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. 8382 Ефимова М.А

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($|\ | \leq 15000$) и текста 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 структуры данных.

В данной программе используются векторы.

Вектор std::vector<int> vector – данный контейнер является аналогом массива π[i], он хранит в себе максимальную длину совпадающих префикса и суффикса подстроки в образе, которая заканчивается i-м символом.

Вектор std::vector<int> answer – это вектор для записи в него ответа.

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. В функции void function(char c, std::string string, int k, int n, int count, std::vector<int>* answer, std::vector<int>* vector) происходит следующее: изначально сравнивается рассматриваемый текущий символ(char c) строки-текста и текущий *l* -й элемент строки-образа.

В случае их равенства, происходит сдвиг *l* -го элемента строки-образа и также увеличивается переменная(int count), которая олицетворяет индекс, указывающий на символ в строке-тексте. Затем после того как выявилось совпадение символов, происходит проверка на равенство *l* номера строки-образа и переменной n(n – это длина строки образа), если это верно, то значит, что вхождение найдено и происходит запись индекса начала вхождения вектор ответа(vector<int> answer).

В случае, когда текущий символ(char c) строки-текста и текущий *l* -й элемент строки-образа не равны, то происходит проверка, не находится ли сейчас в начале(в нуле) индекс, указывающий на текущий элемент строки-образа. Если это верно, увеличиваем на единицу индекс, который указывает на символ в строке-тексте. Иначе, если индекс не равен 0, то происходит перемещение позиции индекса *l* из одной позиции в другую.

Алгоритм завершает работу по окончании строки-текста.

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 функции нахождения образа в тексте, а также вывод ответа.

Кроме того, для удобства, во время работы алгоритма, происходит вывод промежуточной информации.

Описание дополнительных функций.

Функция void createPiArray(std::vector<int>* vector, std::string* string, int length) принимает на вход вектор vector, в который будет записываться максимальная длина совпадающих префикса и суффикса, строка-образ, и длина строки-образа. Данная префикс-функция для i-го символа образа сопоставляет значение, равное максимальной длине совпадающих префикса и суффикса подстроки в образе, которая заканчивается i-м символом. Именно это значение(длина) будет храниться в векторе vector, который будет использоваться в функции самого алгоритма.

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, которая считывается в самом начале.

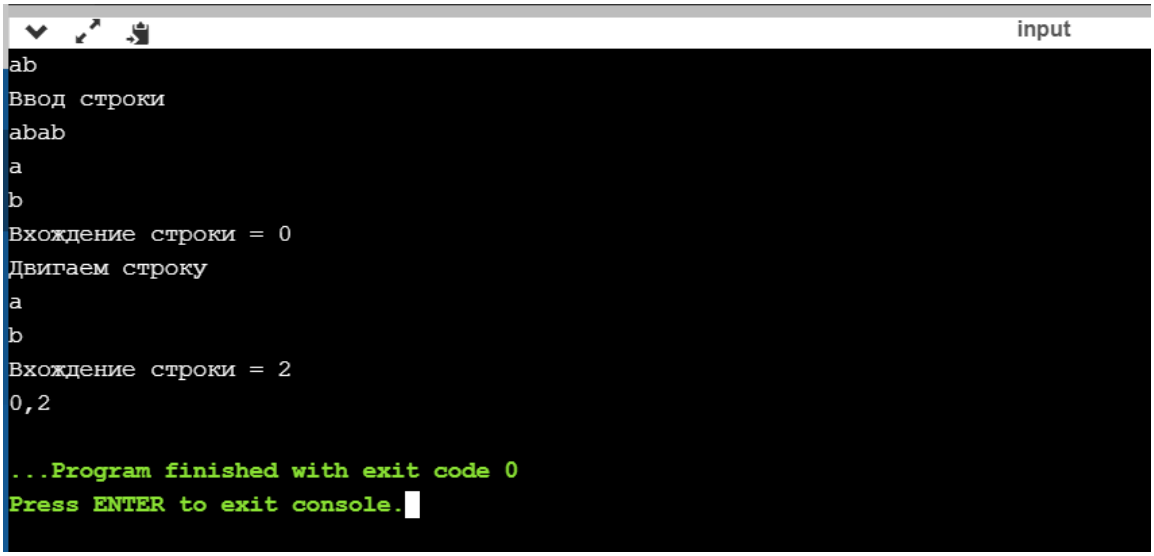
Тестирование.

Входные данные:

ab

abab

Вывод:



Тест№2

Входные данные:

leti

letiletileeeti

Вывод:

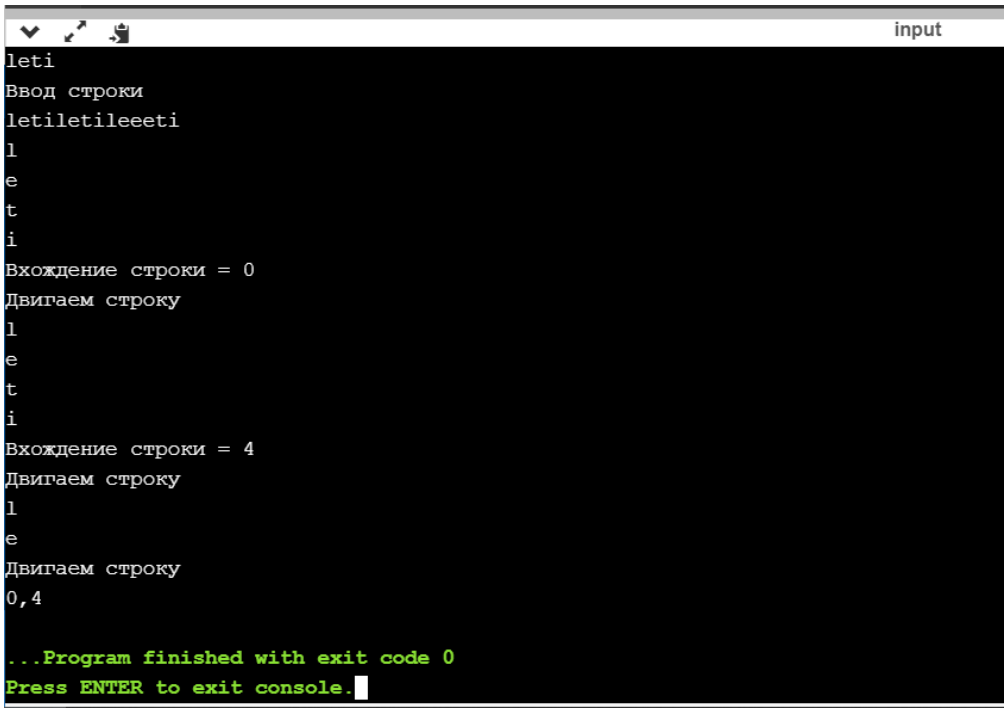


Таблица 1 – Результаты тестирования

Ввод	**Вывод**
ab abab	0,2
abababab abababab	0
work workworkwork	0,4,8
aba abababababa	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>

//вычисление префикс-функции, возвр. знач = макс.длине совпад. преф. и суфф. подстроки в образе
/*
vector<int>* vector - массив pi[i],хранит макс.
pi[0] = 0; i = 1; j = 0;
пока не законч. образ и провер. одно условие:
if(ai == aj) -> pi[i] = j + 1; i++; j++;
else if(j == 0) -> pi[i] = 0; i++ (j стоит в нач. образа)
else j = pi[j - 1] (ai != aj && j != 0)
*/
void Array_Pi(std::vector<int>* vector, std::string* string, int length) {

    int j = 0;
    int i = 1;

    vector->emplace_back(0);//первый символ всегда в ноль

    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 k = индекс,указывающий на символ в строке-тексте
int n - это длина строки образа
сравнивается текущий символ(char c) строки-текста и текущий l -й элемент строки-образа.
Если равны,то сдвиг l -го элемента строки-образа; k++.
после совпадения символов,проверка на равенство l номера строки-образа и переменной n.
если это верно, то вхождение найдено и происходит запись индекса начала вхождения вектор ответа(vector<int>
output).
*/
void function(std::vector<int>* vector, char symbol, int l, int n, std::string string,  int k, std::vector<int>* output)
{
    while (true) {
        bool input = true;//флаг для считывания

        if (symbol == string[l]) {//проверка совпадает ли текущий символ текста с символом строки-образца
            std::cout << symbol << std::endl;
            l++;
            k++;
            if (l == n) {
                std::cout << "Вхождение строки = " << k - n << std::endl;
                output->emplace_back(k - n);
            }
        }
        /*
        текущий символ строки-текста и текущий l -й элемент строки-образа не равны, то проверка,на индекс в нуле.
        Если это верно, увеличиваем на единицу индекс, который указывает на символ в строке-тексте.
        Иначе, если индекс не равен 0, то происходит перемещение позиции индекса l из одной позиции в другую.
        */
        else {
            if (l == 0) {
                k++;
            }
            else {
```

```
            std::cout << "Двигаем строку" << std::endl;
            l = vector->at(l - 1);
            input = false;
        }
    }

    if (input) {
        std::cin.get(symbol);
    }

    if (symbol == '\n') {
        break;
    }
  }
}
/*
прописан ввод строки-образа и посимвольное считывание строки-текста),
вызов функции для составления массива pi для заданного образа и функции нахождения образа в тексте и вывод ответа
*/
int main() {
  std::string string;
  std::cin >> string;

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

  Array_Pi(&vector, &string, string.length());
  std::cout << "Ввод строки" << std::endl;
  int l = 0;
  int n = string.size();
  int k = 0;
  char symbol;
  std::cin.get(symbol);
  std::cin.get(symbol);//считываем первый символ строки-текста
  function(&vector, symbol, l, n, string,  k, &output) ;

  if (!output.empty()) {
    for (size_t z = 0; z < output.size(); ++z) {//output the response
        std::cout << output[z];
        if (z != output.size() - 1)
            std::cout << ",";
    }
  }
  else {
    std::cout << -1 << std::endl;
  }

  return 0;
}
```