

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования  
**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК  
  
Кафедра прикладной математики, механики, управления и программного обеспечения**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ “ХЕШ-ТАБЛИЦЫ” ПО ДИСЦИПЛИНЕ  
СТРУКТУРЫ И АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ОБРАБОТКИ ДАННЫХ**

Направление 02.03.03 «Математическое обеспечение и администрирование  
информационных систем»

Выполнил студент гр. Б8119-02.03.03техпро  
Хмелевский Е.Д.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Проверил: к.т.н., доцент кафедры ПММУ и ПО Остроухова С.Н \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Владивосток  
2021

## Описание

Цель лабораторной работы: разработать собственный класс **H𝑎𝑠ℎ𝑇𝑎𝑏𝑙𝑒** для работы с динамическими хеш-таблицами со следующим необходимым пакетом методов:

1. Конструктор
2. Деструктор
3. hash\_f1(key) −> addr
4. hash\_f2(addr1) −> addr2
5. add(record) −> int
6. del(record)
7. search(record) −> 0/1
8. print()

Параметром функции **ℎ𝑎𝑠ℎ\_𝑓1** является число, получаемое из поля записи **FIO**. Способ разрешения коллизии – повторное хеширование с увеличение на единицу.

## Спецификация

СТРУКТУРА ХЭШ ТАБЛИЦЫ

student – структура, состоящая из строк FIO, passport, hostel и целого положительного числа roomNumber

ПОЛЯ

Класс HashTable состоит из следующих полей:

* max\_size - размер таблицы;
* \*table – таблица, хранящая поля студентов;
* status - вектор состояний, где 0 – пустая ячейка, 1 – заполненная;
* ResizeUpper\_occupancy – процент заполняемости в процентах от 0 до 100;
* occupancy – процент заполненности от 0 до 100.

МЕТОДЫ

1. Конструктор

**HashTable::HashTable(int max\_size, unsigned int ResizeUpper\_occupancy)** – инициализирует такие члены экземпляра класса **HashTable**, как:

* размер таблицы **this->max\_size** типа **int** с помощью входного аргумента max\_size того же типа;
* процент заполняемости **this->ResizeUpper\_occupancy** типа **int** с помощью входного аргумента **ResizeUpper\_occupancy** того же типа.

Так же данный конструктор создаёт массив **table** структуры **student** размера **max\_size** и массив состояний **status** типа **int\*** размера **max\_size,** инициализированный нулями. Данные массивы создаются с помощью выделения памяти в куче.

1. Деструктор

**HashTable::~HashTable()** – очищает память выделенную для массивов **table** и **status** класса **HashTable**.

1. Первичная функция хэширования

**int HashTable::HashF1(string key) const** – функция производит преобразование входного ключа **key** типа **string** в адрес записи с помощью нахождения остатка от деления первого символа ключа на размер таблицы **this->max\_size** и возвращает полученный адрес типа **int**.

1. Вторичная функция хэширования

**int HashTable::HashF2(int h1) const** – функция возвращает результат формулы **h1+1**;

1. Добавление

**int HashTable::Add(student pupil)** – функция добавляет запись **pupil** структуры **student** в массив записей **table** типа **student\*.** Добавление происходит следующим образом – поле **passport** записи **pupil** хэшируется первичной функцией **HashF1** и производится попытка добавления этой записи. Если же соответствующий элемент массива **status** будет равен 0, запись добавится в таблицу, предварительно расширившуюся в два раза, если процент заполненности таблицы больше процента заполняемости **this->ResizeUpper\_occupancy**, так же увеличится общее количество записей в таблице на 1 и функция вернёт 1. Если же при первичном хэшировании соответствующий элемент массива **status** будет равен 1, будет производиться вторичное хэширование пока не найдётся элемент массива **status** равный нулю, тогда запись добавится в массив **table**, количество студентов **this->occupancy** увеличится на 1 и функция вернёт значение 1. Если же при поиске элемента массива **status** равного нулю найдётся ячейка массива **table** с таким же полем **FIO**, как и у входной записи, запись не добавится и в консоль выведется сообщение **"Студент с такой фамилией уже добавлен: "** вместе со всеми полями записи **pupil**. Так же в этом случае функция вернёт значение 0.

1. Удаление

**int HashTable::Del(student pupil)** – функция удаляет запись **pupil** структуры **student** из массива записей **table** типа **student\*.** Удаление происходит следующим образом – поле **passport** записи **pupil** хэшируется первичной функцией **HashF1** и производится попытка удаления этой записи. Если же соответствующий элемент массива **status** будет равен 1, запись удалится из таблицы, предварительно сузившись в два раза, если процент заполненности таблицы меньше процента заполняемости **this->ResizeUpper\_occupancy**, так же уменьшится общее количество записей в таблице на 1 и функция вернёт 1. Если же при первичном хэшировании соответствующий элемент массива **status** будет равен 1, будет производиться вторичное хэширование пока не найдётся элемент массива **status** равный единице, тогда запись удалится из массива **table**, количество студентов **this->occupancy** уменьшится на 1 и функция вернёт значение 1. Если же при поиске элемента массива **status** равного единице не найдётся соответствующая ячейка массива **table**, функция вернёт 0.

1. Расширение/сужение

**void HashTable::Resize(bool check)** – сужает или расширяет таблицу **table** в зависимости от значения аргумента **check**. Если **check = true**, таблица **table** и массив состояний **status** сузятся в двое, перехэшируя все записи из таблицы старого размера. Иначе таблица увеличится вдвое тоже перехэшируя все записи из таблицы старого размера.

1. Поиск

**bool HashTable::Search(student pupil)** – ищет запись в массиве **table** с такими же полями как и у записи **pupil**. Если такая запись найдётся, функция вернёт **true**, иначе **false**.

1. Печать

**void HashTable::WriteHashTable ()** – процедура выводит сообщение **“HASH TABLE"** с последующим переносом строки. Далее выводятся все поля каждой записи массива **table** с помощью перегруженного для структуры **student** оператора вывода **ostream<<** вместе с их порядковыми номерами в таблице. Если все поля какой-нибудь записи будут пустыми, то выведется сообщение **“NULL”**. И если же поля всех записей массива **table** будут пустыми, выведется сообщение **“ Таблица пуста!”**.

## Тесты

**Обозначения:**

**student** student0 = {"Savorov A T", "0000 000000", "6.1", **123**};

**student** student1 = {"Ivanov A T", "1000 000000", "6.1", **342**};

**student** student2 = {"Savorov A T", "1300 000000", "6.1", **234**};

**student** student3 = {"Petrov A T", "3000 000000", "6.1", **104**};

**student** student4 = {"Savorov A T", "4000 000000", "6.1", **432**};

**student** student5 = {"Petrov A T", "6000 000000", "6.1", **231**};

**student** student6 = {"Sorokin A T", "5000 000000", "6.1", **231**};

**student** student7 = {"Vishniakov A T", "1200 000000", "6.1", **345**};

// коллизия с student7 и student1 и student2

**student** student8 = {"Lasarenko A T", "1100 000000", "6.1", **132**};

// паспорт совпал с student1

**student** student9 = {"Kosicina A T", "1000 000000", "6.1", **235**};

1. **HashTable::HashTable(int max\_size, unsigned int ResizeUpper\_occupancy)**

|  |  |  |
| --- | --- | --- |
| **Тестовая ситуация** | **Входные данные** | **Выходные данные** |
| Количество записей не равно нулю. | **Таблица**: table;  **Количество ячеек**: 10;  **Процент заполняемости:** 80; | **return: table** {  *max\_size*: 10;  *this->ResizeUpper\_occupancy: 80;*  *status: new int[this->max\_size];*  *table: new student[this->max\_size];*  } |
| Количество записей равно нулю. | **Таблица**: table;  **Количество ячеек**: 0;  **Процент заполняемости:** 80; | **return: table** {  *max\_size*: 10;  *this->ResizeUpper\_occupancy: 80;*  *status: new int[this->max\_size];*  *table: new student[this->max\_size];*  } |
| Процент заполняемости равен 0. | **Таблица**: table;  **Количество ячеек**: 12;  **Процент заполняемости:** 0; | **return: table** {  *max\_size*: 12;  *this->ResizeUpper\_occupancy: 40;*  *status: new int[this->max\_size];*  *table: new student[this->max\_size];*  } |

1. **int HashTable::Add(student pupil)**

|  |  |  |
| --- | --- | --- |
| **Тестовая ситуация** | **Входные данные** | **Выходные данные** |
| Добавляем первую запись. | **Таблица**: table;  **Количество ячеек**: 10;  **Количество заполненных ячеек**: 0;  **Процент заполняемости:** 80;   |  |  |  |  | | --- | --- | --- | --- | | student | hash1 | hash2 | index | | - | - | - | - |   **Add**: student0. | **return:** 1  //код символа ‘0’ - 48   |  |  |  |  | | --- | --- | --- | --- | | student | hash1 | hash2 | index | | 0 | 8 | - | 8 | |
| Добавляем вторую запись, не образующую коллизию. | **Таблица**: table;  **Количество ячеек**: 10;  **Количество заполненных ячеек**: 1;  **Процент заполняемости:** 80;   |  |  |  |  | | --- | --- | --- | --- | | student | hash1 | hash2 | index | | 0 | 8 | - | 8 |   **Add**: student1. | **return:** 1   |  |  |  |  | | --- | --- | --- | --- | | student | hash1 | hash2 | index | | 1 | 0 | - | 0 | | 0 | 8 | - | 8 | |
| Добавляем третью запись, образующую коллизию. | **Таблица**: fridge;  **Количество ячеек**: 10;  **Количество заполненных ячеек**: 2;   |  |  |  |  | | --- | --- | --- | --- | | student | hash1 | hash2 | index | | 1 | 0 | - | 0 | | 0 | 8 | - | 8 |   **Add**: student2. | **return:** 1   |  |  |  |  | | --- | --- | --- | --- | | student | hash1 | hash2 | index | | 1 | 0 | - | 0 | | 2 | 0 | 1 | 1 | | 0 | 8 | - | 8 | |
| Добавляем четвертую запись, которая снова образует коллизию. | **Таблица**: fridge;  **Количество ячеек**: 10;  **Количество заполненных ячеек**: 3;  **Процент заполняемости:** 80;   |  |  |  |  | | --- | --- | --- | --- | | student | hash1 | hash2 | index | | 1 | 0 | - | 0 | | 2 | 0 | 1 | 1 | | 0 | 8 | - | 8 |   **Add**: student7. | **return**: 1   |  |  |  |  | | --- | --- | --- | --- | | student | hash1 | hash2 | index | | 1 | 0 | - | 0 | | 2 | 0 | 1 | 1 | | 0 | 8 | - | 8 | | 7 | 0 | 1, 2 | 2 | |
| Добавляем запись, которая уже существует в таблице. | **Таблица**: fridge;  **Количество ячеек**: 10;  **Количество заполненных ячеек**: 4;  **Процент заполняемости:** 80;   |  |  |  |  | | --- | --- | --- | --- | | student | hash1 | hash2 | index | | 1 | 0 | - | 0 | | 2 | 0 | 1 | 1 | | 0 | 8 | - | 8 | | 7 | 0 | 1, 2 | 2 |   **Add**: student9. | **return**: 0   |  |  |  |  | | --- | --- | --- | --- | | student | hash1 | hash2 | index | | 1 | 0 | - | 0 | | 2 | 0 | 1 | 1 | | 0 | 8 | - | 8 | | 7 | 0 | 1, 2 | 2 |   **Console:** Студент с таким пасспортом уже добавлен:  Kosicina A T, паспорт: 1000 00000, общежитие: 6.1, комната: 235 |
| Добавляем ещё 5 записей до процента заполненности таблицы равного 80. | **Таблица**: table;  **Количество ячеек**: 10;  **Количество заполненных ячеек**: 4;  **Процент заполняемости:** 80;   |  |  |  |  | | --- | --- | --- | --- | | student | hash1 | hash2 | index | | 1 | 0 | - | 0 | | 2 | 0 | 1 | 1 | | 0 | 8 | - | 8 | | 7 | 0 | 1, 2 | 2 |   **Add**: student3, student4, student5, student6, student8 | **return**: 1, 1, 1, 1, 1  0) NULL  1) NULL  2) NULL  3) NULL  4) NULL  5) NULL  6) NULL  7) NULL  8) Savorov A T, паспорт: 0000 00000, общежитие: 6.1, комната: 123  9) Ivanov A T, паспорт: 1000 00000, общежитие: 6.1, комната: 342  10) Savorov A T, паспорт: 1300 00000, общежитие: 6.1, комната: 234  11) Vishniakov A T, паспорт: 1200 00000, общежитие: 6.1, комната: 345  12) Petrov A T, паспорт: 3000 00000, общежитие: 6.1, комната: 104  13) Savorov A T, паспорт: 4000 00000, общежитие: 6.1, комната: 432  14) Petrov A T, паспорт: 6000 00000, общежитие: 6.1, комната: 231  15) Sorokin A T, паспорт: 5000 00000, общежитие: 6.1, комната: 231  16) Lasarenko A T, паспорт: 1100 00000, общежитие: 6.1, комната: 132  17) NULL  18) NULL  19) NULL  **Количество ячеек**: 20;  **Количество заполненных ячеек**: 8; |

1. **int HashTable::Del(student pupil)**

|  |  |  |
| --- | --- | --- |
| **Тестовая ситуация** | **Тест** | **Выходные данные** |
| Удаляем существующую запись. | **Таблица**: fridge;  **Количество ячеек**: 20;  **Количество заполненных ячеек**: 9;  **Процент заполняемости:** 80;  0) NULL  1) NULL  2) NULL  3) NULL  4) NULL  5) NULL  6) NULL  7) NULL  8) Savorov A T, паспорт: 0000 00000, общежитие: 6.1, комната: 123  9) Ivanov A T, паспорт: 1000 00000, общежитие: 6.1, комната: 342  10) Savorov A T, паспорт: 1300 00000, общежитие: 6.1, комната: 234  11) Vishniakov A T, паспорт: 1200 00000, общежитие: 6.1, комната: 345  12) Petrov A T, паспорт: 3000 00000, общежитие: 6.1, комната: 104  13) Savorov A T, паспорт: 4000 00000, общежитие: 6.1, комната: 432  14) Petrov A T, паспорт: 6000 00000, общежитие: 6.1, комната: 231  15) Sorokin A T, паспорт: 5000 00000, общежитие: 6.1, комната: 231  16) Lasarenko A T, паспорт: 1100 00000, общежитие: 6.1, комната: 132  17) NULL  18) NULL  19) NULL  **Del:** student8. | **return**: 1  0) NULL  1) NULL  2) NULL  3) NULL  4) NULL  5) NULL  6) NULL  7) NULL  8) Savorov A T, паспорт: 0000 00000, общежитие: 6.1, комната: 123  9) Ivanov A T, паспорт: 1000 00000, общежитие: 6.1, комната: 342  10) Savorov A T, паспорт: 1300 00000, общежитие: 6.1, комната: 234  11) Vishniakov A T, паспорт: 1200 00000, общежитие: 6.1, комната: 345  12) Petrov A T, паспорт: 3000 00000, общежитие: 6.1, комната: 104  13) Savorov A T, паспорт: 4000 00000, общежитие: 6.1, комната: 432  14) Petrov A T, паспорт: 6000 00000, общежитие: 6.1, комната: 231  15) Sorokin A T, паспорт: 5000 00000, общежитие: 6.1, комната: 231  16) NULL  17) NULL  18) NULL  19) NULL |
| Удаляем не существующую запись. | **Таблица**: fridge;  **Количество ячеек**: 20;  **Количество заполненных ячеек**: 8;  **Процент заполняемости:** 80;  0) NULL  1) NULL  2) NULL  3) NULL  4) NULL  5) NULL  6) NULL  7) NULL  8) Savorov A T, паспорт: 0000 00000, общежитие: 6.1, комната: 123  9) Ivanov A T, паспорт: 1000 00000, общежитие: 6.1, комната: 342  10) Savorov A T, паспорт: 1300 00000, общежитие: 6.1, комната: 234  11) Vishniakov A T, паспорт: 1200 00000, общежитие: 6.1, комната: 345  12) Petrov A T, паспорт: 3000 00000, общежитие: 6.1, комната: 104  13) Savorov A T, паспорт: 4000 00000, общежитие: 6.1, комната: 432  14) Petrov A T, паспорт: 6000 00000, общежитие: 6.1, комната: 231  15) Sorokin A T, паспорт: 5000 00000, общежитие: 6.1, комната: 231  16) NULL  17) NULL  18) NULL  19) NULL  **Del:** student8. | **return**: 0  0) NULL  1) NULL  2) NULL  3) NULL  4) NULL  5) NULL  6) NULL  7) NULL  8) Savorov A T, паспорт: 0000 00000, общежитие: 6.1, комната: 123  9) Ivanov A T, паспорт: 1000 00000, общежитие: 6.1, комната: 342  10) Savorov A T, паспорт: 1300 00000, общежитие: 6.1, комната: 234  11) Vishniakov A T, паспорт: 1200 00000, общежитие: 6.1, комната: 345  12) Petrov A T, паспорт: 3000 00000, общежитие: 6.1, комната: 104  13) Savorov A T, паспорт: 4000 00000, общежитие: 6.1, комната: 432  14) Petrov A T, паспорт: 6000 00000, общежитие: 6.1, комната: 231  15) Sorokin A T, паспорт: 5000 00000, общежитие: 6.1, комната: 231  16) NULL  17) NULL  18) NULL  19) NULL |
| Удаляем ещё 2 записи | **Таблица**: fridge;  **Количество ячеек**: 20;  **Количество заполненных ячеек**: 8;  **Процент заполняемости:** 80;  0) NULL  1) NULL  2) NULL  3) NULL  4) NULL  5) NULL  6) NULL  7) NULL  8) Savorov A T, паспорт: 0000 00000, общежитие: 6.1, комната: 123  9) Ivanov A T, паспорт: 1000 00000, общежитие: 6.1, комната: 342  10) Savorov A T, паспорт: 1300 00000, общежитие: 6.1, комната: 234  11) Vishniakov A T, паспорт: 1200 00000, общежитие: 6.1, комната: 345  12) Petrov A T, паспорт: 3000 00000, общежитие: 6.1, комната: 104  13) Savorov A T, паспорт: 4000 00000, общежитие: 6.1, комната: 432  14) Petrov A T, паспорт: 6000 00000, общежитие: 6.1, комната: 231  15) Sorokin A T, паспорт: 5000 00000, общежитие: 6.1, комната: 231  16) NULL  17) NULL  18) NULL  19) NULL  **Del:** student5, student6. | **return:** 1, 1  0) Ivanov A T, паспорт: 1000 00000, общежитие: 6.1, комната: 342  1) Savorov A T, паспорт: 1300 00000, общежитие: 6.1, комната: 234  2) Vishniakov A T, паспорт: 1200 00000, общежитие: 6.1, комната: 345  3) Petrov A T, паспорт: 3000 00000, общежитие: 6.1, комната: 104  4) Savorov A T, паспорт: 4000 00000, общежитие: 6.1, комната: 432  5) NULL  6) NULL  7) NULL  8) Savorov A T, паспорт: 0000 00000, общежитие: 6.1, комната: 123  9) NULL  **Количество ячеек**: 10;  **Количество заполненных ячеек**: 6; |

1. **bool HashTable::Search(student pupil)**

|  |  |  |
| --- | --- | --- |
| **Тестовая ситуация** | **Тест** | **Выходные данные** |
| Ищем существующую запись | **Таблица**: table;  **Количество ячеек**: 10;  **Количество заполненных ячеек**: 6;  **Процент заполняемости:** 80;  0) Ivanov A T, паспорт: 1000 00000, общежитие: 6.1, комната: 342  1) Savorov A T, паспорт: 1300 00000, общежитие: 6.1, комната: 234  2) Vishniakov A T, паспорт: 1200 00000, общежитие: 6.1, комната: 345  3) Petrov A T, паспорт: 3000 00000, общежитие: 6.1, комната: 104  4) Savorov A T, паспорт: 4000 00000, общежитие: 6.1, комната: 432  5) NULL  6) NULL  7) NULL  8) Savorov A T, паспорт: 0000 00000, общежитие: 6.1, комната: 123  9) NULL  **Search**: student0 = {"Savorov A T", "0000 000000", "6.1", 123}. | **return**: 1  0) Ivanov A T, паспорт: 1000 00000, общежитие: 6.1, комната: 342  1) Savorov A T, паспорт: 1300 00000, общежитие: 6.1, комната: 234  2) Vishniakov A T, паспорт: 1200 00000, общежитие: 6.1, комната: 345  3) Petrov A T, паспорт: 3000 00000, общежитие: 6.1, комната: 104  4) Savorov A T, паспорт: 4000 00000, общежитие: 6.1, комната: 432  5) NULL  6) NULL  7) NULL  8) Savorov A T, паспорт: 0000 00000, общежитие: 6.1, комната: 123  9) NULL |
| Ищем не существующую запись. | **Таблица**: table;  **Количество ячеек**: 10;  **Количество заполненных ячеек**: 6;  **Процент заполняемости:** 80;  0) Ivanov A T, паспорт: 1000 00000, общежитие: 6.1, комната: 342  1) Savorov A T, паспорт: 1300 00000, общежитие: 6.1, комната: 234  2) Vishniakov A T, паспорт: 1200 00000, общежитие: 6.1, комната: 345  3) Petrov A T, паспорт: 3000 00000, общежитие: 6.1, комната: 104  4) Savorov A T, паспорт: 4000 00000, общежитие: 6.1, комната: 432  5) NULL  6) NULL  7) NULL  8) Savorov A T, паспорт: 0000 00000, общежитие: 6.1, комната: 123  9) NULL  **Search**: student6 = {"Sorokin A T", "5000 000000", "6.1", 231}; | **return**: 0  0) Ivanov A T, паспорт: 1000 00000, общежитие: 6.1, комната: 342  1) Savorov A T, паспорт: 1300 00000, общежитие: 6.1, комната: 234  2) Vishniakov A T, паспорт: 1200 00000, общежитие: 6.1, комната: 345  3) Petrov A T, паспорт: 3000 00000, общежитие: 6.1, комната: 104  4) Savorov A T, паспорт: 4000 00000, общежитие: 6.1, комната: 432  5) NULL  6) NULL  7) NULL  8) Savorov A T, паспорт: 0000 00000, общежитие: 6.1, комната: 123  9) NULL |

1. **void HashTable::WriteHashTable ()**

|  |  |  |
| --- | --- | --- |
| **Тестовая ситуация** | **Входные данные** | **Выходные данные** |
| Таблица пустая. | **Таблица**: table;  **Количество ячеек**: 10;  **Количество заполненных ячеек**: 0;  **Процент заполняемости:** 80. | **Console:**  HASH TABLE  Таблица пуста! |
| Таблица не пустая. | **Таблица**: table;  **Количество ячеек**: 10;  **Количество заполненных ячеек**: 6;  **Процент заполняемости:** 80. | **Console:**  HASH TABLE  0) Ivanov A T, паспорт: 1000 00000, общежитие: 6.1, комната: 342  1) Savorov A T, паспорт: 1300 00000, общежитие: 6.1, комната: 234  2) Vishniakov A T, паспорт: 1200 00000, общежитие: 6.1, комната: 345  3) Petrov A T, паспорт: 3000 00000, общежитие: 6.1, комната: 104  4) Savorov A T, паспорт: 4000 00000, общежитие: 6.1, комната: 432  5) NULL  6) NULL  7) NULL  8) Savorov A T, паспорт: 0000 00000, общежитие: 6.1, комната: 123  9) NULL |