|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений.**

**Отчет**

**по лабораторной работе № 7**

**Вариант № 12**

**Название:** Строки и регулярные выражения

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | Т.М. Курохтин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Цель:** изучить и освоить принципы работы со строками и регулярными выражениями в Java.

**Задание 1:** В каждом слове текста k-ю букву заменить заданным символом. Если k больше длины слова, корректировку не выполнять.

Код программы:

/\*

\\b — начало слова.

\\w{${k-1}} — первые k-1 букв в слове.

(\\w) — k-я буква в слове.

\*/

fun replaceKthLetter(text: String, k: Int, replacement: Char): String {

val regex = Regex("\\b\\w{${k-1}}(\\w)")

return regex.replace(text) { matchResult ->

val word = matchResult.value

if (word.length >= k) {

word.substring(0, k - 1) + replacement + word.substring(k)

} else {

word

}

}

}

fun main() {

val text = "Each word in this text will be processed."

val k = 3

val replacement = '\*'

val result = replaceKthLetter(text, k, replacement)

println(result)

}

Результат работы программы представлен на рисунке 1.

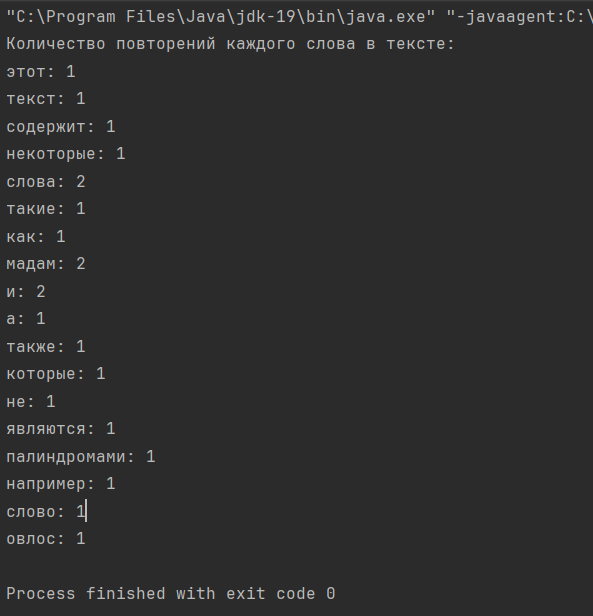


Рисунок 1 – Результат работы программы

**Задание 2:** В русском тексте каждую букву заменить ее порядковым номером в алфавите. При выводе в одной строке печатать текст с двумя пробелами между буквами, в следующей строке внизу под каждой буквой печатать ее номер.

Код программы:

fun charToAlphabetPosition(c: Char): Int? {

val alphabet = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"

return alphabet.indexOf(c.lowercaseChar()).takeIf { it >= 0 }?.plus(1)

}

fun replaceLettersWithPosition(text: String): String {

val regex = Regex("[а-яА-ЯёЁ]")

val positions = StringBuilder()

for (char in text) {

if (regex.matches(char.toString())) {

val pos = charToAlphabetPosition(char)

if (pos != null) {

positions.append(pos).append(" ")

} else {

positions.append(" ")

}

} else {

positions.append(" ")

}

}

return positions.toString().trim()

}

fun main() {

val text = "Пример текста на русском языке."

val positions = replaceLettersWithPosition(text)

println(text)

println(positions)

}

Результат работы программы показан на рисунке 2.

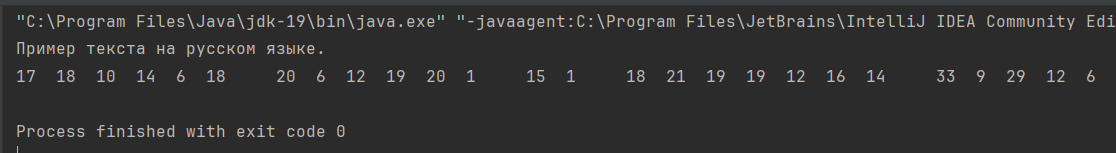


Рисунок 2 – Результат работы программы

**Задание 3:** В тексте найти все пары слов, из которых одно является обращением другого.

Код программы:

fun findReversedWordPairs(text: String): List<Pair<String, String>> {

val regex = Regex("\\b\\w+\\b")

val words = regex.findAll(text).map { it.value }.toList()

val reversedPairs = mutableListOf<Pair<String, String>>()

for (i in words.indices) {

for (j in i + 1..<words.size) {

if (words[i].reversed() == words[j]) {

reversedPairs.add(Pair(words[i], words[j]))

} else if (words[j].reversed() == words[i]) {

reversedPairs.add(Pair(words[j], words[i]))

}

}

}

return reversedPairs

}

fun main() {

val text = "sdfs test drow tset fdsv fdgg asads word"

val pairs = findReversedWordPairs(text)

if (pairs.isNotEmpty()) {

println("Пары слов, одно из которых является обращением другого:")

pairs.forEach { (word1, word2) -> println("$word1 - $word2") }

} else {

println("В тексте не найдены пары слов, одно из которых является обращением другого.")

}

}

Результат работы программы представлен на рисунке 3.

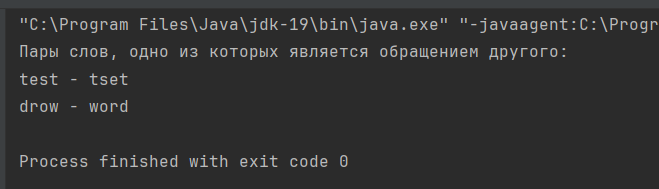
****

Рисунок 3 – Результат работы программы

**Задание 4:** Найти и напечатать, сколько раз повторяется в тексте каждое слово, которое встречается в нем.

Код класса программы:

fun countWordOccurrences(text: String): Map<String, Int> {

// Используем регулярное выражение для нахождения всех слов

val regex = Regex("[а-яА-Я]+")

val words = regex.findAll(text).map { it.value.lowercase() }.toList()

// Считаем количество повторений каждого слова

val wordCount = mutableMapOf<String, Int>()

for (word in words) {

wordCount[word] = wordCount.getOrDefault(word, 0) + 1

}

return wordCount

}

fun main() {

val text = "Этот текст содержит некоторые слова, такие как мадам и мадам, а также слова, которые не являются палиндромами, например слово и овлос."

val wordOccurrences = countWordOccurrences(text)

if (wordOccurrences.isNotEmpty()) {

println("Количество повторений каждого слова в тексте:")

wordOccurrences.forEach { (word, count) -> println("$word: $count") }

} else {

println("В тексте не найдены слова.")

}

}

Результат работы программы показан на рисунке 4.

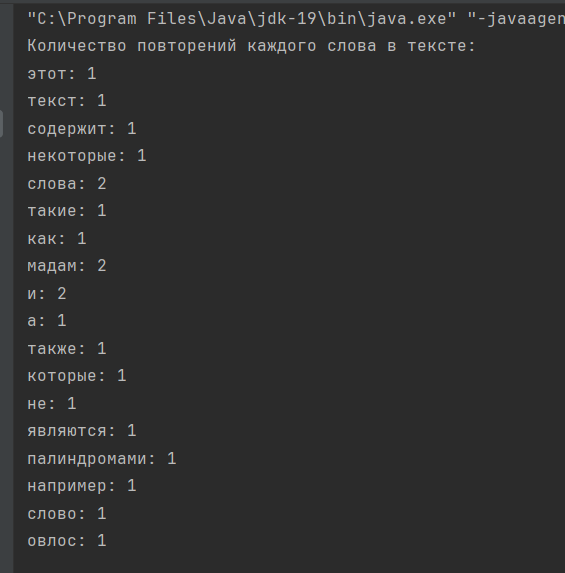


Рисунок 4 – Результат работы программы

**Вывод:** была освоена работа со строками и регулярными выражениями в Java.