

|  |  |
| --- | --- |
| **Министерство образования и науки**  **Российской Федерации**  **Государственное образовательное учреждение высшего образования «МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ**  **ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  **(МАДИ)»** |  |

**Кафедра «Высшая математика»**

**Отчет по дисциплине «Структуры и алгоритмы обработки данных»**

**Лабораторная работа №5**

**«Методы поиска подстроки в строке и обработка текстовых строк»**

**Выполнил:**

Учебная группа 1бПМ\_\_\_\_\_\_\_\_\_  
ФИО Кузнецов А-С.О.\_\_\_\_\_

**Принял:**

Должность \_\_\_\_\_

Звание \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ФИО \_ Кутейников И.А.\_\_\_\_\_\_

«\_\_\_ »\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023г.

Москва 2023

Цель:

Программа должна содержать следующие функции:

1 Инициализация строки с клавиатуры;

2 Инициализация строки случайными символами;

3 Вывод строки на экран;

4 Поиск подстроки;

4.1 Поиск подстроки методом последовательного доступа;

4.2 Поиск подстроки методом Кнута-Морриса-Пратта;

4.3 Поиск подстроки упрощенным методом Бойера-Мура;

4.4 Поиск подстроки встроенной функцией;

4.5 Сравнение времени работы методов;

5 Решение задачи уровня Б;

6 Решение задачи уровня С (опционально);

7 Выход из программы.

Требуется выделить в строке-предложении s все слова, разделенные

символами-разделителями «\_.,;:\n\t!?», и обработать выделенные слова в

соответствии с вариантом задания.

Регулярное слово – слово, состоящее только из больших латинских

букв.

Палиндром – это слово, которое одинаково читается слева направо и

справа налево.

Алфавитный порядок задается таблицей ASCII.

Уровень B

Выделить в строке-предложении s все слова, разделенные символами-

разделителями «\_.,;:\n\t!?». Обработать выделенные слова в соответствии с

вариантом задания.

B5. Напечатать все регулярные слова. Напечатать в перевернутом виде самое

длинное слово, состоящее только из цифр и латинских букв.

Код:

static char[] fillByHand (){  
 Scanner in=new Scanner(System.*in*);  
 String word= in.nextLine();  
 char [] Word=new char[word.length()];  
 for (int i =0; i<word.length();i++){  
 Word[i]=word.charAt(i);  
 }  
 return Word;  
}  
static char[] fillByRandom(char [] array){  
 Scanner in = new Scanner(System.*in*);  
 Random rand= new Random();  
 System.*out*.println("Enter kol-vo el: ");  
 int num=in.nextInt();  
 char [] Word=new char[num];  
 for (int i=0; i< num;i++ ){  
 int num1=rand.nextInt(array.length-1);  
 Word[i]=array[num1];  
 }  
 return Word;  
}  
public static int[] computeLPSArray(String pattern) {  
 int patternLength = pattern.length();  
 int[] lps = new int[patternLength];  
 int len = 0;  
 int i = 1;  
 lps[0] = 0;  
  
 while (i < patternLength) {  
 if (pattern.charAt(i) == pattern.charAt(len)) {  
 len++;  
 lps[i] = len;  
 i++;  
 } else {  
 if (len != 0) {  
 len = lps[len - 1];  
 } else {  
 lps[i] = len;  
 i++;  
 }  
 }  
 }  
 return lps;  
}  
  
public static void KMPSearch(String text, String pattern) {  
 int patternLength = pattern.length();  
 int textLength = text.length();  
  
 int[] lps = *computeLPSArray*(pattern);  
 int i = 0; // Индекс для текста  
 int j = 0; // Индекс для шаблона  
  
 while (i < textLength) {  
 if (pattern.charAt(j) == text.charAt(i)) {  
 i++;  
 j++;  
 }  
 if (j == patternLength) {  
 System.*out*.println("Подстрока найдена на позиции " + (i - j));  
 j = lps[j - 1];  
 } else if (i < textLength && pattern.charAt(j) != text.charAt(i)) {  
 if (j != 0) {  
 j = lps[j - 1];  
 } else {  
 i = i + 1;  
 }  
 }  
 }  
}  
public static List<Integer> boyerMooreSearch(String text, String pattern) {  
 List<Integer> occurrences = new ArrayList<>();  
 int m = pattern.length();  
 int n = text.length();  
 int[] right = new int[256]; // Массив для символов ASCII  
 for (int c = 0; c < 256; c++) {  
 right[c] = -1; // Инициализация массива правых позиций символов  
 }  
 for (int j = 0; j < m; j++) {  
 right[pattern.charAt(j)] = j; // Заполнение массива правых позиций  
 }  
 int skip;  
 for (int i = 0; i <= n - m; i += skip) {  
 skip = 0;  
 for (int j = m - 1; j >= 0; j--) {  
 if (pattern.charAt(j) != text.charAt(i + j)) {  
 skip = Math.*max*(1, j - right[text.charAt(i + j)]); // Вычисление шага сдвига  
 break;  
 }  
 }  
 if (skip == 0) {  
 occurrences.add(i); // Если совпадение найдено, добавляем позицию в список  
 skip = 1;  
 }  
 }  
 return occurrences;  
}  
public static String reverse(String str) {  
 return new StringBuilder(str).reverse().toString();  
}  
public static void main(String[] args) {  
 char[] Elements = {'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z',  
 'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',  
 '0','1','2','3','4','5','6','7','8','9','+','-','/','\*','^','\_','=','(',')','?','&',':',';',',','.','%',  
 '<','>','!','@','"','$','#','№', ' '};  
 char [] elForbword={'\_',',',';','.',':','!','?',' '};  
 char [] elForbwordcase5={'+','-','/','\*','^','\_','=','(',')','?','&',':',';',',','.','%','<','>','!','@','"','$','#','№',' '};  
 char [] elForbwordcase51={'+','-','/','\*','^','\_','=','(',')','?','&',':',';',',','.','%','<','>','!','@','"','$','#','№',' ','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','0','1','2','3','4','5','6','7','8','9'};

System.*out*.println("Все слова:");  
String text=new String(str);  
String[] words = text.split(Arrays.*toString*(elForbword));//words  
int kolvodlov=0;  
for(String word : words){  
 System.*out*.print(word+" ");  
 kolvodlov++;  
}  
String[] words1=words.clone();  
for(int el4=0;el4< words1.length;){  
 for(int el5=-1;el5<elForbwordcase51.length;) {  
 boolean b10=true;  
 int elel=0;  
 do{  
 char[] terarist=words1[el4].toCharArray();  
 el5++;  
 if(terarist[elel]==elForbwordcase51[el5]) {  
 words1[el4] = "b";  
 el5 = -1;  
 el4++;  
 if (el4 == words1.length)  
 b10 = false;  
 elel = 0;  
 if (el4 < words1.length && words1[el4] == "") {  
 el4++;  
 }  
 }  
 if(el5==elForbwordcase51.length-1&&elel!= terarist.length-1){  
 elel++;  
 el5=0;  
 }  
 if (el5==elForbwordcase51.length-1&&elel== terarist.length-1){  
 el4++;  
 if (el4 < words.length && words[el4] == "") {  
 el4++;  
 }  
 el5=-1;  
 elel=0;  
 if(el4== words.length)  
 b10=false;  
 }  
 }while(b10);  
 if (el4== words.length)  
 break;  
 }  
 if (el4== words.length)  
 break;  
}  
System.*out*.println("Регулярные слова:");  
for (int io=0;io< words1.length;io++){  
 if (words1[io].equals("b")==false)  
 System.*out*.println(words1[io]);  
}  
for(int el4=0;el4< words.length;){  
 for(int el5=-1;el5<elForbwordcase5.length;) {  
 boolean b10=true;  
 int elel=0;  
 do{  
 char[] terarist=words[el4].toCharArray();  
 el5++;  
 if(terarist[elel]==elForbwordcase5[el5]) {  
 words[el4] = "b";  
 el5 = -1;  
 el4++;  
 if (el4 == words.length)  
 b10 = false;  
 elel = 0;  
 if (el4 < words.length && words[el4] == "") {  
 el4++;  
 }  
 }  
 if(el5==elForbwordcase5.length-1&&elel!= terarist.length-1){  
 elel++;  
 el5=0;  
 }  
 if (el5==elForbwordcase5.length-1&&elel== terarist.length-1){  
 el4++;  
 if (el4 < words.length && words[el4] == "") {  
 el4++;  
 }  
 el5=-1;  
 elel=0;  
 if(el4== words.length)  
 b10=false;  
 }  
 }while(b10);  
 if (el4== words.length)  
 break;  
 }  
 if (el4== words.length)  
 break;  
}  
String text1=new String();  
for(int i3=0; i3< words.length;i3++){  
 text1+=" "+words[i3];  
}  
text1=text1.trim();  
String longest = Arrays.*stream*(text1.split(Arrays.*toString*(elForbword)))  
 .max(Comparator.*comparingInt*(String::length))  
 .orElse(null);  
longest=*reverse*(longest);  
System.*out*.println("a biggest word:"+longest);

Результат: Menu

1.fill by hand

2.fill by rand

3.print in cons str.

4.find undstr

5.do num b

6.do num c

7.exit

take num menu: 1

Enter the word:

LEGFUI ;JKWHF;lk;hwef

take num menu: 3

LEGFUI ;JKWHF;lk;hwef

take num menu: 4

MenuForP4

1 Поиск подстроки методом последовательного доступа;

2 Поиск подстроки методом Кнута-Морриса-Пратта;

3 Поиск подстроки упрощенным методом Бойера-Мура;

4 Поиск подстроки встроенной функцией;

5 Сравнение времени работы методов;

PunktiMenu= 1

Enter the sub str:

WHF

Подстрока с:10

Время работы послед дост:6481

PunktiMenu= 2

Enter the sub str:

WHF

Подстрока найдена на позиции 10

Время работы поиска методом Кнута-Морриса-Пратта:1221

PunktiMenu= 3

Enter the sub str:

WHF

Occurrences found at positions: [10]

Поиск подстроки упрощенным методом Боера-Мура:1118

PunktiMenu= 4

Enter the sub str:

WHF

Подстрока найдена в позиции 10

Подстрока не найдена

Время работы поиска встроенной функцией:1299

take num menu: 5

Все слова:

LEGFUI JKWHF lk hwef

Регулярные слова:

LEGFUI

JKWHF

a biggest word:IUFGEL

Вывод:

В ходе лабораторной работы было получено понимание того, как работать со строками, а именно:

1. Как их создавать
2. Как работать с каждым элементом
3. Как выделять слова
4. Как находить необходимую подстроку

Также было получено понимание того, как работать со словарем ASCII.