МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт Информационных Технологий

Кафедра Вычислительной Техники

Отчет по выполнению практического задания №2

Тема. Хеширование и организация быстрого поиска данных

Дисциплина Структуры и алгоритмы обработки данных Часть 2

Студент группы: ИКБО-04-20 Нгуен Ван Мань

(Фамилия студента)

Преподаватель Сорокин А.В.

(Фамилия преподавателя)

Москва 2021

Оглавление

Тема: Хеширование и организация быстрого поиска данных.

Цель: Получить навыки по разработке хеш-таблиц и их применении.

Задание

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

Разработайте и реализуйте функции для операций:

1) Хеш-функцию (метод определите сами).

2) Прочитать запись из файла и вставки запись в таблицу (запись включает: ключ и номер записи с этим ключом в файле).

3) Удалить запись из таблицы и соответственно из файла.

4) Найти запись с заданным ключом в файле, используя хеш-таблицу.

5) Выполнить рехеширование.

2. Разработайте такие тесты, чтобы возникли коллизии.

3. Разработайте такие тесты, чтобы требовалось рехеширование.

4. Заполните файл большим количеством записей. Определите время чтения записи с заданным ключом для первой записи файла, для последней и где-то в середине. Убедитесь (или нет), что время доступа для всех записей одинаково.

5. Выведите список индексов, которые формируются при вставке элементов в таблицу.

Вариант:

6	Цепное хеширование	Товар: название, код – шестиразрядное число

Разработка программы

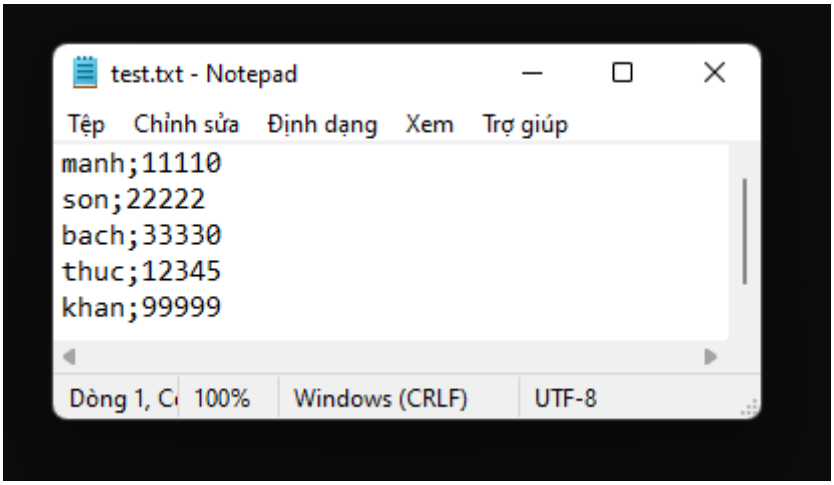
```
#include <list>
#include<fstream>
#include <iostream>
#include <stdlib.h>
#include <string>
#include <time.h>
#define CLOCKS_PER_SEC  1000000   /* [XSI] */

using  namespace std;

const int TABLE_SIZE = 10;

struct Product {
    string name;
    string id;
};

class Hash {
    list<Product>* table;

public:
    Hash() {
        table = new list<Product>[TABLE_SIZE];
    }
    int hashFunction(int x) {
        return (x % TABLE_SIZE);
    }

       void get_file(ifstream& fileIn, Product& prd) {
              getline(fileIn, prd.name, ';');
              getline(fileIn, prd.id);
       }

    void insertData(Product prd[],int& n) {
              ofstream fileOut;
              fileOut.open("C:/Users/taoth/test.txt");
              int number;

              cout << "\nHow many Product do you want to add?" << endl;
              cout << "Please input number: ";
              cin >> number;

              for (int i = 1; i <= number; i++) {
                     cin.ignore();
                     cout << "Input name of Product: " << i << ": ";
                     getline(cin, prd[i].name);
                     cout << "Input Product's number: ";
                     cin >> prd[i].id;
                     cout << endl;
                     int x = atoi(prd[i].id.c_str());
                     int j = hashFunction(x);
                     table[j].push_back(prd[i]);

                     fileOut << prd[i].name << ";" << prd[i].id << endl;
              }
       }



       void readFile() {
              ifstream fileIn;
              fileIn.open("C:/Users/taoth/test.txt");
```

```
		while (fileIn.eof() == false) {
			Product prd;
			get_file(fileIn, prd);
			int x = atoi(prd.id.c_str());
			int j = hashFunction(x);
			table[j].push_back(prd);
		}
	}



	void deleteData(ifstream &fileIn) {
		int readerNumber;
		cout << "Enter Product number to delete: ";
		cin >> readerNumber;
		int index = hashFunction(readerNumber);
		list<Product>::iterator i;
		for (i = table[index].begin(); i != table[index].end(); ++i) {
			if (atoi(i->id.c_str()) == readerNumber) {
				break;
			}
		}
		if (i != table[index].end()) {
			cout << "Reader " << i->name << " has been removed from list!\n";
			table[index].erase(i);
		}
		else {
			cout << "Product number is not in this list!\n";
		}


		ofstream nf;
		nf.open("C:/Users/taoth/temp.txt");
		Product pr1;
		get_file(fileIn, pr1);
		while (!fileIn.eof()) {
			if (pr1.id != to_string(readerNumber)) {
				nf << pr1.name << ";" << pr1.id << endl;
			}
			get_file(fileIn, pr1);
		}
		nf.close();
		fileIn.close();
		remove("C:/Users/taoth/test.txt");
		rename("C:/Users/taoth/temp.txt", "C:/Users/taoth/test.txt");

	}


void search() {
		int readerNumber;
		cout << "Enter Product number to find: ";
		cin >> readerNumber;
		int index = hashFunction(readerNumber);

		list<Product>::iterator i;
		bool check = false;
		for (i = table[index].begin(); i != table[index].end(); ++i) {
			if (i->id == to_string(readerNumber)) {
				check = true;
				cout << "Record found:\n";
				cout << "\tProduct Number\t\tName\n";
				cout << "\t" << i->id << "\t\t\t" << i->name << "\n";
```

```
                }
            }

            if (!check) {
                cout << "Cannot find this Product!\n";
            }
        }



    void displayHash() {
            cout << "Index\t\tProduct Name\n";
            for (int i = 0; i < 10; ++i) {
                cout << i << ": ";
                for (auto x : table[i]) {
                    cout << " --> " << x.name;
                }
                cout << endl;
            }
        }
};



int main()
{
    Product prd[100];
    int n;
    ifstream in("C:/Users/taoth/test.txt", ios::in);
    if (!in) {
        cerr << "File can't be opened! " << endl;
        system("PAUSE");
        exit(1);
    }


       clock_t start, end;
       double time_used;

       Hash t = Hash();
       cout << "------------------------Menu-----------------------\n";
       cout << "[1] - Insert new Product\n";
       cout << "[2] - ReadFile\n";
       cout << "[3] - Delete a Product\n";
       cout << "[4] - Find a Product\n";
       cout << "[5] - Display hash table\n";
       cout << "[0] - Exit program\n";
       cout << "----------------------------------------------------\n";

       int choice = 0;
       do
       {
            cout << "Input your choice: ";
            cin >> choice;
            cout << "----------------------------------------------------\n";

            switch (choice)
            {
            case 1:
                t.insertData(prd, n);
                cout << "Done!\n";
                break;
            case 2:
                t.readFile();
```

```
                break;
        case 3:
                t.deleteData(in);
                break;
        case 4:
                start = clock();
                t.search();
                end = clock();
                time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
                cout << "time: " << time_used << endl;
                break;
        case 5:
                t.displayHash();
                break;
        case 0:
                cout << "Thanks for using program!\n";
                break;
        default:
                break;
        }
        cout << "-----------------------------------------------------\n";
} while (choice != 0);

system("pause");

in.close();
return 0;
                                        }
```

File

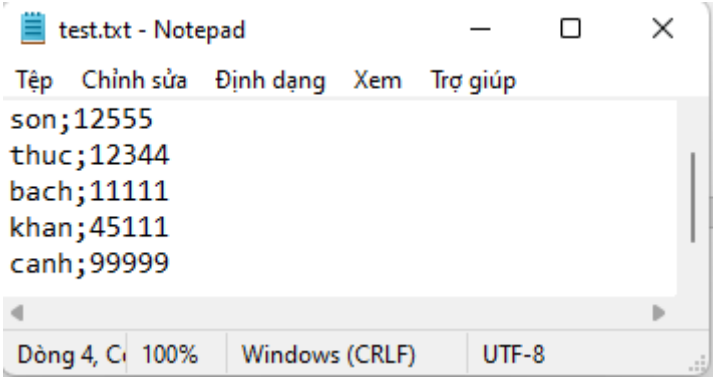


test.txt

Тесты

```
C:\Users\taoth\source\repos\Hashing\Debug\hashing_1.exe
------------------------Menu------------------------
[1] - Insert new Product
[2] - ReadFile
[3] - Delete a Product
[4] - Find a Product
[5] - Display hash table
[0] - Exit program
----------------------------------------------------
Input your choice: 2
----------------------------------------------------
----------------------------------------------------
Input your choice: 5
----------------------------------------------------
Index           Product Name
0:  --> manh --> bach -->
1:
2:  --> son
3:
4:
5:  --> thuc
6:
7:
8:
9:  --> khan
----------------------------------------------------
Input your choice: 3
----------------------------------------------------
Enter Product number to delete: 33330
Reader bach has been removed from list!
----------------------------------------------------
Input your choice: 5
----------------------------------------------------
Index           Product Name
0:  --> manh -->
1:
2:  --> son
3:
4:
5:  --> thuc
6:
7:
8:
9:  --> khan
----------------------------------------------------
Input your choice: 4
----------------------------------------------------
Enter Product number to find: 12345
Record found:
        Product Number          Name
        12345                   thuc
----------------------------------------------------
Input your choice: 0
----------------------------------------------------
Thanks for using program!
----------------------------------------------------
Press any key to continue . . .
```

Определите время чтения записи с заданным ключом для первой записи файла, для последней и где-то в середине. (секунда)




```
C:\Users\taoth\source\repos\Hashing\Debug\hashing_1.exe
-----------------------------------------------------
Index           Product Name
0:  -->
1:  --> bach --> khan
2:
3:
4:  --> thuc
5:  --> manh --> son
6:
7:
8:
9:  --> canh
-----------------------------------------------------
Input your choice: 4
-----------------------------------------------------
Enter Product number to find: 11111
Record found:
        Product Number          Name
        11111                   bach
time: 0.008058
-----------------------------------------------------
Input your choice: 4
-----------------------------------------------------
Enter Product number to find: 99999
Record found:
        Product Number          Name
        99999                   canh
time: 0.003707
-----------------------------------------------------
Input your choice: ^SSS
```

Выводы

В результате проделанной работы, я получить навыки по разработке хештаблиц и их применении. В процессе было разработано приложение, которое использует хеш-таблицу для организации прямого доступа к записям файла.

Ответы на вопросы

1. Расскажите о назначении хеш-функции.

Алгоритм хеш-функции преобразует уникальный ключ записи в её индекс в таблице.

2. Что такое коллизия?

Коллизия – это ситуация, когда разным ключам соответствует одно значение хеш-функции.

3. Что такое «открытый адрес» по отношению к хеш-таблице?

В хеш-таблице элементы могут содержать только пары ключ-значение или ключ - ссылка на значение, где Открытый адрес – это свободная ячейка хештаблицы, закрытый адрес – это занятая ячейка.

4. Как в хеш-таблице с открытым адресом реализуется коллизия?

В случае коллизии происходит смещение на ограниченное кол-во ячеек, определяемое определёнными алгоритмами или просто константа(напр. =1)

5. Какая проблема, может возникнуть после удаления элемента из хеш-таблицы с открытым адресом и как ее устранить?

Выполнение операции удаления записи из хеш-таблицы с открытым адресом имеет свои трудности и связано это с тем, что при удалении нельзя делать ячейку открытой, так как другие ключи при разрешении коллизий были вставлены после удаляемой и поиск свободной выполнялся при закрытой ячейке. Решением этой проблемы является добавление метки “удалено”. А очистка от “удалённых” элементов возможно только при рехешировании.

6. Что определяет коэффициент нагрузки в хеш-таблице?

Кол-во записей в таблице / максимальный размер таблицы. Обычно это значение не должно превышать 0,75.

7. Что такое «первичный кластер» в таблице с открытым адресом?

Первичный кластер(или первичная кластеризация) - это явление, которое происходит, когда обработчик коллизий создает условие роста кластера. Обработчик коллизий со смещением 1 из таблицы с открытым адресом способствует первичной кластеризации. Первичная кластеризация порождает длинные пути.

8. Как реализуется двойное хеширование?

Помимо хеширования ключа, идёт хеширования смещения для устранения первичной кластеризации. Все элементы хранятся непосредственно в хеш-таблице, без использования связных списков. В отличие от хеширования с цепочками, при использовании этого метода может возникнуть ситуация, когда хеш-таблица окажется полностью заполненной, следовательно, будет невозможно добавлять в неё новые элементы. Так что при возникновении такой ситуации решением может быть динамическое увеличение размера хеш-таблицы, с одновременной её рехешированием.