Мобильная разработка

Урок 4

**Массивы**

Массив — это структура данных, в которой хранятся элементы одного типа. Его можно представить, как набор пронумерованных ячеек, в каждую из которых можно поместить какие-то данные (один элемент данных в одну ячейку). Доступ к конкретной ячейке осуществляется через её номер. Номер элемента в массиве также называют индексом. В случае с Java массив однороден, то есть во всех его ячейках будут храниться элементы одного типа. Так, массив целых чисел содержит только целые числа (например, типа int), массив строк — только строки. То есть в Java мы не можем поместить в первую ячейку массива целое число, во вторую String, а в третью вещественное число.

Количество элементов в массиве называется размером массива, а тип элементов — типом массива.

**Объявление и создание массивов.**

При объявлении массива, как и при объявлении переменной, нужно указать имя массива и его тип, например:

Пример кода:

*int[] a;*

*int a[];*

*int [] a, b, c;*

Правила хорошего кода: использовать 1 и 3 варианты объявления массивов.

В отличие от переменной примитивного типа, например, целочисленной, объявления массива недостаточно для начала работы с ним. Следующим шагом является создание массива с заданием его размера. Создание массива осуществляется с помощью операции new и имеет следующий синтаксис:

<имя массива> = new <тип> [<размер>]

Пример кода:

*a = new int [5];*

Объявление массива можно совместить с его созданием, используя следующий синтаксис:

<тип> <имя массива>[] = new <тип> [<размер>]

Пример кода:

*int array[] = new int [5];*

При создании массива происходит его инициализация, то есть присваивание начальных значений элементам массива. По умолчанию компилятор инициализирует числовые массивы нулевыми значениями, символьные — нулевым символом (символом с кодом 0), логические — значением false.

Имеется возможность инициализировать элементы массива другими значениями, добавив в оператор создания массива список выражений, заключенный в фигурные скобки:

Пример кода:

*int x = 1;  
int y = 2;  
int array1[] = new int [] {3, 11, 4, 6, 5};  
int array2[] = new int [] {3, 11, x, 2\*x, y - x};  
int array3[] = {3, 11, x, 2\*x, y - x};*

Что мы сделаем в этой строке? Создадим массив?

int[] array1;

Нет, мы просто объявим переменную массива, а создаем мы его, когда выделяем память под нужное нам количество элементов объявленного массива:

int[] array1 = new int[3];

или

int[] array1 = {1,2,3};

Все, как только мы выделили кусок памяти, мы можем работать только с этим куском - ни больше, ни меньше нам захватить не удастся, поэтому мы не сможем ни уменьшить, ни увеличить количество элементов массива. То есть размер созданного массива нельзя изменить.

После создания массива его размер сохраняется в свойстве length, которое рекомендуется использовать в различных алгоритмах обработки массивов. Обращение к этому свойству имеет синтаксис:

<имя массива>.length

Пример кода:

*System.out.println(a.length);*

Элементы массива нумеруются, начиная с нулевого значения, и номер элемента называется его индексом. Таким образом, первому элементу массива соответствует значение индекса 0, второму — значение индекса 1, элементу с порядковым номером k — значение индекса k-1. Для доступа к отдельным элементам массива используется следующий синтаксис:

<имя массива>[<выражение целого типа>]

Пример:

*a[0] = 1;*

*a[1] = 2;*

*a[2] = 3;*

*a[3] = 4;*

*a[4] = 5;*

Значение выражения в квадратных скобках рассматривается как индекс элемента массива и поэтому должно иметь значение в диапазоне от 0 до length — 1 включительно.

В случае, если в ходе выполнения программа пытается выйти за границы массива (обратиться к элементу массива с индексом вне диапазона от 0 до размерности массива минус 1), то возникнет ошибка — исключение ArrayIndexOutOfBoundsException.

**Цикл for each**

В тех случаях, когда не требуется производить запись новых значений в элементы массива, можно воспользоваться модификацией цикла for, известной как цикл for each. В этом варианте цикл for имеет следующий синтаксис:

for (<тип массива> <имя переменной>: <имя массива>) <тело цикла>

Пример кода:

*int sum = 0;*

*for (int x: a)*

*sum += x;*

Но нельзя увеличить каждый элемент массива на единицу таким кодом.

Пример кода:

*for (int x: a)*

*x++;*

**Цикл for для работы с массивами**

На практике очень редко используются массивы небольших размеров. А если наш массив имеет размерность 100, наш код просто разрастется до огромных размеров, если мы будем обращаться к каждому элементу массива отдельно.

Поэтому при работе с массивами используется цикл for.

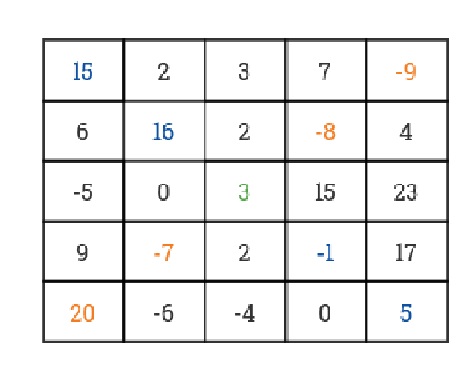
Пример кода:

*int a[] = new int[10];  
for(int i = 0; i < a.length; i++) {  
 a[i] = i \* i;  
 System.out.print(a[i] + " ");  
}*

### **Двумерный массив**

Ранее мы рассматривали одномерные массивы, которые можно представить как цепочку или строку однотипных значений. Но кроме одномерных массивов также бывают и многомерными. Давайте рассмотрим самый простейший случай многомерного массива — двумерный.

Двумерный массив - это массив одномерных массивов. Визуально его можно представить как таблицу.



Матрица — это прямоугольная таблица однотипных элементов или, другими словами, это массив одномерных массивов.

Говоря о квадратной матрице, мы сталкиваемся с понятиями:

- элементы главной диагонали (i=j);

- элементы побочной диагонали (i+j=n−1, n — количество строк или столбцов квадратной матрицы);

- элементы ниже главной диагонали (i>j);

- элементы выше главной диагонали (i<j);

- элементы ниже побочной диагонали (i+j>n−1);

- элементы выше побочной диагонали (i+j<n−1).

По аналогии с одномерными массивами мы сначала должны объявить, а затем инициализировать массив (или сделать это одновременно).

Пример кода:

*int[][] a = new int[3][4];*

*int[] a = new int[] { 0, 1, 2, 3, 4, 5 };*

*int[][] a = { { 0, 1, 2 }, { 3, 4, 5 } };*

Количество квадратных скобок указывает на размерность массива. А числа в скобках - на количество строк и столбцов. И также, используя индексы, мы можем использовать элементы массива в программе:

Первое число в скобках обозначают ряд (строку), а второе число — столбец.

При резервировании памяти под многомерный массив необходимо указать память только для первого измерения. Для остальных измерений память можно выделить отдельно.

Пример кода:

*int[][] array = new int[3][];*

*array[0] = new int[4];*

*array[1] = new int[4];*

*array[2] = new int[4];*

Размер двумерного массива измеряется следующим способом. Длина массива определяется по его первой размерности, то есть вычисляется количество строк.

Пример кода:

*System.out.println("Количество строк массива = " + array.length);*

А если мы хотим узнать количество столбцов в строке? Тогда указываете строку, а затем вычисляете у нее количество столбцов.

*System.out.println("Количество столбцов массива = " + array[0].length);  
System.out.println("Количество столбцов массива = " + array[1].length);*

Обращаться к элементу двумерного массива будем таким образом:

Пример кода:

*array[1][0]=44;  
System.out.println("Значение элемента = " + a[1][0]);*

Для двумерных массивов используются вложенные циклы for. В качестве примера обращения к отдельным элементам двумерного массива рассмотрим циклы заполнения массива случайными числами с последующим выводом полученных значений на экран.

Для этого нам потребуется функция, генерирующая последовательность случайных чисел. Такой функцией является метод random() класса Math.  Метод random() позволяет получить последовательность случайных значений типа double из диапазона [0,1), которые затем можно масштабировать, умножая на соответствующий коэффициент, и преобразовать к целому типу. Например, код:

*int x = (int)(Math.random() \* 10) + 1;*

положит в переменную x случайное число от 1 до 10 включительно.

Создание и заполнение двумерного массива случайными числами.

Пример кода:

*for (int i = 0; i < array.length; i++) {  
 for (int j = 0; j < array[i].length; j++) {  
 array[i][j] = (int)(Math.random() \* 10);  
 System.out.format("Значение элемента [%d] [%d] = %d\n", i, j, array[i][j]);  
 }  
}*

Строки матрицы можно легко переставлять:

Пример кода:

*int[][] array = { { 3, 6, 5, 7 }, { 3, 2, 1, 6 }, { 7, 8, 9, 0 } }; int[] temp = array[0];*

*array[0] = array[1];*

*array[1] = temp;*

В этом фрагменте для перестановки используется буферный массив temp типа int[].

Перестановка строк в рассмотренном примере происходит очень быстро. Потому что меняются местами только ссылки на области памяти, а сами элементы строк при этом не перемещаются.

**Неровные массивы**

Многомерные массивы могут быть также представлены как "зубчатые массивы" или неровные массивы. Ранее мы рассматривали двухмерные массивы, которые имели одинаковое количество строк в каждом столбце, поэтому у нас получалась ровная таблица. Но мы можем каждому элементу в двухмерном массиве присвоить отдельный массив с различным количеством элементов:

Пример кода:

*int[][] array = new int[3][];*

*array[0] = new int[2];*

*array[1] = new int[3];*

*array[2] = new int[5];*

*int[][] matrix2 = { { 6, 5 }, { 3, 2, 1, 6, 7 }, { 7, 8, 9 } };*

Работа с неровными массивами ничем не отличается от работы с матричными массивами.

Пример кода:

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

*for (int j = 0; j < array[i].length; j++) {*

*array[i][j] = (int) (Math.random() \* 10);*

*} }*