

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  
  Кокорин Игорь Вячеславович  
  Проверил:  
  Доцент, к.т.н С.Н.Остроухова

Владивосток 2019

**Неформальная постановка задачи**

Реализовать класс статическая хеш-таблица, которая хранит данные о валютах. В таблице хранится дата, код валюты и её названии. 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
5. Поиск записи в таблице
6. Вывод всей таблицы

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

Первичная хеш-функция складывает значения полей кода и даты в единое число и делит на размерность таблицы.

Вторичная хеш-функция — квадратичное опробирование.

**Спецификация методов класса**

struct element{

std::string data;

unsigned int date;

unsigned int code;

};

class Hashtable {

private:

struct element \*vector;

unsigned int size;

bool \*status;

unsigned int Hashing1(element);

unsigned int Hashing2(unsigned int,unsigned int);

public:

Hashtable(unsigned int Mysize);

bool Del(element);

unsigned int Add(element);

~Hashtable();

unsigned int Search(element element1);

void Statwriter();

void print();

};

void Error(int c);

**Hashtable(N, k)** — конструктор класса хеш-таблица, создаёт таблицу N-ой размерности . Eсли, задан значение <=1 то ,по умолчанию размер таблицы 5.

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

**Unsigned short Add(element)** — 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: 2, 3.

**Bool Del(element)** — метод, который удаляет запись из таблицы. Если удаление произошло успешно возвращается код 0, иначе возвращается код 1.

**Unsigned int search(element)** — 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. Попытка удаления несуществующего элемента
2. Запись уже существует
3. Нельзя добавить ни по одной из хеш-функций.

**Тесты**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| **Тестирование конструктора класса** | |
| //Создание экземпляра класса размер таблицы меньше или равен 1  Hashtable example(1); | Hashtable example(5); |
| Правильно заданное значения размера таблицы  Hashtable example(5); | 0 empty 0  --------------------------------------------------------------  1 empty 0  --------------------------------------------------------------  2 empty 0  --------------------------------------------------------------  3 empty 0  --------------------------------------------------------------  4 empty 0  -------------------------------------------------------------- |
| **Тестирование метода add** | |
| //Добавление в хеш-таблицу элемента с повторяющимся ключом (Проверка сравнения при хеширвании).  Hashtable example(5);  element a;  a.date=2011;  a.code=1000;  a.data=Ruble;  unsigned int c;  c=example.Add(a);  example.print();  Error(c);  c=example.Add(a);  Error(c); | 0 empty 0  1 1000 2011 Ruble Status: 1  2 empty 0  3 empty 0  4 empty 0  (2) repeating data detected  0 empty 0  1 1000 2011 Ruble Status: 1  2 empty 0  3 empty 0  4 empty |
| //Добавление в хеш-таблицу элемента с повторяющимся ключом (Проверка сравнения при перехешировании по вторичной хеш функции).  Hashtable example(5);  element a;  a.date=2011;  a.code=1000;  a.data=Ruble;  unsigned int c;  for (int i=0;i<3;i++){  c=example.Add(a);  a.date+=2;  a.code+=3;  }  example.print();  Error(c);  a.code=1006;  a.date=2015;  c=example.add(a);  Error(c); | 0 1006 2015 Ruble Status: 1  1 1000 2011 Ruble Status: 1  2 1003 2013 Ruble Status: 1  3 empty 0  4 empty 0  (2)repeating data detected  0 1006 2015 Ruble Status: 1  1 1000 2011 Ruble Status: 1  2 1003 2013 Ruble Status: 1  3 empty 0  4 empty 0 |
| //Добавление уже существующей записи после перехеширования хеш-таблицы.  Hashtable example(5);  element a;  a.date=2011;  a.code=1000;  a.data=Ruble;  unsigned int c;  for (int i=0;i<3;i++){  c=example.Add(a);  a.date+=2;  a.code+=3;  }  example.print();  Error(c);  a.code=1003;  a.date=2013;  c=example.Del(a);  Error(c);  example.print();  a.code=1003;  a.date=2013;  c=example.Add(a);  Error(c);  example.print(); | 0 1006 2015 Ruble Status: 1  1 1000 2011 Ruble Status: 1  2 1003 2013 Ruble Status: 1  3 empty 0  4 empty 0  del hash 2  0 empty 0  1 1000 2011 Ruble Status: 1  2 1006 2015 Ruble Status: 1  3 empty 0  4 empty 0  repeating data detected  0 1003 2013 Ruble Status: 1  1 1000 2011 Ruble Status: 1  2 1006 2015 Ruble Status: 1  3 empty 0  4 empty 0 |
| //запись не добавляется ни по одной из хеш функций  Hashtable example(5);  element a;  a.date=2011;  a.code=1000;  a.data=Ruble;  unsigned int c;  for (int i=0;i<4;i++){  c=example.Add(a);  a.date+=2;  a.code+=3;  Error(c);  }  example.print(); | 0 1006 2015 Ruble Status: 1  1 1000 2011 Ruble Status: 1  2 1003 2013 Ruble Status: 1  3 empty 0  4 empty 0  (3)Procedure of rehashing(quadratic) didn't lead to the result. |
| //успешное добавление записи  Hashtable example(5);  element a;  a.date=2011;  a.code=1000;  a.data=Ruble;  unsigned int c;  for (int i=0;i<3;i++){  c=example.Add(a);  a.date+=2;  a.code+=3;  Error(c);  }  example.print(); | 0 1006 2015 Ruble Status: 1  1 1000 2011 Ruble Status: 1  2 1003 2013 Ruble Status: 1  3 empty 0  4 empty 0 |
| **Тестирование метода Del** | |
| //удаление несуществующей записи  Hashtable example(5);  element a;  a.date=2011;  a.code=1000;  a.data=Ruble;  unsigned int c;  for (int i=0;i<3;i++){  c=example.Add(a);  a.date+=2;  a.code+=3;  Error(c);  }  example.print();  a.code=1010;  a.date=2013;  c=example.Del(a);  Error(c); | 0 1006 2015 Ruble Status: 1  1 1000 2011 Ruble Status: 1  2 1003 2013 Ruble Status: 1  3 empty 0  4 empty 0  (1)Attempt to delete non existing element  0 1006 2015 Ruble Status: 1  1 1000 2011 Ruble Status: 1  2 1003 2013 Ruble Status: 1  3 empty 0  4 empty 0 |
| //успешное удаление существующей записи  Hashtable example(5);  element a;  a.date=2011;  a.code=1000;  a.data=Ruble;  unsigned int c;  for (int i=0;i<3;i++){  c=example.Add(a);  a.date+=2;  a.code+=3;  Error(c);  }  example.print();  a.code=1010;  a.date=2013;  c=example.Del(a);  Error(c); | 0 1006 2015 Ruble Status: 1  1 1000 2011 Ruble Status: 1  2 1003 2013 Ruble Status: 1  3 empty 0  4 empty 0  del hash 0  0 empty 0  1 1000 2011 Ruble Status: 1  2 1003 2013 Ruble Status: 1  3 empty 0  4 empty 0 |
| **Тестирование метода search** | |
| //поиск сущестующей записи  Hashtable example(5);  element a;  a.date=2011;  a.code=1000;  a.data=Ruble;  unsigned int c;  for (int i=0;i<3;i++){  c=example.Add(a);  a.date+=2;  a.code+=3;  Error(c);  }  example.print();  a.code=1010;  a.date=2013;  int d=0;  c=example.Search(a);  std::cout<<“pos “<<d<<std::endl; | 0 1006 2015 Ruble Status: 1  1 1000 2011 Ruble Status: 1  2 1003 2013 Ruble Status: 1  3 empty 0  4 empty 0  pos 0  0 1006 2015 Ruble Status: 1  1 1000 2011 Ruble Status: 1  2 1003 2013 Ruble Status: 1  3 empty 0  4 empty 0 |
| //поиск несущестующей записи  Hashtable example(5);  element a;  a.date=2011;  a.code=1000;  a.data=Ruble;  unsigned int c;  for (int i=0;i<3;i++){  c=example.Add(a);  a.date+=2;  a.code+=3;  Error(c);  }  example.print();  a.code=1010;  a.date=2014;  int d=0;  c=example.Search(a);  std::cout<<“pos “<<d<<std::endl; | 0 1006 2015 Ruble Status: 1  1 1000 2011 Ruble Status: 1  2 1003 2013 Ruble Status: 1  3 empty 0  4 empty 0  pos 5  0 1006 2015 Ruble Status: 1  1 1000 2011 Ruble Status: 1  2 1003 2013 Ruble Status: 1  3 empty 0  4 empty 0 |