Физико-математический лицей № 239

**Поиск пары окружностей с минимальными радиусами, содержащими половину точек заданного множества**

Отчёт о домашней работе по информатике

Работу выполнил

Ученик 10-7 класса

Александров Владислав

Санкт-Петербург

2021

**1. Постановка задачи**

На плоскости задано множество точек A. Найти такие две окружности a и b, что их центры находятся в точках заданного множества, внутри каждой из этих окружностей находятся хотя бы половина из всех точек заданного множества, и меньший из двух радиусов минимален.

**2. Уточнение исходных и выходных данных и ограничений на них**

**2.1. Исходные данные**

Во входном файле points.txt содержатся строки с координатами точек в формате **x y**. Например, **0.1 -1.** Количество строк заранее неизвестно. Дробная часть может присутствовать, число может быть отрицательным, и это верно для всех вводимых чисел, так как это координаты. Координаты ограничены размером экрана, то есть, могут иметь значения от -1 до 1.

**2.2. Выходные данные**

Необходимо вывести на экран введённое множество точек, найти две окружности, содержащие хотя бы половину всех этих точек, при этом меньшая окружность должна иметь минимальный радиус, вывести эти две окружности и выделить все точки, лежащие внутри меньшей.

**3. Выбор метода решения**

**3.1. Анализ исходных данных и выбор используемой структуры данных**

Мы имеем координаты точек. Каждую точку нужно нарисовать, при этом они могут добавляться, поэтому лучше всего создать класс Points (для более удобного решения задачи) и создать список класса Points, который будет расширяться по необходимости. У нас также должны быть две окружности и их ещё и нужно найти, поэтому создадим класс Circle, объекты которого будут задаваться координатами центра и величиной радиуса. Вдобавок к этому, создадим два объекта этого класса, которые мы будем рисовать.

**3.2. Выбор метода решения**

Считываем построчно координаты из файла, создаём из них объекты класса Point и автоматически добавляем эти объекты в список points. Создадим метод inside в классе Circle, проверяющий, находится ли выбранная в качестве аргумента точка внутри окружности. Очевидно, что минимальный радиус окружности, в которой содержится какая-то точка, равен расстоянию от центра окружности до этой точки. Поэтому также создадим метод SetR, который в качестве аргумента будет брать точку и менять радиус окружности на минимальный, содержащий эту точку.

Далее перебираем 3 точки из списка для нахождения одной минимальной окружности: одна является центром окружности, вторая – крайней точкой, то есть той точкой которая лежит на окружности, третьи точки мы перебираем отдельно, они используются для того, чтобы выяснить, находится ли внутри той окружности половина множества всех точек. Фиксируем минимальную найденную окружность, после перебора записываем её в заготовленный заранее объект.

Вторую окружность ищем так же, только при условии, что она не равна первой.

Затем ещё раз перебираем все точки, проверяя, находятся ли они внутри первой окружности. Если да, то мы их перекрашиваем.

**4.** **Составление алгоритма**

**4.1. Обобщённая блок-схема алгоритма**

Начало

Если перебираемая окружность имеет радиус, меньше минимального найденного, и удовлетворяет условиям задачи, то делаем её минимальной. В конце циклов будет найдена итоговая минимальная окружность.

Перебор 3 точек: точек центра, крайних точек и внутренних точек окружности.

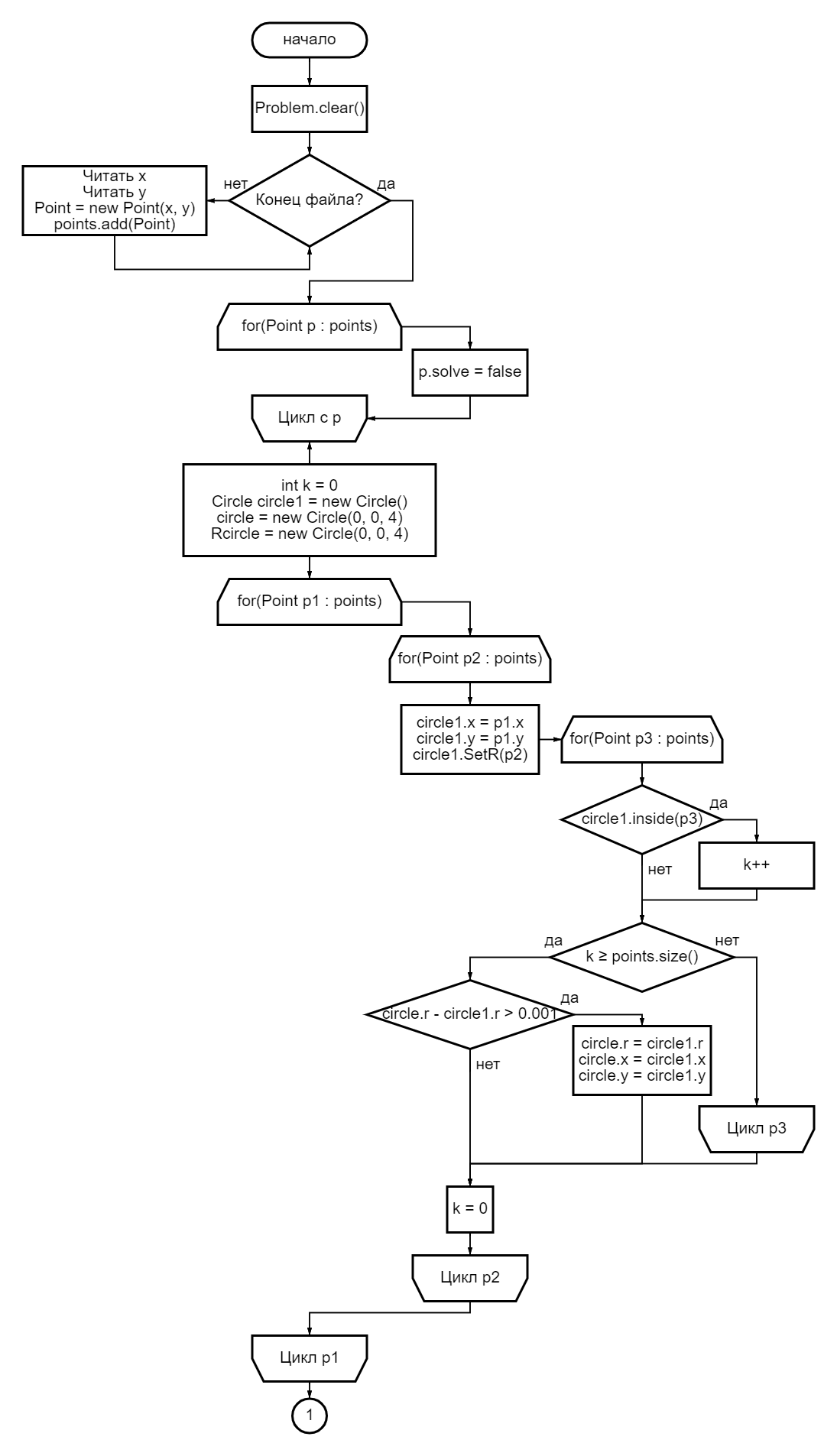
После нажатия кнопки «Решить»: перекраска всех точек в нейтральный цвет.

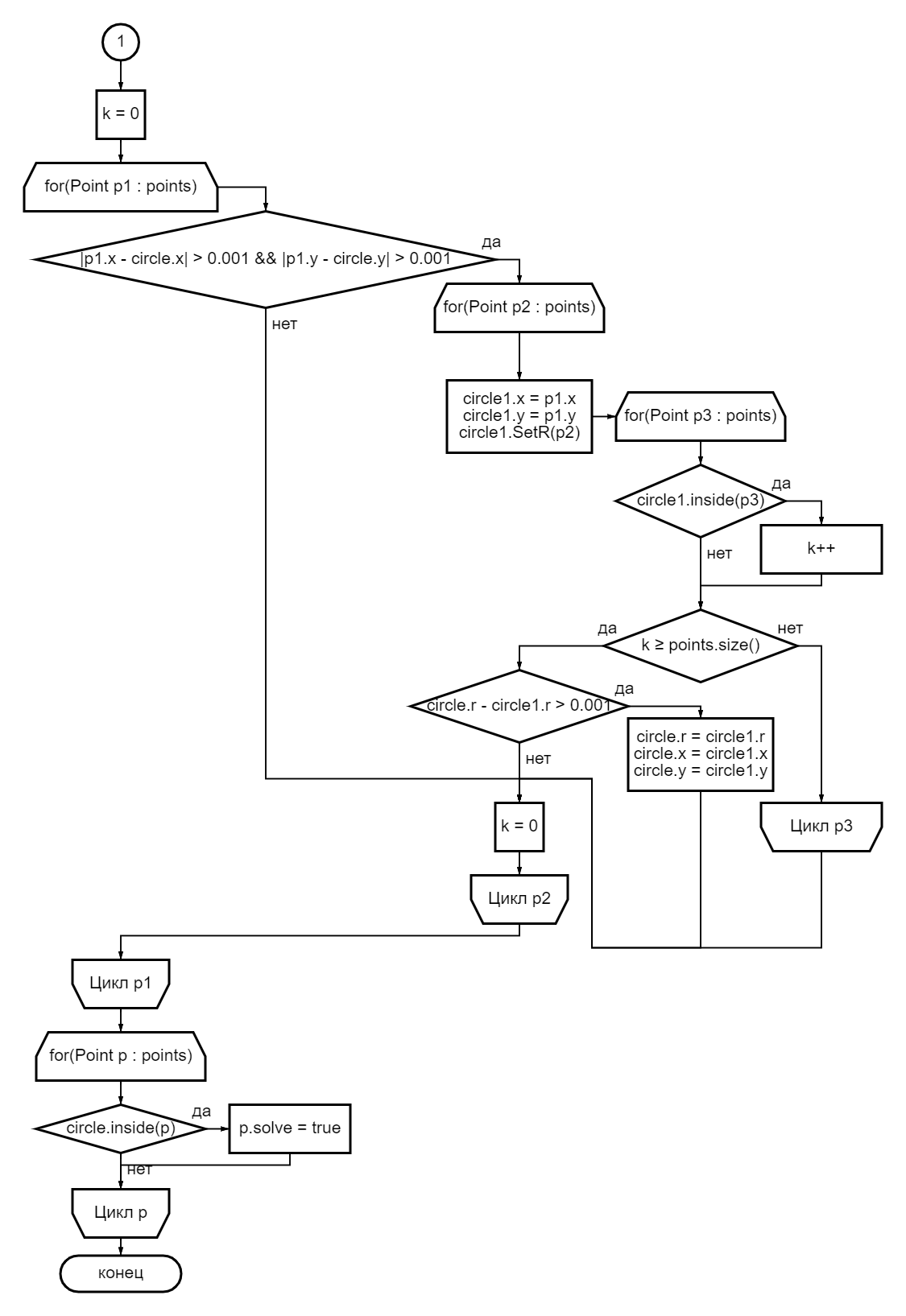
Чтение исходных данных. Рисование множества точек.

Нахождение второй окружности аналогичным образом с проверкой на неравенство первой окружности.

Перебор всех точек с проверкой на нахождение их внутри первой окружности и их последующая перекраска.

Конец

** 4.2. Блок-схема алгоритма**

****

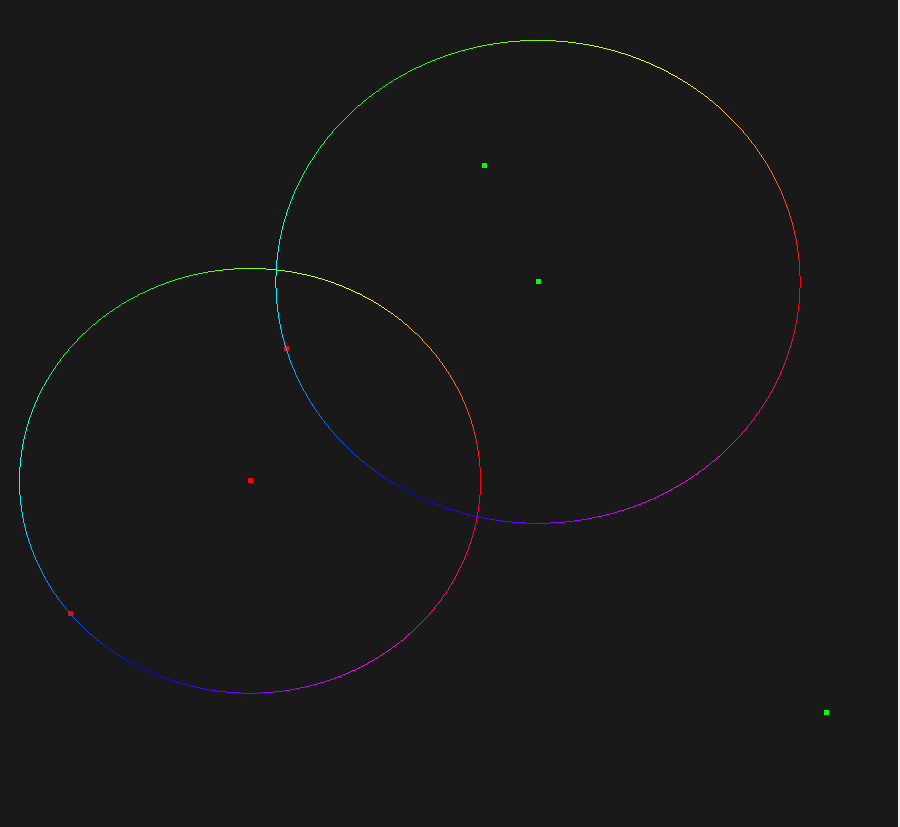
**5. Листинг программы**

import javax.media.opengl.GL2;  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.Random;  
import java.util.Scanner;  
  
*/\*\*  
 \* Класс задачи  
 \*/*public class Problem {  
 */\*\*  
 \* текст задачи  
 \*/* public static final String *PROBLEM\_TEXT* = "ПОСТАНОВКА ЗАДАЧИ:\n" +  
 "На плоскости задано множество точек.\n" +  
 "Найти такие две окружности,\n" +  
 "Что их центры находятся в точках заданного множества, \n" +  
 "Внутри каждой из этих окружностей \n" +  
 "Находится хотя бы половина из всех точек заданного множества. \n" +  
 "Меньший из двух радиусов минимален.";  
  
 */\*\*  
 \* заголовок окна  
 \*/* public static final String *PROBLEM\_CAPTION* = "Итоговый проект ученика 10-7 Александрова Владислава";  
  
 */\*\*  
 \* путь к файлу  
 \*/* private static final String *FILE\_NAME* = "points.txt";  
  
 */\*\*  
 \* список точек  
 \*/* private ArrayList<Point> points;  
 */\*\*  
 \* circle - минимальная окружность  
 \* Rcircle - вторичная.  
 \*/* private Circle circle;  
 private Circle Rcircle;  
  
 */\*\*  
 \* Конструктор класса задачи  
 \*/* public Problem() {  
 points = new ArrayList<>();  
 }  
  
 */\*\*  
 \* Добавить точку  
 \*  
 \** ***@param*** *x координата X точки  
 \** ***@param*** *y координата Y точки  
 \*/* public void addPoint(double x, double y) {  
 Point point = new Point(x, y);  
 points.add(point);  
 }  
  
 */\*\*  
 \* Решить задачу  
 \*/* public void solve() {  
 for (Point p : points){ // Перекрашиваем точки в обычный цвет, если они были перекрашены при предыдущем решении  
 p.solve = false;// solve - поле класса Point, отвечающее за цвет.  
 }  
 Circle circle1 = new Circle(); // Временная окружность, в которую мы перезаписываем все перебираемые окружности  
 circle = new Circle(0, 0, 99);  
 int k = 0; // Счётчик точек, находящихся внутри окружности.  
 for (Point p : points){ // Перебираем точки центра окружности  
 for (Point p2 : points){ // Перебираем точки радиуса окружности  
 circle1.x = p.x;  
 circle1.y = p.y;  
 circle1.SetR(p2);  
 for (Point p3 : points){ // Перебираем точки множества  
 if (circle1.inside(p3)){ // Записываем, сколько их лежит внутри перебираемой окружности  
 k++;  
 }  
 if (((k == points.size() / 2) && (points.size() % 2 == 0)) || ((k == (points.size() / 2) + 1) && (points.size() % 2 == 1))){  
 */\*\*  
 \* Перестаём перебирать точки, когда уже ясно, что внутри окружности содержится половина всего множества точек.  
 \* Если эта окружность меньше минимальной найденной(которая записывается в circle), то записываем её как минимальную.  
 \*/* if (circle.r - circle1.r > 0.001){  
 circle.r = circle1.r;  
 circle.x = circle1.x;  
 circle.y = circle1.y;  
 }  
 break;  
 }  
 }  
 k = 0;  
 }  
 }  
 Rcircle = new Circle(0, 0, 99);  
 k = 0;  
 // Далее проводим аналогичные действия со второй окружность, проверяя, не совпадают ли центры первой и второй.  
 for (Point p : points){  
 for (Point p2 : points){  
 if ((Math.*abs*(p.x - circle.x) > 0.001) && (Math.*abs*(p.y - circle.y) > 0.001)) {  
 circle1.x = p.x;  
 circle1.y = p.y;  
 circle1.SetR(p2);  
 for (Point p3 : points) {  
 if (circle1.inside(p3)) {  
 k++;  
 }  
 if (((k == points.size() / 2) && (points.size() % 2 == 0)) || ((k == (points.size() / 2) + 1) && (points.size() % 2 == 1))) {  
 if (Rcircle.r > circle1.r) {  
 Rcircle.r = circle1.r;  
 Rcircle.x = circle1.x;  
 Rcircle.y = circle1.y;  
 }  
 break;  
 }  
 }  
 }  
 k = 0;  
 }  
 }  
 for (Point p : points){ // Перебираем точки множества, перекрашиваем, если они лежат внутри минимальной окружности.  
 if (circle.inside(p)){  
 p.solve = true;  
 }  
 }  
 }  
  
 */\*\*  
 \* Загрузить задачу из файла  
 \*/* public void loadFromFile() {  
 points.clear();  
 try {  
 File file = new File(*FILE\_NAME*);  
 Scanner sc = new Scanner(file);  
 // пока в файле есть непрочитанные строки  
 while (sc.hasNextLine()) {  
 double x = sc.nextDouble();  
 double y = sc.nextDouble();  
 sc.nextLine();  
 Point point = new Point(x, y);  
 points.add(point);  
 }  
 } catch (Exception ex) {  
 System.*out*.println("Ошибка чтения из файла: " + ex);  
 }  
 }  
  
 */\*\*  
 \* Сохранить задачу в файл  
 \*/* public void saveToFile() {  
 try {  
 PrintWriter out = new PrintWriter(new FileWriter(*FILE\_NAME*));  
 for (Point point : points) {  
 out.printf("%.2f %.2f\n", point.x, point.y);  
 }  
 out.close();  
 } catch (IOException ex) {  
 System.*out*.println("Ошибка записи в файл: " + ex);  
 }  
 }  
  
 */\*\*  
 \* Добавить заданное число случайных точек  
 \*  
 \** ***@param*** *n кол-во точек  
 \*/* public void addRandomPoints(int n) {  
 for (int i = 0; i < n; i++) {  
 Point p = Point.*getRandomPoint*();  
 points.add(p);  
 }  
 }  
  
 */\*\*  
 \* Очистить задачу  
 \*/* public void clear() {  
 points.clear();  
 circle = null;  
 Rcircle = null;  
 }  
  
 */\*\*  
 \* Нарисовать задачу  
 \*  
 \** ***@param*** *gl переменная OpenGL для рисования  
 \*/* public void render(GL2 gl) {  
 for (Point point : points) {  
 point.render(gl);  
 }  
 if (circle != null){circle.render(gl);}  
 if (Rcircle != null){  
 Rcircle.render(gl);  
 }  
 }  
}

**6. Пример работы программы**

**6.1. Исходные данные**

0,20 0,32  
-0,44 -0,16  
0,84 -0,72  
0,08 0,60  
-0,36 0,16  
-0,84 -0,48

**6.2 Выходные данные**