Изображение выглядит как символ, эмблема, герб, нашивка

Контент, сгенерированный ИИ, может содержать ошибки.

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:** АЛГОРИТМЫ ПОИСКА

Дисциплина: Структуры и алгоритмы обработки данных

Выполнил студент Жижикин Л.С.

группа ИКБО-50-24

**Москва 2025**

Содержание

[Постановка задачи. 2](#_Toc209801978)

[Ход работы 3](#_Toc209801979)

[lib.hpp 3](#_Toc209801980)

[Subscription.hpp 4](#_Toc209801981)

[HashTable.hpp 5](#_Toc209801982)

[Структура HashTable 5](#_Toc209801983)

[Метод рехэширования 5](#_Toc209801984)

[Методы вставки 6](#_Toc209801985)

[Метод поиска по ключу 7](#_Toc209801986)

[Метод удаления по ключу 7](#_Toc209801987)

[Методы печати 7](#_Toc209801988)

[main.cpp 8](#_Toc209801989)

[Функция main 8](#_Toc209801990)

[Функция тестирования 8](#_Toc209801991)

[Функция меню выбора действия 9](#_Toc209801992)

[Функция пользовательского интерфейса коммандной строки 9](#_Toc209801993)

[Результат работы функции тестирования 11](#_Toc209801994)

[Примеры отработки программы с пользовательским интерфейсом 13](#_Toc209801995)

[Вывод 15](#_Toc209801996)

[Список литературы 15](#_Toc209801997)

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

**Задание:** Разработать приложение, которое использует хеш-таблицу (пары «ключ – хеш») для организации прямого доступа к элементам динамического множества полезных данных

# Постановка задачи.

Открытая адресация (квадратичное пробирование)

Читательский абонемент: номер читательского - целое пятизначное число, ФИО, адрес

# Ход работы

## lib.hpp

|  |
| --- |
| #include *<cstring>*  #include *<cstdlib>*  #include *<iostream>*  #include *<vector>*  #include *<fstream>*  #include *<ios>*  #include *<string>*  #include *<random>*  #include *<chrono>*  #include *<array>*  #include *<algorithm>*  #include *<queue>*  #include *<termios.h>*  #include *<unistd.h>*  **typedef** unsigned u;  **typedef** std::uint16\_t ui16;  **typedef** std::uint32\_t ui32;  **typedef** std::uint64\_t ui64;  **typedef** std::string string; |

## 

## Subscription.hpp

|  |
| --- |
| 1 *#include* *"lib.hpp"*  2  3 struct Subscription {  4 ui32 number;  5 string full\_name;  6 string address;  7 bool valid;  8  9 static Subscription generate\_random() {  10 const std::string characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";  11 std::random\_device rd; std::mt19937\_64 gk(rd());  12 std::uniform\_int\_distribution<int> num\_gen(10000, 99999);  13 std::uniform\_int\_distribution<size\_t> char\_gen(0, characters.size()-1);  14 *// gen name*  15 string name = "", address = "";  16 name.push\_back(characters[char\_gen(gk) % 26]);  17 name.push\_back('.');  18 name.push\_back(characters[char\_gen(gk) % 26]);  19 name.push\_back('.');  20 name.push\_back(characters[char\_gen(gk) % 26]);  21 for (int i = 0; i < 7 + rand() % 10; ++i)  22 name += characters[26 + char\_gen(gk) % 26];  23 *// gen address*  24 for (int i = 0; i < 10 + rand() % 20; ++i)  25 address += characters[char\_gen(gk)];  26  27 return Subscription(num\_gen(gk), name, address);  28 }  29  30 Subscription() { this->valid = false; }  31  32 Subscription(ui32 number, string full\_name, string address) {  33 this->number = number;  34 this->full\_name = full\_name;  35 this->address = address;  36 this->valid = true;  37 if ((number < 10000) || (number > 99999) || (full\_name == "") || (address == ""))  38 this->valid = false;  39 }  40  41 void print() {  42 std::cout << "#" << number  43 << " registered to " << full\_name  44 << ", " << address << "\n";  45 }  46 }; |

## HashTable.hpp

### Структура HashTable

|  |
| --- |
| 1 struct HashTable {  2 ui32 c, d;  3 ui32 capacity;  4 ui32 num\_of\_elements = 0;  5 std::vector<Subscription> table;  6 *// iteration trait*  7 auto begin() const { return table.begin(); }  8 auto end() const { return table.end(); }  9  10 HashTable() {  11 capacity = 50;  12 *// make c and d coprime with capacity*  13 c = capacity - 1;  14 d = capacity + 1;  15 table = std::vector(capacity, Subscription());  16 }  17 HashTable(ui32 init\_len) {  18 capacity = init\_len \* 2;  19 table = std::vector(capacity, Subscription());  20 c = capacity - 1;  21 d = capacity + 1;  22 while(num\_of\_elements < init\_len)  23 this->insert(Subscription::generate\_random(), false);  24 }  25  26  27 ui32 hashFunction(ui32 number) {  28 return number % capacity;  29 }  30 } |

### Метод рехэширования

|  |
| --- |
| 1 void rehash() {  2 capacity \*= 2;  3 c = capacity - 1;  4 d = capacity + 1;  5 std::vector<Subscription> new\_table(capacity, Subscription());  6  7 ui32 hash\_key;  8 for (auto entry : table) {  9 hash\_key = hashFunction(entry.number);  10 if (new\_table[hash\_key].valid) {  11 int i = 0;  12 while (new\_table[hash\_key].valid)  13 hash\_key = hashFunction(hash\_key + c \* ++i + d \* i\*i);  14 }  15 new\_table[hash\_key] = entry;  16 }  17 table = move(new\_table);  18 std::cout << "Rehashed successfully.\nNew capacity: " << this->table.size() << "\n\n" ;  19 } |

### 

### Методы вставки

|  |
| --- |
| 1 void insert(Subscription hashed\_entry, bool not\_initializing = true) {  2 ui32 hash\_key = hashFunction(hashed\_entry.number);  3  4 if (!hashed\_entry.valid) {  5 std::cout << "Tried to hash invalid subscription:\n";  6 hashed\_entry.print();  7 std::cout << "\n";  8 return;  9 }  10  11 if (++num\_of\_elements > capacity \* 0.7) {  12 std::cout << "Element limit exceeded. Rehashing.\n";  13 rehash();  14  15 }  16  17 if (table[hash\_key].valid) {  18 int i = 0;  19 if (not\_initializing) {  20 std::cout << "Trying to prevent hash collision at " << hash\_key << std::endl;  21 while (table[hash\_key].valid)  22 hash\_key = hashFunction(hash\_key + c \* ++i + d \* i\*i);  23 std::cout << "Found free key " << hash\_key << "\n\n";  24 }  25 else  26 while (table[hash\_key].valid)  27 hash\_key = hashFunction(hash\_key + c \* ++i + d \* i\*i);  28 }  29  30 table[hash\_key] = hashed\_entry;  31 if (not\_initializing) {  32 hashed\_entry.print();  33 std::cout << "Successfully inserted at " << hash\_key << "\n\n";  34 }  35 }  36 void insert(ui32 number, string full\_name, string address, bool not\_initializing = true) {  37 ui32 hash\_key = hashFunction(number);  38 Subscription hashed\_entry = Subscription(number, full\_name, address);  39 this->insert(hashed\_entry, not\_initializing);  40 } |

### 

### Метод поиска по ключу

|  |
| --- |
| 1 Subscription find(ui32 key) {  2 ui32 hash\_key = hashFunction(key);  3 Subscription found = table[hash\_key];  4 if (found.valid && found.number == key)  5 return found;  6  7 int i = 0;  8 while (i < 100) {  9 hash\_key = hashFunction(hash\_key + c \* ++i + d \* i\*i);  10 found = table[hash\_key];  11 if ((found.valid) && (found.number == key))  12 return found;  13 }  14  15 std::cout << "Can't find entry with this key\n";  16 return Subscription();  17 } |

### Метод удаления по ключу

|  |
| --- |
| 1 void del(ui32 key) {  2 ui32 hash\_key = hashFunction(key);  3 Subscription found = table[hash\_key];  4 if (found.valid && found.number == key) {  5 table[hash\_key].print();  6 std::cout << "Successfully deleted from " << hash\_key << "\n\n";  7 table[hash\_key] = Subscription();  8 num\_of\_elements--;  9 return;  10 }  11 int i = 0;  12 while (i < 100) {  13 hash\_key = hashFunction(hash\_key + c \* ++i + d \* i\*i);  14 found = table[hash\_key];  15 if ((found.valid) && (found.number == key)) {  16 table[hash\_key].print();  17 std::cout << "Successfully deleted from " << hash\_key << "\n\n";  18 table[hash\_key] = Subscription();  19 num\_of\_elements--;  20 return;  21 }  22 }  23  24 std::cout << "Can't find entry with this key\n";  25 } |

### 

### Методы печати

|  |
| --- |
| 1 void print() {  2 for(auto entry : \*this)  3 if (entry.valid)  4 entry.print();  5 }  6 void print\_full() {  7 for (int i = 0; i < table.size(); ++i) {  8 Subscription entry = this->table[i];  9 if (entry.valid)  10 std::cout  11 <<"Hash\_key " << i << ": "  12 << "#" << entry.number  13 << " registered to " << entry.full\_name  14 << ", " << entry.address << "\n";  15 }  16 } |

## main.cpp

### Функция main

|  |
| --- |
| 1 *#include* *"lib.hpp"*  2 *#include* *"HashTable.hpp"*  3  4 char user\_selection\_menu();  5 void cli();  6  7 void test();  8  9 int main() {  10 *// cli();*  11 *// test();*  12 } |

### 

### Функция тестирования

|  |
| --- |
| 1 void test() {  2 HashTable table(3);  3 table.print\_full();  4 std::cout << string(30,'-') << std::endl;  5 for (auto entry: table.table) {  6 if (!entry.valid)  7 continue;  8 table.del(entry.number);  9 }  10 std::cout << string(30,'-') << std::endl;  11 table.print\_full();  12 std::cout << string(30,'-') << std::endl;  13 table.insert(78208, "B.B.Bbbbb", "8fj54Qj060");  14 table.insert(42312, "C.C.Ashdghas", "128hd382s");  15 table.insert(10001, "B.B.Bbbbobby", "74328kfds");  16 table.insert(10001, "T.D.Deriuaasdsa", "534634634");  17 table.insert(99999, "V.G.D.Ajdfsjkakisdfjaksf", "a");  18 table.insert(32154, "john", "4104d32m4-2134d02d32d9023\_2934\*1230-f34832");  19 std::cout << string(30,'-') << std::endl;  20 table.print\_full();  21 std::cout << string(30,'-') << std::endl;  22 table.find(32154).print();  23 table.find(99999).print();  24 table.find(10001).print();  25 std::cout << string(30,'-') << std::endl;  26 } |

### Функция меню выбора действия

|  |
| --- |
| 1 char user\_selection\_menu() {  2 std::cout  3 << "\n0. Exit program \t" << "1. Insert element \n"  4 << "2. Delete element \t" << "3. Find element \n"  5 << "4. Print HashTable \t" << "5. Print HashTable with hash info \n"  6 << "Choose what you want to do:\t" ;  7 char input;  8 std::cin >> input;  9 std::cout << "\n";  10 if('0' <= input && input <= '5')  11 return input;  12 else  13 std::cout << "No such option \n\n";  14 return user\_selection\_menu();  15 } |

### 

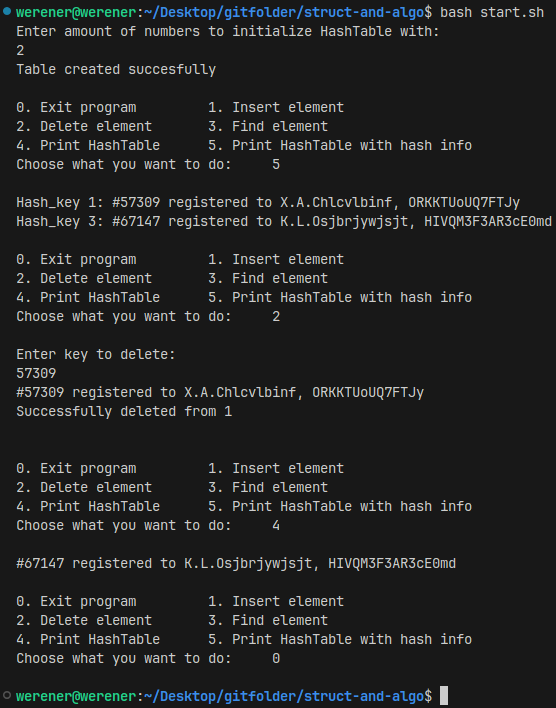
### Функция пользовательского интерфейса коммандной строки

|  |
| --- |
| 1 void cli() {  2 ui32 init\_len;  3 std::cout << "Enter amount of numbers to initialize HashTable with:\n";  4 if(!(std::cin >> init\_len))  5 return;  6  7 HashTable table(init\_len);  8 std::cout << "Table created succesfully\n";  9  10 while (true) {  11 Subscription found;  12 ui32 key;  13 string name, address;  14 char selected\_option = user\_selection\_menu();  15 switch (selected\_option) {  16 case '0':  17 return;  18 case '1':  19 std::cout << "Enter key, name of owner and address (separated by spaces) to insert: \n";  20 std::cin >> key >> name >> address;  21 table.insert(key, name, address);  22 break;  23 case '2':  24 std::cout << "Enter key to delete: \n";  25 std::cin >> key;  26 table.del(key);  27 break;  28 case '3':  29 std::cout << "Enter key to find: \n";  30 std::cin >> key;  31 found = table.find(key);  32 if (!found.valid)  33 break;  34 std::cout << "Found entry: \n";  35 found.print();  36 break;  37 case '4':  38 table.print();  39 break;  40 case '5':  41 table.print\_full();  42 break;  43 }  44 }  45 } |

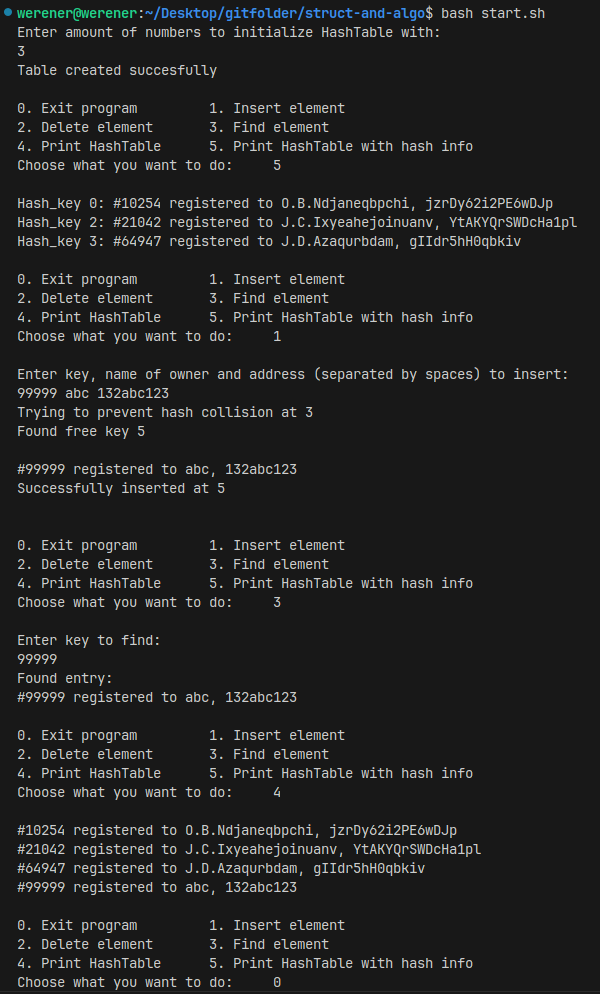
### Результат работы функции тестированияA screenshot of a computer program AI-generated content may be incorrect.

### Примеры отработки программы с пользовательским интерфейсом

Удаление, вывод и выход из программы:



Вставка и поиск:



# Вывод

В ходе данной работы я освоил приёмы хеширования и эффективного поиска элементов множества.

# Список литературы

1. Документация по языку С++ [Электронный ресурс]. URL: https://cppreference. com/ (дата обращения: 11.09.2025).

2. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL: https://online-edu.mirea.ru/course/view.php?id=4020 (дата обращения: 11.09.2025).

3. Керниган Б., Ритчи Д. Язык программирования Си / Керниган Б., Ритчи Д., 3- е изд., 1988.

4. Страуструп Б. Программирование. Принципы и практика с использованием C+ + / Страуструп Б., 2-е изд., 2016.