МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное

образовательное учреждение

высшего образования

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

**по дисциплине**

**“ Алгоритмы построение и анализ”**

Выполнил: студент гр. ФИб-3302-51-00 ё. \_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ПМиИ Разова Е. В. \_\_\_\_\_\_\_\_\_\_\_

Киров 2020

***Задача 1. Циклическая строка***

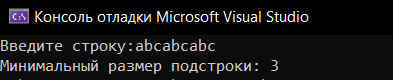
Строка *S* была записана много раз подряд, после чего из получившейся строки взяли подстроку и дали вам. Определить минимально возможную длину исходной строки *S*.

**Входные данные**

На вход программы поступает строка, которая содержит только латинские буквы, длина строки не превышает 50000 символов.

**Выходные данные**

Требуется вывести одно число – ответ на вопрос задачи.



#include <iostream>

#include <vector>

#include <string>

#include <clocale>

using namespace std;

vector<int> findsubstring(string s)

{

int n = (int)s.size();

vector<int> pi(n);

for (int i = 1; i < n; i++)

{

int j = pi[i - 1];

while (j > 0 && s[i] != s[j])

j = pi[j - 1];

if (s[i] == s[j])

j++;

pi[i] = j;

}

return pi;

}

int main()

{

setlocale(0, "");

cout << "Введите строку:";

string s;

cin >> s;

vector <int> p = findsubstring(s);

cout << "Минимальный размер подстроки: ";

cout << s.size() - p.back();

return 0;

}

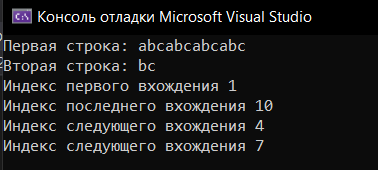
***Задача 2. Поиск подстроки***

Используя наивный алгоритм, найти

А) первое вхождение подстроки *Т* в строку *S*;

В) последнее вхождение подстроки *Т* в строку *S*;

С) все вхождения подстроки *T* в строку *S* (номера символов, начиная с которых подстрока *T* входит в строку *S*, в порядке возрастания).



#include <iostream>

#include <string>

#include <iostream>

#include <algorithm>

#include <clocale>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

int counter(0);

string s = "abcabcabcabc";

string s1 = "bc";

cout << "Первая строка: " << s << endl;

cout << "Вторая строка: " << s1 << endl;

bool flag = false;

int tmpi = 0;

int start = 0, finish = 0;

string tmpstring = "";

for (int i = 0; i < s.length(); i++) {

tmpi = i;

for (int j = 0; j < s1.length(); j++) {

if (s[tmpi] == s1[j]) {

tmpi++;

flag = true;

}

else {

flag = false;

break;

}

}

if (flag == true) {

start = i;

cout << "Индекс первого вхождения " << i << endl;

break;

}

}

string srev = s;

string s1rev = s1;

reverse(srev.begin(), srev.end());

reverse(s1rev.begin(), s1rev.end());

flag = false;

for (int i = 0; i < srev.length(); i++) {

tmpi = i;

for (int j = 0; j < s1rev.length(); j++) {

if (srev[tmpi] == s1rev[j]) {

tmpi++;

flag = true;

}

else {

flag = false;

break;

}

}

if (flag == true) {

finish = s.length() - s1.length() - i;

cout << "Индекс последнего вхождения " << finish << endl;

break;

}

}

for (int i = start + 1; i < finish; i++) {

tmpi = i;

for (int j = 0; j < s1.length(); j++) {

if (s[tmpi] == s1[j]) {

tmpi++;

flag = true;

}

else {

flag = false;

break;

}

}

if (flag == true) {

start = i;

cout << "Индекс следующего вхождения " << i << endl;

}

}

return 0;

}

***Задача 3. Строчки***

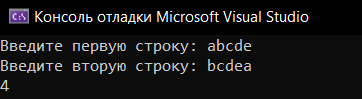
На листе бумаги написана строка, состоящая из больших и маленьких латинских букв. Под данной строкой написана еще одна строчка такой же длины. Утверждается, что вторая строка получена циклическим сдвигом первой строки на несколько шагов вправо (циклический сдвиг строки *abcde* на 2 позиции вправо даст строку *deabc*). По данным строкам выведите минимальный возможный размер сдвига или -1, если вторая строка не может быть получена из первой операцией сдвига.

**Входные данные**

Первые две строки входных данных содержат строки. Длины строк одинаковы, не превышают 10000 и не равны 0.

**Выходные данные**

Единственное число – ответ на вопрос задачи.



#include <iostream>

#include <string>

#include <clocale>

using namespace std;

int pi[10000];

int main()

{

setlocale(LC\_ALL, "Russian");

string s, t;

cout << "Введите первую строку: ";

cin >> s;

cout << "Введите вторую строку: ";

cin >> t;

string q = t + '#' + s;

int n = q.size();

for (int i = 1; i < n; i++)

{

int j = pi[i - 1];

while (j > 0 && q[i] != q[j])

j = pi[j - 1];

if (q[i] == q[j])

++j;

pi[i] = j;

}

if (s == t)

cout << 0;

else

{

if (!pi[n - 1])

cout << -1;

else

cout << pi[n - 1];

}

return 0;

}

***Задача 4. Префикс-функция***

Дана непустая строка *S*, длина которой *N* не превышает 106. Будем считать, что элементы строки нумеруются от 1 до *N*.

Для каждой позиции *i* символа в строке нас будет интересовать подстрока, заканчивающаяся в этой позиции, и совпадающая с некоторым началом всей строки. Вообще говоря, таких подстрок будет несколько, не меньше двух. Самая длинная из них имеет длину *i*, она нас интересовать не будет. А будет нас интересовать самая длинная из остальных таких подстрок (заметим, что такая подстрока всегда существует – в крайнем случае, если ничего больше не найдется, сгодится пустая подстрока).

Значением префикс-функции *π*[*i*] будем считать длину этой подстроки.

Требуется для всех *i* от 1 до *N* вычислить *π*[*i*].

**Входные данные**

Одна строка длины *N*, 0 < *N* ≤ 106, состоящая из маленьких латинских букв.

**Выходные данные**

Выведите *N* чисел – значения префикс-функции для каждой позиции, разделенные пробелом.

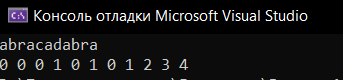
**Пример:**

*входные данные:*

abracadabra

*выходные данные:*

0 0 0 1 0 1 0 1 2 3 4



#include <iostream>

#include <string>

#include <regex>

using namespace std;

vector<int> prefix\_function(const string& s)

{

int len = s.length();

vector<int> p(len);

int k = 0; // счетчик

for (int i = 1; i < len; i++)

{

if (s[k] != s[i])

{

// повторная проверка при k = 0

if (s[0] != s[i]) k = 0;

else k = 1;

}

else

{

k++; // если символы совпадают -> увеличиваем значение счетчика

}

p[i] = k; // значение счетчик в вектор

}

for (int i = 0; i < len; i++) {

cout << p[i] << " ";

}

return p;

}

int main()

{

//string s = "ababcababcabab";

string s = "abracadabra";

cout << s << endl;

prefix\_function(s);

}