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 Фамилия И.О.
Группа ААAA-00-00

**Москва 2024**

# СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ.....3
1.1 Цель работы.....3
1.2 Индивидуальный вариант.....3
2 ХОД РАБОТЫ.....4
2.1 Анализ решения.....4
2.2 Код программы.....5
2.3 Код программы.....5
2.3 Результаты тестирования.....9
5 ВЫВОД.....10
6 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ.....11

# 1 ВВЕДЕНИЕ

## 1.1 Цель работы

Освоить приёмы хеширования и эффективного поиска элементов множества.

## 1.2 Индивидуальный вариант

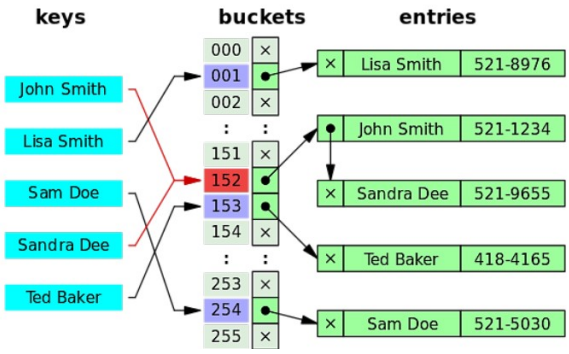
Таблица 1 — Индивидуальный вариант

| Вариант | Метод хеширования | Структура элементов множества |
|---|---|---|
| 19 | Цепное хеширование | Книга: ISBN – 12-значное число, автор, название |

# 2 ХОД РАБОТЫ

## 2.1 Анализ решения

В начале необходимо определить структуру таблицы. Поскольку используется цепное хеширование, будет необходима структура связного списка. Возьмём реализацию из 5-ой работы предыдущего семестра.




Создадим класс HashMap. Внутреннее множество данных будет представлять массив указателей на связные списки, то есть на их первые элементы.

Реализуем алгоритм хеш-функции, поскольку ключем является номер книги, то достаточно взять остаток от деления на размер массива.

Результатом выполнения хеш-функции будет служить индекс в массиве для вставки нового элемента, а также при обращении к ним.

Вставку, поиск, и удаление реализуем непосредственно обращением к соответствующим методам связного списка конкретного элемента массива.

Для реализации рехеширования, организуем подсчёт фактора загрузки, который представляет собой отношения количества элементов к размеру массива. Если фактор загрузки >0.75, то допустим рехеширование. Для этого, запишем все элементы массива во временный, увеличим размер исходного вдвое, и перезапишем все элементы вследствие получая новые хеши для записей.

Реализуем программный интерфейс для доступа к таблице пользователю.

Код программы представлен в листингах 1-3. Результаты тестирования на рис. 1-3.

## 2.2 Код программы

Листинг 1 - Реализация класса HashMap

```
class HashMap
{
private:
    LinkedList *table;
    int size;

    ul hash_func(ul key)
    {
        return key % size;
    }

public:
    HashMap(int _size)
    {
        size = _size;
        table = new LinkedList[_size];
    }

    void insert(ul ISBN, string author, string name)
    {
        table[hash_func(ISBN)].push_back(ISBN, author, name);
    }

    void delete_entry(ul ISBN)
    {
        table[hash_func(ISBN)].remove(ISBN);
    }

    Node get(ul ISBN)
    {
        Node *found = table[hash_func(ISBN)].find(ISBN);
        if (found == nullptr)
            return Node{0, "None", "None"};
        return *found;
    }

    // Get func
    Node operator[](ul ISBN)
    {
        return get(ISBN);
    }

    float load_factor()
    {
        int n = 0;
        for (int i = 0; i < size; i++)
        {
```

```
            n += table[i].count();
        }
        return (float)n / (float)size;
    }

    bool rehash_table()
    {
        if (load_factor() < 0.75)
            return false;
        vector<Node *> tmp_array;
        for (int i = 0; i < size; i++)
            while (!table[i].is_empty())
            {
                tmp_array.push_back(table[i].first);
                table[i].remove_first();
            }
        delete table;

        size *= 2;
        table = new LinkedList[size];
        for (int i = 0; i < tmp_array.size(); i++)
               insert(tmp_array[i]->ISBN, tmp_array[i]->author,
tmp_array[i]->name);
        for (int i = 0; i < tmp_array.size(); i++)
            delete tmp_array[i];

        return true;
    }

    string get_print() {
        string result = "";
        for (int i = 0; i < size; i++)
        {
            string r = table[i].get_print();
            if(r != "") {
                result += to_string(i) + " " + r;
            }
        }

        return result;
    }
};
```

Листинг 2 - Функция тестирования

```
void test()
{
    HashMap hashMap(5);

    hashMap.insert(123456789012, "John Doe", "Bang Bang");
    hashMap.insert(123456789013, "John ff", "Bang ss");
```

```
    hashMap.insert(123456789015, "John cc", "Bang bb");
    hashMap.insert(123459289015, "John bc", "Bang zz");
    hashMap.insert(123321389198, "Nom Jod", "Memes");

    cout << hashMap[123456789012].name << endl;
    cout << hashMap[819376789012].name << endl;

    hashMap.delete_entry(123456789012);
    hashMap.delete_entry(819376789012);

    cout << hashMap[123456789012].name << endl;
    cout << endl;
    hashMap.rehash_table();

    cout << hashMap[123456789015].name << endl;
    cout << hashMap[123456789012].name << endl;
    cout << hashMap[123459289015].name << endl;
    cout << hashMap[123321389198].name << endl;

    cout << hashMap.get_print();
}
```

Листинг 3 - Пользовательский интерфейс

```
int main()
{
    // test();

    HashMap hashMap(8);
    hashMap.insert(111111111111, "John Doe", "Bang Bang");
    hashMap.insert(222222222222, "Dohn Joe", "Gang Gang");
    hashMap.insert(333333333333, "Ohn Doj", "What");
    hashMap.insert(444444444444, "John Von", "C Language");
    hashMap.insert(555555555555, "Linus", "Linux");
    hashMap.insert(666666666666, "Windows", "Windows");
    hashMap.insert(777777777777, "Steve job", "Apple");



     cout << "Командный интерфейс для управления хеш таблицей"
<< endl;
      cout << "В таблицу размера 8 занесены предустановленные
записи с ключами:" << endl;
         cout  <<  "111111111111,  222222222222,  333333333333,
444444444444, 555555555555" << endl;
      cout << "Ключи представляют собой 12-ти значные номера
ISBN" << endl;
     cout << "С ними дополнительно хранятся автор и название
книги" << endl;

    cout << "Меню: 0 - Получить элемент; 1 - Вствить элемент; 2
- Удалить элемент;" << endl;
```

```
    cout << "3 - Произвести рехеширование (Если фактор загрузки
допускает); 4 - Вывод в консоль; 5 - Выход;" << endl;
    string command;
    while (true)
    {
        cout << "Введите команду: ";
        cin >> command;
        if (command == "0")
        {
            cout << "Введите номер ISBN: ";
            ul ISBN;
            cin >> ISBN;
            Node entry = hashMap[ISBN];
            cout << entry.ISBN << " " << entry.author << " " <<
entry.name << endl;
        }
        if (command == "1")
        {
             cout << "Введите значения через пробел ISBN автор
название: ";
            ul ISBN;
            string author, name;
            cin >> ISBN >> author >> name;
            hashMap.insert(ISBN, author, name);
        }
        else if (command == "2")
        {
            cout << "Введите номер ISBN: ";
            ul ISBN;
            cin >> ISBN;
            hashMap.delete_entry(ISBN);
        }
        else if (command == "3")
        {
            if (hashMap.rehash_table())
            {
                cout << "Рехеширование произведено" << endl;
            }
            else
            {
                  cout << "Рехеширование не произведено (Фактор
загрузки < 0.75)" << endl;
            }
        }
        else if (command == "4") {
            cout << hashMap.get_print() << endl;
        }
        else if (command == "5")
        {
            break;
        }
```

```
    }
    return 0;
}
```

### 2.3 Результаты тестирования

```
[rubicus@rubicus output]$ ./"main"
Bang Bang
None
None

Bang bb
None
Bang zz
Memes
```

Рисунок 1 - Тестирующая функция

```
[rubicus@rubicus output]$ ./"main"
Командный интерфейс для управления хеш таблицей
В таблицу размера 10 занесены предустановленные записи с ключами:
111111111111, 222222222222, 333333333333, 444444444444, 555555555555
Ключи представляют собой 12-ти значные номера ISBN
С ними дополнительно хранятся автор и название книги
Меню: 0 - Получить элемент; 1 - Вствить элемент; 2 - Удалить элемент;
3 - Произвести рехеширование (Если фактор загрузки допускает); 4 - Выход
Введите команду: 0
Введите номер ISBN: 111111111111
111111111111 John Doe Bang Bang
Введите команду: 3
Рехеширование не произведено (Фактор загрузки < 0.75)
Введите команду: 1
Введите значения через пробел ISBN автор название: 888877776666 Keila Madnv
Введите команду: 1
Введите значения через пробел ISBN автор название: 999933334444 Manvj Eoqjf
Введите команду: 1
Введите значения через пробел ISBN автор название: 828191924818 KEoqks Gnejkz
Введите команду: 0
Введите номер ISBN: 999933334444
999933334444 Manvj Eoqjf
Введите команду: 3
Рехеширование произведено
Введите команду: 4
[rubicus@rubicus output]$
```

Рисунок 2 - Пользовательский интерфейс

# 5 ВЫВОД

Были освоены приёмы по реализации и применения алгоритмов хеширования для создания хеш-таблиц с применением алгоритма цепного хеширования.

# 6 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Структуры и алгоритмы обработки данных (часть 2): Лекционные материалы / Рысин М. Л. МИРЭА — Российский технологический университет, 2022/23. – 82 с.