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

Студентка гр. 8383 Ишанина Л.Н.

Преподаватель Фирсов М.А.

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.

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.

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.

Пример входных данных

aba

ababa

Пример выходных данных

0, 2

Описание алгоритма.

На вход алгоритма передается строка-образ, вхождения которой нужно найти, и строка-текст, в которой нужно найти вхождения.

Оптимизация – строка-текст считывается посимвольно, в памяти хранится текущий символ.

Алгоритм сначала вычисляет префикс-функцию строки-образа.(createPiArray(&vector, &string, string.length()))

Далее посимвольно считывается строка-текст. Переменная-счетчик изначально $l = 0$. При каждом совпадении l -го символа образа и i-го символа текста счетчик увеличивается на 1. Если l = размеру образа, значит вхождение найдено. Если очередной символ текста не совпал с l -ым символом образа, то сдвигаем образец, причем точно знаем, что первые l символов образца совпали с

символами строки и надо сравнить l +1-й символ образа (его индекс l) с i-м символом строки.

Описание main ( ) :

В функции прописан ввод строки-образа(т.е.вхождение которой программа будет искать) и посимвольное считывание строки-текста(т.е. где будет совершаться поиск), а также вызов функции для составления массива pi для заданного образа и выводы промежуточных данных на консоль.
Также для удобства во время работы алгоритма происходит вывод промежуточной информации.

Сложность алгоритма.

Сложность алгоритма по времени: O (m + n), m – длина образа, n – длина текста.

Сложность алгоритма по памяти: O (m), m – длина образа, так как программа хранит только строку-образ.

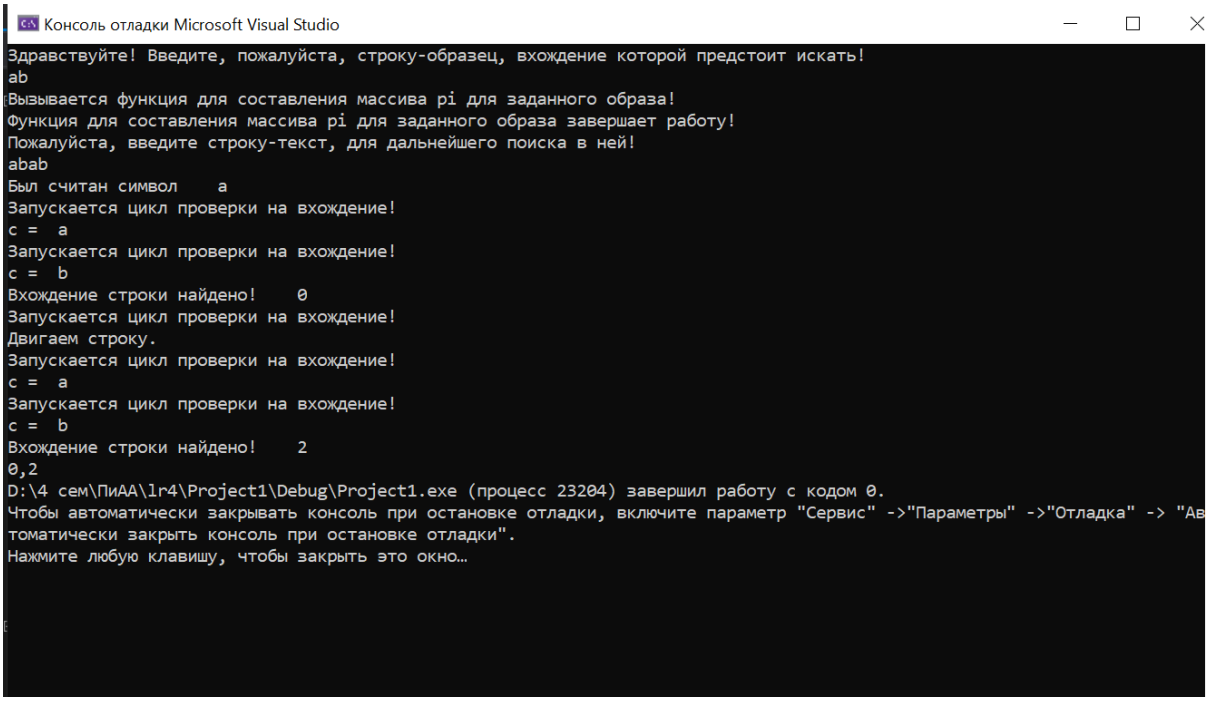
Тестирование.

Входные данные:

ab

abab

Вывод:



Тест№2

Входные данные:

time

hitiime

Вывод:

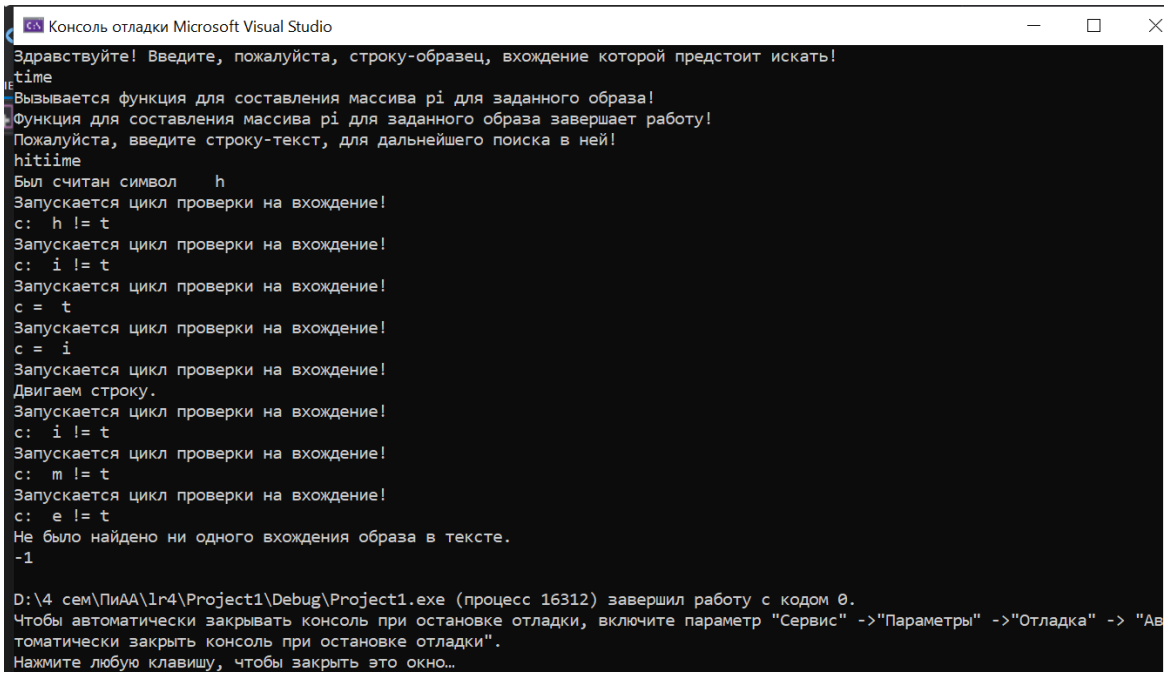


Таблица 1 – Результаты тестирования

Ввод	**Вывод**
ab abab	0,2
ababab ababab	0
work workworkwork	0,4,8
aba ababababa	0,2,4,6

Вывод.

В ходе работы был построен и анализирован алгоритм КМП. Код программы представлен в приложении А.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД

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

void createPiArray(std::vector<int>* vector, std::string* string, int length) {

    int j = 0;
    int i = 1;

    vector->emplace_back(0);//для первого символа любого образа записываем всегда 0

    while (length > i) {
        if (string->at(i) == string->at(j)) {
            vector->emplace_back(j + 1);
            i++;
            j++;
        }
        else {
            if (j == 0) {
                vector->emplace_back(0);
                i++;
            }
            else {
                j = vector->at(j - 1);
            }
        }
    }
}

int main() {
    setlocale(LC_ALL, "Russian");
    std::string string;
    std::cout << "Здравствуйте! Введите, пожалуйста, строку-образец, вхождение которой
предстоит искать!" << std::endl;
    std::cin >> string;

    std::vector<int> vector;
    vector.reserve(0);

    std::vector<int> answer;
    vector.reserve(0);
    std::cout << "Вызывается функция для составления массива pi для заданного образа!" <<
std::endl;
    createPiArray(&vector, &string, string.length());
    std::cout << "Функция для составления массива pi для заданного образа завершает работу!" <<
std::endl;
    std::cout << "Пожалуйста, введите строку-текст, для дальнейшего поиска в ней!" <<
std::endl;
    char c;
    std::cin.get(c);//первый раз, чтобы считать \n

    int l = 0;
    int n = string.size();

    int count = 0;
    std::cin.get(c);//считываем первый символ строки-текста
    std::cout << "Был считан символ    " << c << std::endl;

    while (true) {
        std::cout << "Запускается цикл проверки на вхождение!" << std::endl;
        bool isCinActive = true;//флаг, необходимый для считывания
```

```
        if (c == string[l]) {//проверка совпадает ли текущий символ текста с символом строки-
образца
            std::cout << "c =  " << c << std::endl;
            l++;
            count++;
            if (l == n) {
                std::cout << "Вхождение строки найдено!    " << count - n << std::endl;
                answer.emplace_back(count - n);
            }
        }
        else {
            if (l == 0) {
                std::cout <<"c:  "<< c << " != " << string[l] << std::endl;
                count++;
            }
            else {
                std::cout << "Двигаем строку." << std::endl;
                l = vector.at(l - 1);
                isCinActive = false;
            }
        }

        if (isCinActive) {
            std::cin.get(c);
        }

        if (c == '\n') {
            break;
        }
    }

    if (!answer.empty()) {
        for (size_t m = 0; m < answer.size(); ++m) {//вывод ответа
            std::cout << answer[m];
            if (m != answer.size() - 1)
                std::cout << ",";
        }
    }
    else {
        std::cout <<"Не было найдено ни одного вхождения образа в тексте."<< std::endl;
    }


    return 0;

}
```