**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. 8303 _____ Рудько Д.Ю.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2020

**Цель работы**

Изучение алгоритма Кнута-Морриса-Пратта поиска образца в строке.

**Задание**

• Задание 1

Реализуйте алгоритм КМП и с его помощью для заданных образца P ($|P| \leqslant 15000$) и текста T ($|T| \leqslant 5000000$) 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 ($|A| \leqslant 5000000$) и B ($|B| \leqslant 5000000$). 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 вляется 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

**Индивидуализация вариант 1**

Подготовка к распараллеливанию: работа по поиску разделяется на k равных частей, пригодных для обработки k потоками (при этом длина образца гораздо меньше длины строки поиска).

**Описание алгоритмов**

**КМП**

Рассмотрим сравнение строк на позиции $i$, где образец $S[0, m-1]$ сопоставляется с частью текста $T[i, i+m-1]$. Предположим, что первое несовпадение произошло между $T[i+j]$ и $S[j]$, где $1 < j < m$. Тогда $T[i, i+j-1]=S[0, j-1]=P$ и $a = T[i+j] \neq S[j] = b$.

При сдвиге вполне можно ожидать, что префикс (начальные символы) образца $S$ сойдется с каким-нибудь суффиксом (конечные символы) текста $P$. Длина наиболее длинного префикса, являющегося одновременно суффиксом, есть значение префикс-функции от строки $S$ для индекса $j$.

Это приводит нас к следующему алгоритму: пусть $pi[j]$ — значение префикс-функции от строки $S[0, m-1]$ для индекса $j$. Тогда после сдвига мы можем возобновить сравнения с места $T[i+j]$ и $S[pi[j]]$ без потери возможного местонахождения образца.

**Циклический сдвиг**

В данном алгоритме можно обойтись без удваивания строки. В самом начале происходит проверка на соответствие длин строк. Если соответствия не было обнаружено, то выводится -1. Создаются два счётчика для первой и второй строки. Далее сравниваются символы первой и второй строки, если символы совпадают переход к следующим, счётчики увеличиваются, если совпадения не обнаружено, счётчик для второй строки увеличиваются. В том случае, если счётчик второй строки равен её длине, то сдвиг найден, а если

счётчик первой строки равен её длине, то происходит его обнуление, таким образом строка зацикливается.

**Префикс функция**

Префикс-функция от строки и позиции в ней — длина наибольшего собственного префикса подстроки, который одновременно является суффиксом этой подстроки. То есть, в начале подстроки длины нужно найти такой префикс максимальной длины , который был бы суффиксом данной подстроки .

Сложность алгоритма КМП :

O(n + m), n – длина подстроки, m– длина строки.

Сложность алгоритма поиска циклического сдвига:

O (n+n) = O(n).

**Описание функций**

1) vector<int> prefix_function (string s)

Возвращает значение префикс функции для строки.

Принимает переменную типа string

2) vector<int> **KMP**(string t, string p, vector<int> &pi)

Функция нахождения образца в тексте алгоритмом  Кнута-Морриса-Пратта.

Принимает переменные типа string и ссылку на вектор

string t — исходный текст

string p - образец

vector<int> &pi — ссылка на вектор значений префикс-функции.

3) void **split**(string t, string p, int k, vector<string> &str, vector<int> &ans_current, vector<int> &ans, vector<int> &pi)

Функция разделения исходного текста на части.

Принимает переменные типа string, int и ссылки на 4 вектора

string t — исходный текст

string p — образец

int k — число частей исходного текста

vector<int> &str — ссылка на вектор хранящий части строк исходного текста

vector<int> &ans_current — ссылка на вектор ответов для текущей части исходного текста

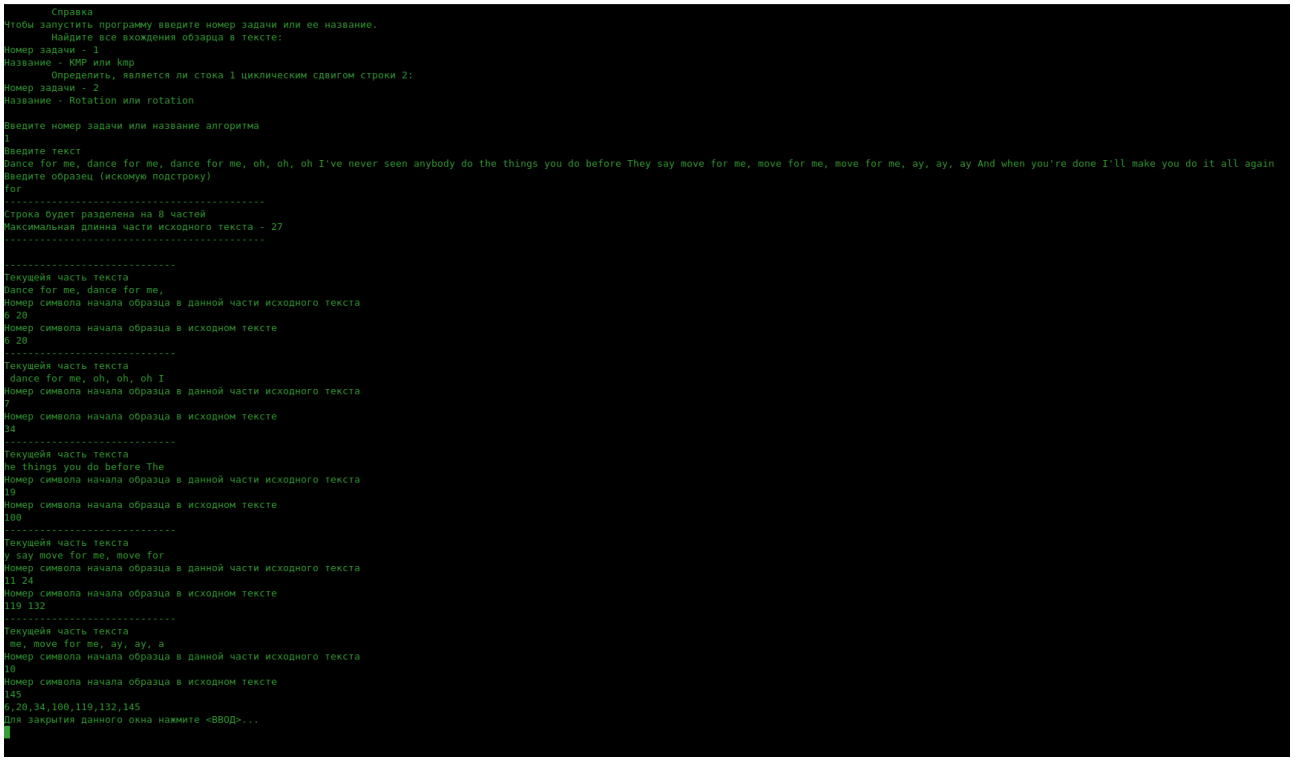
vector<int> &ans — ссылка на вектор ответов для всего текста

vector<int> &pi — ссылка на вектор значений префикс-функции.

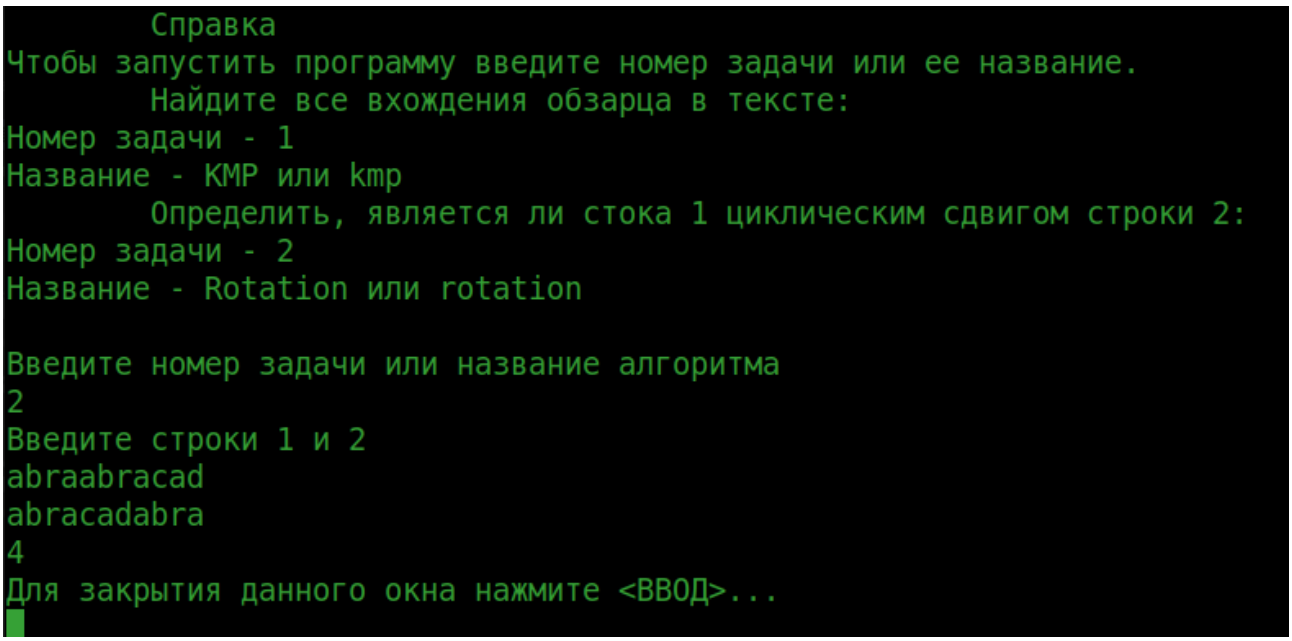
**Тестирование**

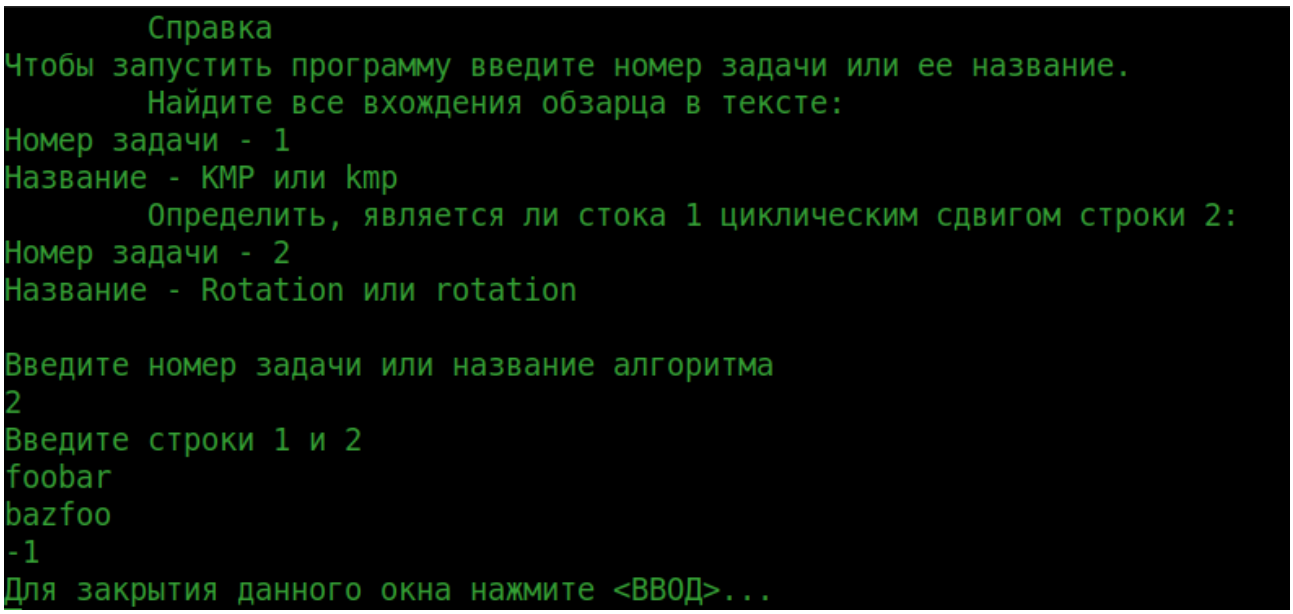
**KMP**

```
        Справка
Чтобы запустить программу введите номер задачи или ее название.
        Найдите все вхождения обзарца в тексте:
Номер задачи - 1
Название - KMP или kmp
        Определить, является ли стока 1 циклическим сдвигом строки 2:
Номер задачи - 2
Название - Rotation или rotation

Введите номер задачи или название алгоритма
1
Введите текст
abababababababbbababababaabba
Введите образец (искомую подстроку)
ab
-------------------------------------------
Строка будет разделена на 8 частей
Максимальная длинна части исходного текста - 4
-------------------------------------------

----------------------------
Текущейя часть текста
abab
Номер символа начала образца в данной части исходного текста
0 2
Номер символа начала образца в исходном тексте
0 2
----------------------------
Текущейя часть текста
abab
Номер символа начала образца в данной части исходного текста
0 2
Номер символа начала образца в исходном тексте
4 6
----------------------------
Текущейя часть текста
abab
Номер символа начала образца в данной части исходного текста
0 2
Номер символа начала образца в исходном тексте
8 10
----------------------------
Текущейя часть текста
abab
Номер символа начала образца в данной части исходного текста
0 2
Номер символа начала образца в исходном тексте
12 14
----------------------------
Текущейя часть текста
bbab
Номер символа начала образца в данной части исходного текста
2
Номер символа начала образца в исходном тексте
18
----------------------------
Текущейя часть текста
abab
Номер символа начала образца в данной части исходного текста
0 2
Номер символа начала образца в исходном тексте
20 22
----------------------------
Текущейя часть текста
aabb
Номер символа начала образца в данной части исходного текста
1
Номер символа начала образца в исходном тексте
25
0,2,4,6,8,10,12,14,18,20,22,25
Для закрытия данного окна нажмите <ВВОД>...
```


Справка
Чтобы запустить программу введите номер задачи или ее название.
Найдите все вхождения обзарца в тексте:
Номер задачи - 1
Название - KMP или kmp
Определить, является ли стока 1 циклическим сдвигом строки 2:
Номер задачи - 2
Название - Rotation или rotation
Введите номер задачи или название алгоритма
1
Введите текст
Dance for me, dance for me, dance for me, oh, oh, oh I've never seen anybody do the things you do before They say move for me, move for me, move for me, ay, ay, ay And when you're done I'll make you do it all again
Введите образец (искомую подстроку)
for
Строка будет разделена на 8 частей
Максимальная длинна части исходного текста - 27
Текущейя часть текста
Dance for me, dance for me,
Номер символа начала образца в данной части исходного текста
6 20
Номер символа начала образца в исходном тексте
6 20
Текущейя часть текста
dance for me, oh, oh, oh I
Номер символа начала образца в данной части исходного текста
7
Номер символа начала образца в исходном тексте
34
Текущейя часть текста
ne things you do before The
Номер символа начала образца в данной части исходного текста
19
Номер символа начала образца в исходном тексте
100
Текущейя часть текста
y say move for me, move for
Номер символа начала образца в данной части исходного текста
11 24
Номер символа начала образца в исходном тексте
119 132
Текущейя часть текста
me, move for me, ay, ay, a
Номер символа начала образца в данной части исходного текста
18
Номер символа начала образца в исходном тексте
145
6,20,34,100,119,132,145
Для закрытия данного окна нажмите <ВВОД>...

# Rotation


Справка
Чтобы запустить программу введите номер задачи или ее название.
Найдите все вхождения обзарца в тексте:
Номер задачи - 1
Название - KMP или kmp
Определить, является ли стока 1 циклическим сдвигом строки 2:
Номер задачи - 2
Название - Rotation или rotation
Введите номер задачи или название алгоритма
2
Введите строки 1 и 2
abraabracad
abracadabra
4
Для закрытия данного окна нажмите <ВВОД>...




**Вывод**

В ходе выполнения лабораторной работы был изучен  и реализован алгоритм Кнута-Морриса-Пратта для поиска подстроки в строке, результатом которого является набор индексов вхождения подстроки. Для работы алгоритма также реализована префикс-функция . Помимо основного алгоритма, так же реализован механизм распараллеливания строки, для запуска алгоритма сразу в нескольких местах.

# ПРИЛОЖЕНИЕ А.

# ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <vector>
#include <string>
#include <thread>
#include <algorithm>

using namespace std;

vector<int> prefix_function (string s) {
    int n = (int) s.length();
    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;
}

vector<int> KMP(string t, string p, vector<int> &pi){
    vector<int> ans;
    int n = t.length();
    int m = p.length();
    if(n == m){
        if(t == p){
            ans.push_back(0);
            return ans;
        }
        else {
            return ans;
        }
    }
    int k = 0, l = 0;
    while(k < n){
        if(t[k] == p[l]){
            k++; l++;
            if(l == m){ans.push_back(k-l);}
        }
        else {
            if(l == 0){
                k++;
            }
            else {
                l = pi[l-1];
            }
        }
    }
    return ans;
}

void split(string t, string p, int k, vector<string> &str, vector<int>
&ans_current, vector<int> &ans, vector<int> &pi){
    int len_parts, flag = 0;
    //-----------------------------------------------------
    //определяем длинну каждой части
    if(t.length() % k){
        len_parts = int(t.length()/k)+1; //длинна части строки
        flag = 1;
    }
    else {
        len_parts = t.length()/k;
    }
    //-----------------------------------------------------
```

```
    int k1 = k - 1;
    int begin = 0;
    string part = "";
    //цикл для получения массива подстрок из текста
    while(k1 > 0){
        part = "";
        part.append(t, begin, len_parts);
        str.push_back(part);
        begin += len_parts;
        k1--;
    }
    if(flag){
        part = "";
        part.append(t, begin, (t.length()-(len_parts*(k-1))));
        str.push_back(part);
    }

    //цикл для получения и проверки подстрок на стыках на каждом стыке
проверяется 2 стрки
    k1 = 1;
    while(k1 < k){
        part = "";
        part.append(t, (len_parts*k1)-1, p.length());
        ans_current = KMP(part,p,pi);
        if(ans_current.size() > 0){
            cout << "----------------------------------" << endl;
            cout << "{стык лево} Текущейя часть текста" << endl;
            cout << part << endl;
            cout << "{стык лево} Номер символа начала образца в данной части
исходного текста" << endl;
            cout << ans_current[0] << endl;
            cout << "{стык лево}Номер символа начала образца в исходном тексте"
<< endl;
            ans_current[0] += (len_parts*k1-1); //определяем номер символа
начала подстроки в исходном тексте
            cout << ans_current[0] << endl;
            ans.insert(ans.end(), ans_current.begin(), ans_current.end());
        }
        part = "";
        part.append(t, (len_parts*k1)-p.length()+1, p.length());
        ans_current = KMP(part,p,pi);
        if(ans_current.size() > 0){
            cout << "----------------------------------" << endl;
            cout << "{стык право} Текущейя часть текста" << endl;
            cout << part << endl;
            cout << "{стык право} Номер символа начала образца в данной части
исходного текста" << endl;
            cout << ans_current[0] << endl;
            cout << "{стык право}Номер символа начала образца в исходном тексте"
<< endl;
            ans_current[0] += (len_parts*k1-p.length()+1); //определяем номер
символа начала подстроки в исходном тексте
            cout << ans_current[0] << endl;
            ans.insert(ans.end(), ans_current.begin(), ans_current.end());
        }
        k1++;
    }
}

int main()
{
    cout << "\tСправка\nЧтобы запустить программу введите номер задачи или ее
название.\n"
            "\tНайдите все вхождения обзарца в тексте:\nНомер задачи - 1\
nНазвание - KMP или kmp\n"
            "\tОпределить, является ли стока 1 циклическим сдвигом строки 2:\
nНомер задачи - 2\nНазвание - Rotation или rotation"<< endl;
    cout << endl;
    string task;
    cout << "Введите номер задачи или название алгоритма" << endl;
    getline(cin, task);
    if(task == "KMP" or task == "kmp" or task == "1"){
```

```
        string p,t;
        cout << "Введите текст" << endl;
        getline(cin, t);
        cout << "Введите образец (искомую подстроку)" << endl;
        getline(cin, p);
        int max_threads = sizeof(thread); // определяем максимально возможное
число потоков
        //-----------------------------------------------
        // определяем на сколько частей можно раделить строку
        double alpha = (double)t.length()/(double)p.length();
        max_threads = min(max_threads, int(alpha)-1);
        if(max_threads == 0)
            max_threads = 1;
        int k = max_threads;
        //-----------------------------------------------
        vector<int> pi = prefix_function(p);
        vector<int> ans, ans_current;
        vector<string> str;
        cout << "-----------------------------------------" << endl;
        cout << "Строка будет разделена на " << k << " частей" << endl;
        if(k == 1)
            ans = KMP(t, p, pi);
        else {
            //-----------------------------------------------
            // определяем длинну каждой части
            int len_parts;
            if(t.length() % k){
                len_parts = int(t.length()/k)+1; //длинна части строки
            }
            else {
                len_parts = t.length()/k;
            }
            cout << "Максимальная длинна части исходного текста - " << len_parts
<< endl;
            cout << "-----------------------------------------" << endl;
            cout << endl;
            split(t, p, k, str, ans_current, ans, pi);
            //-----------------------------------------------
            //заполняем исходный массив ответов
            for(int i = 0; i< str.size(); i++){
                ans_current = KMP(str[i], p, pi);
                if(ans_current.size() > 0){
                    cout << "----------------------------" << endl;
                    cout << "Текущейя часть текста" << endl;
                    cout << str[i] << endl;
                    cout << "Номер символа начала образца в данной части
исходного текста" << endl;
                    for(int j = 0; j < ans_current.size(); j++)
                        cout << ans_current[j] << ' ';
                    cout << endl;
                    for(int j = 0; j < ans_current.size(); j++)
                        ans_current[j] += (len_parts*i); // определяем номер
символа начала образца в исходном тексте
                    cout << "Номер символа начала образца в исходном тексте" <<
endl;
                    for(int j = 0; j < ans_current.size(); j++)
                        cout << ans_current[j] << ' ';
                    cout << endl;
                    ans.insert(ans.end(), ans_current.begin(),
ans_current.end());
                }
            }
        }
        // Ввывод ответа
        if(ans.size() == 0)
            cout << "-1";
        else {
            sort(ans.begin(), ans.end()); //сортируем массив ответов для
читабельности
            for(int i = 0; i < ans.size(); i++){
                if(i == ans.size()-1)
                    cout << ans[i];
```

```
                else {
                    cout << ans[i] << ',';
                }
            }
            cout << endl;
        }
    }
    else{
        if(task == "Rotation" or task == "rotation" or task == "2"){
            string a,b;
            cout << "Введите строки 1 и 2" << endl;
            cin >> a >> b;
            if(b.length() != a.length())
            {
                cout << "-1" << endl;
                return 0;
            }
            if(a == b){
                cout << 0 << endl;
                return 0;
            }
            int it_a = 0, it_b = 0;
            int cikle = 0;
            int al = a.length();
            while(true){
                if(a[it_a] == b[it_b]){
                    it_a++;
                    it_b++;
                }
                if(it_a == al && it_b != al){
                    it_a = 0;
                    cikle++;
                }
                if(it_b == al){
                    cout << it_a << endl;
                    return 0;
                }
                if(a[it_a] != b[it_b]){
                    if(it_b == 0)
                        it_a++;
                    else {
                        it_b = 0;
                    }
                }
                if(cikle >1){
                    cout << -1 << endl;
                    return 0;
                }
            }
        }
    }
    return 0;
}
```