

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ 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 №4 ПО ДИСЦИПЛИНЕ  
«СТРУКТУРЫ И АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ОБРАБОТКИ ДАННЫХ»**

Направление 02.03.03 «Математическое обеспечение и администрирование  
информационных систем»

* Выполнил студент гр. Б8204  
  Проскурин Денис Александрович  
  Проверил:  
  Доцент, к.т.н С.Н.Остроухова

Владивосток 2019

**Неформальная постановка задачи**

Реализовать класс динамическая хеш-таблица, которая хранит данные о сотрудниках предприятия. В таблице хранится номер паспорта(ключ) и ФИО.

При добавлении записи в таблицу ячейка статуса строки ставится в 1, при удалении записи — 0. Класс должен содержать следующие методы:

1. Конструктор
2. Деструктор
3. Добавить запись
4. Удалить запись

* Поиск записи в таблице

1. Вывод всей таблицы

Таблица использует вектор наличия записи в таблице со статусом 0 и 1.

Первичная хеш-функция складывает все цифры уникального идентификатора (серия паспорта) и делит на размерность таблицы.

Вторичная хеш-функция — квадратичное опробирование.

**Спецификация методов класса**

**class** Hashtable {  
**private**:  
   
 **const char** min\_size = 1;-минимальный размер создаваемой таблицы

**const char** max\_k=80;  
 **const char** min\_k=60;  
 **const unsigned int** min\_key=100000;  
 **const unsigned int** max\_key=999999;

**unsigned int** N; //размер таблицы

**unsigned int** default\_N; //size of table  
 **double** k; //коэффициент заполняемости от 50% до 80%  
 **struct** Record \*table; //массив записей  
 **bool** \*status; //вектор статуса строки таблицы  
 **unsigned int** get\_hash\_one(Record); //хеш-функция 1  
 **unsigned int** get\_hash\_two(**unsigned int**, **unsigned int**); //хеш-функция 2  
 **double** count\_koef(**int**);//подсчёт коэффициента заполняемости  
 **void** inc\_size();//увеличение хеш таблицы в 2 раза  
 **void** red\_size();//уменьшение хеш таблицы в 2 раза  
 **void** rehashing(); //перехеширование таблицы   
**public**:  
 Hashtable(**unsigned int**, **double**);//конструктор   
 ~Hashtable();//деструктор  
 **bool** add(Record);//метод добавления записи  
 **bool** remove(Record);//мето удаления записи  
 **unsigned int** search(Record);//поиск записи в таблице  
 **void** print();//печать таблицы на экран

};

**Hashtable(N, k)** — конструктор класса хеш-таблица, создаёт таблицу N-ой размерности с коэффициентом заполняемости k. Eсли, задан некорректный размер,по умолчанию размер таблицы 10. Если задан некорректный коэффициент(60<=k<=80), то коэффициент по умолчанию равен 70.

**~Hashtable()** - диструктор класса хеш-таблица. Освобождает память динамически выделенную под массив записей.

**Unsigned short Add(Record)** — метод, который добавляет запись в таблицу по первичной хеш-функции или по вторичной хеш-функции(квадратичное опробирование) и возвращает код 0, если операция произошла успешно; если запись невозможно добавить возвращаются следующие коды: 1, 2, 3.

**Bool remove(Record)** — метод, который удаляет запись из таблицы. Если удаление произошло успешно возвращается код 0, иначе возвращается код 4.

**Unsigned int search(Record)** — метод ,который возвращает номер строки в которой находится запись, если запись найдена возвращается номер строки в которой она находится, в противном случае- размерность таблицы.

**Void print()**- вывод хеш-таблицы

**Коды ошибок:**

1. Некорректный ключ
2. Запись уже существует
3. Нельзя добавить ни по одной из хеш-функций
4. Запись не существует

**Тесты**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| **Тестирование конструктора класса** | |
| //Создание экземпляра класса размер таблицы меньше иди равен 1  Hashtable example(10, 80); | 1 |
| //Создание экземпляра класса. Задаваемый кожффициент не лежит в диапозоне от 50 до 80  Hashtable example(10, 82); | 2 |
| Hashtable example(1, 40); | 1 |
| **Тестирование метода add** | |
| //Добавление записи с некорректным ключом  Record a; a.key = 2749; a.data = "PDA";  table.add(a); | 3 |
| //Добавление записи с некорректным ключом record.key = 9999992; record.data = "PDA";  table.add(record); | 3 |
| //Добавление уже существующей записи  HashTable  0)350606 PDD  1) empty  2) empty  3) empty  **4)350600 PDA**  5) empty  6)350602 PDB  7) empty  8)350604 PDC  9) empty  record.key = 350600; record.data = "PDA"; table.add(record); | 4 |
| //запись не добавляется ни по одной из хеш функций  HashTable  0)350606 PDD  1) empty  2) empty  3)530500 PDA2  4)350600 PDA  5)530601 PDA1  6)350602 PDB  7) empty  8)350604 PDC  9)530605 PDA3  record.key = 530600; record.data = "PDA\_ERR"; table.add(record); | 5 |
| //расширение таблицы по коэффициенту c последющим перехешированием  HashTable  0)350606 PDD  1)350616 PDI  2)350608 PDE  3) empty  4)350600 PDA  5)350610 PDF  6)350602 PDB  7)350612 PDG  8)350604 PDC  9)350614 PDH  record.key += 2; record.data="NEW\_PDA"; table.add(record); | HashTable  0)350606 PDD  1)350616 PDI  2)350608 PDE  3) empty  4) empty  5)350620 NEW\_PDA  6) empty  7) empty  8) empty  9) empty  10) empty  11) empty  12) empty  13) empty  14)350600 PDA  15)350610 PDF  16)350602 PDB  17)350612 PDG  18)350604 PDC  19)350614 PDH  код возврата 0 |
| //успешное добавление записи  HashTable  0) empty  1) empty  2) empty  3) empty  4) empty  5) empty  6) empty  7) empty  8) empty  9) empty  record.key = 350600; record.data = "PDA"; table.add(record); table.print(); | HashTable  0) empty  1) empty  2) empty  3) empty  **4)350600 PDA**  5) empty  6) empty  7) empty  8) empty  9) empty  код возврата 0 |
| **Тестирование метода remove** | |
| //удаление несуществующей записи  HashTable  0)350606 PDD  1)350616 PDI  2)350608 PDE  3) empty  4)350600 PDA  5)350610 PDF  6)350602 PDB  7)350612 PDG  8)350604 PDC  9)350614 PDH  record.key = 230000; record.data = "DON'T EXISTS"; table.remove(record); | 6 |
| //успешное удаление существующей записи  HashTable  0)350606 PDD  1)350616 PDI  2)350608 PDE  3) empty  4)350600 PDA  5)350610 PDF  6)350602 PDB  7)350612 PDG  8)350604 PDC  9)350614 PDH  record.key = 350600; record.data = "RECORD EXISTS"; table.remove(record); | HashTable  0)350606 PDD  1)350616 PDI  2)350608 PDE  3) empty  **4) empty**  5)350610 PDF  6)350602 PDB  7)350612 PDG  8)350604 PDC  9)350614 PDH  код возврата 0 |
| //успешное удаление записи с перехешированием  HashTable  **0)350606 PDD**  **1)530606 NEW\_PDD**  2) empty  3) empty  4)350600 PDA  5) empty  6)350602 PDB  7) empty  8)350604 PDC  9) empty  record.key = 350606; record.data = "PDD"; table.remove(record); | HashTable  **0)530606 NEW\_PDD**  1) empty  2) empty  3) empty  4)350600 PDA  5) empty  6)350602 PDB  7) empty  8)350604 PDC  9) empty  код возврата 0 |
| //удаление записи с последующим уменьшением размера таблицы и перехешированием  HashTable  0)350606 PDD  1) empty  2) empty  3) empty  4)350600 PDA  5) empty  6)350602 PDB  7) empty  8)350604 PDC  9) empty  record.key = 350600; record.data = "PDA"; table.remove(record); | HashTable  0)350606 PDD  1)350602 PDB  2) empty  3)350604 PDC  4) empty  код возврата 0 |
| //удаление записи с уменьшением размера таблицы до минимального заданного размера  HashTable  0) empty  1)810002 PDB  2) empty  3) empty  4)810000 PDA  удаляем все записи | HashTable  0) empty  1)810002 PDB  2) empty  3) empty  4)810000 PDA |
| **Тестирование метода search** | |
| //поиск сущестующей записи  HashTable  0)350606 PDD  1) empty  2) empty  3) empty  4)350600 PDA  5) empty  6)350602 PDB  7) empty  8)350604 PDC  9) empty  record.key = 350600; record.data = "PDA"; | 4 |
| //поиск несущестующей записи  HashTable  0)350606 PDD  1) empty  2) empty  3) empty  4)350600 PDA  5) empty  6)350602 PDB  7) empty  8)350604 PDC  9) empty  record.key = 100000; record.data = "NOTEXISTS"; | 10 |