МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Построение и анализ алгоритмов»

Тема: Алгоритм Кнута-Морриса-Пратта

Студентка гр. 8383 Сырцова Е.А.

Преподаватель Фирсов М.А.

Санкт-Петербург

2020

Цель работы

Изучить алгоритм Кнута-Морриса-Пратта поиска подстроки в строке, а также реализовать данный алгоритм на языке программирования C++.

Постановка задачи.

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.

   Входные данные

   Первая строка – P.

   Вторая строка – T.

   Выходные данные

   Индексы начал вхождений P в T, разделенных запятой, если P не входит в T, то вывести −1.

   Пример входных данных

   ab

   abab

   Соответствующие выходные данные

   0, 2

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.

   Входные данные

   Первая строка – A.

   Вторая строка – B.

   Выходные данные

   Если A является циклическим сдвигом B, индекс начала строки B в A, иначе вывести −1. Если возможно несколько сдвигов вывести первый индекс.

Пример входных данных

defabc

abcdef

Соответствующие выходные данные

3

Вар. 2. Оптимизация по памяти: программа должна требовать O(m) памяти, где m - длина образца. Это возможно, если не учитывать память, в которой хранится строка поиска.

Реализация задачи

Описание алгоритма и функций

1. Был реализован алгоритм Кнута-Морриса-Пратта для поиска подстроки в строке. Программа считывает строку-образец, вхождение которой нужно найти в строке-тексте.

На первом шаге была реализована префикс-функция для подстроки `vector <int> prefix(string arr)`, которая принимает строку-образец `arr` и определяет наибольшую длину префикса, который одновременно является суффиксом для данной подстроки. Функция возвращает вектор типа int, который хранит максимальные длины суффиксов, которые одновременно являются суффиксами подстроки.

Далее была написана функция `vector <int> KMP(string form, string line)`, которая непосредственно реализует поиск подстроки в строке. В качестве аргументов функция принимает две строки: строку – образ `form` и строку `line`, в которой необходимо производить поиск. Функция работает следующим образом: пока не был достигнут конец строки, выполняется сравнение символов строки и подстроки. Если символы равны, индексы увеличиваются, и функция переходит к сравнению следующих символов. Если символы оказались не равны и индекс образа не указывает на его начало, то новый индекс образа вычисляется с использованием префикс-функции. Иначе

индекс строки увеличивается на 1. После завершения сравнения строки и образа функция возвращает массив индексов начал вхождения подстроки в строку.

Функция вычисления префикс-функции проходится по строке-образцу 1 раз, поэтому требует O(m) времени, где m – длина строки-образца. Процесс поиска вхождений строки-образца в строке-текста выполняется, пока не закончится строка-текст, т.е. требует O(n) времени, где n - длина строки-текста. Итого общая оценка **сложности по времени** алгоритма Кнутта-Морриса-Пратта составляет O(m + n).

Предложенная реализация имеет оценку **по памяти** O(m), т.к. мы вычисляем префикс функцию только для строки P. Это возможно, так как значение префикс-функции не может превысить длину образца.

2. Был реализован алгоритм, который определяет, является ли одна строка циклическим сдвигом второй строки.

На первом шаге была реализована префикс-функция для подстроки `vector <int> prefix(string arr)`, которая принимает строку `arr` и определяет наибольшую длину префикса, который одновременно является суффиксом для данной подстроки. Функция возвращает вектор типа int, который хранит максимальные длины суффиксов, которые одновременно являются суффиксами подстроки.

Далее была реализована функция `int CYCLE_KMP(string s1, string s2)`, которая проверяет на цикличность две строки `s1` и `s2`. Если длины исходных строк не равны или не удалость найти вхождение подстроки в строку, то функция возвращает значение –1. Если строки, переданные данной функции, равны, то функция возвращает значение равное 0. Иначе выполняется поиск подстроки в строке, с условием, что, если был достигнут конец строки, в которой выполняется поиск, то индекс обнуляется, и поиск продолжается

дальше. В результате функция возвращает индекс начала одной строки в другой строке.

Сложность алгоритма **по времени** составляет O(m), где m - длина строк.

Предложенная реализация имеет оценку **по памяти** O(m), т.к. мы вычисляем префикс функцию только для строки P. Это возможно, так как значение префикс-функции не может превысить длину образца.

Тестирование

№	Входные данные	Выходные данные
1	abababcab abababababcab	4
	ab abab	0, 2
	abcbcacb acbacbbcac	-1
	sssssss bvsssssssssb	2,3,4
	a aaaaaa	0,1,2,3,4,5
2	defabc abcdef	3
	ababab bababa	1
	abcdef abcd	Длины строк не равны -1
	abcd abcd	Строки равны 0
	abcd bcde	Строки не циклически сдвинуты -1

Пример вывода промежуточных данных

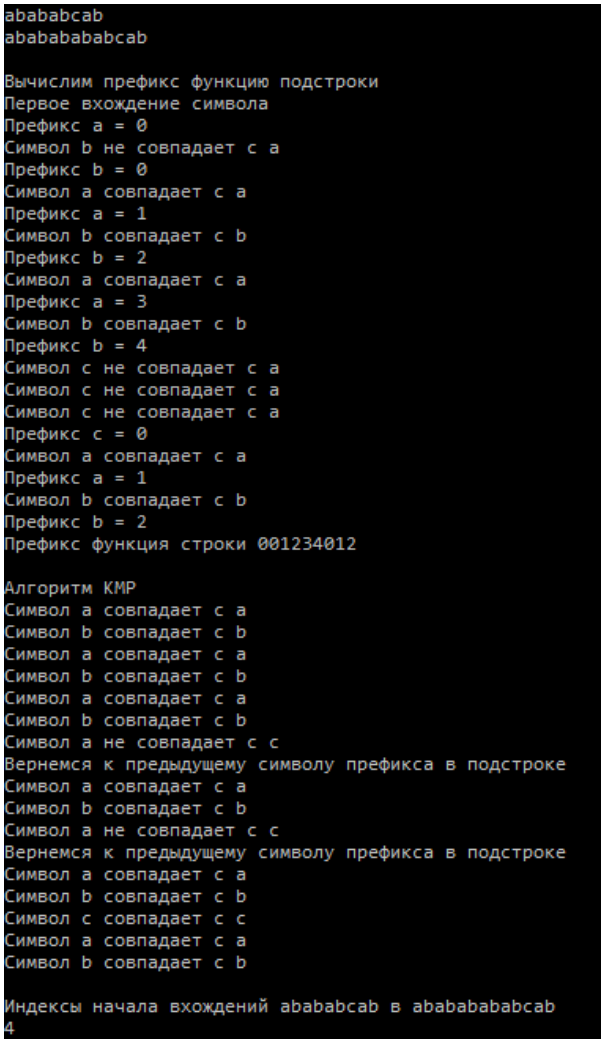

Рисунок 1 – пример вывода промежуточных результатов для поиска подстроки в строке

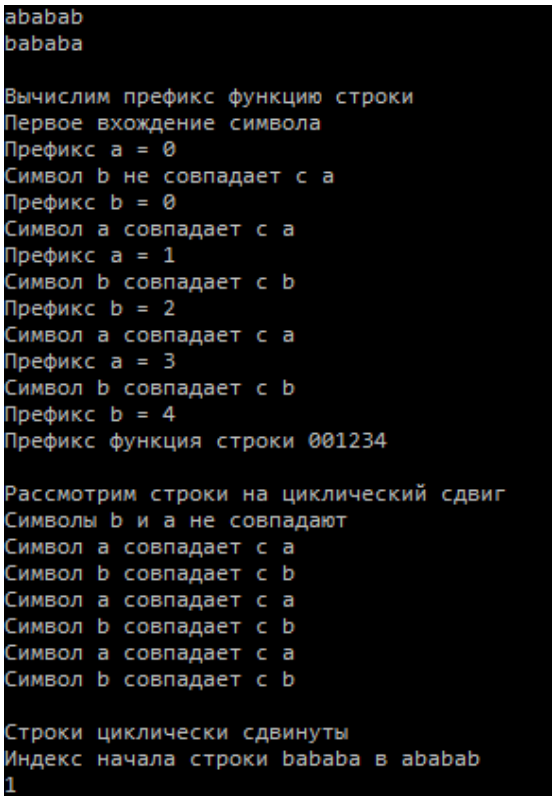

Рисунок 2 – пример вывода промежуточных результатов для поиска циклического сдвига

Вывод

В ходе выполнения лабораторной работы был изучен и реализован на языке программирования C++ алгоритм Кнута-Морриса-Пратта для поиска подстроки в строке и поиска циклического сдвига строк. Для корректной работы алгоритма было изучено понятие префикс-функции, которая определяет длину наибольшего префикса подстроки, совпадающего с ее суффиксом.

ПРИЛОЖЕНИЕ А

ПОИСК ПОДСТРОКИ В СТРОКЕ

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

using namespace std;

vector <int> prefix(string arr) {//префикс функция от подстроки
	cout << endl << "Вычислим префикс функцию подстроки" << endl;
	vector <int> p(arr.size());
	size_t j = 0;
	size_t i = 1;
	p[0] = 0;
	cout << "Первое вхождение символа\nПрефикс " << arr[0] << " = 0"<< endl;
	while (i < arr.length()) {
		if (arr[i] == arr[j]) {//если символы совпадают
			cout << "Символ " << arr[i] << " совпадает с " << arr[j] << endl;
			p[i] = j + 1;//увеличиваем префикс
			cout << "Префикс " << arr[i] << " = " << p[i] << endl;
			i++;
			j++;
		}
		else if (j == 0) {//еще не было совпадений
			p[i] = 0;
			cout << "Символ " << arr[i] << " не совпадает с " << arr[0] << endl;
			cout << "Префикс " << arr[i] << " = " << p[i] << endl;
			i++;
		}
		else {//символы не совпадают
			cout << "Символ " << arr[i] << " не совпадает с " << arr[j] << endl;
			j = p[j - 1];
		}
	}
	cout << "Префикс функция строки ";
	for (i = 0; i < arr.size(); i++) cout << p[i];
	cout << endl;
	return p;
}

vector <int> KMP(string form, string line) {
	vector <int> res;
	vector <int> p = prefix(form);//находим префикс
	size_t i = 0;
	size_t j = 0;
	cout << endl << "Алгоритм КМР" << endl;
	while (i < line.length()) {
		if (line[i] == form[j]) {//если символы равны
			cout << "Символ " << line[i] << " совпадает с " << form[j] << endl;
			i++;
			j++;
			if (j == form.length()) {
				res.push_back(i - form.length());
			}
		}
		else if (j != 0) {
			if (j!=form.length()) cout << "Символ " << line[i] << " не совпадает
с " << form[j] << endl;
			j = p[j - 1];
			cout << "Вернемся к предыдущему символу префикса в подстроке" <<
endl;
		}
		else {
```

```
                        cout << "Символы не совпадают" << endl;
                        i++;
                }
        }
        return res;
}

int main() {
        setlocale(LC_ALL, "Russian");
        string form;//подстрока
        string line;//строка
        getline(cin, form);
        getline(cin, line);
        vector <int> res = KMP(form, line);
        if (res.size() == 0) {
                cout << "\nПодстрока " << form << " не входит в строку " << line << endl;
                cout << "-1";
        }
        else {
                cout << "\nИндексы начала вхождений " << form << " в " << line << endl;
                for (size_t i = 0; i < res.size(); i++) {
                        cout << res[i];
                        if (i != res.size() - 1) {
                                cout << ",";
                        }
                }
        }
        return 0;
}
```

ПРИЛОЖЕНИЕ В

ЦИКЛИЧЕСКИЙ СДВИГ СТРОК

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

using namespace std;

vector <int> prefix(string arr) {//префикс функция
	cout << endl << "Вычислим префикс функцию строки" << endl;
	vector <int> p(arr.size());
	size_t j = 0;
	size_t i = 1;
	p[0] = 0;
	cout << "Первое вхождение символа\nПрефикс " << arr[0] << " = 0" << endl;
	while (i < arr.length()) {
		if (arr[i] == arr[j]) {//если символы совпадают
			cout << "Символ " << arr[i] << " совпадает с " << arr[j] << endl;
			p[i] = j + 1;//увеличиваем префикс
			cout << "Префикс " << arr[i] << " = " << p[i] << endl;
			i++;
			j++;
		}
		else if (j == 0) {//еще не было совпадений
			p[i] = 0;
			cout << "Символ " << arr[i] << " не совпадает с " << arr[0] << endl;
			cout << "Префикс " << arr[i] << " = " << p[i] << endl;
			i++;
		}
		else {//символы не совпадают
			cout << "Символ " << arr[i] << " не совпадает с " << arr[j] << endl;
			j = p[j - 1];
		}
	}
	cout << "Префикс функция строки ";
	for (i = 0; i < arr.size(); i++) cout << p[i];
	cout << endl;
	return p;
}

int CYCLE_KMP(string s1, string s2) {
	size_t i = 0;
	size_t j = 0;
	if (s1.length() != s2.length()) {//длины строк не равны
		cout << "\nДлины строк не равны" << endl;
		return -1;
	}
	else if (s1 == s2) {//строки равны
		cout << "\nСтроки равны" << endl;
		return 0;
	}
	else {//если нет то алгоритм кмп
		vector <int> p = prefix(s2);
		cout << "\nРассмотрим строки на циклический сдвиг" << endl;
		while (true) {
			if (s1[i] == s2[j]) {//символы совпадают
				cout << "Символ " << s1[i] << " совпадает с " << s2[j] <<
endl;
				i++;
				if (i == s1.length()) {
					i -= s1.length();
				}
```

```
                        j++;
                        if (j == s2.length()) {
                                cout << "\nСтроки циклически сдвинуты" << endl;
                                cout << "Индекс начала строки " << s1 << " в " << s2 <<
endl;
                                return i;
                                //return s2.length() - i;
                        }
                }
                else if (j != 0) {//не совпадают, откатываем назад по префиксу
                        if (j != s2.length()) cout << "Символ " << s1[i] << " не
совпадает с " << s2[j] << endl;
                        j = p[j - 1];
                        cout << "Вернемся к предыдущему символу префикса строки" <<
endl;
                }
                else {//не равны
                        cout << "Символы " << s1[i] << " и " << s2[j] << " не
совпадают" << endl;
                        i++;//идем вперед
                        if (i == s1.length()) {//если строка закончилась
                                cout << "\nСтроки не циклически сдвинуты" << endl;
                                return -1;
                        }
                }
        }
    }
}

int main() {
    setlocale(LC_ALL, "Russian");
    string a;//строка 1
    string b;//строка 2
    getline(cin, a);
    getline(cin, b);
    cout << CYCLE_KMP(b, a);
    return 0;
}
```