## Практическая работа 5. Строки в Котлин

## [Kotlin - 4. (Строковые шаблоны или динамические строки) - поиск Яндекса по видео (yandex.ru)](https://yandex.ru/video/preview/1304691395318676641)

## Теоретический материал

Строки String во многих отношениях подобны спискам, хотя формально и не являются ими. Список состоит из однотипных элементов, к которым можно обращаться по индексу и перебирать с помощью цикла **for**. Строки же точно так же состоят из символов Char, к которым также можно обращаться по индексу и которые также можно перебирать с помощью цикла **for**.

Напомним, что строковый *литерал* (явно указанная строка) в Котлине записывается в двойных кавычках. Переменную name произвольного типа можно преобразовать в строку, используя запись "$name" — строковый шаблон, или чуть более сложную запись name.toString() с тем же самым результатом.

Как мы видим, $ внутри строкового литерала имеет специальный смысл — вместо $name в строку будет подставлено содержимое переменной name. Как быть, если мы хотим просто включить в строку символ доллара? В этом случае следует применить так называемое *экранирование*, добавив перед символом доллара символ \. Например: "The price is 9.99 \$".

*Экранирование* может применяться и для добавления в строку различных специальных символов, не имеющих своего обозначения либо имеющих специальный смысл внутри строкового литерала. Например: \n — символ новой строки, \t — символ табуляции, \\ — символ «обратная косая черта», \" — символ «двойная кавычка».

Строки в Котлине являются неизменяемыми, то есть изменить какой-либо символ в уже созданной строке нельзя, можно только создать новую. В этом смысле они аналогичны немутирующим спискам List.

Перечислим наиболее распространённые операции над строками:

1. string1 + string2 — сложение или конкатенация строк, приписывание второй строки к первой.
2. string + char — сложение строки и символа (с тем же смыслом).
3. string.length — длина строки, то есть количество символов в ней.
4. string.isEmpty(), string.isNotEmpty() — получение признаков пустоты и непустоты строки (Boolean).
5. string[i] — индексация, то есть получение символа по целочисленному *индексу* (номеру) i в диапазоне от 0 до string.length - 1.
6. string.substring(from, to) — создание строки меньшего размера (подстроки), в который войдут символы строки string с индексами from, from + 1, …​, to - 2, to - 1. Символ с индексом to не включается.
7. char in string — проверка принадлежности символа char строке string.
8. for (char in list) { …​ } — цикл **for**, перебирающий все символы строки string.
9. string.first() — получение первого символа строки.
10. string.last() — получение последнего символа строки.
11. string.indexOf(char, startFrom) — найти индекс первого символа char в строке, начиная с индекса startFrom.
12. string.lastIndexOf(char, startFrom) — найти индекс первого символа char в строке, идя с конца и начиная с индекса startFrom.
13. string.toLowerCase() — преобразование строки в нижний регистр (то есть, замена прописных букв строчными).
14. string.toUpperCase() — преобразование строки в верхний регистр (замена строчных букв прописными).
15. string.capitalize() — замена первой буквы строки прописной.
16. string.trim() — удаление из строки пробельных символов в начале и конце: " ab c " преобразуется в "ab c"

В качестве примера рассмотрим функцию, проверяющую, является ли строка палиндромом. В палиндроме первый символ должен быть равен последнему, второй предпоследнему и т.д. Пример палиндрома: «А роза упала на лапу Азора». Из этого примера видно, что одни и те же буквы в разном регистре следует считать равными с точки зрения данной задачи. Кроме этого, не следует принимать во внимание пробелы. Решение на Котлине может быть таким:

fun isPalindrome(str: String): Boolean {

val lowerCase = str.toLowerCase().filter { it != ' ' }

for (i in 0..lowerCase.length / 2) {

if (lowerCase[i] != lowerCase[lowerCase.length - i - 1]) return false

}

return true

}

Обратите внимание, что мы с самого начала переписываем исходную строку str в промежуточную переменную lowerCase, преобразуя все буквы в нижний регистр и удаляя из строки все пробелы. Функция filter работает для строк точно так же, как и для списков — в строке оставляются только те символы, для которых функция-аргумент { it != ' '} вернёт **true**. Затем мы перебираем символы в первой половине строки, сравнивая каждый из них с символом из второй половины. Символ с индексом 0 (первый) должен соответствовать символу с индексом length - 1(последнему) и так далее.

1. Метод substringAfter() позволяет вывести из строки текст после символа указанного в скобках метода
2. Метод substringBefore () позволяет вывести из строки текст досимвола указанного в скобках метода

**Преобразование из списка в строку**

Очень часто используемой в Котлине является сложная функция преобразования списка в строку joinToString(). Её заголовок выглядит примерно так:

fun <T> List<T>.joinToString(

separator: String = ", ",

prefix: String = "",

postfix: String = "",

limit: Int = -1,

truncated: String = "...",

transform: (T) -> String = { "$it" }

): String { ... }

Получателем данной функции может быть список с произвольным типом элементов: List<T>. Такая запись описывает так называемую настраиваемую функцию, о них мы будем говорить подробнее позже.

Все пять параметров этой функции имеют так называемые *значения по умолчанию*. Это значит, что при желании мы можем вызвать эту функцию вообще не указывая аргументов. Например, listOf(1, 2, 3).joinToString() даст нам следующий результат: "1, 2, 3" — выводя в строку все элементы списка через запятую. Возможна, однако, более тонкая настройка вывода:

* параметр separator задаёт разделитель между элементами
* параметр prefix задаёт строку, которая выводится перед самым первым элементом списка (префикс)
* аналогично, параметр postfix задаёт строку, которая выводится после самого последнего элемента списка (постфикс)
* параметр limit задаёт максимальное количество выводимых элементов. Значение -1 соответствует неограниченному количеству элементов, но, скажем, вызов listOf(1, 2, 3).joinToString(", ", "", "", 1) будет иметь результат "1, …​" вместо результата по умолчанию "1, 2, 3"
* параметр truncated используется, если задан limit, и заменяет все элементы списка, которые не поместились в строке
* параметр transform задаёт способ преобразования каждого из элементов списка в строку — по умолчанию это "$it" для элемента списка it, может быть изменён с помощью лямбды (см. функции высшего порядка выше)

Рассмотрим простой пример: необходимо написать функцию, которая по заданному списку целых чисел вида [3, 6, 5, 4, 9] сформирует строку, содержащую пример их суммирования: "3 + 6 + 5 + 4 + 9 = 26". На Котлине это записывается так:

fun buildSumExample(list: List<Int>) = list.joinToString(separator = " + ", postfix = " = ${list.sum()}")

В данном случае нам требуется вызов функции joinToString, все параметры которой имеют некоторые значения по умолчанию, то есть не требуют обязательного указания при вызове. Нам требуется указать разделитель чисел " + " и в конце вывода добавить знак = с приписанной к нему суммой чисел из списка. Для этого нам необходимо указать значения параметров separator и postfix, при этом остальные параметры мы указывать не хотим. В этом случае используются так называемые *именованные аргументы*, например: separator = " + ". Эта запись указывает, что аргумент " + " соответствует параметру функции separator. Если бы имена separator и postfix не указывались, возникла бы путаница, поскольку неясно, какому именно из строковых параметров функции соответствует тот или иной аргумент вызова.

Задание

1. Написать программу, которая позволит вывести в словосочетание 1 <по МДК 03.01>, и словосочетание 2 <выполнить контрольную работу> соединенное конкатенацией.
2. Написать программу которая позволит вывести предложение: <В 3 семестре по МДК 03.01 Разработка мобильных приложений нужно выполнить контрольную работу> содержащиеся в переменной типа String вырезать и вывести только текст <нужно выполнить контрольную>
3. Написать программу которая позволит вывести текст:

Россия — священная наша держава, Россия — любимая наша страна. Могучая воля, великая слава — Твое достоянье на все времена!

Разбить строку на список строк с указанным числом символов. В последней строке может выводиться остаток