Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе № 9

тема «Массивы»

по дисциплине «Информатика»

Выполнил: студент группу ИСТ-22-1Б Пеяс В.С.

Проверил: Нетбай Георгий Владимирович

Пермь, 2023

**Содержание**

[Задание 1 4](#_Toc136047746)

[1.1. Постановка задачи 4](#_Toc136047747)

[1.2. Решение задачи, код программы 4](#_Toc136047748)

[1.3. Тестирование работы программы 4](#_Toc136047749)

[Задание 2 5](#_Toc136047750)

[2.1. Постановка задачи 5](#_Toc136047751)

[2.2. Решение задачи, код программы 5](#_Toc136047752)

[2.3. Тестирование работы программы 5](#_Toc136047753)

[Задание 3 6](#_Toc136047754)

[3.1. Постановка задачи 6](#_Toc136047755)

[3.2. Решение задачи, код программы 6](#_Toc136047756)

[3.3. Тестирование работы программы 6](#_Toc136047757)

[Задание 4 6](#_Toc136047758)

[4.1. Постановка задачи 6](#_Toc136047759)

[4.2. Решение задачи, код программы 6](#_Toc136047760)

[4.3. Тестирование работы программы 6](#_Toc136047761)

[Задание 5 6](#_Toc136047762)

[5.1. Постановка задачи 6](#_Toc136047763)

[5.2. Решение задачи, код программы 6](#_Toc136047764)

[5.3. Тестирование работы программы 6](#_Toc136047765)

[Задание 6 6](#_Toc136047766)

[6.1. Постановка задачи 6](#_Toc136047767)

[6.2. Решение задачи, код программы 6](#_Toc136047768)

[6.3. Тестирование работы программы 6](#_Toc136047769)

[Задание 7 7](#_Toc136047770)

[7.1. Постановка задачи 7](#_Toc136047771)

[7.2. Решение задачи, код программы 7](#_Toc136047772)

[7.3. Тестирование работы программы 7](#_Toc136047773)

[Задание 8 7](#_Toc136047774)

[8.1. Постановка задачи 7](#_Toc136047775)

[8.2. Решение задачи, код программы 7](#_Toc136047776)

[8.3. Тестирование работы программы 7](#_Toc136047777)

[Задание 9 7](#_Toc136047778)

[9.1. Постановка задачи 7](#_Toc136047779)

[9.2. Решение задачи, код программы 7](#_Toc136047780)

[9.3. Тестирование работы программы 7](#_Toc136047781)

[Задание 10 7](#_Toc136047782)

[10.1. Постановка задачи 7](#_Toc136047783)

[10.2. Решение задачи, код программы 7](#_Toc136047784)

[10.3. Тестирование работы программы 8](#_Toc136047785)

[Задание 11 8](#_Toc136047786)

[11.1. Постановка задачи 8](#_Toc136047787)

[11.2. Решение задачи, код программы 8](#_Toc136047788)

[11.3. Тестирование работы программы 8](#_Toc136047789)

[Задание 12 8](#_Toc136047790)

[12.1. Постановка задачи 8](#_Toc136047791)

[12.2. Решение задачи, код программы 8](#_Toc136047792)

[12.3. Тестирование работы программы 8](#_Toc136047793)

# Задание 1

## Постановка задачи

Ввести массив, состоящий из N элементов целого типа. Массив является элементами

целого числа в девятеричной системе счисления. Проверить правильность введения элементов числа,

т.е. в массиве должны быть только 0, 1, 2, 3, 4, 5, 6, 7 и 8. Если число введено верно, то перевести его

в десятичную систему счисления и поэлементно записать число в массив.

## 1.2. Решение задачи, код программы

import java.util.Scanner;  
  
  
public class Main {  
  
 public static void main(String[] args) {  
  
 Scanner scanner = new Scanner(System.*in*);  
  
  
 System.*out*.print("Введите количество элементов в массиве: ");  
  
 int n = scanner.nextInt();  
  
  
 int[] octalArray = new int[n];  
  
  
 System.*out*.println("Введите элементы массива (целые числа от 0 до 8):");  
  
 for (int i = 0; i < n; i++) {  
  
 int element = scanner.nextInt();  
  
 if (element < 0 || element > 8) {  
  
 System.*out*.println("Некорректный элемент. Допустимы только числа от 0 до 8.");  
  
 return;  
  
 }  
  
 octalArray[i] = element;  
  
 }  
  
  
 int decimalNumber = *convertOctalToDecimal*(octalArray);  
  
 int[] decimalArray = *convertDecimalToArray*(decimalNumber);  
  
  
 System.*out*.println("В 10-ой:");  
  
 for (int element : decimalArray) {  
  
 System.*out*.println(element);  
  
 }  
  
 }  
  
  
 private static int convertOctalToDecimal(int[] octalArray) {  
  
 int decimalNumber = 0;  
  
 int power = 0;  
  
 for (int i = octalArray.length - 1; i >= 0; i--) {  
  
 decimalNumber += octalArray[i] \* Math.*pow*(9, power);  
  
 power++;  
  
 }  
  
 return decimalNumber;  
  
 }  
  
  
 private static int[] convertDecimalToArray(int decimalNumber) {  
  
 String decimalString = String.*valueOf*(decimalNumber);  
  
 int[] decimalArray = new int[decimalString.length()];  
  
 for (int i = 0; i < decimalString.length(); i++) {  
  
 decimalArray[i] = Integer.*parseInt*(String.*valueOf*(decimalString.charAt(i)));  
  
 }  
  
 return decimalArray;  
  
 }  
  
}

## 1.3. Тестирование работы программы

Таблица 1

Тестирование работы программы.

|  |  |
| --- | --- |
| № п.п. | Решение Java |
| 1 |  |

# 

# Задание 2

## 2.1. Постановка задачи

Пользователем введен одномерный массив данных, определить является ли он последовательностью Перрина. Последовательность Перрина – это целочисленная последовательность  с начальными значениями  и линейным рекуррентным соотношением для определения последующих чисел последовательности  для n > 2.

## 2.2. Решение задачи, код программы

import java.util.Scanner;  
  
  
public class \_2 {  
  
 public static void main(String[] args) {  
  
 Scanner scanner = new Scanner(System.*in*);  
  
  
 System.*out*.print("Введите элементы массива через пробел: ");  
  
 String input = scanner.nextLine();  
  
 String[] elements = input.split(" ");  
  
 int[] sequence = new int[elements.length];  
  
  
 for (int i = 0; i < elements.length; i++) {  
  
 sequence[i] = Integer.*parseInt*(elements[i]);  
  
 }  
  
  
 boolean isPerrinSequence = *checkPerrinSequence*(sequence);  
  
  
 if (isPerrinSequence) {  
  
 System.*out*.println("Введенный массив является последовательностью Перрина.");  
  
 } else {  
  
 System.*out*.println("Введенный массив не является последовательностью Перрина.");  
  
 }  
  
 }  
  
  
 private static boolean checkPerrinSequence(int[] sequence) {  
  
 if (sequence.length < 3) {  
  
 return false;  
  
 }  
  
  
 if (sequence[0] != 3 || sequence[1] != 0 || sequence[2] != 2) {  
  
 return false;  
  
 }  
  
  
 for (int i = 3; i < sequence.length; i++) {  
  
 int nextElement = sequence[i - 2] + sequence[i - 3];  
  
 if (sequence[i] != nextElement) {  
  
 return false;  
  
 }  
  
 }  
  
  
 return true;  
  
 }  
  
}

# 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. Определить номера точек, которые могут являться вершинами равнобедренного треугольника.

## 3.2. Решение задачи, код программы

import java.util.ArrayList;  
  
import java.util.List;  
  
import java.util.Scanner;  
  
  
public class \_3 {  
  
 public static void main(String[] args) {  
  
 Scanner scanner = new Scanner(System.*in*);  
  
  
 System.*out*.print("Введите количество точек (N): ");  
  
 int n = scanner.nextInt();  
  
  
 int[] coordinates = new int[2 \* n];  
  
 System.*out*.println("Введите координаты точек (x, y) через пробел:");  
  
  
 for (int i = 0; i < 2 \* n; i++) {  
  
 coordinates[i] = scanner.nextInt();  
  
 }  
  
  
 List<Integer> trianglePoints = *findIsoscelesTrianglePoints*(coordinates);  
  
  
 if (trianglePoints.isEmpty()) {  
  
 System.*out*.println("В заданном массиве нет точек, которые могут являться вершинами равнобедренного треугольника.");  
  
 } else {  
  
 System.*out*.println("Точки, которые могут являться вершинами равнобедренного треугольника: ");  
  
 for (int point : trianglePoints) {  
  
 System.*out*.println("Номер точки: " + point);  
  
 }  
  
 }  
  
 }  
  
  
 private static List<Integer> findIsoscelesTrianglePoints(int[] coordinates) {  
  
 List<Integer> trianglePoints = new ArrayList<>();  
  
  
 for (int i = 0; i < coordinates.length; i += 2) {  
  
 int x1 = coordinates[i];  
  
 int y1 = coordinates[i + 1];  
  
  
 for (int j = i + 2; j < coordinates.length; j += 2) {  
  
 int x2 = coordinates[j];  
  
 int y2 = coordinates[j + 1];  
  
  
 if (*isIsoscelesTriangle*(x1, y1, x2, y2)) {  
  
 trianglePoints.add(i / 2 + 1);  
  
 trianglePoints.add(j / 2 + 1);  
  
 }  
  
 }  
  
 }  
  
  
 return trianglePoints;  
  
 }  
  
  
 private static boolean isIsoscelesTriangle(int x1, int y1, int x2, int y2) {  
  
 int dx = Math.*abs*(x2 - x1);  
  
 int dy = Math.*abs*(y2 - y1);  
  
 return dx == dy;  
  
 }  
  
}

# Задание 4

## 4.1. Постановка задачи

Дан массив n действительных чисел. Требуется упорядочить его по возрастанию. Выполнить сортировку элементов массива с использованием «гномьей» сортировки: сравниваются два соседних элемента ai и аi+1. Если ai ≤ ai+1, то продвигаются на один элемент вперед. Если ai > ai+1, то производится перестановка и сдвигаются на один элемент назад. Выполнить сравнение «гномьей» сортировки с сортировкой «пузырек», рассмотрев количество итераций, а так же промежуточные результаты сортировки.

## 4.2. Решение задачи, код программы

import java.util.Arrays;

public class Main {

   public static void main(String[] args) {

       int[] array = {9, 5, 2, 7, 1, 8};

       System.*out*.println("Исходный массив: " + Arrays.*toString*(array));

       int[] gnomeSortedArray = *gnomeSort*(array.clone());

       int[] bubbleSortedArray = *bubbleSort*(array.clone());

       System.*out*.println("Гномья сортировка: " + Arrays.*toString*(gnomeSortedArray));

       System.*out*.println("Пузырьковая сортировка: " + Arrays.*toString*(bubbleSortedArray));

       System.*out*.println("Количество итераций для гномьей сортировки: " + *getGnomeSortIterationCount*(array.clone()));

       System.*out*.println("Количество итераций для пузырьковой сортировки: " + *getBubbleSortIterationCount*(array.clone()));

   }

   private static int[] gnomeSort(int[] array) {

       int i = 0;

       while (i < array.length) {

           if (i == 0 || array[i] >= array[i - 1]) {

               i++;

           } else {

*swap*(array, i, i - 1);

               i--;

           }

       }

       return array;

   }

   private static int[] bubbleSort(int[] array) {

       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;

               }

           }

           if (!swapped) {

               break;

           }

       }

       return array;

   }

   private static void swap(int[] array, int i, int j) {

       int temp = array[i];

       array[i] = array[j];

       array[j] = temp;

   }

   private static int getGnomeSortIterationCount(int[] array) {

       int iterations = 0;

       int i = 0;

       while (i < array.length) {

           iterations++;

           if (i == 0 || array[i] >= array[i - 1]) {

               i++;

           } else {

*swap*(array, i, i - 1);

               i--;

           }

       }

       return iterations;

   }

   private static int getBubbleSortIterationCount(int[] array) {

       int iterations = 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++) {

               iterations++;

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

*swap*(array, j, j + 1);

                   swapped = true;

               }

           }

           if (!swapped) {

               break;

           }

       }

       return iterations;

   }

}

# Задание 5

## 5.1. Постановка задачи

Дана квадратная матрица А порядка М. Повернуть ее на угол 180° (при этом элемент А1,1 поменяется местами с АM,M, элемент А1,2 – с АM,M-1 и т.д.). Вспомогательную матрицу не использовать.

## 5.2. Решение задачи, код программы

import java.util.Arrays;  
  
  
public class \_5 {  
  
 public static void main(String[] args) {  
  
 int[][] matrix = {  
  
 {1, 2, 3},  
  
 {4, 5, 6},  
  
 {7, 8, 9}  
  
 };  
  
  
 System.*out*.println("Исходная матрица:");  
  
 *printMatrix*(matrix);  
  
  
 *rotateMatrix180*(matrix);  
  
  
 System.*out*.println("Матрица, повернутая на 180°:");  
  
 *printMatrix*(matrix);  
  
 }  
  
  
 private static void rotateMatrix180(int[][] matrix) {  
  
 int n = matrix.length;  
  
  
 for (int i = 0; i < n / 2; i++) {  
  
 for (int j = 0; j < n; j++) {  
  
 int temp = matrix[i][j];  
  
 matrix[i][j] = matrix[n - 1 - i][n - 1 - j];  
  
 matrix[n - 1 - i][n - 1 - j] = temp;  
  
 }  
  
 }  
  
 }  
  
  
 private static void printMatrix(int[][] matrix) {  
  
 for (int[] row : matrix) {  
  
 System.*out*.println(Arrays.*toString*(row));  
  
 }  
  
 System.*out*.println();  
  
 }  
  
}

# Задание 6

## 6.1. Постановка задачи

Дана матрица размера М × N. Элемент матрицы называется ее локальным максимумом, если он больше всех окружающих его элементов. Элемент матрицы называется ее локальным минимумом, если он меньше всех окружающих его элементов. Найти индексы локальных максимумом и минимумов и записать их в два двумерных массива. Заменить все локальные максимумы данной матрицы на локальные минимумы по порядку их нахождения, если минимумом меньше, чем максимумов, то не достающие члены заменить на противоположные по знаку локальным минимумам в порядке их нахождения. При решении допускается использовать вспомогательную матрицу.

## 6.2. Решение задачи, код программы

import java.util.Arrays;  
  
  
public class \_6 {  
  
 public static void main(String[] args) {  
  
 int[][] matrix = {  
  
 {5, 3, 2, 4},  
  
 {7, 6, 8, 1},  
  
 {9, 5, 2, 7}  
  
 };  
  
  
 int[][] localMaxIndices = *findLocalMax*(matrix);  
  
 int[][] localMinIndices = *findLocalMin*(matrix);  
  
  
 int[][] modifiedMatrix = *replaceLocalMaxWithMin*(matrix, localMaxIndices, localMinIndices);  
  
  
 System.*out*.println("Исходная матрица:");  
  
 *printMatrix*(matrix);  
  
  
 System.*out*.println("Индексы локальных максимумов:");  
  
 *printMatrix*(localMaxIndices);  
  
  
 System.*out*.println("Индексы локальных минимумов:");  
  
 *printMatrix*(localMinIndices);  
  
  
 System.*out*.println("Модифицированная матрица:");  
  
 *printMatrix*(modifiedMatrix);  
  
 }  
  
  
 private static int[][] findLocalMax(int[][] matrix) {  
  
 int m = matrix.length;  
  
 int n = matrix[0].length;  
  
 int[][] localMaxIndices = new int[m][n];  
  
  
 for (int i = 0; i < m; i++) {  
  
 for (int j = 0; j < n; j++) {  
  
 if (*isLocalMax*(matrix, i, j)) {  
  
 localMaxIndices[i][j] = 1;  
  
 }  
  
 }  
  
 }  
  
  
 return localMaxIndices;  
  
 }  
  
  
 private static boolean isLocalMax(int[][] matrix, int row, int col) {  
  
 int m = matrix.length;  
  
 int n = matrix[0].length;  
  
 int current = matrix[row][col];  
  
  
 if (row > 0 && matrix[row - 1][col] >= current) {  
  
 return false;  
  
 }  
  
  
 if (row < m - 1 && matrix[row + 1][col] >= current) {  
  
 return false;  
  
 }  
  
  
 if (col > 0 && matrix[row][col - 1] >= current) {  
  
 return false;  
  
 }  
  
  
 return col >= n - 1 || matrix[row][col + 1] < current;  
  
 }  
  
  
 private static int[][] findLocalMin(int[][] matrix) {  
  
 int m = matrix.length;  
  
 int n = matrix[0].length;  
  
 int[][] localMinIndices = new int[m][n];  
  
  
 for (int i = 0; i < m; i++) {  
  
 for (int j = 0; j < n; j++) {  
  
 if (*isLocalMin*(matrix, i, j)) {  
  
 localMinIndices[i][j] = 1;  
  
 }  
  
 }  
  
 }  
  
  
 return localMinIndices;  
  
 }  
  
  
 private static boolean isLocalMin(int[][] matrix, int row, int col) {  
  
 int m = matrix.length;  
  
 int n = matrix[0].length;  
  
 int current = matrix[row][col];  
  
  
 if (row > 0 && matrix[row - 1][col] <= current) {  
  
 return false;  
  
 }  
  
  
 if (row < m - 1 && matrix[row + 1][col] <= current) {  
  
 return false;  
  
 }  
  
  
 if (col > 0 && matrix[row][col - 1] <= current) {  
  
 return false;  
  
 }  
  
  
 return col >= n - 1 || matrix[row][col + 1] > current;  
  
 }  
  
  
 private static int[][] replaceLocalMaxWithMin(int[][] matrix, int[][] localMaxIndices, int[][] localMinIndices) {  
  
 int m = matrix.length;  
  
 int n = matrix[0].length;  
  
 int[][] modifiedMatrix = new int[m][n];  
  
  
 int localMinCount = *countLocalMin*(localMinIndices);  
  
 int localMaxCount = *countLocalMax*(localMaxIndices);  
  
  
 int minIndex = 0;  
  
 int maxIndex = 0;  
  
  
 for (int i = 0; i < m; i++) {  
  
 for (int j = 0; j < n; j++) {  
  
 if (localMaxIndices[i][j] == 1) {  
  
 if (minIndex < localMinCount) {  
  
 modifiedMatrix[i][j] = *findNextLocalMin*(matrix, localMinIndices, minIndex);  
  
 minIndex++;  
  
 } else {  
  
 modifiedMatrix[i][j] = -*findNextLocalMin*(matrix, localMinIndices, maxIndex);  
  
 maxIndex++;  
  
 }  
  
 } else {  
  
 modifiedMatrix[i][j] = matrix[i][j];  
  
 }  
  
 }  
  
 }  
  
  
 return modifiedMatrix;  
  
 }  
  
  
 private static int countLocalMin(int[][] localMinIndices) {  
  
 int count = 0;  
  
  
 for (int[] row : localMinIndices) {  
  
 for (int value : row) {  
  
 if (value == 1) {  
  
 count++;  
  
 }  
  
 }  
  
 }  
  
  
 return count;  
  
 }  
  
  
 private static int countLocalMax(int[][] localMaxIndices) {  
  
 int count = 0;  
  
  
 for (int[] row : localMaxIndices) {  
  
 for (int value : row) {  
  
 if (value == 1) {  
  
 count++;  
  
 }  
  
 }  
  
 }  
  
  
 return count;  
  
 }  
  
  
 private static int findNextLocalMin(int[][] matrix, int[][] localMinIndices, int startIndex) {  
  
 int m = matrix.length;  
  
 int n = matrix[0].length;  
  
  
 int currentIndex = 0;  
  
  
 for (int i = 0; i < m; i++) {  
  
 for (int j = 0; j < n; j++) {  
  
 if (localMinIndices[i][j] == 1) {  
  
 if (currentIndex == startIndex) {  
  
 return matrix[i][j];  
  
 }  
  
 currentIndex++;  
  
 }  
  
 }  
  
 }  
  
  
 return 0;  
  
 }  
  
  
 private static void printMatrix(int[][] matrix) {  
  
 for (int[] row : matrix) {  
  
 System.*out*.println(Arrays.*toString*(row));  
  
 }  
  
 System.*out*.println();  
  
 }  
  
}

# Задание 7

## 7.1. Постановка задачи

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

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

## 7.2. Решение задачи, код программы

public class \_7 {  
  
 public static void main(String[] args) {  
  
 int n = 6; // Размерность массива  
  
  
 int[][] array = new int[n][n]; // Создаем двумерный массив  
  
  
 int num = 1; // Значение для заполнения массива  
  
  
 int startRow = n - 1; // Начальная строка  
  
 int endRow = 0; // Конечная строка  
  
 int startCol = 0; // Начальный столбец  
  
 int endCol = n - 1; // Конечный столбец  
  
  
 while (startRow >= endRow && startCol <= endCol) {  
  
 // Заполнение столбца сверху вниз  
  
 for (int i = startRow; i >= endRow; i--) {  
  
 array[i][startCol] = num;  
  
 num++;  
  
 }  
  
 startCol++; // Уменьшаем начальный столбец  
  
  
 // Заполнение строки слева направо  
  
 for (int i = startCol; i <= endCol; i++) {  
  
 array[endRow][i] = num;  
  
 num++;  
  
 }  
  
 endRow++; // Увеличиваем конечную строку  
  
  
 // Заполнение столбца снизу вверх  
  
 for (int i = endRow; i <= startRow; i++) {  
  
 array[i][endCol] = num;  
  
 num++;  
  
 }  
  
 endCol--; // Уменьшаем конечный столбец  
  
 }  
  
  
 // Выводим массив на экран  
  
 for (int i = 0; i < n; i++) {  
  
 for (int j = 0; j < n; j++) {  
  
 System.*out*.print(array[i][j] + " ");  
  
 }  
  
 System.*out*.println();  
  
 }  
  
 }  
  
}

## 7.3. Тестирование работы программы

Таблица 4

Тестирование работы программы и проверка результатов решения

|  |  |
| --- | --- |
| № п.п. | Решение Java |
| 1 |  |

# 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 задачи, код программы

public class \_8 {  
  
 public static void main(String[] args) {  
  
  
 int n = 6; // Размерность массива  
  
  
 int[][] array = new int[n][n]; // Создаем двумерный массив  
  
  
 // Заполняем матрицу  
  
 *fillMatrix*(array, n);  
  
  
 // Выводим массив на экран  
  
 for (int i = 0; i < n; i++) {  
  
 for (int j = 0; j < n; j++) {  
  
 System.*out*.print(array[i][j] + " ");  
  
 }  
  
 System.*out*.println();  
  
 }  
  
 }  
  
  
 // Метод для заполнения блока матрицы  
  
 public static int fillBlock(int[][] array, int startRow, int startY, int num, int end, int n, int startYY) {  
  
 for (int i = startRow - 1, j = startY; i >= 0; i--, j++) {  
  
 if (j < end)  
  
 array[i][j] = num++;  
  
 }  
  
 for (int i = startYY; i < n; i++) {  
  
 array[i][end - 1] = num++;  
  
 }  
  
 return num;  
  
 }  
  
  
 // Метод для заполнения всей матрицы  
  
 public static void fillMatrix(int[][] array, int n) {  
  
 int num = 1;  
  
 int startCol = 0;  
  
 int startRow = n;  
  
 int startYY = 1;  
  
 int end = n;  
  
 int numRows = n;  
  
 int numCols = n;  
  
  
 while (numRows > 0 && numCols > 0) {  
  
 num = *fillBlock*(array, startRow, startCol, num, end, n, startYY);  
  
 end--;  
  
 startYY += 2;  
  
 startCol++;  
  
 numRows--;  
  
 numCols--;  
  
 }  
  
 }  
  
}

## 8.3. Тестирование работы программы

Таблица 5

Тестирование работы программы и проверка результатов решения

|  |  |
| --- | --- |
| № п.п. | Решение Java |
| 1 |  |

# Задание 9

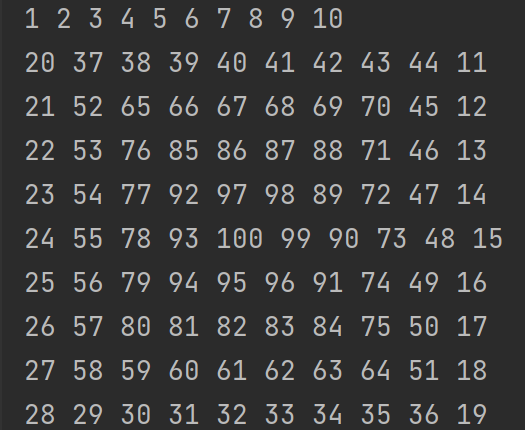
## 9.1. Постановка задачи

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 задачи, код программы

public class \_9 {  
  
 public static void main(String[] args) {  
  
  
 int n = 5;  
  
 int n2 = n \* 2; // Размерность массива  
  
  
 int[][] array = new int[n2][n2]; // Создаем двумерный массив  
  
  
 // Заполняем матрицу  
  
 *fillMatrix*(array, n2);  
  
  
 // Выводим массив на экран  
  
 for (int i = 0; i < n2; i++) {  
  
 for (int j = 0; j < n2; j++) {  
  
 System.*out*.print(array[i][j] + " ");  
  
 }  
  
 System.*out*.println();  
  
 }  
  
 }  
  
  
 // Метод для заполнения блока матрицы  
  
 public static int fillBlock(int[][] array, int num, int numRows, int numCols, int currentRow, int currentCol) {  
  
 /\*for (int j = 0; j < cols; j++) {  
  
 array[currentRow][j] = num++;  
  
 }  
  
 for (int i = 1; i < rowsCount; i++) {  
  
 array[i][col-1] = num++;  
  
 }  
  
 return num;\*/  
  
  
 // Заполнение первой строки  
  
 for (int i = currentRow; i < numCols; i++) {  
  
 array[currentRow][i] = num++;  
  
 }  
  
  
 // Заполнение последнего столбца  
  
 for (int i = currentRow + 1; i < numRows; i++) {  
  
 array[i][numCols - 1] = num++;  
  
 }  
  
  
 // Заполнение первого столбца  
  
 for (int i = currentRow + 1; i < numRows; i++) {  
  
 array[i][currentCol] = num++;  
  
 }  
  
  
 // Заполнение последней строки  
  
 for (int i = currentRow + 1; i < numCols - 1; i++) {  
  
 array[numRows - 1][i] = num++;  
  
 }  
  
 return num;  
  
 }  
  
  
 // Метод для заполнения всей матрицы  
  
 public static void fillMatrix(int[][] array, int n) {  
  
 int num = 1;  
  
 int numRows = n;  
  
 int numCols = n;  
  
 int currentRow = 0;  
  
 int currentCol = 0;  
  
  
 while (numRows > 0 && numCols > 0) {  
  
 num = *fillBlock*(array, num, numRows, numCols, currentRow, currentCol);  
  
 numRows--;  
  
 numCols--;  
  
 currentRow++;  
  
 currentCol++;  
  
 }  
  
 }  
  
}

## 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

import java.util.Random;  
import java.util.Scanner;  
  
public class \_10 {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("input n");  
 int n = in.nextInt();  
 int[][] arr = new int[2][n];  
 *getArr*(arr);  
 *PrintArr*(arr);  
 *PrintAns*(n,arr);  
 }  
 public static void getArr( int[][] arr){  
 Random r = new Random();  
 for (int i = 0; i < arr.length; i++) {  
 for (int j = 0; j < arr[i].length; j++) {  
 arr[i][j] = r.nextInt(-10, 10);  
 }  
 }  
 }  
 public static void PrintArr( int[][] arr){  
 for (int[] ints : arr) {  
 for (int anInt : ints) {  
 System.*out*.print(anInt + "\t");  
 }  
 System.*out*.println();  
 }  
 }  
 public static boolean f1(double x, double y){  
 return ((y > -6 \* x - 54) & (y > 0.67 \* x - 0.67) & (y > 3.67 \* x + 14.33)  
 & (y < 2 \* x + 18) & (y < 0.25 \* x + 7.5))&(!((y > 1.4 \* x + 9.8) & (y < 6 \* x + 42 & y >= 0 & x <= 6)));  
 }  
 public static boolean f2(double x, double y){  
 return ((y > -1.67 \* x - 5.33) & (y > 2.5 \* x - 9.5) &  
 (y < 6 \* x + 10) & (y < 0.4 \* x + 4.4) || (y > - 0.33 \* x - 1 & x >= 3 & x <= 6))&(!((y < x + 2) & (y > - 0.83 \* x + 2)));  
 }  
 public static boolean f1edge (double x, double y){  
 return ((y == -6 \* x - 54) || (y == 0.67 \* x - 0.67) || (y == 3.67 \* x + 14.33)  
 || (y == 2 \* x + 18) || (y == 0.25 \* x + 7.5));  
 }  
 public static boolean f2edge(double x, double y){  
 return (y == -1.67 \* x - 5.33) || (y > 2.5 \* x - 9.5) ||  
 (y == 6 \* x + 10) || (y == 0.4 \* x + 4.4) || (y == -0.33 \* x - 1 & x >= 3 & x <= 6);  
 }  
 public static void PrintAns (int n, int[][] arr){  
 for (int i = 0; i < n; i++){  
 if ((*f1*(arr[0][i], arr[1][i])&*f1edge*(arr[0][i], arr[1][i])) || *f2*(arr[0][i], arr[1][i])&*f2edge*(arr[0][i], arr[1][i]))  
 System.*out*.println("for x=" + arr[0][i] + " and y=" + arr[1][i] + " ans = 0 \t");  
 else if (*f1*(arr[0][i], arr[1][i]) || *f2*(arr[0][i], arr[1][i])) {  
 System.*out*.println("for x=" + arr[0][i] + " and y=" + arr[1][i] + " ans = 1 \t");  
 }  
 else System.*out*.println("for x=" + arr[0][i] + " and y=" + arr[1][i] + " ans = -1 \t");  
 }  
 }  
}  
// fix edges

# Задание 11

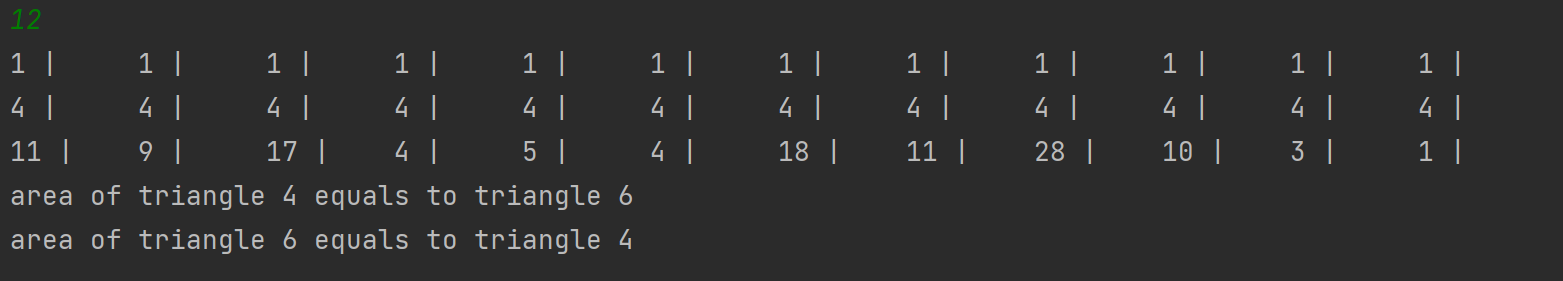
## 11.1. Постановка задачи

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 задачи, код программы

import java.util.\*;  
  
import static java.lang.Math.\*;  
  
public class \_11 {  
 public static void main(String[] args){  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("input n");  
 int n = in.nextInt();  
 int [][] arr = new int[2][3 \* n];  
 *getArr*(arr);  
 double s = 0; double ab; double bc; double ca;  
 double [][] ans = new double[3][n];  
 for (int j = 0, i = 0; j < 3 \* n; j += 3, i++){  
 ab = *sqrt*(*pow*(arr[0][j] - arr[0][j + 1], 2) + *pow*(arr[1][j] - arr[1][j + 1], 2));  
 bc = *sqrt*(*pow*(arr[0][j + 1] - arr[0][j + 2], 2) + *pow*(arr[1][j + 1] - arr[1][j + 2], 2));  
 ca = *sqrt*(*pow*(arr[0][j + 2] - arr[0][j], 2) + *pow*(arr[1][j + 2] - arr[1][j], 2));  
 s = 0.5 \* *abs*((arr[0][j] - arr[0][j + 2]) \* (arr[1][j + 1] - arr[1][j + 2]) - (arr[0][j + 1] - arr[0][j + 2]) \* (arr[1][j] - arr[1][j + 2]));  
 if (s == 0){  
 ans[0][i] = 0;  
 ans[1][i] = 0;  
 }  
 else {  
 ans[0][i] = 1;  
 ans[2][i] = s;  
 }  
 if ((ab == bc & bc == ca) & s != 0) ans[1][i] = 1;  
 else if ((ab == bc || bc == ca || ca == ab) & s != 0) ans[1][i] = 3;  
 else if ((*pow*(ab,2) == *pow*(bc,2) + *pow*(ca, 2) || *pow*(bc,2) == *pow*(ab,2) + *pow*(ca, 2) || *pow*(ca,2) == *pow*(bc,2) + *pow*(ab, 2)) & s!= 0) ans[1][i] = 2;  
 else if (s != 0) ans[1][i] = 4;  
  
 }  
 *PrintOut*(ans);  
  
 for (int i = 0; i < n; i++){  
 for (int j = 0; j < n; j++){  
 if (ans[2][i] == ans[2][j] & i != j) System.*out*.println("area of triangle " + (i + 1) + " equals to triangle " + (j + 1));  
 }  
 }  
 }  
 public static void getArr(int [][] arr){  
 Random r = new Random();  
 for (int i = 0; i < arr.length; i++) {  
 for (int j = 0; j < arr[i].length; j++) {  
 arr[i][j] = r.nextInt(10);  
 }  
 }  
 }  
 public static void PrintOut(double[][] arr){  
 for (double[] ints : arr) {  
 for (double anInt : ints) {  
 System.*out*.print((int)anInt + " | \t");  
 }  
 System.*out*.println();  
 }  
 }  
}

## 11.3. Тестирование работы программы



# 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 задачи, код программы

import java.util.Scanner;  
import static java.lang.System.*exit*;  
  
public class \_12 {  
 public static void main(String[] args){  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("input n");  
 int n = in.nextInt();  
 int [][] arr = new int[n][n];  
 int [][] arr\_1 = new int[n][n];  
 System.*out*.println("input elements of array");  
 for (int i = 0; i < arr.length; i++) {  
 for (int j = 0; j < arr[i].length; j++) {  
 arr[i][j] = in.nextInt();  
 }  
 }  
 System.*out*.println("input elements of array\_1");  
 for (int i = 0; i < arr\_1.length; i++) {  
 for (int j = 0; j < arr\_1[i].length; j++) {  
 arr\_1[i][j] = in.nextInt();  
 }  
 }  
 int [][] e = new int[n][n];  
 for (int i = 0; i < n; i++){  
 for (int j = 0; j < n; j++){  
 for (int k = 0; k < n; k++){  
 e[i][j] += arr[i][k] \* arr\_1[k][j];  
 }  
 }  
 }  
 for (int i = 0; i < n; i++){  
 for (int j = 0; j < i; j++){  
 if (e[i][i] != 1 || e[i][j] != 0 || e[n - i - 1][n - 1 - j] != 0) {  
 System.*out*.println("Wrong input");  
 *exit*(0);  
 }  
 }  
 }  
 }  
}

## 12.3. Тестирование работы программы

