**Лабораторная работа №14**

**Основы программирования на Java: многомерные массивы**

**Цель работы:** Освоить основные способы создания Java-программ либо с помощью обычного редактора, либо с помощью среды разработки. Приобрести навыки программирования с использованием многомерных массивов в Java.

**Теоретические сведения**

В Java есть как одномерные, так и многомерные массивы. Но реализация массивов в Java имеет свои особенности. Во-первых, массив в Java это объект. Этот объект состоит из размера массива (поле length) и собственно массива.

Для двумерных массивов ставится не одна пара скобок, а две, для трехмерных - три и т.д. Например. s = sAray[i][0]; tAray[i][j][k] = 10; Двумерный массив - это массив ссылок на объекты-массивы. Трехмерный массив - это массив ссылок на массивы, которые, в свою очередь, являются массивами ссылок на массивы. Как уже указывалось, массив является объектом, который, в частности, хранит поле length - размер массива. Это позволяет задавать обработку массивов произвольно. Они строятся по принципу "массив массивов". Возможные способы инициализации массивов:

1. явное создание

int ary[][] = new int[3][3];

2. использование списка инициализации

int ary[][] = new int[][] {

{1, 1, 1},

{2, 2, 2},

{1, 2, 3}, };

3. массивы в языке Java являются объектами некоторого встроенного класса, для этого класса существует возможность определить размер массива, обратившись к элементу данных класса с именем length, например:

int[] nAnotherNumbers;

nAnotherNumbers = new int[15];

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

{

nAnotherNumbers[i] = nInitialValue;

Поскольку Вы уже знаете, что такое массивы и как с ними работать, наверняка, для Вас не проблема:

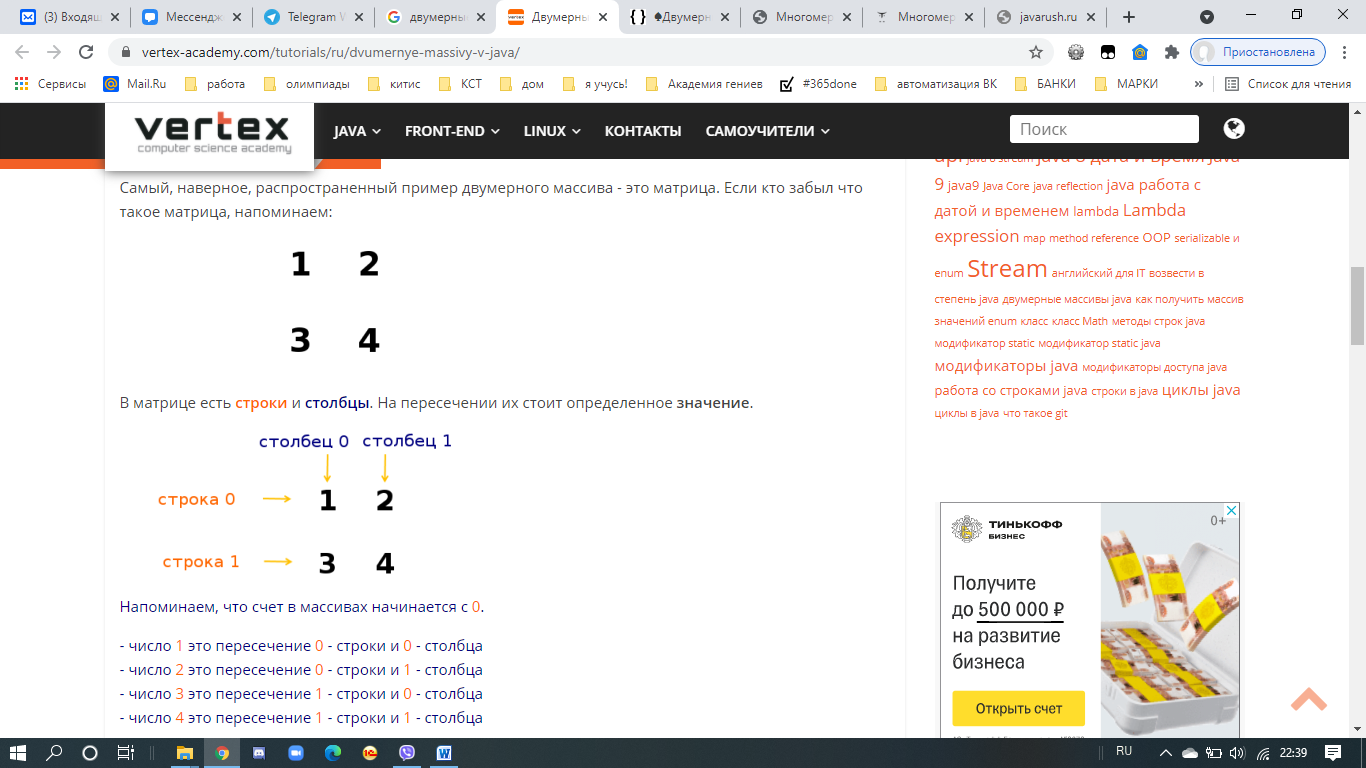
* создать одномерный массив
* заполнить его значениями
* и вывести в консоль

**Например:**

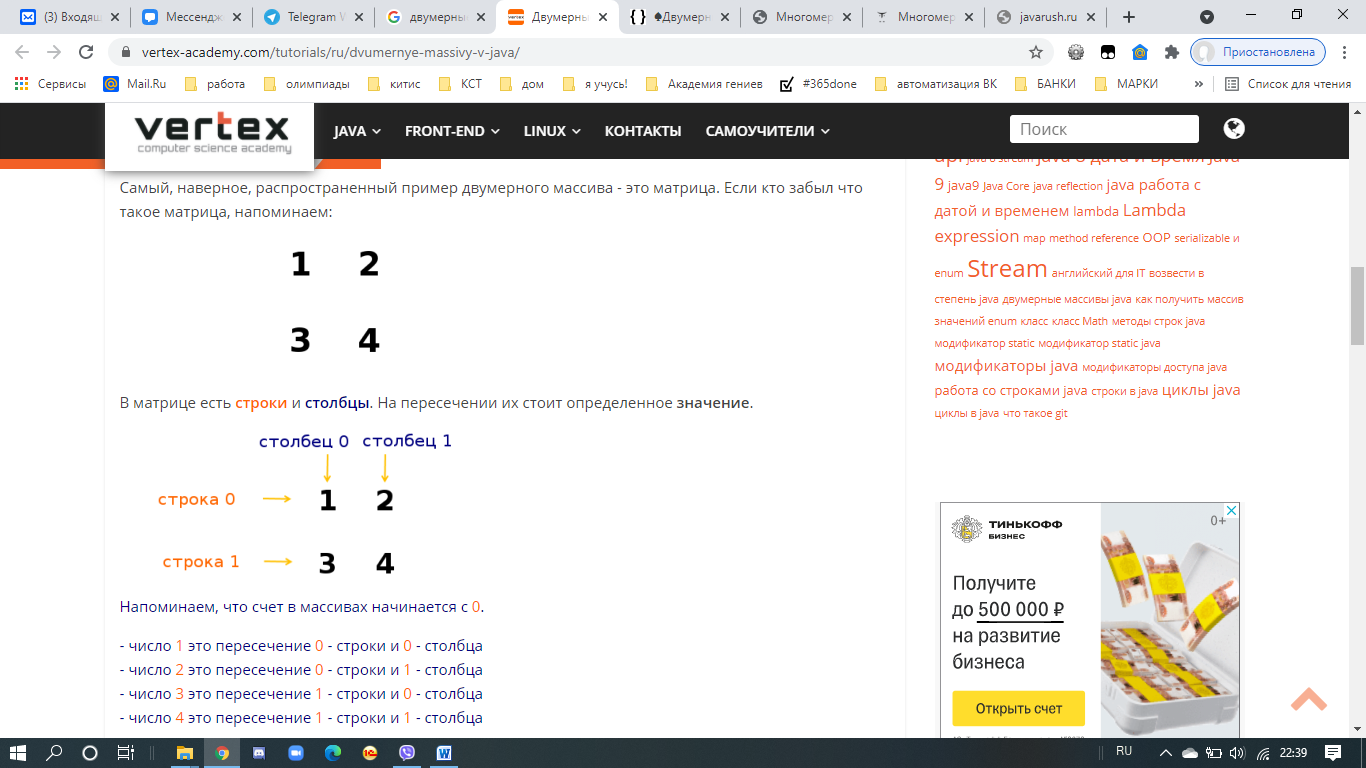
|  |
| --- |
| class Test {    public static void main(String[] args) {             int[] array = {51,136, 387};               for (int i = 0; i < array.length; i++) {             System.out.println(array[i]);             }  }  } |

**Ну, а что же с двумерным массивом?**

Самый, наверное, распространенный пример двумерного массива - это матрица. Если кто забыл что такое матрица, напоминаю:



В матрице есть **строки** и **столбцы**. На пересечении их стоит определенное**значение**.



Напоминаем, что счет в массивах начинается с 0.

- число 1 это пересечение 0 - строки и 0 - столбца  
- число 2 это пересечение 0 - строки и 1 - столбца  
- число 3 это пересечение 1 - строки и 0 - столбца  
- число 4 это пересечение 1 - строки и 1 - столбца

Ниже **пример объявления двумерного массива** на языке программирования Java:

|  |
| --- |
| class Test {    public static void main(String[] args) {          int[][] array = new int[2][2];  }  } |

Или можно сразу объявить содержимое массива:

|  |
| --- |
| class Test {    public static void main(String[] args) {          int[][] array = {{1,2}, {3,4}};  }  } |

При инициализации двумерного массива, можно заметить отличие от обычного массива. В двумерном массиве Вы используете **две квадратные скобки** вместо одной.

* в первой вы пишите количество **строк**
* во второй вы пишите количество **столбцов**

При заполнении двумерного массива Вы указываете в этих скобках **строку** и **столбец**.

|  |  |
| --- | --- |
| **Например** | class Test {    public static void main(String[] args) {  int[][] array = new int[2][2];    array[0][0] = 1;  array[0][1] = 2;  array[1][0] = 3;  array[1][1] = 4;  }  } |

**А как же вывести двумерный массив в консоль?**

Вывод двумерного массива **с помощью цикла for** сильно отличается от вывода обычного массива (когда используется цикл for).

**Чтобы вывести в каждую ячейку двумерного массива значение, не достаточно использовать один цикл for. Необходимо использовать два цикла for, при этом один из них находится в другом.**

**Почему так?**

Ранее упоминалось, что двумерный массив состоит из строк и столбцов. Каждая ячейка такого массива - это пересечение какой-то строки и столбца.

1. Так что **первый цикл for перебирает каждую строку двумерного массива** (которая содержит какое-то количество столбцов).

2. А **второй цикл for перебирает столбцы в этой строке.** Таким образом, можно заполнить значением каждый элемент двумерного массива.

**Например:**

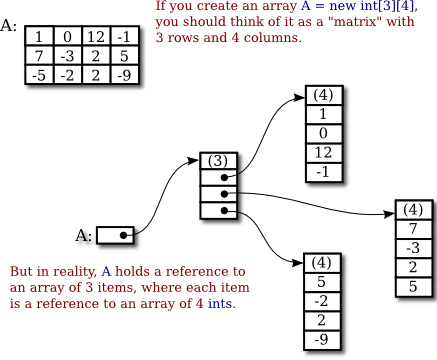
|  |
| --- |
| class Test {    public static void main(String[] args) {  int[][] array = new int[2][2];    array[0][0] = 1;  array[0][1] = 2;  array[1][0] = 3;  array[1][1] = 4;    for (int i = 0; i < array.length; i++) {  for (int j = 0; j < array[i].length; j++) {  System.out.print(array[i][j] + "\t");  }  System.out.println();  }  }  } |

**Если Вы запустите данный код на своем компьютере, в консоли Вы увидите:**

1    2  
3    4

Для представления матриц 3×2 необходимо 2 двумерных массива, состоящих из массива длины 3. Другими словами, каждая строка в двумерном массиве является одномерным массивом.

Java действительно не поддерживает многомерные массивы, но позволяет создавать и использовать массивы любого количества измерений. В истинном 2D массиве все элементы занимают непрерывный блок памяти, но в Java это не так. Вместо этого многомерный массив является массивом массива.



**Сортировка двумерного массива Java**

Пусть нам дан двумерный массив Порядка N X M и номер столбца K (1<=K<=m). Наша задача — отсортировать по значениям в столбце K.

Input : If our 2D array is given as (Order 4X4)

39 27 11 42

10 93 91 90

54 78 56 89

24 64 20 65

Sorting it by values in column 3

Output : 39 27 11 42

24 64 20 65

54 78 56 89

10 93 91 90

Универсальный способ сортировки массива заключается в использовании Arrays.sort.

// Java код для сортировки 2D матрицы

// по любой колонке

**import** java.util.\*;

**class** sort2DMatrixbycolumn {

// Function to sort by column

**public** **static** **void** sortbyColumn(**int** arr[][], **int** col)

{

// Using built-in sort function Arrays.sort

Arrays.sort(arr, **new** Comparator<**int**[]>() {

@Override

// Compare values according to columns

**public** **int** compare(**final** **int**[] entry1,

**final** **int**[] entry2) {

// To sort in descending order revert

// the '>' Operator

**if** (entry1[col] > entry2[col])

**return** 1;

**else**

**return** -1;

}

}); // End of function call sort().

}

// Driver Code

**public** **static** **void** main(**String** args[])

{

**int** matrix[][] = { { 39, 27, 11, 42 },

{ 10, 93, 91, 90 },

{ 54, 78, 56, 89 },

{ 24, 64, 20, 65 } };

// Sort this matrix by 3rd Column

**int** col = 3;

sortbyColumn(matrix, col - 1);

// Display the sorted 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();

}

}

}

Получим:

39 27 11 42

24 64 20 65

54 78 56 89

10 93 91 90

**Ассоциативные массивы в Java**

У ассоциативного массива в качестве ключей применяются строки. Мы можем представить его структуру данных в виде совокупности пары «ключ-значение». В таком массиве, как ассоциативный, любое значение связано с конкретным ключом, а доступ к значению производится по имени ключа.

Что касается языка программирования Java, то в нём ассоциативные массивы не поддерживаются. А ведь в некоторых ситуациях, работая с массивами, было бы неплохо обратиться к элементу не по индексу, а по ключу.

Однако есть способ реализовать похожую структуру данных, используя стандартные средства Java SE. В частности, мы можем создать простейший ассоциативный массив, воспользовавшись классом **HashMap** и установив для его экземпляра строковый тип данных ключа.

Посмотрим, как это работает:

HashMap**<**String**,** Integer**>** fruits **=** **new** HashMap**();**

fruits**.**put**(**"Число апельсинов"**,** 5**);**

fruits**.**put**(**"Число яблок"**,** 10**);**

fruits**.**put**(**"Число мандаринов"**,** 7**);**

System**.**out**.**println**(**fruits**.**get**(**"Число мандаринов"**));**

Мы можем создать ассоциативный массив и с более сложной структурой, поместив объекты HashMap друг в друга и получив тем самым «ассоциативные массивы в ассоциативном массиве». Посмотрим, как это выглядит на практике:

HashMap**<**String**,** Integer**>** fruits **=** **new** HashMap**();**

fruits**.**put**(**"Число апельсинов"**,** 5**);**

fruits**.**put**(**"Число яблок"**,** 10**);**

fruits**.**put**(**"Число мандаринов"**,** 7**);**

HashMap**<**String**,** Integer**>** milkProducts **=** **new** HashMap**();**

milkProducts**.**put**(**"Число пачек творога"**,** 2**);**

milkProducts**.**put**(**"Число пакетов молока"**,** 3**);**

milkProducts**.**put**(**"Число банок сметаны"**,** 17**);**

HashMap**<**String**,** HashMap**<**String**,** Integer**>** **>** eat **=** **new** HashMap**();**

eat**.**put**(**"Фрукты"**,** fruits**);**

eat**.**put**(**"Молочные продукты"**,** milkProducts**);**

System**.**out**.**println**(**eat**.**get**(**"Молочные продукты"**).**get**(**"Число пакетов молока"**));**

В результате мы решим проблему отсутствия ассоциативных массивов в Java.

[МНОГОМЕРНЫЕ МАССИВЫ](https://javarush.ru/groups/posts/mnogomernye-massivy#%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%B5%D1%81%D0%BD%D1%8B%D0%B5-%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B8-%D0%BD%D0%B0-%D0%B4%D0%B2%D1%83%D0%BC%D0%B5%D1%80%D0%BD%D1%8B%D0%B5-%D0%B8-%D1%82%D1%80%D1%91%D1%85%D0%BC%D0%B5%D1%80%D0%BD%D1%8B%D0%B5-%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2%D1%8B)

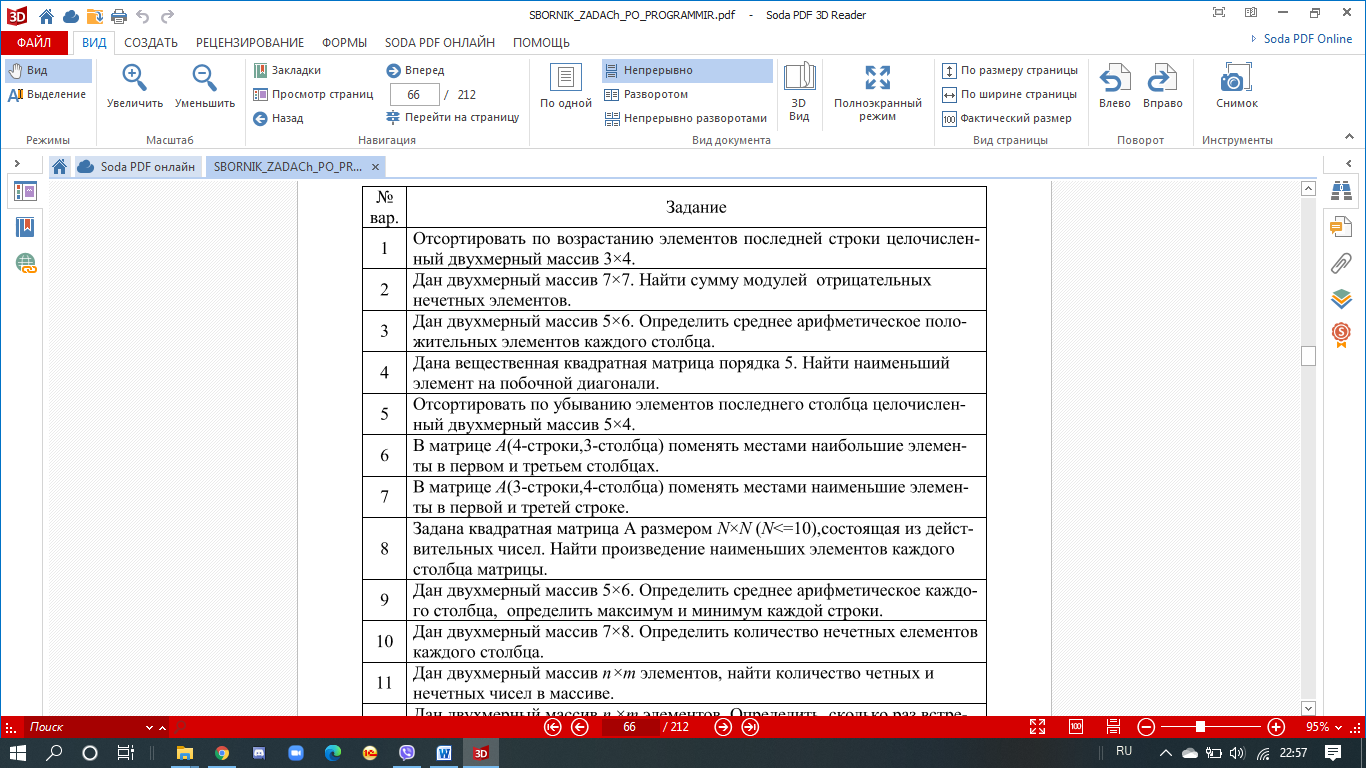
**Задание**

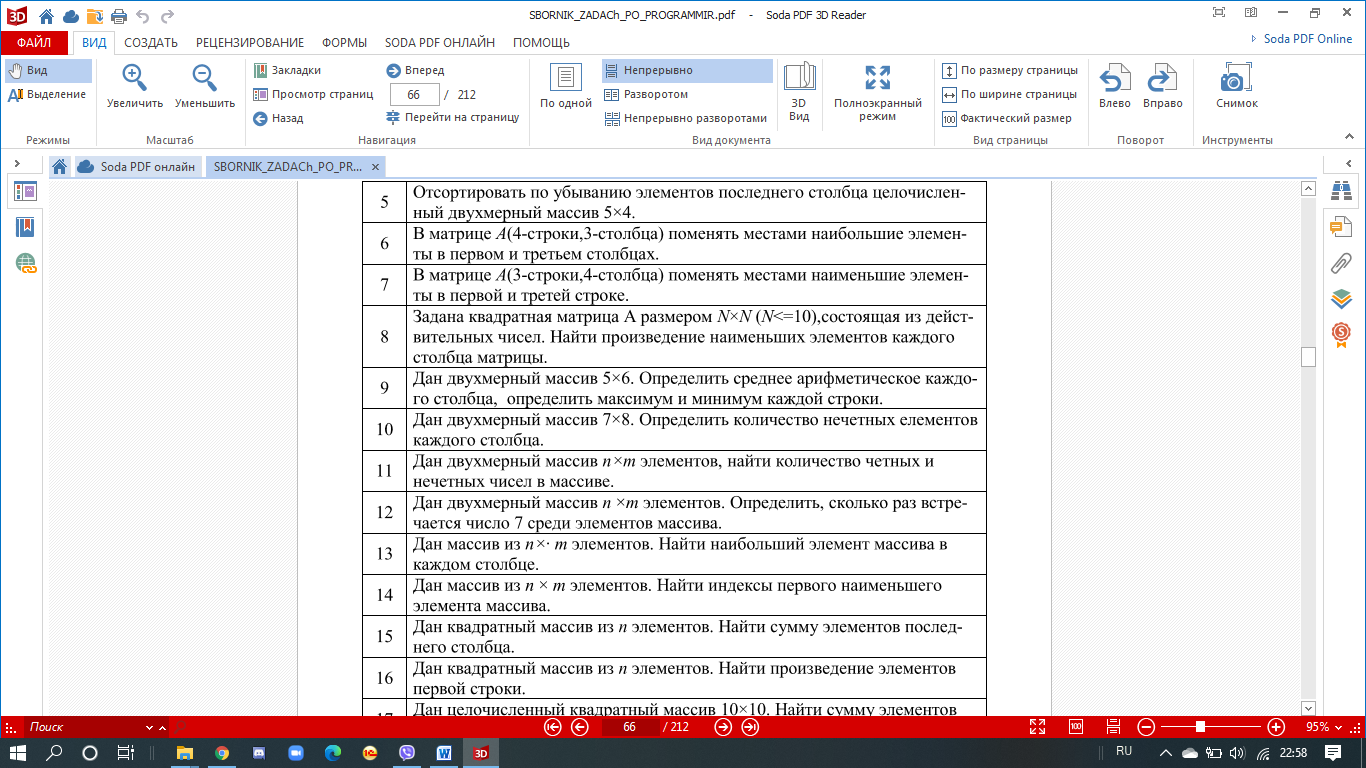
Изучить теоретический материал и выполнить задачи по вариантам.

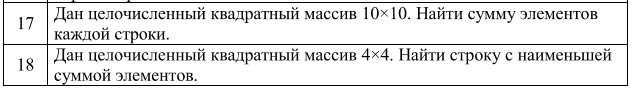
Составить блок-схемы алгоритмов и проекты программ решения задач.

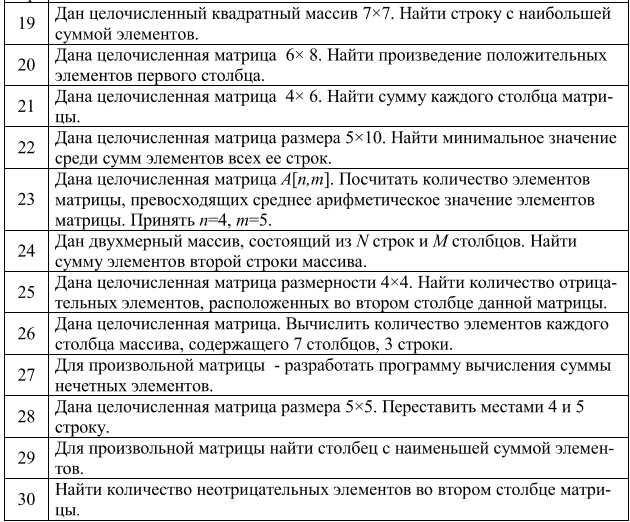
**Задачи**

**Задача №1**

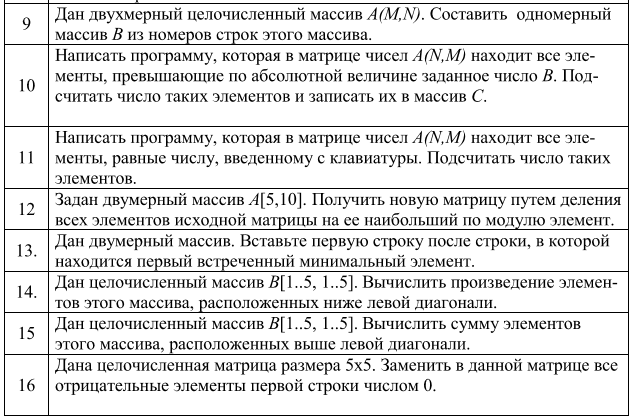
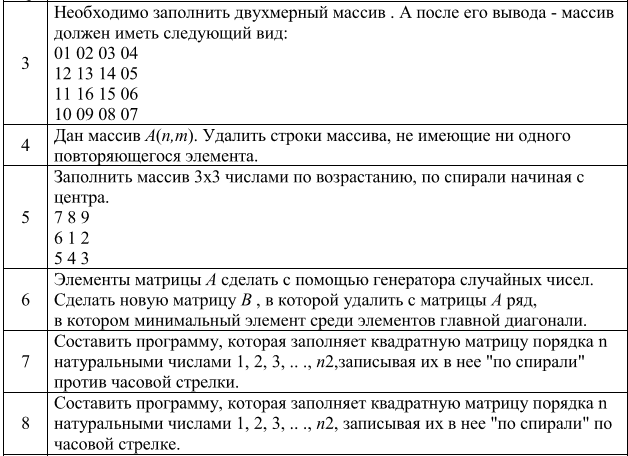
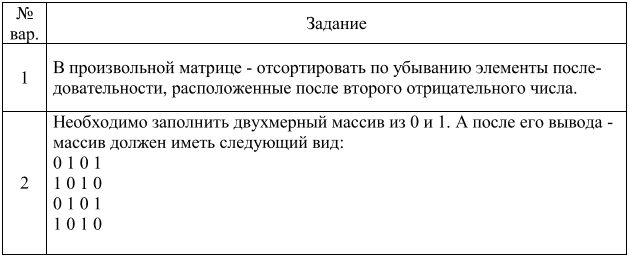


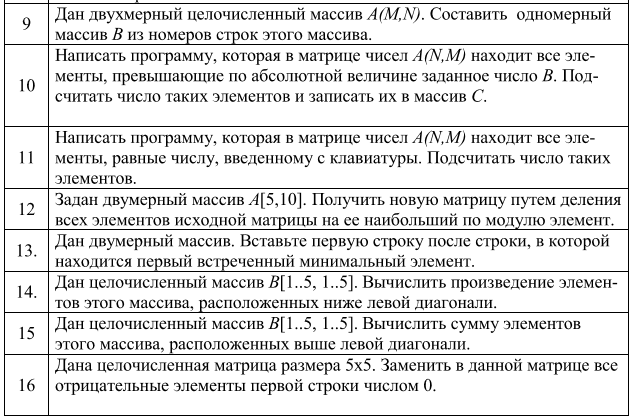


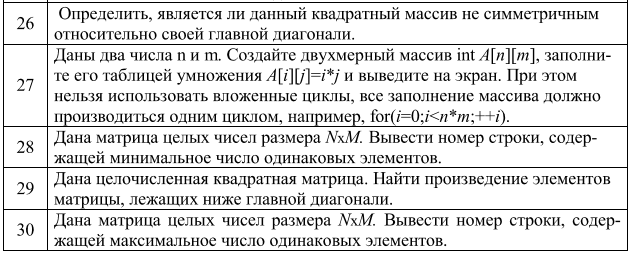
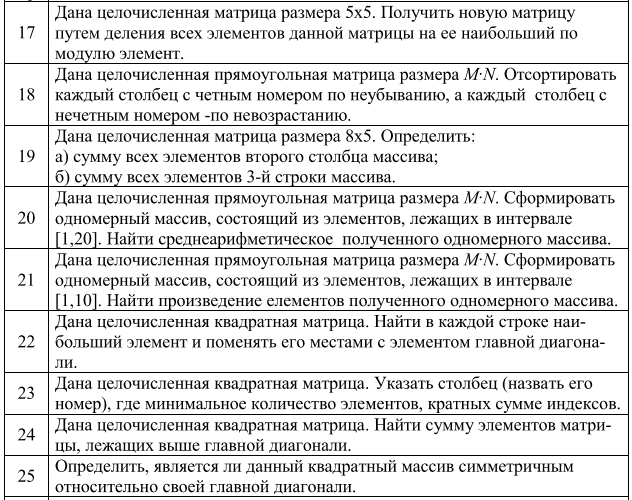




**Задача №2**







**Форма отчета**

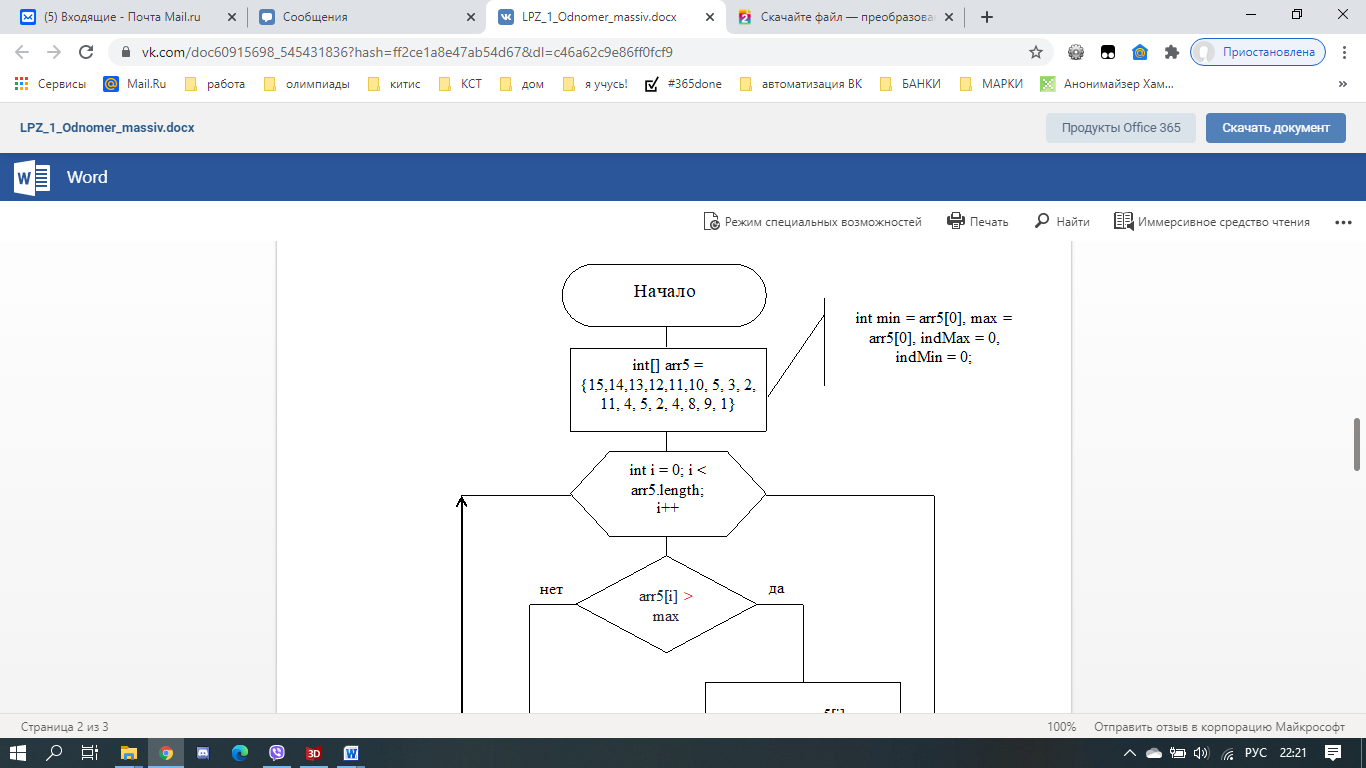
1. Тема, цель, индивидуальное задание

2. Таблица идентификаторов

|  |  |  |
| --- | --- | --- |
| Наименование переменной | Тип данных | Назначение |
|  |  |  |
|  |  |  |

3. Блок-схема алгоритма (см. Приложение)

Пример:



4. Код программы

5. Скриншот результата

**Контрольные вопросы для защиты**

1. Что такое массив?
2. Как объявить массив в Java?
3. Как обратиться к элементу массива? Примеры.
4. Способы инициализации массива.

**Приложение**

**Условные графические обозначения в схемах алгоритмов и программ, отображающие основные операции процесса обработки данных и программирования по ГОСТ 2.708 - 81**

Размер, а следует выбирать из ряда 10, 15, 20 мм. Допускается увеличение размера а на число, кратное 5. Размер b принима­ют равным 1,5а.

|  |  |
| --- | --- |
|  | Процесс. Выполнение операций (группы операций), в результате которых изменяется значение (форма представления, расположение) данных  Решение. Выбор направления алгоритма (програм­мы) в зависимости от некоторых переменных условий |
|  | Решение. Выбор направления алгоритма (програм­мы) в зависимости от некоторых переменных условий |
|  | Модификация. Выполнение операций, меняющих команды (группы команд), изменяющих программу |
|  | Предопределенный процесс. Использование ранее созданных и описанных отдельно алгоритмов (программ) |
|  | Ввод-вывод. Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод) |
|  | Соединитель. Указание связи между прерванны­ми линиями потока (связывющие символы) |
|  | Пуск-останов. Начало, конец, прерывание про­цесса обработки данных или выполнения программы |
|  | Межстрочный соединитель |