

**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: Кнут-Моррис-Пратт**

Студент гр. 9383

\_\_\_\_\_

Хотяков Е.П.

Преподаватель

\_\_\_\_\_

Фирсов М. А.

Санкт-Петербург

2021

## **Цель работы.**

Познакомиться с алгоритмом Кнут-Моррис-Пратта, реализовать алгоритм на одном из языков программирования.

## **Задание.**

### ***1. Поиск индексов вхождения подстроки в строке***

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  $-1$

Sample Input:

ab

abab

Sample Output:

0,2

### ***2. Проверка строки на циклический сдвиг***

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 - В

Выход:

Если А является циклическим сдвигом В, индекс начала строки В в А, иначе вывести -1. Если возможно несколько сдвигов вывести первый индекс.

Sample Input:

defabc

abcdef

Sample Output:

3

### **Основные теоретические положения.**

*Подстрока* — это непустая связная часть строки.

Пусть  $L = c_0 \dots c_{n-1}$  — строка длины  $n$ .

Любая строка  $S = c_i \dots c_j$ , где  $0 \leq i \leq j \leq n-1$ , является *подстрокой*  $L$  длины  $j-i+1$ .

Если  $i=0$ , то  $S$  называется *префиксом*  $L$  длины  $j+1$ .

Если  $j=n-1$ , то  $S$  — *суффикс*  $L$  длины  $j-i+1$ .

*Префикс-функция* от строки  $S$  и позиции  $i$  в ней — длина  $k$  наибольшего собственного (не равного всей подстроке) префикса подстроки  $S[1\dots i]$ , который одновременно является суффиксом этой подстроки.

*Алгоритм Кнута—Морриса—Пратта* (КМП-алгоритм) — эффективный алгоритм, осуществляющий поиск подстроки в строке.

### **Описание алгоритма.**

#### **1. Префикс-функция.**

Элементу с индексом 0 присваиваем значение 0, так как строка длины 1 элемента не рассматривается. Переменная  $i$  отвечает за индекс суффикса,  $j$  за

длину рассматриваемого образца, то есть текущее значение префикс-функции. Далее действует по следующим правилам:

- Если символы не равны и  $j$  отличен от 0, то присваиваем ему значение префикс-функции от  $j-1$ . Если же  $j$  равен 0, то присваиваем нулевое значение префикс-функции текущего элемента  $i$ .
- Если символы равны, то увеличиваем  $j$  на единицу.
- Алгоритм заканчивается, когда  $i$  дойдет до конца строки.

## **2. Алгоритм КМП.**

Дан текст  $t$  и строка  $s$ , требуется найти и вывести позиции всех вхождений строки  $s$  в текст  $t$ .

Обозначим для удобства через  $n$  длину строки  $s$ , а через  $m$  — длину текста  $t$ .

Образуем строку  $s + | + t$ , где символ  $|$  — это разделитель, который не должен нигде более встречаться. Посчитаем для этой строки префикс-функцию. Теперь рассмотрим её значения, кроме первых  $n+1$  (которые, как видно, относятся к строке  $s$  и разделителю). По определению, значение  $pi[i]$  показывает наидлиннейшую длину подстроки, оканчивающейся в позиции  $i$  и совпадающего с префиксом. Но в нашем случае это  $pi[i]$  — фактически длина наибольшего блока совпадения со строкой  $s$  и оканчивающегося в позиции  $i$ . Больше, чем  $n$ , эта длина быть не может — за счёт разделителя. А вот равенство  $pi[i] = n$  (там, где оно достигается), означает, что в позиции  $i$  оканчивается искомое вхождение строки  $s$  (только не надо забывать, что все позиции отсчитываются в склеенной строке  $s + | + t$ ).

Таким образом, если в какой-то позиции  $i$  оказалось  $pi[i] = n$ , то в позиции  $i - (n + 1) - n + 1 = i - 2n$  строки  $t$  начинается очередное вхождение строки  $s$  в строку  $t$ .

Сложность этого алгоритма по операциям будет равна  $O(m+n)$ , где  $n$  — длина подстроки, а  $m$  — длина строки, сложность построения префикс-функции

$O(n)$ , а сложность сравнений в алгоритме  $O(m)$ , так как зависит только от длины строки, по которой мы ходим.

### ***3. Проверка на циклический сдвиг.***

Проверка осуществляется за счет все того же алгоритма КМП. В качестве подстроки мы отправляем строку – оригинал, а в качестве текста – удвоенную строку, которая нуждается в проверке на цикличность. Потом мы просто ищем вхождение подстроки в новую строку, если она была найдена, то строка является циклическим сдвигом.

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

*std::vector<int> PrefixFunction(std::string &KMP\_string)*

Функция для вычисления префикс-функции. Принимает строку и возвращает вектор значение префикс-функций.

*void KMP(std::string &P, std::string &T)*

Функция реализация алгоритма КМП. Принимает на вход подстроку и строку. Также в функции сразу выводится результат работы алгоритма

*void KMP\_Circle(std::string &P, std::string &T)*

Функция для проверки на циклический сдвиг. Принимает основную строку и строку для проверки.

## **Тестирование.**

### **1. Поиск индексов вхождения подстроки в строку**

**Входные данные:**

ab

abab

**Выходные данные:**

0,2

**Входные данные:**

efefe

qqefefeqqefefe

**Выходные данные:**

2,9

**Входные данные:**

abca

qqqq

**Выходные данные:**

-1

### **2. Проверка на циклический сдвиг**

**Входные данные:**

defabc

abcdef

**Выходные данные:**

3

**Входные данные:**

abfdec

abcdef

**Выходные данные:**

-1

**Входные данные:**

ababab

bababa

**Выходные данные:**

1

**Выводы.**

В ходе выполнения лабораторной работы был изучен и реализован алгоритм Кнута—Морриса—Пратта, который находит индексы вхождений подстроки в строку, для чего была реализована функция вычисления префикс-функции. Также был реализован алгоритм проверки строки на циклический сдвиг другой строки на основе алгоритма КМП.

**ПРИЛОЖЕНИЕ А****ИСХОДНЫЙ КОД ПРОГРАММЫ****Файл kmp.cpp:**

```
#include <vector>
```

```

#include <string>

#include <iostream>

std::vector<size_t> prefix_function (const std::string& s)
{
    std::cout << "-----Начало вычисления префикс-функции-
-----\n";

    std::vector<size_t> prefix_array(s.length());

    for (size_t i = 1; i < s.length(); i++)
    {
        size_t j = prefix_array[i-1];
        while ((j > 0) && (s[i] != s[j]))
        {
            j = prefix_array[j-1];
        }
        if (s[i] == s[j])
        {
            j++;
        }

        std::cout << "Значение префикс-функции для первых " <<
i+1 << " элементов = " << j << "\n";

        prefix_array[i] = j;
    }

    std::cout << "-----Конец вычисления префикс-функции-
-----\n\n";

    return prefix_array;
}

```



```

        std::vector<size_t> kmp(const std::string& pattern, const
std::string& text)

    {

        std::vector<size_t> prefix_array = prefix_function(pattern);

        size_t j = 0;

        std::vector<size_t> answer;

        std::cout << "-----Начало вычисления алгоритма Кнут-
Моррис-Пратта-----\n";

        for(size_t i = 0; i < text.size(); i++)
        {

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

            {

                std::cout << "Найдено несовпадение символов <" <<
pattern[j] << "> и <" << text[i] << ">.";

                j = prefix_array[j-1];

                std::cout << " Текущее значение длины цепочки = " <<
j << "\n";

            }

            if(pattern[j] == text[i])

            {

                j++;

                std::cout << "Найдено совпадение символа <" <<
text[i] << ">\n";

                std::cout << "Длина совпадающей цепочки символов = "
<< j << "\n";

            }

            if(j == pattern.size())

            {

```

```

        answer.push_back(i + 1 - pattern.size());

        std::cout << "Найдено выходение подстроки с
индексом: " << i+1-pattern.size() << "\n";

    }

}

    std::cout << "-----Конец вычисления алгоритма Кнут-
Моррис-Патта-----\n\n";

    return answer;

}

int main()
{
    std::string pattern;

    std::string text;

    std::cin >> pattern;

    std::cin >> text;

    if (pattern.length() > text.length())
    {
        std::cout << "Длина подстроки больше длины строки. Код
возврата: -1";

        return 0;
    }

    std::vector<size_t> answer = kmp(pattern, text);

    if(answer.empty())
    {
        std::cout << "Не найдено ни одного вхождения подстроки в
строку. Код возврата: -1";

        return 0;
    }
}

```

```

    }

    std::cout << "-----Полученный результат-----\n";

    for(int i = 0; i < answer.size(); i++)
    {
        if(i == answer.size() - 1)
        {
            std::cout << answer[i];
        }
        else
        {
            std::cout << answer[i] << ",";
        }
    }

    return 0;
}

```

### **Файл cyclic\_shift.cpp:**

```

#include <vector>

#include <string>

#include <iostream>

std::vector<int> prefix_function (const std::string& s)
{
    std::cout << "-----Начало вычисления префикс-функции-
-----\n";

    std::vector<int> prefix_array(s.length());

    for (int i = 1; i < s.length(); i++)

```

```

    {

        int j = prefix_array[i-1];

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

        {

            j = prefix_array[j-1];

        }

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

        {

            j++;

        }

        std::cout << "Значение префикс-функции для первых " <<
i+1 << " элементов = " << j << "\n";

        prefix_array[i] = j;

    }

    std::cout << "-----Конец вычисления префикс-функции-
-----\n\n";

    return prefix_array;

}

std::vector<int> kmp(const std::string& pattern, const
std::string& text)

{

    std::vector<int> prefix_array = prefix_function(pattern);

    int j = 0;

    std::vector<int> answer;

    std::cout << "-----Начало вычисления алгоритма Кнут-
Моррис-Пратта-----\n";

```

```

        for(int i = 0; i < text.size(); i++)
        {
            while((j > 0) && (pattern[j] != text[i]))
            {
                std::cout << "Найдено несовпадение символов <" <<
pattern[j] << "> и <" << text[i] << ">.";
                j = prefix_array[j-1];
                std::cout << " Текущее значение длины цепочки = " <<
j << "\n";
            }
            if(pattern[j] == text[i])
            {
                j++;
                std::cout << "Найдено совпадение символа <" <<
text[i] << ">\n";
                std::cout << "Длина совпадающей цепочки символов = "
<< j << "\n";
            }
            if(j == pattern.size())
            {
                answer.push_back(i + 1 - pattern.size());
                std::cout << "Найдено выходение подстроки с
индексом: " << i+1-pattern.size() << "\n";
            }
        }

        std::cout << "-----Конец вычисления алгоритма Кнут-
Моррис-Пратта-----\n\n";

        return answer;
    }

```

```

        std::vector<int> cyclic_shift(const std::string& pattern, const
std::string& text)

    {

        std::cout << "-----Начало проверки на циклический
сдвиг-----\nСтрока A: " << text << "\n";

        std::cout << "Строка B, измененная для поиска: " <<
pattern+pattern << "\n\n";

        std::vector<int> answer = kmp(text, pattern+pattern);

        std::cout << "-----Конец проверки на циклический
сдвиг-----\n";

        return answer;

    }

int main()

{

    std::string pattern;

    std::string text;

    std::cin >> pattern;

    std::cin >> text;

    if (pattern.size() != text.size())

    {

        std::cout << "-1";

        return 0;

    }

    if (pattern.size() == 0 || text.size() == 0)

    {

        std::cout << "Одна из строк пуста. Код возврата: -1";

```

```

        return 0;
    }

    std::vector<int> answer = cyclic_shift(pattern, text);

    if (answer.empty())
    {
        std::cout << "Строка В не является циклическим сдвигом
строки А. Код возврата: -1";

        return 0;
    }

    std::cout << "Строка В является циклическим сдвигом строки А.
Первый индекс сдвига: " << answer[0];

    return 0;
}

```