Министерство образования Республики Беларусь

Учреждение высшего образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ОТЧЕТ

по лабораторной работе №3

на тему

**Работа со строками**

|  |  |
| --- | --- |
| Студенты группы 150504  Проверила | Шевцов В.С  Крук А.И  Герман Ю.О. |

Минск 2023

Цель: Изучить технику работы со строками в Scala.

**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:

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)

**Задание**

Индивидуальное: Вывести суммарное число всех гласных в собственном тексте.

object Individual

{

def main(args: Array[String]): Unit = {

var string = scala.io.StdIn.readLine()

val regex = "[eyuioaEYUIOЁAУЕЫАОЭИЮёуеыаоэию]".r

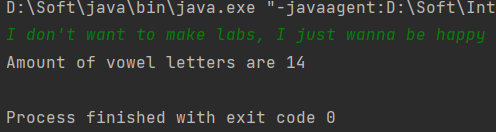
var amount = 0

regex.findAllMatchIn(string).foreach(\_ => amount += 1)

println("Amount of vowel letters are " + amount)

}

}



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

object Main {

def main(args: Array[String]): Unit = {

var str = "1 + 1 = 2"

var regex = "[1-2]".r;

var temp = regex.findAllMatchIn(str).toList

for (i <- temp.indices)

temp.apply(i).toString().toInt match {

case 1 => str = regex.replaceFirstIn(str, "one")

case 2 => str = regex.replaceFirstIn(str, "two")

}

println(str)

}

}



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

object Main2 {

def main(args: Array[String]): Unit = {

val str = "Being quiet buetiful girl she never thought of buety"

val regex = "bue(\\w+)".r

regex.findAllMatchIn(str).foreach(println)

}

}



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

object Main3 {

def main(args: Array[String]): Unit = {

var str = "one plus one is something"

val regex = "\\bone\\b".r

str = regex.replaceAllIn(str, "two")

println(str)

}

}



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

object Main4 {

def main(args: Array[String]): Unit = {

var str = "world is nice"

val regex1 = "\\w+".r

val regex2 = "\\w+$".r

var word1 = regex1.findFirstIn(str).get

var word2 = regex2.findFirstIn(str).get

str = regex2.replaceFirstIn(str, word1)

str = regex1.replaceFirstIn(str, word2)

println(str)

}

}



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

object Main5 {

def main(args: Array[String]): Unit = {

var str = "Hello to all my friends"

val regex = "[aeiouy]".r

str = regex.replaceAllIn(str, "")

println(str)

}

}



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

object Main6 {

def main(args: Array[String]): Unit = {

val str = "Hello to everybody"

var str2 = ""

for (i <- str.indices)

if (i % 2 == 0)

str2 = str2.appended(str.apply(i))

println(str2)

}

}



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

object Main7 {

def main(args: Array[String]): Unit = {

val str = "Hello to everybody"

val regex = "[e]".r

println(regex.replaceAllIn(str, ""))

}

}



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

object Main8 {

def main(args: Array[String]): Unit = {

val str = "Be healthy"

var regex = " ".r

println(regex.replaceFirstIn(str, " always "))

}

}



**Заключение**

В ходе лабораторной работы была изучена техника работы со строками в Scala.