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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Алгоритмы и структуры данных »

Тема: «Хэш таблицы и декодирование»

Студент гр. 9384		Давыдов Д.С.
Преподаватель		Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Добавить введенные пользователем хэш коды и закодированные ими символы в хэш таблицу, а затем декодировать хэш коды введенные пользователем из файла или консоли.

Задание.

Вариант 4.

Декодирование: статическое, коды символов хранятся в хеш-таблице без коллизий.

Анализ задачи.

Была реализована такая структура данных, как хеш-таблица.

Хэш таблица представляет из себя класс, приватное поле которого — массив связных списков (указателей на структуру `linkedList`), который как раз представляет из себя хэш таблицу. Структура имеет 4 поля: `T data` (закодированные данные), `string hash` (хэш код, который пользователь получил при кодировании информации), `linkedList* next` (указатель на следующий элемент связного списка), `bool exist` (флаг который показывает является ли элемент массива пустым, т.е. связного списка не существует).

В коде представлено 2 хэш функции — одна рассчитывает принять на вход хэш код в виде строчки из всевозможных символов (H1INM6 к примеру), а другая бинарный код, к примеру 10111. Первая возвращает ключ или же индекс массива путем сложения ascii кодов всех символов хэш кода а затем получения остатка деления на длину массива, а вторая возвращает ключ переводя бинарный код в десятичное число и также получая его остаток при делении на длину массива.

Если пользователь ввел два одинаковых хэш кода, то останется только тот символ (из этих хэш кодов), который закодирован последним из введенных одинаковых хэш кодов.

Если происходит коллизия (ключи, которые вернула хэш функция для

двух и более хэш кодов, совпадают), то в конец связного списка у элемента массива с индексом этого ключа добавляется новая пара хэш код—информация. Тогда, если хэш код элемента массива с индексом равным ключу не совпадает с хэш кодом, который хочет декодировать пользователь, то в цикле while мы идем до последнего элемента связного списка, пока не найдем тот же хэш код. Когда мы его находим — возвращаем поле data найденного `linkedList` с совпавшим хэш кодом.

Программа допускает, что один символ имеет два разных хэш кода.

Если количество элементов в массиве становиться большим, чем размер массива умноженный на коэффициент 0.75, то создается новая хэш таблица с массивом большим в 2 раза, далее идя по элементам массива прошлой хэш таблицы добавляем в новый массив связные списки, при этом заново подсчитывая их ключи (индексы) в связи с увеличившимся массивом. В конце меняем массив новой хэш таблицы с массивом старой хэш таблицы.

Выполнение работы.

Пользователь вводит хэш коды и закодированную им строчку (может быть и символ) и они добавляются в хэш таблицу, на экран выводится представление хэш таблицы в виде псевдографики. Далее пользователь выбирает как считать информацию, которую он хочет декодировать — из файла или из консоли. Затем декодированные хэш коды выводятся на экран.

Разработанный код смотреть в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 - результаты тестирования.

№ п/п	Входные данные	Выходные данные	Комментарии
1	*Пользователь ввел строчки и хэш коды, с помощью*	*Вывелась хэш таблица:*	Программа успешно заполнила хэш таблицу, считав данные с консоли, заменила строчки с

	которых они были закодированы: 101 a 1011 k 10101 b 10101 cb 1010 !!? 1000 kq1 1 # 11011 kk 10000 notNull 1110 end *Данные для декодировки вводятся из консоли:* 0101 1111111111 1011 10101 10101 1010 0 0001000 11011 1110 10000	|0| -> 10000 : notNull |1| -> 1 : # |2| -> NULL |3| -> NULL |4| -> NULL |5| -> 101 : a -> 10101 : cb |6| -> NULL |7| -> NULL |8| -> 1000 : kq1 |9| -> NULL |10| -> 1010 : !!? |11| -> 1011 : k -> 11011 : kk |12| -> NULL |13| -> NULL |14| -> 1110 : end |15| -> NULL *декодированная информация:* a NONE k cb cb !!? NONE kq1 kk end notNull	одинаковыми хэш кодами, обработала коллизии, увеличила массив и вывела ее на экран. Далее смогла корректно декодировать хэш коды (которые присутствовали в хэш таблице).
2	*Пользователь ввел строчки и хэш коды, с помощью которых они были закодированы:* 0 a 11011 parasite2 1 b 10 c 11 d 11010 parasite1 100 e 101 f 110 g 111 h *Данные для декодировки были получены из файла test1.txt* 0 1 10 11 100 101 110 111 11011	*Вывелась хэш таблица:* |0| -> 0 : a |1| -> 1 : b |2| -> 10 : c |3| -> 11 : d |4| -> 100 : e |5| -> 101 : f |6| -> 110 : g |7| -> 111 : h |8| -> NULL |9| -> NULL |10| -> 11010 : parasite1 |11| -> 11011 : parasite2 |12| -> NULL	Программа успешно заполнила хэш таблицу, считав данные с файла, обработала коллизии, увеличила массив и вывела ее на экран. Далее смогла корректно декодировать хэш коды (которые присутствовали в хэш таблице).

	11010 h	|13| -> NULL |14| -> NULL |15| -> NULL *декодированная информация:* a b c d e f g h parasite2 parasite1	

Вывод.

В ходе выполнения лабораторной работы была создана рабочая программа с необходимым API интерфейсом, которая размещает в хэш таблицу пары значений: закодированная информация—хэш код, а затем декодирует хэш коды, полученные из консоли или файла. На практике были закреплены навыки по написанию и использованию хеш-таблиц.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ.

Название файла: main.cpp

```
#include <iostream>
#include <string>
#include "HashTable.h"
#include <fstream>

int main(){

    HashTable<string> table;
        cout<<R"(Enter your elements like this: "hash code" space "information", example: HR4A cat KJ3V dog)"<<endl;
    table.readElements();
    cout<<"Your hashTable:"<<endl;
    table.printElements();

    cout<<"Choose how to enter encoded information:\n1 - from console\n2 - from file"<<endl;
    char c;
    cin>>c;
    getchar();
    switch(c){
        case '1': {
            cout<<"Enter your information:"<<endl;
            c = '\0';
            string search;
            while (c != '\n') {
                cin >> search;
                table.decode(search);
                c = getchar();
            }
            cout << '\n';
            break;
        }
```

```
        case '2': {
            c = '\0';
            string filename;
            string search;
            cout<<"Enter file name"<<endl;
            cin>>filename;
            ifstream in;
            in.open(filename);

            if(!in.is_open()){
                cerr<<"No such file"<<endl;
                return 2;
            }

            cout<<"Decoded text:"<<endl;
            while(in>>search){
                table.decode(search);
            }
            cout<<'\n';
            break;
        }
        default: cout<<"You haven't chose anything"<<endl; return 1;
    }
}
```

Название файла: HashTable.h

```
#ifndef UNTITLED12_HASHTABLE_H
#define UNTITLED12_HASHTABLE_H
#include <iostream>
#include <string>
#include <cmath>
using namespace std;

template <class T>
class HashTable{
private:
    struct linkedList{
```

```
        T data;
        string hash;
        linkedList* next;
        bool exist = true;
    };

    linkedList* array;
    int size = 0;
    int capacity;
    const float resizeOdd = 0.75;
public:
    int HashFunction2(string &hash){
        int n = stoi(hash);
        hash = std::to_string(n);
        int key = 0, i = 0, remainder;
        while (n != 0) {
            remainder = n % 10;
            n /= 10;
            key += remainder * pow(2, i);
            ++i;
        }
        return key % capacity;
    }

    int HashFunction1(string hash){
        int key = 0;
        for(size_t i = 0;i<hash.size();i++){
            key+=int(hash[i]);
        }
        return key%capacity;
    }

    HashTable(int capacity = 8){
        this->capacity = capacity;
        this->array = new linkedList[capacity];
        for(int i=0 ; i < capacity ; i++)
            array[i].exist = false;
    }
```

```
    void append(T data, string hash){

        if(size>= int(resizeOdd*capacity)) resize();

        int key = HashFunction2(hash);

        if(!array[key].exist){
            array[key].data = data;
            array[key].hash = hash;
            array[key].exist = true;
            array[key].next = nullptr;
            size+=1;
        } else {
            linkedList* newNode = new linkedList;
            linkedList* tmp = &array[key];
            newNode->data = data;
            newNode->hash = hash;
            newNode->next = nullptr;
             if(tmp->hash==hash){ tmp->data = data; return;}//если первый
элемент будет иметь тот же хэш код, что и новый
            while(tmp->next!= nullptr){
                tmp = tmp->next;
                if(tmp->hash==hash){
                    tmp->data = data;
                    return;
                }
            }
            tmp->next = newNode;
            size+=1;
        }
    }

    void readElements(){
        string tmpHash;
        T tmpData;
        char c;
        while(c!='\n'){
            cin>>tmpHash>>tmpData;
            this->append(tmpData,tmpHash);
```

```
            c = getchar();
        }
    }

    void resize(){
        int pastSize = capacity;
        capacity*=2;
        HashTable tmp2(capacity);
        linkedList *tmp1;
        for(int i = 0;i<pastSize;i++){
            if(array[i].exist) {
                tmp2.append(array[i].data, array[i].hash);
                tmp1 = &array[i];
                while (tmp1->next != nullptr) {
                    tmp1 = tmp1->next;
                    tmp2.append(tmp1->data, tmp1->hash);
                }
            }
        }

        swap(array,tmp2.array);
    }

    void decode(string hash2){
        int key = HashFunction2(hash2);
            if(array[key].hash  ==  hash2){  cout<<array[key].data<<  '  ';
return;}
        linkedList* tmp2 = &array[key];
        while(tmp2->next!=nullptr){
            tmp2 = tmp2->next;
            if(tmp2->hash==hash2){  cout<<tmp2->data<< ' '; return;}
        }
        cout<<"NONE ";
    }

    void printElements(){
        linkedList* tmp = array;
        for(int i = 0;i<capacity;i++){
            cout<<"|"<<i<<"| -> ";
```

```
            if(tmp[i].exist) cout<<tmp[i].hash<<" : "<<tmp[i].data;
            else {cout<<"NULL"<<endl; continue;}
            linkedList* tmp2 = &tmp[i];
            while(tmp2->next!=nullptr){
                tmp2 = tmp2->next;
                cout<<" -> "<<tmp2->hash<<" : "<<tmp2->data;
            }
            cout<<"\n";
        }
    }
};
#endif
```

Название файла: Makefile

```
all: goal
	./start

goal: main.o
	g++ main.o -o start

main.o: main.cpp HashTable.h
	g++ -c main.cpp

clean:
	rm -f *.o
```