

**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**

Студентка гр. 9382

Пя С.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2021

### **Цель работы.**

Разработать программу, реализующую поиск подстроки в строке с помощью алгоритма Кнута-Морриса-Пратта. Также разработать программу, определяющую, является ли первая строка циклическим сдвигом второй строки.

### **Задание 1**

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

### **Задание 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.

**Sample Input:**

defabc

abcdef

**Sample Output:**

3

**Описание алгоритма**

На вход подаются подстрока, которую нужно найти в строке, и строка. Алгоритм начинается с вычисления префикс функции для подстроки. Для этого создается список для префикс-функций, по величине равный количеству символов в подстроке.

Префикс – это подстрока, начинающиеся с начала строки.

Суффикс – это подстрока, заканчивающиеся в конце строки.

Префикс и суффикс должны быть равны при сравнении.

Префикс-функция для  $i$ -го символа строки – функция, показывающая максимальную длину совпадающих префикса и суффикса подстроки в строке, которая заканчивается  $i$ -м символом. Для первого элемента префикс-функция будет равна нулю, так как длина подстроки будет равна единице.

В первом задании необходимо найти все вхождения подстроки в строке. Сравниваются символ из подстроки и символ из строки.

В случае их равенства мы начинаем сравнивать следующий символ подстроки и следующий символ строки, если к этому моменту символы подстроки были полностью рассмотрены, значит подстрока была найдена в строке, в ответ записывается индекс начала вхождения подстроки в строку, а

дальше мы будем сравнивать символ подстроки под индексом префикс-функции предыдущего символа, так как подстрока может сливаться с подстрокой при поиске в строке.

В случае неравенства символов, если рассматриваемый символ является первым в подстроке, берем для сравнения следующий символ строки.

В случае неравенства символов, если рассматриваемый символ не является первым в подстроке, следующим рассматриваемым символом в подстроке станет символ под индексом префикс-функции предыдущего символа, чтобы мы учли то, что в рассмотренных символах могла быть часть подстроки.

Алгоритм заканчивается, когда мы сравним все символы строки.

Во втором задании необходимо определить, является ли первая строка циклическим сдвигом второй. Для этого достаточно воспользоваться алгоритмом из предыдущего задания, выполняя поиск второй строки в удвоенной первой. Так как при сложении строк первая будет содержать в себе вторую строку, если первая строка является циклическим сдвигом второй. Алгоритм заканчивается при нахождении первого вхождения второй строки в модифицированной первой либо при сравнении всех символов модифицированной первой строки и не нахождении вхождения. Если данные строки изначально не равны, то по определению первая строка не является циклическим сдвигом второй.

### **Оценка сложности по памяти.**

В обеих программах проходимся по двум строкам и используем вектор со значениями префикс-функции для подстроки. В худшем случае мы не найдем подстроку в строке, тогда сложность составляет  $O(2*N + M)$  для первой программы, где  $N$  – длина первой строки или размер вектора префикс-функции,  $M$  – длина второй строки, и сложность для второй программы

составляет  $O(N + M^2 \cdot 2)$ , так как в худшем случае мы пройдем по всей удвоенной строке.

### Оценка сложности по времени.

Значение префикс-функции вычисляется за  $O(N)$  сравнений, где  $N$  – длина подстроки, так как необходимо обойти всю строку, чтобы определить префикс-функцию.

Поиск подстроки в строке с помощью алгоритма КМП будет занимать  $O(M)$ , где  $M$  – длина строки, так как вся строка будет пройдена один раз. Каждый символ строки будет рассмотрен один раз благодаря префикс-функции.

Итоговая оценка –  $O(N + M)$ .

Для второй программы оценка сложности по времени вычисляется так же.

### Тестирование.

#### Тестирование первой программы.

Нумерация	Входные данные	Выходные данные
1	abbaabbab abbaabbaabbaabbababbaabbab	Хотите считать данные из файла или ввести самостоятельно?(1/2) 1 Начинаем подсчет префикс функции $i$ - индекс первого символа для сравнения, $j$ - индекс второго символа для сравнения Символы не одинаковы, смещаем $i$ "a""b"baabbab 0 0 Символы не одинаковы, смещаем $i$ "a"b"b"aabbab 0 0 0 Символы одинаковы, смещаем $j$ и $i$

	<p>"a"bb"a"abbab</p> <p>0 0 0 1</p> <p>j не равен нулю, и символы не одинаковы</p> <p>Присвоим j значение префикса предыдущего символа, на который указывала j</p> <p>a"b"ba"a"bbab</p> <p>j = 0</p> <p>Символы одинаковы, смещаем j и i</p> <p>"a"bba"a"bbab</p> <p>0 0 0 1 1</p> <p>Символы одинаковы, смещаем j и i</p> <p>a"b"baa"b"bab</p> <p>0 0 0 1 1 2</p> <p>Символы одинаковы, смещаем j и i</p> <p>ab"b"aab"b"ab</p> <p>0 0 0 1 1 2 3</p> <p>Символы одинаковы, смещаем j и i</p> <p>abb"a"abb"a"b</p> <p>0 0 0 1 1 2 3 4</p> <p>j не равен нулю, и символы не одинаковы</p> <p>Присвоим j значение префикса предыдущего символа, на который указывала j</p> <p>abba"a"bba"b"</p> <p>j = 1</p> <p>Символы одинаковы, смещаем j и i</p> <p>a"b"baabba"b"</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Вывод префикс функции</p> <p>a b b a a b b a b</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Начинаем поиск подстроки в строке</p>
--	---

	<p>k - индекс символа строки для сравнения,  l - индекс символа подстроки для  сравнения</p> <p>Символы одинаковы, смещаем k и l</p> <p>"a"bbaabbaabbaabbababbaabbab</p> <p>"a"bbaabbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и l</p> <p>a"b"baabbaabbaabbababbaabbab</p> <p>a"b"baabbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и l</p> <p>ab"b"aabbaabbaabbababbaabbab</p> <p>ab"b"aabbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и l</p> <p>abb"a"abbaabbaabbababbaabbab</p> <p>abb"a"abbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и l</p> <p>abba"a"bbaabbaabbababbaabbab</p> <p>abba"a"bbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaa"b"baabbaabbababbaabbab</p> <p>abbaa"b"bab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaab"b"aabbaabbababbaabbab</p> <p>abbaab"b"ab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaabb"a"abbaabbababbaabbab</p>
--	--

	<p>abbaabb"a"b</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы не одинаковы</p> <p>Присваиваем 1 значение префикса предыдущего символа, на который указывала 1</p> <p>abbaabba"a"bbaabbababbaabbab</p> <p>abbaabba"b"</p> <p>0 0 0 1 1 2 3 4 2</p> <p>l = 4</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaabba"a"bbaabbababbaabbab</p> <p>abba"a"bbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaabbaa"b"baabbababbaabbab</p> <p>abbaa"b"bab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaabbaab"b"aabbababbaabbab</p> <p>abbaab"b"ab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaabbaabb"a"abbababbaabbab</p> <p>abbaabb"a"b</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы не одинаковы</p> <p>Присваиваем 1 значение префикса предыдущего символа, на который указывала 1</p> <p>abbaabbaabba"a"bbababbaabbab</p> <p>abbaabba"b"</p> <p>0 0 0 1 1 2 3 4 2</p>
--	--



	<p><math>l = 4</math></p> <p>Символы одинаковы, смещаем <math>k</math> и <math>l</math></p> <p>abbaabbaabba"a"bbababbaabbab</p> <p>abba"a"bbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем <math>k</math> и <math>l</math></p> <p>abbaabbaabbaa"b"bababbaabbab</p> <p>abbaa"b"bab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем <math>k</math> и <math>l</math></p> <p>abbaabbaabbaab"b"ababbaabbab</p> <p>abbaab"b"ab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем <math>k</math> и <math>l</math></p> <p>abbaabbaabbaabb"a"babbaabbab</p> <p>abbaabb"a"b</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем <math>k</math> и <math>l</math></p> <p>abbaabbaabbaabba"b"abbaabbab</p> <p>abbaabba"b"</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Подстрока найдена!</p> <p>Индекс начала вхождения подстроки в строке 8</p> <p>abbaabbaabbaabbab"a"bbaabbab</p> <p>abbaabbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Присваиваем <math>l</math> значение префикс-функции предыдущего символа, на который указывала <math>l</math></p> <p><math>l = 2</math></p> <p>Символы не одинаковы</p>
--	---

	<p>Присваиваем 1 значение префикса предыдущего символа, на который указывала 1</p> <p>abbaabbaabbaabbab"a"bbaabbab</p> <p>ab"b"aabbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>1 = 0</p> <p>Символы одинаковы, смещаем k и 1</p> <p>abbaabbaabbaabbab"a"bbaabbab</p> <p>"a"bbaabbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и 1</p> <p>abbaabbaabbaabbaba"b"baabbab</p> <p>a"b"baabbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и 1</p> <p>abbaabbaabbaabbabab"b"aabbab</p> <p>ab"b"aabbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и 1</p> <p>abbaabbaabbaabbababb"a"abbab</p> <p>abb"a"abbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и 1</p> <p>abbaabbaabbaabbababba"a"bbab</p> <p>abba"a"bbab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и 1</p> <p>abbaabbaabbaabbababbaa"b"bab</p> <p>abbaa"b"bab</p> <p>0 0 0 1 1 2 3 4 2</p> <p>Символы одинаковы, смещаем k и 1</p> <p>abbaabbaabbaabbababbaa"b"ab</p>
--	--

		abbaab"b"ab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaabbababbaabb"a"b abbaabb"a"b 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaabbababbaabba"b" abbaabba"b" 0 0 0 1 1 2 3 4 2 Подстрока найдена! Индекс начала вхождения подстроки в строке 17 abbaabbaabbaabbababbaabbab abbaabbab 0 0 0 1 1 2 3 4 2 Присваиваем l значение префикс-функции предыдущего символа, на который указывала l $l = 2$ Подстрока входит в строку Индексы начала вхождения подстроки в строку 8,17 Хотите продолжить?(y/n) n
2	abcasda sadfasf	Подстрока не входит в строку -1
3	asddsad asd	Подстрока не входит в строку -1
4	qwe qwerty	Подстрока входит в строку 0

5	love lovelovelovelovelovelveovevloe	Индексы начала вхождения подстроки в строку 0,4,8,12,16,20
---	--	---

### Тестирование второй программы.

Нумерация	Входные данные	Выходные данные
1	abbaabbab abbababba	<p>Хотите считать данные из файла или ввести самостоятельно?(1/2)</p> <p>Для того чтобы определить, является ли А циклическим сдвигом В, воспользуемся алгоритмом поиска подстроки в строке</p> <p>В нашем случае будем искать строку В в модифицированной А строке</p> <p>Модифицируем строку А, сложив ее с собой</p> <p>Начинаем подсчет префикс функции</p> <p>i - индекс первого символа для сравнения, j - индекс второго символа для сравнения</p> <p>Символы не одинаковы, смещаем i</p> <p>"a""b"bababba</p> <p>0 0</p> <p>Символы не одинаковы, смещаем i</p> <p>"a"b"b"ababba</p> <p>0 0 0</p> <p>Символы одинаковы, смещаем j и i</p> <p>"a"bb"a"babba</p> <p>0 0 0 1</p> <p>Символы одинаковы, смещаем j и i</p> <p>a"b"ba"b"abba</p> <p>0 0 0 1 2</p> <p>j не равен нулю, и символы не одинаковы</p>

	<p>Присвоим <math>j</math> значение префикса предыдущего символа, на который указывала <math>j</math></p> <p>ab"b"ab"a"bba</p> <p><math>j = 0</math></p> <p>Символы одинаковы, смещаем <math>j</math> и <math>i</math></p> <p>"a"bbab"a"bba</p> <p>0 0 0 1 2 1</p> <p>Символы одинаковы, смещаем <math>j</math> и <math>i</math></p> <p>a"b"baba"b"ba</p> <p>0 0 0 1 2 1 2</p> <p>Символы одинаковы, смещаем <math>j</math> и <math>i</math></p> <p>ab"b"abab"b"a</p> <p>0 0 0 1 2 1 2 3</p> <p>Символы одинаковы, смещаем <math>j</math> и <math>i</math></p> <p>abb"a"babb"a"</p> <p>0 0 0 1 2 1 2 3 4</p> <p>Вывод префикс функции</p> <p>a b b a b a b b a</p> <p>0 0 0 1 2 1 2 3 4</p> <p>Определяем, является ли A циклическим сдвигом B</p> <p><math>k</math> - индекс символа строки A, <math>l</math> - индекс символа строки B</p> <p>Символы одинаковы, смещаем <math>k</math> и <math>l</math></p> <p>"a"bbaabbababbaabbab</p> <p>"a"bbababba</p> <p>0 0 0 1 2 1 2 3 4</p> <p>Символы одинаковы, смещаем <math>k</math> и <math>l</math></p> <p>a"b"baabbababbaabbab</p> <p>a"b"bababba</p> <p>0 0 0 1 2 1 2 3 4</p> <p>Символы одинаковы, смещаем <math>k</math> и <math>l</math></p> <p>ab"b"aabbababbaabbab</p> <p>ab"b"ababba</p> <p>0 0 0 1 2 1 2 3 4</p>
--	--

	<p>Символы одинаковы, смещаем k и l</p> <p>abb"a"abbababbaabbab</p> <p>abb"a"babba</p> <p>0 0 0 1 2 1 2 3 4</p> <p>Символы не одинаковы</p> <p>Присваиваем l значение префикса предыдущего символа, на который указывала l</p> <p>abba"a"bbababbaabbab</p> <p>abba"b"abba</p> <p>0 0 0 1 2 1 2 3 4</p> <p>l = 1</p> <p>Символы не одинаковы</p> <p>Присваиваем l значение префикса предыдущего символа, на который указывала l</p> <p>abba"a"bbababbaabbab</p> <p>a"b"bababba</p> <p>0 0 0 1 2 1 2 3 4</p> <p>l = 0</p> <p>Символы одинаковы, смещаем k и l</p> <p>abba"a"bbababbaabbab</p> <p>"a"bbababba</p> <p>0 0 0 1 2 1 2 3 4</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaa"b"bababbaabbab</p> <p>a"b"bababba</p> <p>0 0 0 1 2 1 2 3 4</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaab"b"ababbaabbab</p> <p>ab"b"ababba</p> <p>0 0 0 1 2 1 2 3 4</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaabb"a"babbaabbab</p> <p>abb"a"babba</p>
--	--

		<p>0 0 0 1 2 1 2 3 4</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaabba"b"abbaabbab</p> <p>abba"b"abba</p> <p>0 0 0 1 2 1 2 3 4</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaabbab"a"bbaabbab</p> <p>abbab"a"bba</p> <p>0 0 0 1 2 1 2 3 4</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaabbaba"b"baabbab</p> <p>abbaba"b"ba</p> <p>0 0 0 1 2 1 2 3 4</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaabbabab"b"aabbab</p> <p>abbabab"b"a</p> <p>0 0 0 1 2 1 2 3 4</p> <p>Символы одинаковы, смещаем k и l</p> <p>abbaabbababb"a"abbab</p> <p>abbababb"a"</p> <p>0 0 0 1 2 1 2 3 4</p> <p>В в модифицированной А найдена!</p> <p>Индекс начала строки В в А 4</p> <p>abbaabbababba"a"bbab</p> <p>abbababba</p> <p>0 0 0 1 2 1 2 3 4</p> <p>В является циклическим сдвигом А</p> <p>Индекс начала В в А</p> <p>4</p> <p>Хотите продолжить?(y/n)</p> <p>n</p>
2	asd adff	<p>В не является циклическим сдвигом А</p> <p>-1</p>

3	iloveyou youlovei	В не является циклическим сдвигом А -1
4	iloveyou youilove	Индекс начала В в А 5
5	abcdef fabcde	Индекс начала В в А 6

### **Выводы.**

Была разработана программа, занимающаяся поиском подстроки в строке и находящая все ее вхождения, и был изучен алгоритм Кнута-Морриса-Пратта. Также была реализована программа, определяющая, является ли строка циклическим сдвигом другой строки.



## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab4\_1.cpp

```
#include <iostream>
#include <vector>
#include <fstream>

class KMP {
private:
    std::string P, T;
    std::vector<int> answer;
    std::vector<size_t> prefix;
public:
    void readData(std::istream& fin) {
        fin >> P >> T;
    }

    void writePrefix(bool withP, int n) {
        if (withP) {
            std::cout << "Вывод префикс функции\n";
            for (auto i : P) {
                std::cout << i << " ";
            }
            std::cout << "\n";
        }
        for (int i = 0; i < n; i++) {
            std::cout << prefix[i] << " ";
        }
        std::cout << "\n";
    }

    void writeStepForPorT(int j, int i, bool isP) {
        std::string temp = P;
        if (!isP)
            temp = T;
        for (int l = 0; l < temp.length(); l++) {
            if (l == j || l == i)
                std::cout << "\"" << temp[l] << "\"";
            else
                std::cout << temp[l];
        }
        std::cout << "\n";
    }

    void calculatePrefix() {
        std::cout << "Начинаем подсчет префикс функции\n";
        std::cout << "i - индекс первого символа для сравнения, j - индекс  
второго символа для сравнения\n";
        size_t n = P.length();
        prefix = std::vector<size_t>(n);
        for (size_t i = 1; i < n; ++i) {
            size_t j = prefix[i - 1];
            while ((j > 0) && (P[i] != P[j])) {
                std::cout << "j не равен нулю, и символы не одинаковы\n";
                std::cout << "Присвоим j значение префикса предыдущего  
символа, на который указывала j\n";
                writeStepForPorT(j, i, true);
                j = prefix[j - 1];
            }
        }
    }
};
```

```

        std::cout << "j = " << j << "\n";
    }
    if (P[i] == P[j]) {
        std::cout << "Символы одинаковы, смещаем j и i\n";
        writeStepForPorT(j, i, true);
        ++j;
    } else {
        std::cout << "Символы не одинаковы, смещаем i\n";
        writeStepForPorT(j, i, true);
    }
    prefix[i] = j;
    writePrefix(false, i + 1);
}
writePrefix(true, n);
}

void writeVisualisation(int k, int l) {
    writeStepForPorT(k, k, false);
    writeStepForPorT(l, l, true);
    writePrefix(false, prefix.size());
}

void doAlgoritm() {
    int k = 0, l = 0;
    size_t n = P.length(), m = T.length();
    calculatePrefix();
    std::cout << "Начинаем поиск подстроки в строке\n";
    std::cout << "k - индекс символа строки для сравнения, l - индекс
символа подстроки для сравнения\n";
    while (k < m) {
        if (T[k] == P[l]) {
            std::cout << "Символы одинаковы, смещаем k и l\n";
            writeVisualisation(k, l);
            k++;
            l++;
            if (l == n) {
                std::cout << "Подстрока найдена!\nИндекс начала вхождения
подстроки в строке " << k - l << "\n";
                writeVisualisation(k, l);
                answer.push_back(k - l);
                std::cout << "Присваиваем l значение префикс-функции
предыдущего символа, на который указывала l\n";
                l = prefix[l - 1];
                std::cout << "l = " << l << "\n";
            }
        } else if (l == 0) {
            std::cout << "Символы не одинаковы, l = 0, смещаем k\n";
            writeVisualisation(k, l);
            k++;
        } else {
            std::cout << "Символы не одинаковы\n";
            std::cout << "Присваиваем l значение префикса предыдущего
символа, на который указывала l\n";
            writeVisualisation(k, l);
            l = prefix[l - 1];
            std::cout << "l = " << l << "\n";
        }
    }
}

void writeAnswer() {
    if (answer.empty()) {

```

```

        std::cout << "Подстрока не входит в строку\n";
        std::cout << "-1";
        return;
    }
    std::cout << "Подстрока входит в строку\nИндексы начала вхождения
подстроки в строку\n";
    for (auto i : answer) {
        if (i == answer.front())
            std::cout << i;
        else
            std::cout << "," << i;
    }
    std::cout << "\n";
}

};

void startProgram() {
    char answ = 'y';
    while (answ == 'y') {
        std::cout << "Хотите считать данные из файла или ввести
самостоятельно?(1/2)\n";
        std::cin >> answ;
        KMP *kmp = new KMP();
        if (answ == '2') {
            std::cout << "Введите подстроку и строку\n";
            kmp->readData(std::cin);
        } else {
            std::ifstream fin("test1.txt");
            kmp->readData(fin);
            fin.close();
        }
        kmp->doAlgoritm();
        kmp->writeAnswer();
        std::cout << "Хотите продолжить?(y/n)\n";
        std::cin >> answ;
    }
}

int main() {
    startProgram();
    return 0;
}

```

## Название файла: lab4\_2.cpp

```
#include <iostream>
#include <vector>
#include <fstream>
class KMP {
private:
    std::string B, A;
    int answer;
    std::vector<size_t> prefix;
public:
    void readData(std::istream& fin){
        fin >> A >> B;
        std::cout << "Для того чтобы определить, является ли A циклическим
сдвигом B, воспользуемся алгоритмом поиска подстроки в строке\n";
        std::cout << "В нашем случае будем искать строку B в модифицированной
A строке\n";
        std::cout << "Модифицируем строку A, сложив ее с собой\n";
        A = A + A;
    }

    void writePrefix(bool withP, int n) {
        if (withP) {
            std::cout << "Вывод префикс функции\n";
            for (auto i : B) {
                std::cout << i << " ";
            }
            std::cout << "\n";
        }
        for (int i = 0; i < n; i++) {
            std::cout << prefix[i] << " ";
        }
        std::cout << "\n";
    }

    void writeStepForPorT(int j, int i, bool isP) {
        std::string temp = B;
        if (!isP)
            temp = A;
        for (int l = 0; l < temp.length(); l++) {
            if (l == j || l == i)
                std::cout << "\"" << temp[l] << "\"";
            else
                std::cout << temp[l];
        }
        std::cout << "\n";
    }

    void calculatePrefix() {
        std::cout << "Начинаем подсчет префикс функции\n";
        std::cout << "i - индекс первого символа для сравнения, j - индекс
второго символа для сравнения\n";
        size_t n = B.length();
        prefix = std::vector<size_t>(n);
        for (size_t i = 1; i < n; ++i) {
            size_t j = prefix[i - 1];
            while ((j > 0) && (B[i] != B[j])) {
                std::cout << "j не равен нулю, и символы не одинаковы\n";
                std::cout << "Присвоим j значение префикса предыдущего
символа, на который указывала j\n";
                writeStepForPorT(j, i, true);
            }
        }
    }
};
```

```

        j = prefix[j - 1];
        std::cout << "j = " << j << "\n";
    }
    if (B[i] == B[j]) {
        std::cout << "Символы одинаковы, смещаем j и i\n";
        writeStepForPorT(j, i, true);
        ++j;
    } else {
        std::cout << "Символы не одинаковы, смещаем i\n";
        writeStepForPorT(j, i, true);
    }
    prefix[i] = j;
    writePrefix(false, i + 1);
}
writePrefix(true, n);
}

void writeVisualisation(int k, int l) {
    writeStepForPorT(k, k, false);
    writeStepForPorT(l, l, true);
    writePrefix(false, prefix.size());
}

void doAlgoritm() {
    int k = 0, l = 0;
    answer = -1;
    size_t n = B.length(), m = A.length();
    if (n != m / 2) {
        std::cout << "В и А не имеют одинаковую длину\n";
        return;
    }
    calculatePrefix();
    std::cout << "Определяем, является ли А циклическим сдвигом В\n";
    std::cout << "k - индекс символа строки А, l - индекс символа строки
В\n";
    while (k < m) {
        if (A[k] == B[l]) {
            std::cout << "Символы одинаковы, смещаем k и l\n";
            writeVisualisation(k, l);
            k++;
            l++;
            if (l == n) {
                std::cout << "В в модифицированной А найдена!\nИндекс
начала строки В в А " << k - l << "\n";
                writeVisualisation(k, l);
                answer = k - l;
                return;
            }
        } else if (l == 0) {
            std::cout << "Символы не одинаковы, l = 0, смещаем k\n";
            writeVisualisation(k, l);
            k++;
        } else {
            std::cout << "Символы не одинаковы\n";
            std::cout << "Присваиваем l значение префикса предыдущего
символа, на который указывала l\n";
            writeVisualisation(k, l);
            l = prefix[l - 1];
            std::cout << "l = " << l << "\n";
        }
    }
}
}

```

```

void writeAnswer() {
    if (answer == -1) {
        std::cout << "В не является циклическим сдвигом A\n";
    } else {
        std::cout << "В является циклическим сдвигом A\nИндекс начала В в
A\n";
    }
    std::cout << answer << "\n";
}

};

void startProgram() {
    char answ = 'y';
    while (answ == 'y') {
        std::cout << "Хотите считать данные из файла или ввести
самостоятельно? (1/2) \n";
        std::cin >> answ;
        KMP *kmp = new KMP();
        if (answ == '2') {
            std::cout << "Введите А и В\n";
            kmp->readData(std::cin);
        } else {
            std::ifstream fin("test2.txt");
            kmp->readData(fin);
            fin.close();
        }
        kmp->doAlgoritm();
        kmp->writeAnswer();
        std::cout << "Хотите продолжить? (y/n) \n";
        std::cin >> answ;
    }
}

int main() {
    startProgram();
    return 0;
}

```