МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Построение и анализ алгоритмов»

Тема: Алгоритм Кнута-Морриса-Пратта

Студент гр. 8304 _____ Порывай П.А

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2020

Вариант 2

Цель работы.

Реализовать алгоритм КМП с оптимизацией по памяти: программа должна требовать O(m) памяти, где m - длина образца.

Основные теоретические положения.

Реализуйте алгоритм КМП и с его помощью для заданных шаблона P ($|P| \le 15000$) и текста T ($|T| \le 5000000$) 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

Описание алгоритма.

Оптимизация — строка-текст считывается посимвольно.

Сложность алгоритма O(|P| + |T|).

1. Считать значения префикс-функции $\pi[i]$ будем по очереди: от $i = 1$ к $i = n - 1$ (значение $\pi[0]$ просто присвоим равным нулю).
2. Для подсчёта текущего значения $\pi[i]$ мы заводим переменную j, обозначающую длину текущего рассматриваемого образца. Изначально $j = \pi[i - 1]$.
3. Тестируем образец длины j, для чего сравниваем символы $s[j]$ и $s[i]$. Если они совпадают — то полагаем $\pi[i] = j + 1$ и переходим к следующему индексу $i + 1$. Если же символы отличаются, то уменьшаем длину j, полагая её равной $\pi[j - 1]$, и повторяем этот шаг алгоритма с начала.

4. Если мы дошли до длины $j = 0$ и так и не нашли совпадения, то останавливаем процесс перебора образцов и полагаем $\pi[i] = 0$ и переходим к следующему индексу $i + 1$.

Описание функций.

vector<int> prefix_function(string s)	Функция для вычисления префикс-функции строки.
void kmp(string T, string P, vector<int>& answer)	Функция поиска всех вхождений

Вывод промежуточной информации.

Во время основной части работы алгоритма происходит вывод промежуточной информации, а именно, значения префикс функции и проверка идентичности префикса и суффикса первой и второй строки соответственно. Также была реализована запись в файл и из него.

Тестирование.

Таблица 1 – Результаты тестирование

Ввод	**Вывод**
abc abcackabcm;m;ABcabc	Ищем какой префикс можно расширить 0 длина предыдущего префикса-суффикса, возможно нулевая Ищем какой префикс можно расширить 0 длина предыдущего префикса-суффикса, возможно нулевая Префикс функция подсчитана

	Паттерн совпадает с одним из слов в тексте на позиции 0 Паттерн совпадает с одним из слов в тексте на позиции 6 Паттерн совпадает с одним из слов в тексте на позиции 16 Ответ 0,6,16
dqa vsxczmlsdmclkmdksdaqdqafdvm;fdvms; m vfsd;mf;sdvm;fvmd;fvmddddddd ddddddddddddddddddddddd ddddddddddddv;fsmfpdvo	Ищем какой префикс можно расширить 0 длина предыдущего префикса-суффикса, возможно нулевая Ищем какой префикс можно расширить 0 длина предыдущего префикса-суффикса, возможно нулевая Префикс функция подсчитана Паттерн совпадает с одним из слов в тексте на позиции 20 Ответ 20
fkti dsavm;sdvmlkvsdvadlvdmvdsa;vms d;dsv;lvdsmv;samv;smvds;mvsd;lm	Ищем какой префикс можно расширить 0 длина предыдущего префикса-суффикса, возможно нулевая Ищем какой префикс можно расширить 0 длина предыдущего префикса-суффикса, возможно нулевая

	Ищем какой префикс можно расширить 0 длина предыдущего префикса-суффикса, возможно нулевая Префикс функция подсчитана Ответ -1
Musk TeslaSpaceXRogozinMusk	Ищем какой префикс можно расширить 0 длина предыдущего префикса-суффикса, возможно нулевая Ищем какой префикс можно расширить 0 длина предыдущего префикса-суффикса, возможно нулевая Ищем какой префикс можно расширить 0 длина предыдущего префикса-суффикса, возможно нулевая Префикс функция подсчитана Паттерн совпадает с одним из слов в тексте на позиции 18 Ответ 18

Вывод.

В ходе работы был построен и анализирован алгоритм Кнута-Морриса-Пратта на основе решения задачи о вхождении шаблона в строку.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include<map>
#include<vector>
using namespace std;

#include<fstream>

ofstream fout("out.txt"); // создаём объект класса ofstream для записи и связываем его с файлом cppstudio.txt
ifstream fin("inp.txt");


vector<int> prefix_function(string s)
{
	int n = s.length();
	vector<int> pi(n); // в i-м элементе (его индекс i-1) количество совпавших символов в начале и конце для подстроки длины i.
		// p[0]=0 всегда, p[1]=1, если начинается с двух одинаковых
	for (int i = 1; i < n; ++i)
	{


		std::cout << "\nИщем какой префикс можно расширить\n";
		fout<< "\nИщем какой префикс можно расширить\n";

		int j = pi[i - 1];
		std::cout << "\n" << j << " длина предыдущего префикса-суффикса, возможно нулевая\n";
		fout<< "\n" << j << " длина предыдущего префикса-суффикса, возможно нулевая\n";

		while ((j > 0) && (s[i] != s[j])) // этот нельзя расширить,
			j = pi[j - 1];   // берем длину меньшего префикса-суффикса

		if (s[i] == s[j])
			++j;  // расширяем найденный (возможно пустой) префикс-суффикс
		pi[i] = j;
	}
	return pi;
}

void kmp(string T, string P, vector<int>& answer) {
```

```
	int n = T.length() - 1;
	int m = P.length() - 1;
	string a = "";
	for (int i = 1; i < m + 1; i++)
		a += P[i];
	//std::cout << a;
	vector<int> Pi1 = prefix_function(a);
	vector<int> Pi;
	Pi.push_back(0);
	for (int i = 0; i < Pi1.size(); i++)
		Pi.push_back(Pi1[i]);

	int q = 0;
	std::cout << "\nПрефикс функция подсчитана \n";
	fout<< "\nПрефикс функция подсчитана \n";
	for (int i = 1; i <= n; i++) {

		while (q > 0 && (P[q + 1] != T[i]))
			q = Pi[q];



		if (P[q + 1] == T[i]) {
			q += 1;
			//std::cout << "ax";
		}

		if (q == m) {
			//std::cout << i - m << " ";
			std::cout << "\nПаттерн совпадает с одним из слов в тексте
на позиции "<<i-m<<"\n";
			fout<< "\nПаттерн совпадает с одним из слов в тексте на
позиции " << i - m << "\n";
			answer.push_back(i - m);
			q = Pi[q];

		}


	}
}

int main()
{
	setlocale(LC_ALL, "Russian");

	std::cout << "\nВвод с консоли или из файла(1/2)?\n";
	int ind;
	cin >> ind;
	string P1;
	string P;
	string T2;
	string T;

	vector<int> answer;
	if (ind == 1) {
```

```
		cin >> P1;
		P = " ";
		P += P1;
		T2 = " ";
		cin >> T2;
		T = " ";
		T += T2;

	}
	else if (ind == 2) {

		fin >> P1;
		P = " ";
		P += P1;
		T2 = " ";
		fin >> T2;
		T = " ";
		T += T2;


	}

	kmp(T, P, answer);

	std::cout << "\nОтвет\n";
	fout<< "\nОтвет\n";
	if (answer.size() != 0) {
		for (int i = 0; i < answer.size(); i++)
			if (i != answer.size() - 1) {
				std::cout << answer[i] << ",";
				fout<< answer[i] << ",";
			}
			else {
				std::cout << answer[i];
				fout << answer[i];
			}
	}
	else {
		cout << "-1";
		fout << "-1";
	}


}
```