1.5. Массивы, часть 1

> Привет!
>
> На связи шпаргалка урока 1.5. Массивы часть 1. Здесь вы найдете материалы урока и дополнительную информацию по теме массивов, которая поможет решить домашнее задание.
>
> **Время прочтения:** 7 минут.

💡 **Массив** — упорядоченная структура данных фиксированного размера.

Массив позволяет создать последовательность данных определенного типа заранее известной длины.

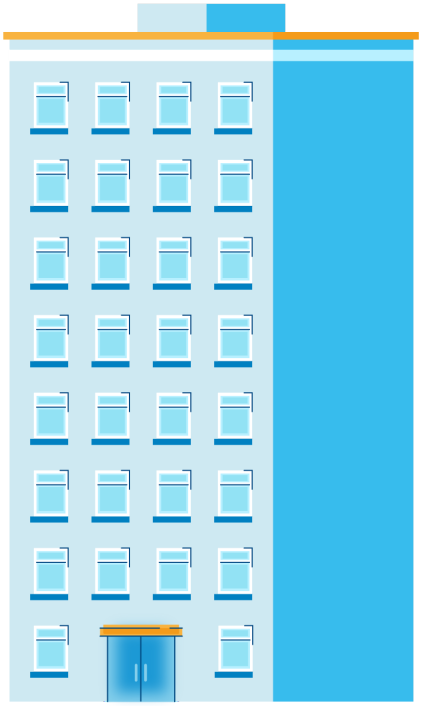
Массив можно сравнить с многоквартирным домом.

В начале строительства мы имеем план, где указаны количество этажей, квартир и размеры самих квартир.

Когда дом сдан, мы уже не можем достроить или отстроить квартиры.

Если представить, что весь наш дом имеет одинаковые квартиры (допустим, все однокомнатные по 50 квадратных метров), это будет пример массива в реальной жизни.

Мы можем влиять на тип данных и размер только на стадии создания массива (т. е. объявления переменной и его инициализации).



Возвращаемся к Java.

Массив может хранить в себе данные исключительно того типа, которым он инициализирован.

Если мы при создании сказали, что это массив типа int, значит в него нельзя будет положить переменную типа boolean, char и даже long.

Когда мы говорим массиву, что он должен хранить в себе 5 значений типа int, Java создает в памяти пять «коробок», которые строго по размеру подходят типу int. Long туда просто не влезет.

Создание массива

Чтобы создать массив, нам нужно написать следующую строку:

```
int[] arr;
```

В коде выше мы создаем переменную типа int[] и присваиваем ей имя arr. Квадратные скобки [] после типа означают, что это не просто переменная этого типа, а именно массив переменных этого типа.

Теперь нам нужно инициализировать нашу переменную, присвоив ей реальный массив, так как в данный момент в переменной arr нет ничего.

Если мы сейчас попробуем ее распечатать, увидим, что печатается значение null.

null — пустота. Т. е. мы заказали в интернет-магазине корзину для яблок, упаковка для корзины нам пришла, а самой корзины внутри нет. Примерно так же с null.

Мы создали переменную определенного типа (упаковку для корзины), а саму корзину туда не положили.

Следовательно, обратиться к упаковке и сказать «Дай мне корзину» мы не можем, ведь ее еще не существует.

Это приведет к падению приложения.

Это значит, что переменная пока не инициализирована, пустая, и работать с ней нельзя.

Инициализируется массив следующим кодом:

```
arr = new int[10];
```

Таким образом, чтобы создать новый массив, нужно написать код в две строчки:

```
int[] arr;
arr = new int[10];
```

Как мы видим, теперь в нашей эксплуатации появляется новое ключевое слово new.

Оно необходимо, так как в отличие от примитивных типов, что мы уже изучили, массив является **объектом**.

В Java объекты разделяются на примитивы и ссылочные типы данных.

С примитивами мы уже познакомились. Это просто число или просто символ.

В случае с объектами речь идет о более сложной структуре, которая внутри себя может иметь не только одно значение, но еще и методы по работе с этим значением, описания этого значения и т. д. В дальнейшем вы столкнетесь с одним из «описаний» значения массива — переменной length, которая хранит в себе размер массива.

И если примитивы в случае отсутствия инициализации будут хранить в себе значение «по умолчанию», это не приведет к падению программы при обращении к неинициализированной переменной. А объекты хранят в себе null. Обращение к null (пустоте) как к объекту приводит к падению приложения.

Все объекты в Java, кроме строк (но об этом узнаете дальше), создаются с помощью ключевого слово new.

Ключевое слово new говорит Java, что нужно в памяти нашего приложения выделить область и поместить туда объект. В случае массива нужно выделить память в размере, который рассчитывается путем умножения количества ячеек на размерность типа.

В нашем примере Java нужно выделить 10 ячеек по 32 бита (тип int занимает 32 бита). Безусловно, массив должен иметь дополнительную служебную ячейку, которая хранит данные о массиве, но об этом вы узнаете в дальнейшем.

Создать и инициализировать переменную массива мы можем в одну строку.

Делить на две, как в примерах выше, не требуется:

```
int[] arr = new int[10];
```

В квадратных скобках, как вы могли заметить ранее, указывается размер массива.

Типы данных массива

Типы данных, которые мы можем хранить в массиве, не ограничиваются int.

Мы можем создать массив для данных любого типа.

Если нам нужно создать массивы других типов, следует int в примере выше заменить на необходимый тип.

```
char[] chars = new char[15];
double[] doubleArr = new double[10];
long[] numbers = new long[82];
boolean[] bools = new boolean[2];
```

Работа с массивами

Возвращаемся к нашему ранее созданному массиву arr.

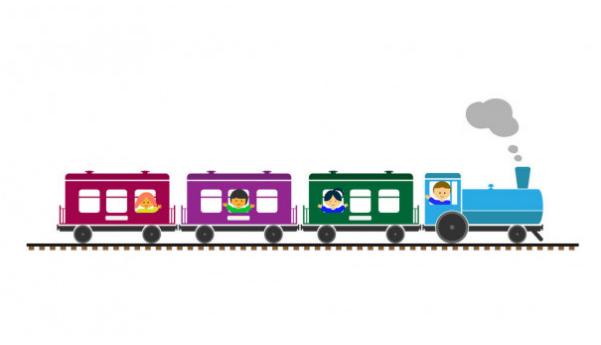
Как мы помним, arr является массивом типа int, емкость которого составляет 10 элементов. Это значит, что мы можем положить в массив 10 значений его типа, в данном случае int.

Чтобы яснее представлять себе работу с массивом, вообразите поезд.

В нашем случае это поезд из 10 вагонов.

Вагоны идут в строгом порядке, от начала до конца.

У каждого вагона есть номер от 1 до 10. Верно?



Верно, но не в нашей ситуации. Во многих языках программирования нумерация ячеек, в отличие от реальных предметов вроде поезда, начинается не с 1, а с 0.

Т. е. первый элемент будет лежать в ячейке под номером 0.

И ячейки, следовательно, будут иметь номера не от 1 до 10, а от 0 до 9 включительно.

Попробуем что-то положить в нулевую ячейку.

Для этого нам нужно написать следующее:

```
int[] arr = new int[10]; // Создали массив переменных типа int на 10 элементов
arr[0] = 5; // Положили в ячейку 0 значение 5
```

Нам никто не мешает положить в массив и значение из ранее созданной переменной.

```
int i = 10; // Создали переменную типа int со значением 10
arr[1] = i; // Положили значение переменной i в первую ячейку
```

Чтобы получить элемент из ячейки по известному нам номеру, нам требуется вызвать похожий на предыдущий пример код.

```
arr[5] = 10; // Положить в пятую ячейку значение 10
int i = arr[5]; // Создать переменную int i и инициализировать ее значение из пятой ячейки массива arr
```

Переменная length

Но что, если нам нужно положить какой-то символ в последнюю ячейку?

Неужели нам нужно для этого помнить (или искать в коде) значение размера, которое мы указывали при создании массива?

Нет. Так как массив является объектом (об этом упоминалось ранее), он может иметь в своей структуре дополнительные переменные.

На уроке по объектам вы научитесь сами создавать объекты и поймете, как это работает «под капотом».

Пока просто следует запомнить, что у любого массива внутри есть переменная length, которая хранит в себе размер массива. Это одна из тех характеристик или описаний объекта, что упоминались ранее.

Для того, чтобы узнать размер массива, нам необходимо написать следующее:

```
int[] arr = new int[10]; // Создали массив на 10 элементов
int arrSize = arr.length; // Присвоили переменной arrSize значение длины массива (10)
```

Обратите внимание

Так как массивы начинаются с нуля, значение arr.length всегда будет иметь индекс последнего элемента + 1. Т. е. последний элемент лежит по индексу (номеру) в ячейке 9, а length хранит в себе 10.

Получить доступ к последней ячейке всегда можно следующим способом:

```
int i = arr[arr.length - 1];
```

> Шпаргалка урока в PDF-формате:
>
> https://s3-us-west-2.amazonaws.com/secure.notion-static.com/1b475748-cc29-4308-b28c-32c3aa47f69a/1.5.__-_1.pdf

> Привет!
>
> На связи шпаргалка урока 1.5. Массивы часть 1. Здесь вы найдете материалы урока и дополнительную информацию по теме массивов,

> которая поможет решить домашнее задание.
>
> **Время прочтения:** 7 минут.

Массив — упорядоченная структура данных фиксированного размера.

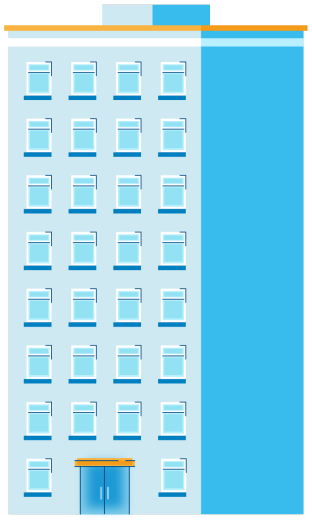
Массив позволяет создать последовательность данных определенного типа заранее известной длины.

Массив можно сравнить с многоквартирным домом.

В начале строительства мы имеем план, где указаны количество этажей, квартир и размеры самих квартир.

Когда дом сдан, мы уже не можем достроить или отстроить квартиры.

Если представить, что весь наш дом имеет одинаковые квартиры (допустим, все однокомнатные по 50 квадратных метров), это будет пример массива в реальной жизни.



Возвращаемся к Java.

Мы можем влиять на тип данных и размер только на стадии создания массива (т. е. объявления переменной и его инициализации).

Массив может хранить в себе данные исключительно того типа, которым он инициализирован.

Если мы при создании сказали, что это массив типа int, значит в него нельзя будет положить переменную типа boolean, char и даже long.

Когда мы говорим массиву, что он должен хранить в себе 5 значений типа int, Java создает в памяти пять «коробок», которые строго по размеру подходят типу int. Long туда просто не влезет.

Создание массива

Чтобы создать массив, нам нужно написать следующую строку:

```
int[] arr;
```

В коде выше мы создаем переменную типа int[] и присваиваем ей имя arr. Квадратные скобки [] после типа означают, что это не просто переменная этого типа, а именно массив переменных этого типа.

Теперь нам нужно инициализировать нашу переменную, присвоив ей реальный массив, так как в данный момент в переменной arr нет ничего.

Если мы сейчас попробуем ее распечатать, увидим, что печатается значение null.

null — пустота. Т. е. мы заказали в интернет-магазине корзину для яблок, упаковка для корзины нам пришла, а самой корзины внутри нет. Примерно так же с null.

Мы создали переменную определенного типа (упаковку для корзины), а саму корзину туда не положили.

Следовательно, обратиться к упаковке и сказать «Дай мне корзину» мы не можем, ведь ее еще не существует.

Это приведет к падению приложения.

Это значит, что переменная пока не инициализирована, пустая, и работать с ней нельзя.

Инициализируется массив следующим кодом:

```
arr = new int[10];
```

Таким образом, чтобы создать новый массив, нужно написать код в две строчки:

```
int[] arr;
arr = new int[10];
```

Как мы видим, теперь в нашей эксплуатации появляется новое ключевое слово new.

Оно необходимо, так как в отличие от примитивных типов, что мы уже изучили, массив является **объектом**.

В Java объекты разделяются на примитивы и ссылочные типы данных.

С примитивами мы уже познакомились. Это просто число или просто символ.

В случае с объектами речь идет о более сложной структуре, которая внутри себя может иметь не только одно значение, но еще и методы по работе с этим значением, описания этого значения и т. д. В дальнейшем вы столкнетесь с одним из «описаний» значения массива — переменной length, которая хранит в себе размер массива.

И если примитивы в случае отсутствия инициализации будут хранить в себе значение «по умолчанию», это не приведет к падению программы при обращении к неинициализированной переменной. А объекты хранят в себе null. Обращение к null (пустоте) как к объекту приводит к падению приложения.

Все объекты в Java, кроме строк (но об этом узнаете дальше), создаются с помощью ключевого слово new.

Ключевое слово new говорит Java, что нужно в памяти нашего приложения выделить область и поместить туда объект. В случае массива нужно выделить память в размере, который рассчитывается путем умножения количества ячеек на размерность типа.

В нашем примере Java нужно выделить 10 ячеек по 32 бита (тип int занимает 32 бита). Безусловно, массив должен иметь дополнительную служебную ячейку, которая хранит данные о массиве, но об этом вы узнаете в дальнейшем.

Создать и инициализировать переменную массива мы можем в одну строку.

Делить на две, как в примерах выше, не требуется:

```
int[] arr = new int[10];
```

В квадратных скобках, как вы могли заметить ранее, указывается размер массива.

Типы данных массива

Типы данных, которые мы можем хранить в массиве, не ограничиваются int.

Мы можем создать массив для данных любого типа.

Если нам нужно создать массивы других типов, следует int в примере выше заменить на необходимый тип.

```
char[] chars = new char[15];
double[] doubleArr = new double[10];
```

```
long[] numbers = new long[82];
boolean[] bools = new boolean[2];
```

Работа с массивами

Возвращаемся к нашему ранее созданному массиву arr.

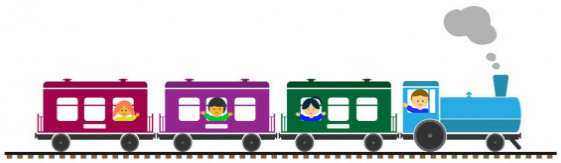
Как мы помним, arr является массивом типа int, емкость которого составляет 10 элементов. Это значит, что мы можем положить в массив 10 значений его типа, в данном случае int.

Чтобы яснее представлять себе работу с массивом, вообразите поезд.

В нашем случае это поезд из 10 вагонов.

Вагоны идут в строгом порядке, от начала до конца.

У каждого вагона есть номер от 1 до 10. Верно?



Верно, но не в нашей ситуации. Во многих языках программирования нумерация ячеек, в отличие от реальных предметов вроде поезда, начинается не с 1, а с 0.

Т. е. первый элемент будет лежать в ячейке под номером 0.

И ячейки, следовательно, будут иметь номера не от 1 до 10, а от 0 до 9 включительно.

Попробуем что-то положить в нулевую ячейку.

Для этого нам нужно написать следующее:

```
int[] arr = new int[10]; // Создали массив переменных типа int на 10 элементов
arr[0] = 5; // Положили в ячейку 0 значение 5
```

Нам никто не мешает положить в массив и значение из ранее созданной переменной.

```
int i = 10; // Создали переменную типа int со значением 10
arr[1] = i; // Положили значение переменной i в первую ячейку
```

Чтобы получить элемент из ячейки по известному нам номеру, нам требуется вызвать похожий на предыдущий пример код.

```
arr[5] = 10; // Положить в пятую ячейку значение 10
int i = arr[5]; // Создать переменную int i и инициализировать ее значение из пятой ячейки массива arr
```

Переменная length

Но что, если нам нужно положить какой-то символ в последнюю ячейку?

Неужели нам нужно для этого помнить (или искать в коде) значение размера, которое мы указывали при создании массива?

Нет. Так как массив является объектом (об этом упоминалось ранее), он может иметь в своей структуре дополнительные переменные.

На уроке по объектам вы научитесь сами создавать объекты и поймете, как это работает «под капотом».

Пока просто следует запомнить, что у любого массива внутри есть переменная length, которая хранит в себе размер массива. Это одна из тех характеристик или описаний объекта, что упоминались ранее.

Для того, чтобы узнать размер массива, нам необходимо написать следующее:

```
int[] arr = new int[10]; // Создали массив на 10 элементов
int arrSize = arr.length; // Присвоили переменной arrSize значение длины массива (10)
```

Обратите внимание

Так как массивы начинаются с нуля, значение arr.length всегда будет иметь индекс последнего элемента + 1. Т. е. последний элемент лежит по индексу (номеру) в ячейке 9, а length хранит в себе 10.

Получить доступ к последней ячейке всегда можно следующим способом:

```
int i = arr[arr.length - 1];
```

> Шпаргалка урока в PDF-формате:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/62155909-89e1-44e5-b7fd-1f9d84413176/1.5.__-_1.pdf