**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

Пояснительная записка

**к курсовой работе**

**по дисциплине «Программирование»**

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

|  |  |  |
| --- | --- | --- |
| Студент гр. 2309 |  | Савин П.А. |
| Преподаватель |  | Калмычков В.А. |

Санкт-Петербург

2022

**Задание на курсовую работу**

Студент: Савин П.А.

Группа: 2309

Тема работы:

Использование массивов для решения геометрических задач.

Исходные данные:

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

Фигура: сердце

Содержание пояснительной записки:

Титульный лист, задание, аннотация, содержание, исходная формулировка, анализ задания, контрольный пример, описание.

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 17.11.2022

Дата сдачи курсовой работы:

Дата защиты курсовой работы:

|  |  |  |
| --- | --- | --- |
| Студент гр. 2309 |  | Савин П.А. |
| Преподаватель |  | Калмычков В.А. |

**Аннотация.**

Формулировка задания: Даны N точек на плоскости. Найти среди них точки, являющиеся вершинами фигуры, содержащей максимальное число заданных точек.

Фигура: сердце

Задача рассмотрена с математической точки зрения (условия существования сердца и условия принадлежности точки сердцу), на основе чего был реализован алгоритм решения задачи на компьютере.

В работе продемонстрирована работа с файлами, хранение и обработка числовых данных через двумерный массив.

Каждая точка в массиве представляет собой упорядоченную пару чисел с плавающей точкой, представляющих координаты точки по осям x и y, соответственно.

Считывание точек происходит последовательно. Точки, введенные неправильно с точки зрения представления чисел с плавающей точкой, не учитываются. В файле вывода указаны следующие данные: число точек, которое является максимумом, точки, из которых составлена(ы) искомая(ые) фигура(ы) и точки для каждой фигуры, которые она содержит.

**Summary.**

Task statement: given an amount of N points on the plane. Find the points that are the vertices of a figure containing the largest amount of given points.

Figure: heart

The problem was considered from a mathematical point of view (checking heart existence conditions and conditions of point belonging the heart), on the basis of which the algorithm for solving the problem on a computer was implemented.

This paper demonstrates file handling, storage and processing of numeric data through a two-dimensional array.

Each point in array is an ordered pair of floating-point numbers that represent the coordinates of the point on axes x and y, respectively.

The points are read sequentially. Points that are incorrect in terms of floating-point number representation are ignored. The output file contains the following data: the maximum amount of points, the points that are the vertices of the desired figure(s), and the points for each figure it contains.

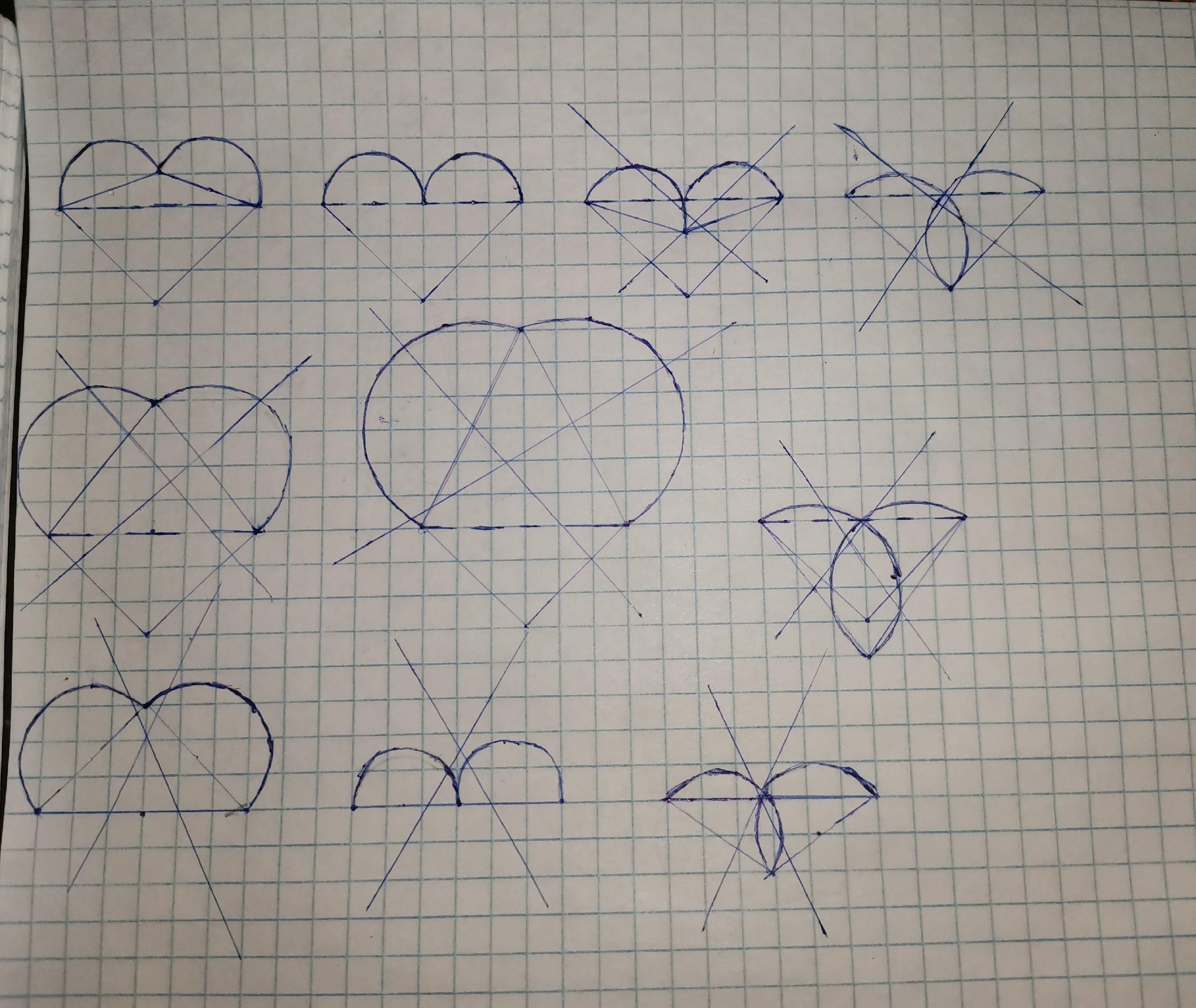
**Формулировка задания.**

Даны N точек на плоскости. Найти среди них точки, являющиеся вершинами фигуры, содержащей максимальное число заданных точек. Фигура:



**Анализ задания.**

Сердце на рисунке представлено лишь в одном из вариантов. Анализ с построением сердец, близких к образцовому, дал следующие возможные варианты:



Зачеркнуты те варианты, которые не будут рассматриваться в задаче как полноценные фигуры (даже несмотря на то, что варианты 5 и 6 еще выглядят правдоподобно; они попадают под один общий случай, при котором фигура выглядит уже не похожей на сердце).

По двум оставшимся случаям нужно учесть, что сердце может поворачиваться в пространстве, так что формулы и закономерности должны учитывать этот факт.

Также стоит уточнить, что точки в пространстве заданы координатами по осям x и y.

**Математическая** **постановка.**

Дано:

Найти:

Точки, которые являются вершинами сердца(ец) с самым большим кол-вом точек внутри.

Решение:

Последовательно перебрать все точки из набора.

Каждый раз выбирать очередные 4 точки и определять, могут ли они образовать сердце. Поскольку каждая точка является выраженной вершиной фигуры, таких вариантов из одной четверки точек будет 12. 12, потому что сердце имеет центральную симметрию: если поставить точки на левые и правые края и выйдет сердце, при перестановке выйдет то же самое сердце; аналогично и ситуацией, когда очередной набор вершин не образует сердце. 24 комбинаторных вариантов сразу становится в два раза меньше.

Ключевые условия образования сердца:

1. В основе фигуры лежит выпуклый дельтоид – два равнобедренных треугольника с общим основанием. В крайнем случае, это равнобедренный треугольник, но с условием, что точка верха лежит строго на прямой точек краев и нигде больше

2. Треугольник, образуемый верхом и краями, не может быть больше треугольника, образуемого низом и краями

Подвязка условий к геометрии:

Пусть – точки левого и правого краев, верха и низа сердца, соответственно.

**Образование дельтоида**: длины отрезков от точки низа до точек краев равны; аналогично с отрезками до точки верха.

(1)

(2)

Также нужно учесть, что точка низа не находится на одной прямой одновременно с обеими точками краев. Проверяется это подсчетом смешанного векторного и псевдоскалярного произведений, по двум известным точкам прямой и третьей, требующей проверки на ее положение относительно прямой.

Пусть для формулы смешанного произведения – известные точки на прямой, а – точка, чье положение нужно проверить. Тогда:

(3)

У этой формулы есть два возможных случая:

1. Значение нулевое – точка находится на прямой

2. Значение ненулевое – точка находится на одной из полуплоскостей относительно прямой

Так вот, для проверки точки низа достаточно проверить, что значение по формуле (3) ненулевое.

**Выпуклость образуемого дельтоида**: точка верха не должна лежать на площади треугольника, образуемого низом и краями.

Проверка этого условия достигается проверкой принадлежности точки треугольнику.

Точка принадлежит треугольнику, если значения по формуле (3) для всех трех сторон имеют одинаковый знак или одно из значений принимает ноль (точка лежит на прямой). Последний случай отбрасывается, так как верх может только находиться на прямой краев, что уже учтено условиями образования сердца.

**Сравнение двух треугольников**: длина отрезка от верха до центра линии краев не больше аналогичной от низа.

Пусть – точка середины линии краев,

(4)

Тогда справедливо следующее неравенство:

(5)

Сердце построено, прогоняемся по набору точек и каждую точку, не занятую под вершины, проверяем на принадлежность фигуре.

Точка принадлежит сердцу, если соблюдено хотя бы одно из следующих условий:

1. Точка принадлежит нижнему треугольнику (нахождение на одной из сторон считается за принадлежность)

2. Точка принадлежит верхнему треугольнику

3. Точка принадлежит одной из полуокружностей сердца, но проверка будет на окружности, тогда нужно еще и проверять, чтобы точка не выходила за пределы продолжений сторон сердца

**Принадлежность треугольнику (верхнему, нижнему)**: применение формулы (3) описано выше.

**Принадлежность окружности**: центры окружностей находятся на отрезках вершин, радиусом является половина длины отрезка. Далее точка проверяется по каноническому уравнению окружности.

Пусть – центры окружностей слева и справа, соответственно,

(6)

– радиусы соответствующих окружностей,

(7)

Подставляя, получаем следующие неравенства для проверки точки:

(8)

(9)

Для включения в сердце по окружностям достаточно соблюдения хотя бы одного неравенства.

Однако все выведенное необходимо ограничить обязательным включением в прямые, поскольку в некоторых случаях продолжения полуокружностей могут выходить за границы фигуры.

Проверка аналогична выведенной системе условий для треугольника, только в ней участвуют две прямые и допускается принадлежность одной из прямых.

Так вот, каждая точка проходит через набор условий и идет в зачет очередному сердцу, если проходит по условиям. Если у очередного сердца число входящих точек превысило установленный максимум, он обновляется.

Как только заканчивается перебор всех возможных фигур, выписываем те, которые поставили максимум по точкам, точки, которые являются вершинами этих фигур и точки, которые входят в очередную фигуру. Это и будет ответ на задачу.

Преимущество в проработанных выше формулах в том, что на их работоспособность не влияет поворот фигуры в пространстве.

**Контрольный пример.**

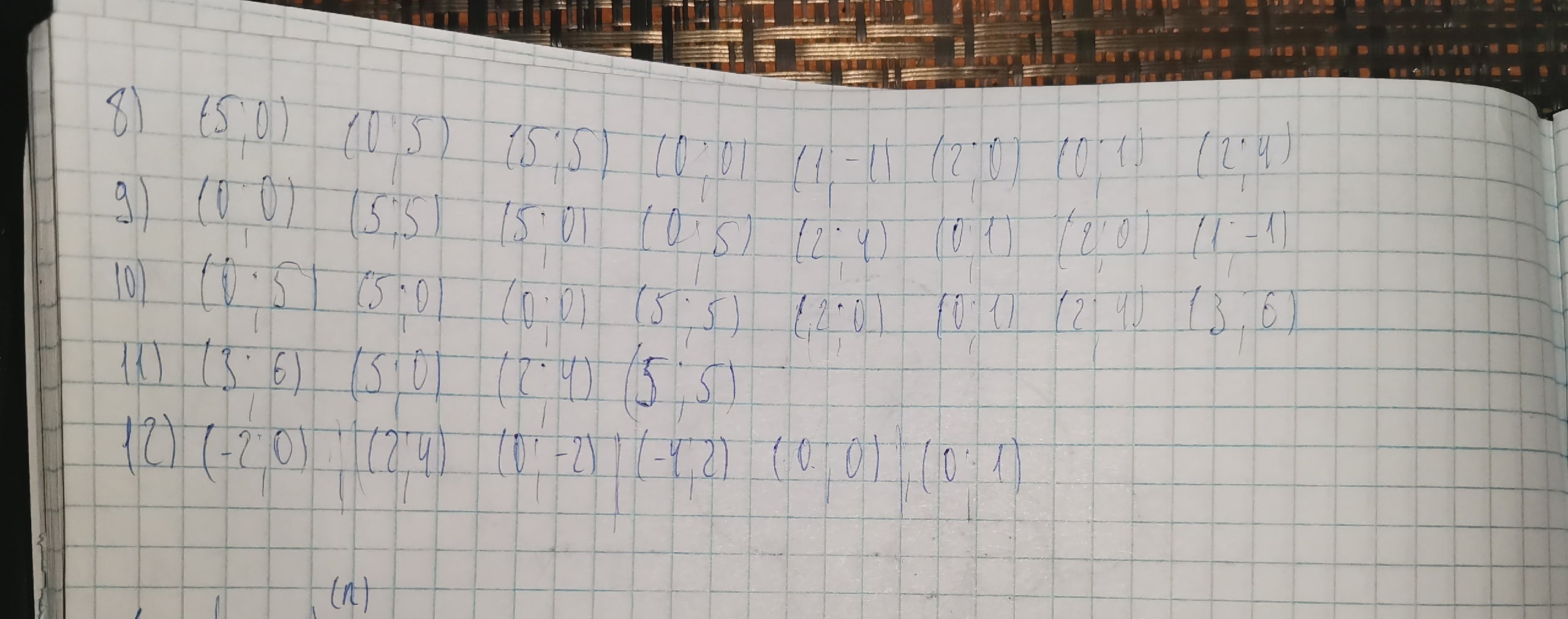
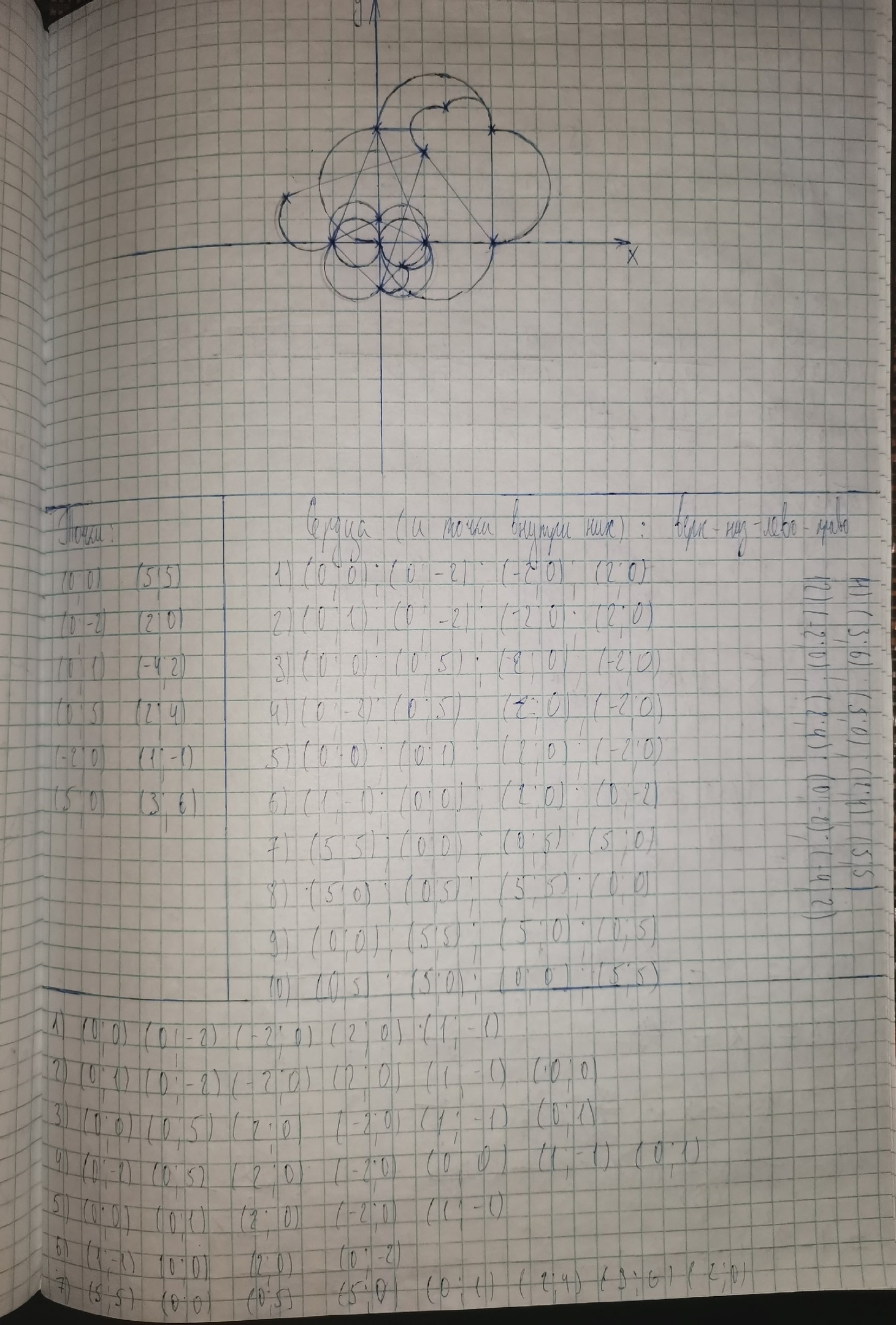
Для контрольного примера были вязты следующие 12 точек:

(0; 0) (0; -2) (0; 1) (0; 5)

(-2; 0) (2; 0) (5; 0) (5; 5)

(-4; 2) (2; 4) (1; -1) (3; 6)

Соответственно, снизу были образованы все возможные сердца и точки, которые этим сердцам принадлежат.



Наибольшее число точек – 9, сердца с таким кол-вом точек:

1) верх - (0; 0); низ - (5; 5); лево - (0; 5); право - (5; 0)

2) верх - (0; 5); низ - (5; 0); лево - (0; 0); право - (5; 5)

3) верх - (5; 0); низ - (0; 5); лево - (0; 0); право - (5; 5)

4) верх - (5; 5); низ - (0; 0); лево - (0; 5); право - (5; 0)

**Особенности реализации задания на компьютере.**

Теоретическая реализация уже подразумевает много однообразных действий, поэтому целесообразно раскидать их по обособленным функциям.

Помимо дробления программы требуется реализовать считывание точек с входного файла, да и в целом реализовать интерфейс через три файла: входной с данными, выходной с ответом на задачу и логирующий с записями всех произошедших действий.

Считывание данных не без нюансов: не гарантировано, что весь файл состоит из корректных строк, в которых две последовательные записи чисел являются координатами очередной точки. Напротив, в файле могут встречаться пустые строки, строки с большим/меньшим положенного числом записей и некорректные записи чисел. Было решено сначала определять, сколько вообще выходит корректных пар координат, а затем таким же способом (с добавлением ограничителей для записи) записывать пары: пытаемся очередную последовательность записать целиком в тип double, если получается – получили число, ищем следующее (или идем до конца строки, если записали второе число); если не получается – снимаем флаги ошибок и идем до следующей непустой последовательности, либо до конца строки, и так и считываем кол-во пар, и записываем сами пары.

В математической постановке задачи не было упомянуто, что в исходном наборе точки могут дублировать друг друга. Было решено эти дубли отбрасывать и адекватно реагировать на возникающий нехваток точек.

Также в математике на компьютере возникает еще один неприятный нюанс: сравнение дробных чисел и арифметика с ними. Это непредсказуемая часть программы ввиду особенностей хранения подобных чисел. Поэтому сравнения чисел были приведены к фиксированной погрешности: если абсолютная разница чисел меньше заданной погрешности (очень малая часть единицы, в программе взято 12 знаков), то будем считать, что числа равны, в противном случае можем безопасно сравнивать их без погрешности. Проблемы арифметики разрешены заменой операций: в программе нет смысла сравнивать корни (например, в формулах (1), (2), (5), (8) и (9)) и вообще извлекать их, так как могут возникать еще большие погрешности, лучше сравнивать квадраты; деление заменить умножением (оно работает быстрее и вызывает меньше проблем с памятью).

**Разработка интерфейса пользователя.**

Интерфейс логирующего файла:

O1: Начало работы, попытка открытия файла

O2: Уведомление о ненайденном файле (в случае, если входной файл не был найден)

O3: Запуск определения точек

O4: Уведомление о недостатке точек или об их отсутствии (в случае, если алгоритм определения не нашел/нашел недостаточно корректных пар)

O5: «Обнаружены точки: »

O6: «ddd) x = ±ddd.ddd; y = ±ddd.ddd»

O7: Запуск записи

O8: «Записаны следующие точки: »

O9: «ddd) x = ±ddd.ddd; y = ±ddd.ddd»

O10: Уведомление о дублях (если таковые обнаружились при записи)

O11: Уведомление о недостатке точек (в случае, если отбрасывание точек привело к нехватке точек или все точки изначально были дублями одной точки)

O12: «Построены сердца: »

O13: «ddd) верх – (±ddd.ddd; ±ddd.ddd); низ – (±ddd.ddd; ±ddd.ddd); лево – (±ddd.ddd; ±ddd.ddd); право – (±ddd.ddd; ±ddd.ddd)»

«Точек внутри: ddd»

«Координаты точек: »

(±ddd.ddd; ±ddd.ddd)

O14: Уведомление об отсутствии сердец (если такое произошло в ходе обработки)

O15: Уведомление об отсутствии однозначного ответа на задачу ввиду отсутствия входящих точек кроме вершин (если такое произошло)

O16: «Определено максимальное число точек, которое могут содержать обнаруженные сердца: ddd.»

«Список таких сердец: »

«ddd) верх – (±ddd.ddd; ±ddd.ddd); низ – (±ddd.ddd; ±ddd.ddd); лево – (±ddd.ddd; ±ddd.ddd); право – (±ddd.ddd; ±ddd.ddd)»

«Координаты точек: »

(±ddd.ddd; ±ddd.ddd)

O17: Завершение программы

Выходной файл повторяет шаблоны O2, O4, O11, O14-16, но в более обобщенном и сглаженном варианте, а также дополнительно предлагает пользователю увидеть полную информацию в лог-файле.

**Описание используемых данных.**

|  |  |  |
| --- | --- | --- |
| Тип | Имя | Назначение |
| int | cur (def\_size, fill\_arr) | Хранение значения текущего положения файлового указателя |
| ti, li, di, ri (log\_pts, pts\_inside) | Индексы точек, занятых под вершины сердца |
| unsigned int | size (def\_size) | Подсчет размера потенциального массива с точками |
| size (process, pts\_inside, log\_pts, clear\_arr) | Размер массива, с которым ведется работа |
| count (fill\_arr) | Кол-во точек, записанных в массив |
| count (pts\_inside) | Кол-во точек внутри очередного сердца |
| max\_points (process) | Максимальное кол-во точек внутри одного сердца |
| hearts (process) | Кол-во найденных сердец (также счетчик для оформления вывода) |
| pts (process) | Кол-во точек внутри текущего сердца |
| pts\_cnt (main) | Кол-во найденных точек для выделения памяти под массив |
| arr\_len (main) | Реальный размер массива точек |
| double | A (main) | Исходный массив точек |
| arr (process, fill\_arr, pts\_inside, log\_pts) | Исходный массив, передаваемый в функции-обработчики |
| tmpx, tmpy (def\_size, fill\_arr) | Заглушки для x и y точки при чтении и записи |
| eps (is\_eq, is\_lt, is\_gt) | Переменная под точность сравнения |
| x, y (is\_eq, is\_lt, is\_gt) | Два значения, которые требуется сравнить |
| x, y (inside\_heart) | Координаты точки, которая передается на проверку на принадлежность сердцу |
| tx, ty, lx, ly, dx, dy, rx, ry (is\_heart, inside\_heart, log\_pts, pts\_inside) | Координаты точек, проверяемые как вершины на образование сердца в is\_heart, в остальных функциях уже вершины очередного сердца |
| leng2dl, leng2dr, leng2tl, leng2tr, cx, cy, leng2dc, leng2tc, lined, linetc, lintel, linetr (is\_heart) | Вспомогательные формулы, описанные в математической постановке, для проверки образования сердца: длины отрезков, середина краевой линии, значения смешанных произведений для проверки выпуклости |
| clx, cly, crx, cry, rad2l, rad2r, c2l, c2r, ldl, ldr, lc, ltl, ltr (inside\_heart) | Вспомогательные формулы, описанные в математической постановке, для проверки принадлежности точки сердцу: центры окружностей, их радиусы, значения смешанных произведений и уравнений окружности |
| ax, ay, bx, by, cx, cy (vec\_prod) | Координаты точек, из которых высчитывается смешанное произведение |
| char | tmpc (def\_size, fill\_arr) | Заглушка для посимвольного чтения файла |
| bool | x, y (def\_size, fill\_arr) | Флаги для встреченных чисел под x и y, соответственно |
| met (fill\_arr) | Флаг для пресечения записи дубля в массив |
| std::ifstream | infile/in (main/def\_size, fill\_arr) | Входной файл с данными |
| std::ofstream | logfile/log (main/def\_size, fill\_arr, log\_pts, process) | Лог-файл для записи всех событий во время работы программы |
| outfile/out/log (main/process/log\_pts) | Выходной файл для записи ответа на задачу или других сообщений в случае невозможности получения ответа |

**Описание используемых функций.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя | Тип | Назначение | Параметры | | | |
| входные | выходные | модифицируемые | транзитные |
| def\_size | unsigned int | Определение потенциального размера массива точек |  |  | in, log |  |
| fill\_arr | unsigned int | Заполнение созданного массива уникальными точками | size |  | in, log, arr |  |
| is\_eq | bool | Проверка на равенство | x, y |  |  |  |
| is\_lt | bool | Проверка на неравенство a < b | x, y |  |  |  |
| is\_gt | bool | Проверка на неравенство a > b | x, y |  |  |  |
| vec\_prod | double | Расчет смешанного произведения по трем точкам | ax, ay, bx, by, cx, cy |  |  |  |
| is\_heart | bool | Проверка на возможность построения сердца по 4 данным вершинам | tx, ty, lx, ly, dx, dy, rx, ry |  |  |  |
| inside\_heart | bool | Проверка точки на принадлежность заданному сердцу | tx, ty, lx, ly, dx, dy, rx, ry, x, y |  |  |  |
| log\_pts | void | Вывод точек в файл | ti, li, di, ri, arr, size |  | log | tx, ty, lx, ly, dx, dy, rx, ry (переходят в inside\_heart) |
| pts\_inside | unsigned int | Подсчет кол-ва точек внутри сердца | ti, li, di, ri, arr, size |  |  | tx, ty, lx, ly, dx, dy, rx, ry (переходят в inside\_heart) |
| process | void | Функция-обработчик задачи | arr, size |  |  | log, out (переходят в log\_pts) |
| clear\_arr | void | Очистка выделенной под массив памяти | size |  | arr |  |

**Организация ввода-вывода.**

|  |  |  |  |
| --- | --- | --- | --- |
| Библиотека | Имя | Команда | Назначение |
| ifstream | infile | >> | Считывание с файла через поток |
| is\_open | Проверка на наличие открытого файла |
| eof | Проверка на достигнутый конец файла |
| setf | Установка флагов чтения |
| unsetf | Снятие флагов чтения |
| tellg | Возврат текущей позиции файлового указателя |
| seekg | Установка позиции файлового указателя |
| clear | Снятие флагов ошибок |
| close | Закрытие файла |
| ofstream | logfile/outfile | << | Запись в файл через поток |
| close | Закрытие файла |

**Представление алгоритма решения задачи.**

При заходе в main программа пытается открыть входной файл, параллельно записывая в лог-файл шаблон O1. Если файл не был найден, в лог-файл и выходной файл записывается шаблон O2. В противном случае работа продолжается.

В лог-файл записывается шаблон O3 и запускается определение числа точек по следующему алгоритму:

На каждую непустую последовательность предпринимаем попытку записи в тип double. Если не удается, последовательность считается некорректной, файл прогоняется посимвольно до следующей непустой последовательности. Если x не записан, записываем сначала x, также и с y – пока x не записан, y не будет записываться. Если указатель достигает перехода на новую строку, весь порядок записи сбрасывается, а на конце файла цикл и вовсе прерывается.

Как только в строке записывается и x, и y, счетчик инкрементируется, в лог-файл делается очередная запись по шаблону O6. После прогона файла значение счетчика возвращается как потенциальный размер массива.

Если было обнаружено меньше 4 точек, программа не имеет дальнейшего смысла, следовательно, в лог-файл и выходной файл записывается шаблон O4.

В противном случае программа работает дальше.

В лог-файл записывается шаблон O7 и запускается запись точек в массив. Алгоритм идентичен чтению, но с добавлением еще одного момента: если x и y были успешно считаны, запускается прогон по уже записанной части массива с проверкой встречи текущей пары. Если пара не была найдена в массиве, она записывается, счетчик инкрементируется, в лог-файл делается очередная запись по шаблону O9. После прогона файла значение счетчика возвращается как реальный размер массива.

Если реальный размер стал меньше потенциального, в лог-файл записывается шаблон O10 о дублях. Если реальный размер стал недостаточным в ходе отбрасывания дублей, в лог-файл и выходной файл записывается шаблон O11, программа завершает работу.

В противном случае программа продолжает работать.

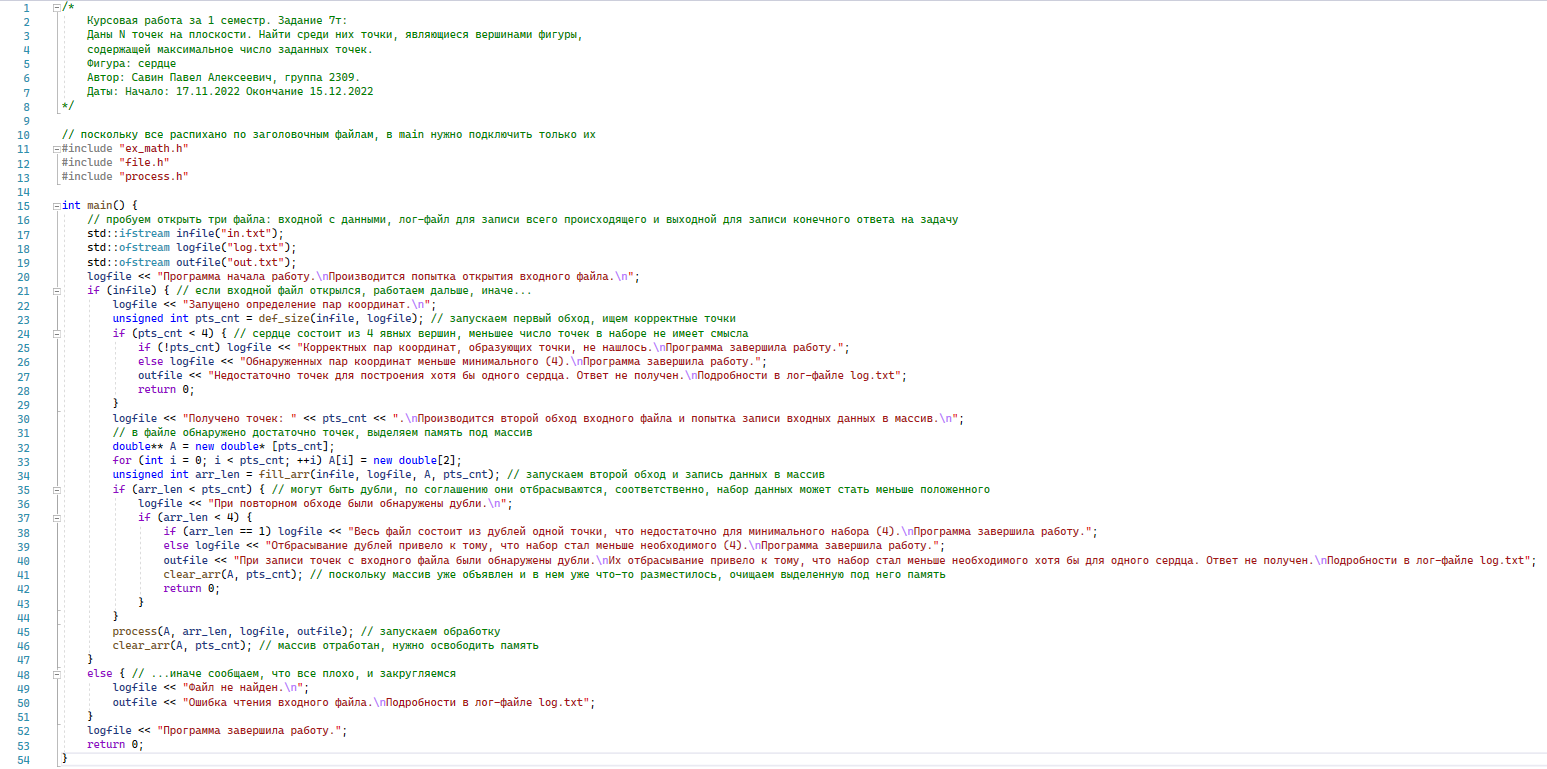
Запускается обработка данных. Описание алгоритма приведено в математической постановке. После прогона сердец возникает проверка: если сердец не было обнаружено или каждое сердце не содержит в себе иных точек, в лог-файл и выходной файл записывается шаблон O14 или O15, соответственно. Если нет, в лог-файл и выходной файл записывается шаблон O16, совершается второй прогон, в котором выводятся только нужные сердца. Программа завершается, очищается память, выделенная под массив, файлы закрываются.

**Блок-схема.**

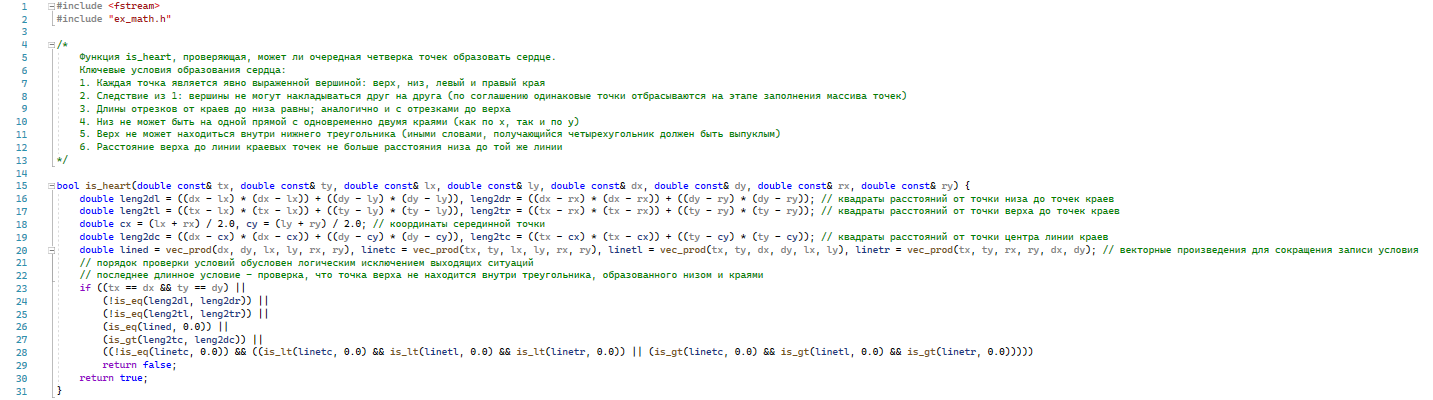
|  |  |  |
| --- | --- | --- |
| is\_eq | is\_lt | is\_gt |
| C:\Users\DOM\Desktop\is_eq.png | C:\Users\DOM\Desktop\is_lt.png | C:\Users\DOM\Desktop\is_gt.png |
| vec\_prod | def\_size | fill\_arr |
| C:\Users\DOM\Desktop\vec_prod.png | C:\Users\DOM\Desktop\def_size.png | C:\Users\DOM\Desktop\fill_arr.png |
| is\_heart | inside\_heart | pts\_inside |
| C:\Users\DOM\Desktop\is_heart_new.png | C:\Users\DOM\Desktop\inside_heart_new.png | C:\Users\DOM\Desktop\pts_inside.png |
| log\_pts | clear\_arr | main |
| C:\Users\DOM\Desktop\log_pts.png | C:\Users\DOM\Desktop\clear_arr_new.png | C:\Users\DOM\Desktop\main.png |
| process | | |
| C:\Users\DOM\Desktop\process1.pngC:\Users\DOM\Desktop\process2.pngC:\Users\DOM\Desktop\process3.pngC:\Users\DOM\Desktop\process4.png | | |

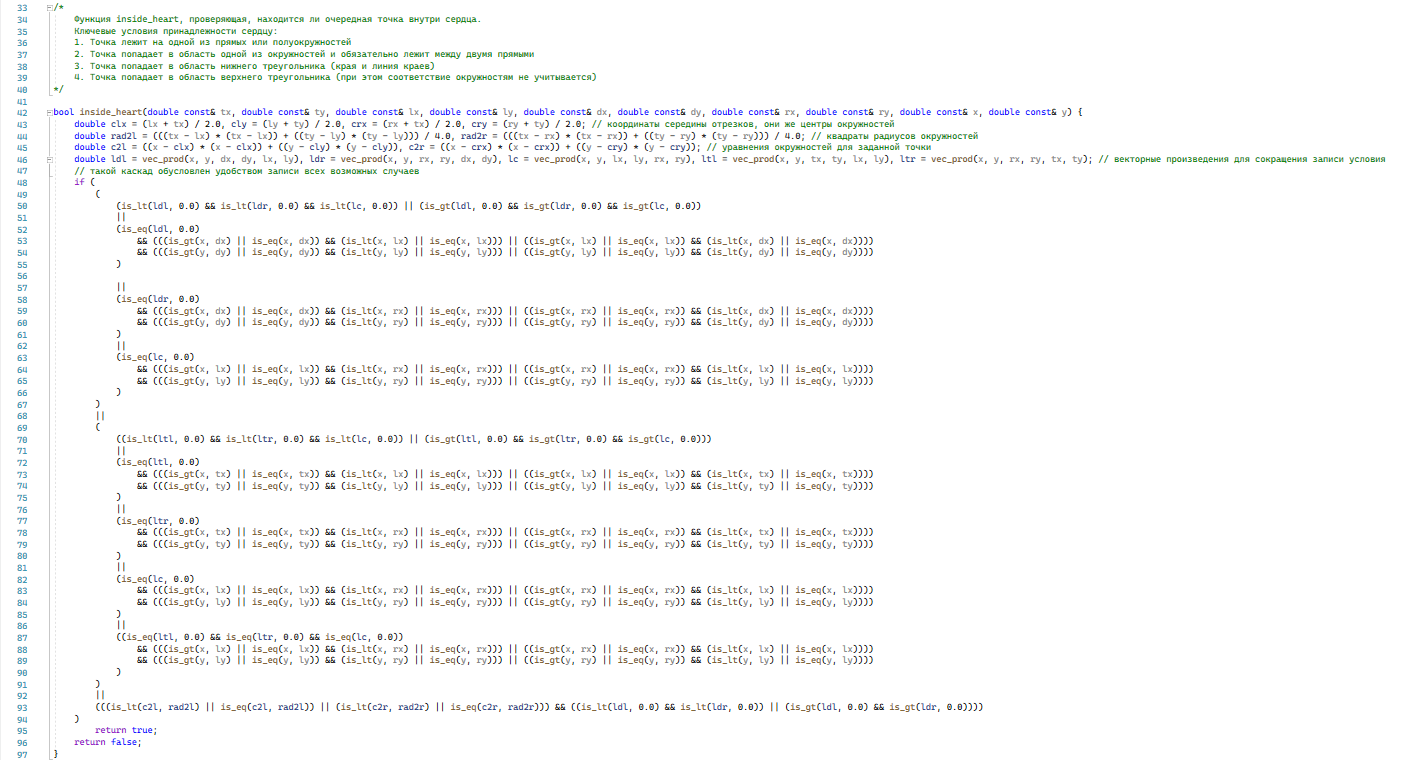
**Текст программы.**

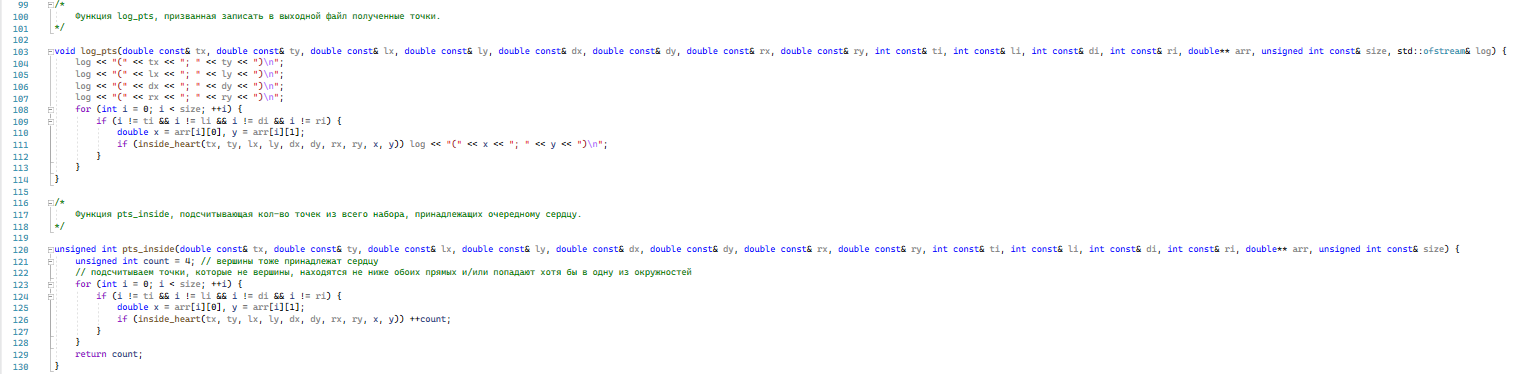
main

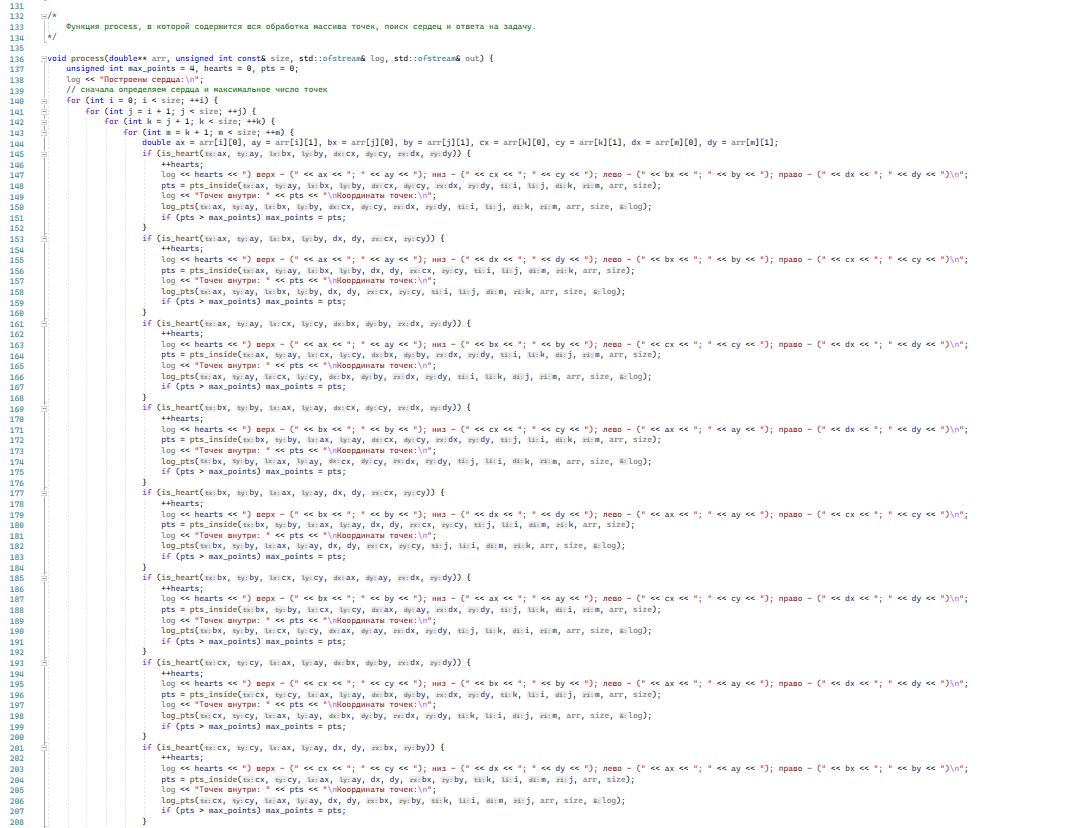
****

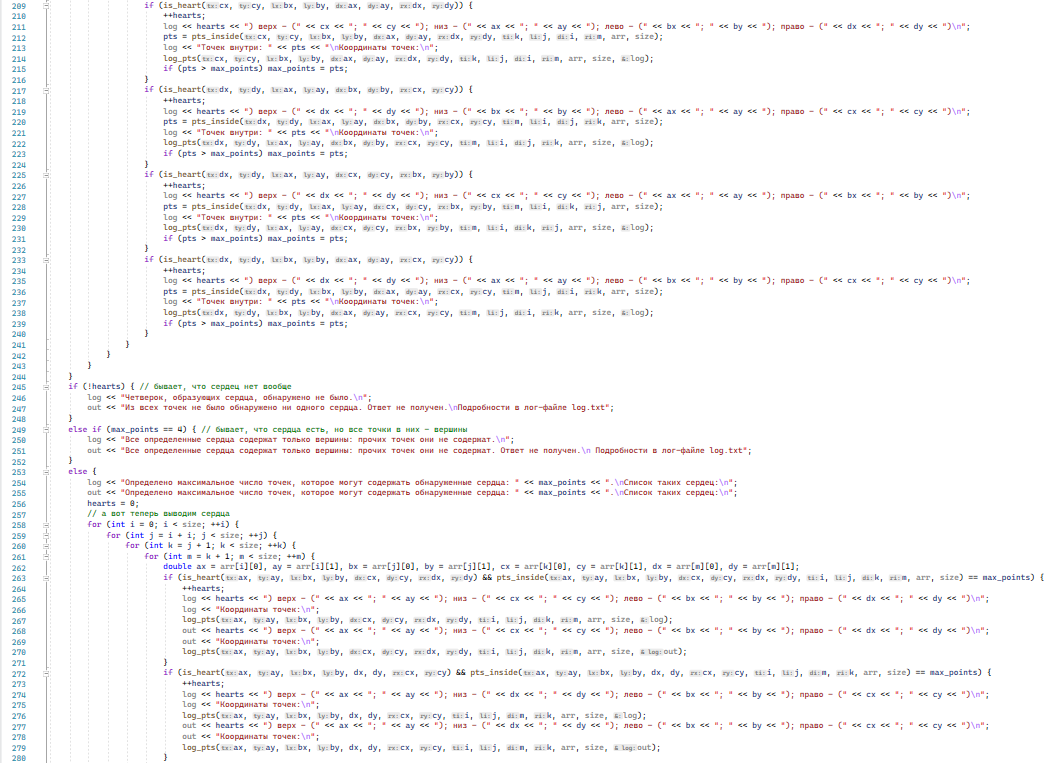
process

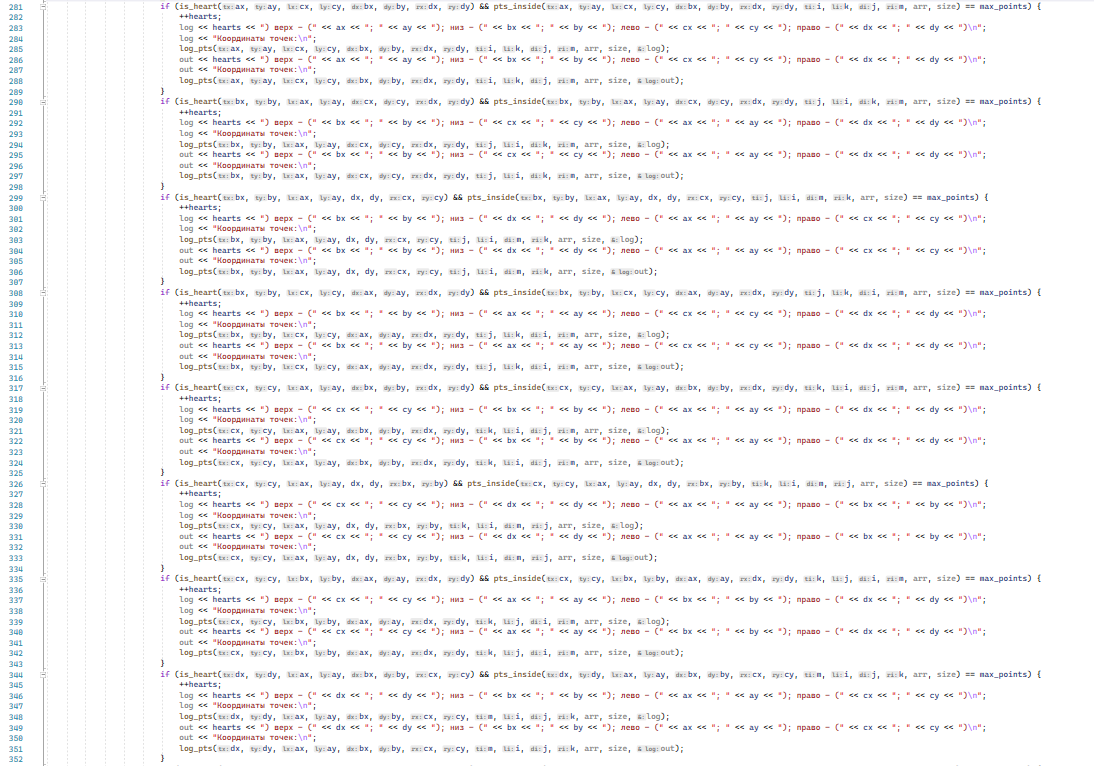


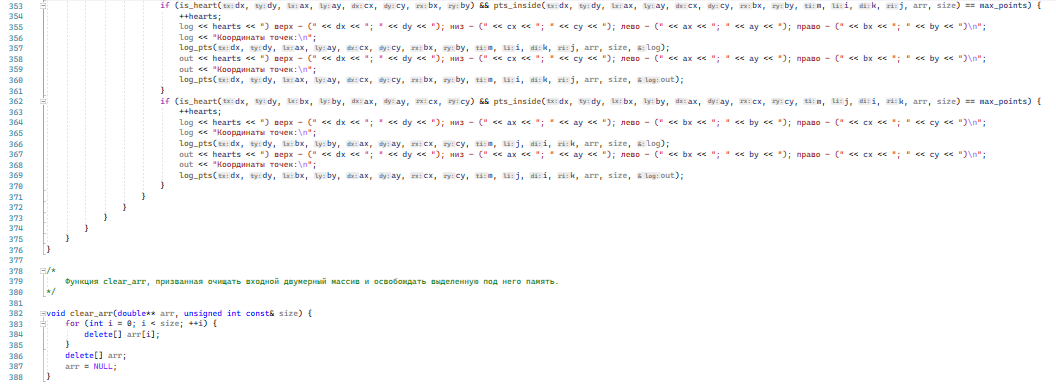




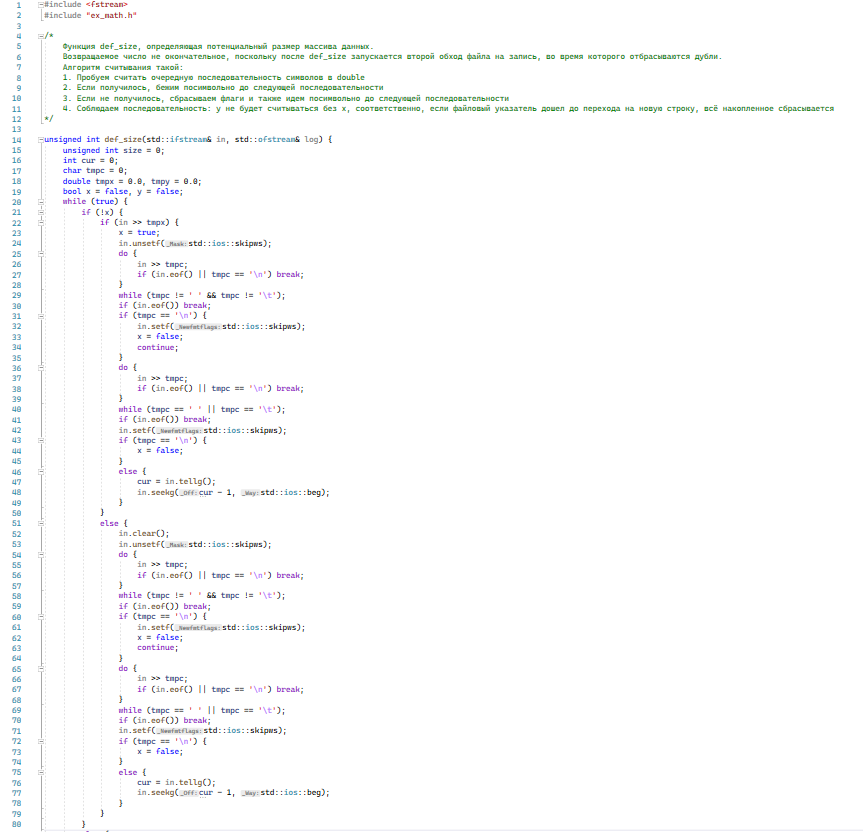


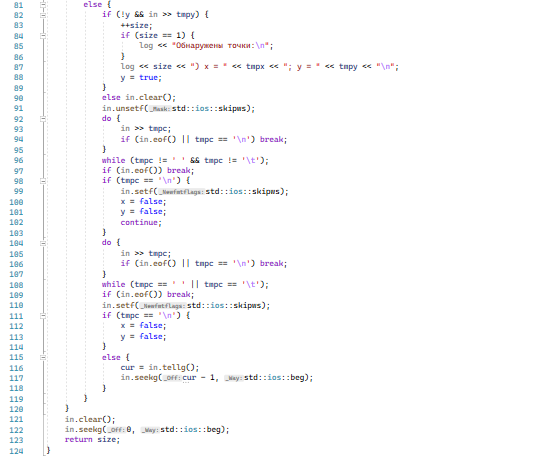


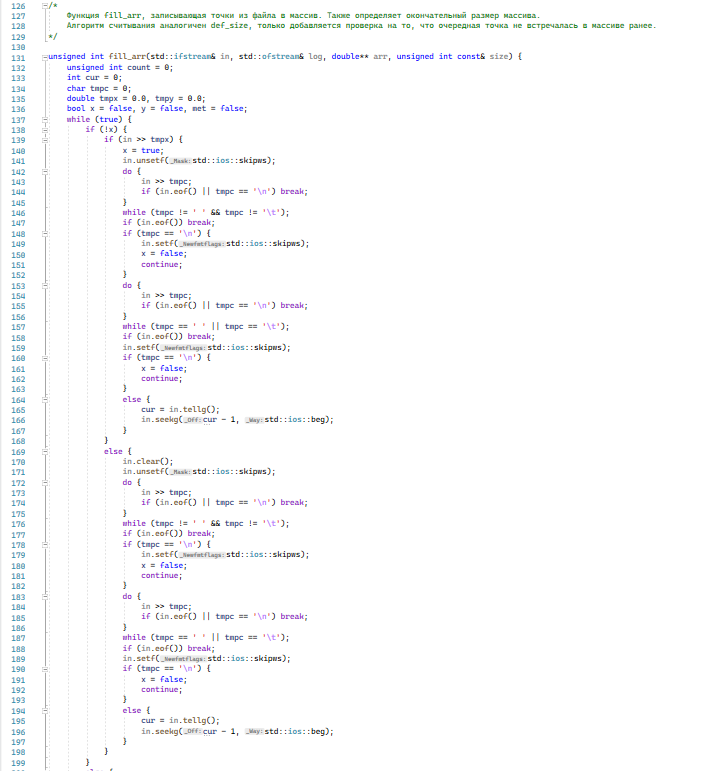




file

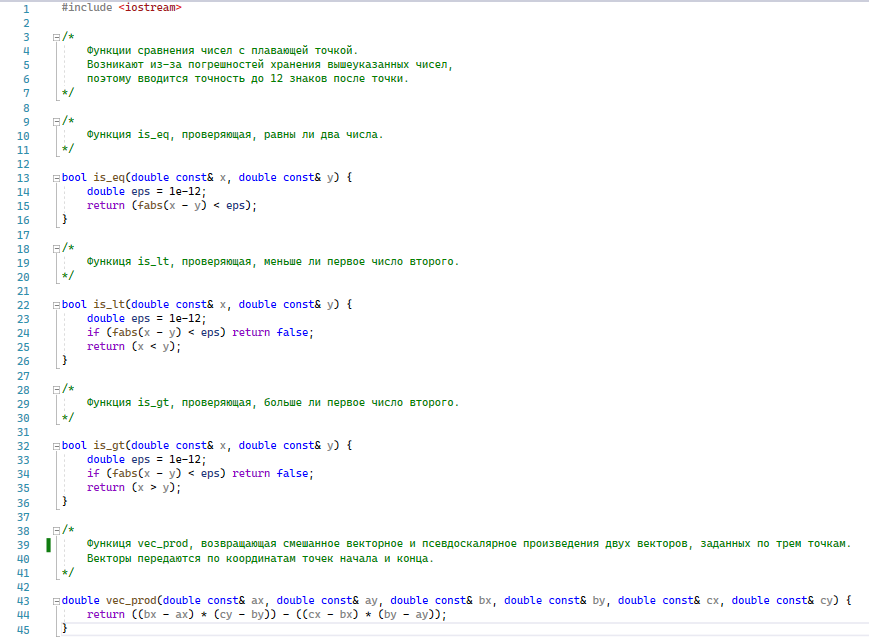






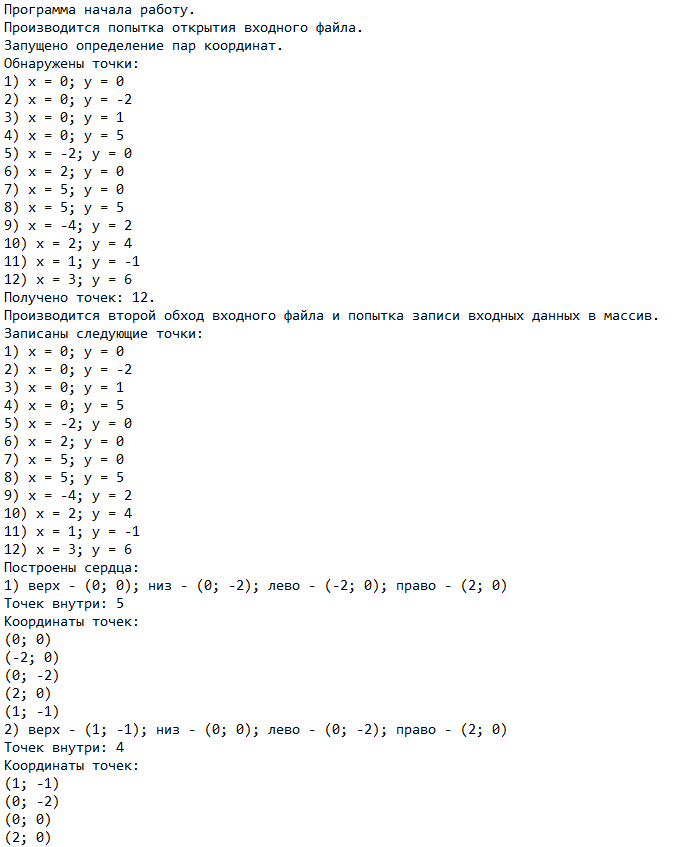


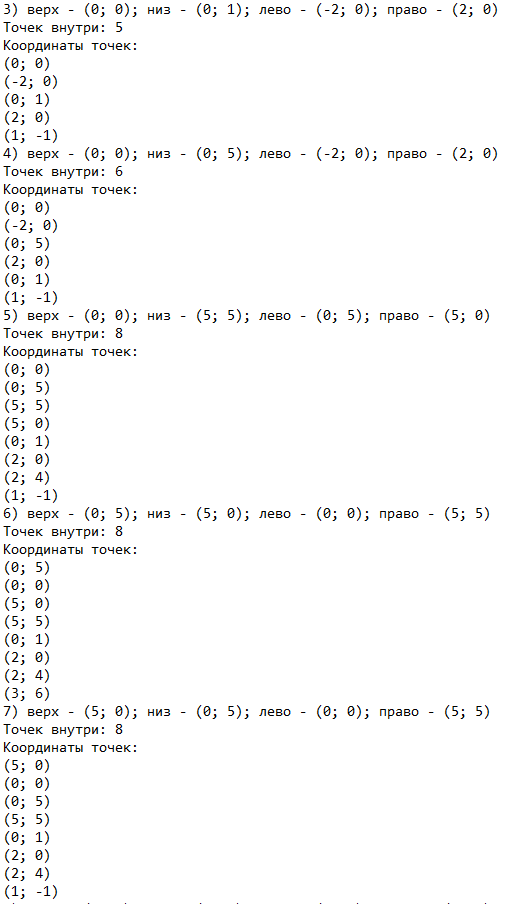
ex\_math

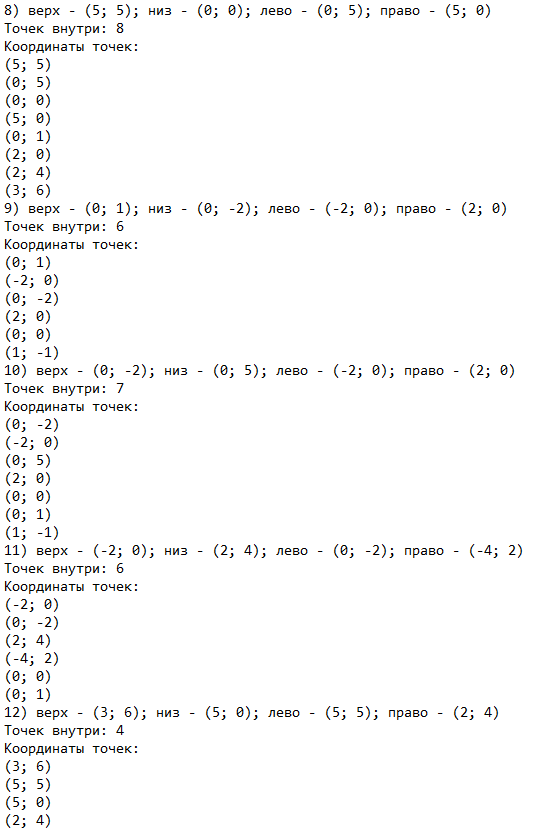


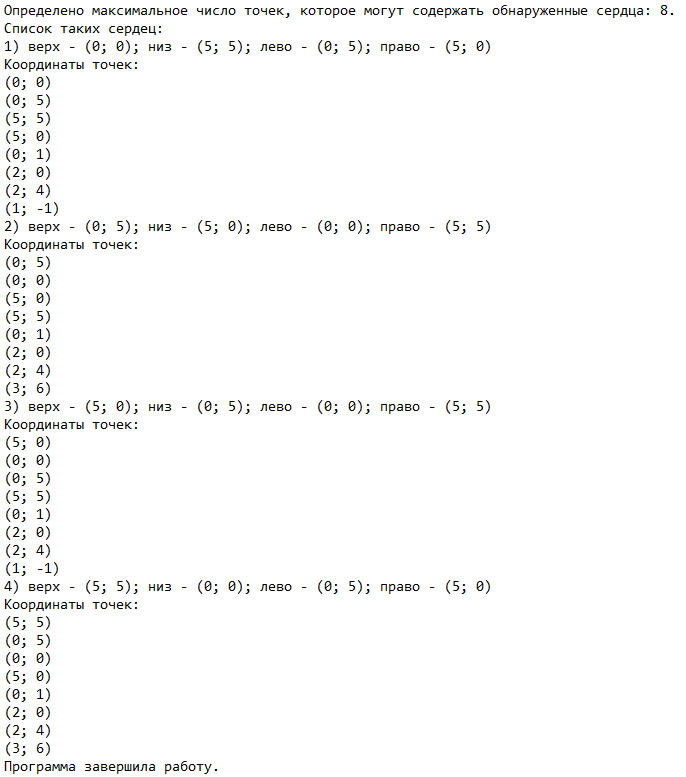
**Результаты работы программы.**

log

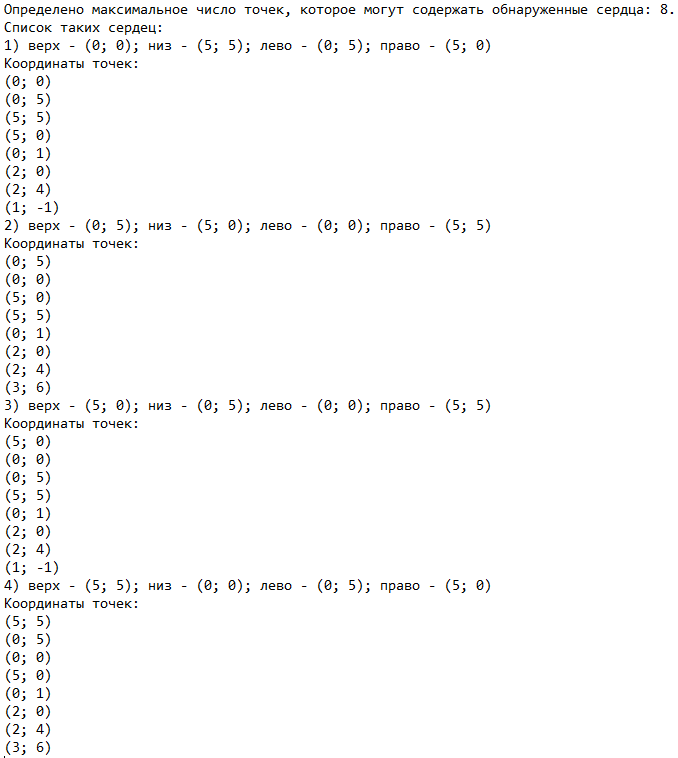
****

****

****

****

out



**Вывод о проделанной работе.**

Проработана аналитическая геометрия с точки зрения решения задач на компьютере.