МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ЭЛЕКТРОТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА «ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АВТОМАТИЗИРОВАННЫХ СИСТЕМ»

**ОТЧЁТ**

**«ЛАБОРАТОРНАЯ №14:**

**АЛГОРИТМ КНУТА — МОРРИСА — ПРАТА»**

Дисциплина: «Программирование»

Выполнил:

Студент группы ИВТ-21-2б

Безух Владимир Сергеевич

Проверил:

Доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь, 2022

Содержание

[1. Постановка задачи 3](#_Toc98955824)

[2. Анализ задачи 4](#_Toc98955825)

[3. Исходный код 5](#_Toc98955826)

[4. Анализ результатов 7](#_Toc98955827)

# Постановка задачи

Найти образец строки в тексте с помощью алгоритма Кнута — Морриса — Прата.

# 

# Анализ задачи

1. Считать значения префикс-функции \pi[i] по очереди: от i=1 к i=n-1 (значение \pi[0] 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.

# Исходный код

#include <iostream>

void computeLPSArray(char\* pattern, int pattern\_length, int\* longest\_proper\_suffix)

{

int length = 0; longest\_proper\_suffix[0] = 0;

int i = 1;

while (i < pattern\_length)

{

if (pattern[i] == pattern[length])

{

length++;

longest\_proper\_suffix[i] = length;

i++;

}

else // (pattern[i] != pattern[length])

{

if (length != 0)

length = longest\_proper\_suffix[length - 1];

else // if (length == 0)

{

longest\_proper\_suffix[i] = 0;

i++;

}

}

}

}

void KMPSearch(char\* pattern, char\* text)

{

int pattern\_length = strlen(pattern);

int text\_length = strlen(text);

int\* longest\_proper\_suffix = new int[pattern\_length];

computeLPSArray(pattern, pattern\_length, longest\_proper\_suffix);

int i = 0;

int j = 0;

while (i < text\_length)

{

if (pattern[j] == text[i]) { j++; i++; }

if (j == pattern\_length)

{

std::cout << "Found pattern at index: " << i - j << '\n';

j = longest\_proper\_suffix[j - 1];

}

else if (i < text\_length && pattern[j] != text[i])

{

if (j != 0) j = longest\_proper\_suffix[j - 1];

else i = i + 1;

}

}

delete [] longest\_proper\_suffix;

}

int main()

{

char text[] = "TENETENET ABCTENCD TEN ET AENET TENTEN";

char first\_pattern[] = "TEN", second\_pattern[] = "TENET";

KMPSearch(first\_pattern, text);

KMPSearch(second\_pattern, text);

}

# Анализ результатов

Результаты работы программы (рис. 1).

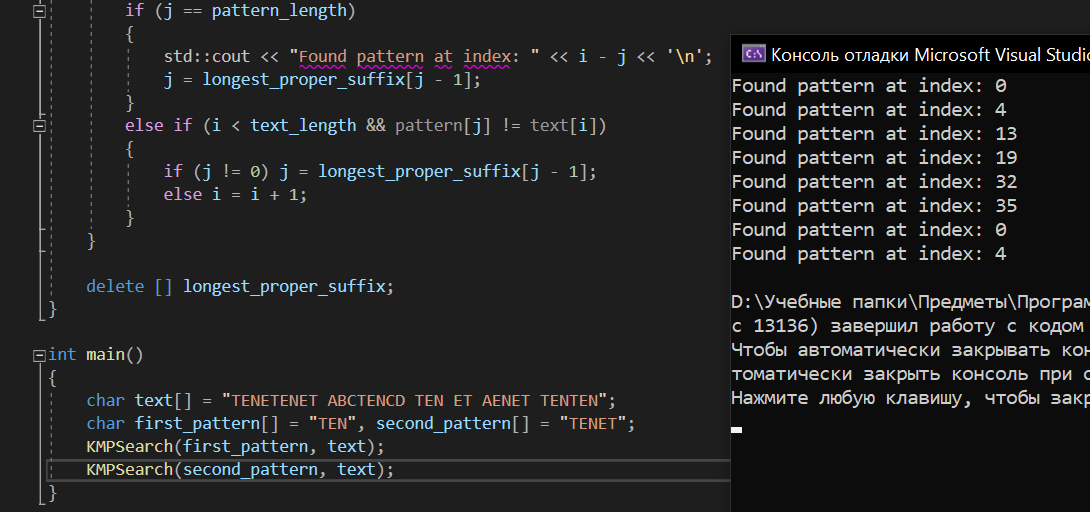


Рисунок 1 — Результаты