

Массивы в Kotlin

В Kotlin имеются "классические" массивы, когда в одном массиве могут быть данные только одного типа, и в массив нельзя добавлять элементы, как в список.

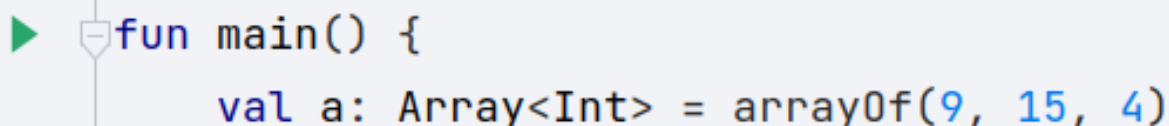
Объявление массива:

```
val ИМЯ: Array<ТИП>
```

Массивы обычно объявляют с помощью `val`, но это не значит, что нельзя изменять значения элементов массива. Это лишь значит, что неизменяемой переменной нельзя присвоить другой массив. Однако необходимость в подобном действии возникает редко.

Инициализация массивов возможна как с помощью функций, так и конструкторов классов. В примере ниже используется встроенная функция `arrayOf()`, которой передается набор значений.

```
fun main() {  
    val a: Array<Int> = arrayOf(9, 15, 4)  
  
    println("Количество элементов: ${a.size}")  
  
    a[0] = 10  
  
    print("Значения элементов: ")  
    for (i in a) {  
        print("$i ")  
    }  
    println()  
}
```



```
fun main() {  
    val a: Array<Int> = arrayOf(9, 15, 4)
```

```
println("Количество элементов: ${a.size}")

a[0] = 10

print("Значения элементов: ")
for (i in a) {
    print("$i ")
}
println()
}
```

Exm1Kt x

```
/usr/lib/jvm/java-19-openjdk-amd64/bin/java -javaa
Количество элементов: 3
Значения элементов: 10 15 4
```

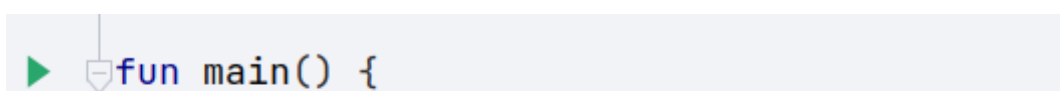
Для измерения длины массива используется свойство класса `size`.

Инициация трех элементов массива нулем при использовании конструктора класса:

```
fun main() {
    val a: Array<Int> = Array(3) {0}

    a[1] = 25

    for (i in a) {
        println(i)
    }
}
```



```
val a: Array<Int> = Array( size: 3) {0}

a[1] = 25

for (i in a) {
    println(i)
}
```

Exm2Kt x

/usr/lib/jvm/java-19-openjdk-amd64/bin/java

0

25

0

Конструктор класса **Array** принимает в качестве второго аргумента лямбда-выражение, которое заключается в фигурные скобки, оно генерирует значения. Такие аргументы в Kotlin обычно выносят за круглые скобки.

При запросе элементов и изменении их значений используется обычный для массивов синтаксис с квадратными скобками. В классе их переопределяют методы `get()` и `set()`. Их также можно вызывать непосредственно по имени.

В Kotlin массивы представлены не только классом **Array**. Есть специальные классы (и соответствующие им функции) для создания массивов, содержащих элементы примитивных типов – **BooleanArray**, **ByteArray**, **ShortArray**, **IntArray**, **LongArray**, **CharArray**, **FloatArray**, **DoubleArray**. Они несколько упрощают создание массивов. Обратим внимание, нет класса "строковый массив". Строки не примитивный тип.

```
fun main() {
    val a: BooleanArray = booleanArrayOf(true, false, false)

    for (i in a)
        println(i)
}
```

Для числовых типов также есть варианты беззнаковых массивов – `UByteArray`, `UShortArray`, `UIntArray`, `ULongArray`.

С помощью логических операторов `in` и `!in` можно проверить наличие определенных значений в массиве.

```
fun main() {
    val b: IntArray = intArrayOf(5, 3, 1, 2)

    println(2 in b)    // true
    println(0 in b)    // false
    println(10 !in b)  // true
}
```

Матрицы

В Kotlin возможно создание двумерных массивов.

```
fun main() {
    val a: Array<IntArray> = Array(3) {IntArray(4) {0} }

    // количество строк
    val n = a.size
    // количество элементов в строке
    val m = a[0].size

    for (i in 0 until n) {
        for (j in 0 until m) {
            a[i][j] = (Math.random() * 10)
                .toInt()
            print(" ${a[i][j]}")
        }
        println()
    }
}
```

Пример выполнения кода:

```
0 1 0 1
3 5 2 9
4 7 9 9
```

Заполнять массив можно также с помощью функций:

```
fun main() {
    val a: Array<IntArray> = Array(2) {IntArray(4) {0} }
    a[0] = intArrayOf(1, 2, 3, 4)
    a[1] = intArrayOf(5, 6, 7, 8)

    for (i in a) {
        for (j in i) {
            print(" $j")
        }
        println()
    }
}
```

[PDF-версия курса с дополнительными уроками](#)