Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Отчет задачам №3

на тему: «Методы поиска в подстроке»

Выполнил: студент группы БВТ1903

Белов Сергей Павлович

Проверил:

Павликов Артём Евгеньевич

# **Строка победитель**

Задача:

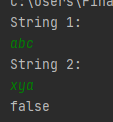
Даны две строки: s1 и s2 с одинаковым размером, проверьте, может ли некоторая перестановка строки s1 “победить” некоторую перестановку строки s2 или наоборот. Строка x может “победить” строку y (обе имеют размер n), если x [i] >= y[i] (в алфавитном порядке) для всех i от 0 до n-1.

Алгоритм: каждая буква имеет свою «силу» в алфавитном порядке. Подсчитывается «сила» всех букв строки и сравнивается с «силой» другой. Если первая строка побеждает, то вывод «true», иначе «False».

Листинг:

import java.util.Scanner;  
  
public class Ex\_1 {  
 public static void main(String[] arg) {  
 //запрашиваем входные данные  
 Scanner s = new Scanner(System.*in*);  
 System.*out*.println("String 1:");  
 String string1 = s.nextLine();//принимаем первую строку  
 System.*out*.println("String 2:");  
 String string2 = s.nextLine();//принимаем вторую строку  
 if (string1.length() != string2.length())  
 System.*out*.println("Different strings");//строки разной длины  
 else {  
 int count1 = 0;//счестчики цены букв  
 int count2 = 0;  
 for (int i = 0; i < string1.length(); i++) {//бежим по строкам  
 count1 += *Method* (string1.charAt(i));//суммируем ценность букв  
 count2 += *Method* (string2.charAt(i));  
 }  
 System.*out*.println(count2<count1);//возвращаем ответ  
 }  
 }  
 public static int Method (char a){//метод сопоставляющий букву из слова и ее ценность  
 char[] arr = {'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'};  
 for (int i=0; i < 28;i++){  
 if (a == arr[i]){  
 return i;  
 }  
 }  
 return 0;  
 }  
}

Результаты работы программы:



# **Самый длинный палиндром**

Задача:

Дана строка s, вернуть самую длинную палиндрому подстроку в s.

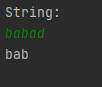
Алгоритм:

Начиная с первой буквы строки идет проверка на палиндром и проверка на то, что является ли самым длинным палиндромом.

Листинг:

import java.util.Scanner;  
  
public class Ex\_2 {  
 public static void main(String[] arg) {  
 //запрашиваем входные данные  
 Scanner s = new Scanner(System.*in*);  
 System.*out*.println("String:");  
 String string1 = s.nextLine();  
 String sub\_max = "";//самая длинная подстрока-полиндром  
 for (int k=0;k<string1.length();k++) {//определяем с какой позиции начинать  
 String sub = "";//текущая подстрока  
 for (int i = k; i < string1.length(); i++) {//добавляем следующие буквы в наше слово  
 sub += string1.charAt(i);  
 if ((sub.equals(*Palindrom*(sub))) == true) {//если слово палиндром, то запоминаемего  
 if (sub.length() > sub\_max.length())//если слово длиннее текущего палиндрома  
 sub\_max = sub;  
 }  
 }  
 }  
 System.*out*.println(sub\_max);  
  
 //основная работа  
  
 }  
 public static String Palindrom (String s){//проверка на палиндром  
 String sub = "";  
 for(int i=(s.length()-1);i>=0;i--){  
 sub+=s.charAt(i);  
 }  
 return sub;  
 }  
  
}

Результаты работы программы:



# **Конкатенация строки**

Задача

Вернуть количество отдельных непустых подстрок текста, которые могут быть записаны как конкатенация некоторой строки с самой собой (т.е. она может быть записана, как a + a, где a – некоторая строка).

Листинг:

import java.util.Scanner;  
  
public class Ex\_3 {  
 public static void main(String[] arg){  
 Scanner s = new Scanner(System.*in*);  
 System.*out*.println("String:");  
 String string1 = s.nextLine();//получаем строку  
 int count = 0;  
 for (int i=0; i<string1.length();i++){//определяем начало поиска  
 String sub = "";  
 for(int j=i; j<string1.length();j++){//добавляем  
 sub+=string1.charAt(j);  
 if (string1.indexOf(sub,j)==i+sub.length()){//если ближайшее вхождение текущего слова находится  
 count++;//сразу полсе его конца, то увеличиваем счетчик искомых слов  
 System.*out*.println(sub+" индекс его ближайшего вхождения: "+string1.indexOf(sub,j));  
 }  
 }  
 }  
 System.*out*.println(count);  
 }  
}

Результаты работы программы:

