МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

---

Отчет по выполнению практического задания №6.2

**Тема: Поиск образца в тексте**

Дисциплина: Структуры и алгоритмы обработки данных

Выполнил студент Зуев Д.А.
Группа ИКБО-68-23

**Москва 2025**

**Цель работы:** освоить приёмы реализации алгоритмов поиска образца в тексте

## ЗАДАНИЕ

**Формулировка задачи**

Разработайте приложения в соответствии с заданиями в индивидуальном варианте.

В отчёте в разделе «Математическая модель решения (описание алгоритма)» разобрать алгоритм поиска на примере. Подсчитать количество сравнений для успешного поиска первого вхождения образца в текст и безуспешного поиска.

Определить функцию (или несколько функций) для реализации алгоритма поиска. Определить предусловие и постусловие.

Сформировать таблицу тестов с указанием успешного и неуспешного поиска, используя большие и небольшие по объему текст и образец, провести на её основе этап тестирования.

Оценить практическую сложность алгоритма в зависимости от длины текста и длины образца и отобразить результаты в таблицу

Персональный вариант:

| № | Задачи варианта |
|---|---|
| 14 | 1. Дан текст, состоящий из слов, разделенных знаками препинания. Переставить первое и последнее слово в тексте.<br><br>2. Дан текст и множество подстрок образцов. Определить сколько раз каждый из образцов входит в исходный текст. Реализовать на алгоритме Рабина-Карпа. Примечание: для всех образцов создать хеш-таблицу |

## ЗАДАНИЕ 1

**Формулировка задачи**

Дан текст, состоящий из слов, разделенных знаками препинания. Переставить первое и последнее слово в тексте.

**Код программы**

```
2168
2169  #include <iostream>
2170  #include <string>
2171  #include <sstream>
2172  #include <vector>
2173
2174  std::string swap_word(const std::string& text) {
2175      std::istringstream stream(text);
2176      std::vector<std::string> words;
2177      std::string word;
2178
2179      while (stream >> word) {
2180          words.push_back(word);
2181      }
2182
2183      if (words.size() < 2) {
2184          return text;  // Если меньше двух слов, возвращаем исходный текст
2185      }
2186
2187      // Меняем местами первое и последнее слово
2188      std::swap(words.front(), words.back());
2189
2190      std::string result;
2191      for (const auto& w : words) {
2192          result += w + " ";
2193      }
2194
2195      return result.substr(0, result.size() - 1);
2196
2197  }
2198
2199  int main() {
2200      std::string text = "swap my text";
2201      std::cout << text << "\n";
2202      std::string swappedText = swap_word(text);
2203
2204      std::cout << swappedText << std::endl;
2205      return 0;
2206  }
2207
```

Рисунок 1 – реализация функции перестановки

## ЗАДАНИЕ 2

### Формулировка задачи

Дан текст и множество подстрок образцов. Определить сколько раз каждый из образцов входит в исходный текст. Реализовать на алгоритме Рабина-Карпа. Примечание: для всех образцов создать хеш-таблицу

### Математическая модель решения (описание алгоритма)

Алгоритм Рабина — Карпа — это алгоритм поиска строки, который ищет шаблон, то есть подстроку, в тексте, используя хеширование. Он был разработан в 1987 году Михаэлем Рабином и Ричардом Карпом.

### Код программы

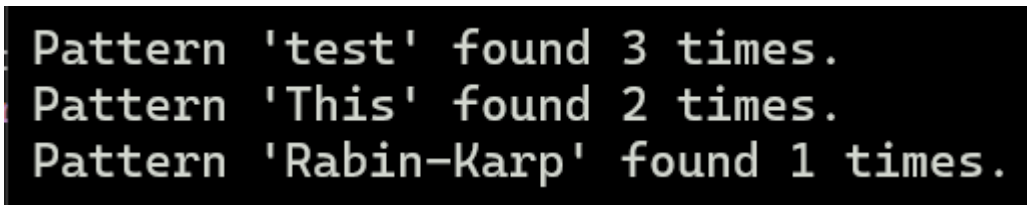
```
2213    const int d = 256; // Размер алфавита (256 для ASCII)
2214    const int q = 101; // Простое число для модуля
2215
2216    void searchPattern(const std::string& text, const std::vector<std::string>& patterns,
2217        std::unordered_map<std::string, int>& result) {
2218        int m, n;
2219
2220        for (const auto& pattern : patterns) {
2221            m = pattern.length();
2222            n = text.length();
2223            int p = 0; // Хэш-значение образца
2224            int t = 0; // Хэш-значение текста
2225            int h = 1;
2226
2227            // Вычисляем значение h = d^(m-1) % q
2228            for (int i = 0; i < m - 1; i++)
2229                h = (h * d) % q;
2230
2231            // Вычисляем хэш-значения образца и первого окна текста
2232            for (int i = 0; i < m; i++) {
2233                p = (d * p + pattern[i]) % q;
2234                t = (d * t + text[i]) % q;
2235            }
2236
2237            // Сравниваем хэш-значения
2238            for (int i = 0; i <= n - m; i++) {
2239                // Если хэш-значения совпадают, проверяем символы
2240                if (p == t) {
2241                    if (text.substr(i, m) == pattern) { // Проверяем саму подстроку
2242                        result[pattern]++;
2243                    }
2244                }
2245
2246                // Вычисляем хэш-значение для следующего окна текста
2247                if (i < n - m) {
2248                    t = (d * (t - text[i] * h) + text[i + m]) % q;
2249                    // Сделать t положительным
2250                    if (t < 0)
2251                        t += q;
2252                }
```

Рисунок 2 – реализация функции для создания хэш-таблицы и алгоритма Рабина - Карпа

```cpp
int main() {
    std::string text = "This is a test. This test is for the test of the Rabin-Karp algorithm.";
    std::vector<std::string> patterns = { "test", "This", "Rabin-Karp" };

    std::unordered_map<std::string, int> result;
    for (const auto& pattern : patterns) {
        result[pattern] = 0; // Инициализируем хэш-таблицу
    }

    searchPattern(text, patterns, result);

    // Вывод результатов
    for (const auto& pattern : patterns) {
        std::cout << "Pattern '" << pattern << "' found " << result[pattern] << " times." << std::endl;
    }

    return 0;
}
```

Рисунок 3 – реализация основной функции программы

**Результаты тестирования**




**Временная сложность**

- Лучший случай: O(n) — когда образец найден сразу.
- Средний случай: O(n/m) — где n — длина текста, m — длина образца.
- Худший случай: O(n·m) — когда текст и образец имеют много совпадений.

**Пространственная сложность**

- O(m) — для хранения хеш-таблицы.

# ВЫВОД

В ходе выполнения задачи была разработана программа, использующая алгоритм Рабина - Карпа для поиска ключевых слов в заданном тексте. Программа, реализующая алгоритм Рабина - Карпа, корректно находит и подсчитывает вхождения ключевых слов в заданном тексте. Она возвращает количество вхождений каждого слова, включая нулевые значения, что позволяет получить полное представление о наличии ключевых слов. Проведенные тесты подтвердили корректность работы алгоритма. Таким образом, цель по созданию эффективного инструмента для поиска подстрок была достигнута.