|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра инструментального и прикладного программного обеспечения | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 6.1** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Работа с данными из файла»** | |
|  | |
| Выполнил студент группы ИКБО-10-23 | Харитонов А.Н. |
| Принял преподаватель | Макеева О.В. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_ \_ \_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2024

# **Цель работы**

Освоить приемы хэширования и эффективного поиска элементов множества.

# **Ход работы**

Для решения поставленного задания была написана структура Subscription, которая хранит в себе ID, Имя и адрес человека.

*struct* Subscription {

*int* id;

    string name;

    string address;

    Subscription(*int* *id*, string *name*, string *address*)

    : id(*id*), name(*name*), address(*address*) {}

}

Приложение 1. Код структуры Subscription

Также был написан класс HashTable, у которого есть вектор table, len – кол-во элементов в таблице, capacity – максимальное количество элементов, а также коэффициент для хэш функции.

 vector<list<Subscription>> table;

*int* capacity;

*int* len;

    const *float* A = (sqrt(5) - 1) / 2;

Приложение 2. Приватные поля класса

Для хэширования была написана функция hashFunction, а также rehash для рехэширования.

*int* hashFunction(*int* *key*){

*float* fractional\_part = (*key* \* A) - *int*(*key* \* A);

        return *int*(capacity \* fractional\_part);

    }

*void* rehash(){

*int* oldCapacity = capacity;

        capacity \*= 2;

        vector<list<Subscription>> newTable(capacity);

         for (*int* i = 0; i < oldCapacity; i++) {

            for (*auto*& sub : table[i]) {

*int* newIndex = hashFunction(sub.id);

                newTable[newIndex].push\_back(sub);

            }

        }

        table = move(newTable);

    }

Приложение 3. Приватные методы класса

Пользователю доступны следующие методы:

- Insert (вставка)

- Search (поиск по id)

- Remove (удаление по id)

- Display (вывод содержимого)

*void* insert(const Subscription& *sub*){

        if (len > capacity \* 0.75){

            rehash();

        }

*int* index = hashFunction(*sub*.id);

        table[index].push\_back(*sub*);

        len++;

    }

    Subscription\* search(*int* *key*){

*int* index = hashFunction(*key*);

        for (*auto*& sub : table[index]) {

            if (sub.id == *key*) {

                return &sub;

            }

        }

        return nullptr;

    }

*void* remove(*int* *key*) {

*int* index = hashFunction(*key*);

        for (*auto* it = table[index].begin(); it != table[index].end(); ++it) {

            if (it->id == *key*) {

                table[index].erase(it);

                len--;

                cout << "Subscription with ID " << *key* << " removed." << endl;

                return;

            }

        }

        cout << "Subscription with ID " << *key* << " not found." << endl;

    }

*void* display() {

        for (*int* i = 0; i < capacity; i++) {

            cout << "Index " << i << ": ";

            for (*auto*& sub : table[i]) {

                cout << "[ID: " << sub.id << ", Name: " << sub.name << ", Address: " << sub.address << "] -> ";

            }

            cout << "nullptr" << endl;

        }

    }

Приложение 4. Публичные методы класса

Для тестирования создадим объект класса и присвоим ему capacity = 7.

Заполним нашу таблицу на 5 значений и выведем содержимое, после чего попытаемся найти и удалить запись.

 HashTable hashTable(7);

    hashTable.insert(Subscription(10001, "John Doe", "123 Elm St"));

    hashTable.insert(Subscription(10002, "Jane Smith", "456 Oak St"));

    hashTable.insert(Subscription(10003, "Alice Johnson", "789 Pine St"));

    hashTable.insert(Subscription(10004, "Bob Brown", "101 Maple St"));

    hashTable.insert(Subscription(10005, "Eve White", "202 Birch St"));

    cout << "Initial hash table state:\n";

    hashTable.display();

*int* searchId = 10003;

    Subscription\* sub = hashTable.search(searchId);

    if (sub) {

        cout << "\nFound subscription with ID " << searchId << ": " << sub->name << ", " << sub->address << endl;

    } else {

        cout << "\nSubscription with ID " << searchId << " not found." << endl;

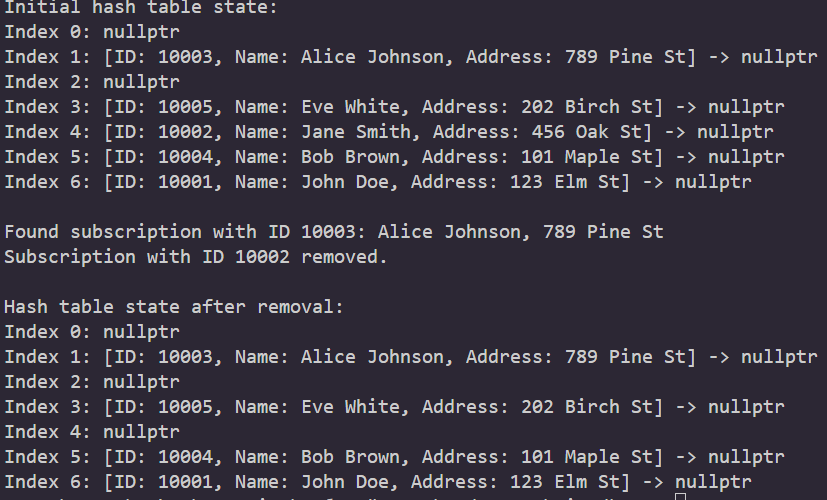
    }

    hashTable.remove(10002);

    cout << "\nHash table state after removal:\n";

    hashTable.display();

Приложение 5. Тестирование программы



Изображение 1. Вывод программы

# **Вывод**

В ходе работы я освоил приемы хэширования и эффективного поиска элементов множетсва.