**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 _____ Бочаров Ф.Д.

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 findstring ($|P| \leq 15000$) и текста inputstring ($|T| \leq 5000000$) найдите все вхождения findstring в inputstring.

Вход:

Первая строка – findstring

Вторая строка – inputstring

Выход:

Индексы начал вхождений findstring в inputstring, разделенных запятой, если findstring не входит в inputstring, 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 findstring сойдется с каким-нибудь суффиксом текста inputstring. Длина наиболее длинного префикса, являющегося одновременно суффиксом, есть значение префикс-функции от строки findstring для индекса j. Пусть $\pi$[j] — значение префикс-функции от строки findstring для индекса j. Тогда после сдвига мы можем возобновить сравнения с места inputstring[i + j] и findstring[$\pi$[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);
```

- метод, находящий 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** |
|---|---|
| ac<br>acacac | Result: 0,2,4 |
| ghf<br>wfjghfwdjghf | Result: 3,9 |
| kfkff<br>sdhggdgrgdgredgf | Result: -1 |
| vvvv<br>aureowubdjdnasd | Result: -1 |
| ltw<br>ltwltwltwsdfjtlwtlt | Result: 0,3,6,13,16 |

**Вывод.**

В ходе выполнения лабораторной 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>


void prefix(const std::string& S, std::vector<int>& n) {
	n[0] = 0;
	for (unsigned long int i = 1; i < S.size(); ++i) {
		int j = n[i - 1];
		while (j > 0 && S[j] != S[i])
			j = n[j - 1];
		if (S[j] == S[i])
			j += 1;
		n[i] = j;
	}
}

void KMP(std::istream& input, std::ostream& output) {
	std::string findstring;
	input >> findstring;
	std::vector<int> len(findstring.size());
	prefix(findstring, len);
	output << "Prefix: ";
	for (int j : len)
		output << j << " ";
	output << std::endl;
	int j = 0; //длина совпадений
	int result = -1;//выводим -1 по условию если не совпадает
	char inputstring;
	input.get(inputstring);
	input.get(inputstring);
	std::vector<int> answer;
	int i = 0;
	while (inputstring != '\n') {
		output << "Changes when i = " << i << " Start value k = " << j << std::endl;
		while (j > 0 && inputstring != findstring[j]) {//пока не совпадут символы
			j = len[j - 1];
			output << " k = " << j << std::endl;
		}
		if (inputstring == findstring[j]) {//если совпали
			j += 1;//увеличиваем значение
			output << " k = " << j << std::endl;
		}
		if (j == findstring.size()) {
			result = i - findstring.size() + 1;//значит ответ получен
			answer.push_back(result);
			output << "---" << std::endl;
			output << "Result found. Index = " << result << std::endl;
			output << "---" << std::endl;
		}
		i += 1;
		input.get(inputstring);
	}

	output << std::endl << "Result: ";
	if (!answer.empty())
		for (unsigned long int i = 0; i < answer.size() - 1; ++i) {
			output << answer[i] << ",";
		}
```

```
        output << result;
}


int main() {

        KMP(std::cin, std::cout);

        return 0;
}
```