
 log<<"Составили трапецию из точек:"<<i<<","<<j<<","<<h<<","<<k<<endl;  
 trap=append\_trap(trap,c,t);  
 c++;  
 }  
 else {  
 log<<"Комбинация "<<i<<","<<j<<","<<h<<","<<k<<" уже записана"<<endl;  
 }  
 }  
 else log<<"Трапеция из точек "<<i<<","<<j<<","<<h<<","<<k<<" не получилась"<<endl;  
 }  
 }  
 }  
 }  
 return trap;  
}  
double\*\* append\_dot(double\*\* mdint, double d[2], int n) {  
  
 double\*\* new\_mdint = new double\*[n+1];  
 for (int i = 0; i < n+1; i++) {  
 new\_mdint[i] = new double[2];  
 }  
  
 for (int i = 0; i < n; i++) {  
 new\_mdint[i][0] = mdint[i][0];  
 new\_mdint[i][1] = mdint[i][1];  
 }  
  
 new\_mdint[n][0] = d[0];  
 new\_mdint[n][1] = d[1];  
  
 for (int i = 0; i < n; i++) {  
 delete[] mdint[i];  
 }  
 delete[] mdint;  
  
 return new\_mdint;  
}  
  
  
int \* dots\_in\_trapezion(double \*\*a,int c,int \*\*trap,int n,double \*\*\* mdint , ofstream& log) {  
 int \*mcount\_dots = new int[c]();  
 for (int i=0;i<c;i++) {  
 log<<"Трапеция №"<<i+1<<endl;  
 double pr1 [4]= {a[trap[i][0]][0],a[trap[i][0]][1],a[trap[i][1]][0],a[trap[i][1]][1]};  
 double pr2 [4]= {a[trap[i][1]][0],a[trap[i][1]][1],a[trap[i][2]][0],a[trap[i][2]][1]};  
 double pr3 [4]= {a[trap[i][2]][0],a[trap[i][2]][1],a[trap[i][3]][0],a[trap[i][3]][1]};  
 double pr4 [4]= {a[trap[i][3]][0],a[trap[i][3]][1],a[trap[i][0]][0],a[trap[i][0]][1]};  
 int count\_dots=0;  
 log<<"В трапецию из точек 1-("<<a[trap[i][0]][0]<<","<<a[trap[i][0]][1]<<") 2-("<<a[trap[i][1]][0]<<","<<a[trap[i][1]][1]<<") 3-("<<a[trap[i][2]][0]<<","<<a[trap[i][2]][1]<<") 4-("<<a[trap[i][3]][0]<<","<<a[trap[i][3]][1]<<")"<<endl;  
 for (int j=0;j<n;j++) {  
 if (j==trap[i][0] ||j==trap[i][1] ||j==trap[i][2]||j==trap[i][3]) continue;  
 if (in\_area(a[j],pr1,pr2,pr3,pr4)) {  
 log<<"Вошла точка ("<<a[j][0]<<","<<a[j][1]<<")\n";  
 mdint[i]=append\_dot(mdint[i],a[j],count\_dots);  
 count\_dots++;  
 }  
 else log<<"Точка ("<<a[j][0]<<","<<a[j][1]<<") не вошла"<<endl;  
 }  
 log <<"В трапецию вошло "<<count\_dots<<" точек\n"<<endl;  
 mcount\_dots[i]=count\_dots;  
 }  
 return mcount\_dots;  
}  
  
  
void output\_dots\_in\_trapezion(double \*\*a, int n,int \*\*trap,int c,int \*mcount\_dots,int &md,double \*\*\* mdia,ofstream &log,ofstream &fout) {  
 fout<<"Считано "<<n<<" точек"<<endl;  
 for (int i=0;i<n;i++) {  
 fout<<"("<<a[i][0]<<","<<a[i][1]<<")"<<endl;  
 }  
 fout<<endl;  
 fout<<"Из "<<n<<" точек можно составить "<<c<<" трапеций\n"<<endl;  
 log<<"Максимальное вхождение точек "<<md<<endl;  
 for (int i=0;i<c;i++) {  
 fout<<"В трапеции "<<i+1<<" состоящией из точек:"<<endl;  
 fout<<"Точка №1 x:"<<a[trap[i][0]][0]<<" y:"<<a[trap[i][0]][1]<<endl;  
 fout<<"Точка №2 x:"<<a[trap[i][1]][0]<<" y:"<<a[trap[i][1]][1]<<endl;  
 fout<<"Точка №3 x:"<<a[trap[i][2]][0]<<" y:"<<a[trap[i][2]][1]<<endl;  
 fout<<"Точка №4 x:"<<a[trap[i][3]][0]<<" y:"<<a[trap[i][3]][1]<<endl;  
 fout<<"находится "<<mcount\_dots[i]<<" точек"<<endl;  
 for (int j=0;j<mcount\_dots[i];j++) {  
 fout<<"("<<mdia[i][j][0]<<","<<mdia[i][j][1]<<")"<<endl;  
 }  
 if (md<mcount\_dots[i]) {md=mcount\_dots[i];log<<"Максимальное вхождение точек "<<md<<endl;}  
 fout<<endl;  
 }  
 fout<<endl<<endl;  
}  
void output\_max\_dots\_in\_trapezion(double \*\*a, int \*\*trap,int c, const int \*mcount\_dots,int md,double \*\*\*mdia,ofstream &fout) {  
 fout<<"----------------------------------------------------------------------"<<endl;  
 fout<<"Максимальное кол-во заданых точек "<<md<<" имеют трапеции\n"<<endl;  
 int nom=0;  
 for (int i=0;i<c;i++) {  
 if (mcount\_dots[i]==md) {  
 nom++;  
 fout<<"Трапеция "<<nom<<" ("<<i+1<<")"<<" состоящая из точек:"<<endl;  
 fout<<"Точка №1 x:"<<a[trap[i][0]][0]<<" y:"<<a[trap[i][0]][1]<<endl;  
 fout<<"Точка №2 x:"<<a[trap[i][1]][0]<<" y:"<<a[trap[i][1]][1]<<endl;  
 fout<<"Точка №3 x:"<<a[trap[i][2]][0]<<" y:"<<a[trap[i][2]][1]<<endl;  
 fout<<"Точка №4 x:"<<a[trap[i][3]][0]<<" y:"<<a[trap[i][3]][1]<<endl;  
 for (int j=0;j<mcount\_dots[i];j++) {  
 fout<<"("<<mdia[i][j][0]<<","<<mdia[i][j][1]<<")"<<endl;  
 }  
 fout<<endl;  
 }  
 }  
 fout<<"Трапеций с максимумом точек внутри: "<<nom<<endl;  
}  
  
int main() {  
 system("chcp 1251>0");  
 ifstream fin("input.txt");  
 ofstream log("protocol.txt");  
 ofstream fout("output.txt");  
 titulnik(fout);  
 if (!fin.is\_open()) exit(1);  
 int n;  
 double \*\*a=input\_dots(fin,n,log);  
 if (n==0) {log<<"!!!!!!!!!!!Нет данных для обработки"<<endl; fout<<"!!!!!!!!!!!Нет данных для обработки"<<endl;exit(0);}  
 log<<"\n\n------------Данные считаны------------\n\n\n"; fin.close();  
 int c;  
 int \*\*trap=trapezium(c,a,n,log);  
 if (c==0){log<<"!!!!!!!!!!!!Нет трапеций для обработки"<<endl;fout<<"!!!!!!!!!!!!Нет трапеций для обработки"<<endl;exit(0);}  
 log<<"\n\n------------Трапеции составлены------------\n\n\n";  
 double \*\*\*mdots\_in\_trapezium = new double\*\*[c];  
 for (int i =0; i<c; i++) {mdots\_in\_trapezium[i]=nullptr;}  
 int \*mcount\_dots=dots\_in\_trapezion(a,c,trap,n,mdots\_in\_trapezium,log);  
 log<<"\n\n------------Точки внутри трапеций подсчитаны------------\n\n\n";  
 int max\_count\_dots=0;  
 output\_dots\_in\_trapezion(a,n,trap,c,mcount\_dots,max\_count\_dots,mdots\_in\_trapezium,log,fout);  
 log<<"\n\n------------Максимум точек подсчитан------------\n\n\n";  
 output\_max\_dots\_in\_trapezion(a,trap,c,mcount\_dots,max\_count\_dots,mdots\_in\_trapezium,fout);  
 log.close();  
 fout.close();  
 for (int i=0;i<n;i++) {  
 delete []a[i];  
 }  
 delete []a;  
 for (int i=0;i<c;i++) {  
 delete []trap[i];  
 for (int j=0;j<mcount\_dots[i];j++) {  
 delete []mdots\_in\_trapezium[i][j];  
 }  
 delete []mdots\_in\_trapezium[i];  
 }  
 delete []trap;  
 delete []mcount\_dots;  
 return 0;  
}

# Результаты работы программы

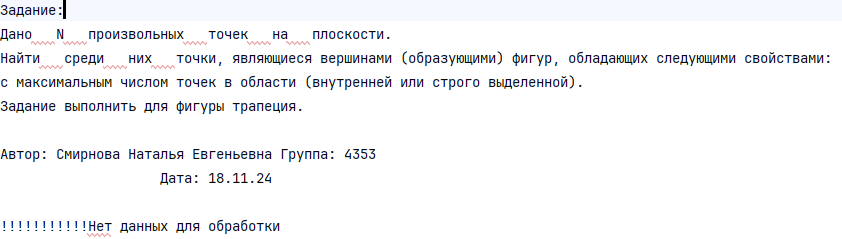
Для контрольного примера



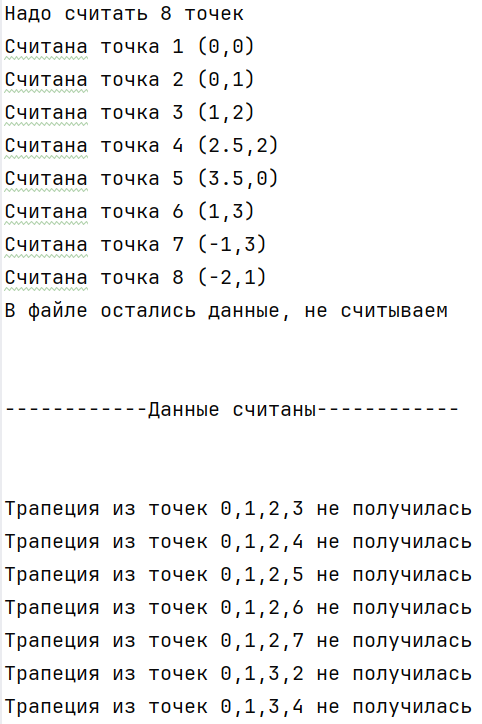
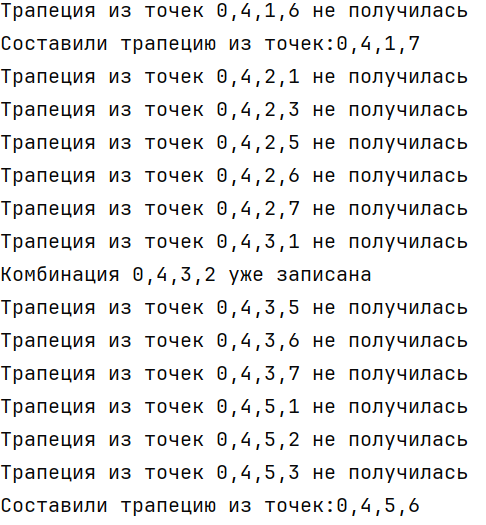
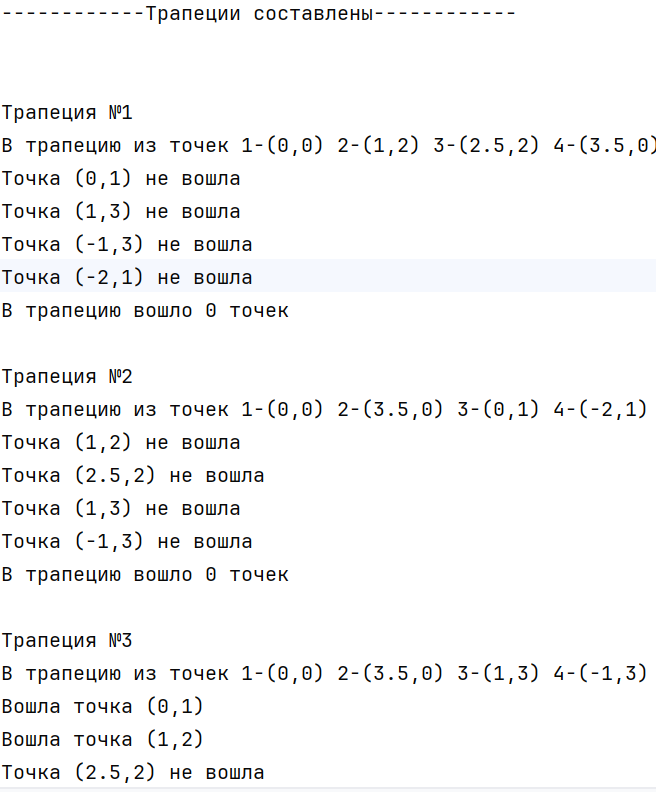
Если из данных точек нельзя составить ни одной трапеции

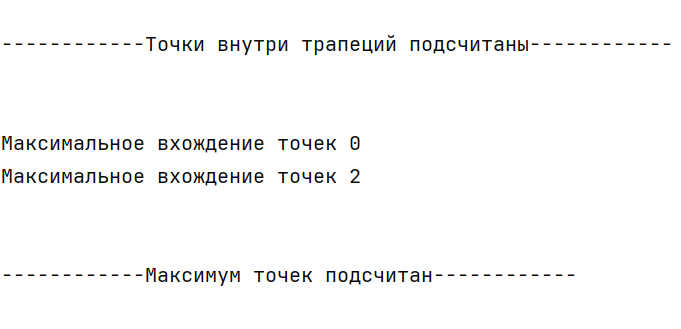


Если в файле нет данных



Пример работы протокола



# Вывод по проделанной работе

В ходе выполнения курсовой работы была разработана программа на языке C++ для анализа строк матрицы. Работа включала в себя изучение и практическое применение базовых и продвинутых конструкций языка программирования C++, связанных с двумерными массивами, указателями и функциями.