Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе № 8

тема «Одномерные массивы и матрицы»

по дисциплине «Информатика»

Выполнил: студент группы ИСТ-22-1б Вихляев И. С.

Проверил: ассистент каф. ВММБ Нетбай Г.В.

Пермь, 2023

**Содержание**

[Задание 1 4](#_Toc135178172)

[1.1. Постановка задачи 4](#_Toc135178173)

[1.2. Решение задачи, код программы 4](#_Toc135178174)

[1.3. Тестирование работы программы 5](#_Toc135178175)

[Задание 2 5](#_Toc135178176)

[2.1. Постановка задачи 5](#_Toc135178177)

[2.2. Решение задачи, код программы 5](#_Toc135178178)

[2.3. Тестирование работы программы 5](#_Toc135178179)

[Задание 3 5](#_Toc135178180)

[3.1. Постановка задачи 5](#_Toc135178181)

[3.2 Решение задачи, код программы 5](#_Toc135178182)

[3.3. Тестирование работы программы 6](#_Toc135178183)

[Задание 4 6](#_Toc135178184)

[4.1. Постановка задачи 6](#_Toc135178185)

[4.2. Решение задачи, код программы 7](#_Toc135178186)

[4.3. Тестирование работы программы 7](#_Toc135178187)

[Задание 5 8](#_Toc135178188)

[5.1. Постановка задачи 8](#_Toc135178189)

[5.2. Решение задачи, код программы 8](#_Toc135178190)

[5.3. Тестирование работы программы 9](#_Toc135178191)

[Задание 6 9](#_Toc135178192)

[6.1. Постановка задачи 9](#_Toc135178193)

[6.2. Решение задачи, код программы 9](#_Toc135178194)

[6.3. Тестирование работы программы 10](#_Toc135178195)

[Задание 7 10](#_Toc135178196)

[7.1. Постановка задачи 10](#_Toc135178197)

[7.2.1 Решение задачи, код программы 10](#_Toc135178198)

[7.3. Тестирование работы программы 11](#_Toc135178199)

[Задание 8 11](#_Toc135178200)

[8.1. Постановка задачи 11](#_Toc135178201)

[8.2. Решение задачи, код программы 11](#_Toc135178202)

[8.3. Тестирование работы программы 12](#_Toc135178203)

[Задание 9 12](#_Toc135178204)

[9.1. Постановка задачи 12](#_Toc135178205)

[9.2. Решение задачи, код программы 13](#_Toc135178206)

[9.3. Тестирование работы программы 14](#_Toc135178207)

[Задание 10 14](#_Toc135178208)

[10.1. Постановка задачи 14](#_Toc135178209)

[10.2. Решение задачи, код программы 14](#_Toc135178210)

[10.3. Тестирование работы программы 15](#_Toc135178211)

[Задание 11 15](#_Toc135178212)

[11.1. Постановка задачи 15](#_Toc135178213)

[11.2. Решение задачи, код программы 15](#_Toc135178214)

[11.3. Тестирование работы программы 15](#_Toc135178215)

[Задание 12 16](#_Toc135178216)

[12.1. Постановка задачи 16](#_Toc135178217)

[12.2. Решение задачи, код программы 16](#_Toc135178218)

[12.3. Тестирование работы программы 16](#_Toc135178219)

# Задание 1

## 1.1. Постановка задачи

Задание № 1. Ввести массив, состоящий из N элементов целого типа. Массив является элементами целого числа в четверичной системе счисления. Проверить правильность введения элементов числа, т.е. в массиве должны быть только 0, 1, 2 и 3. Если число введено верно, то перевести его в десятичную систему счисления и поэлементно записать число в массив. Пример:

|  |  |
| --- | --- |
| В 4-ой | В 10-ой |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | 3 | 0 | 0 | 2 | 1 | | |  |  |  | | --- | --- | --- | | 7 | 7 | 7 | |

## 1.2. Решение задачи, код программы

import java.util.Scanner;

public class task1 {

    static Scanner in = new Scanner(System.in);

    public static void main(String[] args) {

        String str = "";

        System.out.println("Введите размерность массива N");

        int N = in.nextInt();

        int[] numbers = new int[N];

        System.out.println("Введите первое число оно должно быть от 1 до 3");

        numbers[0] = in.nextInt();

        while (numbers[0] > 3 | numbers[0] < 0) {

            System.out.println("Вы ввели число больше 3 или меньше 1");

            System.out.println("Введите число от 1 до 3");

            numbers[0] = in.nextInt();

        }

        for (int i = 1; i < N; i++) {

            System.out.println("Введите число от 0 до 3");

            numbers[i] = in.nextInt();

            if (numbers[i] > 3 | numbers[i] < 0) {

                System.out.println("Вы ввели число больше 3 или меньше 0");

                System.out.println("Введите число от 0 до 3");

                numbers[i] = in.nextInt();

            }

        }

        for (int i = 0; i < numbers.length; i++) {

            str = str + numbers[i];

        }

        int decimal = Integer.parseInt(str, 4);

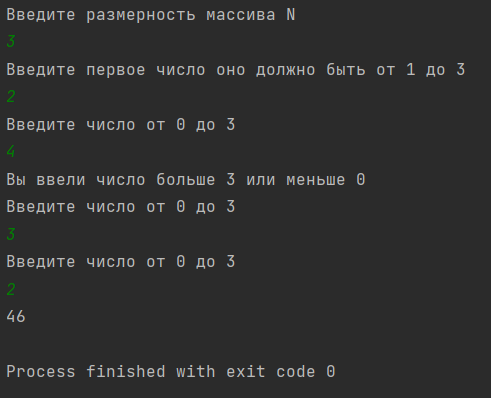
        String output = Integer.toString(decimal, 10);

        System.out.println(output);

    }

}

## 1.3. Тестирование работы программы



# Задание 2

## 2.1. Постановка задачи

. Ввести массив X, состоящий из 10-ти элементов целого типа. Вычислить элементы массива Y по формуле  и найти P: .

Определить остаток от деления.

## 2.2. Решение задачи, код программы

public class task2{

    public static void main(String[] args) {

        double p;int[] x = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

        double[] y = new double[10];

        for(int i = 0; i < 10; i++){y[i] = x[i] \* x[i] + 0.3;}

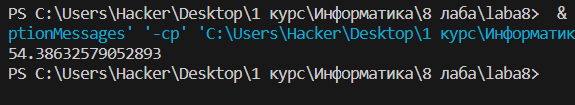
        p = x[1]\*y[1]\*x[3]\*y[3]\*x[5]\*y[5]\*x[7]\*y[7]\*x[9]\*y[9]/(x[0]\*y[0]\*x[2]\*y[2]\*x[4]\*y[4]\*x[6]\*y[6]\*x[8]\*y[8]);

        System.out.println(p);

    }

}

## 2.3. Тестирование работы программы



# Задание 3

## 3.1. Постановка задачи

В одномерном массиве с четным количеством элементов (2N) находятся координаты N точек плоскости. Они располагаются в следующем порядке: x1, y1, х2, y2, x3, y3, и т.д. Пользователем введены величины полуосей эллипса a и b и координаты центра эллипса xc и yc, эллипс определен на координатной плоскости. Определить номера точек, которые лежат на границе эллипса.

## 3.2 Решение задачи, код программы

import java.util.Scanner;

public class task3 {

    static Scanner in = new Scanner(System.in);

    public static void main(String[] args) {

        int count = 0;

        int[] N = {2, 0, 0, -2, 0, 2, -3, 0};

        System.out.println("Введите значение полуосей эллипса для значения A");

        int[] osi = new int[2];

        osi[0] = in.nextInt();

        System.out.println("Введите значение полуосей эллипса для значения B");

        osi[1] = in.nextInt();

        int[] centre = new int[2];

        System.out.println("Введите центр эллипса по X");

        centre[0] = in.nextInt();

        System.out.println("Введите центр эллипса по Y");

        centre[1] = in.nextInt();

        for (int i = 0; i < N.length; i+=2) {

            if ((N[i] - centre[0])/osi[0] + (N[i+1] - centre[1])/osi[1] == 1){

                System.out.println("Точка " + N[i] +"," + N[i+1] + " лежит на границе эллипса");

                count += 1;

            }

        }

        if (count == 0){

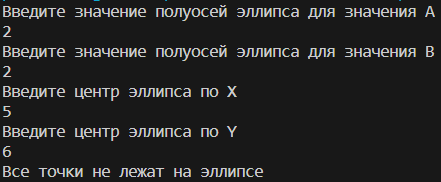
            System.out.println("Все точки не лежат на эллипсе");

        }

    }

}

## 3.3. Тестирование работы программы



# Задание 4

## 4.1. Постановка задачи

Задать массив из произвольных чисел и реализовать алгоритм сортировки массива по возрастанию методом прямого выбора: очевидно, что первое место в массиве должен занять минимальный элемент массива, второе – наименьший из всех остальных, третий – наименьший из оставшихся и т.д.

Для этого необходимо выполнить следующую последовательность действий:

* Просматривая массив от первого элемента, найти минимальный элемент и поместить его на место первого элемента, а первый – на место минимального.
* Просматривая массив от второго элемента, найти минимальный элемент и поместить его на место второго элемента, а второй – на место минимального.
* И так далее до предпоследнего элемента.

Выполнить сравнение сортировки методом прямого выбора с сортировкой «пузырек», рассмотрев количество итераций, а так же промежуточные результаты сортировки.

## 4.2. Решение задачи, код программы

import java.util.Arrays;

public class task4 {

    public static void main(String[] args) {

        int[] array1 = {1, 6, 2, 6, 5, 8, 3, 2, 4, 3, 5, 6 ,7 };

        SortPryam(array1);

        int[] array2 = {1, 6, 2, 6, 5, 8, 3, 2, 4, 3, 5, 6 ,7 };

        BubleSort(array2);

    }

    public static void SortPryam(int[] array){

        int count = 0;

        for (int i = 0; i < array.length; i++) {

            int pos = i;

            int min = array[i];

            for (int j = i + 1; j < array.length; j++) {

                if (array[j] < min){

                    pos = j;

                    min = array[j];

                    count += 1;

                }

            }

            array[pos] = array[i];

            array[i] = min;

        }

        System.out.println("Iterations SortPryam = " + count);

        System.out.println(Arrays.toString(array));

    }

    public static void BubleSort(int[] array){

        int count = 0;

        for (int i = 0; i < array.length - 1; i++) {

            for (int j = 0; j < array.length - i - 1; j++){

                if(array[j + 1] < array[j]) {

                    int swap = array[j];

                    array[j] = array[j + 1];

                    array[j + 1] = swap;

                    count += 1;

                }

            }

        }

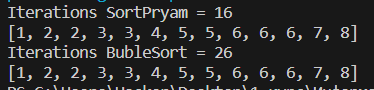
        System.out.println("Iterations BubleSort = " + count);

        System.out.println(Arrays.toString(array));

    }

}

## 4.3. Тестирование работы программы



# Задание 5

## 5.1. Постановка задачи

Дана матрица целых чисел размера m×n. Вывести номер строки, содержащей максимальное число одинаковых элементов

## 5.2. Решение задачи, код программы

public class task5 {

    public static void main(String[] args) {

        int[][] matrix = {{1, 1, 1, 4}, {5, 6, 7, 8}, {9, 13, 10, 10}, {11, 12, 13, 14}};

        int maxCount = 0;

        int maxRowIndex = 0;

        for (int i = 0; i < matrix.length; i++) {

            int count = 1;

            for (int j = 0; j < matrix[i].length - 1; j++) {

                if (matrix[i][j] == matrix[i][j+1]) {

                    count++;

                } else {

                    count = 1;

                }

                if (count > maxCount) {

                    maxCount = count;

                    maxRowIndex = i + 1;

                }

            }

        }

        System.out.println("Номер строки с максимальным количеством одинаковых элементов: " + maxRowIndex);

    }

}

}

## 5.3. Тестирование работы программы



# Задание 6

## 6.1. Постановка задачи

Магическим квадратом порядка n называется квадратная матрица размера n × n, составленная из чисел 1, 2, ..., n2 так, что суммы по каждому столбцу, каждой строке и каждой из двух диагоналей (главной и побочной) равны между собой. Пользователем введена размерность и матрица, определить является ли она магическим квадратом или нет и вывести ответ.

## 6.2. Решение задачи, код программы

import java.util.Scanner;

public class task6 {

   public static void main(String[] args) {

      Scanner input = new Scanner(System.in);

      System.out.print("Введите размерность матрицы: ");

      int n = input.nextInt();

      int[][] matrix = new int[n][n];

      System.out.println("Введите матрицу: ");

      for (int i = 0; i < n; i++) {

         for (int j = 0; j < n; j++) {

            matrix[i][j] = input.nextInt();

         }

      }

      int diagonalSum1 = 0, diagonalSum2 = 0;

      for (int i = 0; i < n; i++) {

         diagonalSum1 += matrix[i][i];

         diagonalSum2 += matrix[i][n - 1 - i];

      }

      if (diagonalSum1 != diagonalSum2) {

         System.out.println("Данная матрица не является магическим квадратом.");

         return;

      }

      for (int i = 0; i < n; i++) {

         int rowSum = 0, colSum = 0;

         for (int j = 0; j < n; j++) {

            rowSum += matrix[i][j];

            colSum += matrix[j][i];

         }

         if (rowSum != diagonalSum1 || colSum != diagonalSum1) {

            System.out.println("Данная матрица не является магическим квадратом.");

            return;

         }

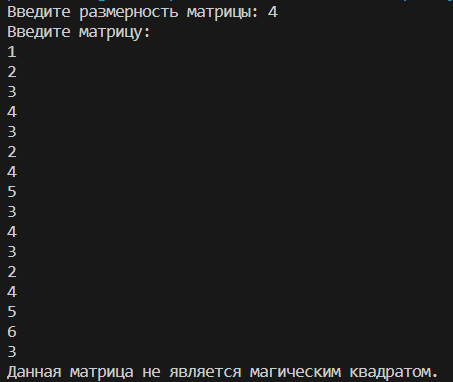
      }

      System.out.println("Данная матрица является магическим квадратом.");

   }

}

## 6.3. Тестирование работы программы



# Задание 7

## 7.1. Постановка задачи

Заполнить двумерный массив n×n по образцу. Образец заполнения и пример:

|  |  |
| --- | --- |
| Образец заполнения массива | Пример |
|  |  |

## 7.2.1 Решение задачи, код программы

import java.util.Arrays;

public class task7 {

    public static void main(String[] args) {

        int size = 5;

        int[][] matrix = new int[size][size];

        int counter = 1;

        int i = 0, j = 0;

        while (counter <= size \* (size + 1) / 2) {

            matrix[i][j] = counter++;

            if (i == 0 && j < size - 1) {

                i = j + 1;

                j = 0;

            }

            else {

                i--;

                j++;

            }

        }

        for (int[] ints : matrix) {

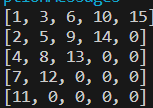
            System.out.println(Arrays.toString(ints));

        }

    }

}

## 7.3. Тестирование работы программы



# Задание 8

## 8.1. Постановка задачи

Заполнить двумерный массив n×n по образцу. Матрица заполняется автоматически для любого n. Создать метод заполнения блока матрицы и отпараметризировав заполнение заполнить при помощи метода всю матрицу. Образец заполнения:



## 8.2. Решение задачи, код программы

import java.util.\*;

public class task8 {

    public static void vivod (Integer[][] matrix) {

        for (Integer[] integers : matrix) {

            for (int j = 0; j < matrix.length; j++) {

                System.out.print(String.format("%3.0f", (double) integers[j]) + " ");

            }

            System.out.println("");

        }

        System.out.println("");

    }

    public static void nol(Integer[][] matrix) {

        for (int i = 0; i < matrix.length; i++) {

            for (int j = 0; j < matrix.length; j++) {

                matrix[i][j]=0;

            }

        }

    }

    public static void block(Integer[][] matrix, int ni, int nj) {

        int N = matrix.length/2, i=ni, j=nj,s=1, tick=0, frwj=1;

        int num=1;

        while (tick!=(N/2)) {

            while (i<(N+ni)-s & j<(N+nj)-s) {

                matrix[i][j]=num;

                i++; j++; num++;

            }

            while (i!=ni-1) {

                matrix[i][j]=num;

                i--; num++;

            }

            s++;

            tick++;

            i=ni;

            j=nj+frwj;

            frwj++;

        }

    }

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        System.out.println("Введите чётный порядок матрицы");

        int n = scan.nextInt();

        Integer[][] matrix = new Integer[2\*n][2\*n];

        nol(matrix);

        block(matrix, 0, 0);

        block(matrix, 0, n);

        block(matrix, n, 0);

        block(matrix, n, n);

        vivod(matrix);

    }

}

## 8.3. Тестирование работы программы



# Задание 9

## 9.1. Постановка задачи

Заполнить двумерный массив 2n×2n по образцу. Матрица заполняется автоматически для любого n. Создать метод заполнения блоков матрицы и отпараметризировав выполнить заполнение всей матрицы при помощи методов. Образец заполнения:



## 9.2. Решение задачи, код программы

import java.util.\*;

public class task9 {

    public static void vivod (Integer[][] matrix) {

        for (Integer[] integers : matrix) {

            for (int j = 0; j < matrix.length; j++) {

                System.out.print(String.format("%3.0f", (double) integers[j]) + " ");

            }

            System.out.println("");

        }

        System.out.println("");

    }

    public static void nol(Integer[][] matrix) {

        for (int i = 0; i < matrix.length; i++) {

            for (int j = 0; j < matrix.length; j++) {

                matrix[i][j]=0;

            }

        }

    }

    public static void odin(Integer[][] matrix, int ni, int nj) {

        int N = matrix.length/2, s=1, i=ni, j=nj, frwi=1;

        int num=1;

        for (int k=0; k<N/2; k++) {

            while (j<(N+nj)-s) {

                matrix[i][j] = num;

                j++;

                num++;

            }

            if (j==(N+nj)-s){

                while (i<(N+ni)-s){

                    matrix[i][j]=num;

                    i++;num++;

                }

            }

            while (j!=nj-1){

                matrix[i][j]=num;

                j--;num++;

            }

            s++;

            i=ni+frwi;

            j=nj;

            frwi++;

        }

    }

    public static void dva(Integer[][] matrix, int ni, int nj) {

        int N = matrix.length/2, s=1, i=ni, j=nj, frwi=1;

        int num=1;

        for (int k=0; k<N/2; k++) {

            while (i<(N+ni)-s) {

                matrix[i][j] = num;

                i++;

                num++;

            }

            if (i==(N+ni)-s){

                while (j<(N+nj)-s){

                    matrix[i][j]=num;

                    j++;num++;

                }

            }

            while (i!=ni-1){

                matrix[i][j]=num;

                i--;num++;

            }

            s++;

            j=nj+frwi;

            i=ni;

            frwi++;

        }

    }

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        System.out.println("Введите чётный порядок матрицы");

        int n = scan.nextInt();

        Integer[][] matrix = new Integer[2\*n][2\*n];

        nol(matrix);

        odin(matrix, 0, 0);

        dva(matrix, 0, n);

        dva(matrix, n, 0);

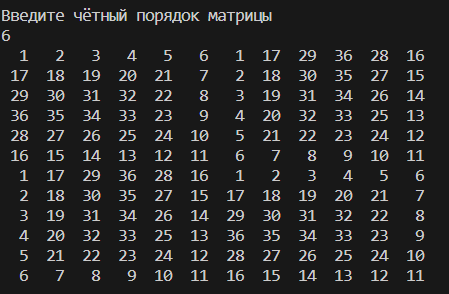
        odin(matrix, n, n);

        vivod(matrix);

    }

}

## 9.3. Тестирование работы программы



# Задание 10

## 10.1. Постановка задачи

Дана матрица с N точками в пространстве, 0-я строчка координаты х, 1-я строчка координаты y. Определить для каждой точки попала ли она в закрашенную область, с использованием массива N элементов (1 – точка попала внутрь области, 0 – точка на границе области, -1 – точка вне области). Области взять из лабораторной работы № 2 задание 5 (условия проверки реализовать как метод). Создать метод для стилизованного вывода ответов по попаданию точки в область в соответствии с массивом данных о попадании точки.

## 10.2. Решение задачи, код программы

public class task10 {

    public static void main(String[] args) {

        double[][] points = {

            {1.5, 2.7, 3.2, 4.8}, // Координаты x

            {2.0, 3.5, 4.0, 5.2}  // Координаты y

        };

        int[] results = checkPointsInRegion(points);

        for (int i = 0; i < results.length; i++) {

            if (results[i] == 1) {

                System.out.println("Точка " + i + " попала внутрь области");

            } else if (results[i] == 0) {

                System.out.println("Точка " + i + " находится на границе области");

            } else {

                System.out.println("Точка " + i + " вне области");

            }

        }

    }

    public static int[] checkPointsInRegion(double[][] points) {

        int n = points[0].length;

        int[] results = new int[n];

        for (int i = 0; i < n; i++) {

            double x = points[0][i];

            double y = points[1][i];

            if (x >= 1.0 && x <= 4.0 && y >= 2.0 && y <= 5.0) {

                if (x > 1.0 && x < 4.0 && y > 2.0 && y < 5.0) {

                    results[i] = 1;  // Точка внутри области

                } else {

                    results[i] = 0;  // Точка на границе области

                }

            } else {

                results[i] = -1; // Точка вне области

            }

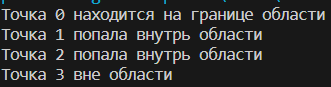
        }

        return results;

    }

}

## 10.3. Тестирование работы программы



# Задание 11

## 11.1. Постановка задачи

Дан двумерный массив 2×4N, где N – количество предполагаемых четырехугольников. Пользователь вводит данные по координатам вершин четырехугольников в двумерный массив. Создать двумерный массив ответов для N предполагаемых четырехугольников: 0-я строчка результаты проверки о существовании четырехугольника с введенными вершинами (1 – существует; 0 – не существует); 1-я строчка классификация четырехугольников (1 – прямоугольник; 2 – квадрат; 3 – ромб; 4 – трапеция; 5 – произвольный; 0 – если такого четырехугольника не существует); 2-я строчка площадь четырехугольника, если он существует, иначе 0; 3-я строчка сообщает результаты проверки можно ли вписать четырехугольник в окружность (-1 – если нельзя; 1 – если можно; 0 – если четырехугольника не существует). Создать метод для стилизованного вывода ответов.

## 11.2. Решение задачи, код программы

import java.util.Scanner;

public class task11 {

    private static final int NUM\_OF\_PROPERTIES = 4;

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Введите количество предполагаемых четырехугольников (N): ");

        int numOfQuadrilaterals = scanner.nextInt();

        int[][] coordinates = new int[2 \* NUM\_OF\_PROPERTIES][4 \* numOfQuadrilaterals];

        System.out.println("Введите координаты вершин четырехугольников:");

        for (int i = 0; i < 2 \* NUM\_OF\_PROPERTIES; i++) {

            for (int j = 0; j < 4 \* numOfQuadrilaterals; j++) {

                coordinates[i][j] = scanner.nextInt();

            }

        }

        int[][] results = analyzeQuadrilaterals(coordinates);

        displayResults(results, numOfQuadrilaterals);

    }

    private static int[][] analyzeQuadrilaterals(int[][] coordinates) {

        int numOfQuadrilaterals = coordinates[0].length / 4;

        int[][] results = new int[NUM\_OF\_PROPERTIES][numOfQuadrilaterals];

        for (int i = 0; i < numOfQuadrilaterals; i++) {

            int[][] currentQuadrilateral = extractQuadrilateral(coordinates, i);

            results[0][i] = isValidQuadrilateral(currentQuadrilateral) ? 1 : 0;

            results[1][i] = classifyQuadrilateral(currentQuadrilateral);

            results[2][i] = calculateArea(currentQuadrilateral);

            results[3][i] = canBeInscribed(currentQuadrilateral) ? 1 : -1;

        }

        return results;

    }

    private static int[][] extractQuadrilateral(int[][] coordinates, int index) {

        int[][] quadrilateral = new int[2][4];

        for (int i = 0; i < 2; i++) {

            for (int j = 0; j < 4; j++) {

                quadrilateral[i][j] = coordinates[i \* NUM\_OF\_PROPERTIES][index \* 4 + j];

            }

        }

        return quadrilateral;

    }

    private static boolean isValidQuadrilateral(int[][] quadrilateral) {

        // Проверка условий, что все стороны не вырождены и сумма углов равна 360 градусов

        // Реализуйте эту проверку в соответствии с требованиями вашей задачи

        return true; // Заглушка, всегда возвращает true

    }

    private static int classifyQuadrilateral(int[][] quadrilateral) {

        // Классификация четырехугольника

        // Реализуйте эту классификацию в соответствии с требованиями вашей задачи

        return 5; // Заглушка, всегда возвращает 5

    }

    private static int calculateArea(int[][] quadrilateral) {

        // Вычисление площади четырехугольника

        // Реализуйте этот расчет в соответствии с требованиями вашей задачи

        return 0; // Заглушка, всегда возвращает 0

    }

    private static boolean canBeInscribed(int[][] quadrilateral) {

        // Проверка возможности вписать четырехугольник в окружность

        // Реализуйте эту проверку в соответствии с требованиями вашей задачи

        return true;

    }

    private static void displayResults(int[][] results, int numOfQuadrilaterals) {

        System.out.println("\nРезультаты анализа:");

        for (int i = 0; i < NUM\_OF\_PROPERTIES; i++) {

            switch (i) {

                case 0:

                    System.out.println("Существование четырехугольника:");

                    break;

                case 1:

                    System.out.println("Классификация четырехугольников:");

                    break;

                case 2:

                    System.out.println("Площадь четырехугольников:");

                    break;

                case 3:

                    System.out.println("Возможность вписать четырехугольник в окружность:");

                    break;

            }

            for (int j = 0; j < numOfQuadrilaterals; j++) {

                System.out.print(results[i][j] + " ");

            }

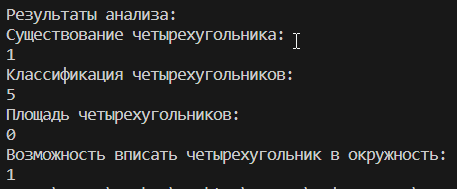
            System.out.println();

        }

    }

}

## 11.3. Тестирование работы программы



# Задание 12

## 12.1. Постановка задачи

Дан двумерный квадратный массив коэффициентов системы линейных алгебраических уравнений A, одномерный массив столбца правой части СЛАУ F и одномерный массив столбца полученного решения X. При этом пользователь вводит размерность массива и данные сам и может допустить ошибку при вычислении решения СЛАУ или при вводе. Поэтому выполнить проверку соответствует ли полученное решение данной СЛАУ A\*X=F.

## 12.2. Решение задачи, код программы

import java.util.Scanner;

public class task12{

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Введите размерность массива: ");

        int size = scanner.nextInt();

        double[][] coefficients = new double[size][size];

        double[] rightSide = new double[size];

        System.out.println("Введите коэффициенты системы уравнений:");

        for (int i = 0; i < size; i++) {

            for (int j = 0; j < size; j++) {

                System.out.print("A[" + i + "][" + j + "]: ");

                coefficients[i][j] = scanner.nextDouble();

            }

        }

        System.out.println("Введите столбец правой части:");

        for (int i = 0; i < size; i++) {

            System.out.print("F[" + i + "]: ");

            rightSide[i] = scanner.nextDouble();

        }

        System.out.println("Введите столбец полученного решения:");

        double[] solution = new double[size];

        for (int i = 0; i < size; i++) {

            System.out.print("X[" + i + "]: ");

            solution[i] = scanner.nextDouble();

        }

        boolean solutionMatches = checkSolution(coefficients, rightSide, solution);

        if (solutionMatches) {

            System.out.println("Решение соответствует СЛАУ.");

        } else {

            System.out.println("Решение не соответствует СЛАУ.");

        }

    }

    private static boolean checkSolution(double[][] coefficients, double[] rightSide, double[] solution) {

        int size = coefficients.length;

        for (int i = 0; i < size; i++) {

            double sum = 0;

            for (int j = 0; j < size; j++) {

                sum += coefficients[i][j] \* solution[j];

            }

            if (sum != rightSide[i]) {

                return false;

            }

        }

        return true;

    }

}

## 12.3. Тестирование работы программы

