

**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: Коммивояжер**  
**Алгоритм Кнута-Морриса-Пратта**

Студентка гр. 1304

\_\_\_\_\_

Хорошкова А.С.

Преподаватель

\_\_\_\_\_

Шевелева А.М.

Санкт-Петербург

2023

## **Цель работы.**

Изучение алгоритма Кнута-Морриса-Пратта для поиска подстроки в строке. Модификация алгоритма для установления размера сдвига одной строки относительно другой.

## **Задание.**

1) Первое 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  $T$ .

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

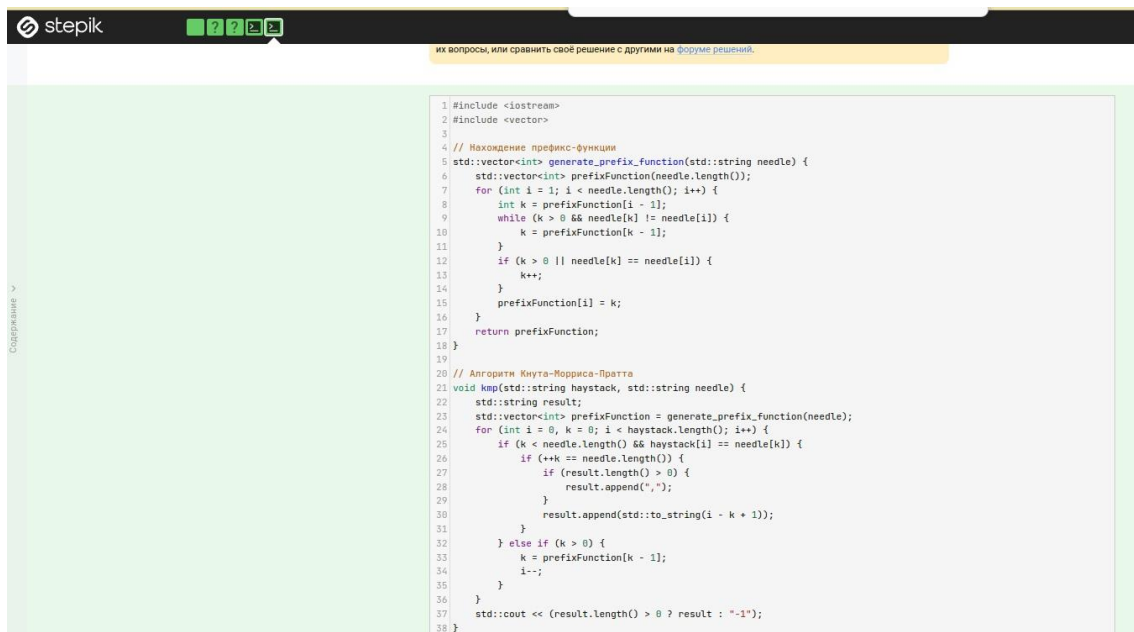
## **Выполнение работы.**

### **Описание алгоритма.**

1) Первое задание. Был использован алгоритм поиска подстроки Кнута-Морриса-Пратта. Был построен вектор префикс-функций на основе строки *needle* (игла) с помощью функции `generate_prefix_function()`. Далее производится поиск строки *needle* в строке *haystack* (стог сена) с помощью функции `kmp()`. Благодаря ранее найденной префикс-функции алгоритм «перескакивает» заведомо совпавшие участки строк, продолжая поиск в месте с неисследованными буквами. Затем если найдены вхождения *needle* в *haystack*, то выводится все индексы этих вхождений, если вхождения не найдены, то выводится  $-1$ .

2) Второе задание. Был модифицирован алгоритм Кнута-Морриса-Пратта в методе `kmp_circle()`. Внесены следующие изменения в классический алгоритм. Если длины строк *needle* и *haystack* не равны, то программа завершается с выводом  $-1$  (строки не являются сдвигом друг друга). Также выводится только первое нахождение *needle*. Далее модифицированный алгоритм запускается на строке *needle* удвоенной строке *haystack*.

Оба алгоритма успешно прошли все тесты на платформе stepik. Результаты представлены на Рисунок 1 - скриншот задания на платформе stepik.



```
1 #include <iostream>
2 #include <vector>
3
4 // Нахождение префикс-функции
5 std::vector<int> generate_prefix_function(std::string needle) {
6     std::vector<int> prefixFunction(needle.length());
7     for (int i = 1; i < needle.length(); i++) {
8         int k = prefixFunction[i - 1];
9         while (k > 0 && needle[k] != needle[i]) {
10             k = prefixFunction[k - 1];
11         }
12         if (k > 0 || needle[k] == needle[i]) {
13             k++;
14         }
15         prefixFunction[i] = k;
16     }
17     return prefixFunction;
18 }
19
20 // Алгоритм Кнута-Морриса-Пракса
21 void kmp(std::string haystack, std::string needle) {
22     std::vector<int> prefixFunction = generate_prefix_function(needle);
23     for (int i = 0, k = 0; i < haystack.length(); i++) {
24         if (k < needle.length() && haystack[i] == needle[k]) {
25             if (++k == needle.length()) {
26                 if (result.length() > 0) {
27                     result.append(",");
28                 }
29                 result.append(std::to_string(i - k + 1));
30             }
31         } else if (k > 0) {
32             k = prefixFunction[k - 1];
33         }
34     }
35     std::cout << (result.length() > 0 ? result : "-1");
36 }
```

Рисунок 1 - скриншот задания на платформе stepik

## Описание функций и структур данных.

Для реализации работы двух алгоритмов были реализованы следующие функции:

`std::vector<int> generate_prefix_function(std::string needle)` — функция, принимающая на вход строки `needle`, и возвращающая на выходе вектор со значениями префикс-функции.

`void kmp(std::string haystack, std::string needle)` — функция, реализующая алгоритм Кнута-Морриса-Пракса. Она выводит на экран индексы букв, с которых начинаются все вхождения строки `needle` в строку `haystack`. Если вхождений не найдено, возвращает -1.

`void kmp_circle(std::string haystack, std::string needle)` — функция, реализующая модификацию алгоритма Кнута-Морриса-Пракса для определения, являются ли строки циклическим сдвигом друг друга. функция выводит на экран индекс начала строки `needle` в строке `haystack`. Если `haystack` не является циклическим сдвигом `needle`, возвращается -1.

`int main()` — точка входа в программу. В методе `main()` считываются строки `needle` и `haystack` и запускается необходимая в данный момент функция (`kmp` или `kmp_circle`).

### **Выводы.**

В ходе лабораторной работы на практике был изучен алгоритм поиска подстроки в строке Кнута-Морриса-Пратта.

В рамках реализации первой задачи был написан классический алгоритм Кнута-Морриса-Пратта. Алгоритм находит первую строку во второй и выводит все индексы начала вхождения первой строки во вторую либо -1, если вхождения не найдены.

В рамках реализации второй задачи был модифицирован алгоритм Кнута-Морриса-Пратта таким образом, чтобы он определял, является ли одна строка циклическим сдвигом второй и выводит либо величину сдвига, либо -1, если строки не являются циклическим сдвигом друг друга.

Оба алгоритма успешно прошли все тесты на платформе stepik. Результаты представлены на Рисунок 1 - скриншот задания на платформе stepik.