Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе № 8

Вариант 13

тема «Массивы»

по дисциплине «Информатика»

Выполнил: студент группу ИСТ-22-1б Ипанов М.В.

Проверил: доцент каф. ВММБ Каменских А.А.

Пермь, 2022

**Содержание**

[Задание 1 3](#_Toc123992370)

[1.1. Постановка задачи 3](#_Toc123992371)

[1.2. Решение задачи, код программы 3](#_Toc123992372)

[1.3. Тестирование работы программы 4](#_Toc123992373)

[Задание 2 5](#_Toc123992374)

[2.1. Постановка задачи 5](#_Toc123992375)

[2.2. Решение задачи. Код программы 5](#_Toc123992376)

[2.3. Тестирование работы программы 6](#_Toc123992377)

[Задание 3 7](#_Toc123992378)

[3.1. Постановка задачи 7](#_Toc123992379)

[3.2. Решение задачи. Код программы 7](#_Toc123992380)

[3.3. Проверка решения 8](#_Toc123992381)

[Задание 4 9](#_Toc123992382)

[4.1. Постановка задачи 9](#_Toc123992383)

[4.2. Решение задачи. Код программы 9](#_Toc123992384)

[4.3. Проверка решения 10](#_Toc123992385)

[Задание 5 11](#_Toc123992386)

[5.1. Постановка задачи 11](#_Toc123992387)

[5.2. Решение задачи. Код программы 12](#_Toc123992388)

[5.3. Проверка решения 13](#_Toc123992389)

# Задание 1

## 1.1. Постановка задачи

Ввести массив, состоящий из N элементов целого типа. Массив является элементами целого дробной части числа в семеричной системе счисления. Проверить правильность введения элементов числа, т.е. в массиве должны быть только 0, 1, 2, 3, 4, 5 и 6. Считается, что в массиве ведены только те цифры, что стоят после запятой. Если число введено верно, то перевести его в десятичную систему счисления и поэлементно записать число в массив, так же только цифры после запятой (ограничиваясь 6 цифрами после запятой, если число при переводе не конечно). Пример:

|  |  |
| --- | --- |
| В 7-ой | В 10-ой |
| |  |  | | --- | --- | | 1 | 5 | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 2 | 4 | 4 | 8 | 9 | 7 | |
| 0,15 | 0,244897….. |

## 1.3. Тестирование работы программы

Таблица 1

Тестирование работы программы и проверка результатов решения

|  |  |  |
| --- | --- | --- |
| № п.п. | Решение Java | Проверка |
| 1 |  | верно |
| 2 |  | верно |
| 3 |  | верно |

Программа работала корректно со всеми введёнными данными.

# Задание 2

## 2.1. Постановка задачи

Дано целое число М и набор из М целых чисел, содержащий по крайней мере два нуля. Вывести сумму чисел из данного набора, расположенных между двумя нулями в одномерный массив (если нули идут подряд, то сумма 0).

## 2.3. Тестирование работы программы

Тестирование работы программы и проверка результатов решения

|  |  |  |
| --- | --- | --- |
| № п.п. | Решение Java | Проверка |
| 1 |  | верно |
| 2 |  | верно |
| 3 |  | верно |

В программе реализована функция rundom которая может облегчить ввод произвольных данных. Она срабатывает только при вводе Y или y. Вывод реализован как указано в задании.

# 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 окружности xc и yc и радиус окружности R, окружность определена на координатной плоскости. Определить номера точек, которые лежат вне окружности.

## 3.3. Проверка решения

Таблица 1

Тестирование работы программы и проверка результатов решения

|  |  |  |
| --- | --- | --- |
| № п.п. | Решение Java | Проверка |
| 1 |  | верно |
| 2 |  | верно |
| 3 |  | верно |

Программа выводит исходные данные, чтобы можно было удостоверится в правильности результатов.

# Задание 4

## 4.1. Постановка задачи

Задать массив из произвольных чисел и реализовать алгоритм «шейкер-сортировки». Образно алгоритм можно описать так: на каждом шаге основного цикла рассматривается массив данных A[left,right], по двум внутренним циклам: 1) от left к right A[i]>=A[i+1] тогда они меняются местами, иначе ничего не происходит, после окончания цикла right = right-1; 2) от right к left A[i-1]>=A[i] тогда они меняются местами, иначе ничего не происходит, после окончания массива left = left+1. После 1 шага минимальный и максимальный элемент в исходном массиве перетекают к краям, минимальный в — A[left], максимальный – в A[right]. Остановкой итерационной процедуры является равенство 0 перестановок элементов в массиве. Выполнить сравнение «шейкер-сортировки» с сортировкой «пузырек», рассмотрев количество итераций, а так же промежуточные результаты сортировки.

## 4.3. Проверка решения

Таблица 1

Тестирование работы программы и проверка результатов решения

|  |  |
| --- | --- |
| № п.п. | Решение Java |
| 1 |  |
| 2 |  |
| 3 |  |

Сравнение ОПТИМИЗИРОВАННЫХ алгоритмов сортировки пузырьком и Шейкер перестановки на одинаковых массивах показали, что они делают всегда одинаковое количество перестановок. Других способов замерить быстродействие я не нашёл.

# Задание 5

## 5.1. Постановка задачи

. В заданной матрице поменять местами две строки: строку с указанным номером и строку, содержащую наименьший элемент матрицы. Если строка с указанным номером совпадает со строкой, содержащей наименьший элемент матрицы, тогда выполнить генерацию двумерного массива еще раз и т.д. пока строка, содержащая наименьший элемент матрицы, не будет совпадать со строкой с указанным номером.

## 5.3. Проверка решения

Тестирование работы программы и проверка результатов решения

|  |  |  |
| --- | --- | --- |
| № п.п. | Решение Java | Проверка |
| 1 |  | верно |
| 2 |  | верно |

При тестировании программы на некоторых реальных и произвольных данных при правильном вводе данных ошибок выявлено не было. Сравнение проверки и ответа, который выдал код показывают правильность написанного кода.

# Задание 6

## 6.1. Постановка задачи

Даны целые положительные числа М, N, число D и набор из М чисел. Сформировать матрицу размера М × N, у которой первый столбец совпадает с исходным набором чисел, а элементы каждого следующего столбца равны сумме соответствующего элемента предыдущего столбца и числа D (в результате каждая строка матрицы будет содержать элементы арифметической прогрессии, проверить данный факт).

## 6.3. Проверка решения

Тестирование работы программы и проверка результатов решения

|  |  |  |
| --- | --- | --- |
| № п.п. | Решение Java | Проверка |
| 1 |  | верно |
| 2 |  | верно |

При проверке мы убедились, что программа составила таблицу арифметической прогрессии

# Задание 7

## 7.1. Постановка задачи

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

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

## 7.3. Проверка решения

Тестирование работы программы и проверка результатов решения

|  |  |  |
| --- | --- | --- |
| № п.п. | Решение Java | Проверка |
| 1 |  | верно |
| 2 |  | верно |

Красивый узор, между прочим, если учесть, что его можно составить и до 1 000 000, но к сожалению, в таблицу он не поместиться.

# 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:



## 8.3. Проверка решения

Тестирование работы программы и проверка результатов решения

|  |  |  |
| --- | --- | --- |
| № п.п. | Решение Java | Проверка |
| 1 |  | верно |
| 2 |  | верно |

Воссоздать искомый узор в оригинале мне не удалось (он на столько плох, что даже не поддаётся описанию). Но некое подобие всё-таки получилось, и его даже можно масштабировать.

# 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:



## 9.3. Проверка решения

Тестирование работы программы и проверка результатов решения

|  |  |  |
| --- | --- | --- |
| № п.п. | Решение Java | Проверка |
| 1 |  | верно |
| 2 |  | верно |
|  |  | верно |

Воссоздать искомый узор в оригинале мне не удалось (он на столько плох, что даже не поддаётся описанию). Но некое подобие всё-таки получилось, и его даже можно масштабировать.

# 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.

## 10.3. Проверка решения

Тестирование работы программы и проверка результатов решения

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

Всё сошлось, получилось даже доработать алгоритм, и сейчас программа определяет попадание точки на границу фигуры. Да, в принципе, что и требовалось в задании. Для читабельности кода сделал вывод «человеческим» языком.

# 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.

## 11.3. Проверка решения

Тестирование работы программы и проверка результатов решения

|  |  |  |
| --- | --- | --- |
| № п.п. | Решение Java | Проверка |
| 1 |  |  |
| 2 |  |  |
|  |  |  |

При тестировании программы на некоторых реальных и произвольных данных при правильном вводе данных ошибок выявлено не было. Сравнение проверки и ответа, который выдал код показывают правильность написанного кода.

# 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.

## 11.3. Проверка решения

Тестирование работы программы и проверка результатов решения

|  |  |  |
| --- | --- | --- |
| № п.п. | Решение Java | Проверка |
| 1 |  | Две единичные матрицы являются обратными |
| 2 |  | Матрицы разной размерности, поэтому невозможно |

При тестировании программы на некоторых данных, полученных при ручном решении, при правильном вводе, ошибок выявлено не было. Сравнение проверки и ответа, который выдал код показывают правильность написанного кода.

# Приложение (весь код)

import java.util.\*;  
import static java.lang.Math.\*;  
  
class func {  
  
 public static boolean num1\_1(int x, int n) {  
 // является ли число n-ричным?  
  
 for (int i = 1; i <= 10; i++) {  
 if (x % 10 >= n) {  
 return false;  
 }  
 x /= 10;  
 }  
 return true;  
 }  
  
 public static int[] num1\_2(int x) {  
 // перевод int в массив  
  
 int N = 0;  
 int x\_copy = x;  
 for (int i = 0; i < 10; i++) {  
 if (x\_copy % 10 > 0) {  
 N++;  
 }  
 x\_copy /= 10;  
 }  
 x\_copy = x;  
 int[] mass = new int[N];  
 for (int i = N - 1; i >= 0; i--) {  
 mass[i] = x\_copy % 10;  
 x\_copy /= 10;  
 }  
 return mass;  
 }  
  
 public static int[] num1\_3(int[] mass, int N) {  
 // перевод массива в N-ричную СС (массив)  
  
 double x\_d = 0;  
 for (int i = 0; i < mass.length; i++) {  
 x\_d += mass[i] \* *pow*(N, -i - 1);  
 }  
  
 int[] mass\_new = new int[6];  
 int x\_i = (int) (x\_d \* *pow*(10, 6));  
 for (int i = 0; i < 6; i++) {  
 mass\_new[5 - i] = x\_i % 10;  
 x\_i /= 10;  
  
 }  
  
 return mass\_new;  
 }  
  
 public static double num4\_1(int[] mass) {  
 // перевод массива в N-ричную СС (массив)  
 int s\_i = 0;  
 for (int i : mass) {  
 s\_i += i;  
 }  
 return ((double) s\_i) / mass.length;  
 }  
  
 public static double[][] num7\_1(int m, int x) {  
  
 double[][] mass = new double[m][m];  
 int S = ((mass.length + 1) \* mass.length) / 2 - 1 + x;  
 mass = *num7\_2*(mass, x, 0, S);  
 return mass;  
 }  
  
 public static double[][] num7\_2(double[][] mass, int x, int m, int S) {  
  
 if (S - x == 5) {  
 mass[m][m \* 2] = x;  
 x += 1;  
 mass[m + 1][m \* 2 + 1] = x;  
 x += 1;  
 mass[m + 2][m \* 2 + 2] = x;  
 x += 1;  
 mass[m + 1][m \* 2 + 2] = x;  
 x += 1;  
 mass[m][m \* 2 + 2] = x;  
 x += 1;  
 mass[m][m \* 2 + 1] = x;  
 return mass;  
 } else if (S - x == 2) {  
 mass[m][m \* 2] = x;  
 x += 1;  
 mass[m + 1][m \* 2 + 1] = x;  
 x += 1;  
 mass[m][m \* 2 + 1] = x;  
 return mass;  
 } else if (S == x) {  
 mass[m][m \* 2] = x;  
 return mass;  
 } else {  
 int n\_1 = m \* 2;  
 for (int m\_i = m; m\_i <= mass.length - 1 - m \* 2; m\_i++) {  
 mass[m\_i][n\_1] = x;  
 n\_1 += 1;  
 x += 1;  
 }  
 for (int m\_i = mass.length - 2 - m \* 2; m\_i >= m; m\_i--) {  
 mass[m\_i][mass.length - 1 - m] = x;  
 x += 1;  
 }  
 for (int n\_i = mass.length - 2 - m; n\_i >= m \* 2 + 1; n\_i--) {  
 mass[m][n\_i] = x;  
 x += 1;  
 }  
  
 *num7\_2*(mass, x, m + 1, S);  
 return mass;  
 }  
 }  
  
 public static double[][] num8\_1(double[][] mass, double[][] start) {  
  
 for (int m = 0; m < mass.length; m++) {  
 for (int n = 0; n < mass.length; n++) {  
 mass[m][n] = start[m % 4][n % 4];  
 }  
 }  
  
 return mass;  
 }  
  
 public static double[][] num9\_1(double[][] mass, double[][] start\_1, double[][] start\_2) {  
  
 for (int m = 0; m < (mass.length); m++) {  
 for (int n = 0; n < (mass.length); n++) {  
 mass[m][n] = start\_2[m % 4][n % 4];  
 }  
 }  
 for (int m = 0; m < (mass.length); m++) {  
 for (int n = 0; n < (mass.length); n++) {  
 if (m % 4 == 0 && n % 4 == 0 && *abs*(m - n) % 8 == 0) {  
 for (int m\_i = m; m\_i < m + 4; m\_i++) {  
 for (int n\_i = n; n\_i < n + 4; n\_i++) {  
 if (n\_i < mass.length && m\_i < mass.length) {  
 mass[m\_i][n\_i] = start\_1[m\_i % 4][n\_i % 4];  
 }  
 }  
 }  
 }  
 }  
 }  
  
 return mass;  
 }  
  
 public static int[][] num10\_1() {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Сколько точек вы хотите проверить?");  
 int n = in.nextInt();  
 int[][] mass = new int[n][3];  
 for (int i = 0; i < n; i++) {  
 System.*out*.printf("x\_%d = ", i + 1);  
 mass[i][0] = in.nextInt();  
 System.*out*.printf("y\_%d = ", i + 1);  
 mass[i][1] = in.nextInt();  
 }  
 return mass;  
 }  
  
 public static int[][] num10\_2(int[][] mass) {  
 int x, y;  
 for (int i = 0; i < mass.length; i++) {  
 x = mass[i][0];  
 y = mass[i][1];  
 // x1, y1, x2, y2  
 if (*tru\_line\_y*(0, x, y, 1, 4, -4, 0) ||  
 *tru\_line\_y*(0, x, y, -4, 0, -2, -6) ||  
 *tru\_line\_y*(0, x, y, -2, -6, -6, -4) ||  
 *tru\_line\_y*(0, x, y, -6, -4, -7, 5) ||  
 *tru\_line\_y*(0, x, y, -7, -5, 1, 4)  
 ||  
 *tru\_line\_y*(0, x, y, 3, -1, 5, 2) ||  
 *tru\_line\_y*(0, x, y, 5, 2, 6, -3) ||  
 *tru\_line\_y*(0, x, y, 6, -3, 3, -4) ||  
 *tru\_line\_y*(0, x, y, 3, -4, 1, -7) ||  
 *tru\_line\_y*(0, x, y, 1, -7, 1, -2) ||  
 *tru\_line\_y*(0, x, y, 1, -2, 3, -1)) {  
 mass[i][2] = 0;  
 } else if ((x <= -6 && func.*tru\_line\_y*(1, x, y, -6, -4, -7, 5) && func.*tru\_line\_y*(-1, x, y, -7, 5, 1, 4))  
 || (x >= -6 && x <= -4 && func.*tru\_line\_y*(-1, x, y, -7, 5, 1, 4) && func.*tru\_line\_y*(1, x, y, -6, -4, -2, -6))  
 || (x >= -4 && func.*tru\_line\_y*(-1, x, y, -2, -6, -4, 0) && func.*tru\_line\_y*(1, x, y, -2, -6, -6, -4))  
 || (x >= -4 && func.*tru\_line\_y*(-1, x, y, -7, 5, 1, 4) && func.*tru\_line\_y*(1, x, y, -4, 0, 1, 4))) {  
// System.out.print("Фигура 1");  
 mass[i][2] = 1;  
 } else if ((x >= 1 && x <= 3 && func.*tru\_line\_y*(-1, x, y, 1, -2, 3, -1) && func.*tru\_line\_y*(1, x, y, 1, -7, 3, -4))  
 || (x >= 3 && func.*tru\_line\_y*(-1, x, y, 3, -1, 5, 2) && func.*tru\_line\_y*(-1, x, y, 5, 2, 6, -3) && func.*tru\_line\_y*(1, x, y, 6, -3, 3, -4))) {  
// System.out.print("Фигура 2");  
 mass[i][2] = 1;  
 } else {  
// System.out.print("Ха, не попал!");  
 mass[i][2] = -1;  
 }  
 }  
 return mass;  
 }  
  
 public static void num10\_3(int[][] mass) {  
 int x;  
 for (int i = 0; i < mass.length; i++) {  
 x = mass[i][2];  
 if (x == 1) {  
 System.*out*.printf("%d точка попала в область.\n", i + 1);  
 } else if (x == -1) {  
 System.*out*.printf("%d точка не попала в область.\n", i + 1);  
 } else {  
 System.*out*.printf("%d точка попала на границу фигуры.\n", i + 1);  
 }  
 }  
 }  
  
 public static double[][] num11\_1(double[][] mass) {  
 Scanner in = new Scanner(System.*in*);  
 for (int i = 0; i < mass[1].length / 3; i++) {  
 System.*out*.printf("%d треугольник: \n", i + 1);  
 for (int j = 0; j < 3; j++) {  
 System.*out*.printf("x\_%d = ", j + 1);  
 mass[0][i \* 3 + j] = in.nextDouble();  
 System.*out*.printf("y\_%d = ", j + 1);  
 mass[1][i \* 3 + j] = in.nextDouble();  
 }  
 }  
 return mass;  
 }  
  
 public static double[][] num11\_2(double[][] mass) {  
  
 double[][] out = new double[mass[0].length / 3][3];  
 int side;  
 boolean right\_triangle;  
 for (int i = 0; i < mass[0].length / 3; i++) {  
 // проверка существования тругольника  
 // если да, то сразу заполнение результатов и переход к next итерации  
 right\_triangle = true;  
 for (int j = 0; j < 3; j++) {  
 if (*tru\_line\_y*(0, mass[0][i \* 3 + j], mass[1][i \* 3 + j],  
 mass[0][i \* 3 + (j + 1) % 3], mass[1][i \* 3 + (j + 1) % 3],  
 mass[0][i \* 3 + (j + 2) % 3], mass[1][i \* 3 + (j + 2) % 3])) {  
 right\_triangle = false;  
 break;  
 }  
 }  
 if (right\_triangle == false) {  
 continue;  
 }  
 out[i][0] = 1;  
  
 double a = *radius*(mass[0][i \* 3 + 0], mass[1][i \* 3 + 0], mass[0][i \* 3 + 1], mass[1][i \* 3 + 1]);  
 double b = *radius*(mass[0][i \* 3 + 1], mass[1][i \* 3 + 1], mass[0][i \* 3 + 2], mass[1][i \* 3 + 2]);  
 double c = *radius*(mass[0][i \* 3 + 2], mass[1][i \* 3 + 2], mass[0][i \* 3 + 0], mass[1][i \* 3 + 0]);  
  
 // проверка равенства сторон треугольника  
 if (a == b && a == c) {  
 out[i][1] = 1;  
 } else if (a == b || a == c || b == c) {  
 out[i][1] = 3;  
 } else {  
 out[i][1] = 4;  
 }  
  
 // проверка прямоугольности тругольника  
 if (*right\_triangle*(a, b, c) || *right\_triangle*(b, a, c) || *right\_triangle*(c, b, a)) {  
 out[i][1] = 2;  
 }  
  
 // площадь треугольника  
 double pp = (a + b + c) / 2;  
 out[i][2] = *sqrt*(pp \* (pp - a) \* (pp - b) \* (pp - c));  
 }  
 return out;  
 }  
  
 public static boolean tru\_line\_y(double up\_doun, double x, double y, double x1, double y1, double x2, double y2) {  
 // up\_doun = 1 если точка выше прямой  
 // = 0 на прямой  
 // = -1 если точка ниже прямой  
 boolean out;  
 if (up\_doun == 1) {  
 if (y > y1 + (x - x1) \* (y2 - y1) / (x2 - x1)) {  
 out = true;  
 } else {  
 out = false;  
 }  
 } else if (up\_doun == -1) {  
 if (y < y1 + (x - x1) \* (y2 - y1) / (x2 - x1)) {  
 out = true;  
 } else {  
 out = false;  
 }  
 } else {  
 if (y == y1 + (x - x1) \* (y2 - y1) / (x2 - x1)) {  
 out = true;  
 } else {  
 out = false;  
 }  
 }  
 return out;  
 }  
  
 public static double radius(double x1, double y1, double x2, double y2) {  
 return *sqrt*(*pow*(x1 - x2, 2) + *pow*(y1 - y2, 2));  
 }  
  
 public static boolean right\_triangle(double a, double b, double c) {  
 double gip = *pow*(a, 2);  
 double summ = *pow*(b, 2) + *pow*(c, 2);  
 gip = *round*(gip \* 1000.0);  
 summ = *round*(summ \* 1000.0);  
  
 if (gip == summ) {  
 return true;  
 } else {  
 return false;  
 }  
 }  
  
 public static boolean num12\_1(double[][] mass\_1, double[][] mass\_2) {  
  
 double[][] ed\_matric = new double[mass\_1.length][mass\_2[0].length];  
 for (int m = 0; m < ed\_matric.length; m++) {  
 for (int n = 0; n < ed\_matric[0].length; n++) {  
 if (m == n) {  
 ed\_matric[m][n] = 1;  
 }  
 }  
 }  
  
 double x;  
 for (int m = 0; m < ed\_matric.length; m++) {  
 for (int n = 0; n < ed\_matric[0].length; n++) {  
 x = 0;  
 for (int i = 0; i < ed\_matric[0].length; i++) {  
 x += mass\_1[m][i] \* mass\_2[i][n];  
 }  
 if (*round*(x \* 1000) != *round*(ed\_matric[m][n] \* 1000)) {  
 return false;  
 }  
 }  
 }  
 return true;  
 }  
  
 public static double[][] mass\_in(int i) { // инициализация массива  
 Scanner in = new Scanner(System.*in*);  
  
 if (i == 2) {  
 System.*out*.println("Введите размерность матрицы:");  
 System.*out*.print("m (строк) = ");  
 int m = in.nextInt();  
 System.*out*.print("n (столбцов) = ");  
 int n = in.nextInt();  
 double[][] mass = new double[m][n];  
 return mass;  
 } else {  
 System.*out*.println("Введите размерность квадратной матрицы:");  
 System.*out*.print("n = ");  
 int n = in.nextInt();  
 double[][] mass = new double[n][n];  
 return mass;  
 }  
 }  
  
 public static double[][] mass\_enter(double[][] mass) { // ручной ввод массива  
 // ввод - пустой массив нужной размерности  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.printf("(ручной ввод массива %d\*%d)\n", mass.length, mass[0].length);  
 for (int i = 0; i < mass.length; i++) {  
 System.*out*.printf("%d строка: \n", i + 1);  
 for (int j = 0; j < mass[0].length; j++) {  
 System.*out*.printf("x\_%d = ", j + 1);  
 mass[i][j] = in.nextDouble();  
 }  
 }  
 return mass;  
 }  
  
  
  
 public static void mass\_out(double[][] mass) { // форматированный вывод массива (sp - промежуток между числами  
  
 int[][] mass\_spase = new int[mass.length][mass[0].length];  
 int k\_last = 0, k\_first = 1, k;  
 int x;  
 // определение позиций элемента в списке (для форматирования)  
 for (int m = 0; m < mass.length; m++) {  
 for (int n = 0; n < mass[0].length; n++) {  
 x = *abs*((int) (mass[m][n] % 1 \* 1000));  
 k = 3;  
 for (int i = 2; i >= 0; i--) {  
 if (x / *stepen*(10, i) > 0) {  
 k = i;  
 }  
 }  
 mass\_spase[m][n] = 3 - k;  
 k\_last = *max*(3 - k, k\_last);  
  
 x = *abs*((int) mass[m][n]);  
 k = 0;  
 if (x == 0) {  
 k++;  
 }  
 while (x > 0) {  
 k++;  
 x /= 10;  
 }  
 mass\_spase[m][n] += k \* 10;  
 k\_first = *max*(k, k\_first);  
 }  
 }  
 // форматированный вывод результатов  
 for (int m = 0; m < mass.length; m++) {  
 for (int n = 0; n < mass[0].length; n++) {  
 if (mass[m][n] >= 0) {  
 System.*out*.print(" ");  
 }  
 for (int i = 0; i < k\_first - mass\_spase[m][n] / 10; i++) {  
 System.*out*.print(" ");  
 }  
 if (mass\_spase[m][n] % 10 == 0) {  
 System.*out*.print((int) mass[m][n]);  
 System.*out*.print(" ");  
 } else {  
 System.*out*.printf("%." + mass\_spase[m][n] % 10 + "f", mass[m][n]);  
 }  
 for (int i = 0; i < k\_last - mass\_spase[m][n] % 10 + 1; i++) {  
 System.*out*.print(" ");  
 }  
 }  
 System.*out*.println();  
 }  
 }  
  
 public static int stepen(int x, int n) { // форматированный вывод массива (sp - промежуток между числами  
 int S = 1;  
 for (int i = 0; i < n; i++) {  
 S \*= x;  
 }  
 return S;  
 }  
  
 public static double[][] mass\_rundom(int m, int n, int min, int max) { // форматированный вывод массива (sp - промежуток между числами  
 double[][] mass = new double[m][n];  
 for (int m\_i = 0; m\_i < mass.length; m\_i++) {  
 for (int n\_i = 0; n\_i < mass[0].length; n\_i++) {  
 mass[m\_i][n\_i] = *round*(*random*() \* (max - min) + min);  
 }  
 }  
 return mass;  
 }  
}  
  
class Work\_is\_start { // вариант 13  
  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
  
 int x\_i, i\_i, n\_i, y\_i, t\_i;  
 double x\_d, y\_d, r\_d;  
 boolean flag = false;  
  
  
 int num;  
 while (true) {  
 System.*out*.println("\_\_\_Введите номер задачи от 1 до 12\_\_\_");  
 num = in.nextInt();  
 switch (num) {  
 case 1:  
 System.*out*.println("ВВедите число, в 7-ричной СС, которое стодержит десятичную часть:");  
 in.nextLine();  
 while (true) {  
 String x\_s = in.nextLine();  
  
 i\_i = x\_s.indexOf(",");  
 if (i\_i == -1) {  
 System.*out*.println("(Проверте правильность написания числа и повторите попытку)");  
 continue;  
 }  
 x\_i = Integer.*parseInt*(x\_s.substring(i\_i + 1));  
 if (func.*num1\_1*(x\_i, 7)) {  
 break;  
 }  
 System.*out*.println("(Проверте правильность написания числа и повторите попытку)");  
 }  
 int[] mass\_1 = func.*num1\_2*(x\_i);  
  
 System.*out*.println(Arrays.*toString*(mass\_1) + " ввод");  
 System.*out*.println(Arrays.*toString*(func.*num1\_3*(mass\_1, 9)) + " ответ");  
 break;  
  
 case 2:  
 System.*out*.println("Введите М - колличество чисел: ");  
 n\_i = in.nextInt();  
 int[] mass\_2 = new int[n\_i];  
 System.*out*.println("Желаете воспользоватеся random (иначе вводить вручную)? Y or N");  
 in.nextLine();  
 String run = in.nextLine();  
  
 if (run.equals("y") || run.equals("Y")) {  
 x\_i = (int) (n\_i \* *random*());  
 y\_i = (int) (n\_i \* *random*());  
 while (x\_i == y\_i) {  
 y\_i = (int) (n\_i \* *random*());  
 }  
 mass\_2[x\_i] = -1;  
 mass\_2[y\_i] = -1;  
 for (int i = 0; i < n\_i; i++) {  
 if (mass\_2[i] == -1) {  
 mass\_2[i] = 0;  
 } else {  
 mass\_2[i] = (int) (*random*() \* 20);  
 }  
 System.*out*.printf("x%d = %d \n", i + 1, mass\_2[i]);  
 }  
 } else {  
 for (int i = 0; i < n\_i; i++) {  
 System.*out*.printf("x%d = ", i + 1);  
 mass\_2[i] = in.nextInt();  
 }  
 x\_i = 0;  
 for (int i = 0; i < n\_i; i++) {  
 if (mass\_2[i] == 0) {  
 x\_i += 1;  
 }  
 }  
 if (x\_i < 2) {  
 System.*out*.println("Всё фигня, давай по новой. (мало нулей написал)");  
 flag = true;  
 }  
 }  
 ArrayList<Integer> mass\_new = new ArrayList<>();  
 for (int i = 1; i < n\_i - 1; i++) {  
 if (mass\_2[i - 1] == 0 && mass\_2[i + 1] == 0) {  
 mass\_new.add(0);  
 while (mass\_2[i] > 0) {  
 mass\_new.set(mass\_new.size() - 1, mass\_new.get(mass\_new.size() - 1) + mass\_2[i] % 10);  
 mass\_2[i] /= 10;  
 }  
 }  
 }  
 System.*out*.println(mass\_new.toString());  
 break;  
  
 case 3:  
 System.*out*.println("Сколько случайных точек вы хотите проверить?\n(в диапазоне от -10 до +10)");  
 n\_i = in.nextInt();  
 double[] mass\_3 = new double[2 \* n\_i];  
 for (int i = 0; i < 2 \* n\_i; i++) {  
 mass\_3[i] = (20 \* (*random*() - 0.5));  
 }  
 System.*out*.print("Введите измерения окружности:\nx = ");  
 x\_d = in.nextDouble();  
 System.*out*.print("y = ");  
 y\_d = in.nextDouble();  
 System.*out*.print("R = ");  
 r\_d = in.nextDouble();  
  
 System.*out*.println(Arrays.*toString*(mass\_3));  
 System.*out*.println("Вне окружности лежат точки:");  
 ArrayList<Double> mass\_3\_arrl = new ArrayList<>();  
 for (int i = 0; i < n\_i; i++) {  
 if (*pow*(*pow*((x\_d - mass\_3[i \* 2]), 2) + *pow*((y\_d - mass\_3[i \* 2 + 1]), 2), 1.0 / 2.0) > r\_d) {  
 System.*out*.printf("%d. x = %.1f, y = %.1f\n", (i + 1), mass\_3[2 \* i], mass\_3[2 \* i + 1]);  
 mass\_3\_arrl.add(mass\_3[2 \* i]);  
 mass\_3\_arrl.add(mass\_3[2 \* i + 1]);  
 }  
 }  
 break;  
 case 4:  
 System.*out*.println("Сколько элементов будет в массиве для сортировки?");  
 n\_i = in.nextInt();  
 System.*out*.println("Сколько итераций провести (чем больше, тем точнее)?");  
 t\_i = in.nextInt();  
  
 int[] mass\_41 = new int[n\_i], mass\_42 = new int[n\_i];  
 int[] time\_1 = new int[t\_i], time\_2 = new int[t\_i];  
 y\_i = 0;  
 for (int t = 0; t < t\_i; t++) {  
  
 for (int i = 0; i < n\_i; i++) {  
 mass\_42[i] = (int) (100 \* *random*());  
 mass\_41[i] = mass\_42[i];  
 }  
  
 i\_i = 0;  
 while (true) {  
 flag = false;  
 for (int j = i\_i; j <= (n\_i - 2) - i\_i; j++) {  
 if (mass\_42[j] > mass\_42[j + 1]) {  
 x\_i = mass\_42[j];  
 mass\_42[j] = mass\_42[j + 1];  
 mass\_42[j + 1] = x\_i;  
 flag = true;  
 y\_i += 1;  
 }  
 }  
 for (int j = (n\_i - 3); j >= i\_i; j--) {  
 if (mass\_42[j] > mass\_42[j + 1]) {  
 x\_i = mass\_42[j];  
 mass\_42[j] = mass\_42[j + 1];  
 mass\_42[j + 1] = x\_i;  
 flag = true;  
 y\_i += 1;  
 }  
 }  
 if (flag == false) {  
 time\_1[t] = y\_i;  
 y\_i = 0;  
 break;  
 }  
 i\_i += 1;  
 }  
  
 i\_i = 0;  
 while (true) {  
 flag = false;  
 for (int j = 0; j <= (n\_i - 2) - i\_i; j++) {  
 if (mass\_41[j] > mass\_41[j + 1]) {  
 x\_i = mass\_41[j];  
 mass\_41[j] = mass\_41[j + 1];  
 mass\_41[j + 1] = x\_i;  
 flag = true;  
 y\_i += 1;  
 }  
 }  
 if (flag == false) {  
 break;  
 }  
 i\_i += 1;  
 }  
 time\_2[t] = y\_i;  
 y\_i = 0;  
 }  
 System.*out*.println("Колличество перестановок в методах составила:");  
 System.*out*.printf(" - Методом Шейкер-сортировки = %.1f\n", func.*num4\_1*(time\_1));  
 System.*out*.printf(" - Методом Пузырька = %.1f\n", func.*num4\_1*(time\_2));  
 break;  
 case 5:  
 System.*out*.println("Какую строчку вы хотите заменить с минимальной? ");  
 int st\_1 = in.nextInt() - 1;  
 int st\_2 = st\_1;  
  
 double x\_min;  
 double[][] mass\_5 = func.*mass\_in*(2);  
 if (mass\_5.length == 1) {  
 System.*out*.println("Невозможно произвести замену.");  
 break;  
 }  
 while (st\_1 == st\_2) {  
 mass\_5 = func.*mass\_rundom*(mass\_5.length, mass\_5[0].length, -100, 100);  
 x\_min = mass\_5[0][0];  
 for (int m = 0; m < mass\_5.length; m++) {  
 for (int n = 0; n < mass\_5[0].length; n++) {  
 if (mass\_5[m][n] < x\_min) {  
 st\_2 = m;  
 x\_min = mass\_5[m][n];  
 }  
 }  
 }  
 }  
 System.*out*.println("Сгенерированна матрица");  
 func.*mass\_out*(mass\_5);  
  
 for (int n = 0; n < mass\_5[1].length; n++) {  
 x\_min = mass\_5[st\_1][n];  
 mass\_5[st\_1][n] = mass\_5[st\_2][n];  
 mass\_5[st\_2][n] = x\_min;  
 }  
 System.*out*.println("Изменённая матрица");  
 func.*mass\_out*(mass\_5);  
 break;  
 case 6:  
 double[][] mass\_6 = func.*mass\_in*(2);  
 System.*out*.print("Коофициэнт b: ");  
 int b = in.nextInt();  
 System.*out*.println("Заполните первый столбец матрицы:");  
 for (int m = 0; m < mass\_6.length; m++) {  
 mass\_6[m][0] = in.nextInt();  
 }  
 for (int m = 0; m < mass\_6.length; m++) {  
 for (int n = 1; n < mass\_6[0].length; n++) {  
 mass\_6[m][n] = mass\_6[m][n - 1] + b;  
 }  
 }  
 func.*mass\_out*(mass\_6);  
 break;  
 case 7:  
 System.*out*.println("Матрицу какого размера вы хотите заполнить?");  
 n\_i = in.nextInt();  
 func.*mass\_out*(func.*num7\_1*(n\_i, 1));  
 break;  
 case 8:  
 double[][] start\_8 = {{1, 0, 0, 0},  
 {9, 2, 0, 0},  
 {8, 10, 3, 0},  
 {7, 6, 5, 4}};  
 double[][] mass\_8 = func.*mass\_in*(1);  
 mass\_8 = func.*num8\_1*(mass\_8, start\_8);  
 func.*mass\_out*(mass\_8);  
 break;  
 case 9:  
 double[][] start\_9\_1 = {{1, 2, 3, 4},  
 {11, 12, 13, 5},  
 {16, 15, 14, 6},  
 {10, 9, 8, 7}};  
 double[][] start\_9\_2 = {{1, 3, 4, 10},  
 {2, 5, 9, 11},  
 {6, 8, 12, 15},  
 {7, 13, 14, 16}};  
 double[][] mass\_9 = func.*mass\_in*(1);  
 if (mass\_9.length % 2 == 1) {  
 System.*out*.println("n - должно быть только чётным.");  
 break;  
 }  
 mass\_9 = func.*num9\_1*(mass\_9, start\_9\_1, start\_9\_2);  
 func.*mass\_out*(mass\_9);  
 break;  
 case 10:  
 int[][] mass\_10 = func.*num10\_1*(); // ввод  
 mass\_10 = func.*num10\_2*(mass\_10); // вычисление  
 func.*num10\_3*(mass\_10); // вывод  
 break;  
 case 11:  
 System.*out*.println("Сколько треугольников вы хотите проверить?");  
 int n = in.nextInt();  
 double[][] mass\_11 = new double[2][3 \* n];  
 mass\_11 = func.*num11\_1*(mass\_11);  
 System.*out*.println("По введённым данным составлен массив:");  
 func.*mass\_out*(mass\_11);  
 System.*out*.println("Ответ:");  
 func.*mass\_out*(func.*num11\_2*(mass\_11));  
 break;  
 case 12:  
 System.*out*.println("-- A:");  
 double[][] mass12\_1 = func.*mass\_in*(1);  
 mass12\_1 = func.*mass\_enter*(mass12\_1);  
  
 System.*out*.println("\n-- B:");  
 double[][] mass12\_2 = func.*mass\_in*(1);  
 mass12\_2 = func.*mass\_enter*(mass12\_2);  
  
 if (mass12\_1.length == mass12\_2.length) {  
 if (func.*num12\_1*(mass12\_1, mass12\_2)) {  
 System.*out*.println("Эти матрицы являются обратными");  
 } else {  
 System.*out*.println("Эти матрицы НЕ являются обратными");  
 }  
 } else {  
 System.*out*.println("Невозможно проверсти умножение матриц");  
 }  
 break;  
 default:  
 System.*out*.println("Вы ввели неверный номер задания");  
 }  
 System.*out*.println();  
 }  
  
  
 }  
}

Конец!