|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования |
| **«МИРЭА – Российский технологический университет»** |
| **РТУ МИРЭА** |
|  |

|  |  |
| --- | --- |
| **Отчет по выполнению практического задания № 6, часть 2** | |
| **Тема:** | |
| **«Быстрый доступ к данным с помощью хеш-таблиц»** | |
| Дисциплина: «Структуры и алгоритмы обработки данных» | |
|  | Выполнил студент: Лисовский И.В |
|  | Группа: ИКБО-21-23 |

Москва – 2024

СОДЕРЖАНИЕ

[1 ЦЕЛЬ 3](#_Toc181881921)

[2 ЗАДАНИЕ 4](#_Toc181881922)

[2.1 Формулировка задачи 4](#_Toc181881923)

[2.2 Математическая модель решения задачи 1 4](#_Toc181881924)

[2.3 Код программы 1 с комментариями 8](#_Toc181881925)

[2.4 Тестирование программы 1 10](#_Toc181881926)

[2.5 Математическая модель решения задачи 2 11](#_Toc181881927)

[2.6 Код программы 2 с комментариями 13](#_Toc181881928)

[2.7 Тестирование программы 2 14](#_Toc181881929)

[3 ВЫВОД 15](#_Toc181881930)

[4 ЛИТЕРАТУРА 17](#_Toc181881931)

# **1 ЦЕЛЬ**

Освоить и закрепить навыки реализации алгоритмов поиска образца в тексте, таких как алгоритмы Бойера-Мура и Рабина-Карпа. Исследовать их эффективность и определить количество сравнений для успешного и безуспешного поиска. Оценить практическую сложность алгоритмов и научиться формировать и анализировать тестовые данные.

# **2 ЗАДАНИЕ**

## **2.1 Формулировка задачи**

Разработайте приложения в соответствии с заданиями в индивидуальном варианте (п.2).

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 (для отчета).

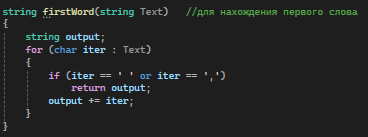
В отчёте сделайте вывод о проделанной работе, основанный на полученных результатах.

**Вариант №19:**

## **2.2 Математическая модель решения задачи 1**

1. **Определение первого слова:**

Выделение первого слова из строки текста. Для этого необходимо итерироваться по строке и собирать символы, пока не встретится первый разделитель (например, пробел или запятая).



Листинг 1.1 — Функция для нахождения первого слова

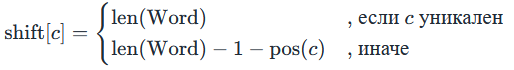
1. **Построение таблицы смещений для алгоритма Бойера-Мура:**

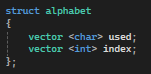
Построение таблицы смещений для символов из первого слова. Она помогает определить, на сколько символов можно сдвинуться вперед при несоответствии символов при сравнении текстовых строк.

Таблица строится следующим образом:

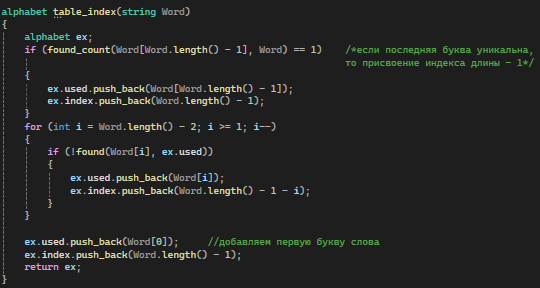
* Для каждого уникального символа, начиная с конца слова, вычисляется максимально возможное смещение, так чтобы пропущенное в сравнении символов не может быть началом новой строки.
* Если символ уникален, его смещение равно длине слова.

Формально для символа c в слове Word:





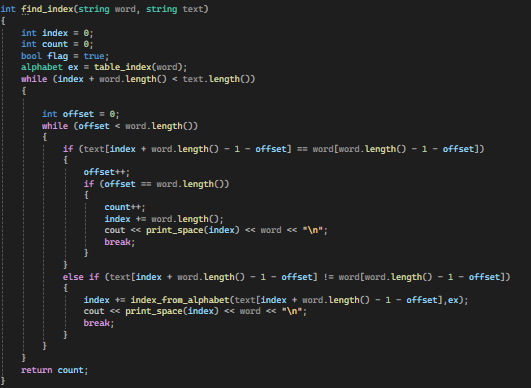
Листинг 1.2 — Структура алфавита слова



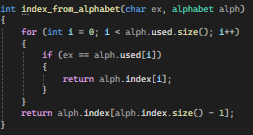
Листинг 1.3 — Функция для создания алфавита

1. **Поиск первого слова в тексте:**

Использование алгоритма Бойера-Мура для поиска всех вхождений первого слова в текст. Сравнивание начинается с конца слова и идет назад. При несоответствии производится сдвиг, основанный на таблице смещений. Для текста и слова Word повторять до конца текста: Если текст [i+j]=Word[j], для всех j от len(Word) − 1 до 0, то найдено вхождение, иначе сдвинуться на текст[i+len(Word)−1].

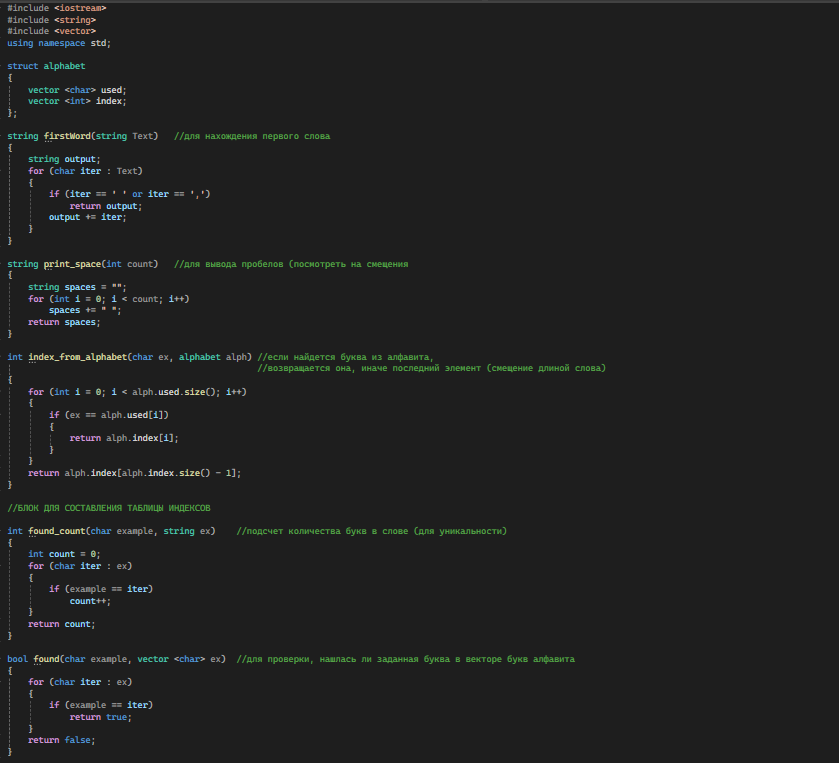


Листинг 1.4 — Функция для нахождения слова с помощью алгоритма БМ

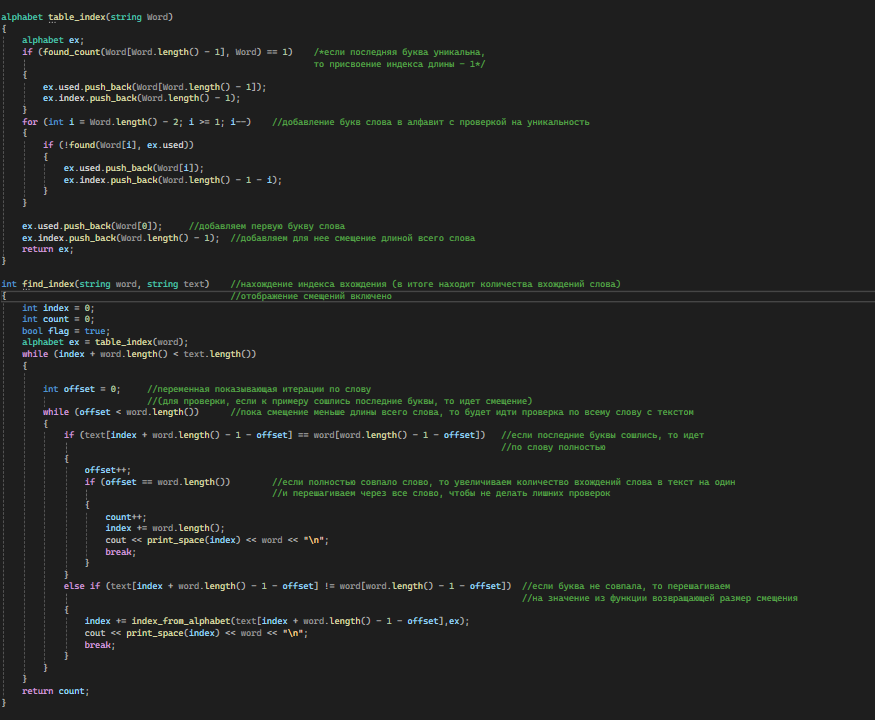


Листинг 1.5 — Вспомогательная функция для возврата значения смещения из алфавита

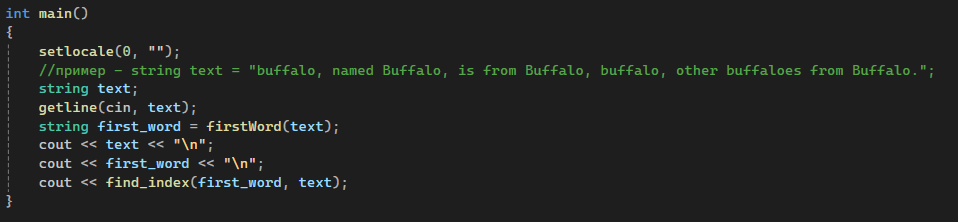
## **2.3 Код программы 1 с комментариями**



Листинг 1.6 — Код программы



Листинг 1.7 — Код программы



Листинг 1.8 — Код программы

## **2.4 Тестирование программы 1**

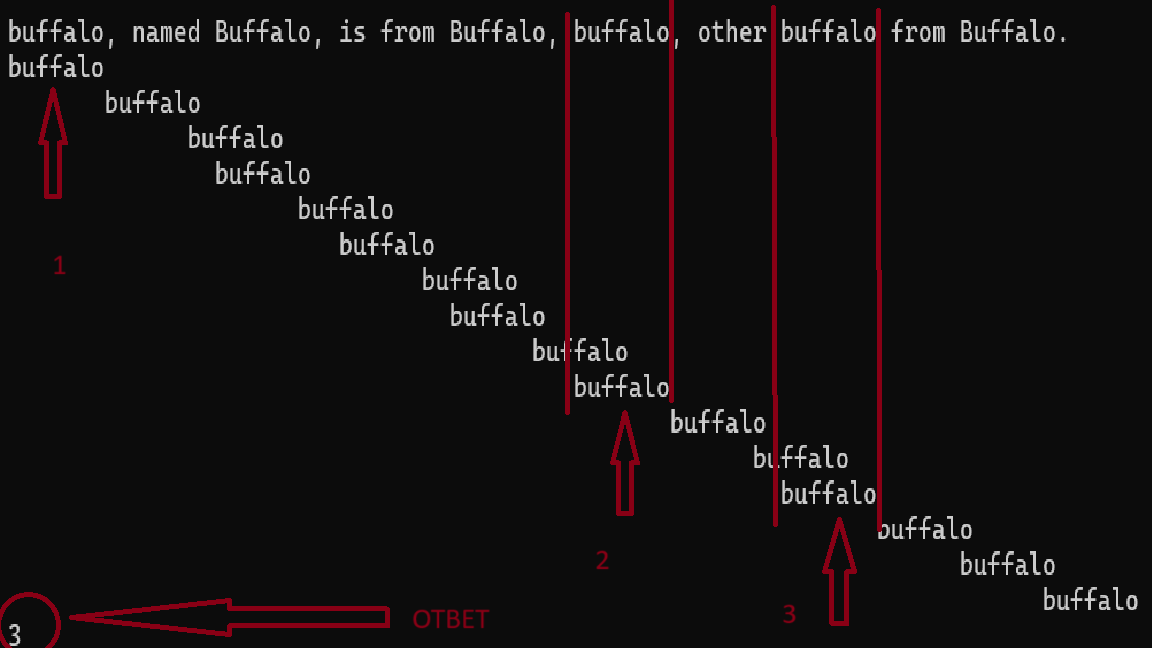
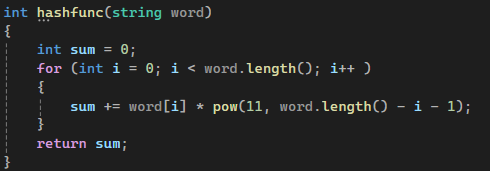


Рисунок 1 — Тестирование кода

## **2.5 Математическая модель решения задачи 2**

1. **Хеш-функция:**

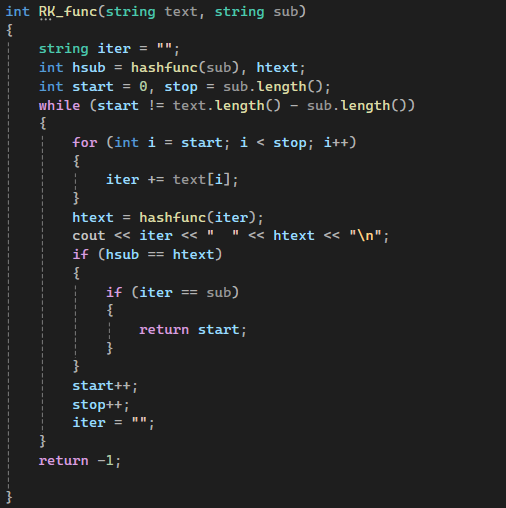
Для каждой подстроки вычисляется хеш-код, который преобразует строку в числовое значение для быстрого сравнения. Используется полиномиальная хеш-функция.



Листинг 2.1 — Функция для вычислений хеша

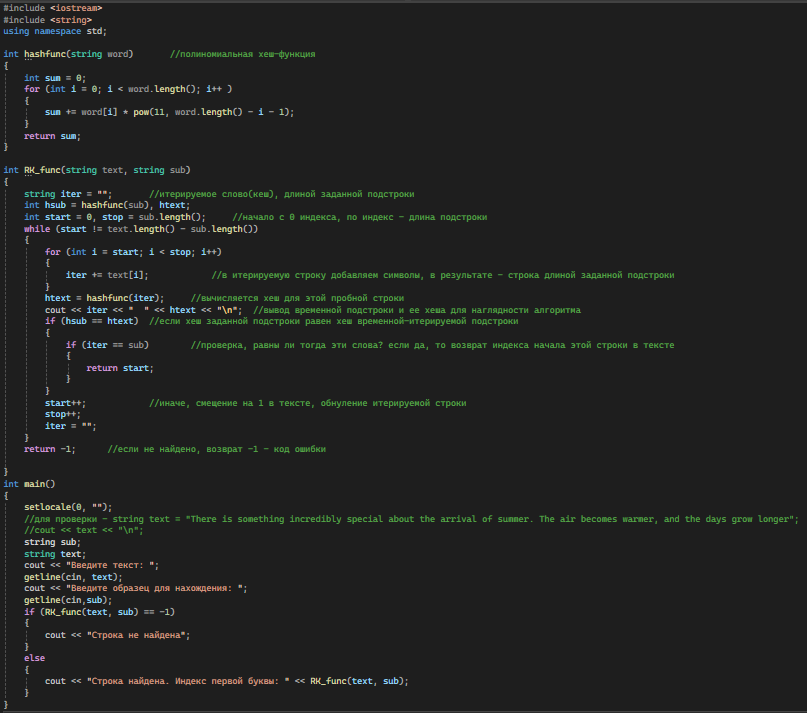
1. **Алгоритм Рабина-Карпа:**

Вычисляется хеш-код искомой подстроки. Сравниваются хеш-коды подстроки и каждого окна в исходном тексте. Если хеши совпадают, выполняется дополнительное посимвольное сравнение для подтверждения. Для каждой позиции в исходном тексте вычисляется хеш-код подстроки той же длины, что и искомая. Если хеши совпадают, производится посимвольное сравнение подстроки и образца. Если совпадение найдено, возвращается индекс начала вхождения. Если совпадение не найдено, продолжается поиск. Если подстрока не найдена, то возвращается -1. Алгоритм Рабина-Карпа по сложности в худшем O(n\*m) случае имеет сложность , по времени работы ничем сильно не выделяется, но при этом делает больше сравнений. Ведь перед тем как сверять строку мы сначала сверяем хеш значения, а если и находится, то целую подстроку.



Листинг 2.2 — Функция, реализовывающая алгоритм Рабина-Карпа

## **2.6 Код программы 2 с комментариями**



Листинг 2.3 — Код программы

## **2.7 Тестирование программы 2**

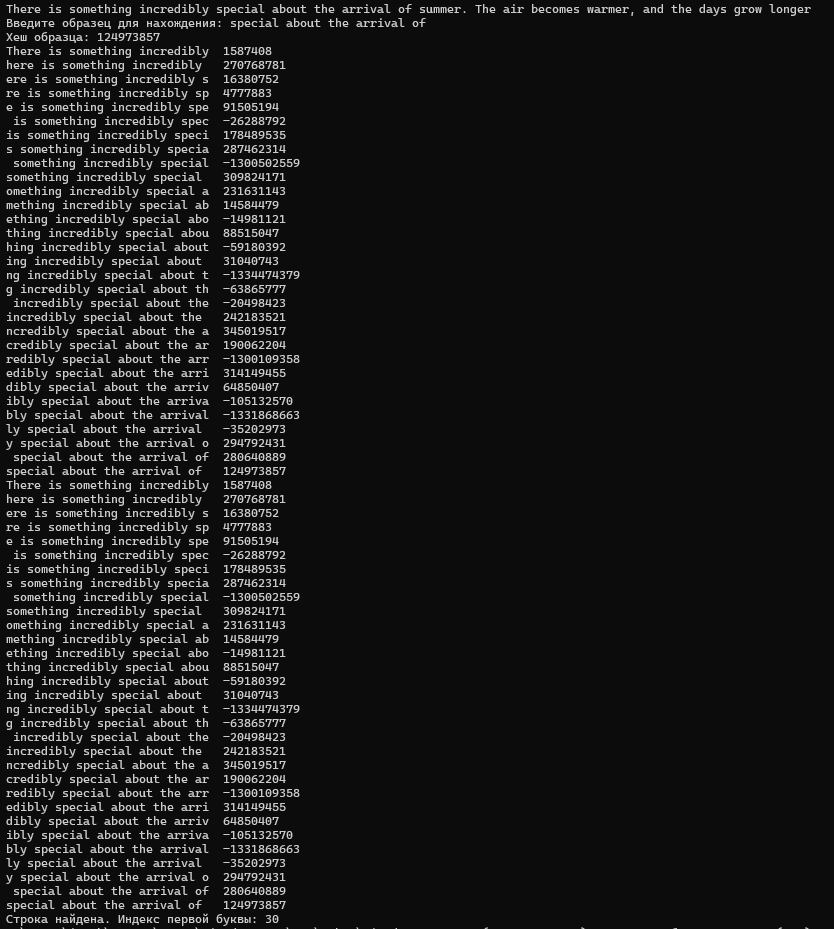


Рисунок 2 — Тестирование кода

# **3 ВЫВОД**

В ходе выполнения практической работы были реализованы и изучены два алгоритма поиска подстрок: Бойера-Мура и Рабина-Карпа. Эти алгоритмы продемонстрировали свои сильные и слабые стороны в контексте различных задач. Оба алгоритма значительно расширили понимание различных подходов к задаче поиска подстрок.

1. **Алгоритм Бойера-Мура:**

**Эффективность:** Этот алгоритм показал высокую скорость поиска за счёт использования таблицы смещений, что позволило уменьшить количество сравнений символов.

**Сложность:** В лучшем случае алгоритм работает за время близкое к линейному, что делает его полезным при анализе больших текстовых массивов.

**Применимость:** Подходит для задач, где требуется найти частое и регулярное вхождение подстрок, как в случае с поиском первого слова в тексте.

1. **Алгоритм Рабина-Карпа:**

**Эффективность:** Предоставил удобный способ поиска подстрок за счёт хеширования, что позволяет быстро отсеять несовпадающие окна.

**Сложность:** Хотя теоретически он работает в среднем за линейное время, на практике может иметь коллизии хешей, что требует дополнительных проверок.

**Применимость:** Идеально подходит для задач, требующих проверки на плагиат, где необходимо многократно проверять сходства между большими текстами.

Реализация данных методов позволила глубже понять их теоретическую основу и увидеть, как на практике они решают задачи различной специфики и сложности. Полученные результаты подтвердили эффективность каждого из алгоритмов в соответствующих контекстах, что будет полезно при выборе подходящего метода для различных практических применений.

# **4 ЛИТЕРАТУРА**

1. Вирт Н. Алгоритмы и структуры данных. Новая версия для Оберона, 2010.

2. Кнут Д. Искусство программирования. Тома 1-4, 1976-2013.

3. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для про-граммистов и любопытствующих, 2017.

4. Кормен Т.Х. и др. Алгоритмы. Построение и анализ, 2013.

5. Лафоре Р. Структуры данных и алгоритмы в Java. 2-е изд., 2013.

6. Макконнелл Дж. Основы современных алгоритмов. Активный обуча-ющий метод. 3-е доп. изд., 2018.

7. Скиена С. Алгоритмы. Руководство по разработке, 2011.

8. Хайнеман Д. и др. Алгоритмы. Справочник с примерами на C, C++, Java и Python, 2017.

9. Гасфилд Д. Строки, деревья и последовательности в алгоритмах. Ин-форматика и вычислительная биология, 2003.

По языку С++:

10. Страуструп Б. Программирование. Принципы и практика с использова-нием C++. 2-е изд., 2016.

11. Павловская Т.А. C/C++. Программирование на языке высокого уровня, 2003.

12. Прата С. Язык программирования С++. Лекции и упражнения. - 6-е изд., 2012.

13. Седжвик Р. Фундаментальные алгоритмы на C++, 2001-2002

14. Хортон А. Visual C++ 2010. Полный курс, 2011.

15. Шилдт Г. Полный справочник по C++. 4-е изд., 2006.