****

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 Королихин В.Н.

группа ИКБО-21-23

**Москва 2024**

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

ЗАДАНИЕ 1

**Формулировка задачи**

Вариант 15

Дан массив ключевых слов языка С++. Упорядочить их, располагая слова в алфавитном порядке, используя обменную сортировку.

**Математическая модель решения (описание алгоритма)**

1. На вход подаётся неупорядоченный динамический массив строк keywords, хранящий все ключевые слова с++.
2. Int-переменная n инициализируется величиной этого массива keywords.size()
3. Для i = [0 … n-2]
4. Для j = [0 … n-i-2]
5. Если keywords[j] > keywords[j + 1], то keywords[j] и keywords[j + 1] меняются местами

**Код программы**

void bubbleSortKeywords(vector<string>& keywords) {

int n = keywords.size();

// Внешний цикл по всем элементам

for (int i = 0; i < n - 1; ++i) {

// Внутренний цикл для сравнения соседних элементов

for (int j = 0; j < n - i - 1; ++j) {

// Если текущий элемент больше следующего, меняем их местами

if (keywords[j] > keywords[j + 1]) {

swap(keywords[j], keywords[j + 1]);

}

}

}

}

**Результаты тестирования**

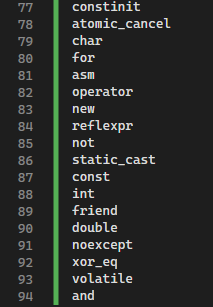


Рисунок 1 – Конец списка ключевых слов до сортировки

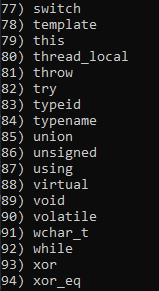


Рисунок 2 – Конец списка ключевых слов после сортировки

ЗАДАНИЕ 2

**Формулировка задачи**

Дан текст и множество подстрок образцов. Определить сколько раз каждый из образцов входит в исходный текст. Реализовать алгоритм Бойера-Мура-Хорспула. Примечание. Для всех образцов создать хеш-таблицу.

**Математическая модель решения (описание алгоритма)**

1. Создаём функцию int findInString(string T, string W), которая будет реализовывать алгоритм Бойера-Мура-Хорспула.
2. Инициализируем хеш-таблицу для хранения сдвигов каждого из символов в образце.
3. Устанавливаем сдвиг последнего символа в образце на длину этого образца
4. Для остальных символов, идя слева направо, устанавливаем их сдвиг равным расстоянию от конца строки до этого самого символа.
5. Инициализируем int-переменную indx = 0, она будет хранить текущий индекс поиска образца в тексте
6. Пока indx + длина образца не превышает длину строки, переходим к пункту 7), иначе к пункту 12)
7. Устанавливаем флаг found на true
8. Ищем первый справа символ в тексте, учитывая индекс поиска, который не совпадает с символом в образце
9. Если несовпадение было найдено, но в хеш-таблице нет несовпадающего символа, то сдвигаем индекс поиска на длину образца и устанавливаем флаг found на false
10. Если несовпадение было найдено, а запись в хеш-таблице есть, то осуществляется сдвиг индекса поиска на величину, хранящуюся в хеш-таблице по этому ключу и устанавливаем флаг found на false
11. Если флаг found в итоге равен false, то снова проходим цикл 6)
12. Если indx + длина образца превышает длину текста, возвращаем -1 (образец не входит в текст)
13. Иначе возвращаем indx

**Код программы**

// алгоритм Бойера-Мура-Хорспула.

int findInString(string T, string W) { // T - исходный текст, W - образец

// хеш-таблица сдвигов

unordered\_map<char, int> skip{};

// заполняем хеш-таблицу

skip[W.back()] = W.length();

for (int i = 0; i < W.length() - 1; i++) {

skip[W[i]] = W.length() - i - 1;

}

// индекс поиска

int indx = 0;

// пока поиск не вышел за границы текста

while (indx + W.length() <= T.length()) {

// флаг

bool found = true;

// перебираем символы в образце

for (int i = 0; i < W.length(); i++) {

// справа налево проверяем совпадение с текстом

if (T[indx + W.length() - i - 1] != W[W.length() - i - 1]) {

// если не совпадает символ. которого нет в хеш-таблице

if (skip[T[indx + W.length() - i - 1]] == 0) {

// значит этого символа в принципе нет в образце W

// и можно делать сдвиг поиска на всю длину образца

indx += W.length();

}

// если не совпадает символ. котороый есть в хеш-таблице

else {

// то сдвигаем на соответствующую велчину

indx += skip[T[indx + W.length() - i - 1]];

}

// обновляем флаг

found = false;

break;

}

}

// нашли первое вхождение?

if (found) break;

}

// Если поиск вышел за пределы текста, возвращаем -1 (образец не был найден в тексте)

if (indx + W.length() > T.length()) return -1;

// Иначе поиск был успешен. Возвращаем индекс поиска

return indx;

}

// Функция поиска колиества вхождений W в T

int countOccurances(string T, string W) {

// Счётчик

int count = 0;

// Индекс поиска

int searchStart = 0;

// Если в оставшемся куске текста ещё есть вхождения

while (findInString(T.substr(searchStart), W) != -1) {

// обновляем счётчик

count++;

// обновляем индекс поиска

searchStart += findInString(T.substr(searchStart), W) + 1;

}

// возвращаем результат

return count;

}

**Результаты тестирования**

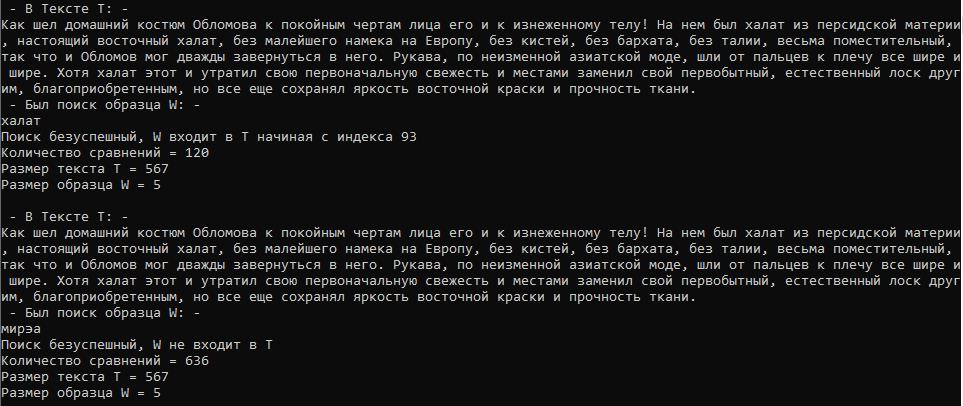


Рисунок 3 – Успешный и безуспешный поиски в большом тексте

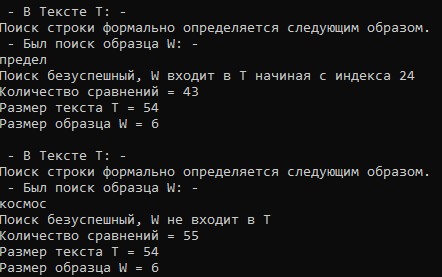


Рисунок 4 – Успешный и безуспешный поиски в малом тексте

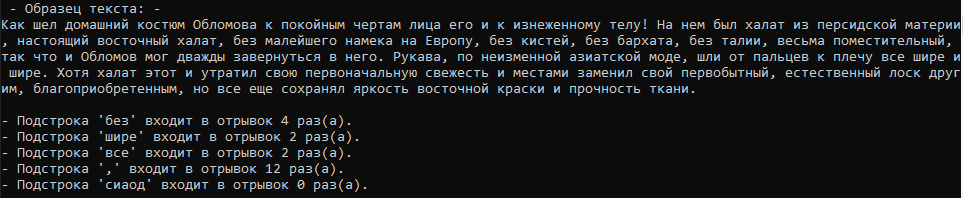


Рисунок 5 – Нахождение количества вхождений множества образцов

ВЫВОДЫ

Алгоритм Бойера-Мура-Хорспула оказался эффективным методом поиска образца в строке. Его преимущество заключается в использовании эвристики смещения, что позволяет пропускать часть символов основной строки при неудачном совпадении. Это делает алгоритм особенно полезным для работы с большими текстами и в случаях, когда алфавит входных данных достаточно велик.

Программа получилась достаточно простой. Простота влечет надежность и эффективность программы, их проще создавать и сопровождать.

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

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

2. Документация по языку С++ [Электронный ресурс]. URL: https://docs.microsoft.com/ruru/cpp/cpp/ (дата обращения 01.09.2021).

3. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL: https://online-edu.mirea.ru/course/view.php?id=4020 (дата обращения 01.09.2021).