**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САРПР**

КУРСОВАЯ РАБОТА

**по дисциплине «Программирование»**

Тема: Программа на C++ для геометрических задач

|  |  |  |
| --- | --- | --- |
| Студент гр. 4352 |  | Редкокаша Р. Ю. |
| Преподаватель |  | Калмычков В. А. |

Санкт-Петербург

2024

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Редкокаша Р. Ю.

Группа 4352

Тема работы: программа на C++ для геометрических задач

Исходные данные: Дан файл, содержащий в первой строке количество точек, в остальных координаты точек. Требуется найти точки, которые образуют требуемую фигуры, содержащие максимальное количество точек.

Содержание пояснительной записки:

1. Введение
2. Оглавление
3. Исходная формулировка задания
4. Определение неясностей
5. Математическая постановка
6. Устройство файла
7. Контрольный пример
8. Ограничения
9. Организация пользовательского интерфейса
10. Реализация ввода-вывода
11. Описание алгоритма
12. Устройство функций
13. Текст программы
14. Результаты работы программы
15. Заключение

|  |  |  |
| --- | --- | --- |
| Предполагаемый объем пояснительной записки:  Не менее \_\_ страниц. | | |
| Дата выдачи задания: 18.11.2024 | | |
| Дата сдачи реферата: \_\_.12.2024 | | |
| Дата защиты реферата: \_\_.12.2024 | | |
| Студент |  | Редкокаша Р. Ю. |
| Преподаватель |  | Калмычков В. А. |

АННОТАЦИЯ

Для решение поставленной задачи использовались стандартные библиотеки C++, такие как iostream, fstream, iomanip. Ввод – вывод осуществлялся через файлы «*Inp.txt*», «*Procces.txt*», «*Protocol.txt*». Для хранения данных из входного файла использовалась динамическая память. Найденные фигуры сохранялись в файл «*Temp.da*t». Для решения задачи использовались геометрические правила и уравнения, например: на основе двух точек считалось: уравнения прямых, вектора, отрезки. Производилось скалярное умножение, счёт длины и проверка перпендикулярности. Находилась координата *y* по уравнению прямой. Определялась принадлежность точек фигурам и пересечения прямых.

SUMMARY

Standard C++ libraries such as iostream, fstream, and iomanip were used to solve this problem. I/O was done through files "Inp.txt ", "Procces.txt ", "Protocol.txt ". Dynamic memory was used to store data from the input file. The found shapes were saved to the Temp.dat file. Geometric rules and equations were used to solve the problem, for example: Based on two points, it was calculated: equations of straight lines, vectors, and segments. Scalar multiplication, length counting, and perpendicular checking were performed. The y coordinate was located according to the equation of the straight line. The points were determined by the shapes of the intersection of straight lines.

Оглавление

[Введение 2](#_Toc185515784)

[1. Исходная формулировка задания 2](#_Toc185515785)

[2. Определение неясностей 2](#_Toc185515786)

[3. Математическая постановка 2](#_Toc185515787)

[4. Устройство файла 2](#_Toc185515788)

[5. Контрольный пример 2](#_Toc185515789)

[6. Ограничения 2](#_Toc185515790)

[7. Организация пользовательского интерфейса 2](#_Toc185515791)

[8. Реализация ввода-вывода 2](#_Toc185515792)

[9. Внутренние представление данных 2](#_Toc185515793)

[10. Устройство функций 2](#_Toc185515794)

[11. Описание алгоритма 2](#_Toc185515795)

[12. Текст программы 2](#_Toc185515796)

[13. Результат работы программы 2](#_Toc185515797)

[Заключение 2](#_Toc185515798)

Введение

**Цель работы:** написать программу, получающая на вход координаты точек и их число из файла, решающая поставленную задачу, на основе полученной информации, выводящая результат в один файл, а промежуточные действия в другой.

# Исходная формулировка задания

Даны N точек на плоскости. Найти среди них точки, являющиеся вершинами фигуры (Рисунок 1), содержащей максимальное число заданных точек.

Рисунок

# Определение неясностей

Не требуется проводить доскональный мат анализ. Разрешается некоторая погрешность в вычислениях. Если для точки не указана одна из координат – пропустить её. Если указана третья координата у точки – использовать только две. Если заявленное число точек больше реального – прировнять к реальному, если отрицательное – к нулю. Могут существовать точки с одинаковыми координатами, считать их разными. Толщина прямоугольника (*DEJG*) может быть нулевая.

# Математическая постановка

Вектор прямой:

Длина прямой

Перпендикулярность:

Уравнение прямой:

На основе считанных из файла данных строится требуемая фигура. Для этого:

1. Берутся две случайные точки (A1 и B1);
2. Ищется такая точка D, чтобы был тупым ;
3. Ищется точка *E*, чтобы ,

Рисунок

1. , ;
2. , , ;
3. , ;
4. , ;
5. , ;
6. , ;

# Устройство файла

В первой строке передаётся количество вводимых точек. В последующих координаты этих точек. Координата для точки указывается в виде двух вещественных чисел для *X* и *Y* координат соответственно. Для каждой точки координаты указываются в новой строке. Вывод данных промежуточных вычислений происходит в файл «*Protocol.txt*». Точки выводятся в скобках «()», вектор – «{}», позиция в списке «[]». Выводятся данные о векторах, прямых, отрезках, их длинах, принадлежность точки всех фигур и т. д. В файл «*Procces.txt*» происходит вывод только ответа на задачу – фигуры и точки, которые они содержат.

# Контрольный пример

-9.2 -21

3.2 -5.666666667

24 10.25

-11 7

-18.5 2

-2.5 36

8.5 13.5

-2 -1.2

-4.333333333 -12

-42 7

12.5 2

46 49

-4 -35

5 -9.4

-6.4 -3.6

-41 -11

1.6 12.5

-8 9.5

9.5 10.25

-7 -47

4.6 6.2

20 2.5

10.75 20

-12.33333333 -27

9.2 10.75

-11 -9.2

-10 15.66666667

-6 29

-7.75 13

15 -5

-4 11.33333333

5.333333333 1.4

-9.5 -2.25

-11.5 -6.4

-15.5 10

6.333333333 1

-5 -25

14.66666667 -13

-2.5 -23

3.4 -1.5

48 -1

17 -16

-12.33333333 6

-20.5 -38

-1.8 -11.5

24 6.25

-0.25 -15.5

2 0

-9.2 -2

-1.5 1.5

-20 0

-10 20

-10 10

10 10

10 20

20 0

10 -20

10 -10

-10 -10

-10 -20

-18.507 14.423

-18.805 25.299

-15.849 21.036

-9.422 25.492

-12.378 29.756

-2.297 25.663

-1.998 14.786

-4.955 19.05

-11.381 14.593

-8.425 10.33

16.099 11.754

7.496 25.006

3.909 17.968

-12.109 26.132

-8.521 33.171

-24.299 32.346

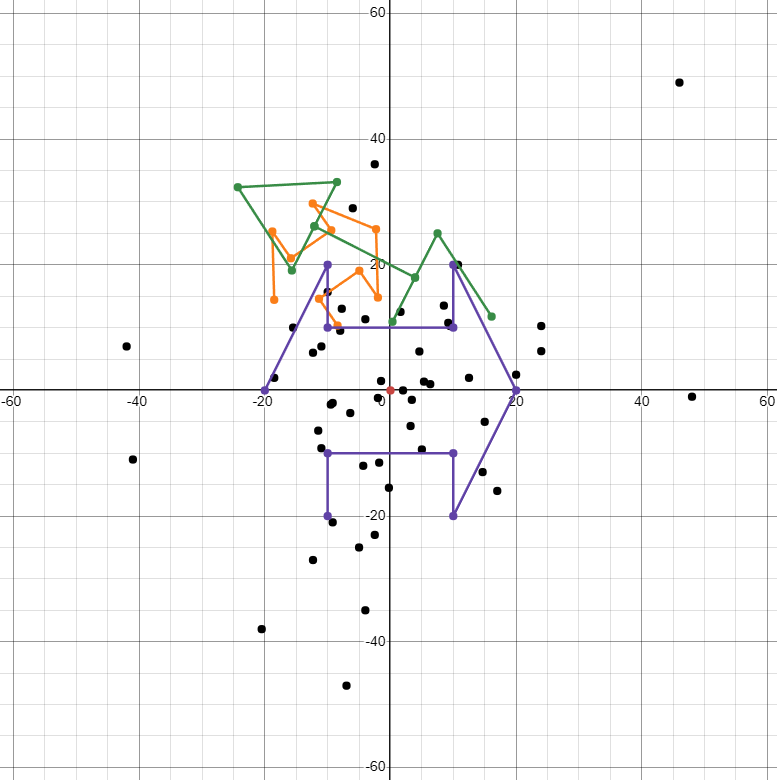
-15.696 19.094

-12.109 26.132

3.909 17.968

0.321 10.9

|  |  |  |
| --- | --- | --- |
| Фигура 1 (оранжевая) | Фигура 2 (зелёная) | Фигура 3 (фиолетовая) |
| -18.507 14.423  -18.805 25.299  -15.849 21.036  -9.422 25.492  -12.378 29.756  -2.297 25.663  -1.998 14.786  -4.955 19.05  -11.381 14.593  -8.425 10.33 | 16.099 11.754  7.496 25.006  3.909 17.968  -12.109 26.132  -8.521 33.171  -24.299 32.346  -15.696 19.094  -12.109 26.132  3.909 17.968  0.321 10.929 | -20 0  -10 20  -10 10  10 10  10 20  20 0  10 -20  10 -10  -10 -10  -10 -20 |



# Ограничения

Размер вводимых чисел ограничен пределами типа данных double (1.7E +/- 308).

# Организация пользовательского интерфейса

Ввод информации происходит через файл «Inp.txt», а вывод «Procces.txt» - итоговый ответ, «Protocol.txt» - промежуточные вычисления и прочее.

*D* = , *C* – символ, *I* = *d\_\_\_d*

|  |  |
| --- | --- |
| **Err** | **Protocol.txt** |
| 1 | Error: Файла нет! |
| 2 | Error: Файл пуст! |
| 3 | Error: Размер меньше 0, не указан или равен 0! |
| 4 | Error: Размер приравнен к реальному, так как меньше указанного! |
| **Prot** | **Protocol.txt** |
| 1 | Чтение размера: |
| 2 | Расчёт реального размера: |
| 3 | Реальный размер переназначен на: *I* |
| 4 | Считан символ: *C* |
| 5 | Считано число: *D* |
| 6 | Файл сброшен. |
| 7 | Пропуск размера: |
| 8 | Запись в массив: |
| 9 | В позиции записано число: *D* |
| 10 | Отрезок с координатами точки 1 (*D, D*) точки 2 (*D, D*) преобразован в вектор {*D, D*} |
| 11 | Длина вектора {*D, D*} = *D* |
| 12 | Вектор {*D, D*} (не) перпендикулярен вектору {*D, D*} |
| 13 | Проверка наличия точки *I* в фигуре: |
| 14 | Сравнение с *I*: |
| 15 | Нет в фигуре / Есть в фигуре |
| 16 | Значение *Y* для прямой (*D, D*) (*D, D*) на координате оси X = *D*: *D* |
| 17 | Точка (*D, D*) между Y фигуры (*D, D*) (*D, D*) … (не) находится |
| 18 | Точка с X координатой *D* между прямой (*D, D*) (*D, D*) (не) находится |
| 19 | Проверка принадлежности точки (*D, D*) фигуре (*D, D*) (*D, D*) …: |
| 20 | Точка (*D, D*) (не) принадлежит фигуре. |
| 21 | Прямая (*D, D*) (*D, D*) прямую (*D, D*) (*D, D*) (не) пересекает. |
| 22 | Проверка на повтор фигуры: №*I* |
| 23 | Фигура новая. / Фигура уже найдена. |
| 24 | Сохранение фигуры. |
| 25 | Максимум: *I*. Поиск фигуры: |
| 26 | Фигура и точки, которые она содержит №*I*: |
| 27 | Счёт точек в фигуре: |
| 28 | Поиск точки №*I*: |
| 29 | Проверка точки *I*: |
| 30 | Точка *I* установлена как №*I*. |
| 31 | Выход из цикла для точки №*I*. |
| 32 | Возврат к точке №*I*. |
| 33 | Поиск фигуры: |
| **Proc** | **Protocol.txt, Procces.txt** |
| 1 | Найдена фигура: |
| 2 | Точка №*I*: (*D, D*) |
| 3 | Содержит *I* точек. |

# Реализация ввода-вывода

|  |  |  |  |
| --- | --- | --- | --- |
| Библиотека | Ввод | Вывод | Дробная часть |
| iomanip |  |  | fstream.setf(ios::fixed);  fstream.precision(7); |
| fstream | ios::in >>, .write() | ios::out <<, .read() |  |

# Внутренние представление данных

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Тип** | **Назначение** |
| Fproc | fstream | Вывод данных отладки |
| Fprot | Вывод решения |
| F | Ввод исходных данных |
| FT | Хранение временных данных |
| MaxC | size\_t | Хранение данных о количестве точек в фигуре, которая хранит максимальное количество оных |
| NumOfFig | Хранение данных о количестве найденных фигур |
| S | Количество заданных точек |
| A | double\*\* | Хранение заданых точек |
| Eps | const double | Данные о минимально допустимой погрешности |

# Устройство функций

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя** | **Параметры** | | | |
| **Входные** | **Выходные** | **Модифицируемые** | **Транслируемые** |
| OutMain | const size\_t& Value, const char\* EndStr |  | fstream FProt, FProc |  |
| const double& Value, const char\* EndStr |
| double\*& Value, const char\* Skoba, const char\* EndStr |
| const char\* TXT |
| OutEr | const char\* TXT |
| OutProt | const size\_t& Value, const char\* EndStr | fstream FRoc |
| const double& Value, const char\* EndStr |
| double\*& Value, const char\* Skoba, const char\* EndStr |
| const char& C, const char\* EndStr |
| const char\* TXT |
| ReadOnlyDouble |  | true/false | fstream F, double& X, char& C |  |
| CalcSize |  | size\_t S |  |  |
| Read | double\*\* A, const size\_t& S |  |  | char C, double X, size\_t TT, T |
| Max |  |  |  | size\_t& A, size\_t& B |
| GetVector |  | double\* A[2] |  | double\*& A1, double\*& A2, double\* A[2] |
| Len |  | double Out |  | double\*& A1, double\*& A2 |
| EqualLen |  | true/false |  | double\*& A1, double\*& A2, double\*& B1, double\*& B2 |
| IsNormal |  | bool Out |  | double\*& A1, double\*& A2, double\*& B1, double\*& B2 |
| CanUse | size\_t\*& B, const size\_t& X, const short& Pos | true/false |  | size\_t& B |
| Spec3 |  | true/false |  | double\*& A1, double\*& A2, double\*& A3 |
| GetYfromX | double\*& A1, double\*& A2, double& X | double Z |  |  |
| BetweenY |  | true/false |  | double\*& A1, double\*& A2, double\*& A3, double\*& A4,  double& B |
|  |  | double\*& A1, double\*& A2, double\*& A3, double\*& B |
| BetweenX | double\*& A1, double\*& A2, double X | true/false |  |  |
| Prinadlejit |  | bool b |  | double\*& A1, double\*& A2, double\*& A3, double\*& B |
|  |  | double\*& A1, double\*& A2, double\*& A3, double\*& A4, double\*& B |
| Peresec |  | true/false |  | double\*& A1, double\*& A2, double\*& A3, double\*& B  double X |
| IsCorrect | double\*\*& A,  size\_t\*& B | true/false |  | double\*& A, size\_t& B, const short& Pos, size\_t& i |
| NotInSaveF |  | fstream FT |  | size\_t\*& B |
| SaveF | size\_t\*& B | fstream FT | size\_t NumOfFig |  |
| OutFig | double\*\*& A, size\_t\*& B |  |  | double\*& A, size\_t& B |
| OutFigMain | double\*\*& A, size\_t\*& B |  |  | double\*& A, size\_t& B |
| FindMax | double\*\*& A, size\_t\* B, size\_t S | fstream FT |  | double\*& A, size\_t& B |
| Schot | double\*\*& A, const size\_t& S | size\_t B | size\_t B | double\*& A, size\_t& B,  size\_t\*& B |
| SubProcces |  |  |  | double\*\*& A, size\_t\* B, const size\_t& S, short Pos |
| Procces |  |  |  | double\*\*& A, size\_t& S,  size\_t B[11] |

|  |  |
| --- | --- |
| **Имя** | **Внешние изменения** |
| OutMain | Выводит получаемые данные в файлы «Procces.txt» и «Protocol.txt» |
| OutEr | Выводит сообщение об ошибке в файлы «Procces.txt» и «Protocol.txt» |
| OutProt | Выводит данные в файл «Protocol» |
| ReadOnlyDouble | Считывает информацию из файла «Inp.txt» |

|  |  |
| --- | --- |
| **Функция** | **Описание** |
| OutMain  OutProc | Получает целочисленное значение и последовательность символов. Выводит их в файл. |
| Получает вещественное значение и последовательность символов. Выводит их в файл. |
| Получает вещественные координаты, скобки, в которые их поместить и последовательность символов. Выводит данные в файл. |
| Получает последовательность символов, выводит её в файл. |
|  |
| OutProc | Получает символ и последовательность символов. Выводит их в файл. |
| OutEr | Получает последовательность символов, выводит её в файл. |
| Read-OnlyDouble | Получает ссылки на вещественное число (*X*) и на символ (*C*). Помещает символы из файла («Inp.txt») на место полученного символа (*C*), до тех пор, пока не считает начало числа, затем помещает число из файла на место полученного числа (*X*). |
| CalcSize | Считает количество чисел в файле, опираясь на указанное в начале файла и реальное. |
| Read | Получает на вход ссылку на динамический массив двумерных массивов (*A*) и ссылку на его размер. Переносит числа из файла в динамический массив (*A*) |
| Max | Принимает ссылки на натуральные числа и возвращает наибольшее. |
| GetVector | Получат на вход 2 ссылки на указатели на пары вещественных координат. Возвращает вектор, образованный из прямой, которой принадлежат эти точки. |
| Len | Получат на вход 2 ссылки на указатели на пары вещественных координат. Преобразует их в вектор и ищет длину. |
| EqualLen | Получат на вход 4 ссылки на указатели на пары вещественных координат. Ищет длину прямых, которые образуют 1 и 2, 3 и 4 точки соответственно. Сравнивает эти длины. |
| IsNormal | Получат на вход 4 ссылки на указатели на пары вещественных координат. Преобразует точки 1 и 2, 2 и 3 в 2 вектора, скалярно их умножает. Сравнивает результат с 0 |
| CanUse | Получает на вход ссылку на указатель на массив (*B*), неизменяемую ссылку на вещественное число (*X*). Проверяет наличие этого числа в массиве. |
| Spec3 | Получат на вход 3 ссылки на указатели на пары вещественных координат. Проверяет, чтобы квадрат отрезка из координат данных 1, 2 был больше квадратов 1, 3 и 2, 3. |
| GetYFromX | Получат на вход 2 ссылки на указатели на пары вещественных координат и ссылку на вещественное число (*X*). Возвращает координату X, которая принадлежит прямой, образованной из 2-х переданных координат. |
| BetweenY | Получат на вход 5(4) ссылок на указатели на пары вещественных координат. Проверяет нахождение между Y координатами фигуры составленной из 1-4(1-3) точки 5(4) . |
| BetweenX | Получат на вход 2 ссылки на указатели на пары вещественных координат и ссылку на вещественную координату оси X. Проверяет находится ли эта точка, между X координатами отрезка, образованного парами координат. |
| Prinadlejit | Получат на вход 5(4) ссылок на указатели на пары вещественных координат. Проверяет принадлежность 5(4) точки фигуре, составленной из 1-4(1-3). |
| Peresec | Получат на вход 4 ссылок на указатели на пары вещественных координат. Проверяет: не пересекает ли прямая 1-2 прямую 1-4. |
| IsCorrect | Получает на вход ссылку на динамический массив пар вещественных координат (*A*), ссылку на указатель на массив id точек фигуры (*B*), ссылку на номер искомой точки в фигуре (*Pos*), ссылку на id сверяемой точки (*i*). Проверяет, удовлетворяет ли условию искомой точки точка с id *i*. |
| NotInSaveF | Получает ссылку на указатель на массив фигуры (*B*). Проверяет: найдена ли такая фигура уже. |
| SaveF | Получает ссылку на указатель на массив фигуры (*B*). Сохраняет её. |
| OutFig | Получает на вход ссылку на динамический массив пар вещественных координат (*A*), ссылку на указатель на массив id точек фигуры (*B*). Выводит точки в файл «Protocol.txt». |
| OutFigMain | Получает на вход ссылку на динамический массив пар вещественных координат (*A*), ссылку на указатель на массив id точек фигуры (*B*). Выводит точки в файл «Procces.txt». |
| FindMax | Получает на вход ссылку на динамический массив пар вещественных координат (*A*), ссылку на указатель на массив id точек фигуры (*B*) и ссылку на количество точек (*S*). Ищет и выводит в файл точки фигуры, которая содержит максимальное количество точек и выводит сами эти точки. |
| Schot | Получает на вход ссылку на динамический массив пар вещественных координат (*A*), ссылку на указатель на массив id точек фигуры (*B*) и ссылку на количество точек (*S*). Возвращает количество точек, которые содержит эта фигура. |
| SubProcces | Получает на вход ссылку на динамический массив пар вещественных координат (*A*), ссылку на указатель на массив id точек фигуры (*B*), ссылку на количество точек(*S*), номер искомой точки в фигуре (*Pos*). Служит вспомогательной ссылкой-рекурсией для поиска всех возможных фигур. |
| Procces | Получает на вход ссылку на динамический массив пар вещественных координат (*A*), ссылку на количество точек(*S*). Активирует процессы ответственные за решение задачи. |

# Описание алгоритма

Программа считывает данные из файла. допускаются «мусорные» символы между числами (пробелы, перенос строки и табуляция). Ищутся все возможные фигуры, удовлетворяющие заданным в задании признакам, считается количество точек в них. Всё это сохраняется в временный файл. Находится все фигуры, содержащие максимальное число точек. Эти фигуры и точки, которые они содержат выводятся в файл «Procces.txt».















# Текст программы

#include "fstream"

#include "iomanip"

using namespace std;

fstream FProc("Procces.txt", ios::out), FProt("Protocol.txt", ios::out), F("Inp.txt", ios::in), FT("Temp.dat", ios::binary | ios::in | ios::out | ios::trunc);

size\_t MaxC = 0, NumOfFig = 0;

const double Eps = 10E-2;

const void OutMain(const size\_t& Value, const char\* EndStr) {

FProc << Value << EndStr;

FProt << "Вывод пользователю: " << Value << EndStr; return;}

const void OutMain(const double& Value, const char\* EndStr) {

FProc << Value << EndStr;

FProt << "Вывод пользователю: " << Value << EndStr; return;}

const void OutMain(double\*& Value, const char\* Skoba, const char\* EndStr) {

FProc << Skoba[0] << Value[0] << ", " << Value[1] << Skoba[1] << EndStr;

FProt << "Вывод пользователю: " << Skoba[0] << Value[0] << ", " << Value[1] << Skoba[1] << EndStr; return;}

const void OutMain(const char\* TXT) {

FProc << TXT;

FProt << "Вывод пользователю: " << TXT;return;}

const void OutProt(const size\_t& Value, const char\* EndStr) {

FProt << Value << EndStr; return;}

const void OutProt(const size\_t& Value1, size\_t& Value2, const char\* Skoba, const char\* EndStr) {

FProt << Skoba[0] << Value1 << ", " << Value2 << Skoba[1] << EndStr; return;}

const void OutProt(const double& Value, const char\* EndStr) {

FProt << Value << EndStr; return;}

const void OutProt(double\* Value, const char\* Skoba, const char\* EndStr) {

FProt << Skoba[0] << Value[0] << ", " << Value[1] << Skoba[1] << EndStr; return;}

const void OutProt(const char\* TXT) {

FProt << TXT; return;}

const void OutProt(const char& C, const char\* EndStr) {

FProt << C << EndStr; return;}

void OutEr(const char\* TXT) {

FProc << "\nError:\t" << TXT << "\n\n";

FProt << "\nError:\t" << TXT << "\n\n";}

bool ReadOnlyDouble(double& X, char& C) {

if (!F.eof()) {

F >> resetiosflags(ios::skipws);

F >> C;

OutProt("\tСчитан символ: ");

OutProt(C, "\n");

if (C >= '0' && C <= '9' || C == '-') {

F.seekg(-1, ios::cur);

F >> setiosflags(ios::skipws);

F >> X;

OutProt("\tСчитано число: ");

OutProt(X, "\n");

return true;}}return false;}

size\_t CalcSize() {

if (!F.is\_open()) {

OutEr("Файла нет!");

return 0;}

if (!F.get()) {

OutEr("Файл пуст!");

return 0;}

F.seekg(0, ios::beg);

char C = '!';

double S = 0;

size\_t RS = 0;

OutProt("Чтение размера:\n");

while (!F.eof())if (ReadOnlyDouble(S, C)) break;

if (S <= 0) {

OutEr("Размер меньше 0, не указан или равен 0!");return 0;}

double X;

size\_t T;

OutProt("Расчёт реального размера:\n");

while (!F.eof() && RS < S) {

T = 0;

while (C != '\n' && !F.eof()) if (ReadOnlyDouble(X, C) && T < 2) {

if (++T == 2){

RS++;

OutProt("\t\tРеальный размер переназначен на: ");OutProt(RS, "\n");}C = '!';}

if (S > RS) {OutEr("Размер приравнен к реальному, так как меньше указанного!\n");return RS; }

OutProt("Размер либо больше, либо равен указаному.\n");return S;}

void Read(double\*\* A, const size\_t& S) {

F.clear();

F.seekg(0, ios::beg);

OutProt("Файл сброшен.\n");

char C = '!';

double X;

OutProt("Пропуск размера:\n");

while (!ReadOnlyDouble(X, C));

size\_t T, TT = 0;

OutProt("Запись в массив:\n");

while (!F.eof() && TT < S) {T = 0;

while (C != '\n' && !F.eof()) if (ReadOnlyDouble(X, C) && T < 2) {

A[TT][T++] = X;

OutProt("\t\tВ позиции ");

OutProt(TT, T, "[]", " записано число: ");

OutProt(X, "\n");}

C = '!';

TT+= (T == 2) ? 1 : 0;}return;}

size\_t& Max(size\_t& A, size\_t& B) {return A > B ? A : B;}

double\* GetVector(double\*& A1, double\*& A2) {

double\* A = new double[2]{ (A2[0] - A1[0]), (A2[1] - A1[1]) };

OutProt("\t\t\tОтрезок с координатами точки 1 ");

OutProt(A1, "()", " и ");

OutProt("точки 2 ");

OutProt(A2, "()", " ");

OutProt("преобразован в вектор ");

OutProt(A, "{}", "\n");return A;}

double Len(double\*& A1, double\*& A2) {

double\* A = GetVector(A1, A2);

double Out = sqrt(pow(A[0], 2) + pow(A[1], 2));

OutProt("\t\t\tДлина вектора ");

OutProt(A, "{}", " = ");

OutProt(Out, "\n");

delete A;///////////////////////

return Out;}

bool EqualLen(double\*& A1, double\*& A2, double\*& B1, double\*& B2) {

return abs(Len(A1, A2) - Len(B1, B2)) <= Eps;}

bool IsNormal(double\*& A1, double\*& A2, double\*& B1, double\*& B2) {

double\* A = GetVector(A1, A2);

double\* B = GetVector(B1, B2);

bool Out = abs(A[0] \* B[0] + A[1] \* B[1]) <= Eps;

OutProt("\t\t\tВектор ");

OutProt(A, "{}", (Out ? " перпендикулярен" : " не перпендикулярен вектору "));

OutProt(B, "{}", "\n");

delete A;///////////////////////

delete B;;///////////////////////

return Out;}

bool CanUse(size\_t\*& B, const size\_t& X, const short& Pos) {

OutProt("\t\t\tПроверка наличия точки ");

OutProt(X, " в фигуре:\n");

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

OutProt("\t\t\t\tСравнение с ");

OutProt(B[i], ":\n");

if (B[i] == X) {

OutProt("\t\t\tЕсть в фигуре.\n");

return false;}}

OutProt("\t\t\tНет в фигуре.\n"); return true;}

bool Spec3(double\*& A1, double\*& A2, double\*& A3) {

return pow(Len(A1, A3), 2) + pow(Len(A2, A3), 2) <= pow(Len(A1, A2), 2) && Len(A1, A3) + Len(A2, A3) > Len(A1, A2);

}

double GetYfromX(double\*& A1, double\*& A2, double& X) {

OutProt("\t\t\t\tЗначение Y для прямой ");

OutProt(A1, "()", " ");

OutProt(A2, "()", " на координате оси X = ");

OutProt(X, ": ");

double K = 0;

if (A1[0] - A2[0]) K = (A1[1] - A2[1]) / (A1[0] - A2[0]);

double Z = K \* X + A1[1] - K \* A1[0];

OutProt(Z, "\n");return Z;}

bool BetweenY(double\*& A1, double\*& A2, double\*& A3, double\*& A4, double\*& B) {

double Y1 = GetYfromX(A1, A2, B[0]), Y2 = GetYfromX(A3, A4, B[0]);

OutProt("\t\t\t\tТочка ");

OutProt(B, "()", " между Y фигуры ");

OutProt(A1, "()", " ");

OutProt(A2, "()", " ");

if (A2 != A3) OutProt(A3, "()", " ");

OutProt(A4, "()", " ");

if (B[1] <= Y1 && B[1] >= Y2 || B[1] >= Y1 && B[1] <= Y2) { OutProt("находится.\n");return true;}

OutProt("не находится.\n");return false;}

bool BetweenY(double\*& A1, double\*& A2, double\*& A3, double\*& B) {return BetweenY(A1, A2, A2, A3, B);}

bool BetweenX(double\*& A1, double\*& A2, double X) {

OutProt("\t\t\t\tТочка с X координатой ");

OutProt(X, " между X координатами прямой ");

OutProt(A1, "()", " ");

OutProt(A2, "()", " ");

if (X >= A1[0] && X <= A2[0] || X <= A1[0] && X >= A2[0]) { OutProt("находится.\n");return true;}

OutProt("не находится.\n");return false;}

bool Prinadlejit(double\*& A1, double\*& A2, double\*& A3, double\*& B) {

bool b = false;

OutProt("\t\t\tПроверка принадлежности точки ");

OutProt(B, "()", " фигуре: ");

OutProt(A1, "()", " ");

OutProt(A2, "()", " ");

OutProt(A3, "()", ":\n");

if (BetweenX(A1, A2, B[0]) && BetweenX(A2, A3, B[0])) b = BetweenY(A1, A2, A3, B);

else if (BetweenX(A2, A3, B[0]) && BetweenX(A3, A1, B[0])) b = BetweenY(A2, A3, A1, B); else if (BetweenX(A3, A1, B[0]) && BetweenX(A1, A2, B[0])) b = BetweenY(A3, A1, A2, B);

OutProt("\t\t\t\tТочка с ");

OutProt(B, "()", (b ? " принадлежит фигуре.\n" : " не принадлежит фигуре.\n"));return b;}

bool Prinadlejit(double\*& A1, double\*& A2, double\*& A3, double\*& A4, double\*& B) {

bool b = false;

OutProt("\t\t\tПроверка принадлежности точки ");

OutProt(B, "()", " фигуре: ");

OutProt(A1, "()", " ");

OutProt(A2, "()", " ");

OutProt(A3, "()", " ");

OutProt(A4, "()", ":\n");

if (BetweenX(A1, A2, B[0]) && BetweenX(A2, A3, B[0])) b = BetweenY(A1, A2, A3, B);

else if (BetweenX(A2, A3, B[0]) && BetweenX(A3, A4, B[0])) b = BetweenY(A2, A3, A4, B);

else if (BetweenX(A3, A4, B[0]) && BetweenX(A4, A1, B[0])) b = BetweenY(A3, A4, A1, B);

else if (BetweenX(A4, A1, B[0]) && BetweenX(A1, A2, B[0])) b = BetweenY(A4, A1, A2, B);

else if (BetweenX(A1, A2, B[0]) && BetweenX(A3, A4, B[0])) b = BetweenY(A1, A2, A3, A4, B);

OutProt("\t\t\t\tТочка ");

OutProt(B, "()", (b ? " принадлежит фигуре." : " не принадлежит фигуре.\n"));return b;}

bool Peresec(double\*& A1, double\*& A2, double\*& A3, double\*& B) {

OutProt("\t\t\tПрямая ");

OutProt(A1, "()", " ");

OutProt(A2, "()", " ");

OutProt("прямую ");

OutProt(A3, "()", " ");

OutProt(B, "()", " ");

double K1 = (A1[1] - A2[1]) / (A1[0] - A2[0]), K2 = (A3[1] - B[1]) / (A3[0] - B[0]);

double X = (K1 \* A1[0] - A1[1] - (K2 \* A3[0] - A3[1])) / (K1 - K2);

if (BetweenX(A1, A2, X) && BetweenX(A3, B, X)) {

OutProt("пересекает.\n");return true;}

OutProt("не пересекает.\n");return false;}

bool IsCorrect(double\*\*& A, size\_t\*& B, const short& Pos, size\_t& i) {

if (Pos > 9) return false;

switch (Pos + 1){

case 10:

if (EqualLen(A[B[1]], A[B[2]], A[B[8]], A[i])&& IsNormal(A[B[2]], A[B[3]], A[B[8]], A[i])) return true; break;

case 9:

if (EqualLen(A[B[2]], A[B[3]], A[B[7]], A[i])&& IsNormal(A[B[1]], A[B[2]], A[B[7]], A[i])) return true;

break;

case 8: if (EqualLen(A[B[1]], A[B[2]], A[B[6]], A[i])

&& IsNormal(A[B[2]], A[B[3]], A[B[6]], A[i])) return true; break;

case 7: if (EqualLen(A[B[0]], A[B[1]], A[B[5]], A[i])

&& IsNormal(A[B[2]], A[B[3]], A[B[4]], A[i])) return true; break;

case 6: if (EqualLen(A[B[0]], A[B[1]], A[B[4]], A[i])

&& IsNormal(A[B[1]], A[B[2]], A[B[0]], A[i])&& Len(A[B[0]], A[i]) > Len(A[B[2]], A[B[3]])) return true; break;

case 5: if (EqualLen(A[B[1]], A[B[2]], A[B[3]], A[i])

&& IsNormal(A[B[2]], A[B[3]], A[B[3]], A[i])) return true;break;

case 4:if (IsNormal(A[B[1]], A[B[2]], A[B[2]], A[i])&& Len(A[B[2]], A[i]) > 0 && !Prinadlejit(A[B[0]], A[B[1]], A[B[2]], A[i])&& !Peresec(A[B[0]], A[B[1]], A[B[2]], A[i])) return true; break;

case 3: if (Spec3(A[B[0]], A[B[1]], A[i])&& Len(A[B[1]], A[i]) > 0) return true; break;

default: return true; break;}return false;}

bool NotInSaveF(size\_t\*& B) {

OutProt("\tПроверка на повтор фигуры:\n");

FT.seekg(0, ios::beg);

FT.clear();

size\_t B2[10];

bool b = true;

for (size\_t i = 0; FT.read(reinterpret\_cast<char\*>(B2), sizeof(B) \* 10); i++) {

OutProt("\t\t№:");

OutProt(i, ":\n");

FT.seekg(sizeof(B), ios::cur);

b = false;

for (size\_t j : B2)

if (CanUse(B, j, 10)) b = true;

if (!b) {OutProt("\t\tФигура уже найдена.\n");return b;}}

OutProt("\t\tФигура новая.\n");return b;}

void SaveF(size\_t\*& B) {

FT.seekp(0, ios::end);

FT.clear();

OutProt("Сохранение фигуры №");

OutProt(NumOfFig, ".\n");

FT.write(reinterpret\_cast<char\*>(B), sizeof(B) \* 11);

NumOfFig++;}

void OutFig(double\*\*& A, size\_t\*& B) {

OutProt("Найдена фигура:\n");

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

OutProt("\t Точка №");

OutProt(B[i], ": ");

OutProt(A[B[i]], "()", "\n"); }

OutProt("\tСодержит: ");

OutProt(B[10], " точек.\n");}

void OutFigMain(double\*\*& A, size\_t\*& B) {

OutMain("Найдена фигура:\n");

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

OutMain("\t Точка №");

OutMain(B[i], ": ");

OutMain(A[B[i]], "()", "\n"); }

OutMain("\tСодержит: ");

OutMain(B[10], " точек.\n");}

void FindMax(double\*\*& A, size\_t\* B, const size\_t& S) {

FT.seekg(0, ios::beg);

FT.clear();

OutMain("Максимум: ");

OutMain(MaxC, ". Поиск фигуры:\n");

for (size\_t i = 0; FT.read(reinterpret\_cast<char\*>(B), sizeof(B) \* 11); i++) {if (B[10] == MaxC) {

OutMain("\tФигура и точки, которые она содержит №");

OutMain(i, ":\n");

OutFigMain(A, B);

for (size\_t i = 0; i < S; i++)

if (CanUse(B, i, 10)&& (Prinadlejit(A[B[0]], A[B[1]], A[B[9]], A[i])|| Prinadlejit(A[B[2]], A[B[3]], A[B[7]], A[B[8]], A[i]) || Prinadlejit(A[B[4]], A[B[5]], A[B[6]], A[i]))) { OutMain("\t\tТочка №"); OutMain(i, ": "); OutMain(A[i], "()", "\n");}}}}

size\_t& Schot(double\*\*& A, size\_t\* B, const size\_t& S) {

B[10] = 0;

OutProt("\tСчёт точек в фигуре:\n");

for (size\_t i = 0; i < S; i++) {

OutProt("\t\tТочка ");

OutProt(i, ": ");

OutProt(A[i], "()", ":\n");

if (CanUse(B, i, 10) && (Prinadlejit(A[B[0]], A[B[1]], A[B[9]], A[i]) || Prinadlejit(A[B[2]], A[B[3]], A[B[7]], A[B[8]], A[i]) || Prinadlejit(A[B[4]], A[B[5]], A[B[6]], A[i]))) B[10]++;}

OutFig(A, B);return B[10];}

void SubProcces(double\*\*& A, size\_t\* B, const size\_t& S, short Pos) {

OutProt("\tПоиск точки №");

OutProt(static\_cast<size\_t>(Pos), ":\n");

for (size\_t i = 0; i < S; i++) {

OutProt("\t\tПроверка точки ");

OutProt(i, ":\n");

if (CanUse(B, i, Pos) && Pos < 10 && IsCorrect(A, B, Pos, i)) {

OutProt("\tТочка ");

OutProt(i, " установлена как №");

OutProt(static\_cast<size\_t>(Pos), ".\n");

B[Pos] = i; if (Pos >= 9) {

if (NotInSaveF(B)) {

MaxC = Max(Schot(A, B, S), MaxC);

SaveF(B); }} elseSubProcces(A, B, S, Pos + 1);}}

OutProt("\tВыход из цикла для точки №");

OutProt(static\_cast<size\_t>(Pos), ".\n");

if (Pos) {

OutProt("\tВвозврат к точке №");

OutProt(static\_cast<size\_t>(Pos - 1), ":\n");}return;}

void Procces(double\*\*& A, size\_t& S) {

size\_t B[11];

OutProt("Поиск фигры:\n");

SubProcces(A, B, S, 0);

FindMax(A, B, S);

return;}

int main() {

setlocale(LC\_ALL, "RUSSIAN");

FProc.setf(ios::fixed);

FProt.setf(ios::fixed);

FProt.precision(7);

FProt.precision(7);

size\_t S = CalcSize();

double\*\* A = new double\* [S];

for (int i = 0; i < S; i++) A[i] = new double[2];

Read(A, S);

Procces(A, S);

FProt.close();

FProc.close();

FT.close();

F.close();

for (int i = 0; i < S; i++) delete[] A[i];return 0;}

# Результат работы программы

|  |  |
| --- | --- |
| **Фигура 1** | **Фигура 2** |
| Максимум: 1. Поиск фигуры:  Фигура и точки, которые она содержит №0:  Найдена фигура:  Точка №13: (-18.507000, 14.423000)  Точка №3: (-8.425000, 10.330000)  Точка №2: (-11.381000, 14.593000)  Точка №1: (-4.955000, 19.050000)  Точка №0: (-1.998000, 14.786000)  Точка №24: (-2.297000, 25.663000)  Точка №23: (-12.378000, 29.756000)  Точка №22: (-9.422000, 25.492000)  Точка №21: (-15.849000, 21.036000)  Точка №20: (-18.805000, 25.299000)  Содержит: 1 точек.  Точка №36: (-10.000000, 15.666667) | Максимум: 2. Поиск фигуры:  Фигура и точки, которые она содержит №0:  Найдена фигура:  Точка №0: (16.099000, 11.754000)  Точка №14: (7.496000, 25.006000)  Точка №3: (3.909000, 17.968000)  Точка №2: (-12.109000, 26.132000)  Точка №1: (-15.696000, 19.094000)  Точка №24: (-24.299000, 32.346000)  Точка №23: (-8.521000, 33.171000)  Точка №22: (-12.109000, 26.132000)  Точка №21: (3.909000, 17.968000)  Точка №4: (0.321000, 10.929000)  Содержит: 2 точек.  Точка №11: (8.500000, 13.500000)  Точка №26: (1.600000, 12.500000) |
| **Фигура 3** | **Фигура 1 с дубликатом точки** |
| Максимум: 20. Поиск фигуры:  Фигура и точки, которые она содержит №0:  Найдена фигура:  Точка №0: (-20.000000, 0.000000)  Точка №1: (-10.000000, 20.000000)  Точка №2: (-10.000000, 10.000000)  Точка №3: (10.000000, 10.000000)  Точка №4: (10.000000, 20.000000)  Точка №5: (20.000000, 0.000000)  Точка №6: (10.000000, -20.000000)  Точка №7: (10.000000, -10.000000)  Точка №8: (-10.000000, -10.000000)  Точка №9: (-10.000000, -20.000000)  Содержит: 20 точек.  Точка №11: (3.200000, -5.666667)  Точка №13: (-11.000000, 7.000000)  Точка №14: (-18.500000, 2.000000)  Точка №17: (-2.000000, -1.200000)  Точка №20: (12.500000, 2.000000)  Точка №23: (5.000000, -9.400000)  Точка №24: (-6.400000, -3.600000)  Точка №27: (-8.000000, 9.500000)  Точка №30: (4.600000, 6.200000)  Точка №35: (-11.000000, -9.200000)  Точка №39: (15.000000, -5.000000)  Точка №41: (5.333333, 1.400000)  Точка №42: (-9.500000, -2.250000)  Точка №43: (-11.500000, -6.400000)  Точка №45: (6.333333, 1.000000)  Точка №49: (3.400000, -1.500000)  Точка №52: (-12.333333, 6.000000)  Точка №57: (2.000000, 0.000000)  Точка №58: (-9.200000, -2.000000)  Точка №59: (-1.500000, 1.500000) | Максимум: 1. Поиск фигуры:  Фигура и точки, которые она содержит №0:  Найдена фигура:  Точка №0: (-18.507000, 14.423000)  Точка №1: (-18.805000, 25.299000)  Точка №2: (-15.849000, 21.036000)  Точка №3: (-9.422000, 25.492000)  Точка №4: (-12.378000, 29.756000)  Точка №5: (-2.297000, 25.663000)  Точка №6: (-1.998000, 14.786000)  Точка №7: (-4.955000, 19.050000)  Точка №8: (-11.381000, 14.593000)  Точка №9: (-8.425000, 10.330000)  Содержит: 1 точек.  Точка №37: (-10.000000, 15.666667)  Фигура и точки, которые она содержит №1:  Найдена фигура:  Точка №0: (-18.507000, 14.423000)  Точка №1: (-18.805000, 25.299000)  Точка №2: (-15.849000, 21.036000)  Точка №3: (-9.422000, 25.492000)  Точка №4: (-12.378000, 29.756000)  Точка №10: (-2.297000, 25.663000)  Точка №6: (-1.998000, 14.786000)  Точка №7: (-4.955000, 19.050000)  Точка №8: (-11.381000, 14.593000)  Точка №9: (-8.425000, 10.330000)  Содержит: 1 точек.  Точка №37: (-10.000000, 15.666667) |
| **Все фигуры и размер больше реального** | **Размер <0** |
| Error: Размер приравнен к реальному, так как меньше указанного!  Максимум: 21. Поиск фигуры:  Фигура и точки, которые она содержит №0:  Найдена фигура:  Точка №50: (-20.000000, 0.000000)  Точка №51: (-10.000000, 20.000000)  Точка №52: (-10.000000, 10.000000)  Точка №53: (10.000000, 10.000000)  Точка №54: (10.000000, 20.000000)  Точка №55: (20.000000, 0.000000)  Точка №56: (10.000000, -20.000000)  Точка №57: (10.000000, -10.000000)  Точка №58: (-10.000000, -10.000000)  Точка №59: (-10.000000, -20.000000)  Содержит: 21 точек.  Точка №1: (3.200000, -5.666667)  Точка №3: (-11.000000, 7.000000)  Точка №4: (-18.500000, 2.000000)  Точка №7: (-2.000000, -1.200000)  Точка №10: (12.500000, 2.000000)  Точка №13: (5.000000, -9.400000)  Точка №14: (-6.400000, -3.600000)  Точка №17: (-8.000000, 9.500000)  Точка №20: (4.600000, 6.200000)  Точка №25: (-11.000000, -9.200000)  Точка №29: (15.000000, -5.000000)  Точка №31: (5.333333, 1.400000)  Точка №32: (-9.500000, -2.250000)  Точка №33: (-11.500000, -6.400000)  Точка №35: (6.333333, 1.000000)  Точка №39: (3.400000, -1.500000)  Точка №42: (-12.333333, 6.000000)  Точка №47: (2.000000, 0.000000)  Точка №48: (-9.200000, -2.000000)  Точка №49: (-1.500000, 1.500000)  Точка №68: (-11.381000, 14.593000) | Error: Размер меньше 0, не указан или равен 0!  Максимум: 0. Поиск фигуры: |

Вывод протокола для первого варианта:

Чтение размера:

Считан символ: 6

Считано число: 60.0000000

Расчёт реального размера:

Считан символ:

Считан символ: -

Считано число: -1.9980000

Считан символ:

Считан символ: 1

Считано число: 14.7860000

Реальный размер переназначен на: 1

Считан символ:

…

Размер либо больше, либо равен указаному.

Файл сброшен.

Пропуск размера:

Считан символ: 6

Считано число: 60.0000000

Запись в массив:

Считан символ:

Считан символ: -

Считано число: -1.9980000

В позиции [0, 1] записано число: -1.9980000

Считан символ:

Считан символ: 1

Считано число: 14.7860000

В позиции [0, 2] записано число: 14.7860000

Считан символ:

Считан символ: -

Считано число: -4.9550000

В позиции [1, 1] записано число: -4.9550000

Считан символ:

Считан символ: 1

Считано число: 19.0500000

В позиции [1, 2] записано число: 19.0500000

Считан символ:

**\*\*\*…\*\*\***

Поиск точки №0:

Проверка точки 0:

Проверка наличия точки 0 в фигуре:

Нет в фигуре.

Точка 0 установлена как №0.

Поиск точки №1:

Проверка точки 0:

Проверка наличия точки 0 в фигуре:

Сравнение с 0:

Есть в фигуре.

Проверка точки 1:

Проверка наличия точки 1 в фигуре:

Сравнение с 0:

Нет в фигуре.

Точка 1 установлена как №1.

Поиск точки №2:

Проверка точки 0:

Проверка наличия точки 0 в фигуре:

Сравнение с 0:

Есть в фигуре.

Проверка точки 1:

Проверка наличия точки 1 в фигуре:

Сравнение с 0:

Сравнение с 1:

Есть в фигуре.

Проверка точки 2:

Проверка наличия точки 2 в фигуре:

Сравнение с 0:

Сравнение с 1:

Нет в фигуре.

Отрезок с координатами точки 1 (-1.9980000, 14.7860000) и точки 2 (-11.3810000, 14.5930000) преобразован в вектор {-9.3830000, -0.1930000}

Длина вектора {-9.3830000, -0.1930000} = 9.3849847

Отрезок с координатами точки 1 (-4.9550000, 19.0500000) и точки 2 (-11.3810000, 14.5930000) преобразован в вектор {-6.4260000, -4.4570000}

Длина вектора {-6.4260000, -4.4570000} = 7.8203788

Отрезок с координатами точки 1 (-1.9980000, 14.7860000) и точки 2 (-4.9550000, 19.0500000) преобразован в вектор {-2.9570000, 4.2640000}

Длина вектора {-2.9570000, 4.2640000} = 5.1889830

**\*\*\*…\*\*\***

Вывод пользователю: Максимум: Вывод пользователю: 0. Поиск фигуры:

Вывод пользователю: Фигура и точки, которые она содержит №Вывод пользователю: 0:

Вывод пользователю: Найдена фигура:

Вывод пользователю: Точка №Вывод пользователю: 13: Вывод пользователю: (-18.5070000, 14.4230000)

Вывод пользователю: Точка №Вывод пользователю: 3: Вывод пользователю: (-8.4250000, 10.3300000)

Вывод пользователю: Точка №Вывод пользователю: 2: Вывод пользователю: (-11.3810000, 14.5930000)

Вывод пользователю: Точка №Вывод пользователю: 1: Вывод пользователю: (-4.9550000, 19.0500000)

Вывод пользователю: Точка №Вывод пользователю: 0: Вывод пользователю: (-1.9980000, 14.7860000)

Вывод пользователю: Точка №Вывод пользователю: 24: Вывод пользователю: (-2.2970000, 25.6630000)

Вывод пользователю: Точка №Вывод пользователю: 23: Вывод пользователю: (-12.3780000, 29.7560000)

Вывод пользователю: Точка №Вывод пользователю: 22: Вывод пользователю: (-9.4220000, 25.4920000)

Вывод пользователю: Точка №Вывод пользователю: 21: Вывод пользователю: (-15.8490000, 21.0360000)

Вывод пользователю: Точка №Вывод пользователю: 20: Вывод пользователю: (-18.8050000, 25.2990000)

Вывод пользователю: Содержит: Вывод пользователю: 0 точек.

Проверка наличия точки 0 в фигуре:

Сравнение с 13:

Сравнение с 3:

Сравнение с 2:

Сравнение с 1:

Сравнение с 0:

Есть в фигуре.

Проверка наличия точки 1 в фигуре:

Сравнение с 13:

Сравнение с 3:

Сравнение с 2:

Сравнение с 1:

Есть в фигуре.

**\*\*\*…\*\*\***

Проверка принадлежности точки (-9.2000000, -21.0000000) фигуре: (-18.5070000, 14.4230000) (-8.4250000, 10.3300000) (-18.8050000, 25.2990000):

Точка с X координатой -9.2000000 между X координатами прямой (-18.5070000, 14.4230000) (-8.4250000, 10.3300000) находится.

Точка с X координатой -9.2000000 между X координатами прямой (-8.4250000, 10.3300000) (-18.8050000, 25.2990000) находится.

Значение Y для прямой (-18.5070000, 14.4230000) (-8.4250000, 10.3300000) на координате оси X = -9.2000000: 10.6446276

Значение Y для прямой (-8.4250000, 10.3300000) (-18.8050000, 25.2990000) на координате оси X = -9.2000000: 11.4476276

Точка (-9.2000000, -21.0000000) между Y фигуры (-18.5070000, 14.4230000) (-8.4250000, 10.3300000) (-18.8050000, 25.2990000) не находится.

Точка с (-9.2000000, -21.0000000) не принадлежит фигуре.

Проверка принадлежности точки (-9.2000000, -21.0000000) фигуре: (-11.3810000, 14.5930000) (-4.9550000, 19.0500000) (-9.4220000, 25.4920000) (-15.8490000, 21.0360000):

Точка с X координатой -9.2000000 между X координатами прямой (-11.3810000, 14.5930000) (-4.9550000, 19.0500000) находится.

Точка с X координатой -9.2000000 между X координатами прямой (-4.9550000, 19.0500000) (-9.4220000, 25.4920000) находится.

Значение Y для прямой (-11.3810000, 14.5930000) (-4.9550000, 19.0500000) на координате оси X = -9.2000000: 16.1057166

Значение Y для прямой (-4.9550000, 19.0500000) (-9.4220000, 25.4920000) на координате оси X = -9.2000000: 25.1718469

Точка (-9.2000000, -21.0000000) между Y фигуры (-11.3810000, 14.5930000) (-4.9550000, 19.0500000) (-9.4220000, 25.4920000) не находится.

Точка (-9.2000000, -21.0000000) не принадлежит фигуре.

Проверка принадлежности точки (-9.2000000, -21.0000000) фигуре: (-1.9980000, 14.7860000) (-2.2970000, 25.6630000) (-12.3780000, 29.7560000):

Точка с X координатой -9.2000000 между X координатами прямой (-1.9980000, 14.7860000) (-2.2970000, 25.6630000) не находится.

Точка с X координатой -9.2000000 между X координатами прямой (-2.2970000, 25.6630000) (-12.3780000, 29.7560000) находится.

Точка с X координатой -9.2000000 между X координатами прямой (-12.3780000, 29.7560000) (-1.9980000, 14.7860000) находится.

Значение Y для прямой (-2.2970000, 25.6630000) (-12.3780000, 29.7560000) на координате оси X = -9.2000000: 28.4656961

Значение Y для прямой (-12.3780000, 29.7560000) (-1.9980000, 14.7860000) на координате оси X = -9.2000000: 25.1726994

Точка (-9.2000000, -21.0000000) между Y фигуры (-2.2970000, 25.6630000) (-12.3780000, 29.7560000) (-1.9980000, 14.7860000) не находится.

Точка с (-9.2000000, -21.0000000) не принадлежит фигуре.

**\*\*\*…\*\*\***

Вывод пользователю: Точка №Вывод пользователю: 36: Вывод пользователю: (-10.0000000, 15.6666667)

Заключение

**Вывод:** написал программу, которая находит фигуры, которые содержат больше всего точек. Объединил все полученные знания за курс в одной работе. Научился работать с сохранением временных данных в файле. Реализовал функции, использующие геометрические правила.