БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе №3

Тема: «Работа со строками»

Вариант 2

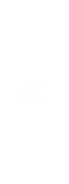
Проверила:

Герман Ю.О.

Выполнили:

Климович А.Н.

Тамашеня В.В.

Минск 2023



**1 ЦЕЛЬ**

Техника работы со строками в Scala.

**2 КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ**

Работа со строками в Scala представляет собой важную часть программирования на этом языке. Строки в Scala являются неизменяемыми объектами, которые могут содержать любые символы, включая буквы, цифры, пробелы и специальные символы.

Для работы со строками в Scala доступны различные методы, которые позволяют выполнять различные операции над строками. Например, методы concat() и **+** позволяют объединять строки, метод length() возвращает длину строки, метод charAt() возвращает символ по указанному индексу, метод substring() возвращает подстроку, начиная с указанного индекса и заканчивая указанным индексом.

Scala также поддерживает использование регулярных выражений для работы со строками. Регулярные выражения позволяют выполнять поиск и замену текста в строках, а также проверять соответствие строки заданному шаблону.

Кроме того, Scala предоставляет удобный синтаксис для форматирования строк. Для этого используется метод format(), который позволяет задавать шаблон форматирования и значения, которые будут подставляться в этот шаблон.

Наконец, Scala также поддерживает многоязыковые строки, которые позволяют работать с текстом на разных языках. Для этого используется специальный синтаксис, который позволяет задавать строки с использованием специальных символов, обозначающих язык.

В целом, работа со строками в Scala представляет собой важную и неотъемлемую часть программирования на этом языке, и знание основных методов и приемов работы со строками является необходимым для разработки качественного и эффективного кода.

**3 ХОД РАБОТЫ**

**3.1 Анализ примеров**

**3.1.1** Функция replaceAll используется для замены всех вхождений строки другой строкой.

Пример 1:

val str = "Hello, World!"

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

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

Результат выполнения данного примера представлен на рисунке 3.1:

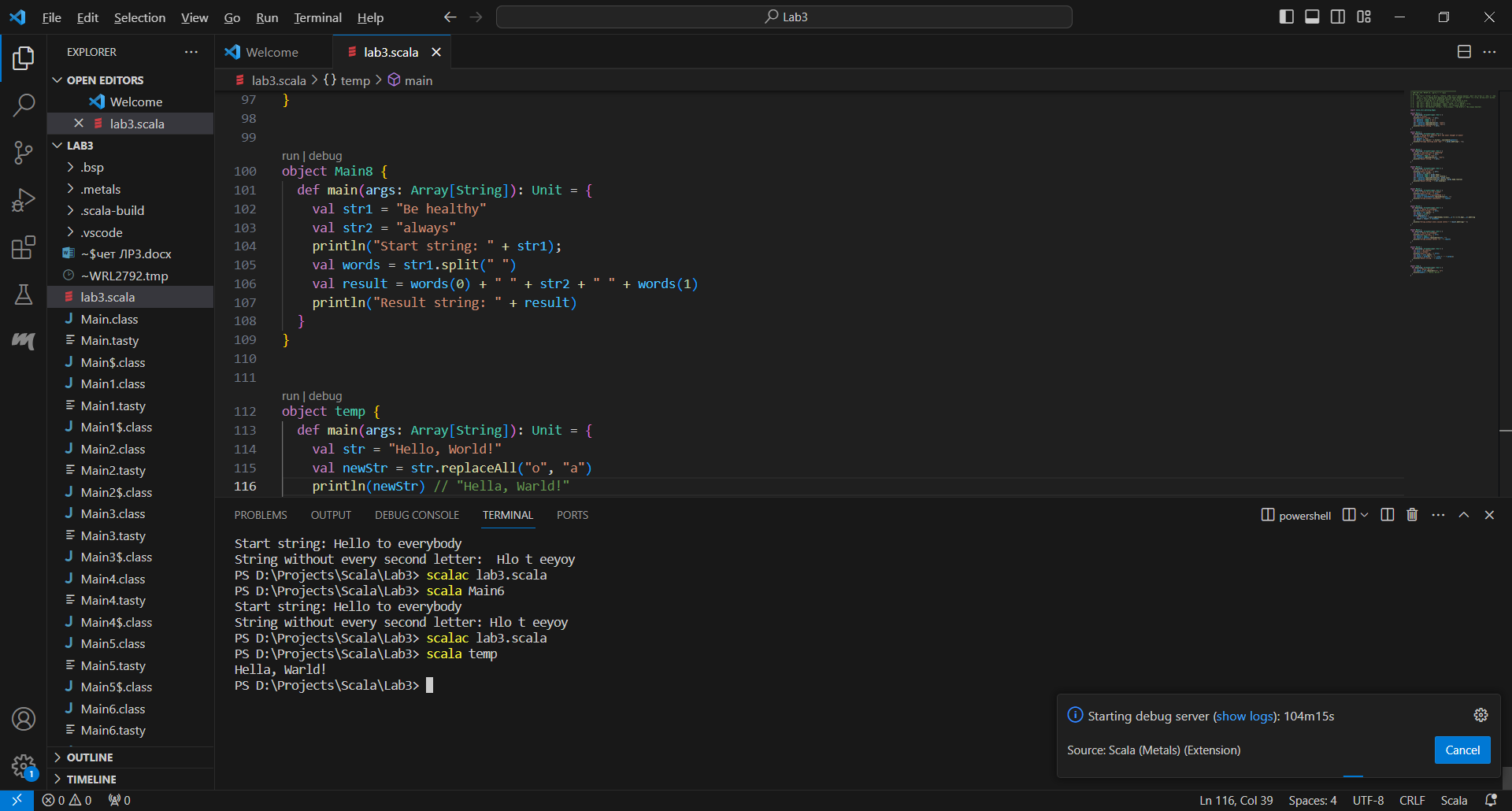


Рисунок 3.1 – Результат выполнения примера 1

**3.1.2** Функция split() используется для разделения строки на массив подстрок на основе разделителя.

Пример 2:

val str = "apple,banana,orange"

val arr = str.split(",")

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

Результат выполнения данного примера представлен на рисунке 3.2:

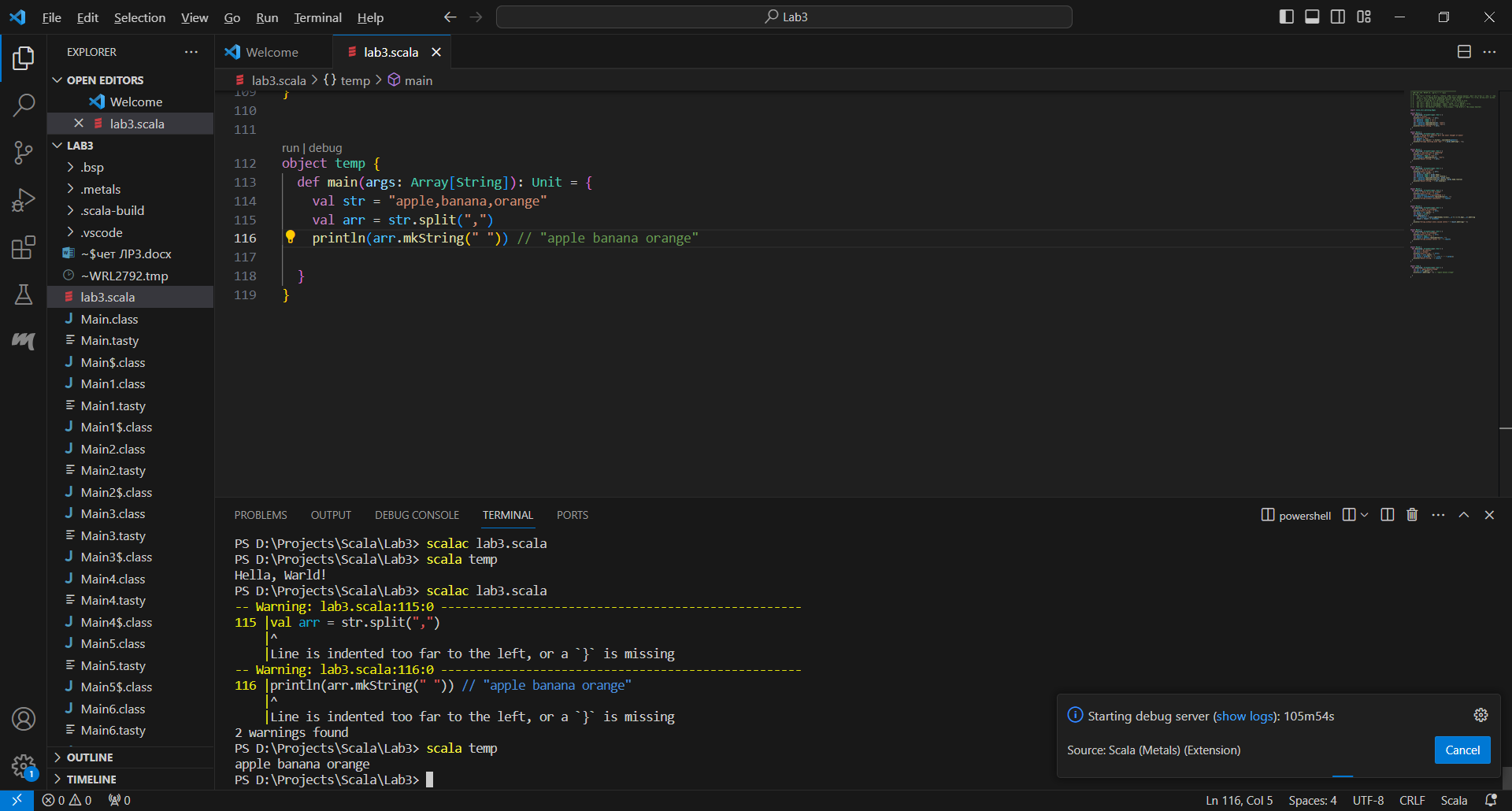


Рисунок 3.2 – Результат выполнения примера 2

**3.1.3** Функции startsWith() и endsWith() проверяют, начинается ли или заканчивается ли строка заданной подстрокой.

Далее представлен код примера 3:

val str = "Hello, World!"

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

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

Результат выполнения данного примера представлен на рисунке 3.3.

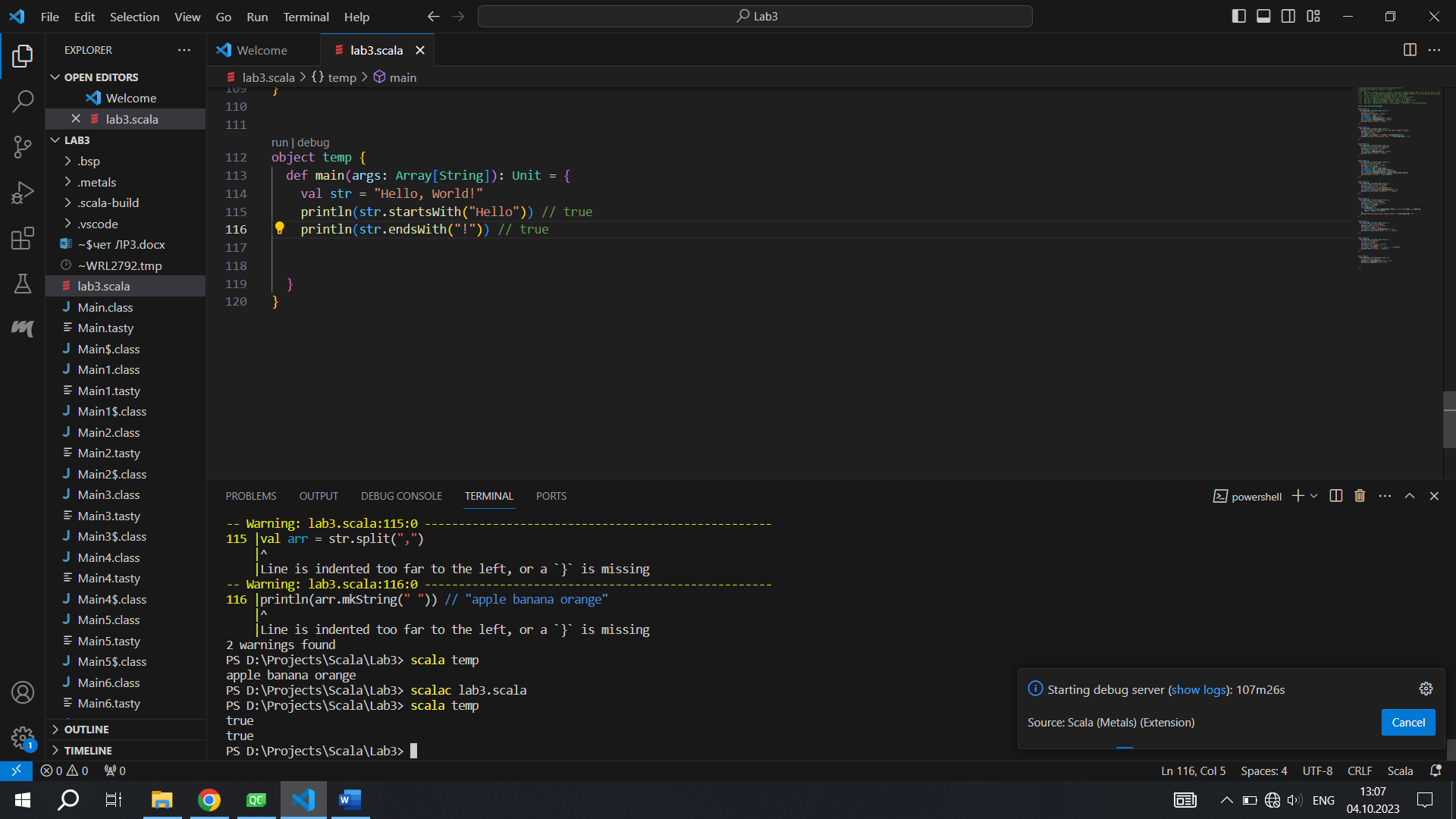


Рисунок 3.3 – Результат выполнения примера 3

**3.1.4** Функция substring() выделяет подстроку из строки.

Далее представлен код примера 4:

val str = "Hello, World!"

val subStr = str.substring(7, 12)

println(subStr) // "World"

Результат выполнения данного примера представлен на рисунке 3.4.

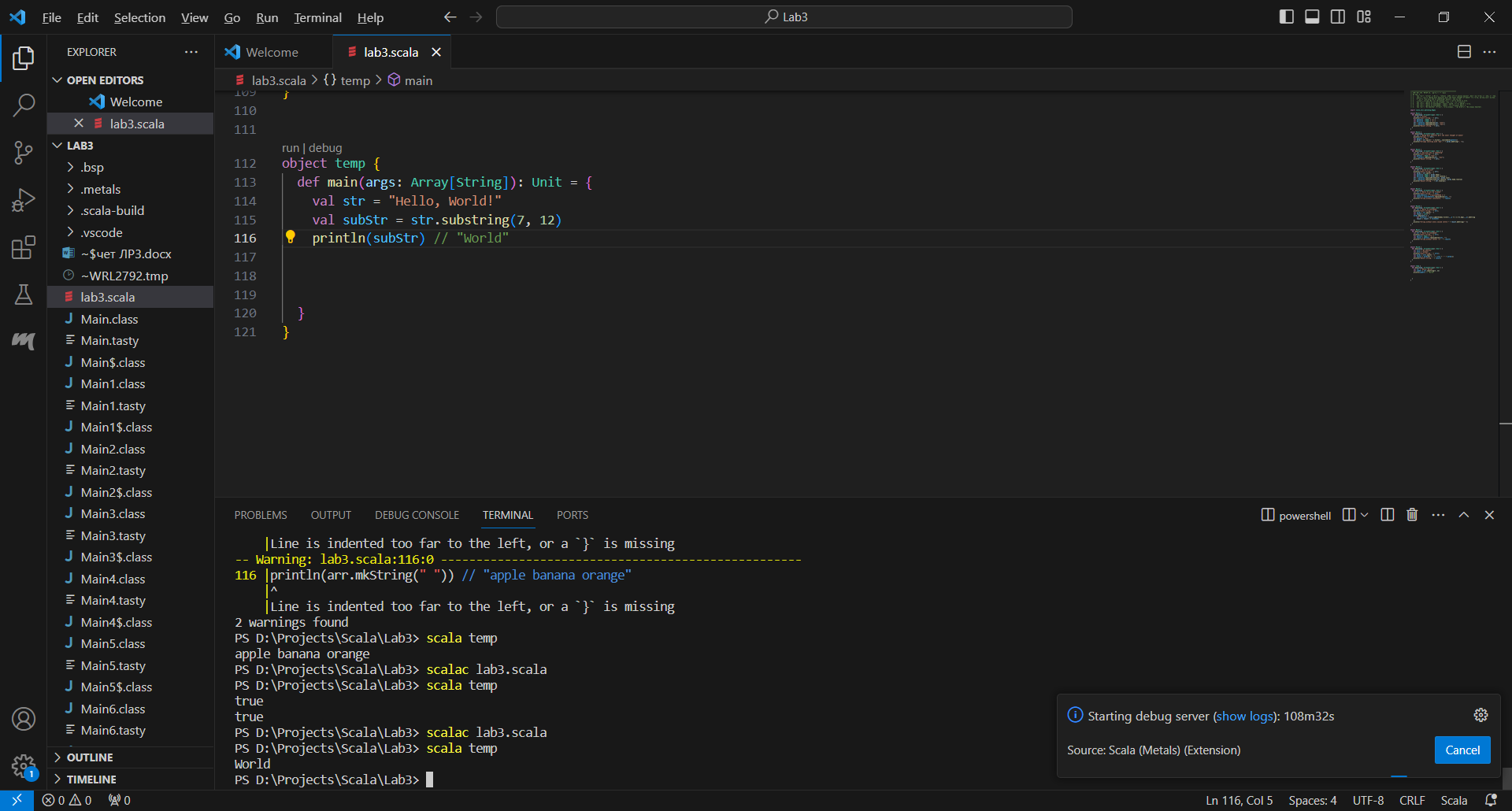


Рисунок 3.4 – Результат выполнения примера 4

**3.1.5** Функция toCharArray() преобразует строку в массив символов.

Далее представлен код примера 5:

val str = "Hello, World!"

val arr = str.toCharArray()

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

Результат выполнения данного примера представлен на рисунке 3.5.

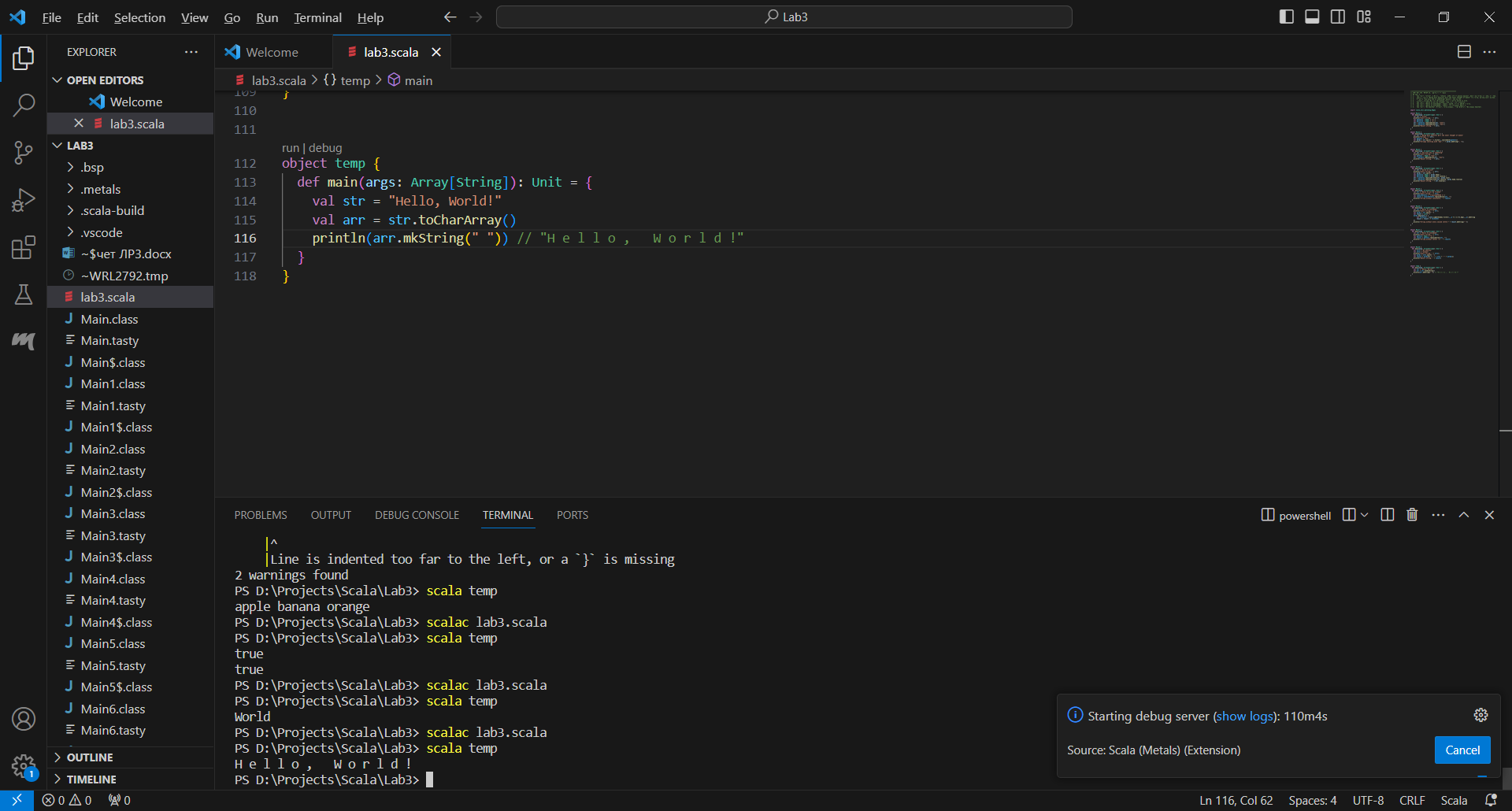


Рисунок 3.5 – Результат выполнения примера 5

**3.1.6** Функции toLoweCase и toUpperCase преобразуют символы строки в верхний и нижний регистры соответственно.

Далее представлен код примера 6:

val str = "Hello, World!"

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

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

Результат выполнения данного примера представлен на рисунке 3.6.

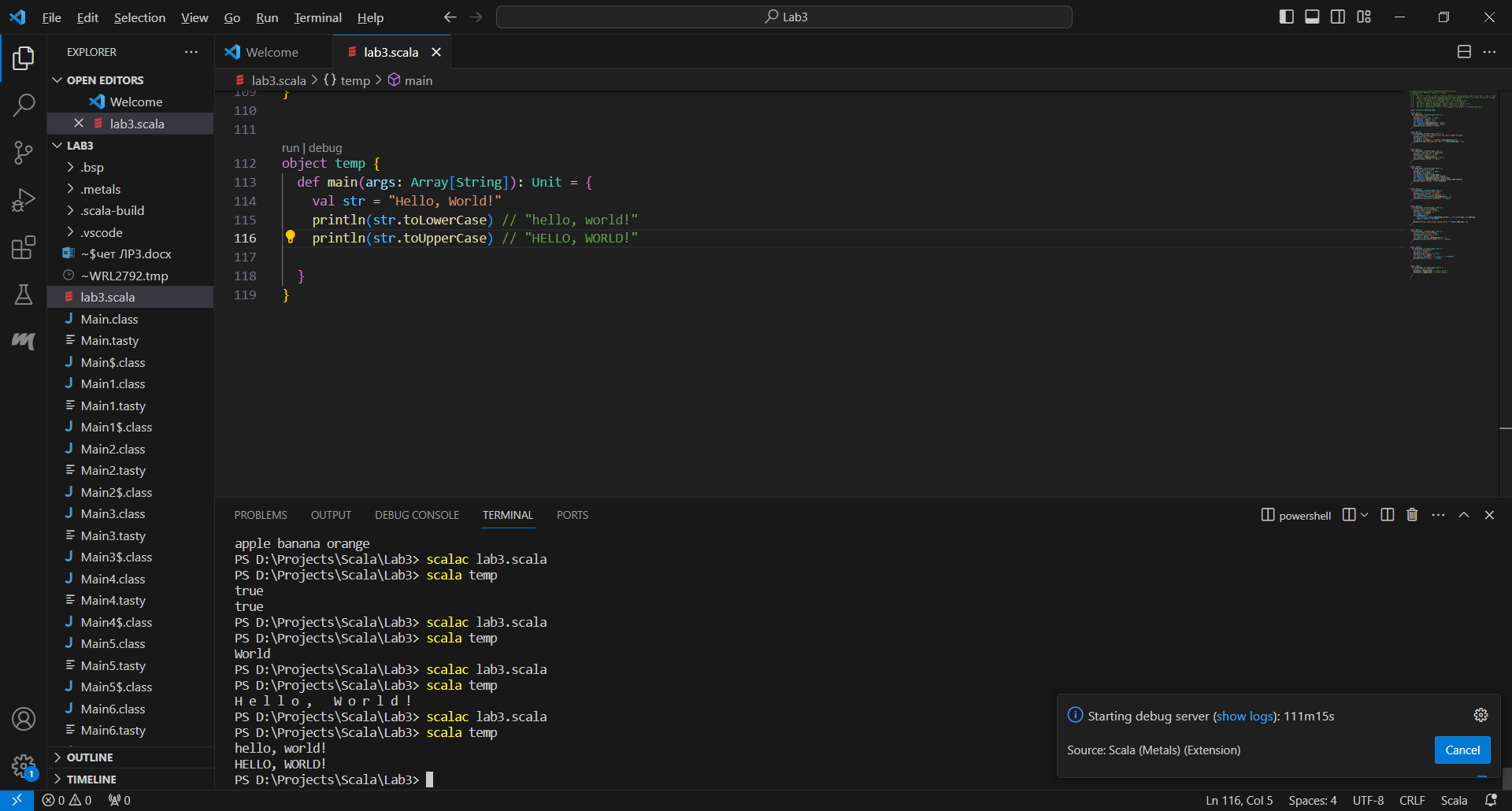


Рисунок 3.6 – Результат выполнения примера 6

**3.1.7** Функция trim() отсекает концевые пробелы.

Далее представлен код примера 7:

val str = " Hello, World! "

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

Результат выполнения данного примера представлен на рисунке 3.7.

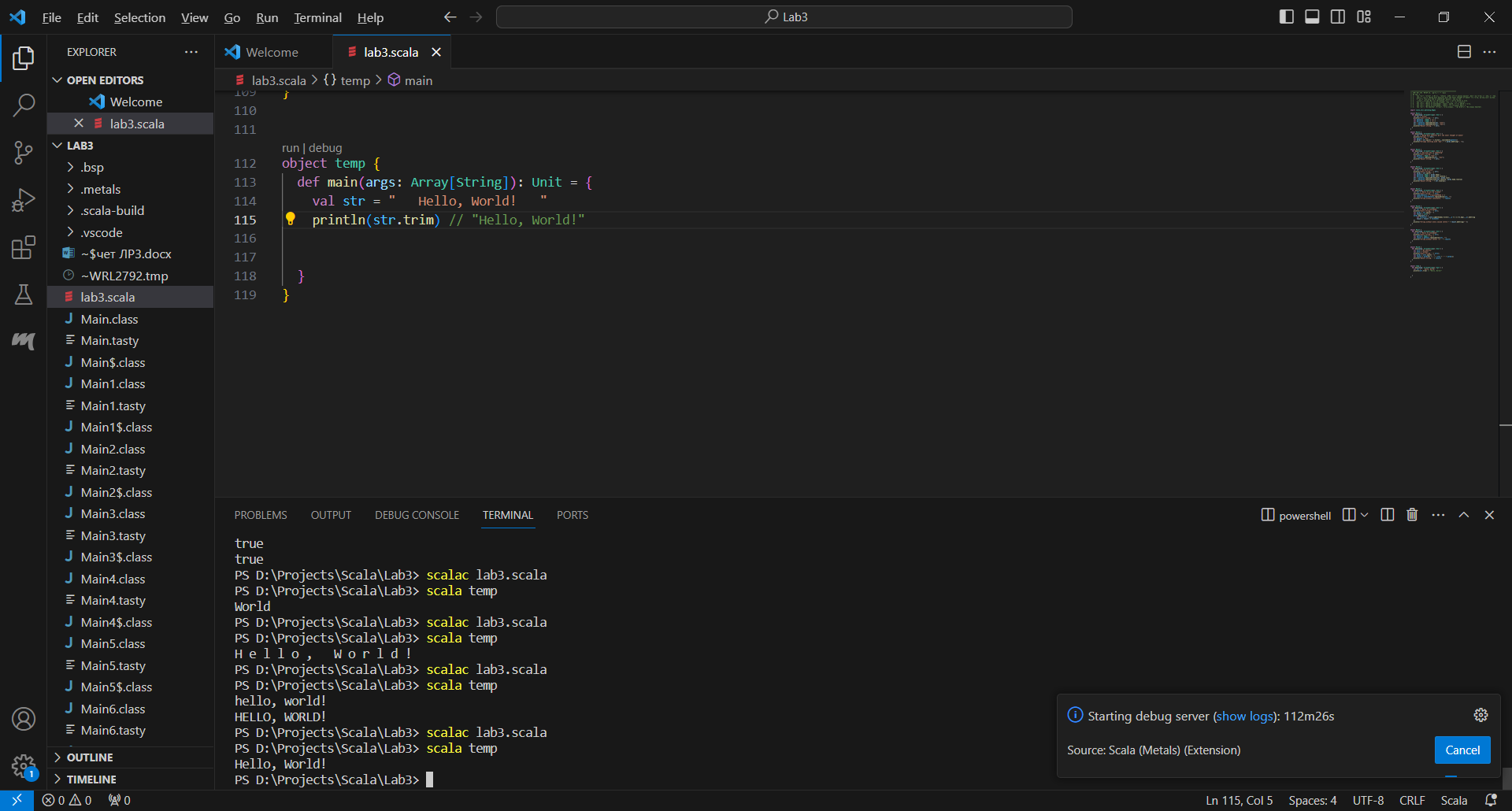


Рисунок 3.7 – Результат выполнения примера 7

**3.1.8** Функции indexOf() и lastIndexOf получает первый и последний индекс подстроки в строке.

Далее представлен код примера 8:

val str = "Hello, World!"

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

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

Результат выполнения данного примера представлен на рисунке 3.8.

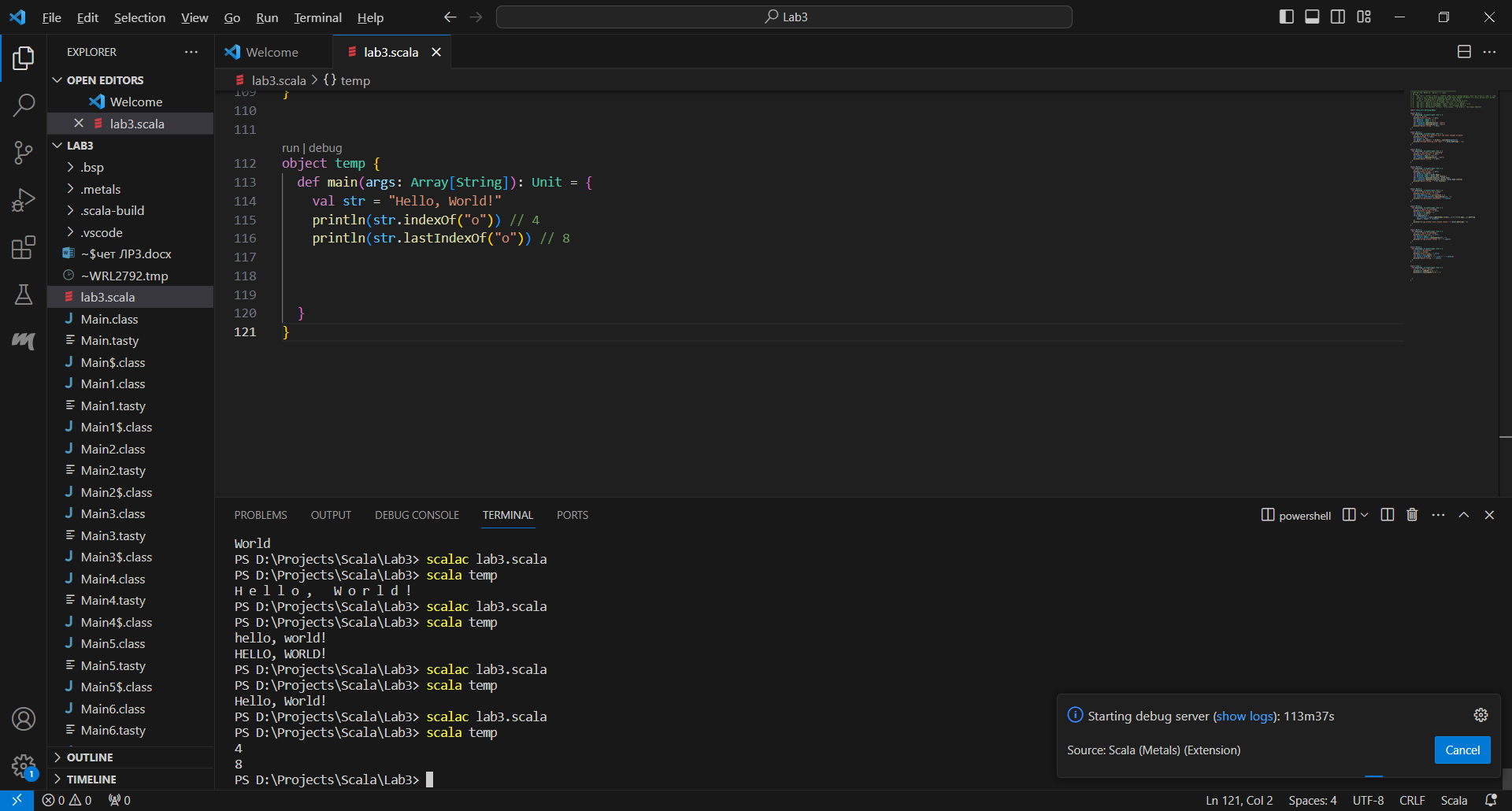


Рисунок 3.8 – Результат выполнения примера 8

**3.1.9** Функция charAt() определяет символ, стоящий на указанной позиции.

Далее представлен код примера 9:

val str = "Hello, World!"

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

Результат выполнения данного примера представлен на рисунке 3.9.

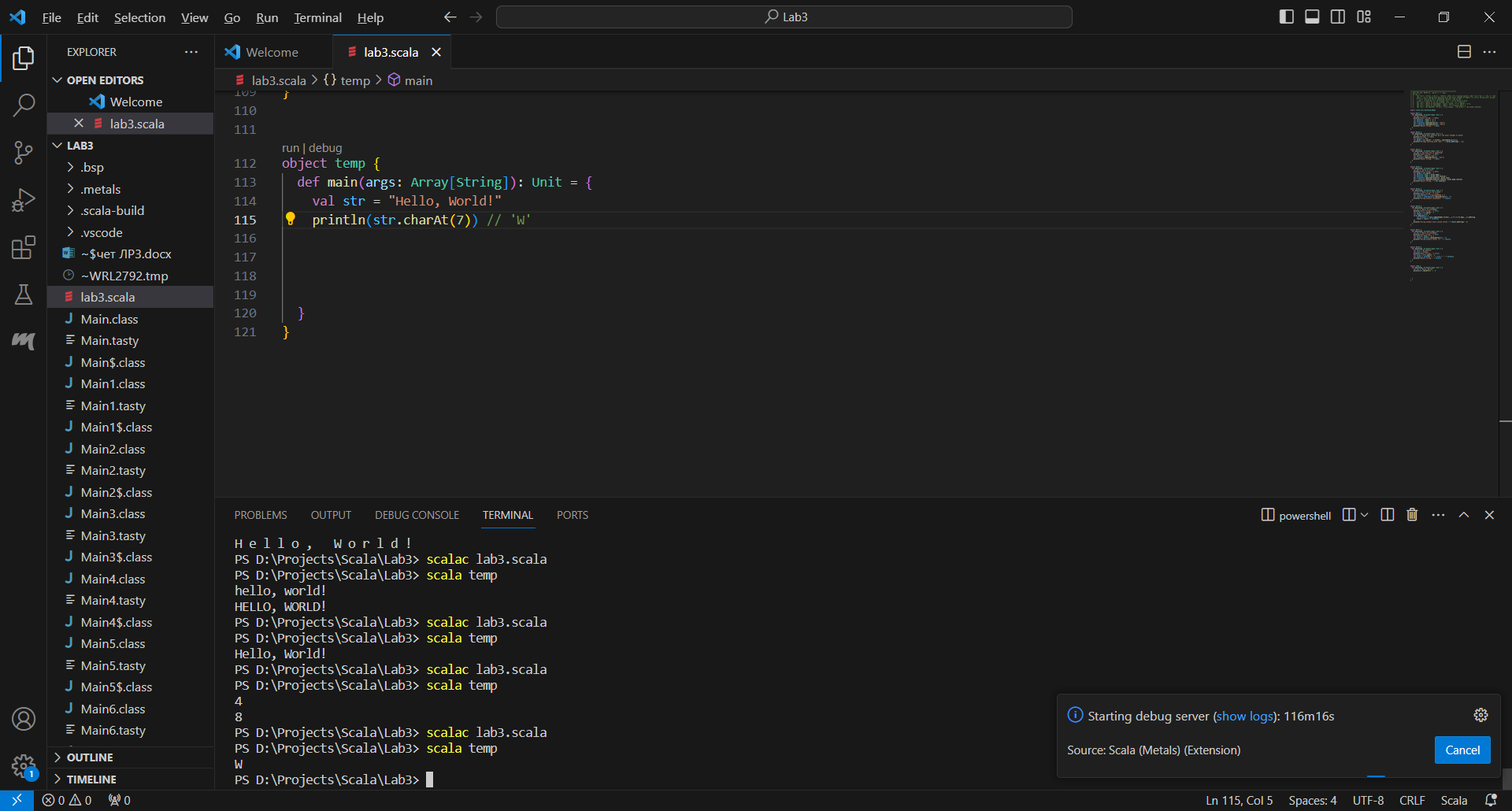


Рисунок 3.9 – Результат выполнения примера 9

**3.1.10** Функция stripMargin() удаляет ведущие пробелы перед строкой.

Далее представлен код примера 10:

val str =

"""

|Hello,

|World!

|""".stripMargin

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

Результат выполнения данного примера представлен на рисунке 3.10.

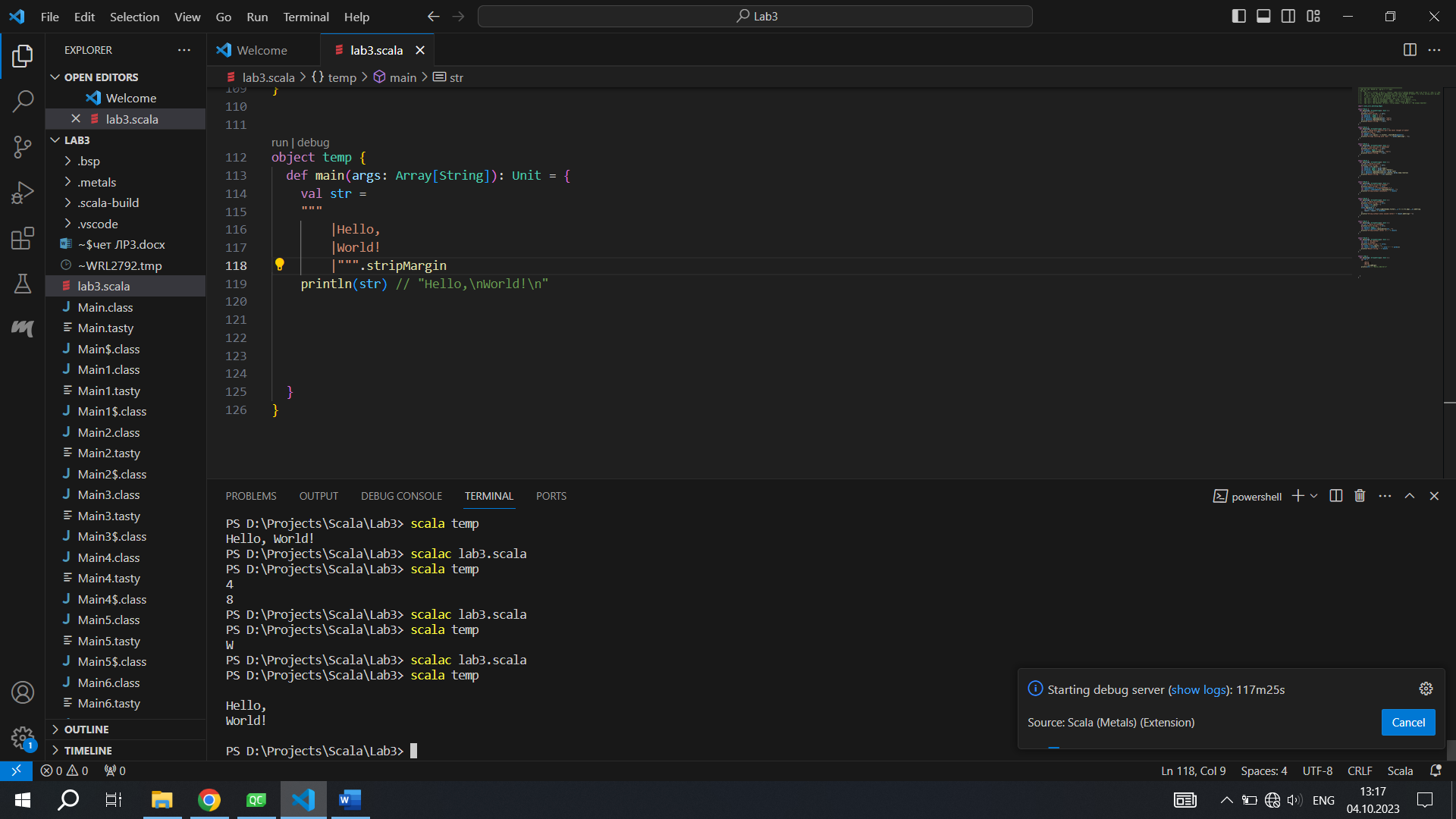


Рисунок 3.10 – Результат выполнения примера 10

**3.1.11**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.

**3.2 Выполнение заданий**

**3.2.1 Задание для всех**

Вывести суммарное число всех гласных в собственном тексте.

object Main {

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

val str = "Hello from Alexei and Vlad!"

val pattern = "[aeiouAEIOU]".r

val result = pattern.findAllIn(str).toList.length

println(“Result: ” + result)

}

}

Результат выполнения общего представлен на рисунке 3.11.

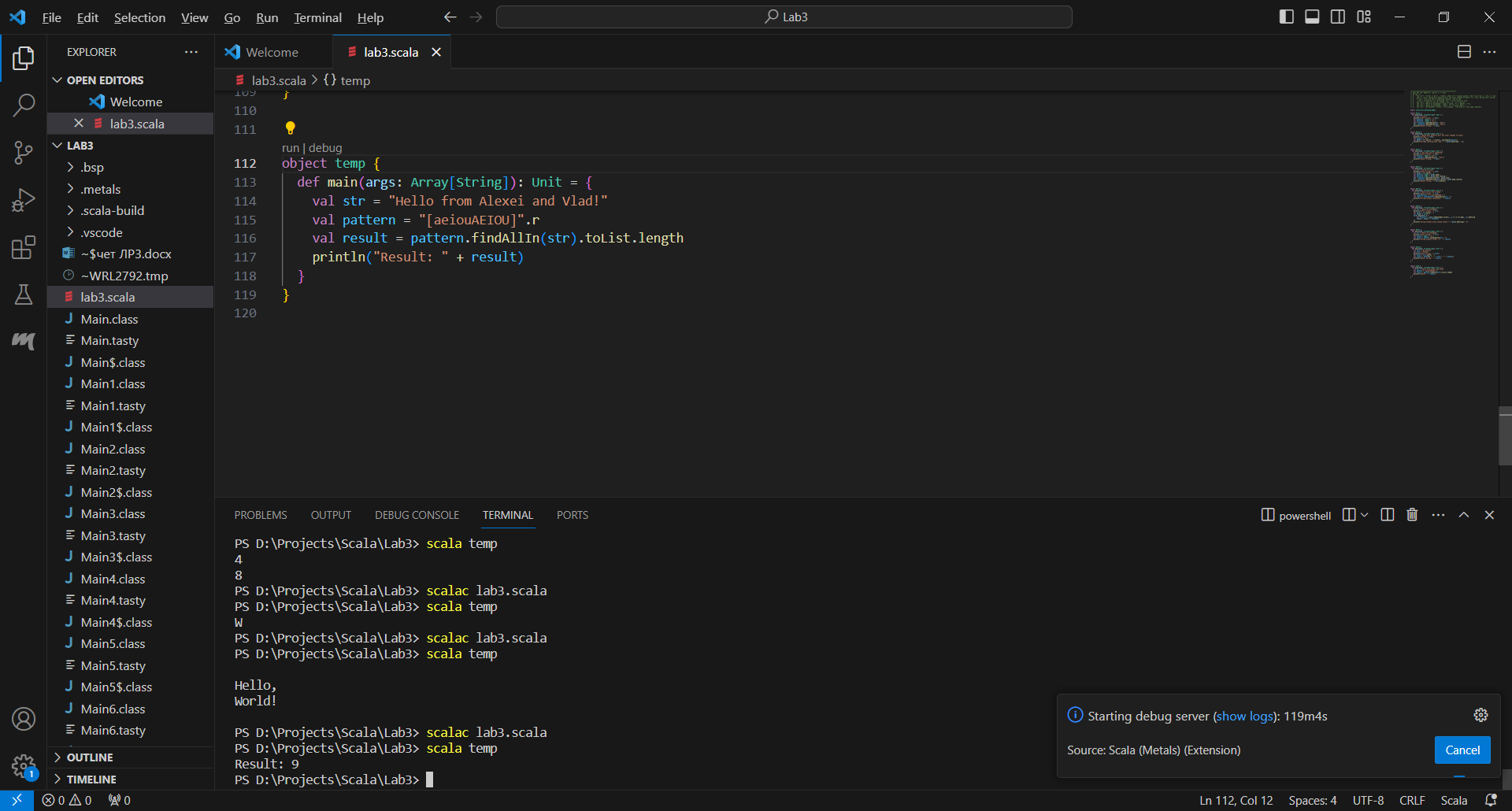


Рисунок 3.11 – Результат выполнения общего задания

**3.2.2 Задание варианта 2**

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

Далее представлен код программы задания 1:

import scala.util.matching.Regex

object Main1 {

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

var str = "1+1=2"

println("Start string: " + str);

val pattern1 : Regex = "1".r

val pattern2 : Regex = "2".r

str = pattern1.replaceAllIn(str, "one");

str = pattern2.replaceAllIn(str, "two");

println("Result string: " + str);

}

}

Результат выполнения задания 1 представлен на рисунке 3.12.

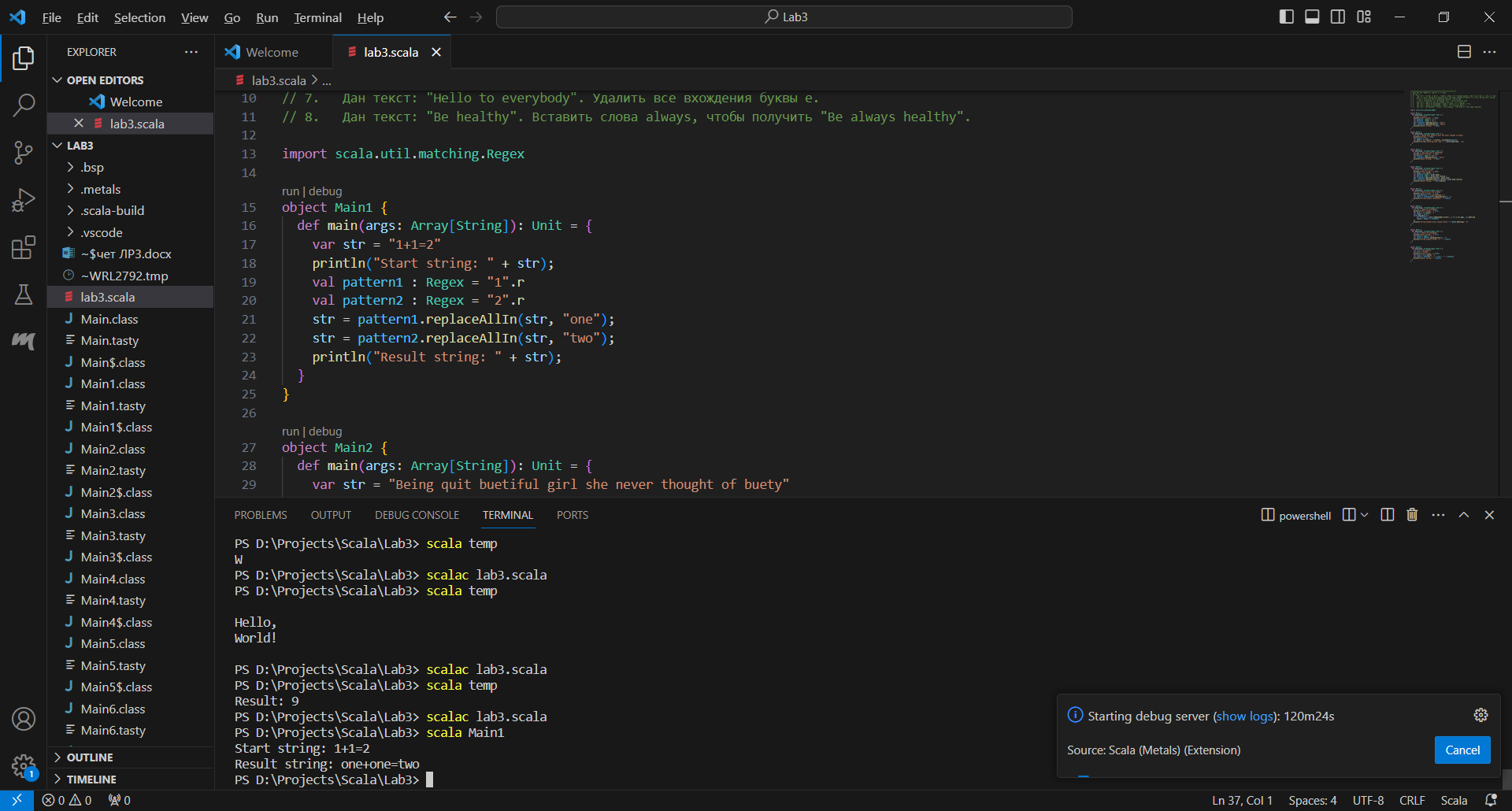


Рисунок 3.12 – Результат выполнения задания 1

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

Далее представлен код программы задания 2:

object Main2 {

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

var str = "Being quit buetiful girl she never thought of buety"

println("String: " + str);

val pattern = "bue"

var words = str.split(" ").filter(\_.startsWith(pattern))

println("Strings starting with 'bue': " + words.mkString(", "));

}

}

Результат выполнения задания 2 представлен на рисунке 3.13.

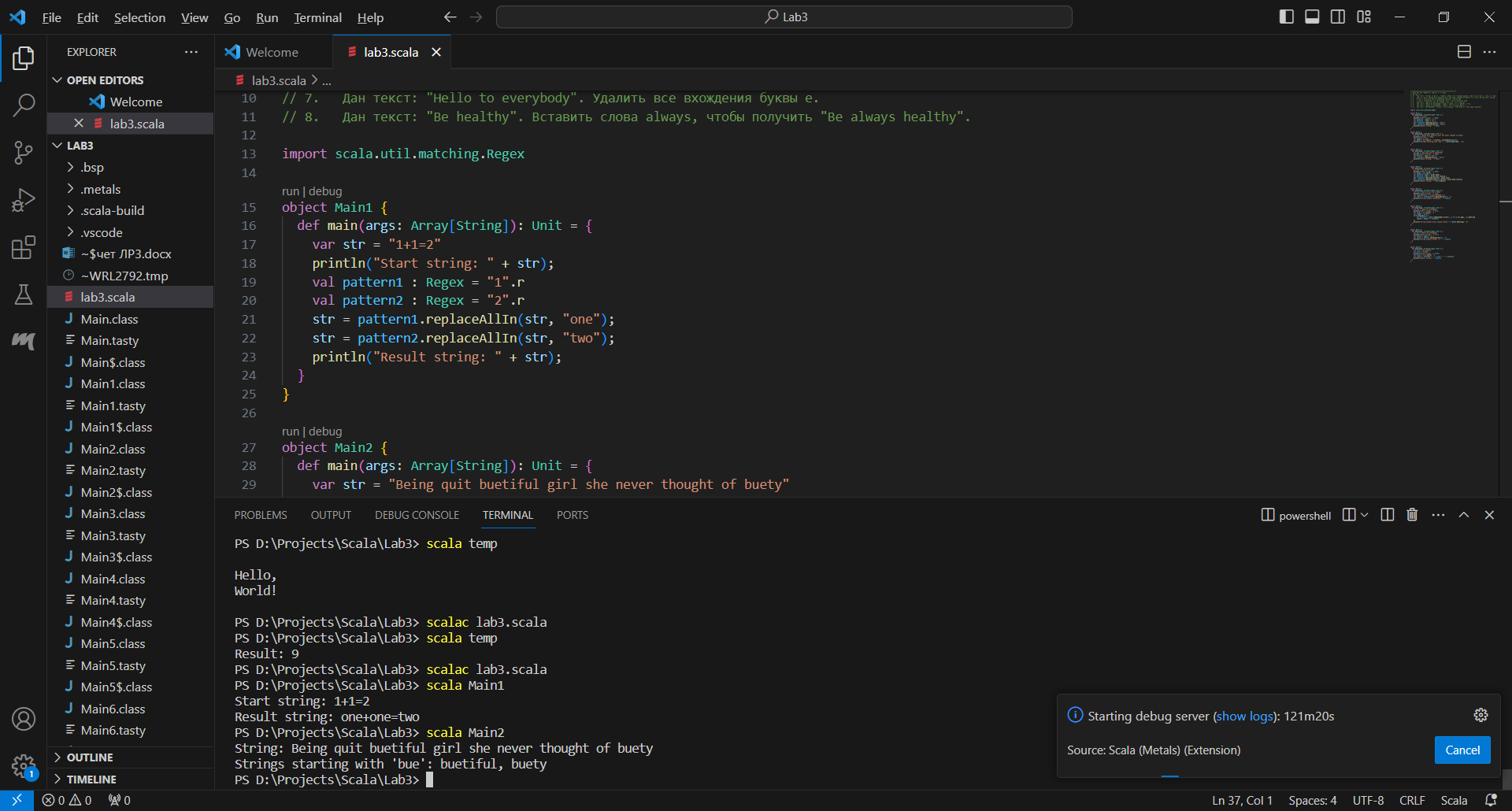


Рисунок 3.13 – Результат выполнения задания 2

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

Далее представлен код программы задания 3:

object Main3 {

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

var str = "one plus one is something"

println("Start string: " + str);

val pattern : Regex = "one".r

str = pattern.replaceAllIn(str, "two");

println("Result string: " + str);

}

}

Результат выполнения задания 3 представлен на рисунке 3.14.

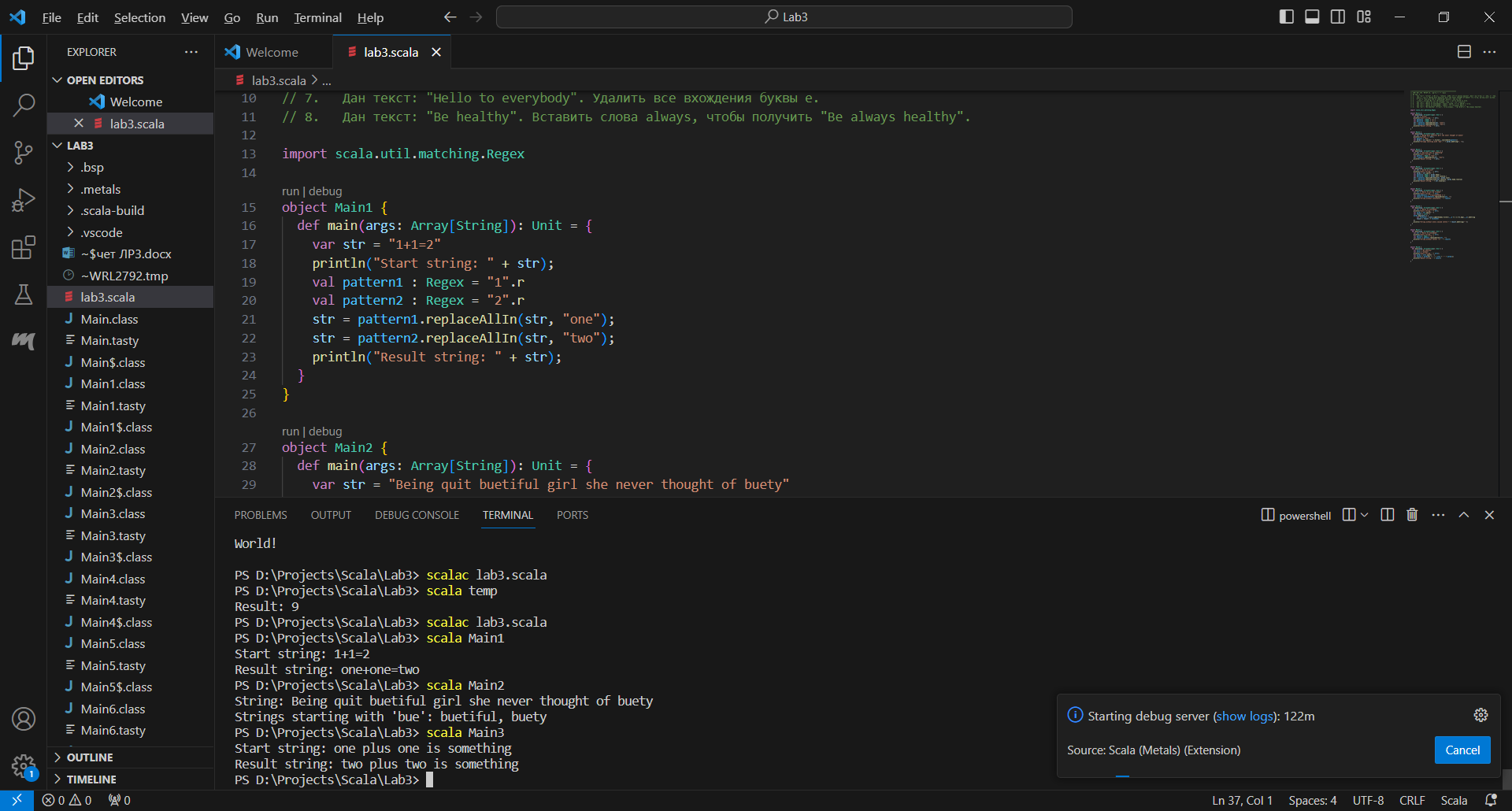


Рисунок 3.14 – Результат выполнения задания 3

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

Далее представлен код программы задания 4:

object Main4 {

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

var str = "world is nice"

println("Start string: " + str);

var words = str.split(" ")

val pattern1: Regex = words.head.r

val pattern2: Regex = words.last.reverse.r

str = pattern1.replaceFirstIn(str, words.last)

str = pattern2.replaceFirstIn(str.reverse, words.head.reverse)

println("Result string: " + str.reverse);

}

}

Результат выполнения задания 4 представлен на рисунке 3.15.

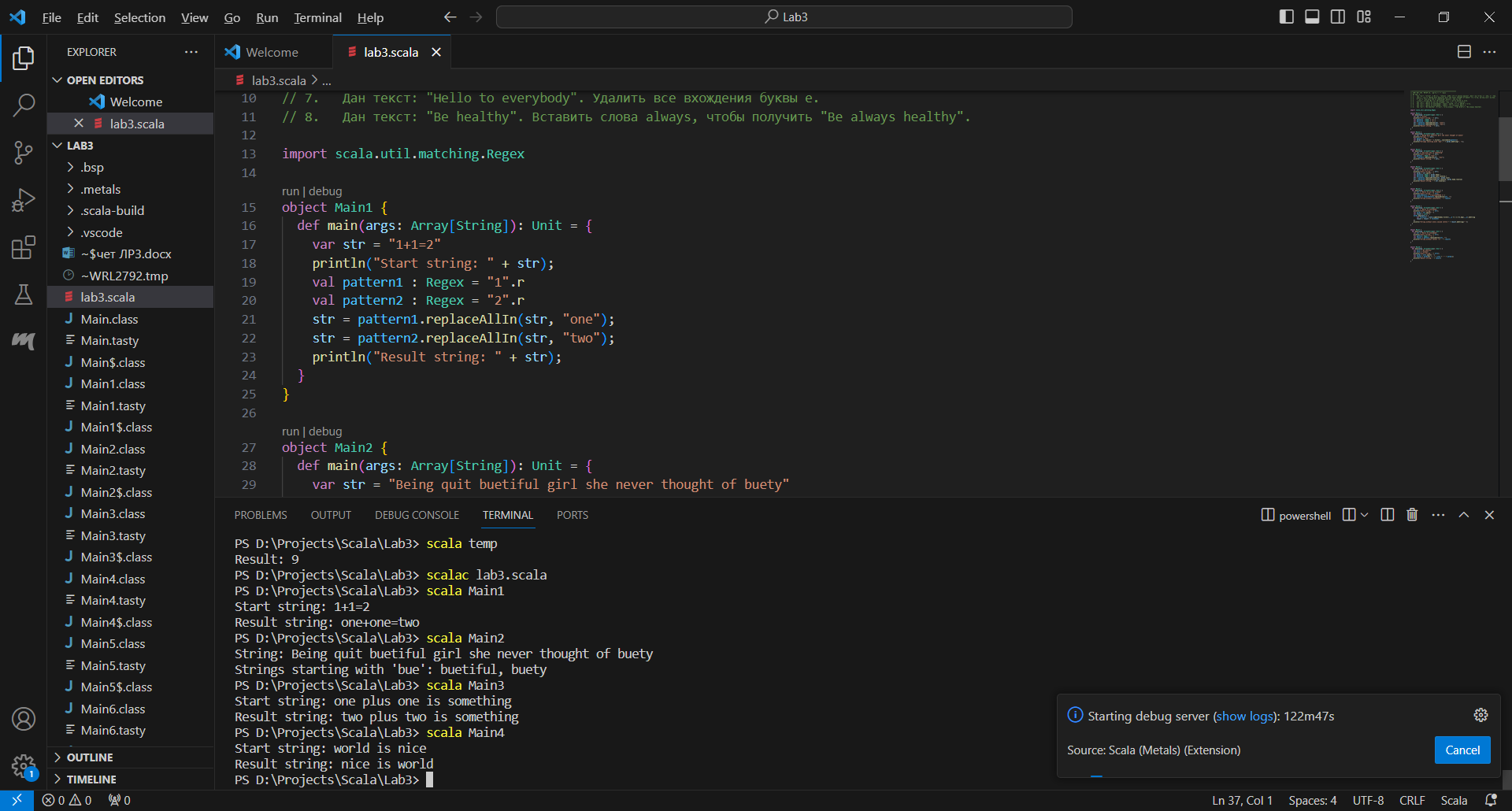


Рисунок 3.15 – Результат выполнения задания 4

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

Далее представлен код программы задания 5:

object Main5 {

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

val str = "Hello to all my friends"

println("Start string: " + str);

val vowelsPattern = "[^eyuioaEYUIOA ]".r

val result = vowelsPattern.replaceAllIn(str, "")

println("String without consonants: " + result)

}

}

Результат выполнения задания 5 представлен на рисунке 3.16.

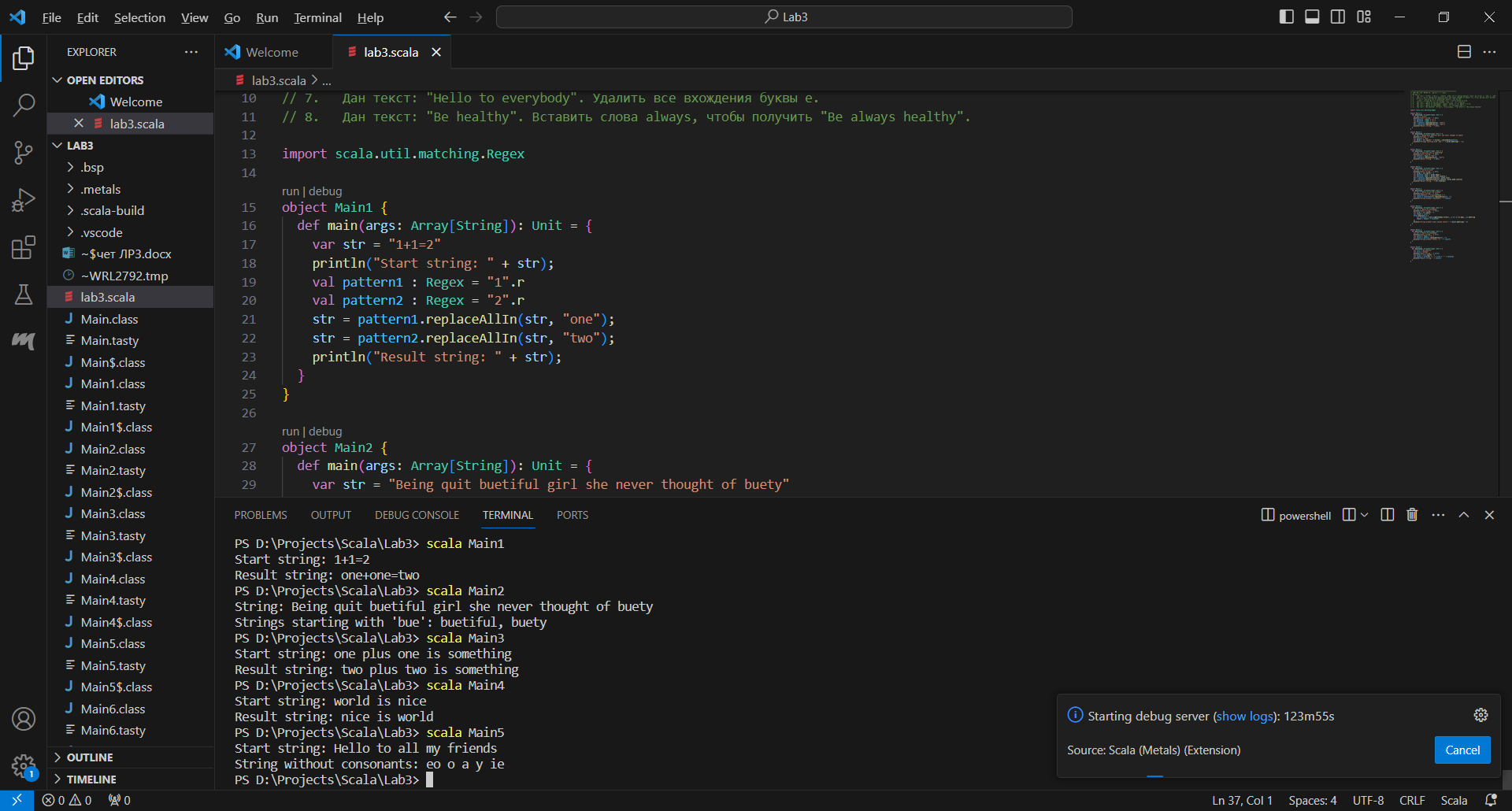


Рисунок 3.16 – Результат выполнения задания 5

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

Далее представлен код программы задания 6:

object Main6 {

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

val str = "Hello to everybody"

println("Start string: " + str);

val words = str.split(" ")

var result = List("")

words.map(word => {

val filterStr = word.zipWithIndex.filter(\_.\_2 % 2 == 0).map(\_.\_1).mkString

result = result :+ filterStr

})

println("String without every second letter:" + result.mkString(" "))

}

}

Результат выполнения задания 6 представлен на рисунке 3.17.

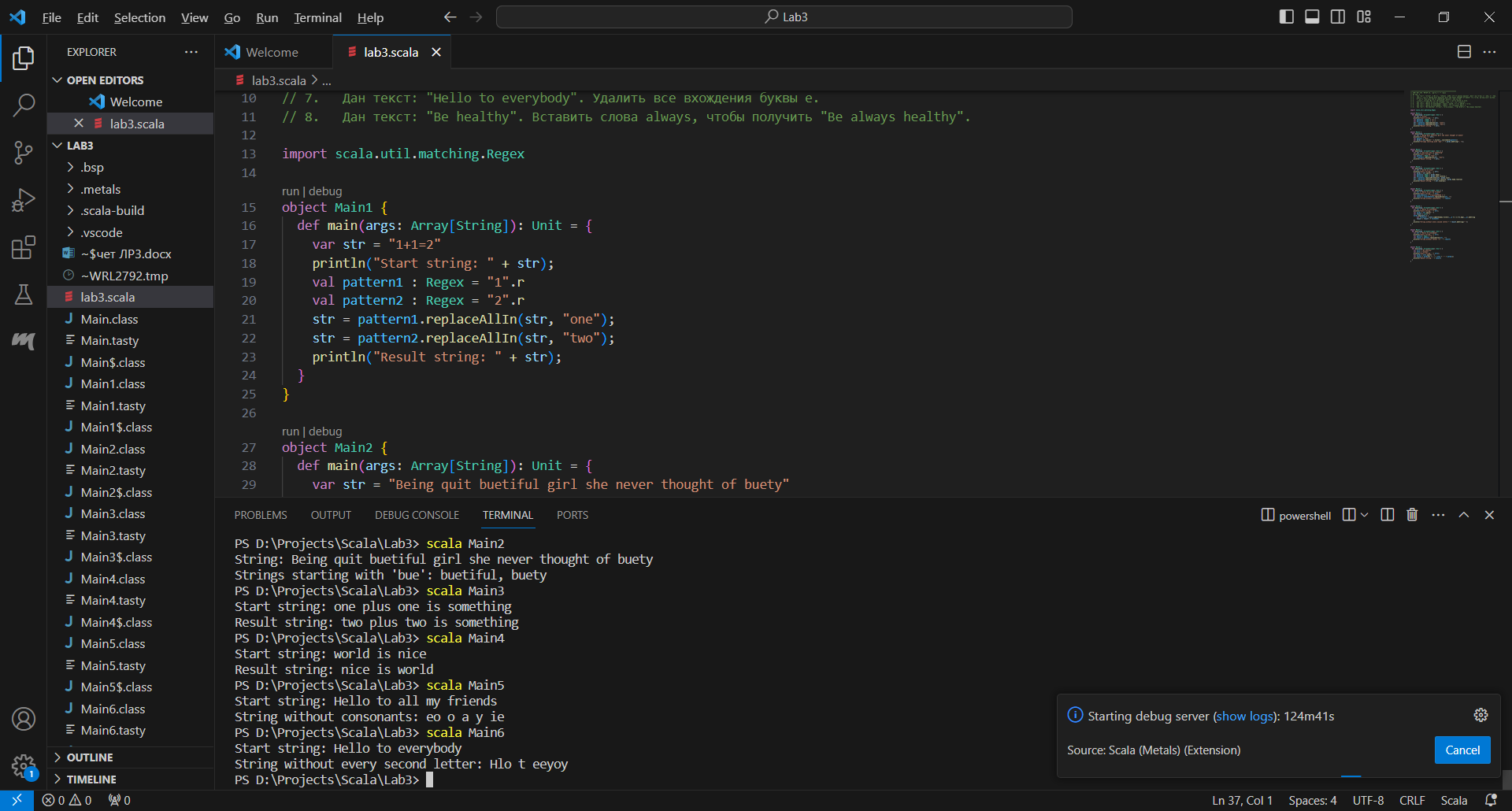


Рисунок 3.17 – Результат выполнения задания 6

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

Далее представлен код программы задания 7:

object Main7 {

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

val str = "Hello to everybody"

println("Start string: " + str);

val pattern: Regex = "e".r

val result = pattern.replaceAllIn(str, "")

println("String without letter 'e': " + result)

}

}

Результат выполнения задания 7 представлен на рисунке 3.18.

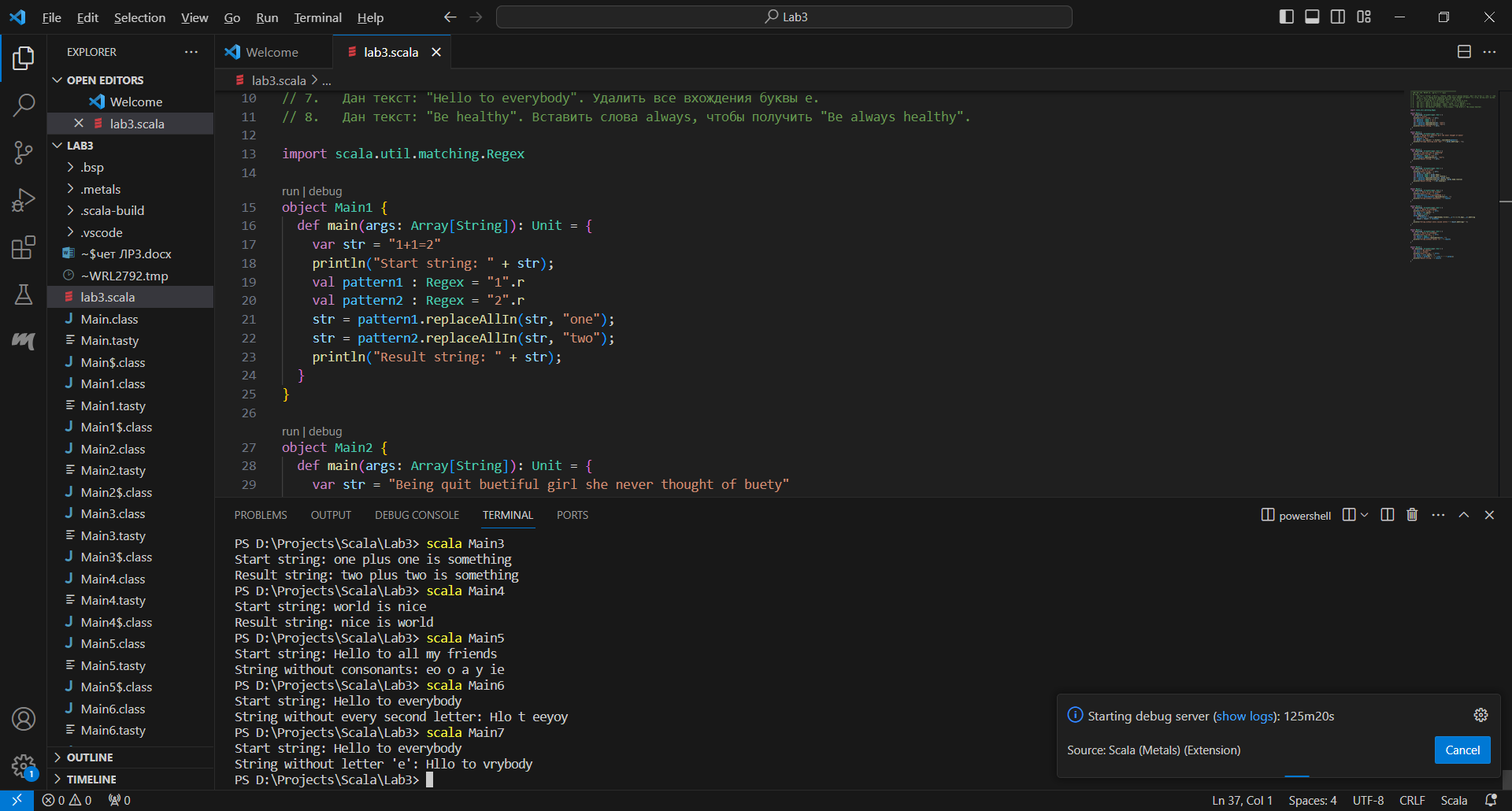


Рисунок 3.18 – Результат выполнения задания 7

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

Далее представлен код программы задания 8:

object Main8 {

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

val str1 = "Be healthy"

val str2 = "always"

println("Start string: " + str1);

val words = str1.split(" ")

val result = words(0) + " " + str2 + " " + words(1)

println("Result string: " + result)

}

}

Результат выполнения задания 8 представлен на рисунке 3.19.

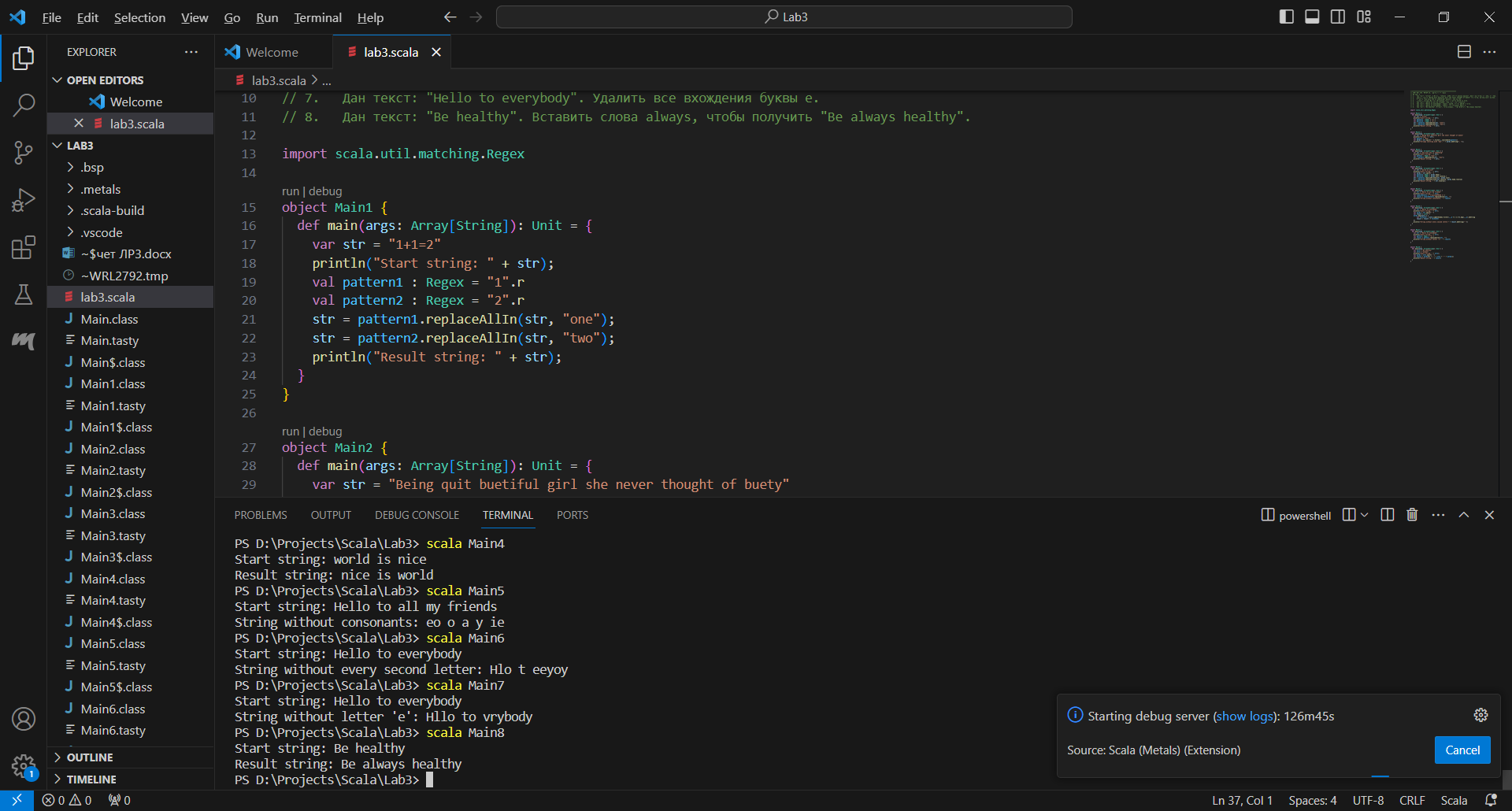


Рисунок 3.19 – Результат выполнения задания 8

**4 ВЫВОД**

Изучили технику работу со строками в Scala.