Урок 1

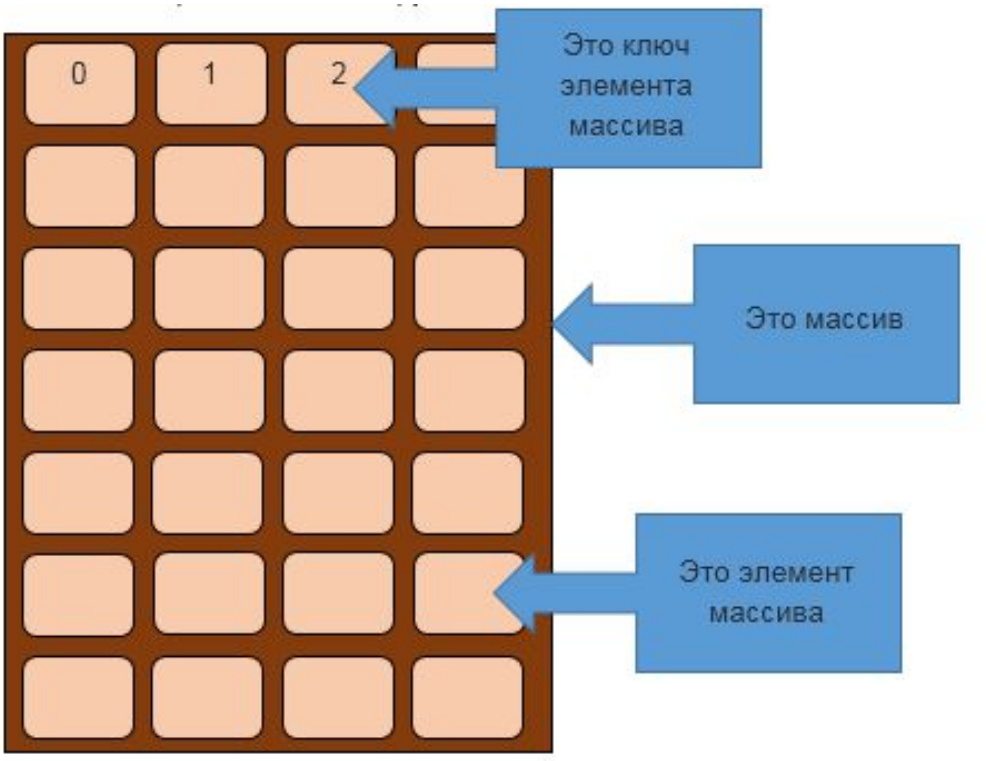
Массивы

Одномерные, двумерные, нерегулярные, многомерные. Синтаксис объявления массива, получение его длины.

© geekbrains.ru

Массивы

Массив — это именованный набор однотипных переменных. Определение можно пояснить через аналогию массива и большого шкафа с множеством ящиков. Сам шкаф — это массив, а ящики — его элементы.



В классическом виде нумерация элементов начинается с нуля. Ключи уникальны и не могут повторяться. Таким образом, массив можно определить как переменную, которая хранит не одно, а множество значений.

Одномерные массивы

Для объявления одномерного массива обычно применяется следующая форма:

```
тип_данных[] имя_массива = new тип_данных[размер_массива];
```

При создании массива сначала объявляется переменная, ссылающаяся на него, затем для него выделяется память, ссылка на которую присваивается переменной массива. Память для массивов в Java динамически распределяется с помощью оператора **new**. В следующей строке кода создается массив типа **int**, состоящий из пяти элементов, а ссылка на него присваивается переменной **arr**:

```
int[] arr = new int[5];
```

В переменной **arr** сохраняется ссылка на область памяти, выделяемой для массива оператором **new**. Ее должно быть достаточно для размещения пяти элементов типа **int**. Доступ к отдельным элементам массива осуществляется с помощью индексов. Индекс обозначает положение элемента в массиве и для

первого равен нулю. Так, если массив **arr** содержит 5 элементов, их индексы находятся в пределах от 0 до 4. Индексирование массива осуществляется по номерам его элементов, заключенным в квадратные скобки. Например, для доступа к первому элементу массива **arr** следует указать **arr[0]**, а к последнему — **arr[4]**. В приведенном ниже примере программы в массиве **arr** сохраняются числа от 0 до 4:

```
public static void main(String args[]) {
    int[] arr = new int[5];
    int i = 0;
    arr[i] = i++;
    arr[i] = i++;
    arr[i] = i++;
    arr[i] = i++;
    arr[i] = i++;
}
```

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]
0	1	2	3	4

Заполнять созданные массивы можно последовательным набором операторов:

```
public static void main(String args[]) {
    int[] nums = new int[4];
    nums[0] = 5;
    nums[1] = 10;
    nums[2] = 15;
    nums[3] = 15;
}
```

В приведенном выше примере массив **nums** заполняется через четыре оператора присваивания. Существует более простой способ решения этой задачи — заполнить массив сразу при его создании:

```
тип_данных[] имя_массива = {v1, v2, v3, ..., vN} ;
```

Здесь **v1-vN** обозначают первоначальные значения, которые присваиваются элементам массива по очереди: слева направо и по порядку индексирования. При этом Java автоматически выделит достаточный объем памяти. Например:

```
public static void main(String args[]) {
    int[] nums = { 5, 10, 15, 20 };
}
```

Границы массива в Java строго соблюдаются: при попытке обратиться к несуществующему элементу произойдет ошибка:

```
public static void main(String args[]) {
    int[] arr = new int[10];
    System.out.println(arr[10]);
}
```

Как только значение переменной **i** достигнет 10, будет сгенерировано исключение **ArrayIndexOutOfBoundsException**, и выполнение программы прекратится.

Распечатать одномерный массив в консоль можно, используя конструкцию вида:

```
System.out.println(Arrays.toString(arr));
```

Двумерные массивы

Среди многомерных массивов наиболее простыми являются двумерные, которые, по сути, представляют ряды одномерных массивов. При работе с двумерными массивами проще их представлять в виде таблицы (см. ниже). Объявим двумерный целочисленный табличный массив **table** размером 10x20:

```
int[][] table = new int[10][20];
```

Чтобы обратиться к элементу массива, надо указывать оба его индекса в соответствующем порядке. Пример для массива **arr[i][j]**:

	j = 0	j = 1	j = 2	j = 3
i = 0	1	2	3	4
i = 1	5	6	7	8
i = 2	9	10	11	12

```
System.out.println(arr[1][3]); // Выведет 8
```

Нерегулярные массивы

Выделяя память под многомерный массив, достаточно указать первый (крайний слева) размер. Память под остальные размеры массива можно выделять по отдельности:

```
int[][] table = new int[3][];
table[0] = new int[1];
table[1] = new int[5];
table[2] = new int[3];
```

Поскольку многомерный массив является «массивом массивов», можно установить разную длину по каждому индексу. Такие массивы могут повысить эффективность программы и снизить затраты памяти — например, если требуется создать большой двумерный массив, в котором используются не все элементы.

Многомерные массивы

В Java допускаются n-мерные массивы. Форма их объявления:

```
тип_данных[][]...[] имя_массива = new тип_данных[размер1][размер2]...[размерN];
```

Объявление трехмерного целочисленного массива размером 2x3x4:

```
int[][][] mdarr = new int[2][3][4];
```

Для многомерного массива можно заключать инициализирующую последовательность для каждого его размера в отдельные фигурные скобки.

Альтернативный синтаксис объявления массивов

Помимо общей формы для объявления массива можно пользоваться такой:

```
тип_данных имя_массива[];
```

Здесь два объявления массивов равнозначны:

```
public static void main(String[] args) {
    int arr[] = new int[3];
    int[] arr2 = new int[3];
}
```

Получение длины массива

При работе с массивами можно программно узнать их размер, воспользовавшись записью ***имя_массива.length***. Это удобно, чтобы пройти циклом **for** по всему массиву:

```
public static void main(String[] args) {
    int[] arr = {2, 4, 5, 1, 2, 3, 4, 5};
    System.out.println("arr.length: " + arr.length);
    for (int i = 0; i < arr.length; i++) {
        System.out.print(arr[i] + " ");
    }
}
Результат:
arr.length: 8
2 4 5 1 2 3 4 5
```