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 Кирьянов Д.И.

Преподаватель Размочаева Н.В.

Санкт-Петербург

2020

Цель работы.

Реализовать алгоритм Кнута-Морриса-Пратта, найти индексы вхождения подстроки в строку, а также разработать алгоритм проверки двух строк на циклический сдвиг.

Задание.

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.

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.

Вариант 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.

Описание алгоритма.

При сдвиге вполне можно ожидать, что префикс образца P сойдется с каким-нибудь суффиксом текста T. Длина наиболее длинного префикса, являющегося одновременно суффиксом, есть значение префикс-функции от строки P для индекса j. Пусть π[j] — значение префикс-функции от строки P для индекса j. Тогда после сдвига мы можем возобновить сравнения с места T[i + j] и P[π[j]] без потери возможного местонахождения образца.

Сложность алгоритма O(m + n), где

m – длина образца,

n – длина строки в которой мы ищем.

Описание основных структур данных и функций.

```
void prefix(const std::string& S, std::vector<int>& n);
```

- префикс-функция, находящая префикс строки S. Результат записывается в вектор. Размер вектора равен длине строки.

```
void KMP(std::istream& input);
```

- функция, находящая все вхождения подстроки P в строку T и выводящая индексы всех вхождений. Если вхождения не найдены, то выводится -1.

Тестирование.

Таблица 1 – Результат работы.

Ввод	**Вывод**
ab abab	Result: 0,2
abcabcd ksdhflsabcabcdlsdafjabclsdjfk	Result: 7
kfkf sdhgjfkfkfsjadfkfkfiuykfkfnmn	Result: 6,15,22
ddd aureowubdjdnasd	Result: -1
lt ltltltltltltltltltlsdfjtltlt	Result: 0,2,4,6,8,10,12,14,16,18,26,28

Вывод.

В ходе выполнения данной работы был реализован алгоритм Кнута-Морриса-Пратта, алгоритм проверки двух строк на циклический сдвиг, а также функция вычисления префикса строки.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД

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

std::string input_way = "C:/Users/Danielka/source/repos/paa_lr4/input";//пути до файлов
std::string output_way = "C:/Users/Danielka/source/repos/paa_lr4/output";

void prefix(const std::string& S, std::vector<int>& n) {
	n[0] = 0;
	for (unsigned long int i = 1; i < S.size(); ++i) {
		int k = n[i - 1];//изначально инициализируем предыдущим
		while (k > 0 && S[k] != S[i])//пока символы не равны
			k = n[k - 1];//возвращаемся к уже найденному значению
		if (S[k] == S[i])//если равны
			k += 1;//увеличиваем значение
		n[i] = k;
	}
}

void KMP(std::istream& input) {
	std::string P;
	input >> P;//считываем шаблон
	std::vector<int> n(P.size());//вектор для префикс функции
	prefix(P, n);//вычисляем префикс функцию
	std::cout << "Prefix: ";
	for (int j : n)
		std::cout << j << " ";
	std::cout << std::endl;
	int k = 0;
	int result = -1;//если нет совпадений, то резльтат останется -1
	char vau = '0';
	input.get(vau);
	input.get(vau);
	std::vector<int> ans;//ответ
	int i = 0;
	while (vau != '\n' && !input.eof()) {//пока не конец строки/файла
		std::cout << "Changes when i = " << i << " Start value k = " << k <<
std::endl;
		while (k > 0 && vau != P[k]) {//пока не совпадут символы
			k = n[k - 1];//сдвигаем
			std::cout << " k = " << k << std::endl;
		}
		if (vau == P[k]) {//если совпали
			k += 1;//увеличиваем значение
			std::cout << " k = " << k << std::endl;
		}
		if (k == P.size()) {//если длина совпадений=длине строки
			result = i - P.size() + 1;//значит ответ получен
			ans.push_back(result);
			std::cout << "-----------------------------------------------------"
<< std::endl;
			std::cout << "Result found with i = " << i << " k = " << k
				<< " Index = " << result << std::endl;
			std::cout << "-----------------------------------------------------"
<< std::endl;
		}
		i += 1;
		input.get(vau);
	}
```

```
	int menu = 0;
	std::cout << std::endl << "How do you want to output?" << std::endl << std::endl
		<< "Press 1 to output by console." << std::endl//выбор как вывести
		<< "Press 2 to output into file." << std::endl;
	while (menu != 1 && menu != 2) {//пока не введется нужная цифра
		std::cin >> menu;
		if (menu == 1) {//вывод на консоль
			std::cout << std::endl << "Result: ";
			if (!ans.empty())
				for (unsigned long int i = 0; i < ans.size() - 1; ++i) {
					std::cout << ans[i] << ",";
				}
			std::cout << result;
		}
		else if (menu == 2) {//вывод в файл
			std::ofstream file;
			file.open(output_way);
			if (!file.is_open()) {
				std::cout << "Can't open file!\n";
			}
			file << "Result: ";
			if (!ans.empty())
				for (unsigned long int i = 0; i < ans.size() - 1; ++i) {
					file << ans[i] << ",";
				}
			file << result;
		}
		else {//если неверно введена цифра, выводится сообщение
			std::cout << std::endl << "Wrong choice! Try again!" << std::endl;
		}//а цикл продолжается
	}
}


int main() {
	std::cout << "How do you want to input?" << std::endl << std::endl
		<< "Press 1 to input from console." << std::endl//выбор как считать
		<< "Press 2 to input from file." << std::endl;
	int menu = 0;
	while (menu != 1 && menu != 2) {//пока не введется нужная цифра
		std::cin >> menu;
		if (menu == 1) {//считывание с консоли
			KMP(std::cin);
		}
		else if (menu == 2) {//считывание из файла
			std::ifstream file;
			file.open(input_way);
			if (!file.is_open()) {
				std::cout << "Can't open file!" << std::endl;
				return 0;
			}
			KMP(file);
		}
		else {//если неверно введена цифра, выводится сообщение
			std::cout << std::endl << "Wrong choice! Try again!" << std::endl;
		}//а цикл продолжается
	}
	return 0;
}
```