МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра ЭВМ

Отчет по предмету

Функциональное программирование

Лабораторная работа №3

**«**Работа со строками**»**

|  |  |
| --- | --- |
| **Выполнили:**  студенты группы 150501  Смоленский Н.О.  Петруненко В.Е. | **Проверила:**  Герман Ю.О. |

Минск

2023

**Цель работы**: Изучить технику работы со строками в Scala.

**Теоретические сведения:**

Scala предоставляет богатый набор функций для работы со строками. Вот некоторые расширенные функции, которые можно использовать со строками в Scala:

1. replaceAll: эта функция используется для замены всех вхождений строки другой строкой. Функция принимает два аргумента: первый аргумент — это заменяемое регулярное выражение, а второй аргумент — строка замены.

Пример:

val str = "Hello, World!"

val newStr = str.replaceAll("o", "a")

println(newStr) // "Hella, Warld!"

split: Эта функция используется для разделения строки на массив подстрок на основе разделителя. Функция принимает один аргумент — строку-разделитель.

Пример:

val str = "apple,banana,orange"

val arr = str.split(",")

println(arr.mkString(" ")) // "apple banana orange"

1. startsWith и endsWith:

Эти функции используются для проверки того, начинается или заканчивается строка заданной подстрокой. Функции принимают один аргумент — проверяемую подстроку.

Пример:

val str = "Hello, World!"

println(str.startsWith("Hello")) // true

println(str.endsWith("!")) // true

1. substring: Выделяет подстроку из строки. Пример:

val str = "Hello, World!"

val subStr = str.substring(7, 12)

println(subStr) // "World"

1. toCharArray: Преобразует строку в массив символов

Пример:

val str = "Hello, World!"

val arr = str.toCharArray()

println(arr.mkString(" ")) // "H e l l o , W o r l d !"

1. toLowerCase и toUpperCase: Преобразует символы строки в верхний и нижний регистр соответственно.

Пример:

val str = "Hello, World!"

println(str.toLowerCase) // "hello, world!"

println(str.toUpperCase) // "HELLO, WORLD!"

1. trim: Отсекает концевые пробелы.

Пример:

val str = " Hello, World! "

println(str.trim) // "Hello, World!"

1. indexOf иlastIndexOf: Получает первый и последний индекс подстроки в строке (то есть номер позиции, с которой начинается подстрока).

Пример:

val str = "Hello, World!"

println(str.indexOf("o")) // 4

println(str.lastIndexOf("o")) // 8

1. charAt: Определяет символ, стоящий на указанной позиции.

Пример:

val str = "Hello, World!"

println(str.charAt(7)) // 'W'

1. stripMargin: Удаляет ведущие пробелы перед строкой.

Пример:

val str =

"""

|Hello,

|World!

|""".stripMargin

println(str) // "Hello,\nWorld!\n"

Теперь обратимся к регулярным выражениям.

В Scala регулярные выражения представлены классом scala.util.matching.Regex, который предоставляет множество методов для сопоставления строк и управления ими на основе регулярных выражений.

Вот пример, демонстрирующий некоторые основные функции регулярных выражений в Scala:

val regex = """(\d{3})-(\d{2})-(\d{4})""".r

val str1 = "123-45-6789"

val str2 = "abc-12-3456"

val match1 = regex.findFirstMatchIn(str1)

val match2 = regex.findFirstMatchIn(str2)

match1 match {

case Some(m) => println(s"Match found: ${m.group(0)}")

case None => println("No match found")

}

match2 match {

case Some(m) => println(s"Match found: ${m.group(0)}")

case None => println("No match found")

}

В этом примере мы определяем шаблон регулярного выражения, который соответствует номеру социального страхования в формате XXX-XX-XXXX, где X — цифра. Затем мы пытаемся сопоставить этот шаблон с двумя разными строками: «123-45-6789» и «abc-12-3456».

Метод findFirstMatchIn возвращает объект Option[Match], представляющий первое совпадение шаблона в заданной строке, если таковое имеется. Мы используем сопоставление с образцом, чтобы извлечь совпадающую подстроку из объекта Match и распечатать ее.

Когда мы запускаем этот пример, мы получаем следующий вывод:

Match found: 123-45-6789

No match found

В этом случае первая строка соответствует шаблону регулярного выражения, поэтому мы получаем объект соответствия с совпадающей подстрокой «123-45-6789». Вторая строка не соответствует шаблону, поэтому мы получаем объект None вместо объекта соответствия.

Обратите внимание, что регулярные выражения могут быть довольно мощными и сложными, и в классе Regex доступно гораздо больше функций и методов для работы с ними.

In Scala, you can use the findAllIn method of the Regex class to find all occurrences of a substring in a string using a regular expression. Here's an example:

val str = "The quick brown fox jumps over the lazy dog"

val pattern = "fox".r

val matches = pattern.findAllIn(str)

matches.foreach(println)

In this example, we define a string str containing the text "The quick brown fox jumps over the lazy dog". We then define a regular expression pattern pattern using the .r method, which matches the substring "fox". We use the findAllIn method of the Regex class to find all occurrences of the pattern in the string, and store the matches in the matches variable.

**Код программы:**

**Вариант 2**

1. Дан текст: ‘1+1=2’. C помощью техники регулярных выражений заменить цифры на слова: 1- one, 2 – two

val numsNames = Map(

'0' -> "Zero",

'1' -> "One",

'2' -> "Two",

'3' -> "Three",

'4' -> "Four",

'5' -> "Five",

'6' -> "Six",

'7' -> "Seven",

'8' -> "Eight",

'9' -> "Nine"

)

def replaceNums(text: String): String = {

val numberPattern: Regex = "\\d".r

val result = numberPattern.replaceAllIn(text, { matchObj =>

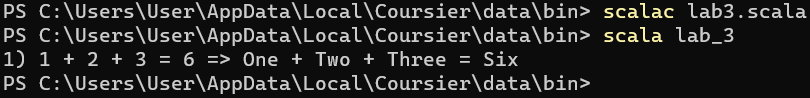
val digit = matchObj.group(0)

numsNames.getOrElse(digit.charAt(0), "0")

})

result

}



1. Найти в тексте ‘Being quiet buetiful girl she never thought of buety’ все слова, начинающиеся на bue.

def getWordsByPart(text: String, subString: String): String = {

var result = ""

val partPattern: Regex = s"\\b\\w\*${Regex.quote(subString)}\\w\*\\b".r

val matchesFound = partPattern.findAllMatchIn(text)

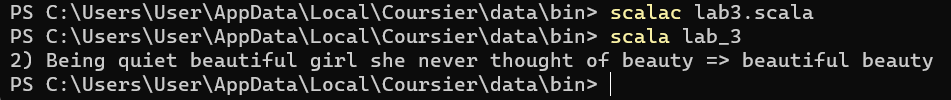
while(matchesFound.hasNext) {

result += matchesFound.next() + " "

}

result

}



1. В тексте ‘one plus one is something’ заменить one на two.

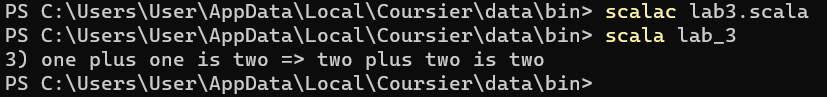
def replaceWords(text: String, substitute: String, replacement: String): String = {

val wordPattern: Regex = s"\\b${Regex.quote(substitute)}\\b".r

val result = wordPattern.replaceAllIn(text, replacement)

result

}



1. Поменять местами первое и последнее слово в тексте world is nice.

def swapWords(text: String): String = {

val stringPattern = s"(\\b\\w+\\b)(.\*)(\\b\\w+\\b)".r

val result = text match{

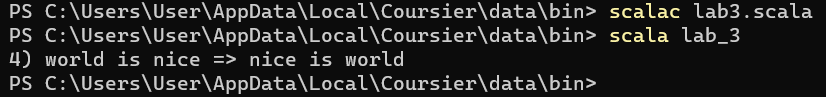
case stringPattern(first, middle, last) => s"$last$middle$first"

case \_ => text

}

result

}



1. Дан текст: ‘Hello to all my friends’. Выбросить все согласные.

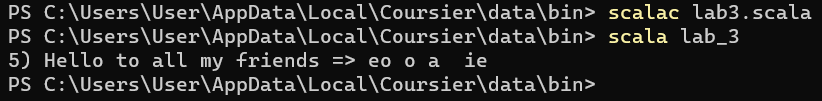
def trimConsonants(text: String): String = {

val consonantsPattern: Regex = "[bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ]".r

val result = consonantsPattern.replaceAllIn(text, "")

result

}



1. Дан текст: ‘Hello to everybody’. Удалить каждую вторую букву в слове

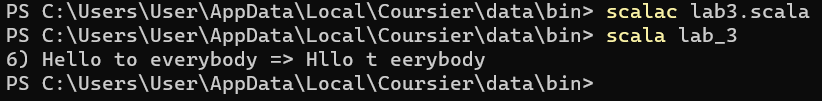
def cutSecondLetter(text: String): String = {

val letterPattern: Regex = "\\b(\\w)(\\w)(\\w\*)\\b".r

val result = letterPattern.replaceAllIn(text, matched => matched.group(1) + matched.group(3))

result

}



1. Дан текст: ‘Hello to everybody’. Удалить все вхождения буквы e

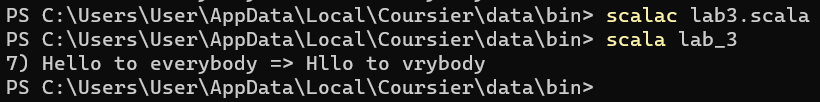
def trimEs(text: String): String = {

val ePattern: Regex = "[e]".r

val result = ePattern.replaceAllIn(text, "")

result

}



1. Дан текст: ‘Be healthy’. Вставить слова always чтобы получить Be always healthy

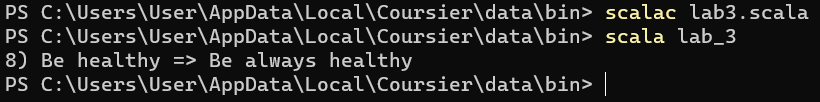
def placeWord(text: String, wordToPlace: String): String = {

val placePattern: Regex = s"^\\b(\\w+)\\b(.\*)".r

val result = placePattern.replaceAllIn(text, matched => matched.group(1) + " " + wordToPlace + matched.group(2))

result

}



**Вывод**

В ходе выполнения лабораторной работы было проведено ознакомление с синтаксисом регулярных выражений в Scala, различными коллекциями и списками. Используя полученную информацию, была написана программа для выполнения различных операций над коллекцией map.

В каждом слове в предложении утроить первую букву. Удалить все пробелы между словами.

**He** **had** a great time yesterday.