Министерство цифрового развития, связи и массовых коммуникаций

Российской Федерации Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский технический университет связи и информатики»

Кафедра информатики

Отчёт по лабораторной работе №3

на тему «Методы поиска подстроки в строке»

по дисциплине «Структуры и алгоритмы обработки данных»

Выполнил: студент группы БВТ1903

Щитов В.М.

Руководитель: Павликов А.Е.

Содержание

[1. Задание 3](#_Toc72401080)

[2. Ход работы 4](#_Toc72401081)

[3. Вывод 9](#_Toc72401082)

# 1. Задание

Реализовать методы поиска подстроки в строке. Добавить возможность ввода строки и подстроки с клавиатуры. Предусмотреть возможность существования пробела. Реализовать возможность выбора опции чувствительности или нечувствительности к регистру. Оценить время работы каждого алгоритма поиска и сравнить его со временем работы стандартной функции поиска, используемой в выбранном языке программирования. Реализуемые алгоритмы:

1. Кнута-Морриса-Пратта.

2. Упрощенный Бойера-Мура.

# 2. Ход работы

Язык программирования, используемый для выполнения работы: C++ (используется стандарт C++14). Для выполнения поставленных задач было создано решение в среде разработки MVS2015, включающее проект «structures-and-algos», исполняемый код которого представлен в файлах Lab3.h и Lab3Strings.h, листинг которых представлен ниже.

Листинг файла Lab3Strings.h:

#ifndef LAB3STRINGS\_H

#define LAB3STRINGS\_H

#include <iostream>

#include <string>

#include <vector>

using namespace std;

// Пространство имён 3 лабораторной работы

namespace lab3 {

// Объявления функций внутри пространства имён

void KMP\_Wrapper();

int KMP\_First(string str, string substr);

int KMP\_Last(string str, string substr);

vector<int> KMP(string str, string substr);

void FindPrefixes(string str, int\* prefixes);

char GetUniqueSym(string str, string substr);

void BM\_Wrapper();

int BM\_First(string str, string substr);

int BM\_Last(string str, string substr);

vector<int> BM(string str, string substr);

// Функция вычисления префикса строки

void FindPrefixes(string str, int\* prefixes) {

int index = 0;

prefixes[0] = 0;

for (size\_t i = 1; i < str.length(); i++) {

while (index >= 0 && str[index] != str[i]) index--;

index++;

prefixes[i] = index;

}

}

// Функция получения уникального символа для 2 строк

char GetUniqueSym(string str, string substr) {

// Находим уникальный символ

char sym = 0;

for (int i = 0; i < 256; i++) {

if (str.find(sym + i) == string::npos && substr.find(sym + i) == string::npos) return sym;

}

// В противном случае возвращаем 0

cout << "Уникальный символ не найден, возможно получение неправильных результатов!\n";

return 0;

}

// Функция для вычисления алгоритма Кнута-Морриса-Пратта

vector<int> KMP(string str, string substr) {

// Получаем префиксы искомой подстроки

string input = substr + GetUniqueSym(str, substr) + str;

size\_t size\_pr = input.length();

int\* prefixes = new int[size\_pr];

FindPrefixes(substr, prefixes);

// Через основной цикл находим вхождения в строку

vector<int> indexes;

for (int k = 0, i = 0; i < (int)str.length(); ++i) {

while ((k > 0) && (substr[k] != str[i])) k = prefixes[k - 1];

k++;

// Добавляем их в вектор

if (k == substr.length()) indexes.push\_back(i - substr.length() + 1);

}

// Если подстрока не найдена

if (indexes.size() == 0) indexes.push\_back(string::npos);

return indexes;

}

// Функция для получения первого вхождения подстроки в строку

int KMP\_First(string str, string substr) {

vector<int> indexes = KMP(str, substr);

return indexes[0];

}

// Функция для получения последнего вхождения подстроки в строку

int KMP\_Last(string str, string substr) {

vector<int> indexes = KMP(str, substr);

return indexes[indexes.size() - 1];

}

// Обёртка для алгоритма поиска КМП

void KMP\_Wrapper() {

string str, substr;

size\_t var = 0;

cout << "Введите строку: ";

cin >> str;

cout << "Введите подстроку, искомую в строке: ";

cin >> substr;

cout << "Выберите действие:\n1. Вывести индекс первого вхождения подстроки в строку.\n";

cout << "2. Вывести индекс последнего вхождения подстроки в строку.\n3. Вывести индексы всех вхождений подстроки в строку.\n";

cin >> var;

if (var == 1) cout << KMP\_First(str, substr) << endl;

else if (var == 2) cout << KMP\_Last(str, substr) << endl;

else if (var == 3) {

vector<int> result = KMP(str, substr);

for (size\_t i = 0; i < result.size(); i++) cout << result[i] << " ";

cout << endl;

}

else cout << "Введено некорректное значение, выход из функции." << endl;

}

// Функция для вычисления алгоритма Бойера-Мура

vector<int> BM(string str, string substr) {

// Формируем массив значений

unsigned int alpha[256];

unsigned int sub\_length = substr.length();

for (size\_t i = 0; i < 256; i++) alpha[i] = sub\_length;

for (size\_t i = 0; i < sub\_length; i++) {

if (alpha[(int)substr[sub\_length - 1 - i]] == substr.length()) {

alpha[(int)substr[sub\_length - 1 - i]] = i;

}

}

// Проходимся по строке

vector<int> indexes;

int index = sub\_length - 1;

while (index < (int)str.length()) {

int temp\_index = index;

size\_t counter = 0;

for (int i = sub\_length - 1; i >= 0; i--) {

if (alpha[(int)str[index]] == alpha[(int)substr[i]]) {

counter += 1;

index -= 1;

}

else {

index = temp\_index + alpha[index];

break;

}

}

// Если нашли, добавляем в вектор

if (counter == sub\_length) {

indexes.push\_back(index + 1);

index = index + sub\_length + 1;

}

}

// Если подстрока не найдена

if (indexes.size() == 0) indexes.push\_back(string::npos);

return indexes;

}

// Функция для получения первого вхождения подстроки в строку

int BM\_First(string str, string substr) {

vector<int> indexes = BM(str, substr);

return indexes[0];

}

// Функция для получения последнего вхождения подстроки в строку

int BM\_Last(string str, string substr) {

vector<int> indexes = BM(str, substr);

return indexes[indexes.size() - 1];

}

// Обёртка для алгоритма поиска БМ

void BM\_Wrapper() {

string str, substr;

size\_t var = 0;

cout << "Введите строку: ";

cin >> str;

cout << "Введите подстроку, искомую в строке: ";

cin >> substr;

cout << "Выберите действие:\n1. Вывести индекс первого вхождения подстроки в строку.\n";

cout << "2. Вывести индекс последнего вхождения подстроки в строку.\n3. Вывести индексы всех вхождений подстроки в строку.\n";

cin >> var;

if (var == 1) cout << BM\_First(str, substr) << endl;

else if (var == 2) cout << BM\_Last(str, substr) << endl;

else if (var == 3) {

vector<int> result = BM(str, substr);

for (size\_t i = 0; i < result.size(); i++) cout << result[i] << " ";

cout << endl;

}

else cout << "Введено некорректное значение, выход из функции." << endl;

}

}

#endif LAB3STRINGS\_H

Листинг файла Lab3.h:

#ifndef LAB3\_H

#define LAB3\_H

#include <iostream>

#include <string>

#include "Lab3Strings.h"

using namespace std;

// Пространство имён 3 лабораторной работы

namespace lab3 {

// Главная функция для лабораторной работы

void Lab3Start() {

// Сначала выполняется поиск строк разными методами

cout << "Алгоритм КМП.\n";

KMP\_Wrapper();

cout << "\nАлгоритм БМ.\n";

BM\_Wrapper();

cout << endl;

}

}

#endif LAB3\_H

Результаты выполнения программы представлены на рисунках ниже.

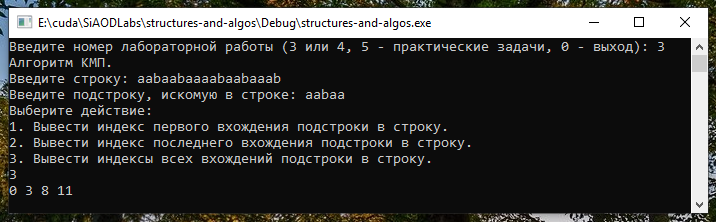


Рисунок 1 – Результат поиска алгоритмом Кнута-Морриса-Пратта

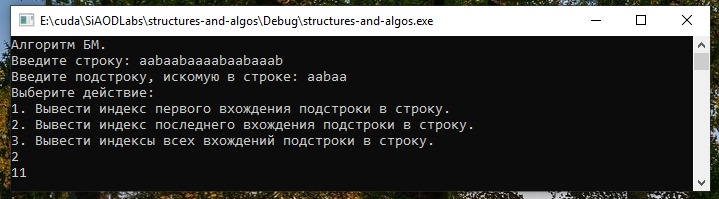


Рисунок 2 – Результат поиска алгоритмом Бойера-Мура

# 3. Вывод

В ходе данной работы были реализованы различные методы поиска подстроки в строке, время выполнения которых незначительно уступает встроенным методам поиска языка C++.