Статья с ссылками находится - https://javarush.com/groups/posts/massivy-java

**Что такое массив?**

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, массив из элементов созданного нами класса Dog будет содержать только объекты Dog. То есть в Java мы не можем поместить в первую ячейку массива целое число, во вторую String, а в третью — “собаку”.

**Объявление массива**

**Как объявить массив?**

Как и любую переменную, массив в Java нужно объявить. Сделать это можно одним из двух способов. Они равноправны, но первый из них лучше соответствует стилю Java. Второй же — наследие языка Си (многие Си-программисты переходили на Java, и для их удобства был оставлен и альтернативный способ). В таблице приведены оба способа объявления массива в Java:

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Объявление массива, Java-синтаксис** | **Примеры** | **Комментарий** |
| 1. | dataType[] arrayName; | **int**[] myArray;  Object[]  arrayOfObjects; | Желательно объявлять массив именно таким способом, это Java-стиль |
| 2. | dataType arrayName[]; | **int** myArray[];  Object  arrayOfObjects[]; | Унаследованный от С/С++ способ объявления массивов, который работает и в Java |

В обоих случаях **dataType** — тип переменных в массиве. В примерах мы объявили два массива. В одном будут храниться целые числа типа int, в другом — объекты типа Object. Таким образом при объявлении массива у него появляется имя и тип (тип переменных массива). **arrayName** — это имя массива.

**Создание массива**

**Как создать массив?**

Как и любой другой объект, создать массив Java, то есть зарезервировать под него место в памяти, можно с помощью оператора **new**. Делается это так:

**new** typeOfArray [length];

Где **typeOfArray** — это тип массива, а **length** — его длина (то есть, количество ячеек), выраженная в целых числах (int). Однако здесь мы только выделили память под массив, но не связали созданный массив ни с какой объявленной ранее переменной. Обычно массив сначала объявляют, а потом создают, например:

**int**[] myArray; // объявление массива

myArray = **new** **int**[10]; // создание, то есть, выделение памяти для массива на 10 элементов типа int

Здесь мы объявили массив целых чисел по имени myArray, а затем сообщили, что он состоит из 10 ячеек (в каждой из которых будет храниться какое-то целое число). Однако гораздо чаще массив создают сразу после объявления с помощью такого сокращённого синтаксиса:

**int**[] myArray = **new** **int**[10]; // объявление и выделение памяти “в одном флаконе”

Обратите внимание: После создания массива с помощью **new**, в его ячейках записаны значения по умолчанию. Для численных типов (как в нашем примере) это будет 0, для boolean — false, для ссылочных типов — null. Таким образом после операции

**int**[] myArray = **new** **int**[10];

мы получаем массив из десяти целых чисел, и, пока это не изменится в ходе программы, в каждой ячейке записан 0.

|  |
| --- |
| Больше информации о массивах есть в статье “[Кое-что о массивах](https://javarush.com/groups/posts/1932-koe-chto-o-massivakh)” |

**Длина массива в Java**

Как мы уже говорили выше, длина массива — это количество элементов, под которое рассчитан массив. **Длину массива нельзя изменить после его создания.** Обратите внимание: в Java элементы массива нумеруются с нуля. То есть, если у нас есть массив на 10 элементов, то первый элемент массива будет иметь индекс 0, а последний — 9. Получить доступ к длине массива можно с помощью переменной **length**. Пример:

**int**[] myArray = **new** **int**[10]; // создали массив целых чисел на 10 элементов и присвоили ему имя myArray

System.out.println(myArray.length); // вывели в консоль длину массива, то есть количество элементов, которые мы можем поместить в массив

Вывод программы:

10

**Инициализация массива и доступ к его элементам**

Как создать массив в Java уже понятно. После этой процедуры мы получаем не пустой массив, а массив, заполненный значениями по умолчанию. Например, в случае int это будут 0, а если у нас массив с данными ссылочного типа, то по умолчанию в каждой ячейке записаны null. Получаем доступ к элементу массива (то есть записываем в него значение или выводим его на экран или проделываем с ним какую-либо операцию) мы по его индексу. Инициализация массива — это заполнение его конкретными данными (не по умолчанию). **Пример:** давайте создадим массив из 4 сезонов и заполним его строковыми значениями — названиями этих сезонов.

String[] seasons = **new** String[4]; /\* объявили и создали массив. Java выделила память под массив из 4 строк, и сейчас в каждой ячейке записано значение null (поскольку строка — ссылочный тип)\*/

seasons[0] = "Winter"; /\* в первую ячейку, то есть, в ячейку с нулевым номером мы записали строку Winter. Тут мы получаем доступ к нулевому элементу массива и записываем туда конкретное значение \*/

seasons[1] = "Spring"; // проделываем ту же процедуру с ячейкой номер 1 (второй)

seasons[2] = "Summer"; // ...номер 2

seasons[3] = "Autumn"; // и с последней, номер 3

Теперь во всех четырёх ячейках нашего массива записаны названия сезонов. Инициализацию также можно провести по-другому, совместив с инициализацией и объявлением:

String[] seasons = **new** String[] {"Winter", "Spring", "Summer", "Autumn"};

Более того, оператор new можно опустить:

String[] seasons = {"Winter", "Spring", "Summer", "Autumn"};

**Как вывести массив в Java на экран?**

Вывести элементы массива на экран (то есть, в консоль) можно, например, с помощью цикла for. Ещё один, более короткий способ вывода массива на экран будет рассмотрен в пункте “Полезные методы для работы с массивами", ниже. А пока рассмотрим пример с циклическим выводом массива:

String[] seasons = **new** String[] {"Winter", "Spring", "Summer", "Autumn"};

**for** (**int** i = 0; i < 4; i++) {

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

}

В результате программа выведет следующий результат:

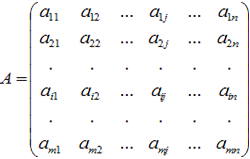
Winter

Spring

Summer

Autumn

**Одномерные и многомерные Java массивы**

А что, если мы захотим создать не массив чисел, массив строк или массив каких-то объектов, а массив массивов? Java позволяет это сделать. Уже привычный нам массив **int[] myArray = new int[8]** — так называемый одномерный массив. А массив массивов называется двумерным. Он похож на таблицу, у которой есть номер строки и номер столбца. Или, если вы учили начала линейной алгебры, — на матрицу. Для чего нужны такие массивы? В частности, для программирования тех же матриц и таблиц, а также объектов, напоминающих их по структуре. Например, игровое поле для шахмат можно задать массивом 8х8. Многомерный массив объявляется и создается следующим образом:

**int**[][] myTwoDimentionalArray = **new** **int** [8][8];

В этом массиве ровно 64 элемента: myTwoDimentionalArray[0][0], myTwoDimentionalArray[0][1], myTwoDimentionalArray[1][0], myTwoDimentionalArray[1][1] и так далее вплоть до myTwoDimentionalArray[7][7]. Так что если мы с его помощью представим шахматную доску, то клетку А1 будет представлять myTwoDimentionalArray[0][0], а E2 — myTwoDimentionalArray[4][1]. Где два, там и три. В Java можно задать массив массивов… массив массивов массивов и так далее. Правда, трёхмерные и более массивы используются очень редко. Тем не менее, с помощью трёхмерного массива можно запрограммировать, например, кубик Рубика.

|  |
| --- |
| **Что еще почитать** |
| [Многомерные массивы](https://javarush.com/groups/posts/mnogomernye-massivy) |

**Полезные методы для работы с массивами**

Для работы с массивами в Java есть класс **java.util.Arrays** (arrays на английском и означает “массивы”). В целом с массивами чаще всего проделывают следующие операции: заполнение элементами (инициализация), извлечение элемента (по номеру), сортировка и поиск. Поиск и сортировка массивов — тема отдельная. С одной стороны очень полезно потренироваться и написать несколько алгоритмов поиска и сортировки самостоятельно. С другой стороны, все лучшие способы уже написаны и включены в библиотеки Java, и ими можно законно пользоваться.

|  |
| --- |
| **Статьи на поиск и сортировку:**  [Алгоритмы сортировки в теории и на практике](https://javarush.com/groups/posts/1997-algoritmih-sortirovki-v-teorii-i-na-praktike)  [Реализация пузырьковой сортировки на Java](https://javarush.com/groups/posts/634-realizacija-puzihrjhkovoy-sortirovki-na-java)  **Сортировка и поиск в курсе CS50:**  [Видеолекция 7 (Week 3)](https://javarush.com/quests/lectures/questharvardcs50.level03.lecture01)  [Алгоритмы сортировки. Пузырьковая сортировка](https://javarush.com/quests/lectures/questharvardcs50.level03.lecture09)  [Алгоритмы сортировки. Сортировка вставками](https://javarush.com/quests/lectures/questharvardcs50.level03.lecture10)  [Алгоритмы сортировки. Сортировка слиянием](https://javarush.com/quests/lectures/questharvardcs50.level03.lecture11) |

Вот три полезных метода этого класса

**Сортировка массива**

Метод void sort(int[] myArray, int fromIndex, int toIndex) сортирует массив целых чисел или его подмассив по возрастанию.

**Поиск в массиве нужного элемента**

int binarySearch(int[] myArray, int fromIndex, int toIndex, int key). Этот метод ищет элемент key в **уже отсортированном массиве** myArray или подмассиве, начиная с fromIndex и до toIndex. Если элемент найден, метод возвращает его индекс, если нет - (-fromIndex)-1.

**Преобразование массива к строке**

Метод String toString(int[] myArray) преобразовывает массив к строке. Дело в том, что в Java массивы не переопределяют toString(). Это значит, что если вы попытаетесь вывести целый массив (а не по элементам, как в пункте “[Вывод массива на экран](https://javarush.com/groups/posts/massivy-java#%D0%9A%D0%B0%D0%BA-%D0%B2%D1%8B%D0%B2%D0%B5%D1%81%D1%82%D0%B8-%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2-%D0%B2-Java-%D0%BD%D0%B0-%D1%8D%D0%BA%D1%80%D0%B0%D0%BD)”) на экран непосредственно (System.out.println(myArray)), вы получите имя класса и шестнадцатеричный хэш-код массива (это определено Object.toString()). Если вы — новичок, вам, возможно, непонятно пояснение к методу toString. На первом этапе это и не нужно, зато с помощью этого метода упрощается вывод массива. Java позволяет легко выводить массив на экран без использования цикла. Об этом — в примере ниже.

**Пример на sort, binarySearch и toString**

Давайте создадим массив целых чисел, выведем его на экран с помощью toString, отсортируем с помощью метода sort и найдём в нём какое-то число.

**class** Main {

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

**int**[] array = {1, 5, 4, 3, 7}; //объявляем и инициализируем массив

System.out.println(array);//пытаемся вывести наш массив на экран без метода toString - получаем 16-ричное число

System.out.println(Arrays.toString(array));//печатаем массив "правильно"

Arrays.sort(array, 0, 4); //сортируем весь массив от нулевого до четвёртого члена

System.out.println(Arrays.toString(array));//выводим отсортированный массив на экран

**int** key = Arrays.binarySearch(array, 5); // ищем key - число 5 в отсортированном массиве.

//метод binarySearch выдаст индекс элемента остортированного массива, в котором "спрятано" искомое число

System.out.println(key);//распечатываем индекс искомого числа

System.out.println(Arrays.binarySearch(array, 0));//а теперь попробуем найти число, которого в массиве нет,

// и сразу же выведем результат на экран

}

}

Вывод программы:

[I@1540e19d

[1, 5, 4, 3, 7]

[1, 3, 4, 5, 7]

3

-1

В первой строке — попытка вывода на экран массива без toString, во второй — вывод массива посредством toString, в третьей выведен отсортированный массив, в четвёртой — индекс искомого числа 5 в отсортированном массиве (помните, что считаем с нуля, поэтому четвёртый элемент массива имеет индекс 3). В пятой строке видим значение **-1**. Такого индекса у массива не бывает. Вывод сигнализирует о том, что искомого элемента (в данном случае, 0) в массиве нет.

|  |
| --- |
| **Больше о методах класса Array**  [Класс Arrays и его использование](https://javarush.com/groups/posts/1933-klass-arrays-i-ego-ispoljhzovanie) — в статье описаны некоторые методы класса Array |

**Главное о массивах**

* Главные характеристики массива: тип помещённых в него данных, имя и длина.  
  Последнее решается при инициализации (выделении памяти под массив), первые два параметра определяются при объявлении массива.
* Размер массива (количество ячеек) нужно определять в int
* Изменить длину массива после его создания нельзя.
* Доступ к элементу массива можно получить по его индексу.
* В массивах, как и везде в Java, элементы нумеруются с нуля.
* После процедуры создания массива он наполнен значениями по умолчанию.
* Массивы в языке Java устроены не так, как в C++. Они почти совпадают с указателями на динамические массивы.