

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ

ПРЕЗИДЕНТСКИЙ ФИЗИКО-МАТЕМАТИЧЕСКИЙ ЛИЦЕЙ №
239

ОТЧЁТ ПО ГОДОВОМУ ПРОЕКТУ

Ученик:

Попельшко Аким

Преподаватель:

Клюнин Алексей Олегович

Класс:

10-3

Санкт-Петербург
2017

Содержание

1	Постановка задачи	3
2	Алгоритм решения задачи	3
2.1	Базовые структуры данных	3
2.2	Построение алгоритма	4

1 Постановка задачи

Задано множество прямых на плоскости (угловым коэффициентом и смещением). Найти тройку прямых, вырезающих треугольник минимальной площади.

Используемые программы: GitHub, IntelliJ IDEA, TeXstudio, OpenGL.

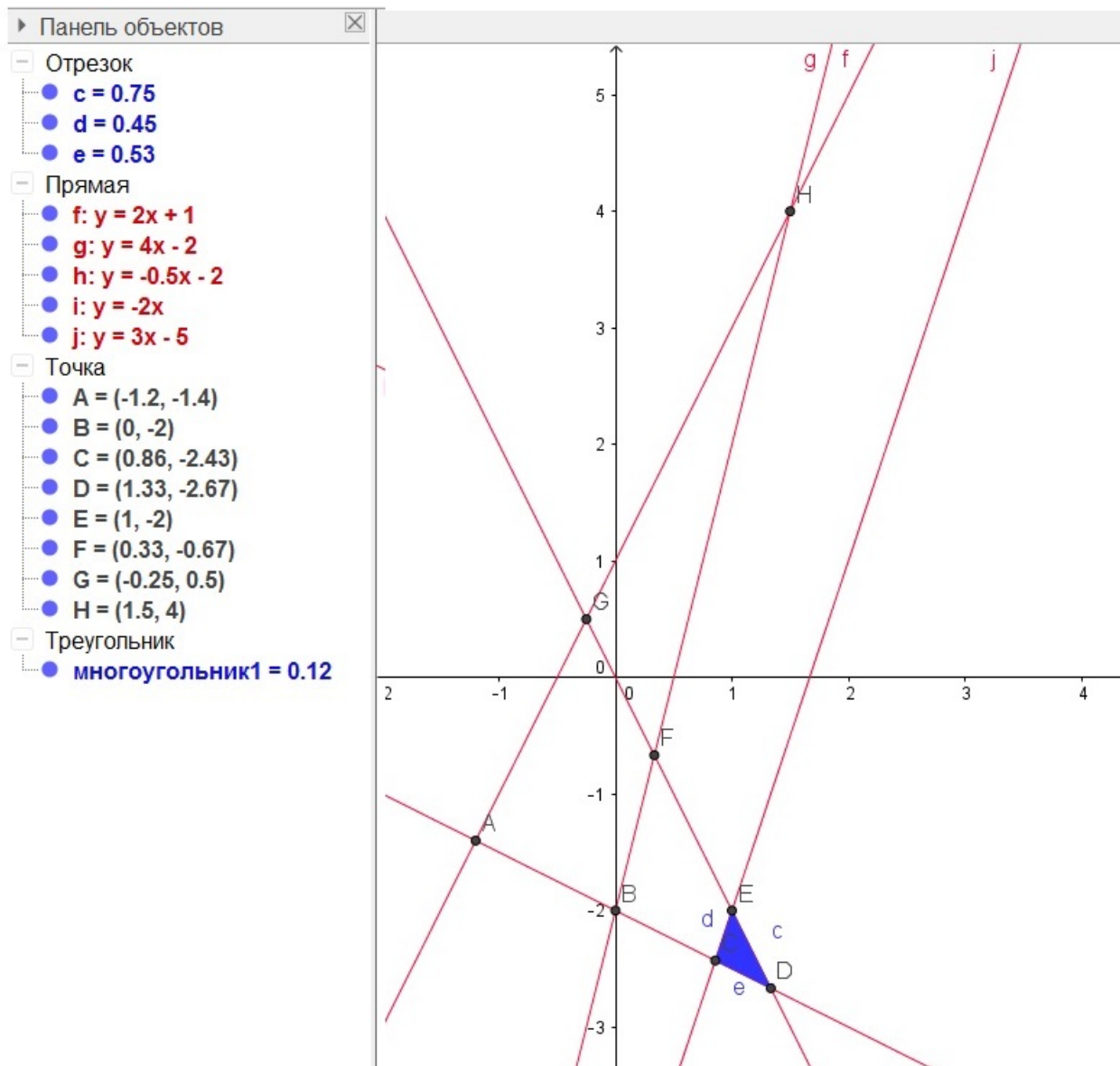


Рис. 1: Пример данного множества прямых. На рисунке также указаны точки пересечения прямых (серые) и искомый треугольник (выделен синим). Выпонено в программе GeoGebra.

2 Алгоритм решения задачи

2.1 Базовые структуры данных

Класс Point (точка) с двумя полями: x и y типа `double`, обозначающих координаты точки. Этот класс будет иметь конструктор от x и y и конструктор по умолчанию; статичный метод `static boolean isLieOnLine(Point a, Point b, Point c)`, который будет определять, лежат ли три данные точки на одной прямой или нет; сеттеры и геттеры для каждой переменной.

Класс Line (прямая) с двумя полями: k и b типа `double`, обозначающих соответственно угловой коэффициент и смещение прямой. Этот класс будет иметь конструкторы от k и

b, от k (тогда $b=0$) и конструктор по умолчанию (прямая $y=x$); статичный метод `static boolean isParallel(Line a, Line b)`, который будет определять, являются ли две данные прямые параллельными или нет; статичный метод `static boolean isCoinciding(Line a, Line b)`, который будет определять, являются ли две данные прямые совпадающими или нет; статичный метод `static Point intersection(Line a, Line b)`, который будет возвращать точку пересечения двух данных непараллельных прямых; сеттеры и геттеры для каждой переменной.

Класс `Triangle` (треугольник) с тремя полями: `a`, `b`, `c` типа `Point`, обозначающих вершины треугольника. Этот класс будет иметь конструктор от `a`, `b`, `c` (если любые две из трёх точек совпадают или три точки лежат на одной прямой, возвращает `null`) и конструктор от трёх прямых, высекающих его отрезками между точек пересечения друг с другом (если любые две из трёх прямых совпадают или параллельны, возвращает `null`); метод `double getArea()`, возвращающий площадь треугольника; сеттеры (с проверкой на существование треугольника) и геттеры для каждой переменной.

Площадь треугольника через координаты вершин считается по формуле
$$S = \frac{1}{2} \begin{vmatrix} x_1 - x_3 & y_1 - y_3 \\ x_2 - x_3 & y_2 - y_3 \end{vmatrix}$$

2.2 Построение алгоритма

На вход подаются пары чисел типа `double`, где каждая пара чисел является соответственно угловым коэффициентом и смещением одной из прямых. Таких пар чисел не может быть больше 100.

Создаётся массив `arr` из объектов класса `Line` длиной в 100 элементов.

В цикле `while(sc.hasNextDouble())` считывается пара чисел, с помощью конструктора создаётся объект класса `Line` и записывается в массив `arr`.

Создаётся переменная `minArea` типа `double`, равная -1, и пустые переменные `a`, `b` и `c` класса `Line`. Затем с помощью трёх вложенных друг в друга циклов `for` для каждой тройки ненулевых элементов массива создаётся объект класса `Triangle` (с помощью конструктора от трёх прямых) и, если треугольник от этих прямых существует (прямые попарно несовпадающие и непараллельные), считается его площадь с помощью метода `getArea()`. Если в `minArea` лежит значение -1, мы присваиваем ей значение площади этого треугольника (если он существует). Если в `minArea` лежит другое значение, мы присваиваем ей значение площади этого треугольника, если площадь меньше, чем `minArea` (и если треугольник существует). Каждый раз при присвоении `minArea` нового значения в переменные `a`, `b` и `c` записываются прямые, образующие этот треугольник.

Прямые `a`, `b` и `c`, а также значение `minArea` выводятся на экран.