МИНОБРНАУКИ
РОССИИ

*Федеральное государственное бюджетное образовательное учреждение высшего образования*
***«МИРЭА – Российский технологический университет»***

**РТУ МИРЭА**

---

Отчет по выполнению практического задания №3
**Тема:** Применение хеш-таблицы для поиска данных в двоичном файле с записями фиксированной длины
**Дисциплина:** Структуры и алгоритмы обработки данных

Выполнил студент Ермишова С. М.

Группа: ИНБО-06-21

Москва 2022

**Заголовок**

# Задание 1

## 1.1 Условие задачи

Условие задачи

Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к записям файла, структура записи которого приведена в варианте.

Задание варианта

| | | |
|---|---|---|
| 6 | Цепное хеширование | Товар: название, код – шестиразрядное число, завод изготовитель, цена, страна (название). |

## 1.2 Постановка задачи

Дано.

Файл двоичный с записями фиксированной длины.

Структура записи файла согласно варианту (изобразить в форме таблицы, указав названия полей).

Результат.

Хеш-таблица.

## 1.3 Подход к решению

1) Файл двоичный из записей фиксированного размера.

Структура записи файла из кода.

```
struct product{
    int code;
    char name[20];
    char factory[20];
    int price;
    char country[20];
};
```

Размер записи файла в байтах = 4+20+20+4+20=68 байт

Прототипы операций по управления двоичным файлом с указанием предусловия и постусловия.

Список подзадач:

- Преобразование бинарного файла в тестовый
- Преобразование текстового файла в бинарный
- Вывод записей двоичного файла
- Доступ к записи по ее порядковому номеру
- Формирование файла из исходного о поставках заданной страны

- Индекс последней записи
- Удаление записи по ключу
- Добавление новой записи в конец файла

**Определение прототипов функций:**

1) Преобразование бинарного файла в тестовый

**int stringBinToText(string fileName, string newfileName)**

Предусловие: string fileName – имя исходного строкового файла, string newfileName – имя нового бинарного файла

Постусловие: целое – результат корректности выполнения функции

2) Преобразование текстового файла в бинарный

**int stringTextToBin(string fileName, string newfileName)**

Предусловие: string fileName – имя исходного бинарного файла, string newfileName – имя нового строкового файла

Постусловие: целое – результат корректности выполнения функции

3) Вывод записей двоичного файла

**int out_bin_file(string filename)**

Предусловие: string filename – имя считываемого файла

Постусловие: целое – результат корректности выполнения функции

4) Доступ к записи по ее порядковому номеру

**int FindRecord(string filename, int n, product& x)**

Предусловие: string filename – имя считываемого файла, int n порядковый номер записи, product& x найденная в файле запись.

Постусловие: целое – результат корректности выполнения функции (-1 или 0). Значение найденной записи, переданное в record x.

5) Формирование файла из исходного о поставках заданной страны

**in CreateNewFileFrom(string filename, string newfilename, string country)**

Предусловие: fileName – имя исходного бинарного файла, string newfileName – имя нового строкового файла, string country – страна-поставщик

Постусловие: целое – результат корректности выполнения функции

6) Индекс последней записи

**int LastRecordIndex(string filename)**

Предусловие: string filename – имя считываемого файла

Постусловие: целое – индекс последней записи

7) Удаление записи по ключу

**int DelByKey(string filename, string key)**

Предусловие: string filename – имя считываемого файла, string key – ключ элемента

Постусловие: целое – результат корректности выполнения функции

8) Добавление новой записи в конец файла

**int AddToEndFile(string filename)**

Предусловие: string filename – имя считываемого файла

Постусловие: целое – результат корректности выполнения функции

2) Хеш-таблица

Структура элемента таблицы:

```
// структура записи
struct typeitem {
    int code=0;       // ключ записи - артикул, шестиразрядное число
    string name="-"; // название
    string factory="-"; // завод изготовитель
    int price = 0; // цена
    string country = "-"; // страна (название)
    int record_number; // порядковый номер записи

    bool openORclose=true;     //свободна ли ячейка
    bool deletedORnot=false;    //не удалялась ли ячейка

};
```

Код элемента таблицы, реализацию структуры таблицы:

```
// структура хеш-таблицы
struct HeshTable {
    int L = 19;
    typeitem **T;// таблица, динимачиский массив из объектов по постановке
задачи
    int insertedcount;//количество вставленных ключей
    int deletedcount; //количество удаленных ключей
    void createHeshTable() {
        T = new typeitem*[L];
        for (int i=0; i<L; i++){
            T[i] = new typeitem[size2];
        }
        insertedcount = 0;
        deletedcount=0;
    }
    void Resize(int newL){  //увеличение размера таблицы
        for (int i=0; i<L; i++){
            delete[] T[i];
        }
```

```
        delete[] T;

        L = newL;

        T = new typeitem*[L];
        for (int i=0; i<L; i++){
            T[i] = new typeitem[size2];
        }
    }
    void Delete(){
        for (int i=0; i<L; i++){
            delete[] T[i];
        }
        delete[] T;
    }
};
```

Код элемента, реализующий вставку ключа в таблицу

```
//вставка с рехешированием
int insertInHeshTable(int code, string name, string factory, int price, string
country, int record_number, HeshTable& t) {

    // рехеширование
    if(float(t.insertedcount) / t.L >= 0.75){ // коэффициент нагрузки
        HeshTable T2;   // создание новой таблицы и увеличивание ее размера в
сравнении с предыдущей вдвое
        T2.createHeshTable();
        T2.Resize(t.L*2);

        // заполнение новой таблицы старыми значениями с учетом нового
значения размера (рехеширую)
        for(int i=0; i<t.L; i++){
            for(int j=0; j<size2; j++){
                // добавляю только непустые неудаленные элементы
                if(t.T[i][j].openORclose == false &&
t.T[i][j].deletedORnot==false){
                    insertInHeshTable(t.T[i][j].code, t.T[i][j].name,
t.T[i][j].factory,
                                      t.T[i][j].price, t.T[i][j].country,
t.T[i][j].record_number, T2);
                }
            }
        }
        // добавление нового элемента в расширенную таблицу
        insertInHeshTable(code, name, factory, price, country, record_number,
T2);
        // увеличивание и изменение исходной таблицы
        t.Resize(T2.L);
        swap(T2, t);
        T2.Delete();
        return 0;
    }

    int i = hesh(code, t.L);
    int j=0;
    //разрешение коллизии
    while (j<size2 && t.T[i][j].openORclose == false)
        j += c;
    if (i < t.L)
    {
        t.T[i][j].code = code; t.T[i][j].name = name;
t.T[i][j].factory=factory;
        t.T[i][j].price=price; t.T[i][j].country=country;
t.T[i][j].record_number = record_number;
        t.T[i][j].openORclose = false;
        t.insertedcount++;
        return 0;
```

```
    }
    else
        return -1;
}
```

Код поиска записи по ключу в таблице

```
// поиск
int* search(HeshTable& t, int code) {
    int i = hesh(code, t.L);
    //ищем по кластеру
    int j=0;
    int* result = new int[2];
    result[0]=-1;
    result[1]=-1;

    while (j < size2 && ((t.T[i][j].openORclose == false &&
t.T[i][j].deletedORnot==false)
                         || (t.T[i][j].openORclose == true &&
t.T[i][j].deletedORnot == true))
           && t.T[i][j].code != code)
        j++;
    if (not (t.T[i][j].openORclose == true && t.T[i][j].deletedORnot ==
false)) {
        result[0] = i;
        result[1] = j;
    }
    return result;
}
```

Код удаление элемента из хеш-таблицы

```
//удаление
int deletedFromHeshTable(HeshTable& t, int code) {
    int* coor;
    coor = search(t, code);
    int i = coor[0], j = coor[1];
    if (i == -1) return 1;//нет такой записи в таблице
    t.T[i][j].deletedORnot = true;
    t.T[i][j].openORclose = true;
    t.deletedcount++;
    return 0;
}
```

3) Алгоритм поиска записи с заданным ключом в файле посредством хеш-таблицы.

Для того, чтобы найти запись по ключу в файле, нам необходимо сначала вызвать метод search, подав на вход функции хеш-таблицу и ключ, чтобы узнать координаты необходимой записи и впоследствии ее номер в файле. После, зная номер записи в файле, мы вызываем функцию FindRecord, подав на вход имя файла и номер записи.

Пример реализации алгоритма:

```
int FindByKey(int key, string fileName, HeshTable& t, product& result){
    int errorCode;
    int* coor;
    coor = search(t, key);
    int i=coor[0], j=coor[1];
    int n = t.T[i][j].record_number;
    errorCode = FindRecord(fileName, n, result);
    return errorCode;
}
```

## 1.4 Код программы

### Bin_Hash.h

```
#include "Binary.h"
#include "Hash.h"

// Прочитать запись из файла и вставить элемент в таблицу
int readAndInsertInHashTable(int number, string filename, HeshTable& t){
    product* record = new product();
    int errorCode;
    errorCode=FindRecord(filename, number, *record);
    if(errorCode == -1) return -1;
    // insertInHeshTable(int code, string name, string factory, int price,
string country, HeshTable& t)
    errorCode = insertInHeshTable(record->code, record->name, record->factory,
                                  record->price, record->country, number, t);
    if(errorCode == -1) return -1;
    return 0;
}

// Удалить запись из таблицы при заданном значении ключа и
//соответственно из файла
int deleteRecord(int key, string filename, HeshTable& t){
    int errorCode;
    errorCode=DelByKey(filename, key);
    if(errorCode == -1) return -1;
    errorCode=deletedFromHeshTable(t,key);
    if(errorCode == -1) return -1;
    return 0;
}
//Найти запись в файле по значению ключа (найти ключ в хеш-таблице,
//получить номер записи с этим ключом в файле, выполнить прямой доступ
//к записи по ее номеру)..

int FindByKey(int key, string fileName, HeshTable& t, product& result){
    int errorCode;
    int* coor;
    coor = search(t, key);
    int i=coor[0], j=coor[1];
    int n = t.T[i][j].record_number;
    errorCode = FindRecord(fileName, n, result);
    return errorCode;
}
```

### TestBinaryHash.h

```
#include "Bin_Hash.h"
#include "windows.h"

void testBinaryHash(){
    SetConsoleOutputCP(CP_UTF8);

    HeshTable T;
    T.createHeshTable();
    string fnameBin;
    string fnameText;
    string country;
    product* record = new product();
    int key;
    string line;

    int choice;
    int number;
    int errorcode;
    do{
        cout<<"1) Прочитать запись из файла и вставить элемент в таблицу\n";
        cout<<"2) Удалить запись из таблицы при заданном значении ключа и
соответственно из файла\n";
```

```
        cout<<"3) Найти запись в файле по значению ключа \n";
        cout<<"4) Вывести хеш-таблицу \n";
        cout<<"5) Вывести файл \n";
        cin >> choice;
        switch(choice){
            case 1:
                cout<<"Введите имя файла и номер записи:";
                cin >> fnameBin >> number;
                errorcode=readAndInsertInHashTable(number, fnameBin, T);
                if(errorcode==-1) cout<<"Error";
                break;
            case 2:
                cout<<"Введите имя файла и ключ:";
                cin >> fnameBin >> key;
                errorcode = deleteRecord(key, fnameBin, T);
                if(errorcode==-1) cout<<"Error";
                break;
            case 3:
                cout<<"Введите имя файла и ключ:";
                cin >> fnameBin >> key;
                errorcode = FindByKey(key, fnameBin, T, *record);
                if(errorcode==-1) cout<<"Error";
                else {
                    cout << record->name<<'\t';
                    cout << record->code<<'\t';
                    cout << record->factory<<'\t';
                    cout << record->price<<'\t';
                    cout << record->country;
                    cout << endl;
                }
                break;
            case 4:
                outTable(T);
                break;
            case 5:
                cout<<"Введите имя файла:";
                cin>>fnameBin;
                errorcode= out_bin_file(fnameBin);
                if(errorcode==-1) cout<<"Error";
                break;
            default:
                break;
        }
    }
    while(choice != -1);
};
```

## TestHash.h

```
#include "Hash.h"
void testHash(){
    HeshTable T;
    T.createHeshTable();
    insertInHeshTable(123, "name1", "factory1", 1, "country1", 1, T);  // 9
    insertInHeshTable(12, "name2", "factory2", 2, "country2", 2, T);   //12
    insertInHeshTable(19, "name3", "factory3", 3, "country3", 3, T);   //0
    insertInHeshTable(9, "name4", "factory4", 4, "country4",4, T);    //(9) 10
коллизия
    insertInHeshTable(28, "name5", "factory5", 5, "country5",5, T);   //(9) 11
коллизия
    cout<< "Table:\n";
    outTable(T);
    cout<< endl;
    typeitem r;

    int* coor = search(T, 9);
    int i=coor[0], j=coor[1];
```

```
    if (i!=-1 && j!= -1){
        r = T.T[i][j];
        cout << r.code << ' ' << r.name << endl;
    }
    else
        cout << "record is not" << '\n';


    i= deletedFromHeshTable(T, 9);
    if (i == 0)cout << "record is deleted\n";
    else
        cout << "record is not\n" << '\n';
    cout<<"Table:\n";
    outTable(T);
    coor = search(T, 28);
    i=coor[0], j=coor[1];
    if (i != -1) {
        r = T.T[i][j];
        cout << r.code << ' ' << r.name << endl;
    }
    else
        cout << "record is not" << '\n';


    cout<<"-----------------------------------------------";
    // добиваюсь рехеширования 15 / 19 =0,789473684210526, что больше 0.75
    insertInHeshTable(1, "name6", "factory6", 6, "country6", 6, T);      // 1
    insertInHeshTable(2, "name7", "factory7", 7, "country7",7, T);      //2
    insertInHeshTable(3, "name8", "factory8", 8, "country8",8, T);      //3
    insertInHeshTable(4, "name9", "factory9", 9, "country9",9, T);      //4
    insertInHeshTable(5, "name10", "factory10", 10, "country10",10, T);
//5
    insertInHeshTable(6, "name11", "factory11", 11, "country11",11, T);      //
6
    insertInHeshTable(7, "name12", "factory12", 12, "country12",12, T);
//7
    insertInHeshTable(8, "name13", "factory13", 13, "country13",13, T);
//8
    insertInHeshTable(10, "name14", "factory14", 14, "country14",14, T);
//10 (тк мы удаляли элемент)
    insertInHeshTable(11, "name15", "factory15", 15, "country15",15, T);
//(11) 13 коллизия
    insertInHeshTable(13, "name16", "factory16", 16, "country16",16, T);   //a
здесь уже требуется рехеширование тк кол-во элементов достигло 15

    cout<< "\nTable:\n";
    outTable(T);
}
```

**main.cpp**

```
#include "TestHash.h"
#include "TestBinary.h"
#include "TestBinaryHash.h"

int main()
{
    testHash();
    testBinary();
    testBinaryHash();
}
```

### 1.5 Результаты тестирования Хеш-таблицы

Пример заполненной таблицы:

```
Table:
0       19      name3   factory3        3       country3        0  0
1       0       -       -       0       -       1  0
2       0       -       -       0       -       1  0
3       0       -       -       0       -       1  0
4       0       -       -       0       -       1  0
5       0       -       -       0       -       1  0
6       0       -       -       0       -       1  0
7       0       -       -       0       -       1  0
8       0       -       -       0       -       1  0
9       123     name1   factory1        1       country1        0  0
9       9       name4   factory4        4       country4        0  0
9       28      name5   factory5        5       country5        0  0
10      0       -       -       0       -       1  0
11      0       -       -       0       -       1  0
12      12      name2   factory2        2       country2        0  0
13      0       -       -       0       -       1  0
14      0       -       -       0       -       1  0
15      0       -       -       0       -       1  0
16      0       -       -       0       -       1  0
17      0       -       -       0       -       1  0
18      0       -       -       0       -       1  0
```

Нахождение записи по ключу и удаление записи:

```
9 name4
record is deleted
Table:
0       19      name3   factory3        3       country3        0  0
1       0       -       -       0       -       1  0
2       0       -       -       0       -       1  0
3       0       -       -       0       -       1  0
4       0       -       -       0       -       1  0
5       0       -       -       0       -       1  0
6       0       -       -       0       -       1  0
7       0       -       -       0       -       1  0
8       0       -       -       0       -       1  0
9       123     name1   factory1        1       country1        0  0
10      0       -       -       0       -       1  0
11      0       -       -       0       -       1  0
12      12      name2   factory2        2       country2        0  0
13      0       -       -       0       -       1  0
14      0       -       -       0       -       1  0
15      0       -       -       0       -       1  0
16      0       -       -       0       -       1  0
17      0       -       -       0       -       1  0
18      0       -       -       0       -       1  0
28 name5
```

Пример рехеширования таблицы:

```
Table:
0       0       -       -       0       -       1  0
1       1       name6   factory6        6       country6        0  0
2       2       name7   factory7        7       country7        0  0
3       3       name8   factory8        8       country8        0  0
4       4       name9   factory9        9       country9        0  0
5       5       name10  factory10       10      country10       0  0
6       6       name11  factory11       11      country11       0  0
7       7       name12  factory12       12      country12       0  0
8       8       name13  factory13       13      country13       0  0
9       123     name1   factory1        1       country1        0  0
10      10      name14  factory14       14      country14       0  0
11      11      name15  factory15       15      country15       0  0
12      12      name2   factory2        2       country2        0  0
13      13      name16  factory16       16      country16       0  0
14      0       -       -       0       -       1  0
15      0       -       -       0       -       1  0
16      0       -       -       0       -       1  0
17      0       -       -       0       -       1  0
18      0       -       -       0       -       1  0
19      19      name3   factory3        3       country3        0  0
20      0       -       -       0       -       1  0
21      0       -       -       0       -       1  0
22      0       -       -       0       -       1  0
23      0       -       -       0       -       1  0
24      0       -       -       0       -       1  0
25      0       -       -       0       -       1  0
26      0       -       -       0       -       1  0
27      0       -       -       0       -       1  0
28      28      name5   factory5        5       country5        0  0
29      0       -       -       0       -       1  0
30      0       -       -       0       -       1  0
31      0       -       -       0       -       1  0
32      0       -       -       0       -       1  0
33      0       -       -       0       -       1  0
34      0       -       -       0       -       1  0
35      0       -       -       0       -       1  0
36      0       -       -       0       -       1  0
37      0       -       -       0       -       1  0
```

### 1.6 Тестирование операций управления файлом посредством хеш-таблицы

```
1) Прочитать запись из файла и вставить элемент в таблицу
2) Удалить запись из таблицы при заданном значении ключа и соответственно из файла
3) Найти запись в файле по значению ключа
4) Вывести хеш-таблицу
5) Вывести файл
1
Введите имя файла и номер записи:output.dat 1
```

```
1) Прочитать запись из файла и вставить элемент в таблицу
2) Удалить запись из таблицы при заданном значении ключа и соответственно из файла
3) Найти запись в файле по значению ключа
4) Вывести хеш-таблицу
5) Вывести файл
3
Введите имя файла и ключ:output.dat 123452
name1  123452  factory1        100     Russia
```

Удаление записи

```
1) Прочитать запись из файла и вставить элемент в таблицу
2) Удалить запись из таблицы при заданном значении ключа и соответственно из файла
3) Найти запись в файле по значению ключа
4) Вывести хеш-таблицу
5) Вывести файл
2
Введите имя файла и ключ:output.dat 123452
```

Хеш-таблица после удаления записи

```
0       0       -       -       0       -       1  0
1       0       -       -       0       -       1  0
2       0       -       -       0       -       1  0
3       0       -       -       0       -       1  0
4       0       -       -       0       -       1  0
5       0       -       -       0       -       1  0
6       0       -       -       0       -       1  0
7       0       -       -       0       -       1  0
8       0       -       -       0       -       1  0
9       123452  name1   factory1        100     Russia  1  1
10      0       -       -       0       -       1  0
11      0       -       -       0       -       1  0
12      0       -       -       0       -       1  0
13      0       -       -       0       -       1  0
14      0       -       -       0       -       1  0
15      0       -       -       0       -       1  0
16      0       -       -       0       -       1  0
17      0       -       -       0       -       1  0
18      0       -       -       0       -       1  0
```

Бинарный файл после удаления записи

```
Введите имя файла:output.dat
name5 123490  factory5        2340    Russia
name2   123455  factory2        1050    Czech
name3   123462  factory3        2300    China
name4   123471  factory4        2100    China
name5   123490  factory5        2340    Russia
```

**Вывод**

В ходе проделанной работы были получены навыки по разработке хеш-таблиц и их применении при поиске данных в других структурах данных (файлах).

**Используемая литература**


1. Материал к занятию 3.pdf
2. Практическая работа 3 Изменения в структре записи файла). (хеш-таблица для поиска записи в файле).pdf