МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра систем автоматизированного проектирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

«Программирование»

Студент гр. 3353 Карпенко А. Ю.

Преподаватель Калмычков В.А.

Санкт-Петербург

2023

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

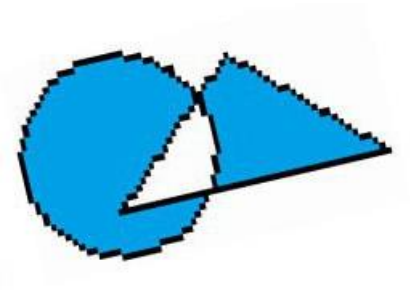
по дисциплине «Программирование»

Студентка: Карпенко А. Ю.

Группа: 3353

Тема работы: Алгоритм поиска заданной фигуры на массиве точек

Исходные данные: Дано N произвольных точек на плоскости. Найти среди них точки, являющиеся вершинами (образующими) фигур с максимальным числом точек в внутренней области.



Содержание пояснительной записки: Аннотация, Анализ задания и выполнение Контрольного примера, Математическая постановка задачи, Особенности выполнения на компьютере, Представление алгоритма решения задачи, Текст программы, Тесты программы

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 14.11.2023

Дата сдачи работы: _____

Дата защиты работы: _____

Студент		Карпенко А.Ю.
Преподаватель		Калмычков В.А

Аннотация

Курсовой проект по дисциплине программирование заключается в написании программы разрешающей задачу поиска определённой фигуры на массиве данных точек и подсчет точек внутри найденной фигуры.

Целью курсовой работы является закрепление и подтверждение навыков и знаний, приобретенных во время семестра

Курсовой проект состоит из программы на языке C++ и пояснительной записки.

Пояснительная записка включает в себя разделы-исходная формулировка, цель математическая постановка, контрольный пример, организация ввода-вывода, организация хранения данных, представление алгоритма решения, текст программы, тесты программы.

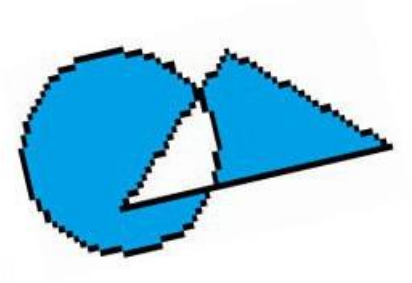
Annotation

The course project in the discipline of programming consists in writing a program that solves the problem of searching for a certain shape on an array of data points and counting points inside the found shape. The purpose of the course work is to consolidate and confirm the skills and knowledge acquired during the semester The course project consists of a C++ program and an explanatory note. The explanatory note includes sections-initial formulation, goal, mathematical formulation, control example, organization of input-output, organization of data storage, presentation of the solution algorithm, program text, program tests

Оглавление:

Исходная формулировка задания

Дано N произвольных точек на плоскости. Найти среди них точки, являющиеся вершинами (образующими) фигур с максимальным числом точек в внутренней области.



Цель работы

Продемонстрировать навыки и знания, приобретенные в течении семестра в том числе: умение хранить и обрабатывать числовые данные на основе файлов и массивов, вложенные циклы, ветвление и тд.

Математическая постановка задания

В первую очередь требуется разрешить задачу возможности построения фигуры на данных точках. С последующим решением задачи количества принадлежащих точек фигуре. Разрешив эти задачи для всех комбинаций точек, можно разрешить задачу в начальной формулировке.

Для решения задачи возможности построения фигуры необходимы 6 точек, например, A B C D G R.

По первым трем точкам будет строиться окружность. Для этого необходимо проверить, что точки не лежат на одной прямой.

Лежат ли точки с координатами $(x_1; y_1), (x_2; y_2), (x_3; y_3)$ на одной прямой, можно проверить по формуле определителя матрицы:

$$x_1(y_2 - y_3) - x_2(y_1 - y_3) + x_3(y_1 - y_2)$$

Если данное выражение равно нулю, то точки лежат на одной прямой. Если полученное значение отлично от нуля, то не лежат и можно построить окружность или треугольник.

Итак, по точкам A B C построена окружность (рис. 1).

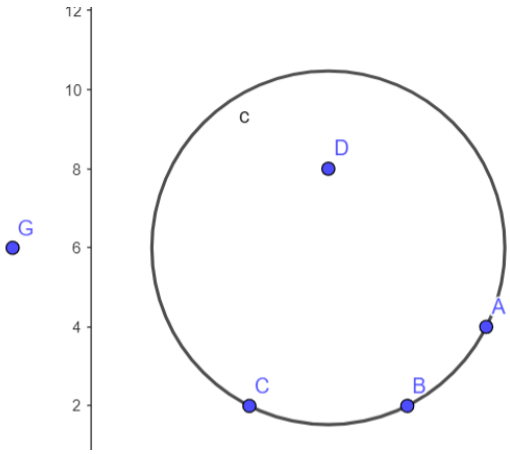


Рис.1 – построена окружность

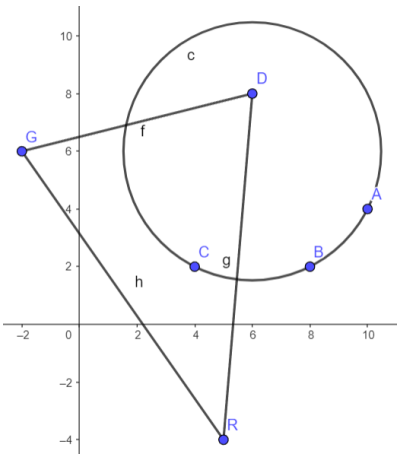


Рис. 2 – построен треугольник

По второй тройке точек строится треугольник (рис.2). Для этого также идет проверка, что точки не лежат на одной прямой.

Чтобы получилась фигура, одна из вершин треугольника должна находиться внутри окружности, а две другие – вне. Для этого необходимо вычислить центр и радиус окружности.

Нахождение центра окружности. Вычислим вспомогательные значения A, B, C, D, E, F и подставим в конечные уравнения

A = x1 - x2

B = y1 - y2

C = x1 - x3

D = y1 - y3

$$E = \frac{(x1 * x1 - x2 * x2) + (y1 * y1 - y2 * y2)}{2}$$

$$F = \frac{(x1 * x1 - x3 * x3) + (y1 * y1 - y3 * y3)}{2}$$

Центр окружности с координатами (h, k)

h = (D * E - B * F) / (A * D - B * C)

k = (A * F - C * E) / (A * D - B * C);

Радиус находится по следующей формуле:

$$\text{radius} = \sqrt{(x1 - h) * (x1 - h) + (y1 - k) * (y1 - k)}$$

Далее вычисляем расстояние от вершины треугольника (x, y) до центра окружности (h, k) по формуле

$$\sqrt{(x - h)^2 + (y - k)^2}$$

Если оно меньше радиуса, то вершина находится внутри окружности. Если больше – вне окружности.

Когда фигура построена, происходит проверка оставшихся точек. Они должны принадлежать области внутри треугольника или окружности (исключается область, которая принадлежит и треугольнику, и окружности одновременно).

Проверка принадлежности точки области внутри окружности:

Считается расстояние от точки до центра окружности по уже известной формуле $\sqrt{(x-h)^2+(y-k)^2}$. Если значение меньше радиуса, то точка находиться внутри окружности, если больше – вне.

Проверка принадлежности точки области внутри треугольника происходит с помощью вычислений площадей по формуле Герона. Проверяемая точка образует с вершинами исходного треугольника 3 маленьких треугольника. По формуле Герона вычисляются их площади.

Проверяемая точка $(x_1; y_1)$, вершины треугольника $(x_2; y_2), (x_3; y_3)$

Стороны треугольника вычисляются по формулам

$$side1 = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

$$side2 = \sqrt{(x_3-x_2)^2+(y_3-y_2)^2}$$

$$side3 = \sqrt{(x_1-x_3)^2+(y_1-y_3)^2}$$

Полупериметр: $\frac{side1+side2+side3}{2}$

Площадь: $\sqrt{s(s-side_1)(s-side_2)(s-side_3)}$

Если точка лежит внутри исходного треугольника, то сумма площадей маленьких будет равна площади исходного. Если же она лежит вне, то сумма будет больше.

Точка принадлежит нужной области, если она принадлежит либо области внутри окружности, либо внутри треугольника.

Контрольный пример

Возьмем следующий набор точек: (10, 4) (8, 2) (-2, 6) (4, 2) (5, -4) (-1, 0) (6, 8)

Из них можно построить 48 фигур.

Проверим одну из образующихся фигур.

Построим окружность по точкам: (10, 4)(8, 2)(-2, 6)

Построим треугольник по точкам: (4, 2)(5, -4)(-1, 0)

Проверим принадлежность точки (6, 8)

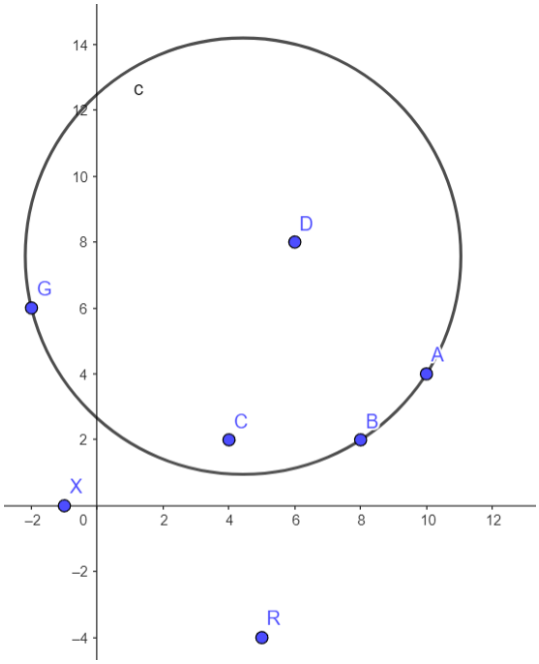
Введем буквенное обозначение точек:

A(10, 4) B(8, 2) G(-2, 6) C(4, 2) R(5, -4) X(-1, 0) D(6, 8)

Чтобы построить окружность по точкам A B G, убедимся, что они не лежат на одной прямой.

$$10(2-6)-8(4-6)-2(4-2)=\ -28$$

Значение отлично от 0, значит окружность можно построить.



Найдем ее центр и радиус

(10, 4)(8, 2)(-2, 6)

x1 - x2=10-8=2

y1 - y2=4-2=2

x1 - x3=10+2=12

y1 - y3=4-6=-2

$$\frac{(x_1^2\ -x_2^2\ )\ +\ (y_1^2-y_2^2)}{2}=\frac{(10^2-8^2)+(4^2-2^2)}{2}=24$$

$$\frac{(x_1^2\ -x_3^2\ )\ +\ (y_1^2-y_3^2)}{2}=\frac{(10^2-(-2)^2)+(4^2-6^2)}{2}=38$$

Центр окружности с координатами (h, k)

h = $\frac{(-2)*24-2*38}{2*(-2)-2*12}$=4,4286

k = $\frac{2*38-12*24}{2*(-2)-2*12}$ =7,5714

Радиус находится по следующей формуле:

$$\text{radius} = \sqrt{(10-4{,}4286)^2+(4-7{,}5714)^2} = 6{,}6178$$

для того, чтобы построить треугольник, проверим, не лежат ли точки на одной прямой

4(-4-0)-5(2-0)-1(2+4) = -32

Значение отлично от нуля, значит можно построить треугольник.

Проверим нахождение вершин треугольника относительно окружности.

Точка C: $\sqrt{(4-4{,}4286)^2+(2-7{,}5714)^2} = 5.5879 < 6{,}6178$

лежит внутри окружности

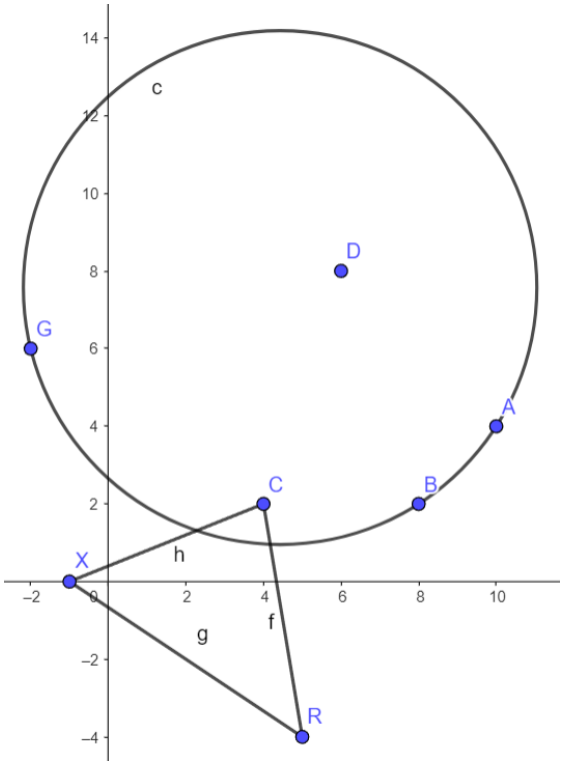
Точка R: $\sqrt{(5-4{,}4286)^2+(-4-7{,}5714)^2} = 11.5855 > 6{,}6178$

Лежит вне окружности

Точка X: $\sqrt{(-1-4{,}4286)^2+(0-7{,}5714)^2} = 9.3164 > 6{,}6178$

Лежит вне окружности

Соответственно, можно построить фигуру.

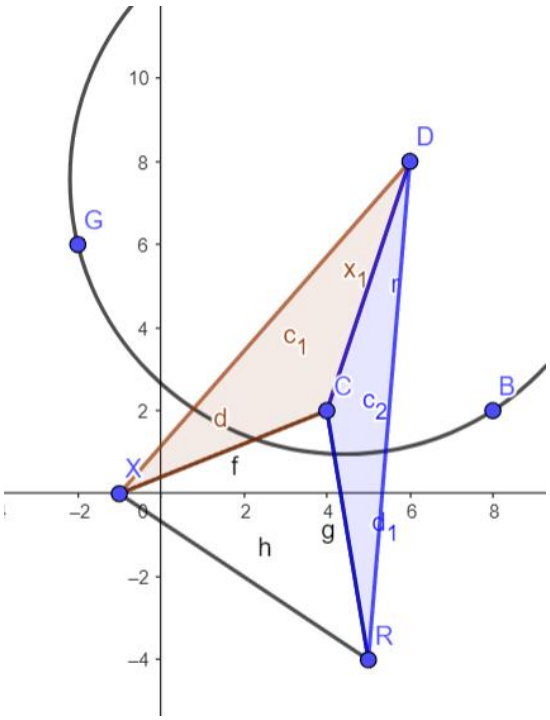


Проверим принадлежность точки D (6, 8) окружности

$$\sqrt{(6-4{,}4286)^2+(8-7{,}5714)^2} = 1.6288 < 6{,}6178$$

Следовательно, принадлежит.

Далее проверка принадлежности треугольнику:



Рассчитаем площадь треугольника XDC

C(4, 2) X(-1, 0) D(6, 8)

Стороны треугольника вычисляются по формулам

$$side1 = \sqrt{(-1-4)^2 + (0-2)^2} = 5{,}3851$$

$$side2 = \sqrt{(6+1)^2 + (8-0)^2} = 10{,}6302$$

$$side3 = \sqrt{(4-6)^2 + (2-8)^2} = 6{,}3246$$

Полупериметр: $\frac{5{,}3851+10{,}6302+6{,}3246}{2} = 11{,}17$

Площадь: $\sqrt{11{,}17(11{,}17-5{,}3851)(11{,}17-10{,}6302)(11{,}17-6{,}3246)}$= 13

Аналогично вычислим остальные площади

Площадь треугольника RCD с координатами C(4, 2) R(5, -4) D(6, 8) = 9

Площадь треугольника XDR с координатами R(5, -4) X(-1, 0) D(6, 8)= 38

Площадь треугольника XCR с координатами C(4, 2) R(5, -4) X(-1, 0)= 16

$$S_{XDC} + S_{RCD} + S_{XDR} = 13 + 9 + 38 = 60 > 16 = S_{XCR}$$

Следовательно, точка D не принадлежит треугольнику XCR.

Итог: точка принадлежит окружности, но не треугольнику, соответственно у данной фигуры есть 1 точка, которая принадлежит нужной области.

Организация ввода-вывода

1) Чтение массива из файла:

Организовано путем взятия каждого элемента функцией оператором извлечения >>

В первую очередь снимается элемент в первой строке несущие желаемый размер. Потом через алгоритм анализа элементов, снимается значения элементов массива начиная со второй строки и в последующих в структуру Point

Ожидаемый формат файла:

$$ddd$$

$$ddd_{x1}ddd_{y1}$$

$$ddd_{x2}ddd_{y2}$$

... ...

$$ddd_{xn}ddd_{yn}$$

Кроме того, после точек может присутствовать текст.

2) Вывод информации в файл протокола:
вывод считанных точек

$$(ddd_{x1}, ddd_{y1})$$
$$(ddd_{x2}, ddd_{y2})$$
... ...
$$(ddd_{xn}, ddd_{yn})$$

Вывод: Количество считанных точек: ddd - сколько удалось снять точек

Далее следует перебор

Вывод: Фигура: ddd строится из точек: -(ddd,ddd) (ddd,ddd) (ddd,ddd) (ddd,ddd) (ddd,ddd) (ddd,ddd)

далее выводится подстановка точек:

не принадлежит (ddd,ddd) - если точка не принадлежит

принадлежит (ddd,ddd) - если точка принадлежит

вывод: Максимальное число точек внутри фигуры: ddd

вывод: Фигура: ddd строится из точек-(ddd,ddd) (ddd,ddd) (ddd,ddd) (ddd,ddd) (ddd,ddd) (ddd,ddd)

принадлежит точка (ddd,ddd)

количество фигур с максимальным числом точек внутри фигуры: ddd

3) Вывод информации в итоговый файл:

Вывод:

Карпенко Анастасия

группа 3353

Даны N произвольных точек на плоскости. Найти среди них точки, которые образуют фигуру с максимальным количеством точек внутри области

вывод снятых из входного файла точек

вывод считанных точек

$$(ddd_{x1}, ddd_{y1})$$
$$(ddd_{x2}, ddd_{y2})$$
$$\dots\ \dots$$
$$(ddd_{xn}, ddd_{yn})$$

Вывод: Количество считанных точек: ddd - сколько удалось снять точек

вывод: Максимальное число точек внутри фигуры: ddd

вывод: Фигура: ddd строится из точек-(ddd,ddd) (ddd,ddd) (ddd,ddd) (ddd,ddd) (ddd,ddd) (ddd,ddd)

принадлежит точка (ddd,ddd)

количество фигур с максимальным числом точек внутри фигуры: ddd

4) Обработка некорректных ситуаций из входного файла:
   1) Цифр в строке больше 2 - Ввод 2 цифр - Вывод «Лишние значения»
   2) Цифр в строке 1 - Ввод *пропуск строки*-Вывод «недостаточно данных для точки»
   3) Пустая строка 1-Ввод *пропуск строки*-Вывод «Пустая строка»
   4) Текст после цифр - Ввод - Ввод 2 цифр - Вывод «Лишние значения» *перенос текса*
   5) Пустой файл -Вывод «Файл пуст»
   6) Файл не открыт-Вывод «Файл не открыт»

Для ввода из файла используем библиотеку fstream

(название потока)>>(имя переменной);

Для вывода в файл используем библиотеку fstream

(название потока)<<«Текст»<<(имя переменной);

Так же стоит упомянуть, что для работы с файлами потребуется о открывать потоки вывода и открыть файлы, а так же:

fstream (название потока) ;

(название потока).open(«(Имя файла)», ios::(in для ввода, out для ввывода и app для дополнения));

(название потока)..close();

Особенности выполнения на компьютере

Тип float представляет вещественное число с плавающей точкой в диапазоне от -3.4E-38 до 3.4E38 и в памяти занимает 4 байта (32 бита), что накладывает свои ограничения на величину переменные этого типа и на элементы массивов в частности . Тип переменной int, который хранит целочисленные значения в диапазоне от –32768 до 32767 и занимает 2 байта (16 бит) накладывает свои огорчения на переменные этого типа.

В процессе выполнения программы потребуется реализация условных конструкций:

if (условие 1) {(выполняется если условие 1 истина);}

else if(условие 2) {выполняется если условие 2 истина;}

else {выполняется если условие 1 И условие 2 ложь;};

А так же потребуется циклы, будем использовать циклы:

while(условие провидения цикла)

{(тело цикла)}

for(начальные условия; условия продолжения; действие после завершения одного тела)

{(тело цикла)}

Организация хранения данных

Имя функции	Назначение	параметры				Возвращаемое значение	Внешние эффекты
		входные	выходные	модифицируемые	транзитные		
equals	Проверка погрешности	float x, float y	-	-	-	bool	-
arePointsCollinear	Проверка, что точки не лежат на одной прямой	Point p1, Point p2, Point p3	-	-	-	bool	-
convert_to_double	Преобразование считанной строки в число	const std::string& str	-	-	-	double result	
read_point_from_line	Считывание данных из входного файла	string& input, ofstream& log_file	-	-	-	result	Считывание данный, вывод комментарие в в файл протокола
circleFromPoints	Метод для построения окружности по трем точкам	Point p1, Point p2, Point p3	-	-	-	Circle	-

get_line	Анализ считанной строки	fstream& inpu				string	
process	Основной процесс	fstream& input, ofstream& log_file, ofstream& result	-	-	-	-	Вывод получившихся фигур и точек в файл протокола и результата

Структура для представления точки

```
struct Point {
    float x;
    float y;
```

функции, представленные в данной структуре

Имя функции	Назначение	параметры		Возвращаемое значение	Внешние эффекты
		входные	выходные		
distanceTo	Метод для вычисления расстояния между двумя точками	const Point& other	-	float	-
isOnSegment	Метод для проверки, лежит ли точка на отрезке	const Point& start, const Point& end	-	bool	-
isInf	Проверка погрешности значения точек	-	-	bool	-
isSame	Проверка погрешности значения точек	-	-	bool	-

Структура для представления окружности

```
struct Circle {
    Point center;
    float radius;
    Point points[3];
    Circle() : center(), radius(1.0f) {
        for (int i = 0; i < 3; ++i) {
            points[i] = Point();
        }
    }
```

функции, представленные в данной структуре

Имя функции	Назначение	параметры		Возвращаемое значение	Внешние эффекты
		входные	выходные		
isPointInside	Метод для проверки, находится ли точка внутри окружности	Point point	-	bool	-
isPointOutside	Метод для проверки, находится ли точка вне окружности	Point point	-	bool	-
contains	Метод для проверки погрешности точек окружности	Point point	-	bool	-

Структура для представления треугольника

```
struct Triangle {
    Point vertex1;
    Point vertex2;
    Point vertex3;
    Triangle() : vertex1(), vertex2(), vertex3() {}
```

Функции, представленные в данной структуре

Имя функции	Назначение	параметры		Возвращаемое значение	Внешние эффекты
		входные	выходные		
isPointOnEdges	Метод для проверки, лежит ли точка на рёбрах треугольника	Point point	-	bool	-
isPointOutside	Метод для определения, лежит ли точка внутри треугольника	Point point	-	bool	-
isPointOutside	Метод для проверки, находится ли точка вне треугольника	Point point	-	bool	-
contains	Метод для проверки погрешности точек – вершин треугольника	Point point	-	bool	-
area	Метод для вычисления площади треугольника по формуле Герона	vertex1(), vertex2(), vertex3()	-	float	-

Структура для представления фигуры

```
struct Figure {
    Circle circle;
```

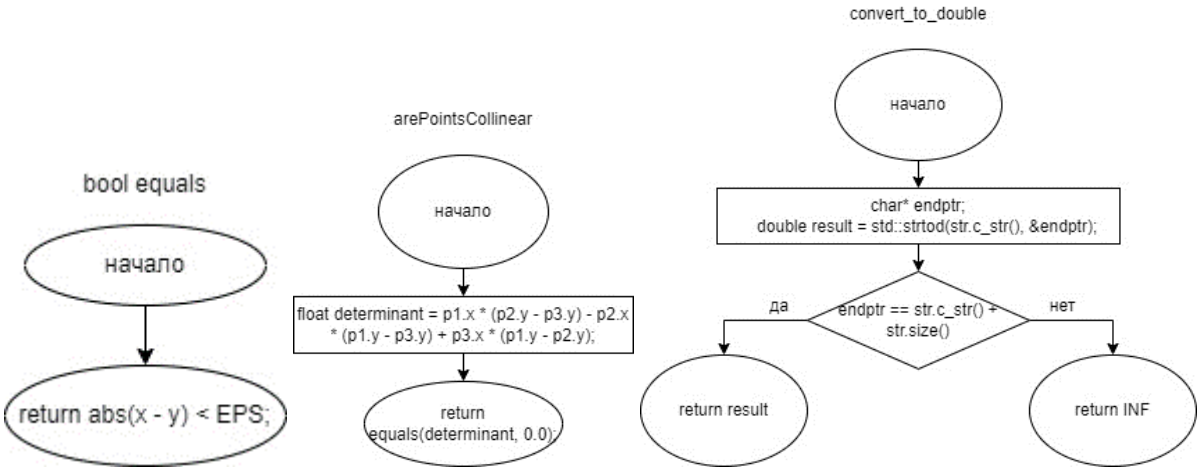
```
    Triangle triangle;
// Конструктор по умолчанию для Figure
    Figure(): circle(), triangle() {}
    // Конструктор для Figure
    Figure(Circle c, Triangle t) : circle(c), triangle(t) {}
```

функции, представленные в данной структуре

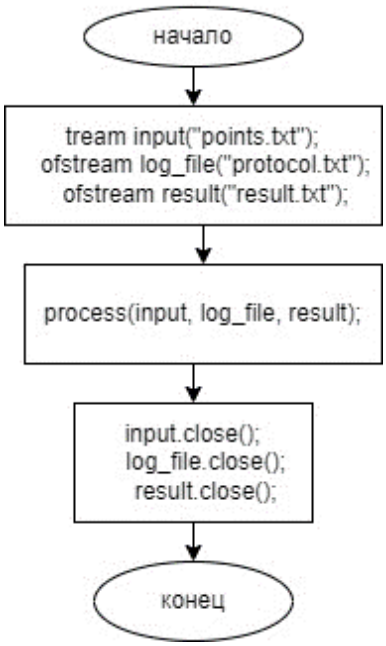
Имя функции	Назначение	параметры		Возвращаемое значение	Внешние эффекты
		входные	выходные		
isPointInside	Проверка принадлежности точки фигуре	Point point	-	bool	-
contains	Проверка погрешности точек фигуры	Point point	-	bool	-

Представление алгоритма решения задачи

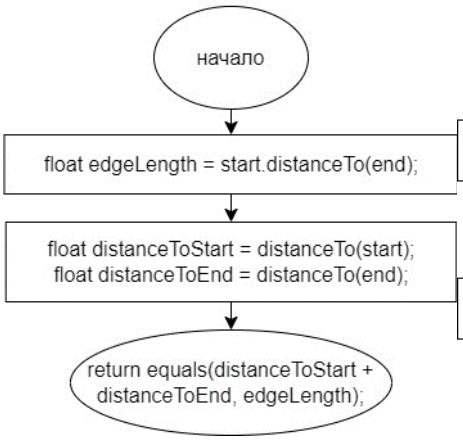
В первую очередь происходит считывание массива точек, потом методом перебора точек и проверок возможности находим фигуры и достраиваем их, считаем количество принадлежащих им точек, в процессе каждого действия формируем файл процесса, после завершения всех постановок формируем файл вывода.

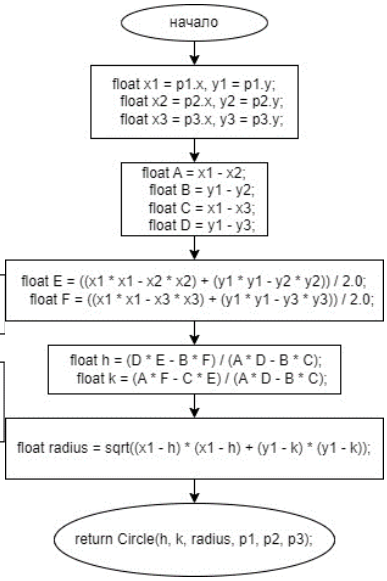


main

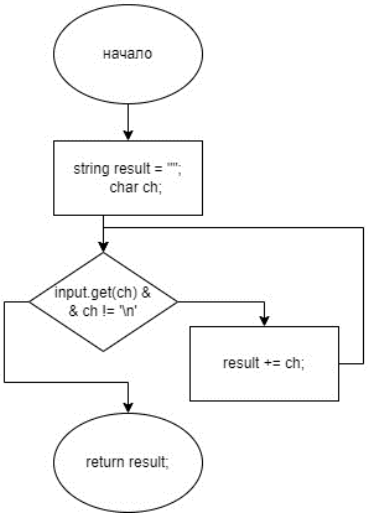


isOnSegment

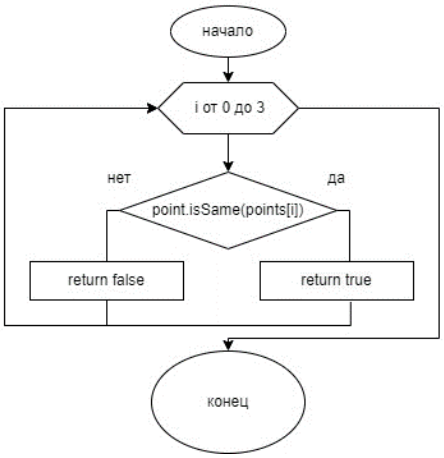




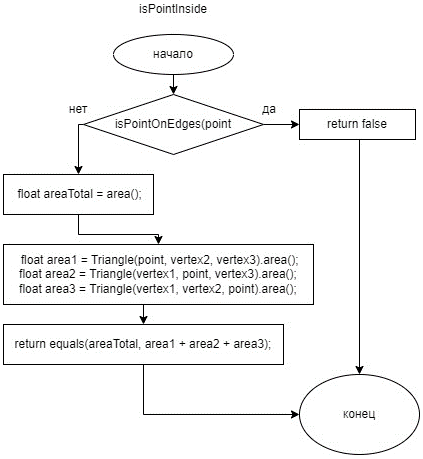
get_line

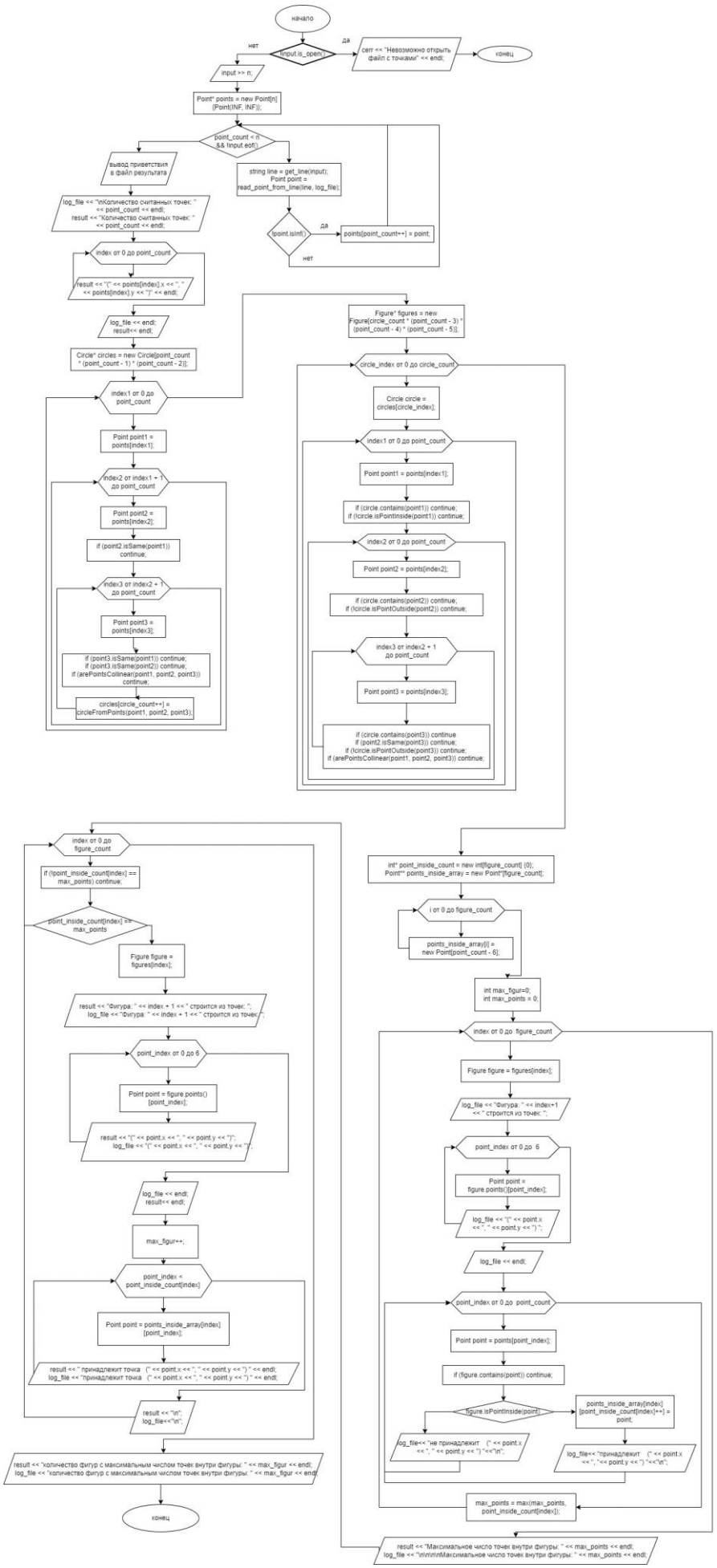


contains



isPointInside




начало
input.is_open()
нет
да
cerr << "Невозможно открыть файл с точками" << endl;
конец
input >> n;
Point* points = new Point[n] (Point(){INF, INF});
point_count < n && !input.eof()
вывод приветствия в файл результата
string line = get_line(input); Point point = read_point_from_line(line, log_file);
!point.isInf()
да
нет
points[point_count++] = point;
log_file << "\nКоличество считаных точек: " << point_count << endl; result << "Количество считаных точек: " << point_count << endl;
index от 0 до point_count
result << "(" << points[index].x << ", " << points[index].y << ")" << endl;
log_file << endl; result++ endl;
Circle* circles = new Circle[point_count * (point_count - 1) * (point_count - 2)];
index1 от 0 до point_count
Point point1 = points[index1];
index2 от index1 + 1 до point_count
Point point2 = points[index2];
if (point2.isSame(point1)) continue;
index3 от index2 + 1 до point_count
Point point3 = points[index3];
if (point3.isSame(point1)) continue; if (point3.isSame(point2)) continue; if (arePointsCollinear(point1, point2, point3)) continue;
circles[circle_count++] = circleFromPoints(point1, point2, point3);
Figure* figures = new Figure[circle_count * (point_count - 3) * (point_count - 4) * (point_count - 5)];
circle_index от 0 до circle_count
Circle circle = circles[circle_index];
index1 от 0 до point_count
Point point1 = points[index1];
if (circle.contains(point1)) continue; if (!circle.isPointInside(point1)) continue;
index2 от 0 до point_count
Point point2 = points[index2];
if (circle.contains(point2)) continue; if (!circle.isPointOutside(point2)) continue;
index3 от index2 + 1 до point_count
Point point3 = points[index3];
if (circle.contains(point3)) continue if (point2.isSame(point3)) continue; if (!circle.isPointOutside(point3)) continue; if (arePointsCollinear(point1, point2, point3)) continue;
int* point_inside_count = new int[figure_count] {0}; Point** points_inside_array = new Point*[figure_count];
i от 0 до figure_count
points_inside_array[i] = new Point[point_count - 6];
int max_figur=0; int max_points = 0;
index от 0 до figure_count
Figure figure = figures[index];
log_file << "Фигура: " << index+1 << " строится из точек: ";
point_index от 0 до 6
Point point = figure.points()[point_index];
log_file << "(" << point.x << ", " << point.y << ") ";
log_file << endl;
point_index от 0 до point_count
Point point = points[point_index];
if (figure.contains(point)) continue;
figure.isPointInside(point)
points_inside_array[index][point_inside_count[index]++] = point;
log_file<< "не принадлежит (" << point.x << ", " << point.y << ") "<<"\n";
log_file<< "принадлежит (" << point.x << ", "<< point.y << ") "<<"\n";
max_points = max(max_points, point_inside_count[index]);
result << "Максимальное число точек внутри фигуры: " << max_points << endl; log_file << "\n\n\nМаксимальное число точек внутри фигуры: " << max_points << endl;
index от 0 до figure_count
if (!point_inside_count[index] == max_points) continue;
point_inside_count[index] == max_points
Figure figure = figures[index];
result << "Фигура: " << index + 1 << " строится из точек: "; log_file << "Фигура: " << index + 1 << " строится из точек: ";
point_index от 0 до 6
Point point = figure.points()[point_index];
result << "(" << point.x << ", " << point.y << ")"; log_file << "(" << point.x << ", " << point.y << ") ";
log_file << endl; result<< endl;
max_figur++;
point_index < point_inside_count[index]
Point point = points_inside_array[index][point_index];
result << " принадлежит точка (" << point.x << ", " << point.y << ") " << endl; log_file << "принадлежит точка (" << point.x << ", " << point.y << ") " << endl;
result << "\n"; log_file<<"\n";
result << "количество фигур с максимальным числом точек внутри фигуры: " << max_figur << endl; log_file << "количество фигур с максимальным числом точек внутри фигуры: " << max_figur << endl;
конец

Текст программы

```
#include <iostream>
#include <sstream>
#include <string>
#include <cmath>
#include <fstream>
using namespace std;
const float INF = 1e+10;
const float EPS = 1e-6;
bool equals(float x, float y) {
    return abs(x - y) < EPS;
};
// Структура для представления точки
struct Point {
    float x;
    float y;
    // Конструктор по умолчанию
    Point() : x(0.0f), y(0.0f) {}
    // Конструктор с параметрами
    Point(float xCoord, float yCoord) : x(xCoord), y(yCoord) {}
    // Метод для вычисления расстояния между двумя точками
    float distanceTo(const Point& other) const {
        return sqrt(pow(x - other.x, 2) + pow(y - other.y, 2));
    }
    // Метод для проверки, лежит ли точка на отрезке
    bool isOnSegment(const Point& start, const Point& end) {
        // Вычисляем длину ребра
        float edgeLength = start.distanceTo(end);
        // Вычисляем расстояние от точки до начальной и конечной точек ребра
        float distanceToStart = distanceTo(start);
        float distanceToEnd = distanceTo(end);
        // Проверяем, что расстояние от крайних точек ребра до точки равно длине ребра
        return equals(distanceToStart + distanceToEnd, edgeLength);
    }
    bool isInf() {
        return equals(x, INF) && equals(y, INF);
    }
    bool isSame(Point point) {
        return equals(x, point.x) && equals(y, point.y);
    }
};
//Проверка, что точки не лежат на одной прямой
bool arePointsCollinear(Point p1, Point p2, Point p3) {
    // Определитель матрицы
    float determinant = p1.x * (p2.y - p3.y) - p2.x * (p1.y - p3.y) + p3.x * (p1.y - p2.y);
    // Точки лежат на одной прямой, если определитель равен нулю
    return equals(determinant, 0.0);
}
// Структура для представления окружности
struct Circle {
    Point center;
    float radius;
    Point points[3];
    // Конструктор по умолчанию
    Circle() : center(), radius(1.0f) {
        for (int i = 0; i < 3; ++i) {
            points[i] = Point();
        }
    }
    // Конструктор с параметрами
    Circle(float x, float y, float r, Point p1, Point p2, Point p3) : center(x, y), radius(r) {
        points[0] = p1;
        points[1] = p2;
        points[2] = p3;
    }
    // Метод для проверки, находится ли точка внутри окружности
    bool isPointInside(Point point) const {
        return point.distanceTo(center) < radius - EPS;
    }
    // Метод для проверки, находится ли точка вне окружности
    bool isPointOutside(Point point) const {
        return point.distanceTo(center) > radius + EPS;
    }
    bool contains(Point point) {
        for (int i = 0; i < 3; ++i) {
            if (point.isSame(points[i])) return true;
        }
        return false;
    }
    // Метод для вывода информации об окружности
    void printInfo() const {
        cout << "Circle: Center(" << center.x << ", " << center.y << "), Radius: " << radius << endl;
    }
};
struct Triangle {
    Point vertex1;
    Point vertex2;
    Point vertex3;
    Triangle() : vertex1(), vertex2(), vertex3() {}
    Triangle(const Point& v1, const Point& v2, const Point& v3) : vertex1(v1), vertex2(v2), vertex3(v3) {}
    // Метод для проверки, лежит ли точка на рёбрах треугольника
    bool isPointOnEdges(Point point) const {
        if (point.isSame(vertex1) or point.isSame(vertex2) or point.isSame(vertex3)) return true;
        return point.isOnSegment(vertex1, vertex2) || point.isOnSegment(vertex2, vertex3) || point.isOnSegment(vertex3, vertex1);
    }
    // Метод для определения, лежит ли точка внутри треугольника
    bool isPointInside(Point point) const {
        if (isPointOnEdges(point)) return false;
        float areaTotal = area();
        float area1 = Triangle(point, vertex2, vertex3).area();
        float area2 = Triangle(vertex1, point, vertex3).area();
        float area3 = Triangle(vertex1, vertex2, point).area();
        // Точка лежит внутри треугольника, если сумма площадей подтреугольников равна площади всего треугольника
```

```
        return equals(areaTotal, area1 + area2 + area3);
    }
    // Метод для проверки, находится ли точка вне треугольника
    bool isPointOutside(Point point) const {
        if (isPointOnEdges(point)) return false;
        return !isPointInside(point);
    }
    bool contains(Point point) {
        return (point.isSame(vertex1) or point.isSame(vertex2) or point.isSame(vertex3));
    }
    // Метод для вычисления площади треугольника по формуле Герона
    float area() const {
        float side1 = sqrt(pow(vertex2.x - vertex1.x, 2) + pow(vertex2.y - vertex1.y, 2));
        float side2 = sqrt(pow(vertex3.x - vertex2.x, 2) + pow(vertex3.y - vertex2.y, 2));
        float side3 = sqrt(pow(vertex1.x - vertex3.x, 2) + pow(vertex1.y - vertex3.y, 2));
        float s = (side1 + side2 + side3) / 2.0f;
        return sqrt(s * (s - side1) * (s - side2) * (s - side3));
    }
};
double convert_to_double(const std::string& str) {
    char* endptr;
    double result = std::strtod(str.c_str(), &endptr);
    // Если endptr указывает на конец строки, значит, строку можно преобразовать в число
    if (endptr == str.c_str() + str.size()) return result; else return INF;
};
Point read_point_from_line(string& input, ofstream& log_file) {
    istringstream iss(input);
    string token;
    Point result(INF, INF);
    while (iss >> token) {
        float value = convert_to_double(token);
        if (!equals(value, INF)) {
            if (equals(result.x, INF)) {
                result.x = value;
            }
            else {
                result.y = value;
                log_file << " (" << result.x << ", " << result.y << ") ";
                while (iss >> token) {
                    log_file << "лишние данные   ";
                    log_file << token;
                }
                log_file<<endl;
                return result;
            }
        }
    }
    if (input == "") {
        log_file << "Пустая строка" << endl;
        return Point(INF, INF);
    }
    else{log_file << "недостаточно данных для точки" << endl;}
    return Point(INF, INF);
};
// Метод для построения окружности по трем точкам
Circle circleFromPoints(Point p1, Point p2, Point p3) {
    float x1 = p1.x, y1 = p1.y;
    float x2 = p2.x, y2 = p2.y;
    float x3 = p3.x, y3 = p3.y;
    // Решение системы уравнений для нахождения центра окружности (h, k)
    float A = x1 - x2;
    float B = y1 - y2;
    float C = x1 - x3;
    float D = y1 - y3;
    float E = ((x1 * x1 - x2 * x2) + (y1 * y1 - y2 * y2)) / 2.0;
    float F = ((x1 * x1 - x3 * x3) + (y1 * y1 - y3 * y3)) / 2.0;
    float h = (D * E - B * F) / (A * D - B * C);
    float k = (A * F - C * E) / (A * D - B * C);
    // Нахождение радиуса
    float radius = sqrt((x1 - h) * (x1 - h) + (y1 - k) * (y1 - k));
    return Circle(h, k, radius, p1, p2, p3);
}
struct Figure {
    Circle circle;
    Triangle triangle;
    // Конструктор по умолчанию для Figure
    Figure(): circle(), triangle() {}
    // Конструктор для Figure
    Figure(Circle c, Triangle t) : circle(c), triangle(t) {}
    bool isPointInside(Point point) {
        return (circle.isPointInside(point) and !triangle.isPointInside(point)) or (!circle.isPointInside(point) and triangle.isPointInside(point));
    }
    Point* points() {
        Point* points = new Point[6];
        points[0] = circle.points[0];
        points[1] = circle.points[1];
        points[2] = circle.points[2];
        points[3] = triangle.vertex1;
        points[4] = triangle.vertex2;
        points[5] = triangle.vertex3;
        return points;
    }
    bool contains(Point point) {
        return circle.contains(point) or triangle.contains(point);
    }
};
string get_line(fstream& input) {
    string result = "";
    char ch;
    while (input.get(ch) && ch != '\n') {
        result += ch;
    }
    return result;
}
void process(fstream& input, ofstream& log_file, ofstream& result) {
```

```
if (!input.is_open()) {
    cerr << "Невозможно открыть файл с точками" << endl;
    return;
}
int n;
input >> n;
Point* points = new Point[n] {Point(INF, INF)};
int point_count = 0;
// Считываем данные из файла и заполняем массив точек
while (point_count < n && !input.eof()) {
    string line = get_line(input);
    Point point = read_point_from_line(line, log_file);
    if (!point.isInf()) {
        points[point_count++] = point;
    }
}
result <<"Карпенко Анастасия"<<"\n"<<"группа 3353"<<"\n";
result << "Даны N произвольных точек на плоскости. Найти среди них точки, которые образуют фигуру с максимальным количеством точек внутри области" << "\n\n";
log_file << "\nКоличество считанных точек: " << point_count << endl;
result << "Количество считанных точек: " << point_count << endl;
for (int index = 0; index < point_count; ++index) {
    result << "(" << points[index].x << ", " << points[index].y << ")" << endl;
}
log_file << endl;
result<< endl;
Circle* circles = new Circle[point_count * (point_count - 1) * (point_count - 2)];
int circle_count = 0;
for (int index1 = 0; index1 < point_count; ++index1) {
    Point point1 = points[index1];
    for (int index2 = index1 + 1; index2 < point_count; ++index2) {
        Point point2 = points[index2];
        if (point2.isSame(point1)) continue;
        for (int index3 = index2 + 1; index3 < point_count; ++index3) {
            Point point3 = points[index3];
            if (point3.isSame(point1)) continue;
            if (point3.isSame(point2)) continue;
            if (arePointsCollinear(point1, point2, point3)) continue;
            circles[circle_count++] = circleFromPoints(point1, point2, point3);
        }
    }
}
Figure* figures = new Figure[circle_count * (point_count - 3) * (point_count - 4) * (point_count - 5)];
int figure_count = 0;
for (int circle_index = 0; circle_index < circle_count; ++circle_index) {
    Circle circle = circles[circle_index];
    for (int index1 = 0; index1 < point_count; ++index1) {
        Point point1 = points[index1];
        if (circle.contains(point1)) continue;
        if (!circle.isPointInside(point1)) continue;
        for (int index2 = 0; index2 < point_count; ++index2) {
            Point point2 = points[index2];
            if (circle.contains(point2)) continue;
            if (!circle.isPointOutside(point2)) continue;
            for (int index3 = index2 + 1; index3 < point_count; ++index3) {
                Point point3 = points[index3];
                if (circle.contains(point3)) continue;
                if (point2.isSame(point3)) continue;
                if (!circle.isPointOutside(point3)) continue;
                if (arePointsCollinear(point1, point2, point3)) continue;
                Triangle triangle(point1, point2, point3);
                figures[figure_count++] = Figure(circle, triangle);
            }
        }
    }
}
int* point_inside_count = new int[figure_count] {0};
Point** points_inside_array = new Point*[figure_count];
for (int i = 0; i < figure_count; ++i) {
    points_inside_array[i] = new Point[point_count - 6];
}
int max_figur=0;
int max_points = 0;
for (int index = 0; index < figure_count; index++) {
    Figure figure = figures[index];
   log_file << "Фигура: " << index+1 << " строится из точек: ";
    for (int point_index = 0; point_index < 6; point_index++) {
        Point point = figure.points()[point_index];
        log_file << "(" << point.x << ", " << point.y << ") ";
    }
    log_file << endl;
    for (int point_index = 0; point_index < point_count; point_index++) {
        Point point = points[point_index];
        if (figure.contains(point)) continue;
        if (figure.isPointInside(point)) {
            points_inside_array[index][point_inside_count[index]++] = point;
            log_file<< "принадлежит    (" << point.x << ", " << point.y << ") "<<"\n";
        }
        else{
            log_file<< "не принадлежит    (" << point.x << ", " << point.y << ") "<<"\n";
        }
    }
    max_points = max(max_points, point_inside_count[index]);
}
result << "Максимальное число точек внутри фигуры: " << max_points << endl;
log_file << "\n\n\n\nМаксимальное число точек внутри фигуры: " << max_points << endl;
for (int index = 0; index < figure_count; index++) {
    if (!point_inside_count[index] == max_points) continue;
    if (point_inside_count[index] == max_points){
    Figure figure = figures[index];
    result << "Фигура: " << index + 1 << " строится из точек: ";
    log_file << "Фигура: " << index + 1 << " строится из точек: ";
    for (int point_index = 0; point_index < 6; point_index++) {
        Point point = figure.points()[point_index];
        result << "(" << point.x << ", " << point.y << ")";
        log_file << "(" << point.x << ", " << point.y << ")";
```

```
        }
        log_file << endl;
        result<< endl;
        max_figur++;
        for (int point_index = 0; point_index < point_inside_count[index]; point_index++) {
            Point point = points_inside_array[index][point_index];
            result << " принадлежит точка   (" << point.x << ", " << point.y << ") " << endl;
            log_file << "принадлежит точка   (" << point.x << ", " << point.y << ") " << endl;
        }
        result << "\n";
        log_file<<"\n";}
    }
    result << "количество фигур с максимальным числом точек внутри фигуры: " << max_figur << endl;
    log_file << "количество фигур с максимальным числом точек внутри фигуры: " << max_figur << endl;
    delete[] points;
    delete[] circles;
    delete[] figures;
    delete[] point_inside_count;
    for (int i = 0; i < figure_count; ++i) {
        delete[] points_inside_array[i];
    }
    delete[] points_inside_array;
}
int main() {
    fstream input("points.txt");
    ofstream log_file("protocol.txt");
    ofstream result("result.txt");
    process(input, log_file, result);
    input.close();
    log_file.close();
    result.close();
    return 0;
}
```

Тесты программы

Тест на наборе точек из контрольного примера:

Входной файл:

10
10 4
8 2 point
4 2
6 8
-2 6
5 -4

-1 0

Файл – протокол:

(10, 4)
(8, 2) point
(4, 2)
(6, 8)
(-2, 6)
(5, -4)
недостаточно данных для точки
(-1, 0)
Количество считанных точек: 7
Фигура: 1 строится из точек: (10, 4) (8, 2) (4, 2) (6, 8) (-2, 6) (5, -4)
не принадлежит (-1, 0)
Фигура: 2 строится из точек: (10, 4) (8, 2) (4, 2) (6, 8) (-2, 6) (-1, 0)
не принадлежит (5, -4)
……
Фигура: 48 строится из точек: (6, 8) (-2, 6) (-1, 0) (4, 2) (8, 2) (5, -4)
не принадлежит (10, 4)
Максимальное число точек внутри фигуры: 1
Фигура: 4 строится из точек: (10, 4)(8, 2)(-2, 6)(4, 2)(5, -4)(-1, 0)
принадлежит точка (6, 8)
……
Фигура: 39 строится из точек: (4, 2)(6, 8)(5, -4)(8, 2)(-2, 6)(-1, 0)

принадлежит точка (10, 4)
количество фигур с максимальным числом точек внутри фигуры: 17

Файл - результат:
Карпенко Анастасия
группа 3353
Даны N произвольных точек на плоскости. Найти среди них точки, которые образуют фигуру с максимальным количеством точек внутри области
Количество считанных точек: 7
(10, 4)
(8, 2)
(4, 2)
(6, 8)
(-2, 6)
(5, -4)
(-1, 0)
Максимальное число точек внутри фигуры: 1
Фигура: 4 строится из точек: (10, 4)(8, 2)(-2, 6)(4, 2)(5, -4)(-1, 0)
 принадлежит точка (6, 8)
…….
Фигура: 39 строится из точек: (4, 2)(6, 8)(5, -4)(8, 2)(-2, 6)(-1, 0)
 принадлежит точка (10, 4)
количество фигур с максимальным числом точек внутри фигуры: 17

Тест на наборе из 15 точек:
Входной файл:
100
-68 5
-12 -77
-87 -36
98 82
-99 -7 точка
74 -24

91 -2
-11 39
-76 73
-49 -89
-89 -100
22 -8
-6 40
25 -44
70 93

Файл – протокол:
(-68, 5)
 (-12, -77)
 (-87, -36)
 (98, 82)
 (-99, -7) лишние значения точка
 (74, -24)
недостаточно данных для точки
 (91, -2)
 (-11, 39)

(-76, 73)
(-49, -89)
(-89, -100)
(22, -8)
(-6, 40)
(25, -44)
(70, 93)
Пустая строка
Количество считанных точек: 15
Фигура: 1 строится из точек: (-68, 5) (-12, -77) (98, 82) (74, -24) (-87, -36) (-99, -7)
принадлежит (91, -2)
принадлежит (-11, 39)
не принадлежит (-76, 73)
не принадлежит (-49, -89)
не принадлежит (-89, -100)
принадлежит (22, -8)
принадлежит (-6, 40)
принадлежит (25, -44)
принадлежит (70, 93)
……
Фигура: 32289 строится из точек: (-6, 40) (25, -44) (70, 93) (22, -8) (-49, -89) (-89, -100)
не принадлежит (-68, 5)
не принадлежит (-12, -77)
не принадлежит (-87, -36)
принадлежит (98, 82)
не принадлежит (-99, -7)
принадлежит (74, -24)
принадлежит (91, -2)
не принадлежит (-11, 39)
не принадлежит (-76, 73)

Максимальное число точек внутри фигуры: 9
Фигура: 1692 строится из точек: (-68, 5)(-87, -36)(70, 93)(-12, -77)(-99, -7)(-76, 73)
принадлежит точка (98, 82)
принадлежит точка (74, -24)
принадлежит точка (91, -2)
принадлежит точка (-11, 39)
принадлежит точка (-49, -89)
принадлежит точка (-89, -100)
принадлежит точка (22, -8)
принадлежит точка (-6, 40)
принадлежит точка (25, -44)
……

Фигура: 30089 строится из точек: (-76, 73)(-49, -89)(70, 93)(25, -44)(98, 82)(-89, -100)
принадлежит точка (-68, 5)
принадлежит точка (-12, -77)
принадлежит точка (-87, -36)
принадлежит точка (-99, -7)
принадлежит точка (74, -24)
принадлежит точка (91, -2)
принадлежит точка (-11, 39)
принадлежит точка (22, -8)

принадлежит точка (-6, 40)
количество фигур с максимальным числом точек внутри фигуры: 213

Файл - результат:
Карпенко Анастасия
группа 3353
Даны N произвольных точек на плоскости. Найти среди них точки, которые образуют фигуру с максимальным количеством точек внутри области
Количество считанных точек: 15
(-68, 5)
(-12, -77)
(-87, -36)
(98, 82)
(-99, -7)
(74, -24)
(91, -2)
(-11, 39)
(-76, 73)
(-49, -89)
(-89, -100)
(22, -8)
(-6, 40)
(25, -44)
(70, 93)
Максимальное число точек внутри фигуры: 9
Фигура: 1692 строится из точек: (-68, 5)(-87, -36)(70, 93)(-12, -77)(-99, -7)(-76, 73)
 принадлежит точка (98, 82)
 принадлежит точка (74, -24)
 принадлежит точка (91, -2)
 принадлежит точка (-11, 39)
 принадлежит точка (-49, -89)
 принадлежит точка (-89, -100)
 принадлежит точка (22, -8)
 принадлежит точка (-6, 40)
 принадлежит точка (25, -44)
……
Фигура: 30089 строится из точек: (-76, 73)(-49, -89)(70, 93)(25, -44)(98, 82)(-89, -100)
 принадлежит точка (-68, 5)
 принадлежит точка (-12, -77)
 принадлежит точка (-87, -36)
 принадлежит точка (-99, -7)
 принадлежит точка (74, -24)
 принадлежит точка (91, -2)
 принадлежит точка (-11, 39)
 принадлежит точка (22, -8)
 принадлежит точка (-6, 40)
количество фигур с максимальным числом точек внутри фигуры: 213

Вывод

В процессе выполнения работы были закреплены знания и навыки, полученные за семестр, они применены для решения задачи поиска фигуры на массиве точек и нахождение максимальной из них по заданному признаку.