**Что такое массив в Java?**

В статье ["Переменные в Java.Создание переменной"](https://vertex-academy.com/tutorials/ru/sozdanie-peremennyx-i-tipy-peremenny/) мы говорили о том, что переменная - **это некоторый контейнер**, в котором может храниться значение для дальнейшего использования в программе.

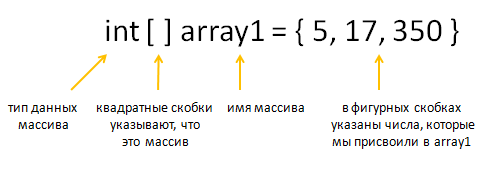
Так вот:



* обычная переменная может содержать **только 1 значение**.
* массив - это простая переменная, которая**может содержать не одно, а сразу несколько значений**, в отличие от обычной переменной, содержащей единственное значение. Массив - это такой себе "магический" контейнер, в который мы можем положить сразу несколько значений ("коробочек").

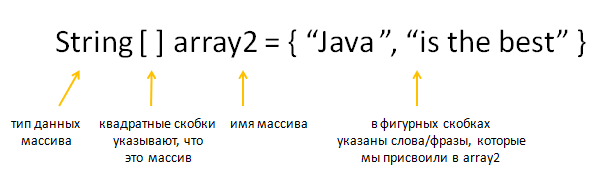
Ниже приводим Вам примеры массивов.

**Пример №1**



Как видите, мы создали целочисленный массив, который назвали array1 и присвоили три числа: 5, 17, 350

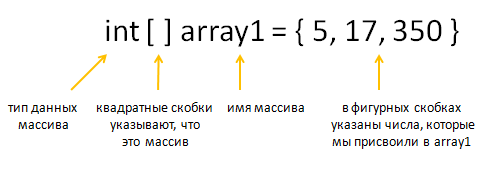
**Пример №2**



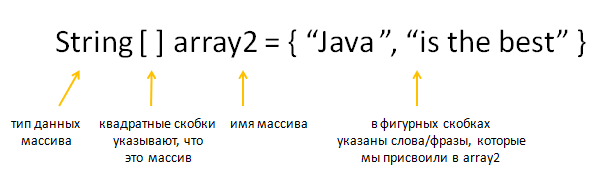
Как видите, мы создали строковый массив, который назвали array2 и присвоили слово "Java " и фразу "is the best". Причем мы их написали в двойных кавычках, потому что это массив типа String.

###### ****Любой массив имеет длину****

Например, длина этого массива = 3. Потому что в массиве 3 числа: 5, 17, 350

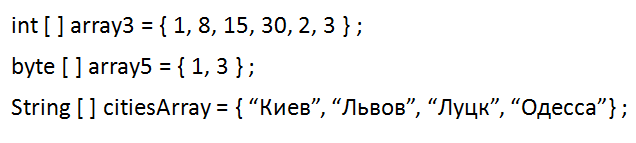


А длина этого массива = 2. Потому что в массиве 2 значения: "Java " и "is the best"



Если сказать проще, то длина массива - это количество элементов, которые в нем хранятся (количество маленьких "коробочек" в большой).

**Предлагаем теперь Вам ответить какая длина у этих 3 массивов?**



**Правильные ответы:**

massiv3_vertex-academy

Длина этого массива 6. Потому что в этом массиве 6 чисел: 1, 8, 15, 30, 2, 3

massiv4_vertex-academy

Длина этого массива 2. Потому что в массиве 2 числа: 1 и 3

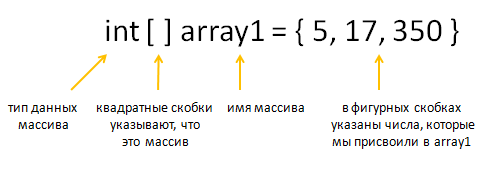
massiv5_vertex-academy

Длина этого массива 4. Потому что в массиве 4 слова: "Киев", "Львов", "Луцк", "Одесса".

###### ****А как сделать так, чтоб автоматически получать длину массива?****

Подсчитывать вручную количество элементов в массиве - это хорошо, но что, если элементов очень много... Вряд ли у Вас хватит терпения подсчитать 3 000 элементов. Если использовать свойство массива **length**, длина массива будет подсчитана автоматически.

Например, у нас есть вот такой массив. 



Пишем **int k = array1.length;** .

В результате создастся переменная k, в которой будет лежать значение 3.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | class Test {    public static void main(String[] args) {  int [] array1 = {5,17,350};  int k = array1.length;  System.out.println(k);  }  } |

**В любом массиве значения элементов сохраняются под порядковыми номерами (индексами)**

Причем нумерация начинается **с "нуля"**. Предлагаем рассмотреть на примерах.

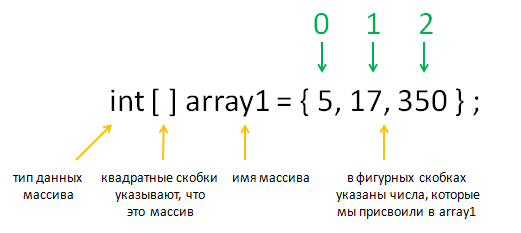
**Пример №1**

В данном массиве:

5 - это элемент с порядковым номером 0

17 - это элемент с порядковым номером 1

350 - это элемент с порядковым номером 2



**Пример №2**

В данном массиве:

"Java " - это элемент с порядковым номером 0

"is the best" - это элемент с порядковым номером 1



Если Вы очень любите сладости, то можете представить массив в виде макаронов.



То есть индекс элемента - это его номер по порядку "минус" 1.

Как видно, длина массива из макаронов =3. И применяется та же логика к порядковым номерам (индексам) в массиве.

**Давайте разберемся зачем в массиве значения элементов сохраняются под порядковыми номерами (индексами)?**

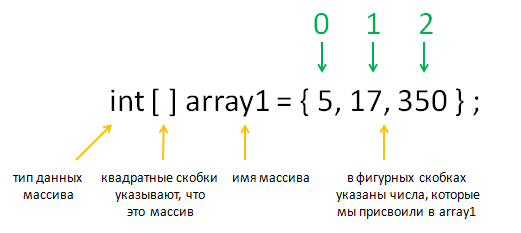
А ответ очень и очень простой. **Чтобы можно было обращаться к конкретным элементам массива по индексу**. Поскольку из того, что мы в большую коробку хаотично накидаем другие коробки, толку мало. Нам нужно знать, где лежит маленькая коробка с тем, что нам нужно. По этому все, что мы кладем в большую коробку, имеет свой номер.

Для того, чтобы получить то, что лежит в маленькой коробочке, нам нужно обратится к ней по индексу. Для этого пишем название переменной (большая коробка), после чего без пробелов в квадратных скобках индекс коробки.

массив[номер]

Давайте рассмотрим это на практике.

1. **Как мы можем обратиться к элементу массива array1 под индексом 0?  Ответ: array1[0]**
2. **Как мы можем обратиться к элементу массива array1 под индексом 1? Ответ: array1[1]**
3. **Как мы можем обратиться к элементу массива array1 под индексом 2? Ответ: array1[2]**



**А как создать пустой массив?**



Во всех предыдущих примерах мы создавали массив и **сразу**инициализировали его какими-то значениями. Однако **есть возможность сначала создать массив нужного размера, а позже наполнять его значениями (если нам при создании не известно, какие значения там будут находиться).**Например, врач говорит женщине, что у нее будет двойня, но какого пола будут дети пока не известно. Вот Вам массив на 2 элемента типа "ребенок", но значения пока не известны.

Чтобы создать такой массив, нужно воспользоваться ключевым словом**new.**Теперь создание массива будет выглядеть так:

тип [] название = **new**тип[размер];

Запомните, что тип до "равно" и после, если мы говорим о массиве из примитивных типов, должны быть **одинаковыми!**

В будущем мы увидим, что это правило может нарушаться, но это уже другая история.

В программе это выглядело бы вот так:

class Test {

public static void main(String[] args) {

int [] array1 = new int[3];

}

}

Давайте разберем эту строчку кода:

|  |  |
| --- | --- |
| 1 | int[] array1=new int[3]; |

С помощью**int[] array1** мы указали, что будет целочисленный массив с именем array1

С помощью **= new int[3];** мы создали пустой массив из 3 элементов.

**Обратите внимание: несмотря на то, что массив array1 считает пустым, на самом деле в нем все-таки есть значения по умолчанию...** **Во время создания пустого массива элементам присваиваются значения в зависимости от типов данных:**

* для int - 0
* для float, double - 0.0
* для String - значение null
* для char - \0
* для boolean - значение false

Поскольку у нас в приведенном примере массив array1 целочисленный, - это значит, что **все 3 элемента в нем равны нулю**. То есть int [] array1 = new int [3]; это то же самое что и int [] array1 = {0,0,0}

**А как наполнить такой массив значениями?**

Попробуйте запустить этот код на своем компьютере:

class Test {

public static void main(String[] args) {

int[] array1 = new int[3];

array1[0]=5;

array1[1]=17;

array1[2]=350;

System.out.println(array1[0]);

System.out.println(array1[1]);

System.out.println(array1[2]);

}

}

С помощью вот этих строчек мы **обращаемся к каждому элементу массива** и наполняем его нужным значением:

|  |  |
| --- | --- |
| 1  2  3 | array1[0]=5;  array1[1]=17;  array1[2]=350; |

То есть, если array1[0] стоит справа от "равно", то мы получаем его значение, если слева - присваиваем ему значение.

А эти строчки кода мы добавили, чтоб Вы могли увидеть, что таки да, каждому элементу массива были присвоены значения 5, 17, 350 соответственно.

|  |  |
| --- | --- |
| 1  2  3 | System.out.println(array1[0]);  System.out.println(array1[1]);  System.out.println(array1[2]); |

В консоли Вы увидите:

5

17

350

###### ****Цикл - ваш лучший помощник при работе с массивами****

Вот мы заполнили массив значениями и вывели их на экран. Но, честно говоря, это было сделано не самым удобным способом. **Цикл** облегчил бы нам работу с массивами ( а особенно с огромными, элементов в тысячу). Вот, посмотрите:

class Test {

public static void main(String[] args) {

int[] array1 = new int[3];

array1[0]=5;

array1[1]=17;

array1[2]=350;

              for( int i = 0; i < array1.length; i++){

System.out.println(array1[i]);

              }

}

}

Немного короче, правда?

Мы добавили цикл **for**,  который поочередно обращается к каждому элементу массива **array1** по индексу и выводит их на экран.

Циклобращается к элементам от **0** (как вы помните, первый элемент массива имеет индекс 0) до **array1.length** (как вы помните, свойство, которое считает количество элементов массива за вас).

Но обратите внимание: **array1.length** равняется **3**, а индексы наших элементов - **0, 1, и 2**. То-есть array1.length на 1 больше, чем индекс последнего элемента. Именно из-за этого мы пишем в условии "**i < array1.length**", а не "**i <= array1.length**".

Вообще, циклы можно использовать для двух основных целей: чтобы **заполнять массивы значениями**, или чтобы **выводить их на экран**.

Мы уже видели, как можно вывести на экран содержимое массива - просто нужно создать цикл **от 0 до array.length**. Например:

public class Test{

public static void main(String[] args) {

boolean[] arr = new boolean[1000];

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

    System.out.println(arr[i]);

                }

}

}

Как вы помните, по умолчанию элементы **boolean** хранят **false**,  поэтому в результате выполнения этого кода мы видим на экране просто 1000 **false**.

Но если бы у нас был массив со значениями, имеющими какую-то логику (например, каждое значение в два раза больше предыдущего), то **его тоже можно было заполнить циклом**.

Возьмем наш массив из прошлого примера,и сделаем так, чтобы он выводил не **false**, a **true**:

public class Test{

public static void main(String[] args) {

boolean[] arr = new boolean[1000];

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

    arr[i] = true;

        }

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

    System.out.println(arr[i]);

        }

}

}

Тут мы использовали тот же цикл **от 0 до arr.length**.

А теперь давайте сделаем так, чтобы выводилось:

**true**

**false**

**true**

**...**

Для этого сделаем следующее:

public class Test{

public static void main(String[] args) {

boolean[] arr = new boolean[1000];

        for( int i = 0; i< arr.length; i=i+2){

    arr[i] = true;

        }

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

    System.out.println(arr[i]);

        }

}

}

Тут мы заполнили массив "через один" значениями **true**, потому что использовали не **i++** как обычно, а **i=i+2**. То-есть мы "перепрыгивали", обращаясь только к каждому второму элементу.

На самом деле, фантазия в использовании циклов ограничивается только вашей задачей. Например, что делать, если нужно **посчитать сумму элементов массива**? Подойдет такой код:

public class HelloWorld{

public static void main(String[] args) {

int[] array1 = new int[3];

array1[0]=5;

array1[1]=17;

array1[2]=350;

        int sum = 0;

        for( int i = 0; i < array1.length; i++){

  sum += array1[i];

              }

              for( int i = 0; i < array1.length; i++){

System.out.println(array1[i]);

              }

      System.out.println("Сумма - "+ sum);

}

}

###### ****Еще один важный момент - размер созданного массива нельзя изменить****

Давайте вместе посмотрим на вот эту строчку кода:

|  |  |
| --- | --- |
| 1 | int[] array1;  Как Вы думаете: "Этой строчкой кода мы создали массив?" Ответ: нет. Как так? Ведь вроде бы мы:   * указали с помощью [] что это массив * указали с помощью int, что массив целочисленный * указали, что название массива array1   А массив не создан??? Да, массив не создан.  **Массив создается, когда под него выделяется память. А выделяется память под массив только в момент выполнения следующей ступеньки:** |

|  |  |
| --- | --- |
| 1 | int[] array1 = new int[3]; |

то есть когда мы дописали **= new int [3];**

**или этой:**

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

Вы не можете добавлять элементы в массив свыше указанного количества, а незаполненные вами ячейки просто будут хранить значение "0". В Java есть другие инструменты, которые позволяют делать "безразмерные" списки элементом - с ними

**Подытожим:**

1. Массив - это переменная, в которую можно положить не одно, а сразу несколько значений.

2. Все элементы массива имеют одинаковый тип.

3. При создании пустого массива элементам присваиваются значения в зависимости от типа данных массива:

* для int - 0
* для float, double - 0.0
* для String - значение null
* для char - \0
* для boolean - значение false

4. Размер массива нельзя изменить после его создания.

# Двумерные массивы в Java

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

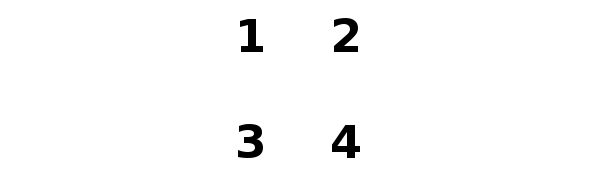
* создать одномерный массив
* заполнить его значениями
* и вывести в консоль

**Например:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | 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, напоминаем:



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 приводим **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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 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**

|  |  |
| --- | --- |
| 4  5  6  7  8  9  10  11 | 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