

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №4**  
**по дисциплине «Построение и анализ алгоритмов»**  
**Тема: Алгоритм Кнута-Морриса-Пратта**

Студент гр. 8383

\_\_\_\_\_

Колмыков В.Д.

Преподаватель

\_\_\_\_\_

Фирсов М.А.

Санкт-Петербург

2020

### **Цель работы.**

Реализовать алгоритм Кнута-Морриса-Пратта, найти индексы вхождения подстроки в строку, а также разработать алгоритм проверки двух строк на циклический сдвиг.

### **Вариант 2.**

Оптимизация по памяти: программа должна требовать  $O(m)$  памяти, где  $m$  - длина образца. Это возможно, если не учитывать память, в которой хранится строка поиска.

### **Задание.**

Реализуйте алгоритм КМП и с его помощью для заданных шаблона  $P$  ( $|P| \leq 15000$ ) и текста  $T$  ( $|T| \leq 5000000$ ) найдите все вхождения  $P$  в  $T$ .

Вход:

Первая строка -  $P$

Вторая строка -  $T$

Выход:

индексы начал вхождений  $P$  в  $T$ , разделенных запятой, если  $P$  не входит в  $T$ , то вывести -1

Пример ввода:

ab

abab

Пример вывода:

0, 2

Заданы две строки  $A$  ( $|A| \leq 5000000$ ) и  $B$  ( $|B| \leq 5000000$ ).

Определить, является ли  $A$  циклическим сдвигом  $B$  (это значит, что  $A$  и  $B$  имеют одинаковую длину и  $A$  состоит из суффикса  $B$ , склеенного с префиксом  $B$ ). Например, `defabc` является циклическим сдвигом `abcdef`.

Вход:

Первая строка -  $A$

Вторая строка -  $B$

Выход:

Если  $A$  является циклическим сдвигом  $B$ , индекс начала строки  $B$  в  $A$ , иначе вывести  $-1$ . Если возможно несколько сдвигов вывести первый индекс.

Пример ввода:

`defabc`

`abcdef`

Пример вывода:

3

### **Описание алгоритма КМП.**

На вход алгоритма передается строка-образец, вхождения которой нужно найти, и строка-текст, в которой нужно найти вхождения.

Оптимизация — строка-текст считывается посимвольно, в памяти хранится только текущий символ.

Алгоритм сначала вычисляет префикс-функцию (длина наибольшего собственного префикса подстроки, который совпадает с суффиксом этой подстроки) строки-образца.

Далее посимвольно считывается строка-текст. Переменная-счетчик изначально  $k = 0$ . Переменная-позиция в образце изначально  $i = 0$ ; При каждом совпадении  $k$ -го символа образца и  $i$ -го символа текста переменный увеличиваются на 1. Если  $k = \text{размер образца}$ , значит вхождение найдено. Если очередной символ текста не совпал с  $k$ -ым символом образца, то сдвигаем образец, причем точно знаем, что первые  $k$  символов образца совпали с символами строки и надо сравнить  $k+1$ -й символ образца (его индекс  $k$ ) с

символом строки, который будет равен значению префикс-функции  $i-1$  – ого символа в образце.

Сложность алгоритма по операциям:  $O(m + n)$ ,  $m$  – длина образца,  $n$  – длина текста (линейное число раз алгоритм пробегается по строке и массиву префикс функции).

Сложность алгоритма по памяти:  $O(m)$ ,  $m$  – длина образца(в памяти находится только образец и текущий символ текста).

### **Описание алгоритма проверки циклического сдвига.**

Для того, чтобы вычислить, является ли одна строка циклическим сдвигом другой, можно воспользоваться префикс функцией.

Сначала алгоритм сравнивает размеры строк, если они не совпадают – строки не могут являть циклическим сдвигом.

Затем алгоритм сравнивает строки, если они равны, то они циклический сдвиг друг друга со смещением 0.

Если алгоритм не завершил работу, складывается первая строка с двумя вторыми (лексикографически), далее вычисляется префикс-функция от строки результата. Строка результат имеет вид  $AB_1B_2$  ( $A$  – первая строка,  $B$  – вторая строка). Если в  $B_2$  у какого-нибудь символа префикс-функция равна длине образа, то строки являются циклическим сдвигом.

Сложность алгоритма по операциям:  $O(n)$ ,  $n$  – длина строки (линейное число раз (2) алгоритм пробегается по массиву префикс функции)

Сложность алгоритма по памяти:  $O(n)$ ,  $n$  – длина строки (содержится 5 строк размером с введенные).

### **Описание функций и структур данных.**

**void algorithmKMP(std::string& templ)**

Функция реализация алгоритма КМП.

templ – строка, содержащая в себе образец для поиска.

**void makeArr(int\* arr, int len, std::string& templ)**

Функция для вычисления значения префикс функции.

arr – указатель на массив, который необходимо заполнить значениями функции.

len – длина массива.

templ – строка, содержащая в себе образец, для которого и считается функция.

**void checkCyclicShift(std::string& firstStr, std::string& secondStr)**

Функция для вычисления размера циклического сдвига первой строки относительно второй.

firstStr – первая строка.

secondStr – вторая строка.

**int\* makePrefix(char\* templ, int size)**

Функция для вычисления значения префикс функции для char\*.

templ – образец для вычисления.

size – размер образца.

Возвращает массив со значением префикс функции для символов образца.

**Тестирование.**

**Алгоритм КМП:**

Ввод	Вывод
ab abab	0, 2
vlad qwevladqwervlaqwevladqwf	3, 17
vlad vlasfvlalefvlas	-1

### Проверка циклического сдвига:

Ввод	Вывод
defabc abcdef	3
abcd qwer	-1
vladislav vladislav	0

Пример вывода промежуточной информации представлен на рис. 1 и 2.

```

vlad
Поиск вхождения:
Вычисление префикс функции:
Значение префикс функции для символа v равно 0
Символы с индексами 1 и 0 не совпадают (l и v). Первый индекс увеличивается
Значение префикс функции для символа l равно 0
Символы с индексами 2 и 0 не совпадают (a и v). Первый индекс увеличивается
Значение префикс функции для символа a равно 0
Символы с индексами 3 и 0 не совпадают (d и v). Первый индекс увеличивается
Значение префикс функции для символа d равно 0
Значение префикс функции:
v - 0
l - 0
a - 0
d - 0
qwevladqwer
Символы номер 0 в строке и 0 в образе не совпадают (q и v)
Номер символа в тексте увеличивается на 1 (1)
Символы номер 1 в строке и 0 в образе не совпадают (w и v)
Номер символа в тексте увеличивается на 1 (2)
Символы номер 2 в строке и 0 в образе не совпадают (e и v)
Номер символа в тексте увеличивается на 1 (3)
Символы номер 3 в строке и 0 в образе совпадают (v)
Номер символа в образе увеличивается на 1(1)
Номер символа в тексте увеличивается на 1 (4)
Символы номер 4 в строке и 1 в образе совпадают (l)
Номер символа в образе увеличивается на 1(2)
Номер символа в тексте увеличивается на 1 (5)
Символы номер 5 в строке и 2 в образе совпадают (a)
Номер символа в образе увеличивается на 1(3)
Номер символа в тексте увеличивается на 1 (6)
Символы номер 6 в строке и 3 в образе совпадают (d)
Номер символа в образе увеличивается на 1(4)
Номер символа в образе равен длине образа -> вхождение найдено: 3
Номер символа в образе равен 0
Номер символа в тексте увеличивается на 1 (7)
Символы номер 7 в строке и 4 в образе не совпадают (q и )
Номер символа в образе становится равным значению префикс функции символа с номером на один меньше (0)
Символы номер 7 в строке и 0 в образе не совпадают (q и v)
Номер символа в тексте увеличивается на 1 (8)
Символы номер 8 в строке и 0 в образе не совпадают (w и v)
Номер символа в тексте увеличивается на 1 (9)
Символы номер 9 в строке и 0 в образе не совпадают (e и v)
Номер символа в тексте увеличивается на 1 (10)
Символы номер 10 в строке и 0 в образе не совпадают (r и v)
Номер символа в тексте увеличивается на 1 (11)
^Z
Строка закончилась
Результат:
Для продолжения нажмите любую клавишу . . .

```

Рисунок 1 – вывод отладочной информации для алгоритма КМП

```

vlad
advl
Создана строка, состоящая из второй строки и двух первых:
advlvladvld
Вычисление префикс функции:
Префикс функция для первого символа равна 0, первый индекс равен 1, второй индекс равен 0
Символы с индексами 1 и 0 не совпадают (d и a). Первый индекс увеличивается
Значение префикс функции для символа d равно 0
Символы с индексами 2 и 0 не совпадают (v и a). Первый индекс увеличивается
Значение префикс функции для символа v равно 0
Символы с индексами 3 и 0 не совпадают (l и a). Первый индекс увеличивается
Значение префикс функции для символа l равно 0
Символы с индексами 4 и 0 не совпадают (v и a). Первый индекс увеличивается
Значение префикс функции для символа v равно 0
Символы с индексами 5 и 0 не совпадают (l и a). Первый индекс увеличивается
Значение префикс функции для символа l равно 0
Символы с индексами 6 и 0 совпадают (a). Оба индекса увеличиваются.
Значение префикс функции для символа a равно 1
Символы с индексами 7 и 1 совпадают (d). Оба индекса увеличиваются.
Значение префикс функции для символа d равно 2
Символы с индексами 8 и 2 совпадают (v). Оба индекса увеличиваются.
Значение префикс функции для символа v равно 3
Символы с индексами 9 и 3 совпадают (l). Оба индекса увеличиваются.
Значение префикс функции для символа l равно 4
Символы с индексами 10 и 0 не совпадают (a и a). Второй индекс становится равным значению префикс функции предыдущего символа (0)
Символы с индексами 10 и 0 совпадают (a). Оба индекса увеличиваются.
Значение префикс функции для символа a равно 1
Символы с индексами 11 и 1 совпадают (d). Оба индекса увеличиваются.
Значение префикс функции для символа d равно 2
Значение префикс функции:
0      a      0
1      d      0
2      v      0
3      l      0
4      v      0
5      l      0
6      a      1
7      d      2
8      v      3
9      l      4
10     a      1
11     d      2
У символа в строке под номером 9 (l) префикс функция равна длине изначальной строки 4
Результат:
2
Для продолжения нажмите любую клавишу . . .

```

Рисунок 2 — вывод отладочной информации для алгоритма поиска цикла

## Выводы.

В ходе выполнения лабораторной работы был реализован алгоритм КМП и алгоритм проверки двух строк на циклический сдвиг, а также функция вычисления префикса строки.

## ПРИЛОЖЕНИЕ А

### КОД ПРОГРАММ

#### Реализация алгоритма КМП:

```
#include <iostream>
#include <string>
#include <vector>
#include <Windows.h>

int algorithmKMP(std::string& templ); //Функция реализации алгоритма
void makeArr(int* arr, int len, std::string& templ); //Функция вычисления
префикс функции

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    std::string templ;
    std::cin >> templ;
    algorithmKMP(templ);
    system("pause");
}

int algorithmKMP(std::string& templ) {
    std::vector<int> res;
    std::cout << "Поиск вхождения:\n";
    bool isFirst = true;
    int* arr = new int[templ.length()];
    makeArr(arr, templ.length(), templ); //Рассчитываем значение префикс
функции для образца
    int k = 0; //Инициализируем индексы на начало строк
    int l = 0;
    char curr;
    std::cin >> curr;
    while (!std::cin.eof()) { //Пока текст не считан полностью
        if (curr == templ[l]) { //Если совпадение с образом
            std::cout << "Символы номер " << k << " в строке и " << l <<
" в образе совпадают (" << templ[l] << ")\n";
            l++;
            std::cout << "Номер символа в образе увеличивается на 1 (" <<
l << ")\n";
            if (l == templ.length()) { //Если вхождение найдено
                std::cout << "Номер символа в образе равен длине
образа -> вхождение найдено: " << k - templ.length() + 1 << std::endl;
                res.push_back(k - templ.length() + 1);
                isFirst = false;
                l = 0;
                std::cout << "Номер символа в образе равен 0\n";
            }
            k++;
            std::cout << "Номер символа в тексте увеличивается на 1 ("
<< k << ")\n";
            std::cin >> curr;
        }
        else if (l == 0) { //Если образ очередной символ не совпадает, а
индекс образа уже указывает на начало
            std::cout << "Символы номер " << k << " в строке и " << l <<
" в образе не совпадают (" << curr << " и " << templ[l] << ")\n";

```



```

        k++;
        std::cout << "Номер слова в тексте увеличивается на 1 (" <<
k << ") \n";

        std::cin >> curr;
        if (std::cin.eof() && isFirst) {
            std::cout << "Номер символа в тексте равен длине
текста, а вхождение так и не было найдено\n";
            res.push_back(-1);
        }
        else { //Если индекс образа не указывает на начало
            std::cout << "Символы номер " << k << " в строке и " << l <<
" в образе не совпадают (" << curr << " и " << templ[l] << ") \n";
            std::cout << "Номер слова в образе становится равным
значению префикс функции символа с номером на один меньше (" << arr[l - 1] <<
") \n";

            l = arr[l - 1]; //Индекс образа становится равным значением
префикс функции для предыдущего символа
        }
    }
    if (std::cin.eof() && res.empty()) { //Если не было найдено совпадений
        res.push_back(-1);
    }
    std::cout << "Строка закончилась\nРезультат:\n";
    for (int i = 0; i < res.size(); i++) {
        if (i != 0) {
            std::cout << ",";
        }
        std::cout << res[i];
    }
    return 0;
}

void makeArr(int* arr, int len, std::string& templ) {
    std::cout << "Вычисление префикс функции:\n";
    arr[0] = 0; //Первый символ - всегда 0
    std::cout << "Значение префикс функции для символа " << templ[0] << "
равно 0\n";
    int j = 0; //Инициализируем индексы ообраза и массива
    int i = 1;
    while (i < len) { // Пока не прошли весь образ
        if (templ[i] == templ[j]) { //Если символы совпадают
            arr[i] = j + 1; //функция для символа на один больше, чем
функция от предыдущего
            std::cout << "Значение префикс функции для символа " <<
templ[i] << " равно" << arr[i] << "\n";
            i++;
            j++;
        }
        else if (j == 0) { //Если не овпадают и индекс в массиве уже 0
            arr[i] = 0; //Функция для символа равна 0
            std::cout << "Значение префикс функции для символа " <<
templ[i] << " равно 0\n";
            i++;
        }
        else { //Если индекс в массиве не 0
            j = arr[j - 1]; //Индекс в массиве равен префикс функции от
предыдущего символа
        }
    }

    std::cout << "Значение префикс функции:\n";

```

```

        for (int i = 0; i < len; i++) {
            std::cout << templ[i] << " - " << arr[i] << std::endl;
        }
    }
}

```

## Программа для вычисления сдвига:

```

#include <iostream>
#include <string>
#include <Windows.h>

void checkCyclicShift(std::string& firstStr, std::string&
secondStr); //Функция вычисления сдвига
int* makePrefix(char* str, int size); //Функция вычисления префикс функции

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    std::string firstStr;
    std::string secondStr;
    std::cin >> firstStr;
    std::cin >> secondStr;
    checkCyclicShift(firstStr, secondStr);
    std::cout << std::endl;
    system("pause");
}

void checkCyclicShift(std::string& firstStr, std::string& secondStr) {
    if (firstStr.length() != secondStr.length()) { //Если размеры строк
разные - возврат
        std::cout << "Размеры строк разные - возврат\n";
        std::cout << -1;
        return;
    }
    else if (firstStr == secondStr) { //Если строки равны - сдвиг равен 0
        std::cout << "Строки равны - возврат\n";
        std::cout << 0;
        return;
    }
    int size = firstStr.size();
    char* buff = new char[size * 3]; //Создается буффер размером с 3 поданные
строки, который заполняется второй строкой и двумя первыми
    int i = 0;
    for (; i < size; ++i) {
        buff[i] = secondStr[i];
    }
    for (int k = 0; k < 2; ++k) {
        for (int j = 0; j < size; ++j) {
            buff[i++] = firstStr[j];
        }
    }
    std::cout << "Создана строка, состоящая из второй строки и двух
первых:\n";
    for (int i = 0; i < size * 3; i++) {
        std::cout << buff[i];
    }
    std::cout << std::endl;

    int* prefixArr = makePrefix(buff, size * 3); //Для буффера вычисляется
префикс функция
    for (int i = 2 * size - 2; i < 3 * size; i++) {

```

```

        if (prefixArr[i] == size) {
            std::cout << "У символа в строке под номером " << i << " ("
<< buff[i] << ") префикс функция равна длине изначальной строки " << size <<
std::endl;

            std::cout << "Результат:\n" << i + 1 - 2 * size;
            delete[] buff;
            delete[] prefixArr;
            return;
        }
    }
    std::cout << "Не было найдено символа с нужным значением префикс
функции\nРезультат:\n";
    std::cout << -1;
    delete[] buff;
    delete[] prefixArr;
}

int* makePrefix(char* templ, int size) {
    std::cout << "Вычисление префикс функции:\n";
    int* prefix = new int[size];
    prefix[0] = 0;
    int j = 0;
    int i = 1;
    std::cout << "Префикс функция для первого символа равна 0, первый индекс
равен 1, второй индекс равен 0\n";
    while (i < size) { // Пока не прошли весь образ
        if (templ[i] == templ[j]) { //Если символы совпадают
            std::cout << "Символы с индексами " << i << " и " << j << "
совпадают (" << templ[i] << "). Оба индекса увеличиваются.\n";
            prefix[i] = j + 1; //функция для символа на один больше, чем
функция от предыдущего
            std::cout << "Значение префикс функции для символа " <<
templ[i] << " равно " << prefix[i] << "\n";
            i++;
            j++;
        }
        else if (j == 0) { //Если не совпадают и индекс в массиве уже 0
            std::cout << "Символы с индексами " << i << " и " << j << "
не совпадают (" << templ[i] << " и " << templ[j] << "). Первый индекс
увеличивается\n";
            prefix[i] = 0; //Функция для символа равна 0
            std::cout << "Значение префикс функции для символа " <<
templ[i] << " равно 0\n";
            i++;
        }
        else { //Если индекс в массиве не 0
            j = prefix[j - 1]; //Индекс в массиве равен префикс функции
от предыдущего символа
            std::cout << "Символы с индексами " << i << " и " << j << "
не совпадают (" << templ[i] << " и " << templ[j] << "). Второй индекс
становится равным значению префикс функции предыдущего символа (" << j <<
")\n";
        }
    }

    std::cout << "Значение префикс функции:\n";
    for (int i = 0; i < size; i++) {
        std::cout << i << "\t" << templ[i] << "\t" << prefix[i] <<
std::endl;
    }
    return prefix;
}

```