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 «Массивы»

по дисциплине «Информатика»

Вариант 1.

Выполнил: студент группы ИСТ-23-1Б Шустов А.В.

Проверил: Нетбай Георгий Владимирович.

Пермь, 2023

Оглавление

[Задание 1 4](#_Toc162981262)

[1.1 Постановка задачи 4](#_Toc162981263)

[1.2 Решение на java 4](#_Toc162981264)

[1.3 Проверка результатов 5](#_Toc162981265)

[Задание 2 5](#_Toc162981266)

[2.1 Постановка задачи 5](#_Toc162981267)

[2.2 Решение на Java 5](#_Toc162981268)

[2.3 Проверка результатов 6](#_Toc162981269)

[Задание 3 6](#_Toc162981270)

[3.1 Постановка задачи 6](#_Toc162981271)

[3.2 Решение на java 6](#_Toc162981272)

[3.3 Проверка результатов 7](#_Toc162981273)

[Задание 4 7](#_Toc162981274)

[4.1 Постановка задачи 7](#_Toc162981275)

[4.2 Решение на Java 7](#_Toc162981276)

[4.3 Проверка результатов 8](#_Toc162981277)

[Задание 5 8](#_Toc162981278)

[5.1 Постановка задачи 8](#_Toc162981279)

[5.2 Решение на java 8](#_Toc162981280)

[5.3 Проверка результатов 9](#_Toc162981281)

[Задание 6 9](#_Toc162981282)

[6.1 Постановка задачи 9](#_Toc162981283)

[6.2 Решение на java 9](#_Toc162981284)

[6.3 Проверка результатов 10](#_Toc162981285)

[Задание 7 10](#_Toc162981286)

[7.1 Постановка задачи 10](#_Toc162981287)

[7.2 Решение на Java 10](#_Toc162981288)

[7.3 Проверка результатов 11](#_Toc162981289)

[Задание 8 11](#_Toc162981290)

[8.1 Постановка задачи 11](#_Toc162981291)

[8.2 Решение на java 11](#_Toc162981292)

[8.3 Проверка решения 12](#_Toc162981293)

[Задание 9 12](#_Toc162981294)

[9.1 Постановка задачи 12](#_Toc162981295)

[9.2 Решение на java 12](#_Toc162981296)

[9.3 Проверка решения 14](#_Toc162981297)

[Задание 10 14](#_Toc162981298)

[10.1 Постановка задачи 14](#_Toc162981299)

[10.2 Решение на java 14](#_Toc162981300)

[10.3 Проверка решения 16](#_Toc162981301)

[Задание 11 16](#_Toc162981302)

[11.1 Постановка задачи 16](#_Toc162981303)

[11.2 Решение на java 17](#_Toc162981304)

[11.3 Проверка решения 19](#_Toc162981305)

[Задание 12 19](#_Toc162981306)

[12.1 Постановка задачи 19](#_Toc162981307)

[12.2 Решение на Java 19](#_Toc162981308)

[12.3 Проверка результатов 20](#_Toc162981309)

# Задание 1

## 1.1 Постановка задачи

Ввести массив, состоящий из N элементов целого типа. Массив является элементами целого числа в двоичной системе счисления. Проверить правильность введения элементов двоичного числа, т.е. в массиве должны быть только 0 и 1. Если число введено верно, то перевести его в семеричную систему счисления и поэлементно записать число в массив.

## 1.2 Решение на java

import java.util.ArrayList;  
import java.util.Scanner;  
  
public class Task1 {  
 public static void main(String[] args) {  
 var scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter N: ");  
 var n = scanner.nextInt();  
 var inputArray = new int[n];  
 for (int i = 0; i < n; i++) {  
 var bit = 0;  
 while (true) {  
 System.*out*.print("[bit" + i + "] Enter value (0/1): ");  
 bit = scanner.nextInt();  
 if (bit == 0 || bit == 1) {  
 break;  
 }  
 System.*out*.println("Bad input! Only 0 and 1 allowed.");  
 }  
 inputArray[i] = bit;  
 }  
 *printArray*(*binaryToSeptenary*(inputArray));  
 }  
 public static int[] binaryToSeptenary(int[] binaryArray) {  
 int number = *convertBinaryArrayToDecimal*(binaryArray);  
 return *convertDecimalToSeptenaryArray*(number);  
 }  
 public static int convertBinaryArrayToDecimal(int[] binaryArray) {  
 int decimalNumber = 0;  
 int n = binaryArray.length;  
  
 for (int i = 0; i < n; i++) {  
 decimalNumber += binaryArray[i] \* Math.*pow*(2, n - 1 - i);  
 }  
  
 return decimalNumber;  
 }  
 public static int[] convertDecimalToSeptenaryArray(int decimalNumber) {  
 ArrayList<Integer> septenaryList = new ArrayList<>();  
  
 if (decimalNumber == 0) {  
 septenaryList.add(0);  
 } else {  
 while (decimalNumber > 0) {  
 septenaryList.add(decimalNumber % 7);  
 decimalNumber /= 7;  
 }  
 }  
 int[] septenaryArray = new int[septenaryList.size()];  
 for (int i = 0; i < septenaryList.size(); i++) {  
 septenaryArray[i] = septenaryList.get(septenaryList.size() - 1 - i);  
 }  
  
 return septenaryArray;  
 }  
 public static void printArray(int[] array) {  
 for (int value : array) {  
 System.*out*.print(value + " ");  
 }  
 System.*out*.println();  
 }  
}

## 1.3 Проверка результатов

|  |  |
| --- | --- |
| Java | Ответ |
|  |  |
|  |  |

# Задание 2

## 2.1 Постановка задачи

Задан целочисленный массив. Определить количество участков массива, на котором элементы монотонно возрастают (каждое следующее число больше предыдущего), вывести их на экран в виде No участка – участок.

## 2.2 Решение на Java

public class Task2 {  
 public static void main(String[] args) {  
 var array = new int[] { 1, 2, 3, 4, 0, 1, 2, 7, 3, 2, 1 };  
  
 var areaIndex = 0;  
 var areaStart = 0;  
 var area = "";  
 for (int i = 1; i < array.length; i++) {  
 if (array[i-1] >= array[i]) {  
 if (Math.*abs*(areaStart - i) > 1) {  
 System.*out*.print("[" + areaIndex + "] ");  
 for (int x = areaStart; x < i; x++) {  
 System.*out*.print(array[x] + " ");  
 }  
 System.*out*.println();  
 }  
 areaIndex += 1;  
 areaStart = i;  
 }  
 }  
 }  
}

## 2.3 Проверка результатов

|  |  |
| --- | --- |
| Исходные данные | Java |
| 1, 2, 3, 4, 0, 1, 2, 7, 3, 2, 1 |  |

# 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 координаты центра и радиус окружности 1 и координаты центра и радиус окружности 2. Кольцевая область определена на координатной плоскости. Определить номера точек, которые лежат на границах кольца.

## 3.2 Решение на java

import java.util.Scanner;  
  
public class Task3 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Введите количество точек (N): ");  
 int N = scanner.nextInt();  
  
 double[] coordinates = new double[2 \* N];  
  
 for (int i = 0; i < 2 \* N; i += 2) {  
 System.*out*.print("["+i+"] Введите координату X точки: ");  
 coordinates[i] = scanner.nextDouble();  
 System.*out*.print("["+(i)+"] Введите координату Y точки: ");  
 coordinates[i + 1] = scanner.nextDouble();  
 }  
  
 System.*out*.println("Введите координаты и радиус внутренней окружности (Xc1, Yc1, R1): ");  
 double Xc1 = scanner.nextDouble();  
 double Yc1 = scanner.nextDouble();  
 double R1 = scanner.nextDouble();  
  
 System.*out*.println("Введите координаты и радиус внешней окружности (Xc2, Yc2, R2): ");  
 double Xc2 = scanner.nextDouble();  
 double Yc2 = scanner.nextDouble();  
 double R2 = scanner.nextDouble();  
  
 System.*out*.println("Точки на границе кольца:");  
  
 for (int i = 0; i < N; i++) {  
 double x = coordinates[2 \* i];  
 double y = coordinates[2 \* i + 1];  
  
 double distanceInnerToCenter = Math.*sqrt*(Math.*pow*(x - Xc1, 2) + Math.*pow*(y - Yc1, 2));  
 double distanceOuterToCenter = Math.*sqrt*(Math.*pow*(x - Xc2, 2) + Math.*pow*(y - Yc2, 2));  
  
 if (distanceInnerToCenter >= R1 && distanceOuterToCenter <= R2) {  
 System.*out*.println("Точка " + (i / 2 + 1) + ": (" + x + ", " + y + ")");  
 }  
 }  
 }  
}

## 3.3 Проверка результатов

|  |  |
| --- | --- |
| Java | Проверка |
|  |  |

# Задание 4

## 4.1 Постановка задачи

Задать массив из произвольных чисел и реализовать алгоритм «шейкер-сортировки». Выполнить сравнение «шейкер-сортировки» с сортировкой «пузырек», рассмотрев количество итераций, а так же промежуточные результаты сортировки.

## 4.2 Решение на Java

public class Task4 {  
 public static void main(String[] args) {  
 int[] array = {5, 2, 9, 1, 5, 6, 10, 12, 1, 2, 9, 7, 4, 3, 6, 4, 7, 6, 1, 3, 4, 9, 7, 4, 1, 3, 2, 6, 8, 9, 0};  
  
 *shakerSort*(array.clone());  
 *bubbleSort*(array.clone());  
 }  
  
 private static void shakerSort(int[] array) {  
 int count = 0;  
  
 int left = 0;  
 int right = array.length - 1;  
  
 do {  
 for (int i = left; i < right; i++) {  
 if (array[i] > array[i + 1]) {  
 *swap*(array, i, i + 1);  
 }  
 count++;  
 }  
 right--;  
  
 for (int i = right; i > left; i--) {  
 if (array[i - 1] > array[i]) {  
 *swap*(array, i - 1, i);  
 }  
 count++;  
 }  
 left++;  
 } while (left <= right);  
  
 System.*out*.println("[SnakerSort] count:"+count);  
 }  
 private static void bubbleSort(int[] array) {  
 int count = 0;  
  
 int n = array.length;  
 boolean swapped;  
  
 for (int i = 0; i < n - 1; i++) {  
 swapped = false;  
  
 for (int j = 0; j < n - i - 1; j++) {  
 if (array[j] > array[j + 1]) {  
 *swap*(array, j, j + 1);  
 swapped = true;  
 }  
 count++;  
 }  
  
 if (!swapped) {  
 break;  
 }  
 }  
 System.*out*.println("[BubbleSort] count:"+count);  
 }  
  
 private static void swap(int[] array, int i, int j) {  
 int temp = array[i];  
 array[i] = array[j];  
 array[j] = temp;  
 }  
  
 private static void printArray(int[] array) {  
 for (int num : array) {  
 System.*out*.print(num + " ");  
 }  
 System.*out*.println();  
 }  
}

## 4.3 Проверка результатов

Подсчитаем количество итераций.



# Задание 5

## 5.1 Постановка задачи

Дан целочисленный двумерный массив В [n; n]. Вычислить произведение элементов этого массива, расположенных ниже главной диагонали.

## 5.2 Решение на java

public class Task5 {  
 public static void main(String[] args) {  
 int n = 4;  
 int[][] B = {  
 {1, 2, 3, 4},  
 {5, 6, 7, 8},  
 {9, 10, 11, 12},  
 {13, 14, 15, 16}  
 };  
 int product = 1;  
 for (int i = 1; i < n; i++) {  
 for (int j = 0; j < i; j++) {  
 product \*= B[i][j];  
 }  
 }  
 System.*out*.println("Result: " + product);  
 }  
}

## 5.3 Проверка результатов

|  |  |  |
| --- | --- | --- |
| Исходные данные | Java | Расчёты |
|  |  | 1228500 |

# Задание 6

## 6.1 Постановка задачи

Дан двумерный массив данных произвольного размера. Найти строку с максимальным элементом и отсортировать строку по возрастанию.

## 6.2 Решение на java

import java.util.Arrays;  
  
public class Task6 {  
 public static void main(String[] args) {  
 int[][] matrix = {  
 {5, 12, 8},  
 {3, 15, 7},  
 {9, 6, 11}  
 };  
 System.*out*.println("Было: ");  
 *printMatrix*(matrix);  
  
 int maxRow = 0;  
 int maxElement = matrix[0][0];  
  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 if (matrix[i][j] > maxElement) {  
 maxElement = matrix[i][j];  
 maxRow = i;  
 }  
 }  
 }  
  
 Arrays.*sort*(matrix[maxRow]);  
  
 System.*out*.println("Стало: ");  
 *printMatrix*(matrix);  
 }  
  
 private static void printMatrix(int[][] matrix) {  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 System.*out*.print(matrix[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 }  
}

## 6.3 Проверка результатов

|  |  |
| --- | --- |
| java | Ожидание |
|  | Вторая строчка должна быть отсортирована |

# Задание 7

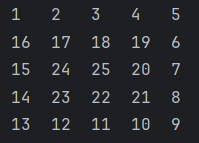
## 7.1 Постановка задачи

Заполнить двумерный массив n×n по образцу.

## 7.2 Решение на Java

public class Task7 {  
 public static void main(String[] args) {  
 int n = 5;  
 int[][] matrix = new int[n][n];  
  
 int num = 1;  
 int startRow = 0;  
 int endRow = n - 1;  
 int startCol = 0;  
 int endCol = n - 1;  
  
 while (num <= n \* n) {  
 for (int i = startCol; i <= endCol; i++) {  
 matrix[startRow][i] = num++;  
 }  
 for (int i = startRow + 1; i <= endRow; i++) {  
 matrix[i][endCol] = num++;  
 }  
 for (int i = endCol - 1; i >= startCol; i--) {  
 matrix[endRow][i] = num++;  
 }  
 for (int i = endRow - 1; i > startRow; i--) {  
 matrix[i][startCol] = num++;  
 }  
  
 startRow++;  
 endRow--;  
 startCol++;  
 endCol--;  
 }  
  
 *printMatrix*(n, matrix);  
 }  
  
 private static void printMatrix(int n, int[][] matrix) {  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 System.*out*.print(matrix[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 }  
}

## 7.3 Проверка результатов



# Задание 8

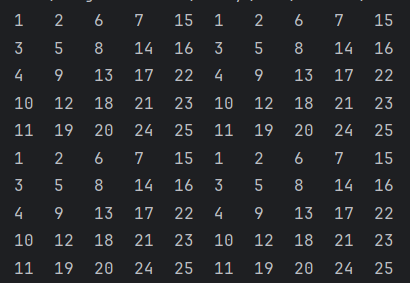
## 8.1 Постановка задачи

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.

## 8.2 Решение на java

public class Task8 {  
 public static void main(String[] args) {  
 int n = 10;  
 int[][] matrix = new int[n][n];  
  
 *magic*(n/2, matrix, 0, 0);  
 *magic*(n/2, matrix, n/2, 0);  
 *magic*(n/2, matrix, 0, n/2);  
 *magic*(n/2, matrix, n/2, n/2);  
  
 *printMatrix*(matrix);  
 }  
  
 private static void magic(int n, int[][] matrix, int startX, int startY) {  
 int index = 0;  
 int y = 1;  
 int x = -1;  
  
 while (index < n\*n) {  
 while ((y > 0) && (x < n - 1)) {  
 y -= 1;  
 x += 1;  
 index += 1;  
 matrix[startY + y][startX + x] = index;  
 }  
 if ((y == 0) && (x < n - 1)) {  
 x += 1;  
 } else {  
 y += 1;  
 }  
 index += 1;  
 matrix[startY + y][startX + x] = index;  
  
 while ((y < n - 1) && (x > 0)) {  
 y += 1;  
 x -= 1;  
 index += 1;  
 matrix[startY + y][startX + x] = index;  
 }  
 if ((x == 0) && (y < n - 1)) {  
 y += 1;  
 } else {  
 x += 1;  
 }  
 index += 1;  
 matrix[startY + y][startX + x] = index;  
 }  
 }  
 private static void printMatrix(int[][] matrix) {  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 System.*out*.print(matrix[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 }  
}

## 8.3 Проверка решения



# Задание 9

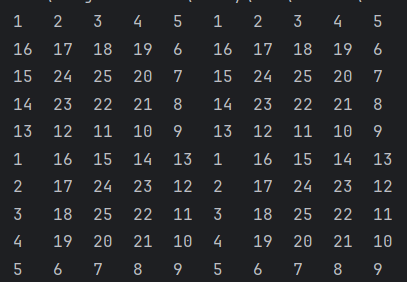
## 9.1 Постановка задачи

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.

## 9.2 Решение на java

public class Task9 {  
 public static void main(String[] args) {  
 int n = 5;  
 int[][] matrix = new int[n\*2][n\*2];  
  
 *magic1*(n, matrix, 1, 0, n-1, 0, n-1);  
 *magic2*(n, matrix, 1, n, 2\*n-1, 0, n-1);  
 *magic1*(n, matrix, 1, 0, n-1, n, 2\*n-1);  
 *magic2*(n, matrix, 1, n, 2\*n-1, n, 2\*n-1);  
  
 *printMatrix*(matrix);  
 }  
  
 private static void magic1(int n, int[][] matrix, int startNum, int startRow, int endRow, int startCol, int endCol) {  
 int num = 1;  
  
 var x = (endRow - startRow + 1) \* (endCol - startCol + 1);  
 while (num <= x) {  
 for (int i = startCol; i <= endCol; i++) {  
 matrix[startRow][i] = num++ + startNum - 1;  
 }  
 for (int i = startRow + 1; i <= endRow; i++) {  
 matrix[i][endCol] = num++ + startNum - 1;  
 }  
 for (int i = endCol - 1; i >= startCol; i--) {  
 matrix[endRow][i] = num++ + startNum - 1;  
 }  
 for (int i = endRow - 1; i > startRow; i--) {  
 matrix[i][startCol] = num++ + startNum - 1;  
 }  
  
 startRow++;  
 endRow--;  
 startCol++;  
 endCol--;  
 }  
 }  
 private static void magic2(int n, int[][] matrix, int startNum, int startRow, int endRow, int startCol, int endCol) {  
 int num = 1;  
  
 var x = (endRow - startRow + 1) \* (endCol - startCol + 1);  
 while (num <= n \* n) {  
 for (int i = startRow; i <= endRow; i++) {  
 matrix[i][startCol] = num++ + startNum - 1;  
 }  
 for (int i = startCol + 1; i <= endCol; i++) {  
 matrix[endRow][i] = num++ + startNum - 1;  
 }  
 for (int i = endRow - 1; i >= startRow; i--) {  
 matrix[i][endCol] = num++ + startNum - 1;  
 }  
 for (int i = endCol - 1; i > startCol; i--) {  
 matrix[startRow][i] = num++ + startNum - 1;  
 }  
  
 startRow++;  
 endRow--;  
 startCol++;  
 endCol--;  
 }  
 }  
 private static void printMatrix(int[][] matrix) {  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 System.*out*.print(matrix[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 }  
}

## 9.3 Проверка решения



# 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 лабораторнойработы No 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 на java

public class Task10 {  
 public static void main(String[] args) {  
 var n = 3;  
 var points = new int[2][n];  
  
 points[0][0] = 0; // x  
 points[1][0] = 0; // y  
  
 points[0][1] = 3; // x  
 points[1][1] = 0; // y  
  
 points[0][2] = -1; // x  
 points[1][2] = 2; // y  
  
 *printMatrix*(points);  
  
 *CheckPoints*(n, points);  
 }  
  
 private static void CheckPoints(int n, int[][] points) {  
 for (int index = 0; index < n; index++) {  
 var pointX = points[0][index];  
 var pointY = points[1][index];  
  
 var result = *checkArea*(pointX, pointY);  
 String textedResult = "";  
 switch (result) {  
 case 1:  
 textedResult = "InArea";  
 break;  
 case 0:  
 textedResult = "OnBorder";  
 break;  
 case -1:  
 textedResult = "NotInArea";  
 break;  
 }  
  
 System.*out*.println("{"+pointX+";"+pointY+"}\tis\t" + textedResult);  
 }  
 }  
  
 private static int checkArea(int x, int y) {  
 if (x <= -1) {  
 var result = *inCircle*(-1, 3, 2, x, y);  
 if (result != -1) {  
 return result;  
 }  
 }  
 var area1 = *inArea1*(x, y);  
 if (area1 != -1) {  
 return area1;  
 }  
 if ((x <= 2 && x >= -2) && (y <= 4 && y >= 2)) { // cube  
 return 1;  
 }  
 if (y <= 0) {  
 var result = *inCircle*(5, 0, 2, x, y);  
 if (result != -1) {  
 return result;  
 }  
 }  
 var area2 = *inArea2*(x, y);  
 if (area2!= -1) {  
 return area2;  
 }  
 if ((x <= 6 && x >= 4) && (y <= 0 && y >= -1)) { // cube 2  
 return 1;  
 }  
  
 return -1;  
 }  
 private static void printMatrix(int[][] matrix) {  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 System.*out*.print(matrix[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 }  
  
 public static int inCircle(double circleX, double circleY, double r, double pointX, double pointY) {  
 var dist = Math.*sqrt*(Math.*pow*(circleX - pointX, 2) + Math.*pow*(circleY - pointY, 2));  
 if (Math.*abs*(dist - r) < 0.01f) {  
 return 0;  
 }  
 if (dist < r) {  
 return 1;  
 }  
 return -1;  
 }  
 public static int inArea1(double pointX, double pointY) {  
 var inArea = ( (pointY >= (1/3d \* pointX + 4/3d)) && (pointX >= -1 && pointX <= 2) && (pointY >= 1 && pointY <= 2) )  
 || ( (pointY >= (pointX \* 2 - 2)) && (pointX <= 3 && pointX >= 2) && (pointY <= 4 && pointY >= 2) )  
 || ( (pointY <= (pointX \* -0.25 + 4.75)) && (pointX >= -1 && pointX <= 3) && (pointY >= 4 && pointY <= 5) );  
 if (inArea == false) {  
 return -1;  
 }  
 var onLine = ((1/3d \* pointX + 4/3d) == pointY) || ((pointX \* 2 - 2) == pointY) || ((pointX \* -0.25 + 4.75) == pointY);  
 if (onLine) {  
 return 0;  
 } else {  
 return 1;  
 }  
 }  
 public static int inArea2(double pointX, double pointY) {  
 var inArea = (pointY <= -0.2d \* pointX + 1.4d) // big line  
 && (pointY >= 0) // down border  
 && (pointX >= 2 && pointX <= 7) // in x area  
 && (pointY >= (-1 \* pointX + 3)) // small line  
 && (pointY <= 1); // up border  
 if (inArea == false) {  
 return -1;  
 }  
 var onLine = ((-0.2d \* pointX + 1.4d) == pointY) || ((-1 \* pointX + 3) == pointY);  
 if (onLine) {  
 return 0;  
 } else {  
 return 1;  
 }  
 }  
}

## 10.3 Проверка решения

|  |
| --- |
| Java |
|  |

# Задание 11

## 11.1 Постановка задачи

Дан двумерный массив 2×3N, где N – количество предполагаемых треугольников. Пользователь вводит данные по координатам вершин треугольников в двумерный массив. Создать двумерный массив ответов для N предполагаемых треугольников: 0-я строчка результаты проверки о существовании треугольника с введенными вершинами (1 –существует; 0 – не существует); 1-я строчка классификация треугольников (1 – равносторонний; 2 – прямоугольный; 3 – равнобедренный; 4 – произвольный; 0 – если такого треугольника не существует); 2-я строчка площадь треугольника, если он существует, иначе 0. Создать метод для стилизованного вывода ответов. Проверить есть ли треугольники одинаковой площади, вывести ответ, если такие треугольники есть, то указать их номера.

## 11.2 Решение на java

import java.util.Scanner;  
  
public class Task11 {  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Введите количество треугольников (N): ");  
 int N = scanner.nextInt();  
  
 int[][] coordinates = new int[2][3 \* N];  
  
 System.*out*.println("Введите координаты вершин треугольников:");  
 for (int i = 0; i < 3 \* N; i += 3) {  
 System.*out*.println("Треугольник " + ((i / 3) + 1) + ":");  
 for (int j = 0; j < 3; j++) {  
 System.*out*.print("Введите координату x для вершины " + (j + 1) + ": ");  
 coordinates[0][i + j] = scanner.nextInt();  
 System.*out*.print("Введите координату y для вершины " + (j + 1) + ": ");  
 coordinates[1][i + j] = scanner.nextInt();  
 }  
 }  
  
 int[][] results = *analyzeTriangles*(coordinates);  
  
 *displayResults*(results);  
  
 *checkSameAreaTriangles*(results);  
 }  
  
 public static int[][] analyzeTriangles(int[][] coordinates) {  
 int N = coordinates[0].length / 3;  
 int[][] results = new int[3][N];  
  
 for (int i = 0; i < N; i++) {  
 int[][] triangle = {{coordinates[0][i \* 3], coordinates[0][i \* 3 + 1], coordinates[0][i \* 3 + 2]},  
 {coordinates[1][i \* 3], coordinates[1][i \* 3 + 1], coordinates[1][i \* 3 + 2]}};  
 results[0][i] = *isTriangleExist*(triangle);  
 if (results[0][i] == 1) {  
 results[1][i] = *classifyTriangle*(triangle);  
 results[2][i] = *calculateArea*(triangle);  
 }  
 }  
  
 return results;  
 }  
  
 public static int isTriangleExist(int[][] triangle) {  
 int[] sideLengths = new int[3];  
 sideLengths[0] = *calculateDistance*(triangle[0][0], triangle[1][0], triangle[0][1], triangle[1][1]);  
 sideLengths[1] = *calculateDistance*(triangle[0][1], triangle[1][1], triangle[0][2], triangle[1][2]);  
 sideLengths[2] = *calculateDistance*(triangle[0][2], triangle[1][2], triangle[0][0], triangle[1][0]);  
  
 if (sideLengths[0] + sideLengths[1] > sideLengths[2] &&  
 sideLengths[1] + sideLengths[2] > sideLengths[0] &&  
 sideLengths[2] + sideLengths[0] > sideLengths[1]) {  
 return 1;  
 } else {  
 return 0;  
 }  
 }  
  
 public static int calculateDistance(int x1, int y1, int x2, int y2) {  
 return (int) Math.*sqrt*(Math.*pow*(x2 - x1, 2) + Math.*pow*(y2 - y1, 2));  
 }  
  
 public static int classifyTriangle(int[][] triangle) {  
 int[] sideLengths = new int[3];  
 sideLengths[0] = *calculateDistance*(triangle[0][0], triangle[1][0], triangle[0][1], triangle[1][1]);  
 sideLengths[1] = *calculateDistance*(triangle[0][1], triangle[1][1], triangle[0][2], triangle[1][2]);  
 sideLengths[2] = *calculateDistance*(triangle[0][2], triangle[1][2], triangle[0][0], triangle[1][0]);  
  
 if (sideLengths[0] == sideLengths[1] && sideLengths[1] == sideLengths[2]) {  
 return 1; // Равносторонний  
 } else if (sideLengths[0] \* sideLengths[0] + sideLengths[1] \* sideLengths[1] == sideLengths[2] \* sideLengths[2] ||  
 sideLengths[1] \* sideLengths[1] + sideLengths[2] \* sideLengths[2] == sideLengths[0] \* sideLengths[0] ||  
 sideLengths[2] \* sideLengths[2] + sideLengths[0] \* sideLengths[0] == sideLengths[1] \* sideLengths[1]) {  
 return 2; // Прямоугольный  
 } else if (sideLengths[0] == sideLengths[1] || sideLengths[1] == sideLengths[2] || sideLengths[2] == sideLengths[0]) {  
 return 3; // Равнобедренный  
 } else {  
 return 4; // Произвольный  
 }  
 }  
  
 public static int calculateArea(int[][] triangle) {  
 int a = *calculateDistance*(triangle[0][0], triangle[1][0], triangle[0][1], triangle[1][1]);  
 int b = *calculateDistance*(triangle[0][1], triangle[1][1], triangle[0][2], triangle[1][2]);  
 int c = *calculateDistance*(triangle[0][2], triangle[1][2], triangle[0][0], triangle[1][0]);  
  
 double p = (a + b + c) / 2.0;  
 return (int) Math.*sqrt*(p \* (p - a) \* (p - b) \* (p - c));  
 }  
  
 public static void displayResults(int[][] results) {  
 System.*out*.println("Результаты анализа треугольников:");  
 for (int i = 0; i < results[0].length; i++) {  
 System.*out*.println("Треугольник " + (i + 1) + ":");  
 System.*out*.println("Существует: " + (results[0][i] == 1 ? "Да" : "Нет"));  
 if (results[0][i] == 1) {  
 System.*out*.println("Классификация: " + *getTriangleType*(results[1][i]));  
 System.*out*.println("Площадь: " + results[2][i]);  
 }  
 }  
 }  
  
 public static String getTriangleType(int type) {  
 switch (type) {  
 case 1:  
 return "Равносторонний";  
 case 2:  
 return "Прямоугольный";  
 case 3:  
 return "Равнобедрененный";  
 case 4:  
 return "Произвольный";  
 default:  
 return "Неизвестный";  
 }  
 }  
  
 public static void checkSameAreaTriangles(int[][] results) {  
 System.*out*.println("\nПроверка на треугольники с одинаковой площадью:");  
 for (int i = 0; i < results[0].length; i++) {  
 for (int j = i + 1; j < results[0].length; j++) {  
 if (results[0][i] == 1 && results[0][j] == 1 && results[2][i] == results[2][j]) {  
 System.*out*.println("Треугольники " + (i + 1) + " и " + (j + 1) + " имеют одинаковую площадь.");  
 }  
 }  
 }  
 }  
}

## 11.3 Проверка решения

# Задание 12

## 12.1 Постановка задачи

Дан двумерный квадратный массив A и двумерный квадратный массив обратный к первому A^-1. При этом пользователь вводит размерность массива и данные сам и может допустить ошибку при вычислении обратной матрицы или при вводе. Поэтому выполнить проверку соответствуют ли массивы свойству A\* A^-1=E, где E – это единичная матрица.

## 12.2 Решение на Java

import java.util.Scanner;  
  
public class Task12 {  
 // 1 2 -1 1  
 // 2 2 1 -0,5  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Введите размерность матрицы:");  
 int dimension = scanner.nextInt();  
  
 System.*out*.println("Введите элементы матрицы A:");  
 double[][] matrixA = *inputMatrix*(dimension, scanner);  
  
 System.*out*.println("Введите элементы матрицы A^-1:");  
 double[][] matrixAInverse = *inputMatrix*(dimension, scanner);  
  
 double[][] result = *multiplyMatrices*(matrixA, matrixAInverse);  
 boolean isValid = *isIdentityMatrix*(result);  
  
 if (isValid) {  
 System.*out*.println("Матрицы соответствуют свойству A\*(A^-1) = E");  
 } else {  
 System.*out*.println("Матрицы не соответствуют свойству A\*(A^-1) = E");  
 }  
 }  
  
 public static double[][] inputMatrix(int dimension, Scanner scanner) {  
 double[][] matrix = new double[dimension][dimension];  
 for (int i = 0; i < dimension; i++) {  
 for (int j = 0; j < dimension; j++) {  
 matrix[i][j] = scanner.nextDouble();  
 }  
 }  
 return matrix;  
 }  
  
 public static double[][] multiplyMatrices(double[][] matrixA, double[][] matrixB) {  
 int m = matrixA.length;  
 int n = matrixB[0].length;  
 int p = matrixB.length;  
 double[][] result = new double[m][n];  
  
 for (int i = 0; i < m; i++) {  
 for (int j = 0; j < n; j++) {  
 for (int k = 0; k < p; k++) {  
 result[i][j] += matrixA[i][k] \* matrixB[k][j];  
 }  
 }  
 }  
  
 return result;  
 }  
  
 public static boolean isIdentityMatrix(double[][] matrix) {  
 int n = matrix.length;  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 if ((i == j && matrix[i][j] != 1) || (i != j && matrix[i][j] != 0)) {  
 return false;  
 }  
 }  
 }  
 return true;  
 }  
}

## 12.3 Проверка результатов

Воспользуемся Excel и формулой =МУМНОЖ() для того что бы умножить две матрицы. Если получившаяся матрица является единичной, то условие из задания выполнено.

|  |  |
| --- | --- |
| java | Проверка |
|  |  |