Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Кубанский государственный университет»

Кафедра вычислительных технологий

**ОТЧЕТ**

о выполнении лабораторной работы №13

по дисциплине функциональное и логическое программирование

Выполнил: ст. гр. 26/1

Минин К.С.

Проверил:

Жук А.С.

Краснодар,

2021

**Ход работы**

Рассмотрим некоторые методы класса String…

# String

class String : [Comparable](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-comparable/index.html)<String>, [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html)

The String class represents character strings. All string literals in Kotlin programs, such as "abc", are implemented as instances of this class.

### Св-ва

#### [**length**](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-string/length.html)

Возвращает длину этой последовательности символов.

val length: Int

### Ф-ции

#### [**compareTo**](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-string/compare-to.html)

Сравнивает этот объект с указанным объектом для заказа. Возвращает ноль, если этот объект равен указанному [другому](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-comparable/compare-to.html#kotlin.Comparable$compareTo(kotlin.Comparable.T)/other) объекту, отрицательное число, если оно меньше [другого](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-comparable/compare-to.html#kotlin.Comparable$compareTo(kotlin.Comparable.T)/other) , или положительное число, если оно больше [другого](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-comparable/compare-to.html#kotlin.Comparable$compareTo(kotlin.Comparable.T)/other) .

### fun compareTo(other: String): Int

#### [**equals**](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-string/equals.html)

Указывает, равен ли какой-либо другой объект этому. Реализации должны соответствовать следующим требованиям:

* Рефлексивный: для любого значения ненулевого x, x.equals(x)должно возвращать верно.
* Симметричный: для любых значений ненулевых xи y, x.equals(y)должны возвращать истинны тогда и только тогда , когда y.equals(x)возвращает истина.
* Транзитивный: для любых ненулевых значений x, yи z, если x.equals(y)возвращает истину и y.equals(z)возвращает истину, тогда x.equals(z)должно возвращать истину.
* Согласованно: для любых ненулевых значений xи y, несколько вызовов x.equals(y)последовательно возвращают истину или последовательно возвращают ложь, при условии, что никакая информация, используемая при equalsсравнении объектов, не изменяется.
* Никогда не равен нулю: для любого значения ненулевым x, x.equals(null)должен возвращать ложь.

fun equals(**other**: Any?): Boolean

#### [**get**](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-string/get.html)

Returns the character of this string at the specified [index](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-string/get.html#kotlin.String$get(kotlin.Int)/index).

fun get(**index**: Int): Char

#### [**hashCode**](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-string/hash-code.html)

Returns a hash code value for the object.

* Каждый раз, когда он вызывается для одного и того же объекта более одного раза, hashCodeметод должен последовательно возвращать одно и то же целое число, при условии, что никакая информация, используемая при equalsсравнении объекта, не изменяется.
* Если два объекта равны согласно equals()методу, то вызов hashCodeметода для каждого из двух объектов должен привести к одинаковому целочисленному результату.

fun hashCode(): Int

#### [**plus**](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-string/plus.html)

Возвращает строку, полученную путем объединения этой строки со строковым представлением данного [другого](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-string/plus.html#kotlin.String$plus(kotlin.Any?)/other) объекта.

operator fun plus(**other**: Any?): String

#### [**subSequence**](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-string/sub-sequence.html)

Возвращает новую последовательность символов, которая является подпоследовательностью этой последовательности символов, начиная с указанного [startIndex](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/sub-sequence.html" \l "kotlin.CharSequence$subSequence(kotlin.Int,%20kotlin.Int)/startIndex) и заканчивая прямо перед указанным [endIndex](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/sub-sequence.html" \l "kotlin.CharSequence$subSequence(kotlin.Int,%20kotlin.Int)/endIndex) .

fun subSequence(**startIndex**: Int, **endIndex**: Int): [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html)

#### [**toString**](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-string/to-string.html)

Returns a string representation of the object.

fun toString(): String

# contains

operator fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).contains(  
    **other**: [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html),  
    **ignoreCase**: Boolean = false  
): Boolean

Returns true if this char sequence contains the specified [other](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/contains.html#kotlin.text$contains(kotlin.CharSequence,%20kotlin.CharSequence,%20kotlin.Boolean)/other) sequence of characters as a substring.

ignoreCase - true to ignore character case when comparing strings. By default false.

operator fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).contains(  
    **char**: Char,  
    **ignoreCase**: Boolean = false  
): Boolean

Returns true if this char sequence contains the specified character [char](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/contains.html#kotlin.text$contains(kotlin.CharSequence,%20kotlin.Char,%20kotlin.Boolean)/char).

ignoreCase - true to ignore character case when comparing characters. By default false.

operator fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).contains(**regex**: [Regex](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/-regex/index.html)): Boolean

Returns true if this char sequence contains at least one match of the specified regular expression [regex](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/contains.html#kotlin.text$contains(kotlin.CharSequence,%20kotlin.text.Regex)/regex).

# count

fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).count(): Int

Returns the length of this char sequence.

inline fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).count(  
    **predicate**: (Char) -> Boolean  
): Int

Returns the number of символов matching the given [predicate](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/count.html#kotlin.text$count(kotlin.CharSequence,%20kotlin.Function1((kotlin.Char,%20kotlin.Boolean)))/predicate).

# drop

fun String.drop(**n**: Int): String

Returns a string with the first [n](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/drop.html#kotlin.text$drop(kotlin.String,%20kotlin.Int)/n) characters removed.

### Exceptions - IllegalArgumentException - if [n](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/drop.html#kotlin.text$drop(kotlin.String,%20kotlin.Int)/n) is negative.

# dropLast

fun String.dropLast(**n**: Int): String

Returns a string with the last [n](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/drop-last.html#kotlin.text$dropLast(kotlin.String,%20kotlin.Int)/n) characters removed.

### Exceptions - IllegalArgumentException - if [n](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/drop-last.html#kotlin.text$dropLast(kotlin.String,%20kotlin.Int)/n) is negative.

# dropLastWhile

inline fun String.dropLastWhile(  
    **predicate**: (Char) -> Boolean  
): String

Returns a string containing all characters except last characters that удовлетворяют the given [predicate](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/drop-last-while.html#kotlin.text$dropLastWhile(kotlin.String,%20kotlin.Function1((kotlin.Char,%20kotlin.Boolean)))/predicate).

### val string = "<<<First Grade>>>"

### println(string.drop(6)) // st Grade>>>

### println(string.dropLast(6)) // <<<First Gr

### println(string.dropWhile { !it.isLetter() }) // First Grade>>>

### println(string.dropLastWhile { !it.isLetter() }) // <<<First Grade

# dropWhile

inline fun String.dropWhile(  
    **predicate**: (Char) -> Boolean  
): String

Returns a string containing all characters except first characters that удовлетворяют the given [predicate](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/drop-while.html#kotlin.text$dropWhile(kotlin.String,%20kotlin.Function1((kotlin.Char,%20kotlin.Boolean)))/predicate).

# elementAtOrNull

fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).elementAtOrNull(**index**: Int): Char?

Returns a character at the given [index](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/element-at-or-null.html#kotlin.text$elementAtOrNull(kotlin.CharSequence,%20kotlin.Int)/index) or null if the [index](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/element-at-or-null.html#kotlin.text$elementAtOrNull(kotlin.CharSequence,%20kotlin.Int)/index) is out of bounds of this char sequence.

### val list = listOf(1, 2, 3)

### println(list.elementAtOrNull(0)) // 1

### println(list.elementAtOrNull(2)) // 3

### println(list.elementAtOrNull(3)) // null

### val emptyList = emptyList<Int>()

### println(emptyList.elementAtOrNull(0)) // null

# endsWith

### fun String.endsWith(     suffix: String,     ignoreCase: Boolean = false ): Boolean

fun String.endsWith(  
    **suffix**: String,  
    **ignoreCase**: Boolean  
): Boolean

Returns true if this string ends with the specified suffix.

fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).endsWith(  
    **char**: Char,  
    **ignoreCase**: Boolean = false  
): Boolean

Returns true if this char sequence ends with the specified character.

# filter

inline fun String.filter(  
    **predicate**: (Char) -> Boolean  
): String

Returns a string containing only those characters from the original string that match the given [predicate](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/filter.html#kotlin.text$filter(kotlin.String,%20kotlin.Function1((kotlin.Char,%20kotlin.Boolean)))/predicate).

### val text = "a1b2c3d4e5"

### val textWithOnlyDigits = text.filter { it.isDigit() }

### println(textWithOnlyDigits) // 12345

# filterIndexedTo

inline fun <C : [Appendable](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/-appendable/index.html)> [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).filterIndexedTo(  
    **destination**: C,  
    **predicate**: (**index**: Int, Char) -> Boolean  
): C

Appends all characters matching the given [predicate](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/filter-indexed-to.html#kotlin.text$filterIndexedTo(kotlin.CharSequence,%20kotlin.text.filterIndexedTo.C,%20kotlin.Function2((kotlin.Int,%20kotlin.Char,%20kotlin.Boolean)))/predicate) to the given [destination](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/filter-indexed-to.html#kotlin.text$filterIndexedTo(kotlin.CharSequence,%20kotlin.text.filterIndexedTo.C,%20kotlin.Function2((kotlin.Int,%20kotlin.Char,%20kotlin.Boolean)))/destination).

### predicate - function that takes the index of a character and the character itself and returns the result of predicate evaluation on the character.

### val numbers: List<Int> = listOf(0, 1, 2, 3, 4, 8, 6)

### val numbersOnSameIndexAsValue = mutableListOf<Int>()

### println(numbersOnSameIndexAsValue) // []

### numbers.filterIndexedTo(numbersOnSameIndexAsValue) { index, i -> index == i }

### println(numbersOnSameIndexAsValue) // [0, 1, 2, 3, 4, 6]

# find

inline fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).find(  
    **predicate**: (Char) -> Boolean  
): Char?

Returns the first character matching the given [predicate](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/find.html#kotlin.text$find(kotlin.CharSequence,%20kotlin.Function1((kotlin.Char,%20kotlin.Boolean)))/predicate), or null if no such character was found.

### val numbers = listOf(1, 2, 3, 4, 5, 6, 7)

### val firstOdd = numbers.find { it % 2 != 0 }

### val lastEven = numbers.findLast { it % 2 == 0 }

### println(firstOdd) // 1

### println(lastEven) // 6

# findLast

inline fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).findLast(  
    **predicate**: (Char) -> Boolean  
): Char?

Returns the last character matching the given [predicate](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/find-last.html#kotlin.text$findLast(kotlin.CharSequence,%20kotlin.Function1((kotlin.Char,%20kotlin.Boolean)))/predicate), or null if no such character was found.

# first

fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).first(): Char

Returns first character.

### Exceptions - NoSuchElementException - if the char sequence is empty.

inline fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).first(  
    **predicate**: (Char) -> Boolean  
): Char

Returns the first character matching the given [predicate](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/first.html#kotlin.text$first(kotlin.CharSequence,%20kotlin.Function1((kotlin.Char,%20kotlin.Boolean)))/predicate).

# fold

inline fun <R> [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).fold(  
    **initial**: R,  
    **operation**: (**acc**: R, Char) -> R  
): R

Accumulates value starting with [initial](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/fold.html#kotlin.text$fold(kotlin.CharSequence,%20kotlin.text.fold.R,%20kotlin.Function2((kotlin.text.fold.R,%20kotlin.Char,%20)))/initial) value and applying [operation](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/fold.html#kotlin.text$fold(kotlin.CharSequence,%20kotlin.text.fold.R,%20kotlin.Function2((kotlin.text.fold.R,%20kotlin.Char,%20)))/operation) from left to right to current accumulator value and each character.

Returns the specified [initial](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/fold.html#kotlin.text$fold(kotlin.CharSequence,%20kotlin.text.fold.R,%20kotlin.Function2((kotlin.text.fold.R,%20kotlin.Char,%20)))/initial) value if the char sequence is empty.

operation - function that takes current accumulator value and a character, and calculates the next accumulator value.

# forEach

inline fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).forEach(**action**: (Char) -> Unit)

Performs (выполняет) the given [action](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/for-each.html#kotlin.text$forEach(kotlin.CharSequence,%20kotlin.Function1((kotlin.Char,%20kotlin.Unit)))/action) on each character.

# // format ?

fun String.format(vararg **args**: Any?): String

Uses this string as a format string and returns a string obtained (полученную) by substituting (подстановкой) the specified (указанных) arguments, using the default locale (с использованием языкового стандарта по умолчанию).

# getOrNull

fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).getOrNull(**index**: Int): Char?

Returns a character at the given [index](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/get-or-null.html#kotlin.text$getOrNull(kotlin.CharSequence,%20kotlin.Int)/index) or null if the [index](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/get-or-null.html#kotlin.text$getOrNull(kotlin.CharSequence,%20kotlin.Int)/index) is out of bounds of this char sequence.

val list = listOf(1, 2, 3)

println(list.getOrNull(0)) // 1

println(list.getOrNull(2)) // 3

println(list.getOrNull(3)) // null

val emptyList = emptyList<Int>()

println(emptyList.getOrNull(0)) // null

# groupBy

inline fun <K> [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).groupBy(  
    **keySelector**: (Char) -> K  
): [Map](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-map/index.html)<K, [List](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-list/index.html)<Char>>

Groups characters of the original char sequence by the key returned by the given [keySelector](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/group-by.html" \l "kotlin.text$groupBy(kotlin.CharSequence,%20kotlin.Function1((kotlin.Char,%20kotlin.text.groupBy.K)))/keySelector) function applied to each character and returns a map where each group key is associated with a list of corresponding characters.

The returned map preserves the entry iteration order of the keys produced from the original char sequence.

val words = listOf("a", "abc", "ab", "def", "abcd")

val byLength = words.groupBy { it.length }

println(byLength.keys) // [1, 3, 2, 4]

println(byLength.values) // [[a], [abc, def], [ab], [abcd]]

val mutableByLength: MutableMap<Int, MutableList<String>> = words.groupByTo(mutableMapOf()) { it.length }

// same content as in byLength map, but the map is mutable

println("mutableByLength == byLength is ${mutableByLength == byLength}") // true

# ifEmpty

inline fun <C, R> C.ifEmpty(  
    **defaultValue**: () -> R  
): R where C : [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html), C : R

Returns this char sequence if it's not empty or the result of calling [defaultValue](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/if-empty.html" \l "kotlin.text$ifEmpty(kotlin.text.ifEmpty.C,%20kotlin.Function0((kotlin.text.ifEmpty.R)))/defaultValue) function if the char sequence is empty.

val empty = ""

val emptyOrNull: String? = empty.ifEmpty { null }

println(emptyOrNull) // null

val emptyOrDefault = empty.ifEmpty { "default" }

println(emptyOrDefault) // default

val nonEmpty = "abc"

val sameString = nonEmpty.ifEmpty { "def" }

println("nonEmpty === sameString is ${nonEmpty === sameString}") // true

# indexOfFirst

inline fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).indexOfFirst(  
    **predicate**: (Char) -> Boolean  
): Int

Returns index of the first character matching the given [predicate](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/index-of-first.html#kotlin.text$indexOfFirst(kotlin.CharSequence,%20kotlin.Function1((kotlin.Char,%20kotlin.Boolean)))/predicate), or -1 if the char sequence does not contain such character.

# indexOfLast

inline fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).indexOfLast(  
    **predicate**: (Char) -> Boolean  
): Int

Returns index of the last character matching the given [predicate](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/index-of-last.html#kotlin.text$indexOfLast(kotlin.CharSequence,%20kotlin.Function1((kotlin.Char,%20kotlin.Boolean)))/predicate), or -1 if the char sequence does not contain such character.

# isEmpty

fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).isEmpty(): Boolean

Returns true if this char sequence is empty (contains no characters).

# isNotBlank

fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).isNotBlank(): Boolean

Returns true if this char sequence is not empty and contains some characters except of whitespace characters.

fun validateName(name: String): String {

require(name.isNotBlank()) { "Name cannot be blank" }

return name

}

println(validateName("Adam")) // Adam

// validateName("") // will fail

// validateName(" \t\n") // will fail

# isNullOrEmpty

fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html)?.isNullOrEmpty(): Boolean

Returns true if this nullable char sequence is either null or empty.

# last

fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).last(): Char

Returns the last character.

val string = "Kotlin 1.4.0"

println(string.last()) // 0

println(string.last { it.isLetter() }) // n

println(string.lastOrNull { it > 'z' }) // null

// string.last { it > 'z' } // will fail

val emptyString = ""

println(emptyString.lastOrNull()) // null

// emptyString.last() // will fail

### Exceptions - NoSuchElementException - if the char sequence is empty.

# lowercase

fun String.lowercase(): String

Returns a copy of this string converted to lower case using Unicode mapping rules of the invariant locale.

println("Iced frappé!".lowercase()) // iced frappé!

# Match

fun String.match(**regex**: String): Array<String>?\

# maxByOrNull

inline fun <R : [Comparable](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-comparable/index.html)<R>> [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).maxByOrNull(  
    **selector**: (Char) -> R  
): Char?

Returns the first character yielding (дающий) the largest value of the given function or null if there are no characters.

# minOrNull

fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).minOrNull(): Char?

Returns the smallest character or null if there are no characters.

# none

fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).none(): Boolean

Returns true if the char sequence has no characters.

inline fun [CharSequence](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-char-sequence/index.html).none(  
    **predicate**: (Char) -> Boolean  
): Boolean

Returns true if no characters match the given [predicate](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/none.html#kotlin.text$none(kotlin.CharSequence,%20kotlin.Function1((kotlin.Char,%20kotlin.Boolean)))/predicate).

val emptyList = emptyList<Int>()

println("emptyList.none() is ${emptyList.none()}") // true

val nonEmptyList = listOf("one", "two", "three")

println("nonEmptyList.none() is ${nonEmptyList.none()}") // false