|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | |  |
|  | Институт информационных технологий (ИТ) | |
|  | Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ) | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**  **«Алгоритмы поиска»** | | | |  |
| **по дисциплине «Структуры и алгоритмы обработки данных (часть 2/2)»** | | | |  |
|  | | | |  |
| Выполнил студент группы ИКБО-41-23 | | Попов А.В. | |  |
|  | |  | |  |
| Принял  *Ассистент* | | Рысин М.Л. | |  |
| Практические работы выполнены | «\_\_\_»\_\_\_\_\_\_\_2024 г. | | (подпись студента) | |
| «Зачтено» | «\_\_\_»\_\_\_\_\_\_\_2024 г. | | (подпись преподавателя) | |
|  |  | |  | |

СОДЕРЖАНИЕ

[1 ПОСТАНОВКА ЗАДАЧИ 3](#__RefHeading___Toc815_1995867043)

[2 ЗАДАНИЕ 4](#__RefHeading___Toc817_1995867043)

[2.1 Формулировка задачи 4](#__RefHeading___Toc819_1995867043)

[2.2 Математическая модель решения 5](#__RefHeading___Toc716_30053032)

[2.3 Реализация задачи 7](#__RefHeading___Toc821_1995867043)

[2.4 Результаты тестирования 10](#__RefHeading___Toc718_30053032)

[3 ВЫВОД 13](#__RefHeading___Toc839_1995867043)

[4 Ответы на вопросы 14](#__RefHeading___Toc720_30053032)

# 1 ПОСТАНОВКА ЗАДАЧИ

Освоить приёмы хеширования и эффективного поиска элементов множества. Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к элементам динамического множества полезных данных. Реализовать текстовый командный интерфейс пользователя для возможности вызова методов в любой произвольной последовательности.

# 2 ЗАДАНИЕ

## 2.1 Формулировка задачи

Разработать приложение, которое использует хеш-таблицу (пары «ключ – хеш») для организации прямого доступа к элементам динамического множества полезных данных. Реализовать множество на массиве, структура элементов (перечень полей) которого приведена в индивидуальном варианте.

Приложение должно содержать класс с базовыми операциями: вставки, удаления, поиска по ключу, вывода. Включить в класс массив полезных данных и хеш-таблицу. Хеш-функция подберается самостоятельно, с использованием правил выбора функции.

Реализовать расширение размера таблицы и рехеширование, когда это требуется, в соответствии с типом разрешения коллизий.

Предусмотреть автоматическое заполнение таблицы 5-7 записями.

Реализовать текстовый командный интерфейс пользователя для возможности вызова методов в любой произвольной последовательности, сопроводить вывод достаточными для понимания происходящего сторонним пользователем подсказками.

Провести полное тестирование программы (все базовые операции, изменение размера и рехеширование).

Персональный вариант №11:

Метод хеширования (тип последовательностей проб): открытая адресация (двойное хеширование).

Структура элемента множества: номер телефона – последовательность 10 символов, адрес.

## 2.2 Математическая модель решения

Хеш-таблица использует две хеш-функции:

1. Первая хеш-функция:

*h1( k ) = ( k mod p1 ) mod m*

где *k* – ключ (номер телефона), *p1* – простое число (по умолчанию 31), *m* – текущий размер таблицы.

1. Вторая хеш-функция:

*h2( k ) = ( k mod p2 ) mod m*

где *k* – ключ (номер телефона), *p2* – другое простое число (по умолчанию 17), *m* – текущий размер таблицы.

Эти функции помогают определить начальную позицию ключа в таблице, а также шаг для разрешения коллизий. Коллизия разрешается методом двойного хеширования: при столкновении (когда ячейка уже занята) следующая проверяемая позиция вычисляется по формуле:

*indexследующий = ( indexтекущий + h2( k ) ) mod m*

Когда нагрузка таблицы (отношение числа элементов к размеру таблицы) превышает 0.5, таблица удваивается в размере. Для этого выполняются следующие действия:

1. Создается новая таблица, в два раза больше предыдущей.
2. Все существующие элементы из старой таблицы перехешируются в новую. Для этого каждый ключ обрабатывается заново, с использованием обновленных значений размера таблицы.

Интерфейс позволяет пользователю управлять хеш-таблицей с помощью команд: добавления, удаления, поиска и отображения элементов. Каждая команда обрабатывается соответствующей функцией. При необходимости интерфейс вызывает операции рехеширования или сообщает об ошибках, например, если пользователь пытается найти или удалить несуществующий элемент.

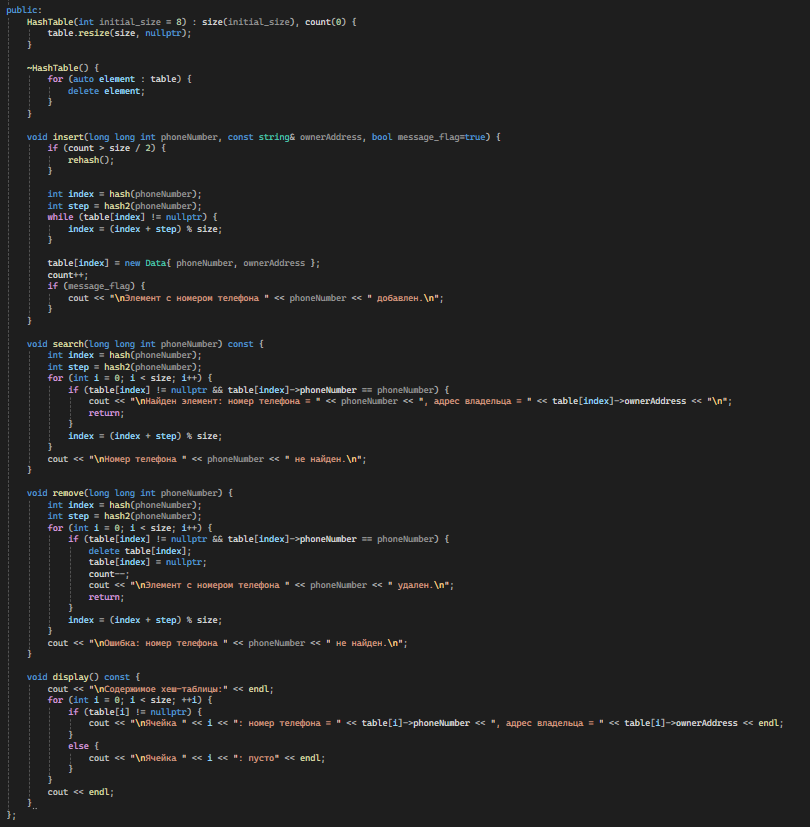
Для хранения номера телефона в программе используется тип данных ***long long int***. Этот выбор обусловлен следующими соображениями:

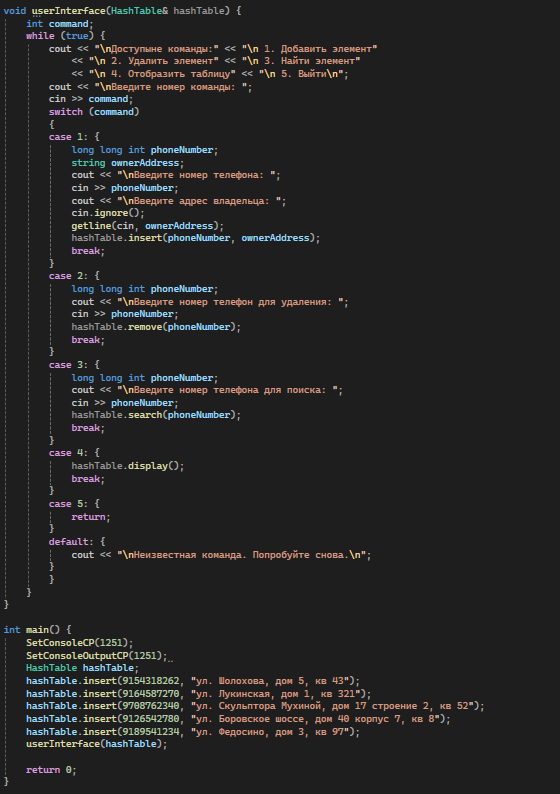
* Номер телефона состоит из 10 цифр. Например, номер вида 9154318262 представляет собой числовое значение, которое занимает 10 разрядов. Тип ***long long int*** гарантирует, что такие числа могут быть корректно представлены без риска переполнения.
* Использование числового представления позволяет легко вычислять хеш-функции. В данном случае операции вычисления остатка от деления и другие арифметические операции с номерами телефонов выполняются быстро и эффективно.

Если бы номера телефонов хранились как строки (***string***), обработка данных потребовала бы дополнительных вычислительных затрат. Таким образом, выбор ***long long int*** для хранения телефонных номеров делает программу более простой, эффективной и безопасной с точки зрения корректного представления данных.

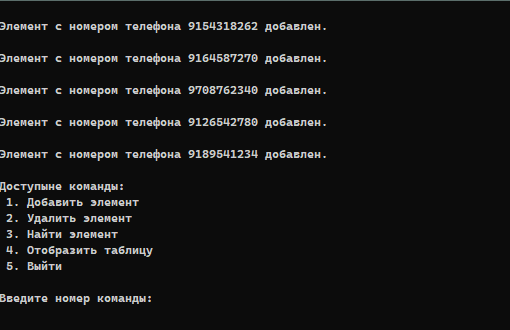
## 2.3 Реализация задачи

Рисунок 1 — Структура множества полезных данных, класс хеш-таблицы и описанные в нём методы хеширования и рехеширования.

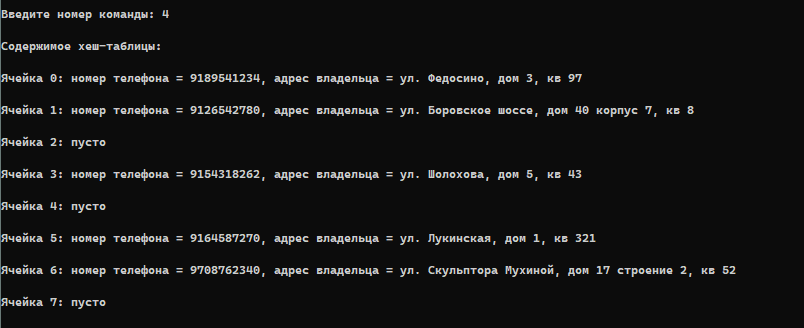
Рисунок 2 — Конструктор и декструктор таблицы, методы добавления, поиска, удаления и вывода хеш-таблицы

Рисунок 3 — функция, реализующая пользовательский интерфейс, основная функция программы и автоматическое заполнение таблицы записями

## 2.4 Результаты тестирования

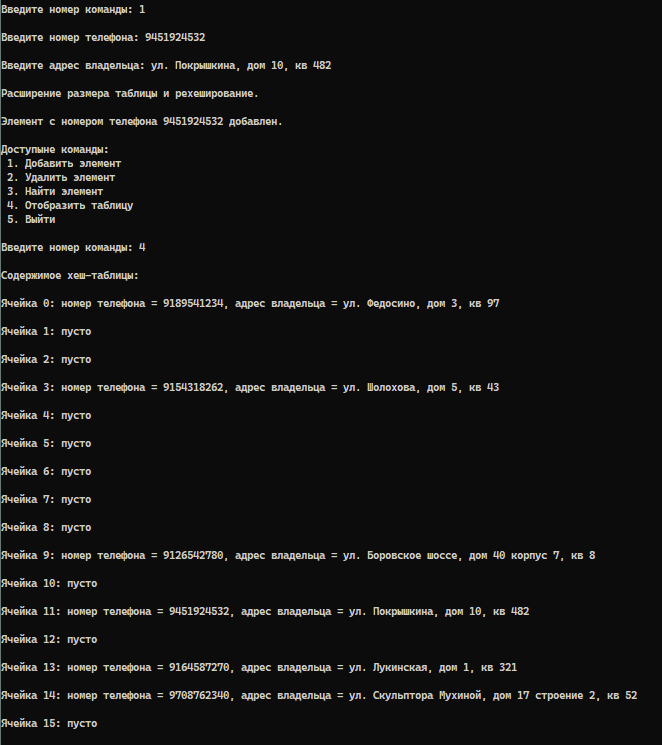
Рисунок 4 — Начальное состояние после запуска

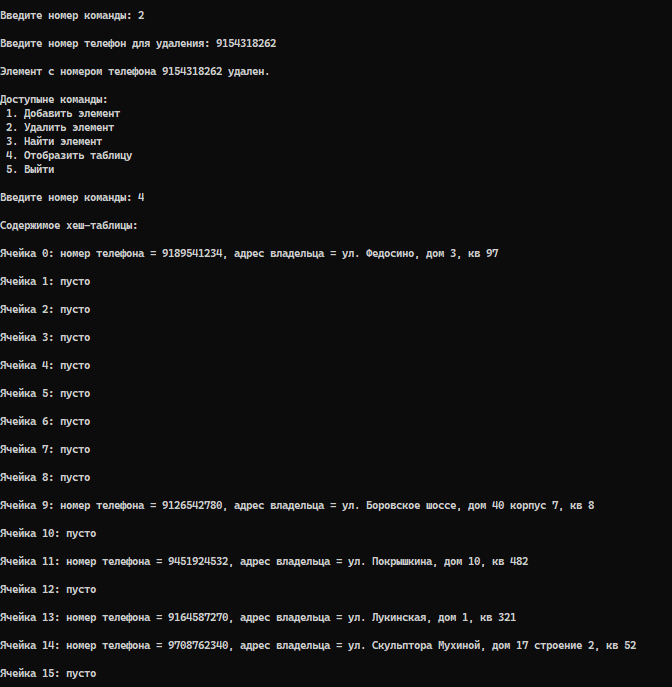
После запуска программы мы имеем на выбор 5 команд (Рисунок 4). Для их выполнения следует ввести номер необходимой команды и заполнить её аргументы.

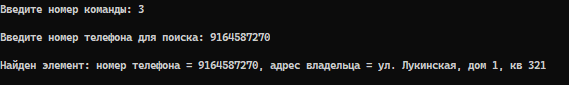
Рисунок 5 — Результат вывода хеш-таблицы

Как мы можем заметить (Рисунок 5), выводятся все ячейки текущей хеш-таблицы, в формате «Ячейка N: множество полезных данных», где N – хеш, полученный в результате применения алгоритма хеширования на номере телефона (который является ключевым значением).

Так как хеш-таблица уже заполнена больше, чем на половину, 5 из 8 элементов, то после добавления ещё одного таблица будет рехеширована и новое значения появится в ней (Рисунок 6).

Рисунок 6 — Добавление нового элемента и рехеширование таблицы

Рисунок 7 — Удаление элемента

Рисунок 8 — Поиск элемента по номеру (ключу)

# 3 ВЫВОД

В ходе выполнения работы было разработано приложение, использующее хеш-таблицу для организации прямого доступа к данным. В приложении реализованы основные операции: вставка, удаление, поиск по ключу и вывод. Хеш-таблица работает на массиве, где в качестве ключа используется номер телефона.

Для работы с коллизиями была реализована рехешировка и расширение таблицы, когда она заполняется. Также добавлен текстовый командный интерфейс для удобного взаимодействия с пользователем.

Все основные операции были протестированы и успешно выполнены. Программа корректно работает при различных тестах, включая рехеширование и изменение размера таблицы.

# 4 Ответы на вопросы

1. Хеширование — процесс преобразования данных произвольной длины в фиксированный хеш с помощью хеш-функций. Применяется в хранении данных (хеш-таблицы), криптографии, сетевых технологиях, поиске дубликатов и анализе больших данных.

2. Основные свойства хеш-функции: определённость, высокая скорость, односторонность, равномерность распределения, стойкость к коллизиям.

3. Алгоритмы хеширования основаны на арифметических и побитовых операциях, использовании простых чисел и комбинировании блоков данных.

4. Константная вычислительная сложность (O(1)) означает, что время выполнения операции не зависит от размера данных. Хеширование позволяет реализовать её за счёт прямого доступа к данным в хеш-таблице.

5. Коллизия — ситуация, когда разные данные имеют одинаковый хеш. Методы устранения: цепное хеширование, открытая адресация, двойное хеширование, рехеширование.

6. Цепное хеширование — метод разрешения коллизий, при котором элементы с одинаковым хешем хранятся в списке. Проблема: увеличение сложности операций из-за длинных цепочек при частых коллизиях.

7. Рехеширование — изменение размера хеш-таблицы и перераспределение элементов при переполнении или высоком коэффициенте заполнения.

8. Открытая адресация — способ разрешения коллизий, при котором поиск свободной ячейки осуществляется внутри самой таблицы.

9. Наиболее распространённые схемы последовательности проб: линейное, квадратичное пробирование и двойное хеширование.