Диапазоны, или интервалы, и прогрессии в Kotlin

В Kotlin диапазоны, они же интервалы, представляют собой объекты, которые содержат все целые числа в заданном промежутке последовательно от меньшего к большему. Прогрессии в Kotlin могут содержать числа от большего к меньшему, а также с заданным шагом.

Одно из наиболее частых применений интервалов и прогрессий – их использование в заголовке цикла `for`.

Диапазоны и прогрессии не являются итераторами. Для их превращения в итератор вызывается функция `iterator()`.

Только целочисленные типы могут формировать интервалы и прогрессии. Есть классы диапазонов `IntRange`, `LongRange`, `CharRange`. По-умолчанию используется `IntRange`. В случае `CharRange` для указания границ диапазона используются символы. Как известно, на самом деле символы кодируются числами. Если при создании был задан обратный порядок и/или использовался шаг, то создается объект класса `IntProgression`. Также есть `LongProgression`, `CharProgression`.

В Kotlin диапазоны создаются с помощью операторов интервала – двух точек `..` и `until`. В классах их переопределяют соответствующие функции. В обоих случаях объект будет содержать линейный набор значений. В случае двух точек создается истинный интервал, содержащий все значения от находящегося слева до находящегося справа от оператора, включая крайние. В случае `until` крайнее верхнее значение в диапазон входить не будет.

```
fun main() {
    val a = 1..4
    val b = 1 until 4

    for (i in a) print(i) // 1234
    println()
```

```
    for (i in b) print(i) // 123
}
```

Принадлежность значения интервалу проверяется операторами `in` и `!in`.

```
fun main() {
    val a = "String"

    val b = if (a[3] in 'k'..'z')
        "Yes"
    else
        a[3].toString()

    println(b)  // i
}
```

С оператором `when`:

```
fun main() {
    val a = "String"

    val b = when (a[3]) {
        in 'k'..'z' -> "Yes"
        else -> a[3].toString()
    }

    println(b)  // i
}
```

Проверка на невхождение в интервал:

```
fun main() {
    val a = 1..3
    val b = 1

    println(b !in a) // false
}
```

Для обратного хода и шага используются операторы-функции `downTo` и `step`:

```
fun main() {
    val a = 1..9 step 3
    val b = 1 until 10 step 2
    val c = 0 downTo -4
    val d = 'z' downTo 'a' step 5

    for (i in a) print(i) // 147
    println()
    for (i in b) print(i) // 13579
    println()
    for (i in c) print(" $i") // 0 -1 -2 -3 -4
    println()
    for (i in d) print(i) // zupkfa
}
```

Функция `toList()` преобразует интервал в список.

```
fun main() {
    val a = 1..3
    val b = 1 until 3
    val c = 3 downTo 1

    println(a.toList()) // [1, 2, 3]
    println(b.toList()) // [1, 2]
    println(c.toList()) // [3, 2, 1]
}
```

PDF-версия курса с дополнительными уроками