БГУИР

Кафедра ЭВМ

Отчёт по лабораторной работе №3

**Тема: «Работа со строками»**

Выполнили:

студенты группы 150504

Горбачевский К.В.

Горбачевский М.В.

Проверила:

**к.т.н., доцент Герман Ю.О.**

Минск 2023

1. **ЦЕЛЬ РАБОТЫ**

Изучить технику работы со строками в Scala.

1. **КРАТКИЕ 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 1: заменить все буквы “o” на “a”

Листинг кода:

    def example1(): Unit = {

        val str = "Hello, World!"

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

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

    }

Результат выполнения:



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

Пример 2: создать массив из строк из строки

Листинг кода:

1. def example2(): Unit = {
2. val str = "apple,banana,orange"
3. val arr = str.split(",")
4. println(arr.mkString(" ")) // "apple banana orange"
5. }

Результат выполнения:



1. startsWith и endsWith:

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

Пример 3: проверить начинаетя(заканчивается) строка подстрокой

Листинг кода:

    def example3(): Unit = {

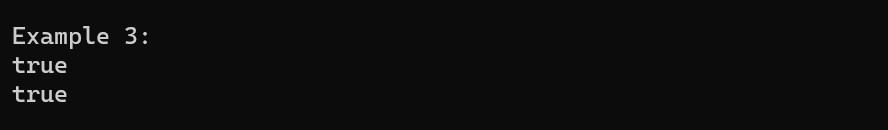
        val str = "Hello, World!"

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

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

    }

Результат выполнения:



1. substring: Выделяет подстроку из строки. Пример:

Пример 4: Выделить подстроку из строки

Листинг кода:

    def example4(): Unit = {

        val str = "Hello, World!"

        val subStr = str.substring(7, 12)

        println(subStr) // "World"

    }

Результат выполнения:



1. toCharArray: Преобразует строку в массив символов

Пример 5: Преобразовать строку в массив символов

Листинг кода:

    def example5(): Unit = {

        val str = "Hello, World!"

        val arr = str.toCharArray()

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

    }

Результат выполнения:



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

Пример 6: Преобразовать строку в нижний и верхний регистр

Листинг кода:

    def example6(): Unit = {

        val str = "Hello, World!"

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

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

    }

Результат выполнения:



1. trim: Отсекает концевые пробелы.

Пример 7: Удалить из строки концевые пробелы

Листинг кода:

    def example7(): Unit = {

        val str = "   Hello, World!   "

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

    }

Результат выполнения:



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

Пример 8: Получить индекс первого и последнего вхождения символа в строке

Листинг кода:

    def example8(): Unit = {

        val str = "Hello, World!"

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

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

    }

Результат выполнения:



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

Пример 9: Получить символ по индексу

Листинг кода:

1. def example9(): Unit = {
2. val str = "Hello, World!"
3. println(str.charAt(7)) // 'W'
4. }

Результат выполнения:



1. stripMargin: Удаляет ведущие пробелы перед строкой.

Пример 10: Удалить пробелы перед строкой

Листинг кода:

    def example10(): Unit = {

        val str =

        """

            |Hello,

            |World!

            |""".stripMargin

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

    }

Результат выполнения:



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:

Пример 1: Проверить соответствует ли строка регулярному выражению

Листинг кода:

    def example11(): Unit = {

        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:

Результат выполнения



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.

Finally, we loop through the matches variable using the foreach method and print each match to the console.

1. **Ход работы**

***Общее задание:*** Вывести суммарное число всех гласных в собственном тексте.

***Листинг программы:***

*def allVowels(text: String): Int = {*

*val pattern = "[aeiouyAEIOUY]".r*

*var vowels = pattern.findAllMatchIn(text)*

*return vowels.size*

*}*

***Результат работы программы:***



*Рисунок 1 – Результат работы общего задания*

***Задание 1:*** Дан текст: ‘Hello to everybody’. C помощью техники регулярных выражений заменить латинские буквы на русские (или на цифры, если русский шрифт не поддерживается)

***Листинг программы:***

*def latinToNumbers(string: String): String = {*

*val pattern: String = "[A-Za-z]"*

*return string.replaceAll(pattern, "0");*

*}*

***Результат работы программы:***



*Рисунок 1.1 – Результат работы задания 1*

***Задание 2:*** Найти в тексте “When executing the exercise extract all extra words” все слова, начинающиеся на ext.

***Листинг программы:***

*def findWords(): String = {*

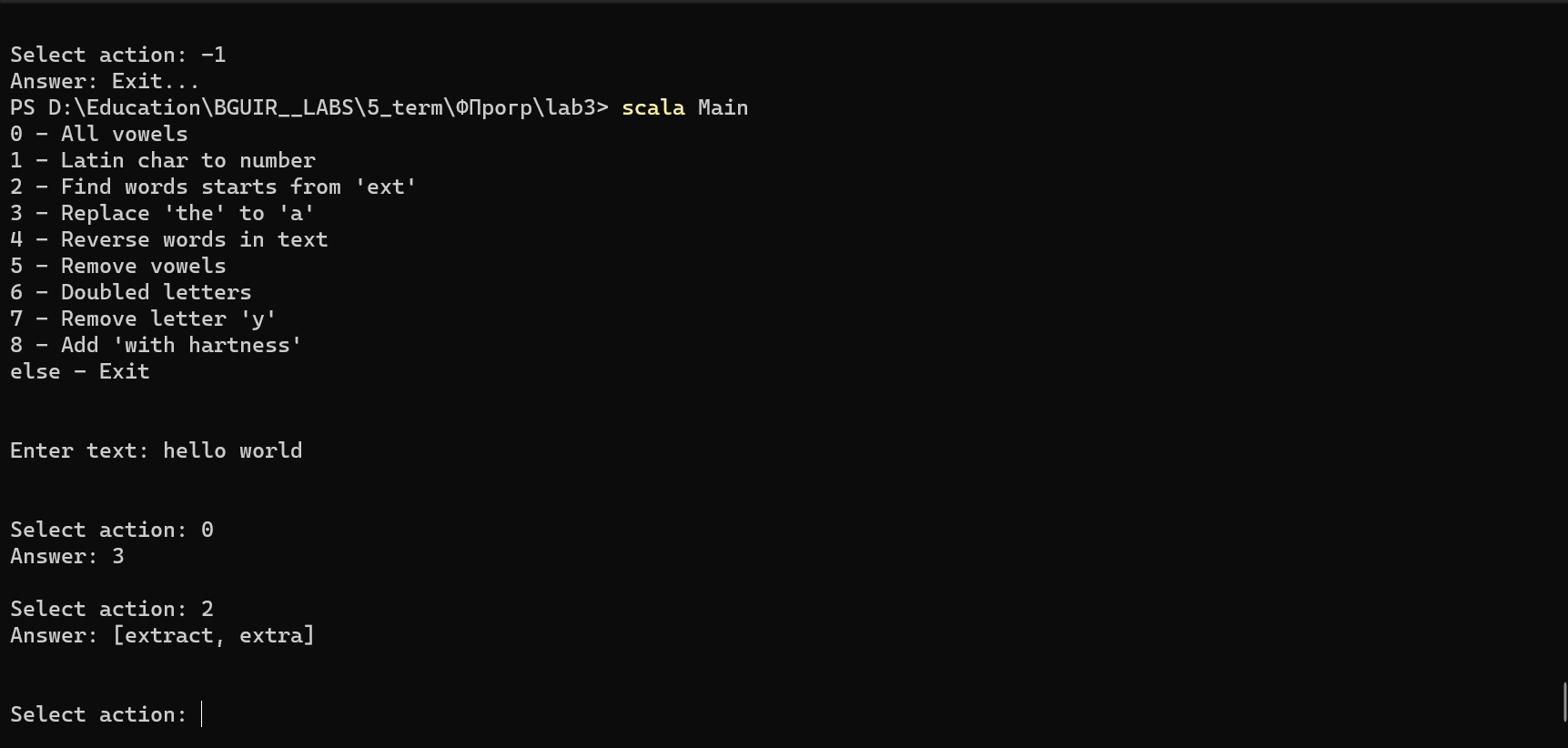
*var words = "When executing the exercise extract all extra words".split(" ");*

*val pattern = "ext"*

*return words.filter(word => word.startsWith(pattern)).mkString(", ")*

*}*

***Результат работы программы:***



*Рисунок 2.1 – Результат работы задания 2*

***Задание 3:*** В тексте ‘A big round ball fall to the ground’ заменить артикль the на a.

***Листинг программы:***

*case 3 => {*

*val answer = "A big round ball fall to the ground".replaceAll("the", "a")*

*println(answer)*

*}*

***Результат работы программы:***



*Рисунок 3.1 – Результат работы задания 3*

***Задание 4:*** Записать все слова в тексте в обратном порядке.

***Листинг программы:***

*def reversWordsInText(text: String): String = {*

*var array = text.split(" ")*

*return array.map(word => word.reverse).mkString(" ")*

*}*

***Результат работы программы:***



*Рисунок 4.1 – Результат работы задания 4*

***Задание 5:*** Дан текст: ‘Hello to everybody’. Выбросить все гласные.

***Листинг программы:***

*case 5 => {*

*print(msg.replaceAll("[aeiouyAEIOUY]", ""))*

*}*

***Результат работы программы:***



*Рисунок 5.1 – Результат работы задания 5*

***Задание 6:*** Дан текст: ‘Hello to everybody’. Удвоить каждую букву в слове

***Листинг программы:***

*case 6 => {*

*var answer: StringBuilder = StringBuilder();*

*msg.map(ch => answer.append(ch.toString() + ch.toString()))*

*print(answer.toString())*

*}*

***Результат работы программы:***



*Рисунок 6.1 – Результат работы задания 6*

***Задание 7:*** Дан текст: ‘Hello to everybody’. Удалить все вхождения буквы y

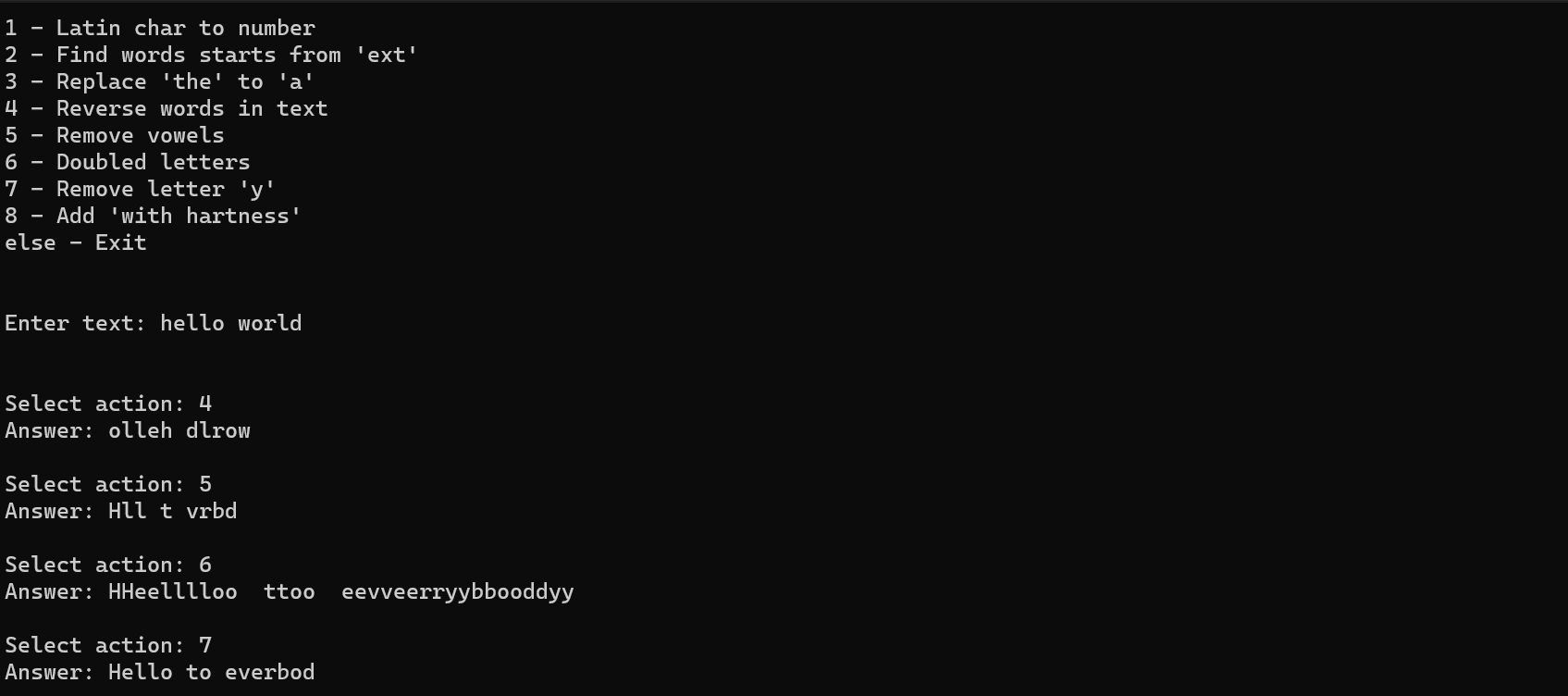
***Листинг программы:***

*case 7 => {*

*print(msg.replaceAll("y", ""))*

*}*

***Результат работы программы:***



*Рисунок 7.1 – Результат работы задания 7*

***Задание 8:*** Дан текст: ‘Hello to everybody’. Вставить слова with heartness чтобы получить “Hello with heartness to everybody”

***Листинг программы:***

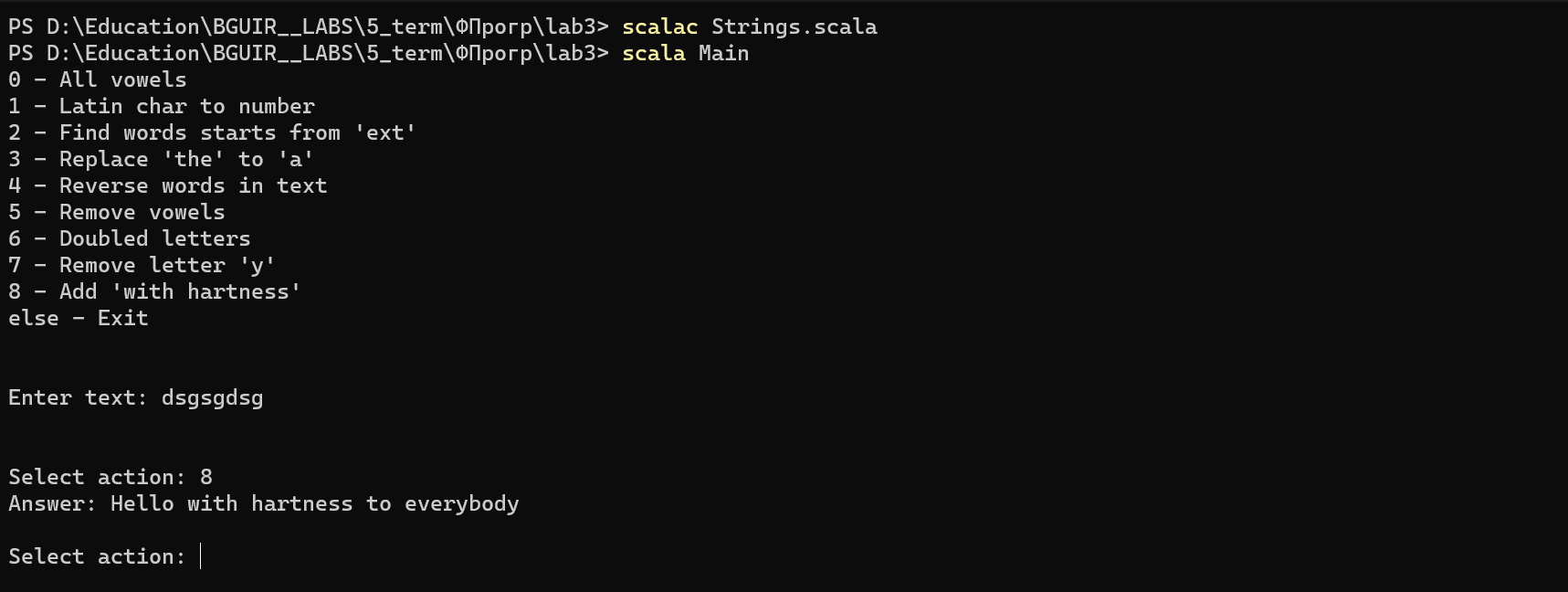
*case 8 => {*

*val insertion = "with hartness "*

*print(msg.substring(0, 6) + insertion + msg.substring(6, msg.length))*

*}*

***Результат работы программы:***



*Рисунок 8.1 – Результат работы задания 8*

1. **Вывод**

В данной лабораторной работе были освоены навыки работы в Scala, была изучена техника работы со строками в Scala. Были выполнены все поставленные задачи: написан код программы к выданному варианту и приведены результаты работы всех частей программы.