МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра МОиСИТ

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №6.2

«Поиск образца в тексте»
по дисциплине
«Структуры и алгоритмы обработки данных»

Выполнил студент группы ИКБО-10-24 Бикташев И. И.

Практическую работу выполнил «__»_____2025 г.

«Зачтено» «__»_____2025 г.

Москва 2025

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

Задание: Дано предложение, состоящее из слов, разделенных одним пробелом, удалить из него слова, встретившиеся более одного раза.

Описание алгоритма: Заводим словарь, в котором в качестве ключа хранится само слово и значения – количество вхождений данного слова в тексте. Если в словаре количество вхождений данного слова не равно 1, то данное слово в результат выполнения не записывается.

Реализация: Для реализации функционала словаря подключим библиотеку unordered_map. Чтобы быстро разделить строку по пробелам используем текстовый поток из библиотеки sstream. Считываем слова из потока и записываем их в вектор слов и в словарь вхождений. После перебираем все слова из вектора слов и записываем их в результат выполнения программы, если количество их вхождений равно единице (листинг 1)

Листинг 1 - Функция удаления повторов

```
string removeRepeatedWords(string sentence) {
    unordered_map<string, int> wordCount;
    istringstream in(sentence);
    string word;
    vector<string> words;

    while (in >> word) {
        wordCount[word]++;
        words.push_back(word);
    }

    string result;
    for (auto w : words) {
        if (wordCount[w] == 1) {
            if (!result.empty())
                result += " ";
            result += w;
        }
    }
    return result;
}
```

Проверим результат работы программы, введя несколько с повторяющимися словами. Результаты работы программы предоставлены на рисунках 1 и 2.

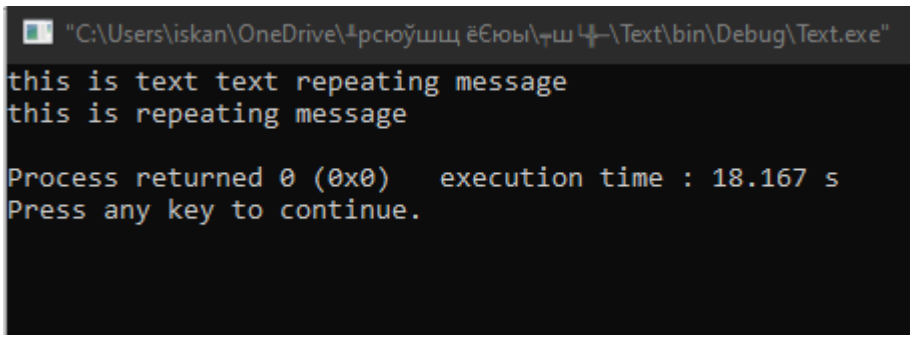

Рисунок 1 - Результат работы программы

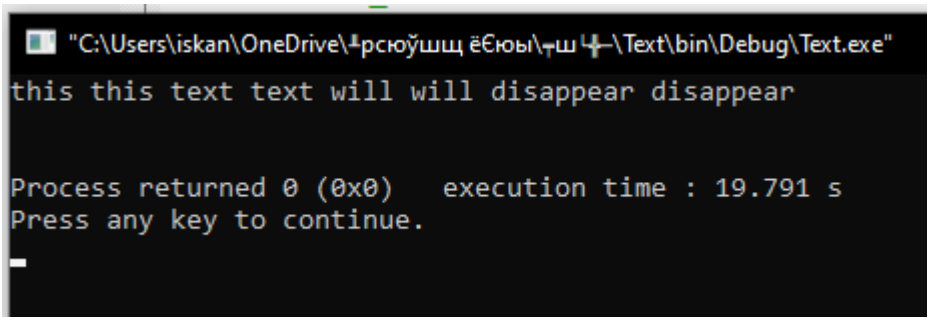

Рисунок 2 - Результат работы программы

Задание: Дано предложение, состоящее из слов, разделенных одним пробелом. Удалить из предложения все вхождения заданного слова, применяя для поиска слова в тексте метод Кнута-Мориса-Пратта.

Описание алгоритма: Идея алгоритма состоит в том, чтобы не прикладывать подстроку к строке со сдвигом всего в один символ, а максимально увеличить это расстояние, сократив таким образом количество сравнений. Для начала надо найти все префиксы текста, которые равно суффиксам, той же длины, то есть найти массив граней текста. Затем прикладывать подстроку к строке со сдвигом, где — текущая позиция в слове, — величина грани слова в позиции.

Реализация: Для реализации алгоритма потребуется три функции: функция для поиска массива граней (префикс функция), функция самого алгоритма КМП и функция которая удаляет из строки слова, найденные алгоритмом КМП.

Префикс функция для i-го символа образа возвращает значение, равное максимальной длине совпадающих префикса и суффикса подстроки в образе, которая заканчивается i-м символом (листинг 2)

Листинг 2 - Префикс функция

```
vector<int> prefixFunc(string pattern) {
    int N = pattern.length();
    vector<int> pi(N, 0);
    int j = 0;
    int i = 1;

    while (i < N) {
        if (pattern[i] == pattern[j]) {
            pi[i] = j + 1;
            i++;
            j++;
        } else if (j == 0) {
            pi[i] = 0;
            i++;
        } else {
            j = pi[j - 1];
        }
    }

    return pi;
}
```

Сама функция алгоритма КМП перебирает позиции в тексте и позиции в образе пока символы совпадают. Когда i-ый символ текста перестаёт быть равен j-ому символу образа, тогда присваиваем j значение массива граней в

предыдущем символе образа, либо на единицу если j == 0. Если образ в тексте найден записываем его в множество вхождений образа в текст (листинг 3).

Листинг 3 - Функция алгоритма КМП

```
unordered_set<int> KMP(string str, string sub) {
    unordered_set<int> solution;
    vector<int> pi =  prefixFunc(sub);
    int i = 0, j = 0;

    while (i < str.length()) {
        if (str[i] == sub[j]) {
            i++;
            j++;
            if (j == sub.size()) {
                solution.insert(i - j);
                j = pi[j - 1];
            }
        } else if (j == 0) {
            i++;
        } else {
            j = pi[j - 1];
        }
    }

    return solution;
}
```

Сама функция удаления образов в тексте берет результат выполнения алгоритма КМП и перебирает каждый символ текста. Если индекс данного символа встречается в результате работы алгоритма КМП, то индекс сдвигает на длину образа, иначе символ добавляется в результат выполнения функции (листинг 4).

Листинг 4 - Функция удаления вхождений в тексте

```
string removeWordOccurrences(string text, string pattern) {
    unordered_set<int> toDelete = KMP(text, pattern);
    string res;
    for (int i = 0; i < text.size(); i++) {
        if (toDelete.count(i) == 1) {
            i += pattern.size();
        }
        res += text[i];
    }

    // Удалить все пробелы
    istringstream in(res);
    string result, word;

    while (in >> word) {
        result += word + " ";
    }

    return result;
}
```

Результаты работы программы предоставлены н а рисунках 3 и 4.

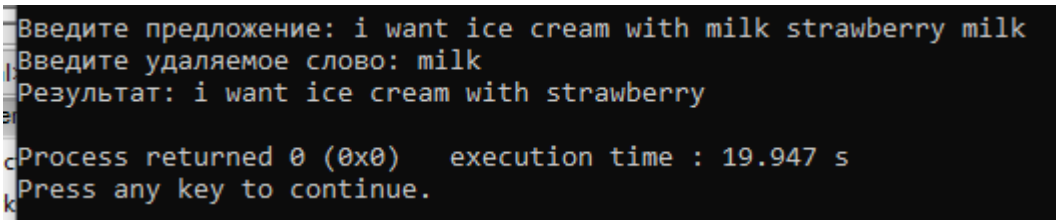

Рисунок 3 - Результат работы программы

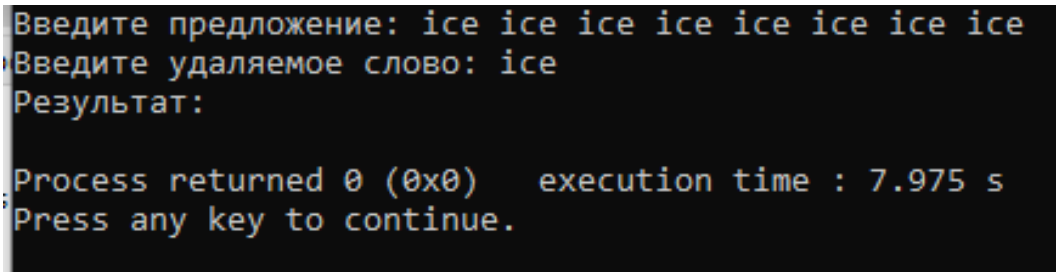

Рисунок 4 - Результат работы программы

Вывод

Поставленные задачи выполнены: реализованы алгоритмы поиска вхождений образа в строку с помощью простого поиска и метода Кнута-Мориса-Пратта. Создана функция удаления вхождений некоторого образа в подстроку с помощью поиска вхождения по методу Кнута-Мориса-Пратта.

Литература

1. Страуструп Б. Программирование. Принципы и практика с использованием C++. 2-е изд., 2016.

2. Документация по языку C++ [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/cpp/cpp/ (дата обращения 31.09.2025).

3. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL: https://online-edu.mirea.ru/course/view.php?id=4020 (дата обращения 31.09.2025).