МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК**

**Кафедра прикладной математики, механики, управления и программного обеспечения**

**О Т Ч Е Т**

по лабораторной работе №4 по дисциплине “Структуры и алгоритмы компьютерной обработки данных”

|  |
| --- |
| Выполнил студент гр. Б8204  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. А. Федоренко |
| Проверил к.т.н. доцент  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.Н. Остроухова |

г. Владивосток

2018

**Неформальная постановка задачи**

Разработать класс Hash-таблица.

**Спецификация класса**

struct Hash {

int hash\_key = -1; // Значение hash-функции у записи

unsigned short int status = 0; // статус каждой записи (0 – свободна, 1-занята,

2 - занята)

string FIO = "----------", // ФИО преподавателя

name\_of\_lesson = "--------", // Название предмета

class\_number = "----", // Номер аудитории

lesson\_number = "-"; // Номер пары

};

class HashTable

{

public:

HashTable(int n = 10, float k = 0.8); // Конструктор

int addRecord (Hash hash); // Функция добавления записи в таблицу

int searchRecord (Hash hash); // Функция поиска записи в таблице

int deleteRecord (Hash hash); // Функция удаления записи из таблицы

void print(); // Функция печати таблицы

~HashTable(); // Деструктор

private:

Hash\* hash\_table;

int SIZE; // Переменная, хранящая размер таблицы

int count = 0; // Счётчик записей в таблице

float RATIO; // Переменная, хранящая коэффициент заполненности

таблицы, при превышении которого, таблица должна

увеличиваться

int hash1 (string, string); // 1-я функция хеширования

int hash2 (int); // 2-я функция хеширования

void add(Hash, int); // вспомогательная функция добавления записи в

таблицу

int compare(Hash, int); // функция сравнения двух записей

void resizeTable(); // Функция увеличения размера таблицы

};

**Спецификация функций**

int **addRecord** (Hash **hash**) - Функция добавления записи в таблицу.

На вход принимает запись **hash** и возвращает целочисленное значение:

* -1, если запись **hash** уже содержится в таблице
* -2, если запись с таким же ключом, как и у **hash**, уже содержится в таблице
* 0, если добавление записи **hash** прошло успешно

int **searchRecord** (Hash **hash**) - Функция поиска записи в таблице.

На вход принимает запись **hash** и возвращает целочисленное значение:

* -1, если запись **hash** найти не удалось
* Значение поля hash\_key в записи, которую нашла функция

int **deleteRecord** (Hash **hash**) - Функция удаления записи из таблицы.

На вход принимает запись **hash** и возвращает целочисленное значение:

* 1, если запись **hash** не содержится в таблице
* 0, если удаление записи **hash** из таблицы прошло успешно

void **print**() - Функция печати таблицы на экран.

На вход ничего не принимает и ничего не возвращает.

int **hash1** (string **s1**, string **s2**) – 1-я функция хеширования.

На вход принимает ключи **s1** и **s2** и возвращает целочисленное значение по правилу: коды символов **s1** и **s2** складываются, и это число берётся по модулю размера таблицы(SIZE)

int **hash2** (int **i**) – 2-я функция хеширования.

На вход принимает целочисленное значение **i**, полученное в результате хеширования, которое привело у коллизии,и возвращает целочисленное значение по формуле: (i+(SIZE-1))%SIZE

void **resizeTable**() - Функция увеличения размера таблицы.

Функция увеличивает размер таблицы, если заполненность таблицы превысила значение в переменной RATIO.

На вход ничего не принимается и ничего не возвращается.

**Тесты**

**NonRecursiveQuickSort:**

|  |  |  |  |
| --- | --- | --- | --- |
| Входные данные | Результат сортировки | Кол-во сравнений | Кол-во обменов |
| 10 9 8 7 6 5 4 3 2 1 | 1 2 3 4 5 6 7 8 9 10 | 34 | 5 |
| 1 2 3 4 5 6 7 8 9 10 | 1 2 3 4 5 6 7 8 9 10 | 31 | 0 |
| 1 1 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 1 1 | 30 | 13 |
| 10 3 8 1 16 15 1 1 4 20 8 | 1 1 1 3 4 8 8 10 15 16 20 | 45 | 10 |
| 20 1 3 10 17 28 21 4 1 20 20 | 1 1 3 4 10 17 20 20 20 21 28 | 51 | 9 |

**TooWayInsertion:**

|  |  |  |  |
| --- | --- | --- | --- |
| Входные данные | Результат сортировки | Кол-во сравнений | Кол-во перестановок |
| 10 9 8 7 6 5 4 3 2 1 | 1 2 3 4 5 6 7 8 9 10 | 9 | 10 |
| 1 2 3 4 5 6 7 8 9 10 | 1 2 3 4 5 6 7 8 9 10 | 9 | 10 |
| 1 1 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 1 1 | 9 | 10 |
| 10 3 8 1 16 15 1 1 4 20 8 | 1 1 1 3 4 8 8 10 15 16 20 | 19 | 22 |
| 20 1 3 10 17 28 21 4 1 20 20 | 1 1 3 4 10 17 20 20 20 21 28 | 31 | 21 |