Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 1

Тема Комплексная геометрическая задача с использованием графических средств для представления результата

Студент Буртелов Н.Н.

Группа ИУ7-43Б

Оценка (баллы) _____

Преподаватель Куров А.В.

Москва.
2020 г.

ЦЕЛЬ РАБОТЫ: научиться применять знания аналитической геометрии для решения практических задач машинной графики, осуществлять построение изображения (в СКУ) объектов, расположенных в МСК.

РЕЗУЛЬТАТ: должна быть разработана программа, выполняющая ввод исходных данных, проверку их корректности, решение поставленной геометрической задачи и отображение полученного результата в графическом режиме.

Описание условия задачи.

На плоскости дано два множества точек, на точках первого множества строятся треугольники. На точках второго множества также строятся треугольники. Найти два (пару) таких треугольников, у которых прямая, соединяющая вершины тупых углов, образует максимальный угол с осью абсцисс.

Вывести изображение в графическом режиме.

Технические характеристики.

Используемый язык программирования для реализации поставленной задачи: Java, версия 12.

Описание алгоритма:

1. Ввод данных

    1.1 Проверка ввода данных

2. Перебор точек первого и второго множеств для составления всех возможных треугольников на основе точек первого множества и всех возможных треугольников на основе точек второго множества, имеющих тупой угол.

    2.1 Проверка на существование треугольника

    2.2 Проверка на наличие тупого угла

    2.3 Сохранение данных о подходящем треугольнике в массив.

3. Перебор треугольников на основе первого множества и треугольников на основе второго множества для нахождения угла, образующего максимальный угол с осью абсцисс.

    3.1 Составление пары, состоящей из треугольника на основе первого множества и треугольника на основе второго множества.

    3.2 Поиск угла, образующего максимальный угол с осью абсцисс.

    3.3 Сравнение найденного угла с текущим максимальным углом, для нахождения наибольшего.

4. Вывод результата в графическом виде.

Входные данные:

1. Строка, содержащая пары точек первого множества.

2. Строка, содержащая пары точек второго множества

Выходные данные:

1. Графическое представление элементов: пара треугольников, линия соединяющая вершины тупых углов.

Аварийные ситуации

1. Некорректный ввод строк, содержащих пары точек первого и второгмножества (наличие неподходящих символов).

2. Некорректный ввод строк, содержащих пары точек первого и второго множества (количество символов недостаточно для построения треугольника)

3. Невозможность построить треугольник по точкам первого или (и) второго множеств.

4. Невозможность построить треугольник, имеющий тупой угол, по точкам первого или (и) второго множеств.

5. Точки первого множества идентичны точкам второго множества. (Если кол-во точек небольшое, то возможна ситуация, что вершины образованных треугольники, будут иметь одинаковые координаты).

Код программы.

1. Функция считывания данных из полей ввода и запись их в массивы, тип double.

absDataPointOne, absDataPointTwo – массивы содержащие абсциссы точек первого и второго множеств соответственно.

ordDataPointOne, ordDataPointTwo – массивы содержащие ординаты точек первого и второго множеств соответственно.

Параметры метода: taOne, taTwo – поля ввода данных.

```
private void readTextArea(TextArea taOne, TextArea taTwo) {
    System.out.println(taOne.getText() + taTwo.getText());
    String[] str1 = taOne.getText().split( regex: " ");
    String[] str2 = taTwo.getText().split( regex: " ");

    double[] absDataOne = new double[str1.length / 2];
    double[] ordDataOne = new double[str1.length / 2];
    double[] absDataTwo = new double[str2.length / 2];
    double[] ordDataTwo = new double[str2.length / 2];

    for (int i = 0, j = 0; j < str1.length; i++, j += 2) {
        absDataOne[i] = Double.valueOf(str1[j]);
        ordDataOne[i] = Double.valueOf(str1[j + 1]);
    }
    for (int i = 0, j = 0; j < str2.length; i++, j += 2) {
        absDataTwo[i] = Double.valueOf(str2[j]);
        ordDataTwo[i] = Double.valueOf(str2[j + 1]);
    }
    absDataPointOne = absDataOne;
    absDataPointTwo = absDataTwo;
    ordDataPointOne = ordDataOne;
    ordDataPointTwo = ordDataTwo;
}
```

2. Перебор точек первого и второго множеств для составления всех возможных треугольников на основе точек первого множества и всех возможных треугольников на основе точек второго множества, имеющих тупой угол.

Переменные функции: absDataPoint – массив абсцисс точек, ordDataPoint – массив ординат точек, dataPoint – массив, содержащий координаты вершин треугольника.

Возвращаемое значение: dataPoint.

В функции совершается перебор точек для составления треугольника, присутствует проверка на существование треугольника (функция checkExist) и проверка на наличие тупого угла (функция getObtuseAngle).

Если текущий набор точек удовлетворяет всем проверкам, то осуществляется добавления этих точек в массив dataPoint (функция appendArr).

```
private double[][][] getTriangle(double[] absDataPoint, double[] ordDataPoint, double[][][] dataPoint) {
    double x1, x2, x3, y1, y2, y3, id;
    boolean flag = false;
    double[][] dataCord = new double[3][2];

    for (int i = 0; i < absDataPoint.length - 2; i++) {
        x1 = absDataPoint[i];
        y1 = ordDataPoint[i];
        for (int j = i + 1; j < absDataPoint.length - 1; j++) {
            x2 = absDataPoint[j];
            y2 = ordDataPoint[j];
            for (int k = j + 1; k < absDataPoint.length; k++) {
                x3 = absDataPoint[k];
                y3 = ordDataPoint[k];

                if (checkExist(x1, x2, x3, y1, y2, y3)) {
                    id = getObtuseAngle(x1, x2, x3, y1, y2, y3);

                    if (id == 1)
                        dataCord = setArr(x1, y1, x2, y2, x3, y3, dataCord);
                    else if (id == 2)
                        dataCord = setArr(x2, y2, x1, y1, x3, y3, dataCord);
                    else if (id == 3)
                        dataCord = setArr(x3, y3, x1, y1, x2, y2, dataCord);

                    if (id != -1) {
                        if (flag)
                            dataPoint = appendArr(dataPoint, dataCord);
                        else {
                            double[][][] dataPointNow = new double[1][3][2];
                            dataPoint = appendArr(dataPointNow, dataCord);
                            flag = true;
                        } } } } } }
    return dataPoint;
}
```

3. Функция проверки треугольника на существование.

Переменные функции: координаты вершин предполагаемого треугольника.

Для того, чтобы проверить существует ли треугольник, достаточно посчитать его площадь, если она равна 0, то треугольник не будет существовать.

Площадь треугольника ABC с вершинами в точках A(x1;y1), B(x2;y2), C(x3;y3) можно вычислить с помощью формулы.

$$S=0,5*[(x1-x3)(y2-y3)-(x2-x3)(y1-y3)].$$

Нам же достаточно узнать, не равна ли площадь 0, поэтому умножать на 0.5 не имеет смысла.

Возвращаемое значение: переменная типа boolean (true – если треугольник существует, false – если нет).

```
//Проверка на существование треугольника
private boolean checkExist(double x1, double x2, double x3, double y1, double y2, double y3) {
    double s = Math.abs((x1 - x3) * (y2 - y3) - (x2 - x3) * (y1 - y3));

    if (s - 0.00001 >= eps)
        return true;

    return false;
}
```

4. Функция проверки на наличие тупого угла.

Переменные функции: координаты вершин треугольника.

Возвращаемое значение: переменная типа int (номер вершины, содержащий тупой угол).

Используется ф-ия **getSideLength** – отвечает за подсчет длин сторон треугольника.

Проверка на наличие тупого угла осуществляется следующим образом:

1. Находим длины сторон треугольника.

2. Возводим длины сторон в квадрат.

3. Пусть **a, b, c** – длины сторон треугольника. Тогда, если $\mathbf{a^2 > b^2 + c^2}$, то напротив стороны **«a»** лежит тупой угол, аналогично и с другими сторонами. Данное свойство вытекает из теоремы косинусов.

Функция возвращает номер вершины с тупым углом.

```
//проверка на наличие тупого угла
private int getObtuseAngle(double x1, double x2, double x3, double y1, double y2, double y3) {
    int index = -1;
    double a = Math.pow(getSideLength(x2, x3, y2, y3), 2);
    double b = Math.pow(getSideLength(x1, x3, y1, y3), 2);
    double c = Math.pow(getSideLength(x1, x2, y1, y2), 2);

    if (a > b + c)
        index = 1;
    else if (b > a + c)
        index = 2;
    else if (c > a + b)
        index = 3;

    return index;
}
```

5. Подсчет длин сторон треугольника

Переменные функции: координаты крайних точек отрезка(сторона треугольника)

Возвращаемое значение: длина стороны.

Подсчет длины стороны осуществляется по теореме Пифагора.

```
//Подсчет длин сторон
private double getSideLength (double x1, double x2, double y1, double y2) {
    return Math.sqrt(Math.abs((x1 - x2)*(x1 - x2) - (y1 - y2)*(y1 - y2)));
}
```

6. Функция нахождения пары треугольников, у которых прямая соединяющая вершины тупых углов образует максимальной угол с осью абсцисс.

```
double maxAngle = -1; //Искомый угол
private double eps = 0.00001;
```

Изначально переменная **maxAngle**, отвечающая за хранение градусной меры тупого угла равна -1. При поиске угла, образующего максимальный угол с осью абсцисс, значение берется по модулю. Поэтому значение переменной **maxAngle** при расчетах будет лежать в отрезке от 0 до 180 градусов.

Работа функции:

1. Нахождение пар треугольников, имеющих тупой угол, на основе точек первого множества и точек второго множества соответственно.

2. Поиск пары треугольников, чей угол между прямой, соединяющей вершины тупых углов, и осью абсцисс будет наибольшим.

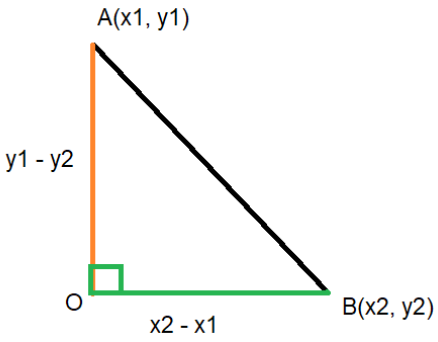
Угол ищется с помощью тангенса.

AB - прямая, соединяющая вершины тупых углов
угол **ABO –** угол между **AB** и осью абсцисс.

tg(ABO) = AO / OB = (y1 – y2) / (x2 – x1)

угол ABO = arctg(AO / OB)

Наибольший угол будет равен: 180° - угол **ABO**



3. Сохранение данных о паре треугольников.

```
boolean trianglesSearch() {
    double angle;
    int idOne = -1, idTwo = -1;

    dataPointOne = getTriangle(absDataPointOne, ordDataPointOne, dataPointOne);
    dataPointTwo = getTriangle(absDataPointTwo, ordDataPointTwo, dataPointTwo);

    if (dataPointOne.length == 0 || dataPointTwo.length == 0)
        return false;

    for (int i = 0; i < dataPointOne.length; i++) {
        for (int j = 0; j < dataPointTwo.length; j++) {
            angle = Math.atan((double) (dataPointOne[i][0][1] - dataPointTwo[j][0][1])
                    / (double) (dataPointOne[i][0][0] - dataPointTwo[j][0][0]));
            angle = 180 - Math.abs(Math.toDegrees(angle));

            if (angle - maxAngle > eps) {
                maxAngle = angle;
                idOne = i;
                idTwo = j;
            }
        }
    }
    if (idOne != -1 & idTwo != -1) {
        dataTriangleOne = Arrays.copyOf(dataPointOne[idOne], dataPointOne[idOne].length);
        dataTriangleTwo = Arrays.copyOf(dataPointTwo[idTwo], dataPointTwo[idTwo].length);
        return true;
    } else
        return false;
}
```

В функции используются вышеупомянутые методы.

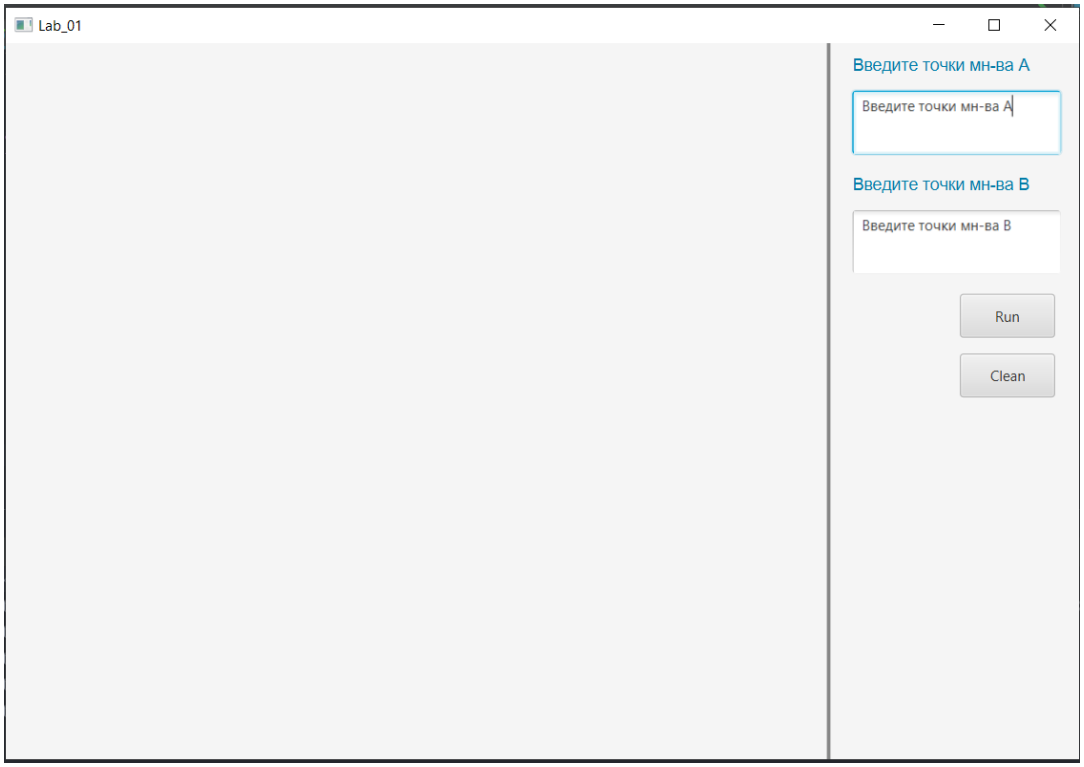
7. Вывод изображения

Переменные функции: массивы, содержащие координаты вершин треугольников.

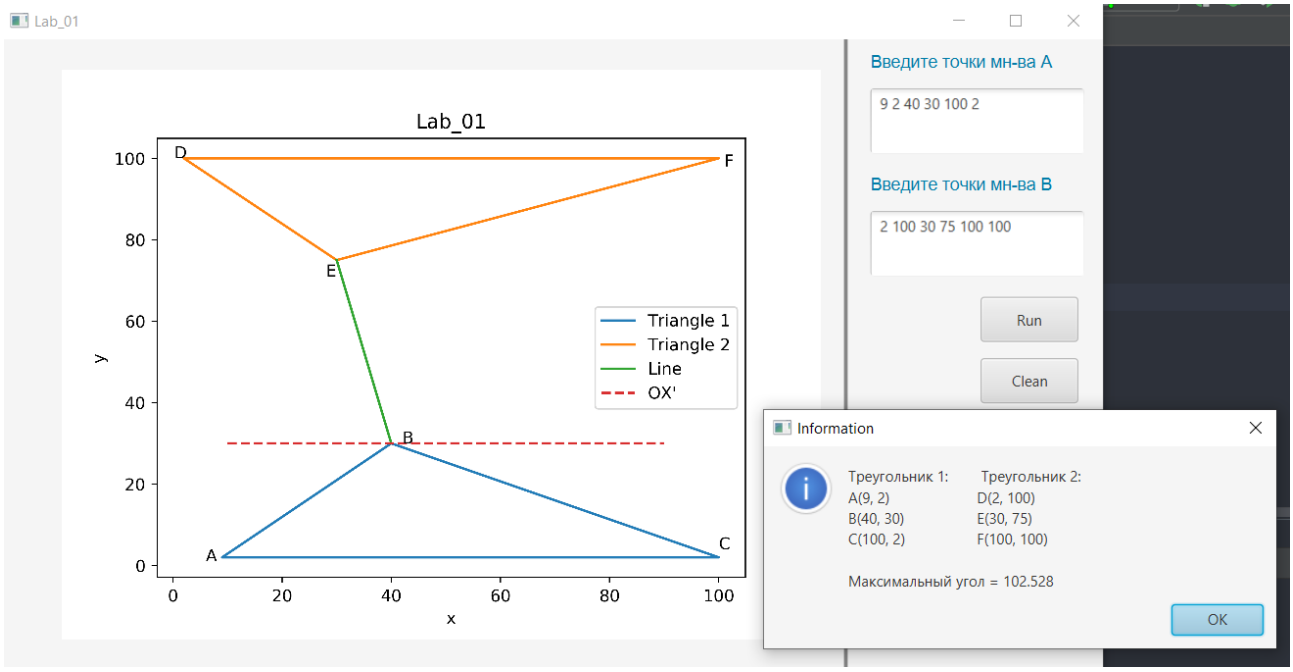
```
private void drawTriangle(double[][] dataArrOne, double[][] dataArrTwo) {
    try {
        Plot plt = Plot.create();

        plt.plot()
                .add(Arrays.asList(dataArrOne[0][0], dataArrOne[1][0], dataArrOne[2][0], dataArrOne[0][0]),
                        Arrays.asList(dataArrOne[0][1], dataArrOne[1][1], dataArrOne[2][1], dataArrOne[0][1]))
                .label( s: "Triangle 1")
                .linestyle( s: "-");

        plt.plot()
                .add(Arrays.asList(dataArrTwo[0][0], dataArrTwo[1][0], dataArrTwo[2][0], dataArrTwo[0][0]),
                        Arrays.asList(dataArrTwo[0][1], dataArrTwo[1][1], dataArrTwo[2][1], dataArrTwo[0][1]))
                .label( s: "Triangle 2")
                .linestyle( s: "-");

        plt.plot()
                .add(Arrays.asList(dataArrOne[0][0], dataArrTwo[0][0]),
                        Arrays.asList(dataArrOne[0][1], dataArrTwo[0][1]))
                .label( s: "Line")
                .linestyle( s: "-");

        //plt.xscale(ScaleBuilder.Scale.log);
        //plt.yscale(ScaleBuilder.Scale.log);
        plt.xlabel( s: "x");
        plt.ylabel( s: "y");
        plt.title( s: "Lab_01");
        plt.legend();
        plt.savefig( s: "C:/Users/user/Pictures/doc_bmstu/histogram.png").dpi(500);
        plt.executeSilently();
        //plt.show();
    }
```

Пример работы программы:

Начальный интерфейс программы:

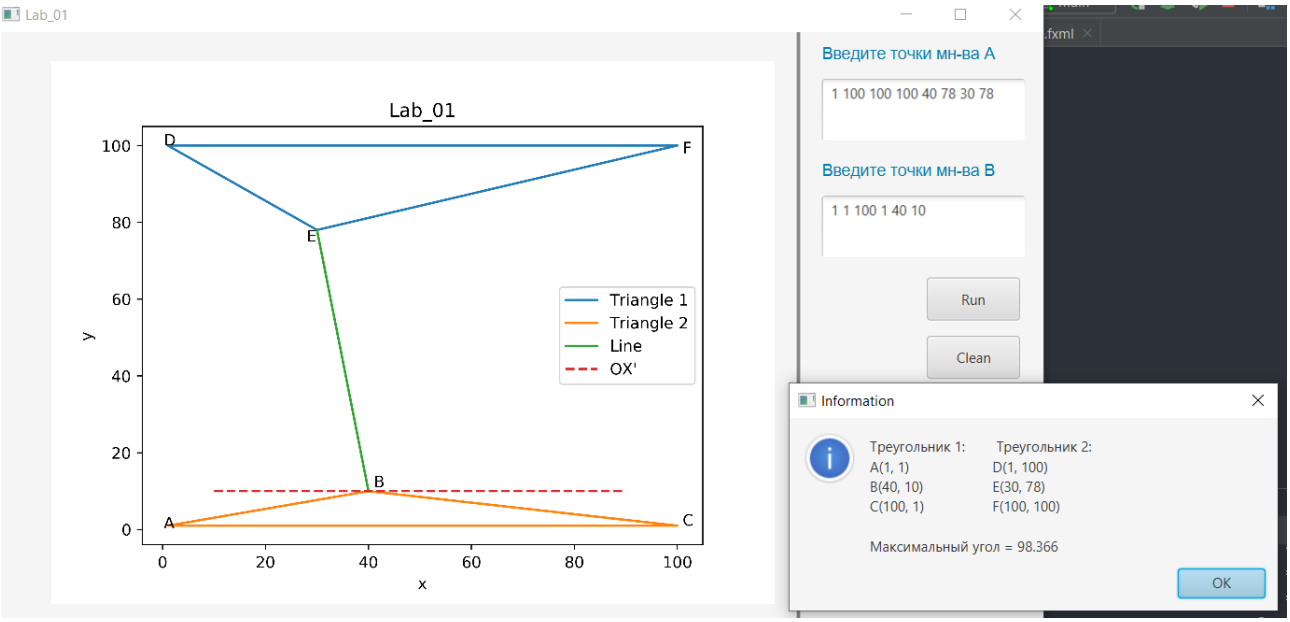


Пример 1.



`OX'` параллельно оси абсцисс. Данные о треугольниках и значении максимального угла выводятся в поле вывода.

Пример 2.



Убедимся, что программа выбирает необходимую пару треугольников.

На вход поступает точки второго мн-ва: 1 1, 100 1, 40 10

Из данного набора может быть составлен один треугольник, у которого будет тупой угол — треугольник ABC. A(1, 1) , B(40, 10), C(100, 1). Очевидно, что вершина B является вершиной тупого угла.

Точки первого мн-ва: 1 100, 100 100, 40 78, 30 78

Из данного набора могут быть составлены два треугольника, у которых будет тупой угол — треугольники KLM и DEF.

K(1, 100), L(40, 78), M(100, 100).

D(1, 100), E(30, 78), F(100, 100).

Очевидно, что вершины L и E являются вершинами тупых углов.

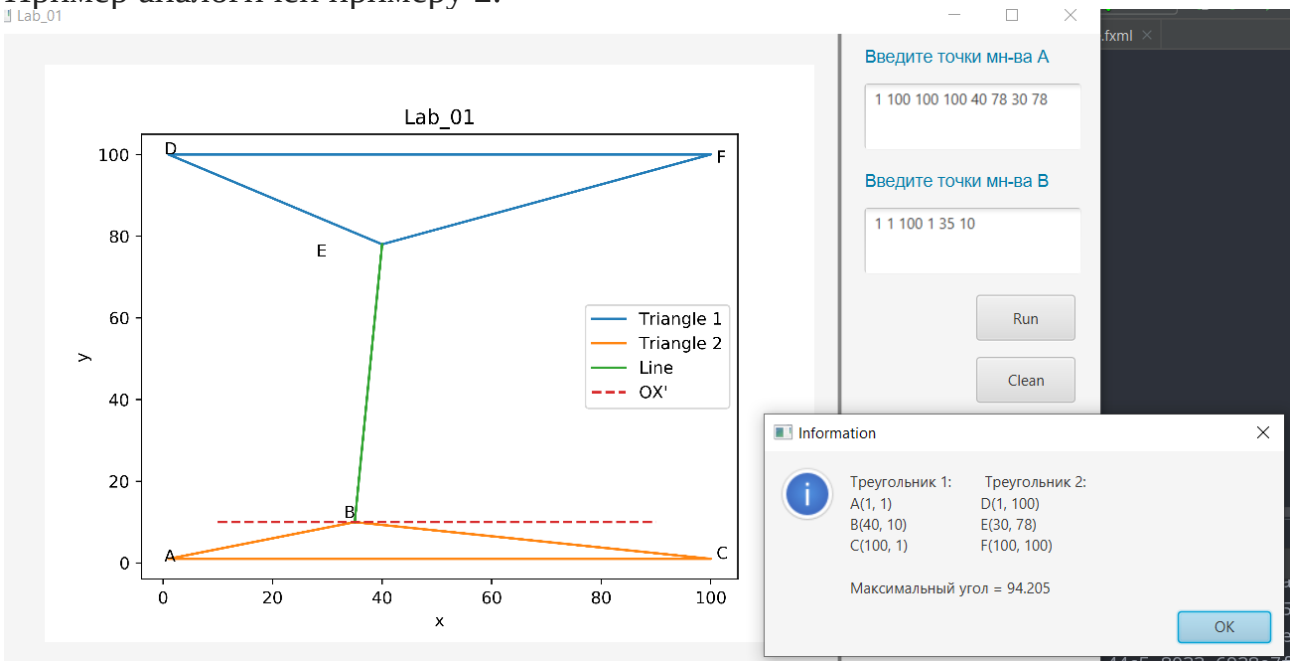
Программа выводит треугольники ABC и DEF, т.к. угол между прямой OX и отрезком BE будет очевидно больше чем между прямой OX и отрезком BL (если бы были выбраны треугольника ABC и KLM)

Угол между прямой OX и отрезком BE = 98.366°

Угол между прямой OX и отрезком BL = 90°

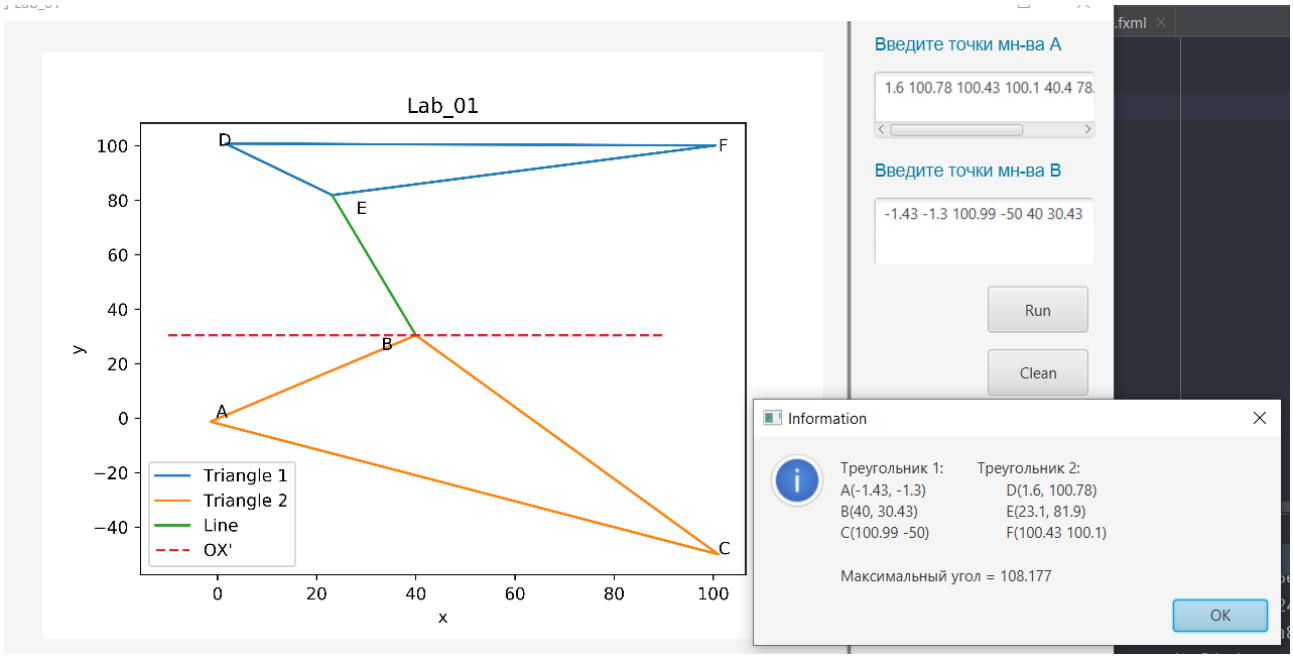
Пример 3.

Пример аналогичен примеру 2.

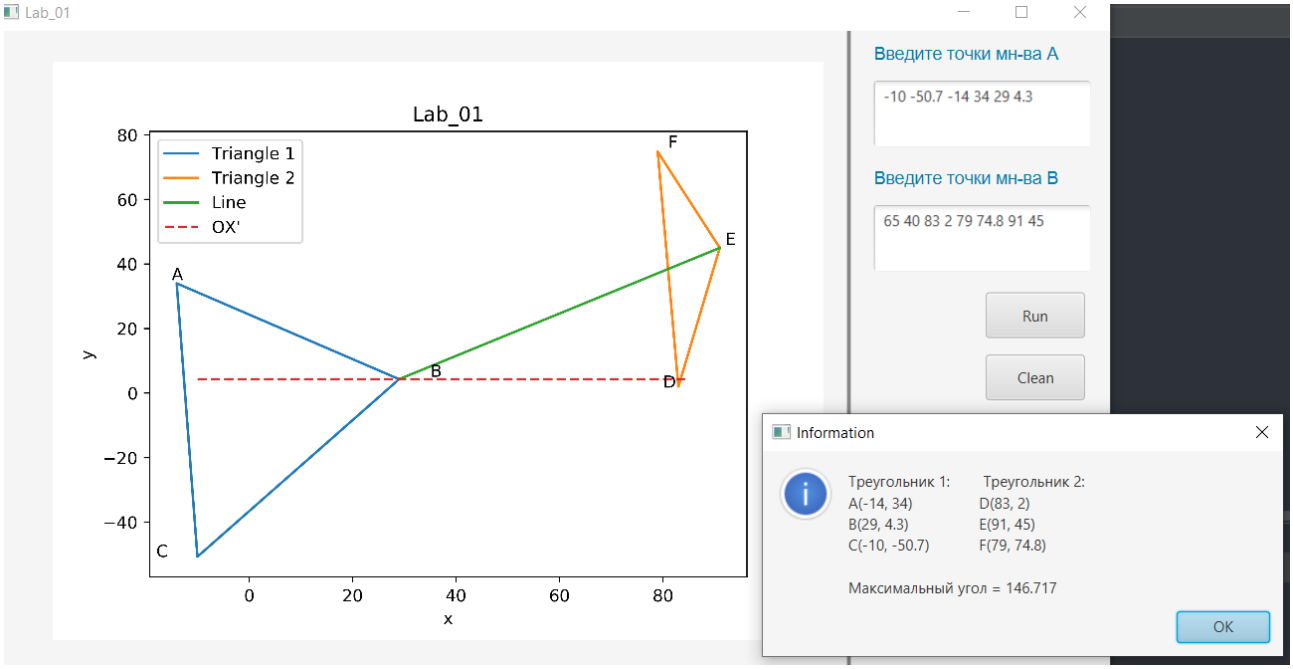


Пример 4.

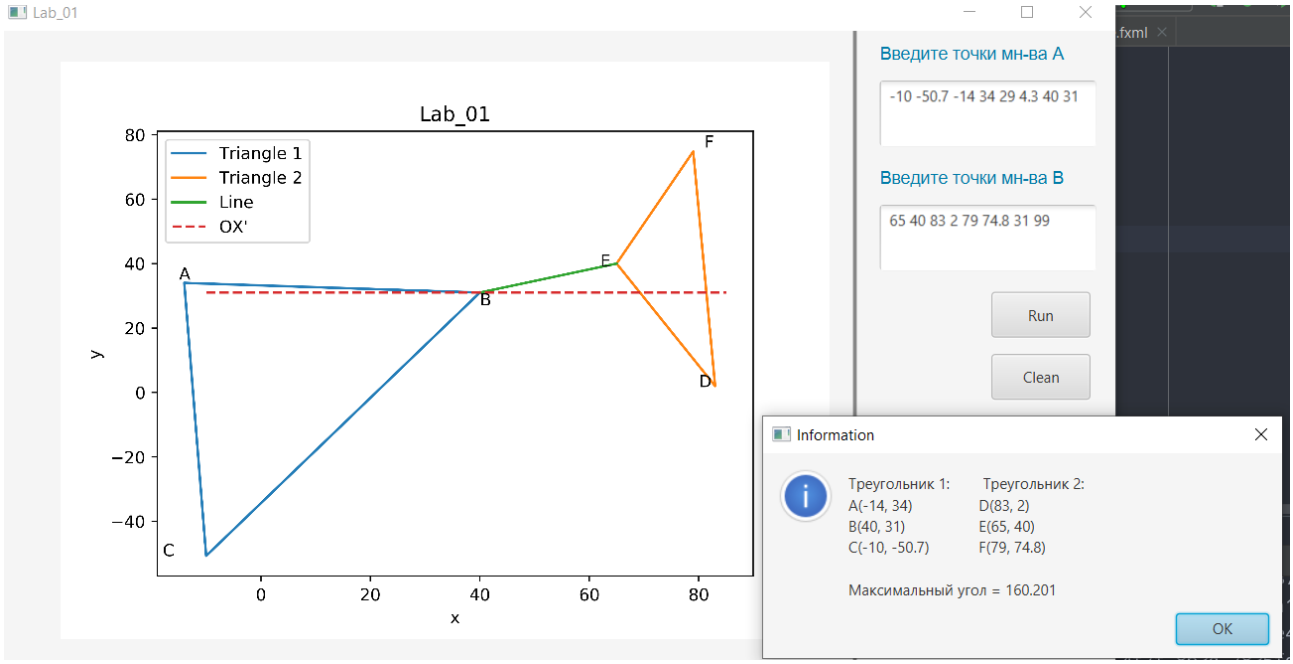
Работа с отрицательными и вещественными числами



Пример 5.



Пример 6.

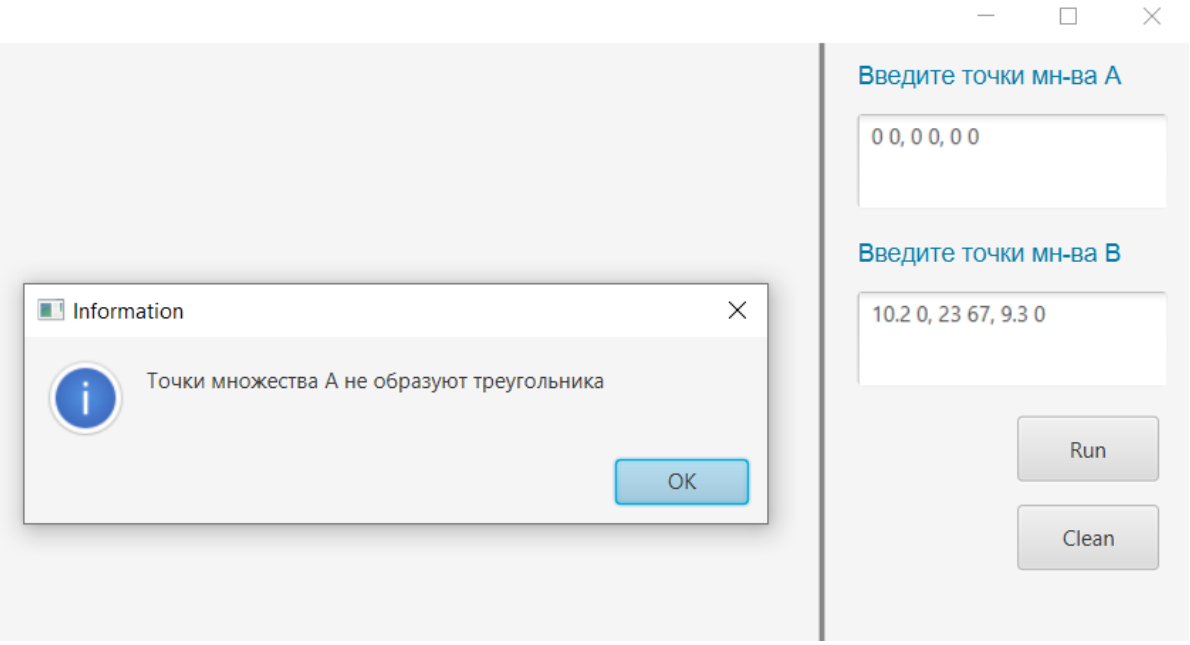

Lab_01
Lab_01
Triangle 1
Triangle 2
Line
OX'
A
B
C
D
E
F
x
y
Введите точки мн-ва A
-10 -50.7 -14 34 29 4.3 40 31
Введите точки мн-ва B
65 40 83 2 79 74.8 31 99
Run
Clean
Information
Треугольник 1:
A(-14, 34)
B(40, 31)
C(-10, -50.7)
Треугольник 2:
D(83, 2)
E(65, 40)
F(79, 74.8)
Максимальный угол = 160.201
OK

Пример 7.

Проверка на существование треугольника.

Некорректный ввод данных — точки множества A не образуют треугольника.

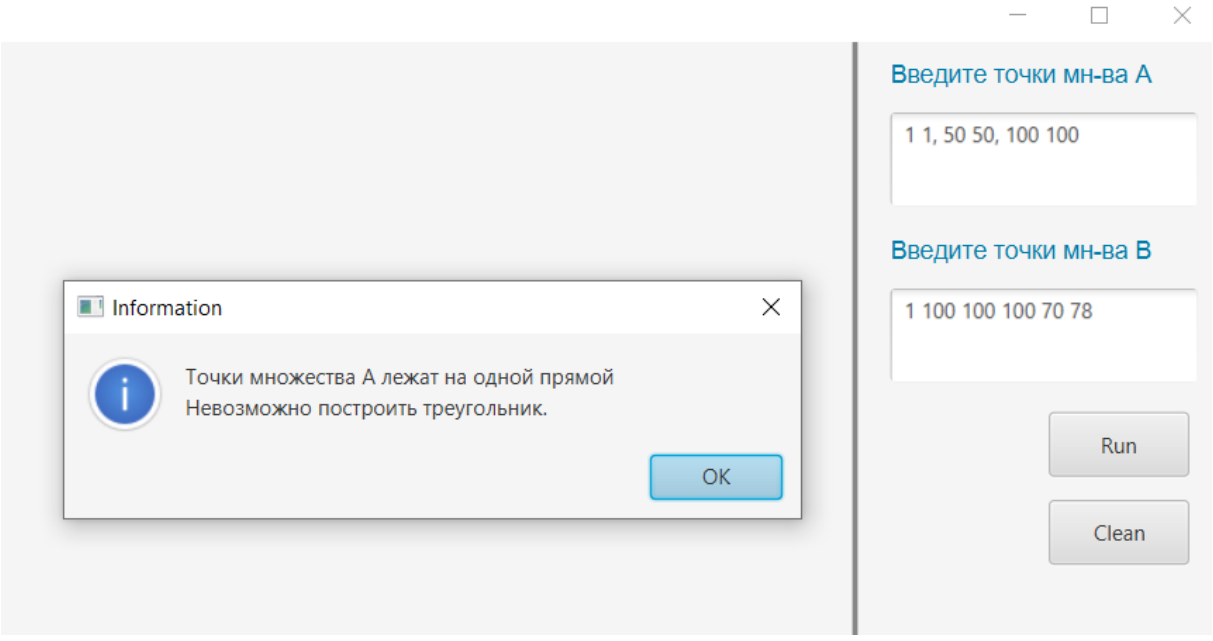
Множество **A** состоит из трех одинаковых точек. Для образования треугольника необходимо минимум три различные точки.



Пример 8.

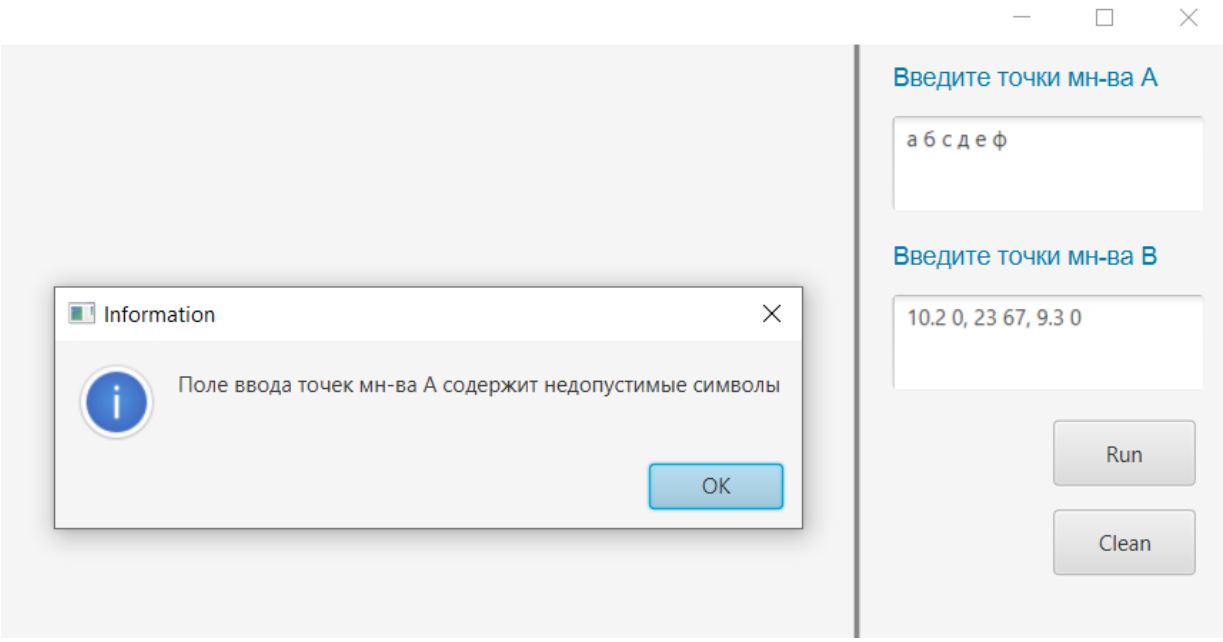
Проверка на существование треугольника.

Точки множества **A** лежат на одной прямой, поэтому треугольник по данным точкам образовать нельзя.



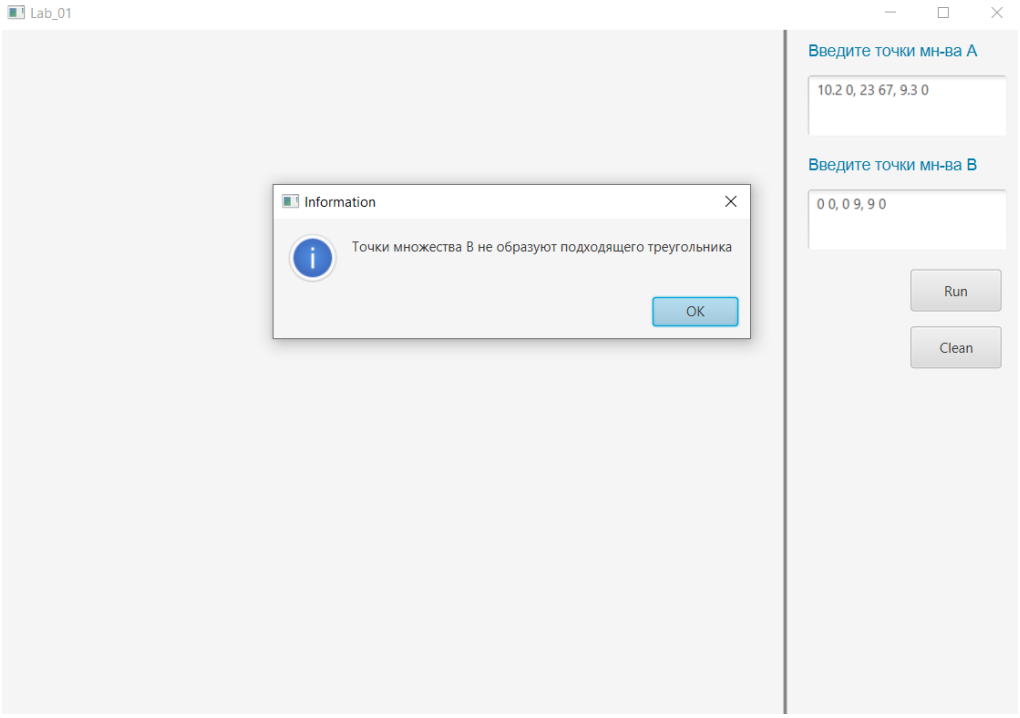
Пример 9.

Некорректный ввод данных.



Пример 10.

Точки множества B не образуют подходящего треугольника, т.к. из набора (0,0), (0, 9), (9, 0) нельзя построить треугольник с тупым углом.



Вывод.

В результате проделанной работы научился применять знания аналитической геометрии для решения практических задач машинной графики, осуществлять построение изображения (в СКУ) объектов, расположенных в МСК.