|  |  |
| --- | --- |
|  | **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**

**Отчет**

**по лабораторной работе № 7**

**Название:** Строки. Регулярные выражения

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

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

Москва, 2023

1. ****Введение и задание****

**Целью лабораторной работы является формирование навыков использования регулярных выражений при работе со строками.**

**Номер в списке – 10.**

**Вариант 1.**

**Задание 10.** Удалить из текста его часть, заключенную между двумя символами, которые вводятся (например, между скобками ‘(’ и ‘)’ или между звездочками ‘\*’ и т.п.).

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

**Вариант 2.**

**Задание 10.** В тексте найти первую подстроку максимальной длины, не содержащую букв.

**Задание 1.** В тексте найти все пары слов, из которых одно является обращением другого.

Вариант 3.

Задание 10. Рассортировать слова русского текста по возрастанию доли гласных букв (отношение количества гласных к общему количеству букв в слове).

Задание 1. В тексте нет слов, начинающихся одинаковыми буквами. Напечатать слова текста в таком порядке, чтобы последняя буква каждого слова совпадала с первой буквой последующего слова. Если все слова нельзя напечатать в таком порядке, найти такую цепочку, состоящую из наибольшего количества слов.

Вариант 4.

Задание 10. Исключить из текста подстроку максимальной длины, начинающуюся и заканчивающуюся одним и тем же символом.

Задание 1. Все слова английского текста рассортировать по возрастанию количества заданной буквы в слове. Слова с одинаковым количеством расположить в алфавитном порядке.

1. ****Ход работы****

**Выполнение задания №10 первого варианта лабораторной работы:**

**import java.util.Scanner;**

**public class Main {**

**public static void main(String[] args) {**

**Scanner scanner = new Scanner(System.in);**

**String text = scanner.nextLine();**

**char openChar = scanner.next().charAt(0);**

**char closeChar = scanner.next().charAt(0);**

**System.out.println(text.replaceAll("\\%c.\*\\%c".formatted(openChar, closeChar), ""));**

**}**

**}**

**Выполнение задания №1 первого варианта лабораторной работы:**

**import java.util.Scanner;**

**public class Main {**

**public static void main(String[] args) {**

**Scanner scanner = new Scanner(System.in);**

**String text = scanner.nextLine();**

**int k = scanner.nextInt();**

**scanner.nextLine();**

**char symbol = scanner.nextLine().charAt(0);**

**System.out.println(text.replaceAll("\\b(\\S{" + (k-1) + "})\\S(\\S\*)", "$1\\" + symbol + "$2"));**

**}**

**}**

**Выполнение задания №10 второго варианта лабораторной работы:**

**import java.util.Comparator;**

**import java.util.Scanner;**

**import java.util.regex.MatchResult;**

**import java.util.regex.Matcher;**

**import java.util.regex.Pattern;**

**public class Main {**

**public static void main(String[] args) {**

**Scanner scanner = new Scanner(System.in);**

**String input = scanner.nextLine();**

**Matcher matcher = Pattern.compile("[^A-Za-z]+").matcher(input);**

**String substr = matcher.results().map(MatchResult::group)**

**.max(Comparator.comparing(String::length)).orElse("");**

**System.out.println(substr);**

**}**

**}**

**Выполнение задания №1 второго варианта лабораторной работы:**

**import java.util.Scanner;**

**import java.util.regex.MatchResult;**

**import java.util.regex.Matcher;**

**import java.util.regex.Pattern;**

**import java.util.stream.Collectors;**

**public class Main {**

**public static void main(String[] args) {**

**Scanner scanner = new Scanner(System.in);**

**String input = scanner.nextLine();**

**Pattern pattern = Pattern.compile("\\b(\\S)(\\S)(?:(\\S)(?:(\\S)(?:(\\S)(?:(\\S)(?:(\\S)(?:(\\S)(?:(\\S))?)?)?)?)?)?)?\\b(?=.\*?\\b((?:(?:(?:(?:(?:(?:(?:\\9)?\\8)?\\7)?\\6)?\\5)?\\4)?\\3)?\\2\\1)\\b)");**

**Matcher matcher = pattern.matcher(input);**

**System.out.println(matcher.results().map(MatchResult::group).collect(Collectors.toList()));**

**}**

**}**

**Выполнение задания №10 первого варианта лабораторной работы:**

import java.util.Arrays;

import java.util.Comparator;

public class Main {

public static void main(String[] args) {

String text = "В белом плаще с кровавым подбоем, шаркающей кавалерийской походкой, ранним утром четырнадцатого числа весеннего месяца нисана в крытую колоннаду между двумя крыльями дворца Ирода Великого вышел прокуратор Иудеи Понтий Пилат. Более всего на свете прокуратор ненавидел запах розового масла, и все теперь предвещало нехороший день, так как запах этот начал преследовать прокуратора с рассвета. Прокуратору казалось, что розовый запах источают кипарисы и пальмы в саду, что к запаху кожаного снаряжения и пота от конвоя примешивается проклятая розовая струя. От флигелей в тылу дворца, где расположилась пришедшая с прокуратором в Ершалаим первая когорта Двенадцатого Молниеносного легиона, заносило дымком в колоннаду через верхнюю площадку сада, и к горьковатому дыму, свидетельствовавшему о том, что кашевары в кентуриях начали готовить обед, примешивался все тот же жирный розовый дух.";

String[] words = text.split("[^а-яА-Я]+");

Arrays.sort(words, new Comparator<String>() {

@Override

public int compare(String s1, String s2) {

return Double.compare(getVowelFraction(s1), getVowelFraction(s2));

}

private double getVowelFraction(String word) {

int vowelCount = (int) word.chars().mapToObj(c -> (char) c).filter(c -> Character.isLetter(c) && "аеёиоуыэюяАЕЁИОУЫЭЮЯ".indexOf(c) != -1).count();

int letterCount = (int) word.chars().mapToObj(c -> (char) c).filter(Character::isLetter).count();

return (double) vowelCount / (double) letterCount;

}

});

System.out.println(Arrays.toString(words));

}

}

**Выполнение задания №1 первого варианта лабораторной работы:**

import java.util.\*;

public class Main {

public static void main(String[] args) {

String text = "В белом плаще с кровавым подбоем, шаркающей кавалерийской походкой, ранним утром четырнадцатого числа весеннего месяца нисана в крытую колоннаду между двумя крыльями дворца Ирода Великого вышел прокуратор Иудеи Понтий Пилат. Более всего на свете прокуратор ненавидел запах розового масла, и все теперь предвещало нехороший день, так как запах этот начал преследовать прокуратора с рассвета. Прокуратору казалось, что розовый запах источают кипарисы и пальмы в саду, что к запаху кожаного снаряжения и пота от конвоя примешивается проклятая розовая струя. От флигелей в тылу дворца, где расположилась пришедшая с прокуратором в Ершалаим первая когорта Двенадцатого Молниеносного легиона, заносило дымком в колоннаду через верхнюю площадку сада, и к горьковатому дыму, свидетельствовавшему о том, что кашевары в кентуриях начали готовить обед, примешивался все тот же жирный розовый дух.";

ArrayList<String> words = new ArrayList<>(Arrays.asList(text.split("[^а-яА-Я]+")));

WordChain wordChain = new WordChain(words);

List<String> longestChain = wordChain.getLongestChain();

if (longestChain.isEmpty()) {

System.out.println("No word chain found.");

} else {

System.out.println("Longest word chain:");

for (String word : longestChain) {

System.out.println(word);

}

}

}

}

class WordChain {

private Map<Character, List<String>> wordMap;

public WordChain(List<String> words) {

wordMap = new HashMap<>();

for (String word : words) {

char firstChar = word.charAt(0);

if (!wordMap.containsKey(firstChar)) {

wordMap.put(firstChar, new ArrayList<>());

}

wordMap.get(firstChar).add(word);

}

}

public List<String> getLongestChain() {

List<String> longestChain = new ArrayList<>();

for (char c : wordMap.keySet()) {

List<String> chain = dfs(c, new HashSet<>());

if (chain.size() > longestChain.size()) {

longestChain = chain;

}

}

return longestChain;

}

private List<String> dfs(char c, Set<String> visited) {

List<String> longestChain = new ArrayList<>();

for (String word : wordMap.getOrDefault(c, Collections.emptyList())) {

if (!visited.contains(word)) {

visited.add(word);

List<String> chain = dfs(word.charAt(word.length() - 1), visited);

chain.add(0, word);

if (chain.size() > longestChain.size()) {

longestChain = chain;

}

visited.remove(word);

}

}

return longestChain;

}

}

**Выполнение задания №10 второго варианта лабораторной работы:**

**import java.util.regex.Matcher;**

**import java.util.regex.Pattern;**

**public class Main {**

**public static void main(String[] args) {**

**String input = "lorem ipsum dolor sit amet";**

**String longestSubstring = "";**

**for (char c : input.toCharArray()) {**

**Pattern pattern = Pattern.compile("%c.\*%c".formatted(c, c));**

**Matcher matcher = pattern.matcher(input);**

**if (matcher.find() && longestSubstring.length() < matcher.group().length()) {**

**longestSubstring = matcher.group();**

**}**

**}**

**System.out.println(input.replaceFirst(longestSubstring, ""));**

**}**

**}**

**Выполнение задания №1 второго варианта лабораторной работы:**

import java.util.Arrays;

import java.util.Comparator;

public class Main {

public static final String REFERENCE\_CHARS = "u";

public static void main(String[] args) {

String text = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut sed ex vitae justo rhoncus feugiat sit amet et nulla. Sed finibus justo vitae massa accumsan dapibus. In vel sem vel risus volutpat scelerisque non quis nunc. Praesent id turpis vitae lacus rutrum porta. Cras ut imperdiet arcu, a posuere est.";

String[] words = text.split("[^a-zA-Z]+");

Arrays.sort(words, new Comparator<String>() {

@Override

public int compare(String s1, String s2) {

int comp1 = getReferenceCharAmount(s1);

int comp2 = getReferenceCharAmount(s2);

return comp1 != comp2 ? Integer.compare(comp1, comp2) : s1.toLowerCase().compareTo(s2.toLowerCase());

}

private int getReferenceCharAmount(String word) {

return (int) word.toLowerCase().chars().mapToObj(c -> (char) c).filter(c -> Character.isLetter(c) && REFERENCE\_CHARS.indexOf(c) != -1).count();

}

});

System.out.println(Arrays.toString(words)); }

}

1. ****Заключение****

**В процессе выполнения лабораторной работы были сформированы навыки использования регулярных выражений при работе со строками в языке программирования java, были изучены соответствующие синтаксические конструкции и особенности работы.**