Бюджетное учреждение высшего образования Ханты-Мансийского автономного округа – Югры «СУРГУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт

Кафедра автоматики и компьютерных систем

Отчет

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

Проверила: старший преподаватель кафедры АиКС,

Карамышев. Э. Р

Выполнил: студент группы 605-31,

Хайитов Ш. Д.

Сургут 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc194868567)

[ОСНОВНАЯ ЧАСТЬ 4](#_Toc194868568)

[ЗАКЛЮЧЕНИЕ 10](#_Toc194868569)

# **ВВЕДЕНИЕ**

**Цель работы:** изучить основные принципы работы алгоритмов поиска подстрок, исследовать их свойства, закрепить навыки структурного программирования

**Задание:**

1. Разработать функцию, реализующую любой из алгоритмов поиска подстрок.

2. Исследовать алгоритмы поиска реализованной функции и функции стандартной библиотеки str str: построить и сравнить зависимости среднего времени выполнения от размера текста и размера образца. При исследовании алгоритмов обратить особое внимание на формирование текста, в котором осуществляется поиск.

3. Составить отчет, в котором привести графики полученных зависимостей, анализ свойств алгоритмов и выводы по работе.

# **ОСНОВНАЯ ЧАСТЬ**

Индивидуальное задание:

* Алгоритм Рабина-Карпа

Реализация алгоритма:

#define **S** 119

#define **d** 256

// Алгоритм Рабина-Карпа

int **SubStringRK**(char T[], int N, char P[], int M) {

    int h = 1;

    int p = 0;

    int t = 0;

    for (int i = 0; i < M - 1; i++)

        h = (**d** \* h) % **S**;

    for (int i = 0; i < M; i++) {

        p = (**d** \* p + P[i]) % **S**;

        t = (**d** \* t + T[i]) % **S**;

    }

    for (int i = 0; i <= N - M; i++) {

        if (p == t) {

            int j = 0;

            while (j < M && P[j] == T[i + j])

                j++;

            if (j == M)

                return i; // Подстрока найдена

        }

        if (i < N - M) {

            t = (**d** \* (t - h \* T[i]) + T[i + M]) % **S**;

            if (t < 0) t += **S**;

        }

    }

    return -1; // Подстрока не найдена

}

Функция strstr:

Функция strstr в языке С используется для поиска подстроки в строке. В определении, эта функция ищет первое вхождение строки needle (подстроки) в строке haystack (строке, в которой происходит поиск) и возвращает указатель на начало найденной подстроки. Если подстрока не найдена, функция возвращает NULL.

Табличные данные для построения графика зависимости T(n):

Таблица 1. Данные зависимости T(m) при фиксированном размере текста

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| РАЗМЕР МАССИВА - **9338324** | | | | | |
|  | 457852 | 2538864 | 5293639 | 7573051 | 9338324 |
| Алгоритм Рабина-Карпа | 0,004974 | 0,02852 | 0,050429 | 0,090926 | 0,095328 |
| strstr | 0,000521 | 0,001506 | 0,004208 | 0,006401 | 0,007162 |

График зависимости Алгоритма Рабина-Карпа представлен на рисунке 1.

Рисунок 1 – График зависимости Алгоритма Рабина-Карпа.

График зависимости функции strstr представлен на рисунке 2.

Рисунок 2 – График зависимости функции strstr.

Сравнение алгоритмов strstr и Рабина-Карпа при фиксированном размере представлен на рисунке 3.

Рисунок 3 – Графики сравнения зависимостей.

Табличные данные для построения графика зависимости T(m):

Таблица 2. Данные зависимости T(m) при фиксированном расположении текста.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| РАСПОЛОЖЕНИЕ КЛЮЧА - 748411 | | | | | |
| Размер | 748411 | 2265386 | 3858178 | 4515924 | 4914110 |
| Алгоритм Рабина-Карпа | 0 | 0,0009 | 0,006505 | 0,007727 | 0,008771 |
| strstr | 0 | 0 | 0 | 0,00094 | 0,001226 |

График зависимости Алгоритма Рабина-Карпа представлен на рисунке 4.

Рисунок 4 – График зависимости Алгоритма Рабина-Карпа.

График зависимости функции strstr представлен на рисунке 5.

Рисунок 5 – График зависимости функции strstr.

Сравнение алгоритмов strstr и Рабина-Карпа представлен на рисунке 6.

Рисунок 6 – Графики сравнения зависимостей.

По результатам экспериментов сравнения среднего времени поиска реализованной и стандартной функции во всех случаях можно увидеть, что стандартная функция языка Си strstr затрачивает очень малое количество времени в сравнении с алгоритмом поиска Рабина-Карпа.

**Листинг программы:**

#include <iostream>

#include <cstring>   // Для strlen, strstr

#include <fstream>

#include <chrono>    // Для измерения времени

using namespace std;

#define **S** 119

#define **d** 256

// Алгоритм Рабина-Карпа

int **SubStringRK**(char T[], int N, char P[], int M) {

    int h = 1;

    int p = 0;

    int t = 0;

    for (int i = 0; i < M - 1; i++)

        h = (**d** \* h) % **S**;

    for (int i = 0; i < M; i++) {

        p = (**d** \* p + P[i]) % **S**;

        t = (**d** \* t + T[i]) % **S**;

    }

    for (int i = 0; i <= N - M; i++) {

        if (p == t) {

            int j = 0;

            while (j < M && P[j] == T[i + j])

                j++;

            if (j == M)

                return i; // Подстрока найдена

        }

        if (i < N - M) {

            t = (**d** \* (t - h \* T[i]) + T[i + M]) % **S**;

            if (t < 0) t += **S**;

        }

    }

    return -1; // Подстрока не найдена

}

int **main**() {

**system**("chcp 65001 > nul"); // Установка кодировки UTF-8 для консоли

    // Открываем файл

    ifstream **infile**("example.txt");

    if (**!**infile) {

        cerr **<<** "Не удалось открыть файл.\n";

        return 1;

    }

    // Получаем размер файла

    infile.**seekg**(0, ios::end);

    size\_t size = infile.**tellg**();

    infile.**seekg**(0, ios::beg);

    // Читаем содержимое файла

    char\* text = new char[size + 1];

    infile.**read**(text, size);

    text[size] = '\0';

    infile.**close**();

    char pattern[] = "EVREYEBANIY";

    int N = **strlen**(text);

    int M = **strlen**(pattern);

    // Поиск алгоритмом Рабина-Карпа

    auto start = chrono::high\_resolution\_clock::**now**();

    int resultRK = **SubStringRK**(text, N, pattern, M);

    auto end = chrono::high\_resolution\_clock::**now**();

    chrono::duration<double> elapsed\_seconds\_RK = end **-** start;

    cout **<<** "\n=== Поиск с помощью алгоритма Рабина-Карпа ===" **<<** **endl**;

    if (resultRK != -1)

        cout **<<** "Подстрока найдена по индексу: " **<<** resultRK **<<** **endl**;

    else

        cout **<<** "Подстрока не найдена." **<<** **endl**;

    cout **<<** "Время выполнения: " **<<** elapsed\_seconds\_RK.**count**() **<<** " секунд." **<<** **endl**;

    // Поиск с использованием strstr

    start **=** chrono::high\_resolution\_clock::**now**();

    char\* strstr\_result = **strstr**(text, pattern);

    end **=** chrono::high\_resolution\_clock::**now**();

    chrono::duration<double> elapsed\_seconds\_strstr = end **-** start;

    cout **<<** "\n=== Поиск с использованием функции strstr ===" **<<** **endl**;

    if (strstr\_result) {

        int index = strstr\_result - text; // Вычисляем индекс найденной подстроки

        cout **<<** "Подстрока найдена по индексу: " **<<** index **<<** **endl**;

    } else {

        cout **<<** "Подстрока не найдена." **<<** **endl**;

    }

    cout **<<** "Время выполнения: " **<<** elapsed\_seconds\_strstr.**count**() **<<** " секунд." **<<** **endl**;

    // Освобождаем память

    delete[] text;

    return 0;

}

# **ЗАКЛЮЧЕНИЕ**

в ходе лабораторной работы были изучены основные принципы работы алгоритмов поиска подстрок, была разработана функция, реализующая алгоритм Рабина-Карпа, было произведено сравнение среднего времени выполнения реализованной функции и функции стандартной библиотеки strstr от размера текста и размера образца при успешном и неуспешном поисках, были построены соответствующие графики зависимостей