

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №5
по дисциплине «Алгоритмы и структуры данных»
Тема: БДП и Хеш-таблицы

Студент гр. 9382

Савельев И.С.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

Реализовать хеш-таблицу методом цепочек на C++.

Основные теоретические положения.

Хеш-таблица (англ. *hash-table*) — структура данных, реализующая интерфейс ассоциативного массива. Существует два основных вида хеш-таблиц: *с цепочками* и *открытой адресацией*. Хеш-таблица содержит некоторый массив H , элементы которого есть пары (хеш-таблица с открытой адресацией) или списки пар (хеш-таблица с цепочками). Выполнение операции в хеш-таблице начинается с вычисления хеш-функции от ключа. Хеш-код $i=h(key)$ играет роль индекса в массиве H , а зная индекс, мы можем выполнить требующуюся операцию (добавление, удаление или поиск). Количество коллизий зависит от хеш-функции; чем лучше используемая хеш-функция, тем меньше вероятность их возникновения.

Задание.

Вариант 23

Хеш-таблица: с цепочками; действие: $1+2a$

- 1) По заданной последовательности элементов $Elem$ построить структуру данных определённого типа – БДП или хеш-таблицу;
- 2) Выполнить одно из следующих действий:
 - а) Для построенной структуры данных проверить, входит ли в неё элемент e типа $Elem$, и если входит, то в скольких экземплярах. Добавить элемент e в структуру данных. Предусмотреть возможность повторного выполнения с другим элементом.
 - б) Для построенной структуры данных проверить, входит ли в неё элемент e типа $Elem$, и если входит, то удалить элемент e из структуры данных (первое обнаруженное вхождение). Предусмотреть возможность повторного выполнения с другим элементом.

Функции и СД.

Функция `string readFile(const string& file_name)` получает на вход строку с названием файла создает экземпляр класса `ifstream f`, которому передает в качестве аргумента `file_name`, затем создается экземпляр класса `stringstream ss` и с помощью метода `rdbuf()` класс `ifstream`, потоковый буфер передается `ss`. Функция возвращает строку с помощью метода `str()` класс `stringstream`.

Класс `Hash` использует шаблонный тип `T` по умолчанию равный `string` имеет две `private` переменные `vector<vector<T>> table`, `int size=10` и следующие `public` методы.

`Hash()` конструктор класс `Hash`, в котором с помощью метода `reserve()` выделяется место под `size` массивов.

Метод `int hashFunction(T value)`, принимает шаблонный тип `value`, внутри метода создается переменная счетчик `s`, в которой с помощью цикла `for` суммируются `ASCII` коды каждого символа элемента. Метод возвращает `s` по модулю размера таблицы или другими словами ключ элемента.

Метод `void insertElem(T value)`, принимает шаблонный тип `value`, который передается в `hashFunction()` и возвращаемое ею значение записывается в переменную `key`, после чего с помощью метода `push_back` в соответствующую ячейку таблицы со строкой `key` записывается значение `value`.

Метод `int findElem(T value)`, принимает шаблонный тип `value`, который передается в `hashFunction()` и возвращаемое ею значение записывается в переменную `key` и создается переменная счетчик `count`. Затем в цикле `for` обходится строка таблицы со значением `key` и при совпадении столбца таблицы и значения `value`, `count` увеличивается на один. Метод возвращает количество элементов `value` в таблицы.

Метод `void displayHash()` выводит таблицу в консоль с помощью двух циклов `for`

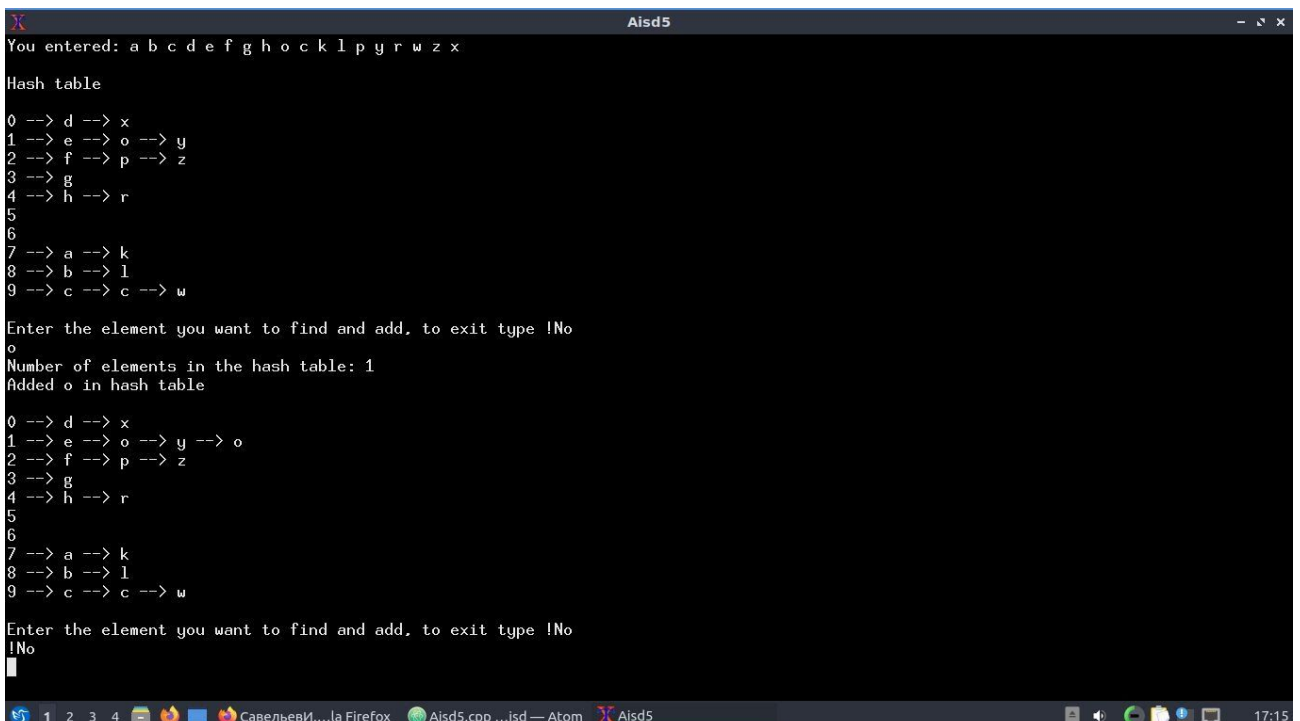
в первом(внешнем) обходятся все строки таблицы, а во втором(внутреннем) все столбцы соответствующей строки таблицы.

Алгоритм.

В начале работы программы ей на вход подается строка считанная из файла. Она разбивается на элементы, которые передаются в хэш-таблицу, где каждому с помощью хеш функции присваивается ключ и он помещается в соответствующую цепочку хеш-таблицы. Далее выводится строка считанная из файла и получившаяся хеш-таблица. Затем пользователю предлагается либо завершить программу, либо ввести элемент, который он хочет добавить в таблицу и подсчитать количество его вхождений в неё. Если пользователь выбрал второй вариант, то выводится сообщение с количеством данного элемента в таблице и сама таблица, пользователю опять предлагают ввести элемент или завершит программу. Для поиска элемента в таблице создается его ключ, указывающий на определенную цепочку в хеш-таблице, которая перебирается в поисках элемента. При добавлении нового элемента в хеш-таблицу для него также создается ключ и он помещается в соответствующую цепочку.

Тестирование.

тест 1



```
Aisd5
You entered: a b c d e f g h o c k l p y r w z x
Hash table
0 --> d --> x
1 --> e --> o --> y
2 --> f --> p --> z
3 --> g
4 --> h --> r
5
6
7 --> a --> k
8 --> b --> l
9 --> c --> c --> w

Enter the element you want to find and add, to exit type !No
o
Number of elements in the hash table: 1
Added o in hash table

0 --> d --> x
1 --> e --> o --> y --> o
2 --> f --> p --> z
3 --> g
4 --> h --> r
5
6
7 --> a --> k
8 --> b --> l
9 --> c --> c --> w

Enter the element you want to find and add, to exit type !No
!No
```

тест 2

```
Aisd5
You entered: 1 2 3 4 5 6 7 8 9

Hash table
0 --> 2
1 --> 3
2 --> 4
3 --> 5
4 --> 6
5 --> 7
6 --> 8
7 --> 9
8
9 --> 1

Enter the element you want to find and add, to exit type !No
0
Number of elements in the hash table: 0
Added 0 in hash table

0 --> 2
1 --> 3
2 --> 4
3 --> 5
4 --> 6
5 --> 7
6 --> 8
7 --> 9
8 --> 0
9 --> 1

Enter the element you want to find and add, to exit type !No
!No
```

тест 3

```
Aisd5
You entered: Hello_Helen Bod Sad Cat Dog

Hash table
0 --> Sad --> Cat
1
2 --> Dog
3
4
5
6
7 --> Hello_Helen --> Bod
8
9

Enter the element you want to find and add, to exit type !No
67
Number of elements in the hash table: 0
Added 67 in hash table

0 --> Sad --> Cat
1
2 --> Dog
3
4
5
6
7 --> Hello_Helen --> Bod
8
9 --> 67

Enter the element you want to find and add, to exit type !No
!No
```

тест 4

```
Aisd5
You entered:
Hash table
0
1
2
3
4
5
6
7
8
9
Enter the element you want to find and add, to exit type !No
666
Number of elements in the hash table: 0
Added 666 in hash table
0
1
2 --> 666
3
4
5
6
7
8
9
Enter the element you want to find and add, to exit type !No
█
```

тест 5

```
Aisd5
You entered: #$$%^&* $^&^*& adadad 1262 221.2213 HDD
Hash table
0
1 --> adadad
2
3 --> 1262
4
5 --> 221.2213
6
7
8 --> HDD
9 --> #$$%^&* --> $^&^*&
Enter the element you want to find and add, to exit type !No
221.2213
Number of elements in the hash table: 1
Added 221.2213 in hash table
0
1 --> adadad
2
3 --> 1262
4
5 --> 221.2213 --> 221.2213
6
7
8 --> HDD
9 --> #$$%^&* --> $^&^*&
Enter the element you want to find and add, to exit type !No
█
```

тест 6

```
Aisd5
You entered: 5 5 5 5 5 7 7 7 7
Hash table
0
1
2
3 --> 5 --> 5 --> 5 --> 5
4
5 --> 7 --> 7 --> 7 --> 7
6
7
8
9
Enter the element you want to find and add, to exit type !No
5
Number of elements in the hash table: 5
Added 5 in hash table
0
1
2
3 --> 5 --> 5 --> 5 --> 5 --> 5
4
5 --> 7 --> 7 --> 7 --> 7
6
7
8
9
Enter the element you want to find and add, to exit type !No
|
```

тест 7

```
Aisd5
You entered: 1 3 da net
Hash table
0
1 --> 3
2
3
4
5
6
7 --> da --> net
8
9 --> 1
Enter the element you want to find and add, to exit type !No
5
Number of elements in the hash table: 0
Added 5 in hash table
0
1 --> 3
2
3 --> 5
4
5
6
7 --> da --> net
8
9 --> 1
Enter the element you want to find and add, to exit type !No
2
Number of elements in the hash table: 0
Added 2 in hash table
0 --> 2
1 --> 3
2 --> 5
3
4
5
6
7 --> da --> net
8
9 --> 1
Enter the element you want to find and add, to exit type !No
|
```

Выводы.

В ходе выполнения данной лабораторной работы была реализована хеш-таблица с цепочками на языке программирования C++.

ИСХОДНЫЙ КОД ПРОГРАММЫ.

```
#include<iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include <iterator>

#define FILE "/home/indiora/C++/Aisd/read.txt"

using namespace std;

string readFile(const string& file_name) {
    ifstream f(file_name); // Экземпляр класса
    stringstream ss;
    ss << f.rdbuf(); // Возвращает указатель на потоковый буфер
    return ss.str(); // Возвращает строку
}

template <typename T=string>
class Hash {
private:
    vector<vector<T>> table;
    int size=10;
public:
    // Конструктор
    Hash() {
        table.reserve(size);
    }
    // Хеш функция реализованная адаптивным методом
    int hashFunction(T x) {
        int s = 0;
        for(int i = 0 ; i < x.length(); i++)
            s += x[i];
        return s % size;
    }
    // Вставка elem
    void insertElem(T value) {
        // Расчитываем ключ от значения
        int key = hashFunction(value);
        // Добавляем в соответствующую ячейку таблицы
        table[key].push_back(value);
    }
    // Поиск elem
    int findElem(T value) {
        // Расчитываем ключ от значения
        int key = hashFunction(value);
        int count = 0;
        // В цикле обходим цепочку с соответствующим ключом
        for (typename vector<T>::iterator i = begin(table[key]); i !=
end(table[key]); ++i) {
            // При нахождении elem увеличиваем счетчик
            if (*i == value){
                count++;
            }
        }
        // Возвращаем количество элементов
        return count;
    }
}
```



```

void displayHash() {
    // Обходим всю таблицу
    for (int i = 0; i < size; i++) {
        cout << i;
        // Выводим элементы цепочки
        for (auto x : table[i]) {
            cout << " --> " << x;
        }
        cout << "\n";
    }
    cout << "\n";
}

};

int main(int argc, char const *argv[]) {
    // Создаем экземпляр класса Hash
    Hash table;
    // Создаем экземпляр класса iss.. и передаем ему строку считанную из файла
    istream iss(readFile(FILE));
    // Разбиваем считанную строку на Elem
    vector<string> tokens{istream_iterator<string>{iss},
    istream_iterator<string>{}};
    // Заносим Elem в хэш таблицу
    for(int i = 0; i < tokens.size(); i++){
        table.insertElem(tokens[i]);
    }
    // Выводим строку считанную из файла
    cout << "You entered: ";
    cout << iss.str() << '\n';
    // Выводим хэш таблицу
    cout << "Hash table" << "\n\n";
    table.displayHash();

    string input;

    while( input != "!No"){
        cout << "Enter the element you want to find and add, to exit type !No" <<
"\n";
        // Считываем значение введенное пользователем
        cin >> input;
        if (input != "!No") {
            // Выводим количество elem в таблице
            cout << "Number of elements in the hash table: " <<
table.findElem(input) << "\n";
            cout << "Added " << input << " in hash table"<< "\n\n";
            // Вставляем elem в таблицу
            table.insertElem(input);
            // Выводим таблицу
            table.displayHash();
        }
    }

    return 0;
}

```