|  |  |
| --- | --- |
| 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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ и системы  
 управления**

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

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа,**

**обработки и интерпретации больших данных**

**Отчет**

**по лабораторной работе №7**

**Название:** Строки. Регулярные выражения

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

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

Москва, 2023

**Задания**

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.

package lab7

import java.util.\*

/\*\*\*

\* 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.

\*

\*/

fun main(args: Array<String>) {

val text = "Привет, мир!"

val textLC = text.lowercase(Locale.getDefault())

val numbersBuilder = StringBuilder()

for (element in textLC) {

if (Character.isLetter(element)) {

val num = element.code - 'а'.code + 1

numbersBuilder.append(num).append(" ")

} else {

numbersBuilder.append(" ")

}

}

val numbers = numbersBuilder.toString().trim { it <= ' ' }

println(text.replace("".toRegex(), " ").trim { it <= ' ' })

println(numbers) // print the ordinal numbers below each letter

}

3. В тексте после буквы Р, если она не последняя в слове, ошибочно напечатана буква А вместо О. Внести исправления в текст.

package lab7

/\*\*\*

\* 3. В тексте после буквы Р, если она не последняя в слове,

\* ошибочно напечатана буква А вместо О. Внести исправления в текст.

\*/

fun main(args: Array<String>) {

val text = "Part of students pass exam"

var correctedText = text.replace("pa".toRegex(), "po")

correctedText = correctedText.replace("Pa".toRegex(), "Po")

println("Original text: $text")

println("Corrected text: $correctedText")

}

2. Найти и напечатать, сколько раз повторяется в тексте каждое слово, которое встречается в нем.

package lab7

/\*\*\*

\* 2. Найти и напечатать, сколько раз повторяется в тексте каждое слово, которое встречается в нем.

\*/

fun main(args: Array<String>) {

val text = "This is a sample text with several words. Here are some more words. And here are some more words."

val words = text.split("[\\s.,]+".toRegex()).dropLastWhile { it.isEmpty() }.toTypedArray()

val wordFreq: MutableMap<String, Int> = HashMap()

for (word in words) {

if (wordFreq.containsKey(word)) {

wordFreq[word] = wordFreq[word]!! + 1

} else {

wordFreq[word] = 1

}

}

for ((key, value) in wordFreq) {

println("$key : $value")

}

}

3. В тексте найти и напечатать n символов (и их количество), встречающихся наиболее часто.

package lab7

import java.util.\*

/\*\*\*

\* 3. В тексте найти и напечатать n символов (и их количество), встречающихся наиболее часто.

\*/

fun main(args: Array<String>) {

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

val n = 3

val freqMap: MutableMap<Char, Int> = HashMap()

for (c in text.toCharArray()) {

freqMap[c] = freqMap.getOrDefault(c, 0) + 1

}

val pq = PriorityQueue { (key, value): Map.Entry<Char, Int>,

(key1, value1): Map.Entry<Char, Int> ->

if (value != value1)

value1 - value

else key.code - key1.code

}

pq.addAll(freqMap.entries)

println("Top $n most frequent characters:")

var i = 0

while (i < n && !pq.isEmpty()) {

val (key, value) = pq.poll()

println("$key: $value")

i++

}

}

2. Найти наибольшее количество предложений текста, в которых есть одинаковые слова.

package lab7

import java.util.\*

/\*\*\*

\* 2. Найти наибольшее количество предложений текста, в которых есть одинаковые слова.

\*/

fun main(args: Array<String>) {

val text = ("This is a sample text. It contains multiple sentences. "

+ "Some sentences may contain the same words as others")

val sentences =

text.split("[.?!]".toRegex()).dropLastWhile { it.isEmpty() }.toTypedArray()

val wordCounts: MutableMap<String, Int> = HashMap()

var maxSentenceCount = 0

for (sentence in sentences) {

val words =

sentence.replace("[^a-zA-Z ]".toRegex(), "").lowercase(Locale.getDefault()).split("\\s+".toRegex())

.dropLastWhile { it.isEmpty() }

.toTypedArray()

val uniqueWords: Set<String> =

HashSet(listOf(\*words))

for (word in uniqueWords) {

val count = wordCounts.getOrDefault(word, 0) + 1

wordCounts[word] = count

}

val sentenceCount = Collections.max(wordCounts.values)

if (sentenceCount > maxSentenceCount) {

maxSentenceCount = sentenceCount

}

}

println("The largest number of sentences with identical words is: $maxSentenceCount")

}

3. Найти такое слово в первом предложении, которого нет ни в одном из остальных предложений.

package lab7

import java.util.\*

/\*\*\*

\* 3. Найти такое слово в первом предложении, которого нет ни в одном из остальных предложений.

\*/

fun main(args: Array<String>) {

val text = ("This is the first sentence. "

+ "This is the second sentence. "

+ "The third sentence is different.")

val sentences = text.split("\\.\\s+".toRegex()).dropLastWhile { it.isEmpty() }.toTypedArray()

val words = sentences[0].split("\\s+".toRegex()).dropLastWhile { it.isEmpty() }

.toTypedArray()

val uniqueWords: MutableSet<String> = HashSet(Arrays.asList(\*words))

for (i in 1 until sentences.size) {

val sentenceWords = sentences[i].split("\\s+".toRegex()).dropLastWhile { it.isEmpty() }

.toTypedArray()

for (word in sentenceWords) {

if (uniqueWords.contains(word)) {

uniqueWords.remove(word)

}

}

}

println("The word that is not in any other sentence is: " + uniqueWords.iterator().next())

}

2. Ввести текст и список слов. Для каждого слова из заданного списка найти, сколько раз оно встречается в тексте, и рассортировать слова по убыванию количества вхождений.

package lab7

import java.util.\*

/\*\*\*

\* 2. Ввести текст и список слов. Для каждого слова из заданного списка найти,

\* сколько раз оно встречается в тексте, и рассортировать слова по убыванию количества вхождений.

\*

\* 3. Все слова текста рассортировать в порядке убывания их длин,

\* при этом все слова одинаковой длины рассортировать в порядке возрастания в них количества гласных букв.

\*/

fun main(args: Array<String>) {

val scanner = Scanner(System.`in`)

print("Enter text: ")

val text = scanner.nextLine().lowercase(Locale.getDefault())

print("Enter list of words to count (separated by spaces): ")

val wordsToCount =

scanner.nextLine().lowercase(Locale.getDefault()).split("\\s+".toRegex()).dropLastWhile { it.isEmpty() }

.toTypedArray()

val wordCount: MutableMap<String, Int> = HashMap()

for (word in wordsToCount) {

var count = 0

var index = 0

while (text.indexOf(word, index).also { index = it } != -1) {

count++

index += word.length

}

wordCount[word] = count

}

val sortedWords: List<Map.Entry<String, Int>> = ArrayList<Map.Entry<String, Int>>(wordCount.entries)

sortedWords.sortedWith(Collections.reverseOrder(java.util.Map.Entry.comparingByValue<String, Int>()))

println("Word counts:")

for ((key, value) in sortedWords) {

println("$key: $value")

}

}

3. Все слова текста рассортировать в порядке убывания их длин, при этом все слова одинаковой длины рассортировать в порядке возрастания в них количества гласных букв.

package lab7

import java.util.\*

/\*\*\*

\* 3. Все слова текста рассортировать в порядке убывания их длин,

\* при этом все слова одинаковой длины рассортировать в порядке возрастания в них количества гласных букв.

\*/

fun main() {

val scanner = Scanner(System.`in`)

print("Enter the text: ")

val text = scanner.nextLine()

val words = text.replace("[^a-zA-Z ]".toRegex(), "").lowercase(Locale.getDefault()).split("\\s+".toRegex())

.dropLastWhile { it.isEmpty() }

.toTypedArray()

Arrays.sort(words, Comparator.comparingInt { word: String -> word.length }

.thenComparing(Comparator.comparingInt { obj: String? -> countVowels(obj!!) }))

val sortedWords = listOf(\*words)

Collections.reverse(sortedWords)

println("Sorted words:")

for (word in sortedWords) {

println(word)

}

}

fun countVowels(s: String): Int {

var count = 0

for (c in s.toCharArray()) {

if ("aeiou".indexOf(c) != -1) {

count++

}

}

return count

}

**Вывод:** в ходе выполнения данной лабораторной работы были освоены основы работы со строками в языке программирования Kotlin.