

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ЭЛЕКТРОТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА «ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И
АВТОМАТИЗИРОВАННЫХ СИСТЕМ»

ОТЧЁТ

**«ЛАБОРАТОРНАЯ №13:
АЛГОРИТМ БОЙЕРА — МУРА»**
Дисциплина: «Программирование»

Выполнил:

Студент группы ИВТ-21-26

Безух Владимир Сергеевич

Проверил:

Доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь, 2022

Содержание

1.	Постановка задачи	3
2.	Анализ задачи.....	4
3.	Исходный код.....	6
4.	Анализ результатов.....	9

1. Постановка задачи

Найти образец строки в тексте с помощью алгоритма Бойера — Мура.

2. Анализ задачи

Алгоритм Бойера — Мура основан на трёх идеях.

1. Сканирование слева направо, сравнение справа налево. Совмещаются начала текста и образца строки, проверка начинается с последнего символа образца строки. Если символы совпадают, производится сравнение предпоследнего символа образца строки и т. д. Если все символы образца строки совпали с наложенными символами текста, значит, вхождение образца строки в текст найдено, и выполняется поиск следующего вхождения образца строки.

Если же какой-то символ образца строки не совпадает с соответствующим символом текста, образец строки сдвигается на несколько символов вправо, и проверка снова начинается с последнего символа.

2. Эвристика стоп-символа. Предположим, что мы производим поиск слова «колокол». Первая же буква не совпала — «к» (назовём эту букву стоп-символом). Тогда можно сдвинуть образец строки вправо до последней его буквы «к».

```
Текст:          * * * * * к * * * * *
Образец строки:  к о л о к о л
Следующий шаг:   к о л о к о л
```

Если стоп-символа в образце строки нет, образец строки смещается за этот стоп-символ.

```
Текст:          * * * * * а л * * * * *
Образец строки:  к о л о к о л
Следующий шаг:   к о л о к о л
```

Если стоп-символ «к» оказался за другой буквой «к», эвристика стоп-символа не работает.

```
Текст :      * * * * к к о л * * * * *
Образец строки:  к о л о к о л
Следующий шаг:  к о л о к о л
```

В таких ситуациях применяется третья идея алгоритма Бойера — Мура.

3. Эвристика совпавшего суффикса. Если при чтении образца строки справа налево совпал суффикс S , а символ b , стоящий перед S в образце строки (то есть образец строки имеет вид PbS), не совпал, то эвристика совпавшего суффикса сдвигает образец строки на наименьшее число позиций вправо так, чтобы текст S совпал с образцом строки, а символ, предшествующий в образце строки данному совпадению S , отличался бы от b (если такой символ вообще есть). Например:

```
Текст:      * * * * * * р к а * * * * *
Образец строки: с к а л к а л к а
Следующий шаг:      с к а л к а л к а
```

В данном случае совпал суффикс «ка», и образец строки сдвигается вправо до ближайшего «ка», перед которым нет буквы «л».

```
Текст:      * * т о к о л * * * * *
Образец строки:  к о л о к о л
Следующий шаг:      к о л о к о л
```

В данном случае совпал суффикс «окол», и образец строки сдвигается вправо до ближайшего «окол», перед которым нет буквы «л». Если подстроки «окол» в образце строки больше нет, но он начинается на «кол», сдвигается до «кол», и т. д.

3. Исходный код

```
#include <vector>
#include <string>
#include <iostream>
#include <unordered_map>

// good suffix heuristic
std::vector<int> prefix_func(const std::string& text)
{
    std::vector<int> prefix(text.length());

    int k = 0; prefix[0] = 0;
    for (int i = 1; i != text.length(); ++i)
    {
        while (k > 0 && text[k] != text[i])
            k = prefix[k - 1];

        if (text[k] == text[i])
            ++k;

        prefix[i] = k;
    }

    return prefix;
}
```

```

std::vector<int> find(std::string& text, std::string& pattern)
{
    if (text.length() < pattern.length())
        return std::vector<int>(1, -1);

    if (!pattern.length())
        return std::vector<int>(1, text.length());

    typedef std::unordered_map<char, int> TStopTable;
    typedef std::unordered_map<int, int> TSufficsTable;

    TStopTable stop_table;
    TSufficsTable suffics_table;

    // bad character heuristic
    for (int i = 0; i != static_cast<int>(pattern.length()); ++i)
        stop_table[pattern[i]] = i;

    std::string reverse_pattern(pattern.rbegin(), pattern.rend());
    std::vector<int> prefix = prefix_func(pattern), reverse_prefix =
prefix_func(reverse_pattern);
    for (int i = 0; i != pattern.length() + 1; ++i)
        suffics_table[i] = pattern.length() - prefix.back();

    for (int i = 1; i != pattern.length(); ++i) {
        int j = reverse_prefix[i];
        suffics_table[j] = std::min(suffics_table[j], i - reverse_prefix[i] + 1);
    }

    std::vector<int> shifts;
    for (int shift = 0; shift <= static_cast<int>(text.length()) -
static_cast<int>(pattern.length());)
    {
        int position = pattern.length() - 1;

        while (pattern[position] == text[position + shift]) {
            if (position == 0) { shifts.emplace_back(shift); break; }
            --position;
        }

        if (position == pattern.length() - 1) {
            TStopTable::const_iterator stop_symbol = stop_table.find(text[position
+ shift]);
            int stop_symbol_additional = position - (stop_symbol !=
stop_table.end() ? stop_symbol->second : -1);
            shift += stop_symbol_additional;
        } else shift += suffics_table[pattern.length() - 1 - position];
    }

    if (!shifts.empty())
        return shifts;

    return std::vector<int>(1, -1);
}

```

```

int main()
{
    std::string text = "TENETENET ABCTENCD TEN ET AENET TENTEN";
    std::string first_pattern = "TEN", second_pattern = "TENET";

    std::cout << "Text: " << text << std::endl;

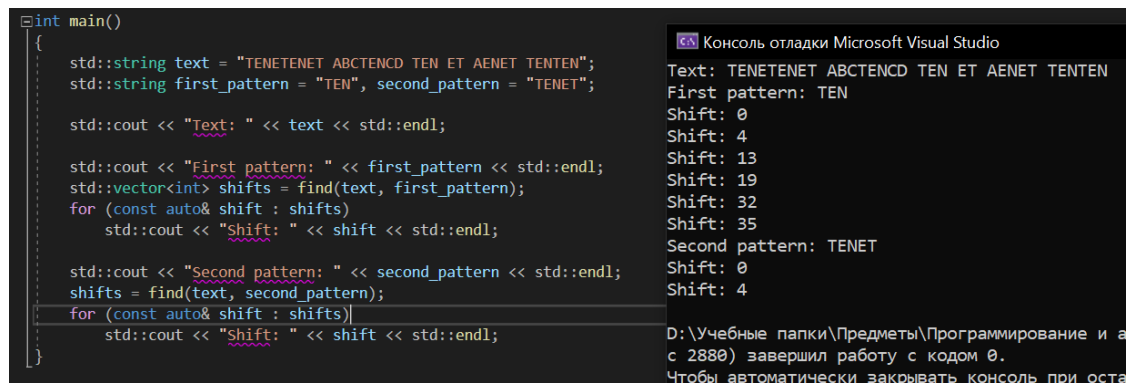
    std::cout << "First pattern: " << first_pattern << std::endl;
    std::vector<int> shifts = find(text, first_pattern);
    for (const auto& shift : shifts)
        std::cout << "Shift: " << shift << std::endl;

    std::cout << "Second pattern: " << second_pattern << std::endl;
    shifts = find(text, second_pattern);
    for (const auto& shift : shifts)
        std::cout << "Shift: " << shift << std::endl;
}

```


4. Анализ результатов

Результаты работы программы (рис. 1).



```
int main()
{
    std::string text = "TENETENET ABCTENCND TEN ET AENET TENTEN";
    std::string first_pattern = "TEN", second_pattern = "TENET";

    std::cout << "Text: " << text << std::endl;

    std::cout << "First pattern: " << first_pattern << std::endl;
    std::vector<int> shifts = find(text, first_pattern);
    for (const auto& shift : shifts)
        std::cout << "Shift: " << shift << std::endl;

    std::cout << "Second pattern: " << second_pattern << std::endl;
    shifts = find(text, second_pattern);
    for (const auto& shift : shifts)
        std::cout << "Shift: " << shift << std::endl;
}
```

Консоль отладки Microsoft Visual Studio

Text: TENETENET ABCTENCND TEN ET AENET TENTEN
First pattern: TEN
Shift: 0
Shift: 4
Shift: 13
Shift: 19
Shift: 32
Shift: 35
Second pattern: TENET
Shift: 0
Shift: 4

D:\Учебные папки\Предметы\Программирование и а с 2880) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при оста

Рисунок 1 — Результаты