****

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 6.1

**Тема: Быстрый доступ к данным с помощью хеш-таблиц**

Дисциплина: Структуры и алгоритмы обработки данных

Выполнил студент Королихин В.Н.

группа ИКБО-21-23

**Москва 2024**

**Цель работы:** освоить приёмы хеширования и эффективного поиска элементов множества.

ЗАДАНИЕ 1

**Формулировка задачи**

Разработайте приложение, которое использует хеш-таблицу (пары «ключ – хеш») для организации прямого доступа к элементам динамического множества полезных данных. Множество реализуйте на массиве, структура элементов (перечень полей) которого приведена в индивидуальном варианте (п.3).

**Математическая модель решения (описание алгоритма)**

1. Создадим структуру данных tableItem, которая будет представлять одну ячейку в хеш-таблице. Эта структура будет содержать 2 флага (taken – занята ли ячейка хеш-таблицы, removed – была ли удалена запись из этой ячейки) а также 3 поля “полезных данных” (key – ключ, он же номер банковского счёта, name – имя владельца счёта, adress – адрес проживания владельца счёта);
2. Создадим класс hashTable. Он будет иметь 3 приватных поля:

* TableItem\* items – массив ячеек хеш-таблицы,
* Int m – размер хеш-таблицы,
* Int n = 0 – количество записей в таблице.

А также 6 публичных методов:

1. void addItem(int key, string name, string adress) - метод добавления элемента в таблицу. Поиск свободной ячейки в хеш-таблице происходит методом линейного пробирования.
2. Создаём переменную int probe, которая будет содержать текущую пробу.
3. Пока ячейка по индексу probe занята, прибавляем к пробе 1.
4. Когда свободная ячейка найдена, в неё осуществляется запись.
5. Увеличиваем this->n на 1
6. Если коэффициент загрузки хеш-таблицы после этого стал больше 0.7, ты вызываем метод 2.4), осуществляя рехеширование;
7. tableItem\* findItem(int key) - метод поиска элемента в таблице по ключу.
8. Снова создаём int-переменную probe, а также attempts – счётчик попыток,
9. Пока ключ в ячейке по индексу probe не совпадает с искомым, а также количество попыток attempts не достигло размера хеш-таблицы, инкрементируем probe и attempts,
10. Если после цикла 2.2.2) искомый ключ в таблице найден и значение по нему не было удалено, то ссылка на эту запись возвращается функцией, иначе возвращается нулевой указатель;
11. void removeItem(int key) - метод удаления учейки из таблицы по ключу.
12. Снова создаём int-переменные probe и attempts,
13. Пока ключ в ячейке по индексу probe не совпадает с искомым, а также количество попыток attempts не достигло размера хеш-таблицы, инкрементируем probe и attempts,
14. Если после цикла 2.2.2) искомый ключ в таблице найден и значение по нему не было удалено, то флаг removed устанавливается на true, по сути удаляя элемент, иначе выводится сообщение об ошибке;
15. void expandTable() - метод расширения и рехеширования таблицы,
16. Создаём временный массив элементов tableItem,
17. Копируем в него содержимое исходной таблицы,
18. Удаляем из хеш-таблицы исходный массив,
19. Создаём в объекте хеш-таблицы новый массив размером 2 \* m (в 2 раза больше старого массива),
20. Обновляем значения m и n (2 \* m и 0 соотв.),
21. Перебираем все элементы копии исходного массива и поочередно заносим их в новый массив методом addItem;
22. void printTable() - метод вывода содержимого таблицы;
23. hashTable (int size) – конструктор.

**Код программы**

Код программ представлен на рис. 1 – 3.

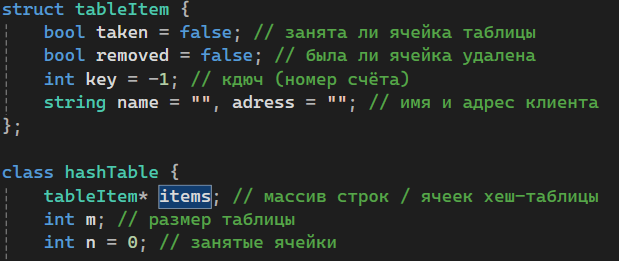


Рисунок 1 – Структура tableItem и класс hashTable

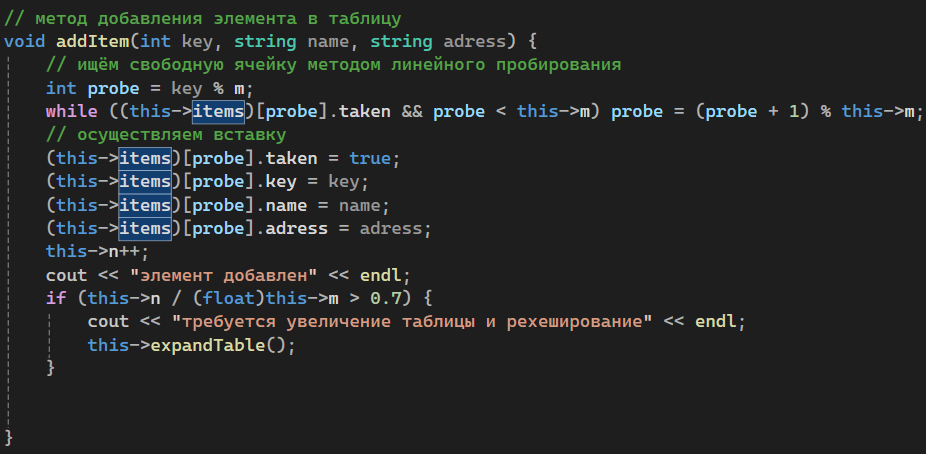


Рисунок 2 – Метод занесения записи в таблицу

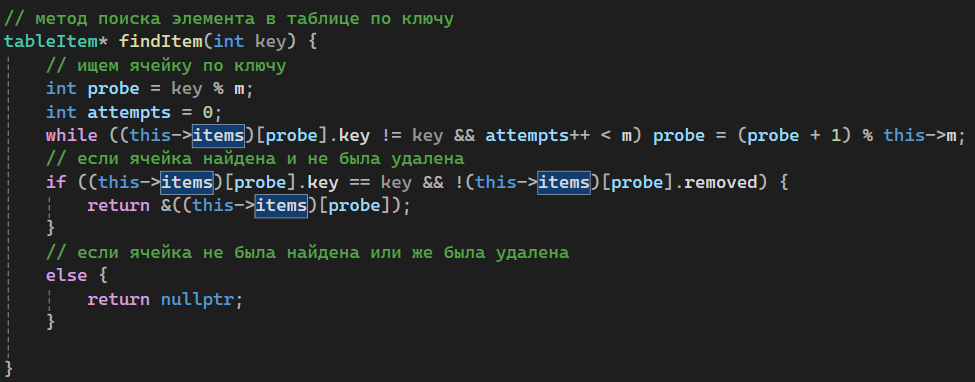


Рисунок 3 – Метод поиска записи в таблице по ключу

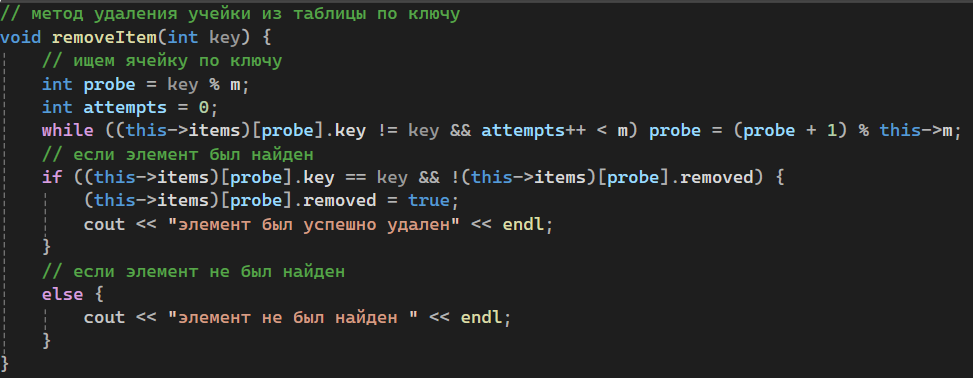


Рисунок 4 – Метод удаления записи в таблице по ключу

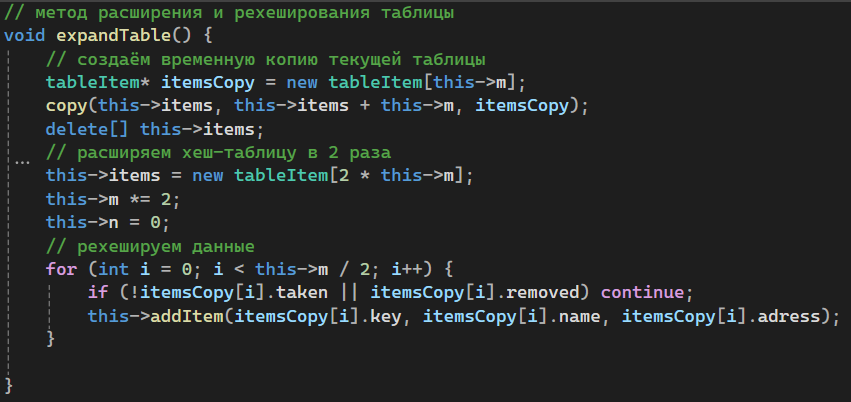


Рисунок 5 – Метод расширения и рехеширования таблицы

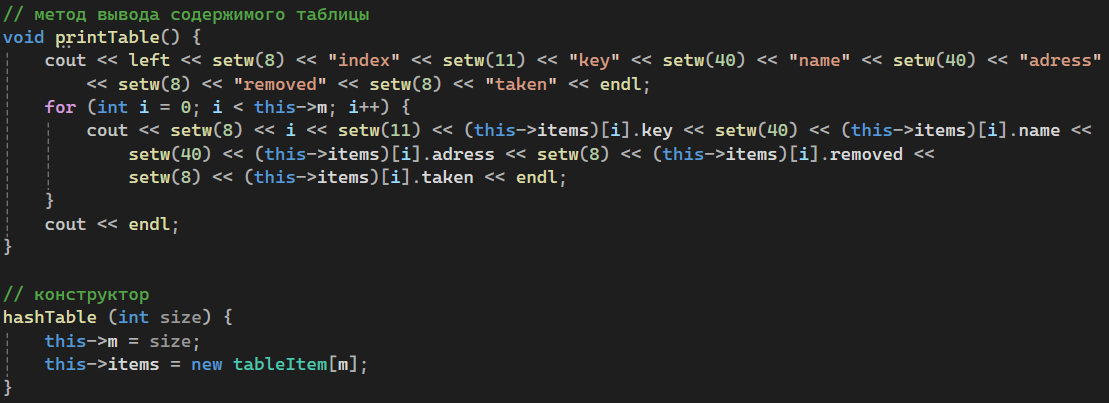


Рисунок 6 – Конструктор и методы вывода содержимого таблицы

**Результаты тестирования**

Результаты тестирования представлены на рис. 6-9

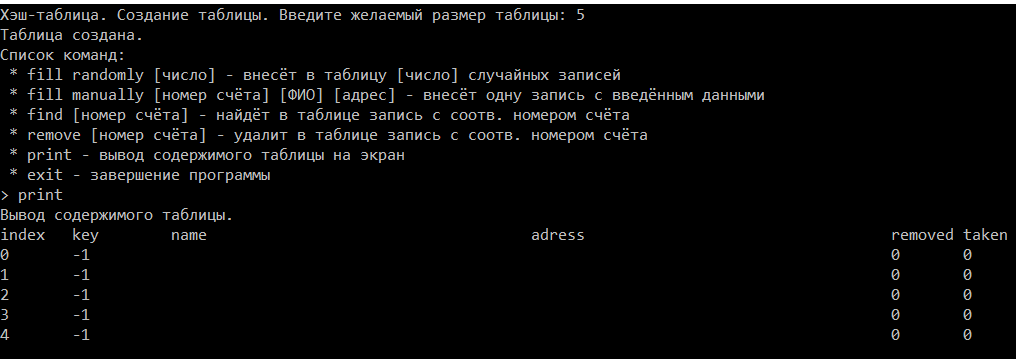


Рисунок 6 – Тестирование программы

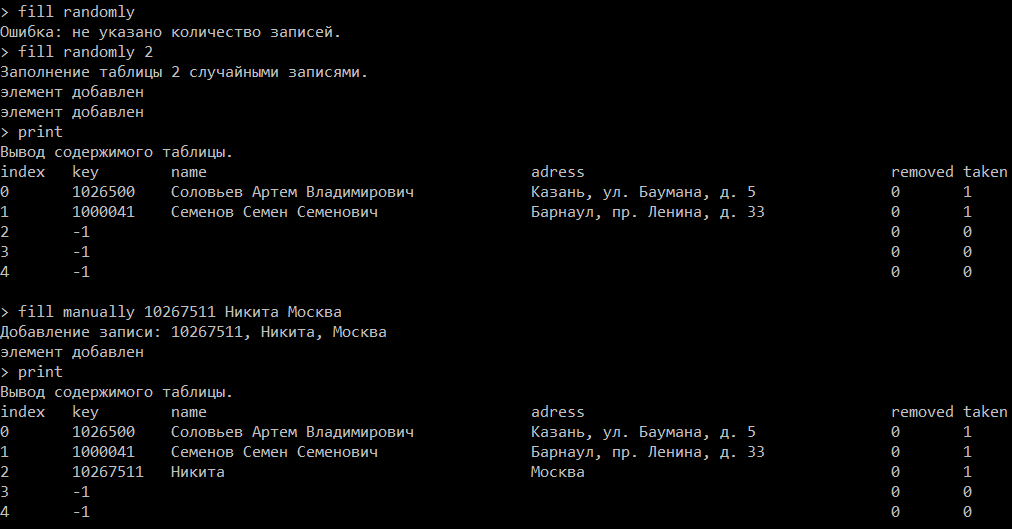


Рисунок 7 – Тестирование заполнения таблицы

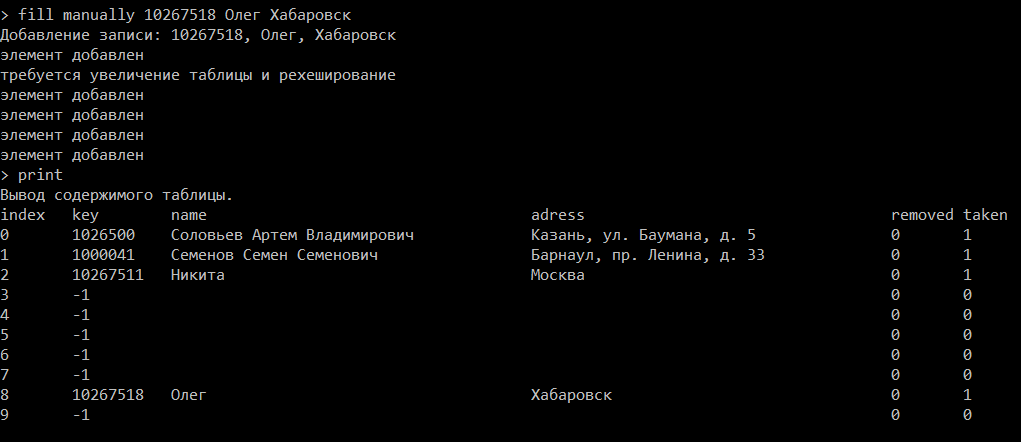


Рисунок 8 – Рехеширование

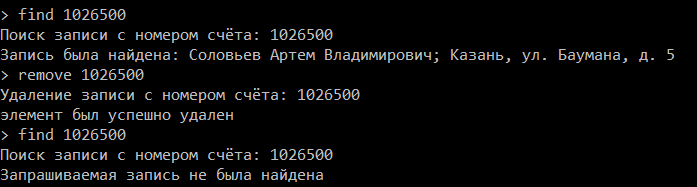


Рисунок 9 – Удаление и поиск

ВЫВОДЫ

В ходе работы были изучены хеш-таблица, их имплементация, возможные подводные камни и пути их обхода. Была спроектирована собственная хеш-таблица, а ее работа успешно протестирована.

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).