

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**"МИРЭА - Российский технологический университет"**

РТУ МИРЭА

Институт информационных технологий (ИИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ**  
**ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 6.1**

**«Быстрый доступ к данным с помощью хеш-таблиц»**

**по дисциплине**

**«**Структуры и алгоритмы обработки данных**»**

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-11-24 |  | Авраменко А.А. |
|  |  |  |
| Принял доцент каф. МОСИТ |  | Сартаков М.В. |

Практическую работу выполнил «\_\_»\_\_\_\_\_\_\_2025 г.

«Зачтено» «\_\_»\_\_\_\_\_\_\_2025 г.

Москва 2025

**Практическое задание:**

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).

Приложение должно содержать класс с базовыми операциями: вставки, удаления, поиска по ключу, вывода. Включите в класс массив полезных данных и хеш таблицу. Хеш функцию подберите самостоятельно, используя правила выбора функции.

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

Предусмотрите автоматическое заполнение таблицы 5-7 записями.

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

Проведите полное тестирование программы (все базовые операции, изменение размера и рехеширование), тест примеры определите самостоятельно. Результаты тестирования включите в отчет по выполненной работе.

**Решение:**

1. Модель данных и обозначения

Пусть U — универсум ключей (пятизначные целые числа K ∈ [10 000; 99 999]). Хранимое множество записей: S = { (K, FIO, Addr) }.

Хеш-таблица представлена массивом корзин T длины m (число ячеек/слотов). Каждая корзина T[i] — связный список записей, чьи ключи дали одинаковый хеш (цепное хеширование). В итоге структура — «массив списков».

1. Хеш-функция

Хеш-функция определяется как: h(K) = K mod m, где K — ключ, m — размер таблицы.

1. Разрешение коллизий

Коллизии разрешаются методом цепочек: все элементы с одинаковым значением хеша записываются в список в соответствующей корзине.

1. Базовые операции
   1. Поиск (Find): вычисляется индекс i = h(K), далее в списке T[i] ищется запись с ключом K.
   2. Вставка (Insert): вычисляется индекс i = h(K), проверяется отсутствие дубликата в списке T[i], затем запись добавляется в конец списка.
   3. Удаление (Remove): вычисляется индекс i = h(K), из списка T[i] удаляется запись с ключом K (если найдена).
2. Коэффициент загрузки и рехеширование

Коэффициент загрузки таблицы вычисляется по формуле: α = n / m, где n — число элементов, m — размер таблицы. Если α ≥ 0.75, выполняется рехеширование: создаётся новый массив корзин удвоенного размера, и все элементы перезаписываются в новые корзины с пересчётом хеша.

1. Сложность операций

В среднем: поиск, вставка и удаление выполняются за O(1).

В худшем случае (все элементы в одной корзине): за O(n).

Рехеширование выполняется за O(n), но редко, поэтому амортизированная сложность операций остаётся O(1).

1. Реализация:

Программа реализована на C++ и состоит из класса Reader для хранения данных и класса HashTable для организации хеш-таблицы с операциями вставки, удаления, поиска и вывода. В main.cpp добавлен простой текстовый интерфейс для работы пользователя.

Листинг 1.1 – файл Reader.h

#ifndef READER\_H  
#define READER\_H  
#include <string>  
using namespace std;  
  
  
class Reader {  
 int ticketNumber;  
 string fio;  
 string address;  
  
public:  
 Reader();  
  
 Reader(int ticket, string &f, string &a);  
  
 int getTicketNumber() const;  
  
 void setTicketNumber(int ticketNumber);

Продолжение листинга 1.1

string getFio();  
  
 void setFio(string fio);  
  
 string getAddress();  
  
 void setAdress(string adress);  
};  
  
  
#endif *//READER\_H*

Листинг 1.2 – файл Reader.cpp

#include "Reader.h"  
  
Reader::Reader(){}  
  
Reader::Reader(int ticket, string &f, string &a) : ticketNumber(ticket), fio(f), address(a) {}  
  
int Reader::getTicketNumber() const {  
 return ticketNumber;  
}  
  
void Reader::setTicketNumber(int ticketNumber) {  
 this->ticketNumber = ticketNumber;  
}  
  
string Reader::getFio() {  
 return fio;  
}  
  
void Reader::setFio(string fio) {  
 this->fio = fio;  
}  
  
string Reader::getAddress() {  
 return address;  
}  
  
void Reader::setAdress(string address) {  
 this->address = address;  
}

Листинг 1.3 – файл Hashtable.h

#ifndef HASHTABLE\_H

#define HASHTABLE\_H

#include <vector>

#include <list>

#include "Reader.h"

using namespace std;

class HashTable {

Продолжение листинга 1.3

vector<list<Reader> > table;

size\_t tableSize = table.size();

int count;

int hash(int key, size\_t size) const;

public:

HashTable(int initialSize);

bool insert(const Reader &reader);

bool remove(int key);

bool find(int key, Reader &out);

void display();

void rehash();

void fillDemo();

};

#endif //HASHTABLE\_H

Листинг 1.4 – файл Hashtable.cpp

#include "HashTable.h"

#include <iostream>

#include <random>

HashTable::HashTable(int initialSize) : tableSize(initialSize), count(0) {

table.resize(tableSize);

};

int HashTable::hash(int key, size\_t size) const {

return key % size;

}

bool HashTable::insert(const Reader &reader) {

Reader tmp;

if (static\_cast<float>(count) / tableSize >= 0.75f) {

rehash();

}

if (find(reader.getTicketNumber(), tmp)) {

return false;

}

int index = hash(reader.getTicketNumber(), tableSize);

table[index].push\_back(reader);

count++;

return true;

}

Продолжение листинга 1.4

bool HashTable::remove(int key) {

auto &l = table[hash(key, tableSize)];

if (l.empty()) {

return false;

}

for (auto it = l.begin(); it != l.end(); it++) {

if (it->getTicketNumber() == key) {

l.erase(it);

return true;

}

}

return false;

}

bool HashTable::find(int key, Reader &out) {

if (table.empty()) {

return false;

}

auto &l = table[hash(key, tableSize)];

if (l.empty()) {

return false;

}

for (auto it = l.begin(); it != l.end(); it++) {

if (it->getTicketNumber() == key) {

out = \*it;

return true;

}

}

return false;

}

void HashTable::rehash() {

size\_t newTableSize = tableSize \* 2;

vector<list<Reader> > newTable(newTableSize);

for (int i = 0; i < tableSize; i++) {

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

int newHash = hash(it->getTicketNumber(), newTableSize);

newTable[newHash].push\_back(\*it);

}

}

table = newTable;

tableSize = newTableSize;

}

void HashTable::display() {

for (int i = 0; i < tableSize; i++) {

if (table[i].empty()) {

continue;

}

Продолжение листинга 1.4

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

std::cout << it->getTicketNumber() << ":" << it->getFio() << ":" << it->getAddress() << std::endl;

}

}

}

void HashTable::fillDemo() {

std::random\_device rd;

std::mt19937 gen(rd());

std::uniform\_int\_distribution<int> dist(10000, 99999);

std::vector<std::string> names = {

"Иванов Иван Иванович",

"Петров Петр Петрович",

"Сидорова Анна Сергеевна",

"Кузнецов Дмитрий Андреевич",

"Смирнова Ольга Владимировна",

"Федоров Николай Павлович"

};

std::vector<std::string> addresses = {

"ул. Ленина, д. 10",

"ул. Пушкина, д. 25",

"пр. Гагарина, д. 3",

"ул. Советская, д. 42",

"пер. Школьный, д. 7",

"ул. Мира, д. 15"

};

for (int i = 0; i < names.size(); i++) {

int ticket = dist(gen);

string fio = names[rand() % names.size()];

string address = addresses[rand() % addresses.size()];

Reader r(ticket, fio, address);

insert(r);

}

}

Листинг 1.5 – файл main.cpp

#include "Reader.h"

#include "HashTable.h"

#include <iostream>

#include <vector>

#include <string>

#include <random>

#include <limits>

using namespace std;

Продолжение листинга 1.5

static void printMenu() {

cout << "\n================ Хеш-таблица (цепное хеширование) ================\n";

cout << "1) Добавить читателя\n";

cout << "2) Удалить читателя по номеру\n";

cout << "3) Найти читателя по номеру\n";

cout << "4) Вывести таблицу\n";

cout << "5) Автозаполнение 5–7 записями (рандом)\n";

cout << "0) Выход\n";

cout << "-------------------------------------------------------------------\n";

cout << "Выберите пункт меню: ";

}

static int promptInt(const string &msg) {

cout << msg;

int x{};

while (!(cin >> x)) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Некорректный ввод. Повторите: ";

}

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

return x;

}

static string promptLine(const string &msg) {

cout << msg;

string s;

getline(cin, s);

return s;

}

int main() {

HashTable table(10);

while (true) {

printMenu();

int choice{};

if (!(cin >> choice)) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Введите номер пункта меню.\n";

continue;

}

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

if (choice == 0) {

cout << "Выход. Спасибо!\n";

break;

}

Продолжение листинга 1.5

switch (choice) {

case 1: {

int ticket = promptInt("Введите номер читательского билета (5 цифр): ");

string fio = promptLine("Введите ФИО: ");

string addr = promptLine("Введите адрес: ");

Reader r(ticket, fio, addr);

if (table.insert(r)) {

cout << "Добавлено: [" << ticket << "] " << fio << ", " << addr << "\n";

} else {

cout << "Запись с таким номером уже существует.\n";

}

break;

}

case 2: {

int ticket = promptInt("Введите номер для удаления: ");

if (table.remove(ticket)) {

cout << "Удалено: [" << ticket << "]\n";

} else {

cout << "Элемент с номером " << ticket << " не найден.\n";

}

break;

}

case 3: {

int ticket = promptInt("Введите номер для поиска: ");

Reader found;

if (table.find(ticket, found)) {

cout << "Найдено: [" << found.getTicketNumber() << "] "

<< found.getFio() << ", " << found.getAddress() << "\n";

} else {

cout << "Не найдено.\n";

}

break;

}

case 4: {

cout << "Содержимое таблицы:\n";

table.display();

break;

}

case 5: {

table.fillDemo();

cout << "Готово. Таблица пополнена.\n";

break;

}

default:

cout << "Нет такого пункта меню. Повторите выбор.\n";

break;

}

}

return 0;

}

1. Тестирование

Листинг 1.6 – тестирование (ввод в консоль)

"/Users/artem/Desktop/Универ/2 курс/3 сем/СИАОД/siaod-rtu-mirea/cmake-build-debug/practice6/Hash/hash\_chain"

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 1

Введите номер читательского билета (5 цифр): 12345

Введите ФИО: ААА

Введите адрес: Москва

Добавлено: [12345] ААА, Москва

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 1

Введите номер читательского билета (5 цифр): 67895

Введите ФИО: БББ

Введите адрес: Москва

Добавлено: [67895] БББ, Москва

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 1

Введите номер читательского билета (5 цифр): 12345

Введите ФИО: ВВВ

Введите адрес: Мытищи

Запись с таким номером уже существует.

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 3

Введите номер для поиска: 12345

Найдено: [12345] ААА, Москва

Продолжение листинга 1.6

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 3

Введите номер для поиска: 67895

Найдено: [67895] БББ, Москва

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 3

Введите номер для поиска: 55555

Не найдено.

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 2

Введите номер для удаления: 55555

Элемент с номером 55555 не найден.

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 2

Введите номер для удаления: 67895

Удалено: [67895]

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 3

Введите номер для поиска: 67895

Не найдено.

Продолжение листинга 1.6

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 2

Введите номер для удаления: 67895

Элемент с номером 67895 не найден.

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 2

Введите номер для удаления: 12345

Удалено: [12345]

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 2

Введите номер для удаления: 10015

Элемент с номером 10015 не найден.

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 4

Содержимое таблицы:

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 4

Содержимое таблицы:

Продолжение листинга 1.6

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 5

Готово. Таблица пополнена.

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 4

Содержимое таблицы:

26551:Иванов Иван Иванович:ул. Мира, д. 15

14182:Федоров Николай Павлович:ул. Пушкина, д. 25

21973:Смирнова Ольга Владимировна:пр. Гагарина, д. 3

91875:Петров Петр Петрович:ул. Пушкина, д. 25

41005:Иванов Иван Иванович:пр. Гагарина, д. 3

35576:Федоров Николай Павлович:пр. Гагарина, д. 3

================ Хеш-таблица (цепное хеширование) ================

1) Добавить читателя

2) Удалить читателя по номеру

3) Найти читателя по номеру

4) Вывести таблицу

5) Автозаполнение 5–7 записями (рандом)

0) Выход

-------------------------------------------------------------------

Выберите пункт меню: 0

Выход. Спасибо!

Process finished with exit code 0

В ходе тестирования были добавлены записи с номерами 12345 и 67895; повторная вставка записи с уже существующим номером корректно отклонена. Поиск по ключам возвращал ожидаемые результаты: существующие записи находились, несуществующие – выдавали сообщение об отсутствии. Удаление выполнялось корректно: записи удалялись по ключу, при повторных или ошибочных обращениях система сообщала об их отсутствии. Автозаполнение успешно добавило случайные записи, содержимое таблицы корректно выводилось, завершение работы прошло без ошибок.

**Вывод**

В ходе работы были изучены принципы построения хеш-таблиц, методы разрешения коллизий и рехеширования. Реализована программа с базовыми операциями вставки, удаления, поиска и вывода элементов. Тестирование подтвердило корректность работы разработанной структуры данных.