**Санкт-Петербургский Национальный Исследовательский Университет Информационных технологий, механики и оптики**

**Отчет**

**Дисциплина: Объектно-ориентированное программирование.**

**Лабораторная работа 1. Алгоритмы хеширования.**

**Выполнил: Влазнев Д.В.**

**Группа № K3223**

**Проверил: доцент Иванов С.Е.**

**Санкт-Петербург**

**2024**

**Цель работы**:

Изучить алгоритмы хеширования, реализовать алгоритм Рабина-Карпа поиска подстроки в строке с применением хеширования.

**Задачи**:

1. Изучить алгоритм Рабина-Карпа.
2. Выполнить задания для лабораторной работы.

**Задание для лабораторной работы:**

1. Реализовать поиск одинаковых строк. Дан список строк S[1..n], каждая длиной не более m символов. Требуется найти все повторяющиеся строки и разделить их на группы, чтобы в каждой группе были только одинаковые строки.
2. Реализовать алгоритм Рабина-Карпа поиска подстроки в строке за O (n).

# Содержание

[Содержание 3](#_Toc160972639)

[Задание 1 4](#_Toc160972640)

[Задание 2 5](#_Toc160972641)

[Контрольные вопросы 8](#_Toc160972642)

[Выводы 10](#_Toc160972643)

# Задание 1

Для того, чтобы найти одинаковые строки, необходимо найти хэш каждой строки и сравнить их. Для этого напишем функцию вычисления полиномиального хэша (См. Рисунок 1).



Рисунок 1 – Код программы

Таким образом, считывая строки и вычисляя их кэш, группируем их по хэшу (См. Рисунок 2–3).



Рисунок 2 – Код программы

Изображение выглядит как текст, электроника, снимок экрана, мультимедиа

Автоматически созданное описание

Рисунок 3 – Пример вывода программы

# Задание 2

Для реализации алгоритма Рабина-Карпа для начала нужно также задать функцию хэша. Также для оптимизации была разработана функция возведения в степень с постоянным делением по модулю, чтобы не произошел оверфлоу (См. Рисунок 4).



Рисунок 4 – Код программы

При создании класса необходимо указать базовое число, которое будет возводиться в степень, по умолчанию 256, и простое число для модуляции, по умолчанию, 3673 (См. Рисунок 5).



Рисунок 5 – Код программы

Основная функция находит все вхождения паттерна в тексте и возвращает список индексов. Для улучшения производительности используется класс Span для сравнения строк и кольцевой кэш (См. Рисунок 6–7).



Рисунок 6 – Код программы

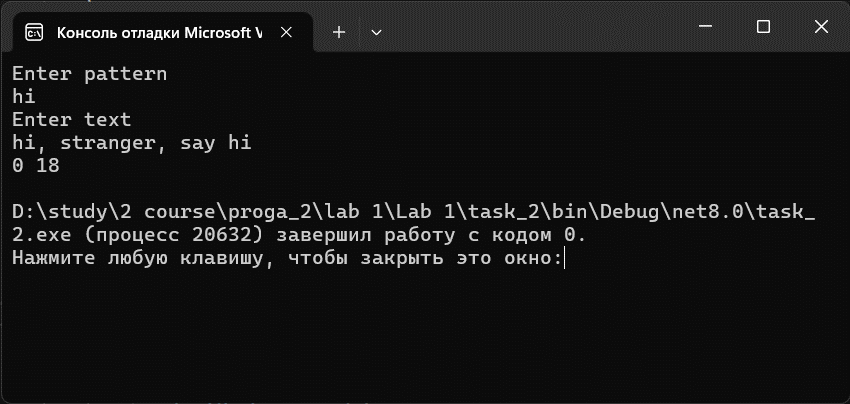


Рисунок 7 – Пример вывода программы

# Контрольные вопросы

1. Алгоритмы хеширования используются для решения различных задач, включая:

- Уникальная идентификация данных или объектов.

- Быстрый поиск и доступ к данным по ключу.

- Хранение пар "ключ-значение" в хеш-таблицах.

- Цифровая подпись и проверка целостности данных.

2. Проблема коллизий возникает, когда два различных входных данных дают одинаковый хеш-код. Для решения этой проблемы существуют различные методы:

- Метод цепочек (открытое хеширование): при коллизии элементы с одинаковым хешем добавляются в связанный список или другую структуру данных.

- Линейное пробирование: при коллизии происходит поиск следующего свободного слота в хеш-таблице.

- Двойное хеширование: используется вторая хеш-функция для определения нового индекса при коллизии.

3. Ограничения и условия на применение алгоритма хеширования включают:

- Хеш-функция должна быть быстрой и равномерно распределенной.

- Коллизии должны быть эффективно обрабатываемыми.

- Размер хеш-таблицы должен быть достаточно большим для уменьшения коллизий.

4. Для реализации алгоритма хеширования в C# часто используются следующие конструкции ООП:

- Классы для представления хеш-таблиц и элементов данных.

- Методы для вычисления хеш-кода объектов.

5. Сложность алгоритма хеширования оценивается по времени выполнения операций вставки, поиска и удаления элементов в хеш-таблице. Обычно она выражается как O(1) для успешных операций при равномерном распределении данных.

6. Точность алгоритма хеширования оценивается по уровню коллизий и равномерности распределения хеш-кодов. Чем меньше коллизий и лучше распределены значения хеш-кодов, тем выше точность алгоритма.

# Выводы

В ходе выполнения работы изучены и реализованы средствами ООП C# алгоритмы хеширования.

Ссылка на код: <https://github.com/geherious/OOP-2/tree/main/lab%201>