

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

КУРСОВАЯ РАБОТА
по дисциплине «Алгоритмы и структуры данных»
Тема: Хеш-таблицы с цепочками – вставка и исключение.

Студент гр. 9382

Савельев И.С.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2020

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Савельев И.С.

Группа 9382

Тема работы : Хеш-таблицы с цепочками – вставка и исключение.

Исходные данные:

На вход программе подается последовательность элементов разделенных пробелами.

Содержание пояснительной записки:

«Содержание», «Введение», «Задание», «Ход выполнения работы»,
«Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 31.10.2020

Дата сдачи реферата: 29.11.2020

Дата защиты реферата: 29.11.2020

Студент

Савельев И.С.

Преподаватель

Фирсов М.А.

АННОТАЦИЯ

В курсовой работе реализована хеш-таблица с цепочками. Программа демонстрирует вставку и удаление из хеш-таблицы элементов введенных пользователем с консоли. В ходе выполнения программ каждый шаг сопровождается объяснениями, которые выводятся в терминал.

SUMMARY

In the course work, a hash table with chains is implemented. The program demonstrates the insertion and removal from the hash table of elements entered by the user from the console. During the execution of programs, each step is accompanied by explanations that are displayed in the terminal.

СОДЕРЖАНИЕ

	Введение	4
1.	Задание	6
2.	Ход выполнения работы	6
2.1.	Описание алгоритма	6
2.2.	Ход работы	6
2.2.	Описание структур данных и функций	7
2.3.	Описание интерфейса пользователя	8
2.4.	Тестирование	9
	Заключение	13
	Список использованных источников	14
	Приложение А. Название приложения	14

ВВЕДЕНИЕ

Целью работы являлось изучить такую структуру данных, как хеш-таблица. И реализовать ее с использованием цепочек. Для это пришлось изучить алгоритм построения хеш-таблицы, алгоритмы добавления и исключения элементов из неё. Результатом является программа, которая получает на вход последовательность элементов и создает хеш таблицу из них, также пользователь может добавить или исключить элемент из таблицы, все действия сопровождаются визуализацией.

1. ЗАДАНИЕ

Вариант 23.

Хеш-таблицы с цепочками – вставка и исключение. Демонстрация.

2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1. Описание алгоритма

В начале работы программы в хэш-таблицу передаются элементы, каждому с помощью хеш функции присваивается ключ и он помещается в соответствующую цепочку хеш-таблицы. Хеш функция получает элемент и в цикле проходится по его символам суммируя их коды из таблицы ASCII в одной переменной, функция возвращает остаток от деления суммы элементов на размер таблицы, это сделано для того, чтобы функция не вернула номер цепочки, которой нет в таблице. Затем пользователю предлагается либо завершить программу, либо добавить элемент в таблицу, либо удалить элемент. Для удаления элемента из таблицы 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.

2.2. Ход работы

При запуске 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 завершить программу, добавить элемент в таблицу, удалить элемент. Если пользователь выбрал первый вариант, то программа завершается и выводится сообщение сообщающее об этом. Если пользователь выбрал второй вариант, то элемент добавляется в таблицу выводится промежуточная информация,

пользователю опять предлагается выбрать, что сделать. Если пользователь выбрал третий вариант, то элемент удаляется из таблицы, если он там есть, выводится промежуточная информация, пользователю опять предлагается выбрать, что сделать.

2.3. Описание структур данных и функций

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 howHash(T value)`, принимает шаблонный тип `value`, в цикле суммирует его символы из таблицы ASCII и выводит их на экран, в конце работы он выводит ключ цепочки в которую поместят `value`, метод предназначен для вывода промежуточной информации.

Метод `void howInsert(T value)` принимает шаблонный тип `value`, который передается в `hashFunction()` и возвращаемое ею значение записывается в переменную `key`, метод предназначен для вывода в терминал информации поясняющей работу программы.

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 `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.

Метод `void deleteElem(T value)` принимает шаблонный тип `value`, который передается в `hashFunction()` и возвращаемое ею значение записывается в переменную `key`, после чего с помощью цикла в соответствующей цепочки таблицы со значением `key` перебираются все элементы и при нахождении элемента соответственно равного `value` он удаляется с помощью метода `erase()` цикл прерывается с помощью `break`.

Метод `void howDelete(T value)` принимает шаблонный тип `value`, который передается в `hashFunction()` и возвращаемое ею значение записывается в переменную `key`, метод предназначен для вывода в терминал информации поясняющей работу программы.

2.4. Описание интерфейса пользователя

Пользователь вводит в отдельный файл через пробел элементы, которые он хочет видеть в таблице изначально, затем в начале выполнения программы в терминал выводится строка пользователя считанная из файла и хеш-таблица построенная по этой строке. После чего пользователю предлагается выбрать либо закончить программу для это надо ввести в консоль 3, либо вставить элемент для это надо ввести в консоль 2, либо удалить элемент из таблицы для это надо ввести в консоль 1, если пользователь выбрал первый вариант то программа завершается с соответствующим сообщением, если второй или третий то пользователя предлагается ввести элемент который он хочет вставить или удалить, после считывания с консоли ввода пользователя, выводится промежуточная информация, а затем и сама хеш-таблица.

2.5. Тестирование

1) Входная строка: cat sad I_am bob_nop round what

```
You entered: sad cat Bob_top hey_Bill dog happy
```

For each of these elements, the key is calculated using a hash function and it is placed in the corresponding cell of the table.

Hash table

```
0
1
2 --> sad --> cat
3
4 --> dog
5
6 --> happy
7
8 --> hey_Bill
9 --> Bob_top
```

```
To remove type : "1"
To insert type : "2"
To leave type : "3"
```

Your input :

2) Входная строка: 12 54.31 814 4.444 124.99

```
You entered: 12 54.31 814 4.444 124.99
```

For each of these elements, the key is calculated using a hash function and it is placed in the corresponding cell of the table.

Hash table

```
0
1 --> 54.31 --> 124.99
2
3
4 --> 4.444
5
6
7 --> 814
8
9 --> 12
```

```
To remove type : "1"
To insert type : "2"
To leave type : "3"
```

Your input :

3) Входная строка:

```
You entered:

For each of these elements, the key is calculated using a hash function and it is placed in the corresponding cell of the table.

Hash table

0
1
2
3
4
5
6
7
8
9

To remove type : "1"
To insert type : "2"
To leave type : "3"

Your input : █
```

4) Удаление элемента: cat

```
To remove type : "1"
To insert type : "2"
To leave type : "3"

Your input : 1
You want delete : cat

For this we have to find its key using a hash function.
In the loop we go through the element summing up its symbols from ASCII
Sum 1 symbol(s) 99
Sum 2 symbol(s) 196
Sum 3 symbol(s) 312
Modulo division of the sum by the size of the table, and we get the key : 2

When we find the key we just need to delete the "cat" from the 2 chain of the table.
To do this in the loop we go through the 2 chain in search of a "cat" and if we meet it we delete it.

Hash table after delete

0
1
2 --> sad
3
4 --> dog
5
6 --> happy
7
8 --> hey_Bill
9 --> Bob_top
```

5) Удаление элемента: 4.444

```

Your input : 1
You want delete : 4.444

For this we have to find its key using a hash function.
In the loop we go through the element summing up its symbols from ASCII
Sum 1 symbol(s) 52
Sum 2 symbol(s) 98
Sum 3 symbol(s) 150
Sum 4 symbol(s) 202
Sum 5 symbol(s) 254
Modulo division of the sum by the size of the table, and we get the key : 4

When we find the key we just need to delete the "4.444" from the 4 chain of the
table.
To do this in the loop we go through the 4 chain in search of a "4.444" and if w
e meet it we delete it.

Hash table after delete

0
1 --> 54.31 --> 124.99
2
3
4
5
6
7 --> 814
8
9 --> 12

```

6) Вставка элемента: hey

```

Your input : 2
You want insert : hey

For this we have to find its key using a hash function.
In the loop we go through the element summing up its symbols from ASCII
Sum 1 symbol(s) 104
Sum 2 symbol(s) 205
Sum 3 symbol(s) 326
Modulo division of the sum by the size of the table, and we get the key : 6

When we find the key we just need to insert the "hey" into the 6 chain of the ta
ble.

Hash table after insert

0
1
2 --> sad --> cat
3
4 --> dog
5
6 --> happy --> hey
7
8 --> hey_Bill
9 --> Bob_top

```

7) Завершение работы программы

You entered: sad cat Bob_top hey_Bill dog happy

For each of these elements, the key is calculated using a hash function and it is placed in the corresponding cell of the table.

Hash table

```
0
1
2 --> sad --> cat
3
4 --> dog
5
6 --> happy
7
8 --> hey_Bill
9 --> Bob_top
```

To remove type : "1"

To insert type : "2"

To leave type : "3"

Your input : 3

Program end !

ЗАКЛЮЧЕНИЕ

В ходе работы над курсовой работой была изучена такая структура данных, как хеш-таблица с цепочками. Также была реализована программа, которая позволяет продемонстрировать вставку и удаление элементов из хеш-таблицы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Справочная информация о языке программирования C++ // The C++ Resources Network // URL: <https://www.cplusplus.com> (дата обращения: 27.11.2020)

ПРИЛОЖЕНИЕ А

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;
    // размер таблицы
    int size=10;
public:
    // Конструктор
    Hash() {
        table.reserve(size);
    }
    // Хеш функция реализованная адаптивным методом
    int hashFunction(T value) {
        int s = 0;
        for(int i = 0 ; i < value.length(); i++)
            s += value[i];
        // возвращает ключ
        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);
    }
    // Пояснения к хеш функции
    void howHash(T value) {
        int s = 0;
        std::cout << "In the loop we go through the element summing up its symbols
from ASCII" << '\n';
        for(int i = 0 ; i < value.length(); i++) {
            s += value[i];
            std::cout << "Sum " << i + 1 << " symbol(s) " << s << '\n';
        }
        std::cout << "Modulo division of the sum by the size of the table, and we
get the key : ";
        std::cout << "\x1b[32m" << s % size << "\x1b[0m" << '\n';
        std::cout << '\n';
    }
    // Пояснения для вставки
```

```

void howInsert(T value) {
    // Расчитываем ключ от значения
    int key = hashFunction(value);
    std::cout << "\nFor this we have to find its key using a hash function." <<
'\n';
    // Пояснения для ключа
    howHash(value);
    std::cout << "When we find the key we just need to insert the \"" <<
"\x1b[4;32m" << value << "\x1b[0m" << "\" into the " << "\x1b[32m" << key <<
"\x1b[0m" << " chain of the table." << "\n\n";
}
// Пояснения для удаления
void howDelete(T value) {
    // Расчитываем ключ от значения
    int key = hashFunction(value);
    std::cout << "\nFor this we have to find its key using a hash function." <<
'\n';
    // Пояснения для ключа
    howHash(value);
    std::cout << "When we find the key we just need to delete the \"" <<
"\x1b[4;32m" << value << "\x1b[0m" << "\" from the " << "\x1b[32m" << key <<
"\x1b[0m" << " chain of the table." << '\n';
    std::cout << "To do this in the loop we go through the " << "\x1b[32m" <<
key << "\x1b[0m" << " chain in search of a \"" << "\x1b[4;32m" << value <<
"\x1b[0m" << "\" and if we meet it we delete it." << "\n\n";
}
// Удаление элемента
void deleteElem(T value) {
    // получаем хеш-индекс ключа
    int key = hashFunction(value);
    // В цикле обходим цепочку с соответствующим ключом
    for (typename vector<T>::iterator i = begin(table[key]); i !=
end(table[key]); ++i) {
        // При нахождении элем удаляем его и прерываем цикл
        if (*i == value){
            table[key].erase(i);
            break;
        }
    }
}

void displayHash() {
    // Обходим всю таблицу
    cout << "\n";
    for (int i = 0; i < size; i++) {
        cout << "\x1b[36m" << i << "\x1b[0m";
        // Выводим элементы цепочки
        for (auto x : table[i]) {
            cout << "\x1b[35m --> \x1b[0m" << x;
        }
        cout << "\n";
    }
    cout << "\n";
}

};

int main(int argc, char const *argv[]) {
    // Создаем экземпляр класса Hash
    Hash table;
    // Создаем экземпляр класс is.. 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 << "\n\nYou entered: \n" << iss.str() << "\n";
        std::cout << "For each of these elements, the key is calculated using a hash
function and it is placed in the corresponding cell of the table." << '\n';
        // Выводим хэш таблицу
        cout << "\n\nHash table\n" << "\n";
        table.displayHash();

        string input;
        while(true) {
            cout << "\n\nTo remove type : \"1\"\n"
                "\n\nTo insert type : \"2\"\n"
                "\n\nTo leave type : \"3\"\n\n";
            // Считываем значение введенное пользователем
            std::cout << "Your input : ";
            cin >> input;
            // Для удаления элемента
            if (input == "1") {
                std::cout << "You want delete : ";
                cin >> input;
                table.howDelete(input);
                table.deleteElem(input);
                std::cout << "\n\nHash table after delete\n" << '\n';
                table.displayHash();
            }
            // Для добавления элемента
            else if (input == "2") {
                std::cout << "You want insert : ";
                cin >> input;
                table.howInsert(input);
                table.insertElem(input);
                std::cout << "\n\nHash table after insert\n" << '\n';
                table.displayHash();
            }
            // Для выхода
            else if (input == "3") {
                break;
            }
            // Иначе
            else {
                std::cout << "\n\nIncorrect input please try again." << '\n';
            }
        }
        std::cout << "\n\nProgram end !" << '\n';
        return 0;
    }
}

```