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.

Закодировать некоторое количество символов без коллизий с помощью хеш-таблицы, а затем декодировать текст из командной строки или файла.

Задание.

Вариант 4.

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

Анализ задачи.

Был реализована такая структура данных, как хеш-таблица.

Ключи или те же индексы массива для символов получаются следующим образом: берется остаток деления ascii кода символа на размер хеш-таблицы.

Если происходит коллизия, то сначала проверяется, не является ли добавляемый элемент копией того, который уже находится в хеш-таблице, если это так - то программа ничего не делает. Иначе используется такой алгоритм, как линейное зондирование: проверяется i+1 элемент массива, если он является пустым, то записываем наш символ туда и теперь его код равен +1 от того, что мы получили изначально от хеш-функции. Если там уже есть символ, то идем дальше, пока не найдем свободное место. Там в цикле используется конструкция `key%=size` – в случае достижения конца массива (`key == size`) начинаем поиск с начала массива. Если мы находим копию нашего символа в ходе поиска, то прекращаем работу функции. Если после полного обхода хеш-таблицы мы не находим свободного места — то создаем новый массив размером +1, копируем туда наши значения и на последнее место ставим наш символ.

Выполнение работы.

Был разработан примитивный API, благодаря которому пользователь может выбрать какие символы будут закодированы — лично им введенные или

случайно сгенерированные (количество которых определяет пользователь).

Если он выбрал первый вариант, то с терминала будут последовательно считаны а затем добавлены символы в массив функцией append.

Если он выбрал второй вариант, то в массив будут добавлены случайные элементы от a-z, которые могут повторятся.

Далее с файла считываются закодированные данные через пробел (1 код — 1 символ). Декодирование происходит следующим образом: берется элемент хеш-таблицы с индексом равным коду, который мы прочитали из файла.

Таким образом декодируем весь файл. Если встречается несуществующий индекс элемента массива (код символа) или на его месте пустое значение, то этот код на экран мы выводим как NULL.

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	*символы для кодировки сгенерированы случайно.* *Размер массива и количество символов — 5.* 0: d (ascii code 100) 1: y (ascii code 121) 2: t (ascii code 116) 3: v (ascii code 118) 4: m (ascii code 109)	d NONE d v v NONE y t m t NONE	Программа успешно закодировала случайные 5 символов, а затем декодировала файл, не смотря на не одинарные пробелы между кодами. Коды, которых нет в хеш-таблице были помечены как NULL. Формат, где первые цифры кода могут быть нулями считаны правильно.

	Закодированные данные получены из файла test1.txt 0 41 00 3 003 33 1 02 4 2 6		
2	*символы для кодировки были введены пользователем, изначальный размер массива равен 5* a ab c c f g f ! 1 0: f (ascii code 102) 1: g (ascii code 103) 2: a (ascii code 97) 3: b (ascii code 98) 4: c (ascii code 99) 5: ! (ascii code 33) 6: 1 (ascii code 49) *Закодированные данные были введены из консоли, пользователь указал, что хочет ввести 9 символов* 56 5 6 1 4 3 0002 2 00 1 2	NONE ! 1 g c b a a f	Программа успешно закодировала 7 символов введенных пользователем. Изначальный размер массива был равен 5. Символы a, b, c были сразу закодированы, они без коллизий, их дубликаты добавлены не были. Далее с помощью линейного зондирования символу f был присвоен код (индекс массива) 0, а символу g код 1, тк возникла коллизия для этих символов. Далее программа успешно декодировала введенный пользователем текст, несмотря на пробелы между кодами. Коды, которых нет в хеш-таблице были помечены как NULL. Формат, где первые цифры кода могут быть нулями считаны правильно. Пользователь ввел 11 кодов, но так как ранее указал, что введет только 9, коды 1 и 2 не были считаны.

Вывод.

В ходе выполнения лабораторной работы была создана рабочая программа с необходимым API интерфейсом, которая кодирует введенные пользователем символы либо символы, которые были случайно выбраны из английского алфавита и декодирует текст из файла или командной строки. Программа способна избежать коллизий данных, путем использования линейного зондирования и в худшем случае расширения массива. На практике были закреплены навыки по написанию и использованию хеш-таблиц.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ.

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <time.h>
#include "HashTable.h"
using namespace std;

HashTable<char> generate(int size){
    HashTable<char> tmp(size);
    srand (time(NULL));
    char characters[] = "abcdefghijklmnopqrstuvwxyz";
    char c;
    cout<<"Random characters:"<<endl;
    for(int i = 0; i<size; i++){
        c = characters[rand() % 26];
        cout<<c<<' ';
        tmp.append(c);
    }
    cout<<"\n";
    return tmp;
}



int main() {
     cout<<"Choose how to create hash table with encoded symbols, there codes - array index:\n1 - create it randomly (from a to z)\n2 - input it manually now"<<endl;
    char c1;
    cin>>c1;

    cout<<"Enter size of hash table:"<<endl;
    int c2;
```

```
    cin>>c2;
    getchar();
    HashTable<char> table(c2);

    switch(c1){
        case '1': table = generate(c2); break;
        case '2':
            cout<<"Enter your symbols like this a b c..."<<endl;
            c1 ='\0';
            while(c1!='\n'){
                scanf("%c",&c1);
                if(c1==' ') continue;
                else if(c1 =='\n') break;
                else table.append(c1);
            }
            break;
        default: cout<<"You haven't chose anything"<<endl; return 1;
    }
    cout<<"Your hash table:"<<endl;
    table.printHashTable();

        cout<<"Choose how to enter text:\n1 - from file\n2 - from
console"<<endl;
    c1 = '\0';
    cin>>c1;
    getchar();
    c2 = '\0';
    switch(c1){
        case '1':{
            string filePathIn;
            cout<< "Enter input file:"<<endl;
            cin>>filePathIn;
            ifstream in;
            in.open(filePathIn);
            if(!in.is_open()){
                cerr<<"No such file"<<endl;
                return 2;
            }
            cout<<"Decoded text:"<<endl;
```

```
            while(in>>c2){
                if(c2 < table.size && c2 >= 0 && table.array[c2] != NULL)
{
                    cout<<table.array[c2]<<' ';
                } else cout<<"NONE ";
            }
            cout<<'\n';
            break;
        }
        case '2':{
            int size;
             cout<<"How many elements you will enter?\nP.S. Decoded text
will be printed print after your codes"<<endl;
            cin>>size;
            c2 = '\0';
            for(int i = 0;i<size;i++){
                scanf("%d",&c2);
                if(c2 < table.size && c2 >= 0 && table.array[c2] != NULL)
{
                    cout<<table.array[c2]<<' ';
                } else cout<<"NONE ";
            }
            cout<<'\n';
            break;
        }
        default: cout<<"You haven't chose anything"<<endl; break;
    }
    return 0;
}
```

Название файла: HashTable.h

```
#ifndef HASHTABLE_HASHTABLE_H
#define HASHTABLE_HASHTABLE_H
#include <iostream>
#include <fstream>
#include <string>
#include <time.h>
using namespace std;
template <class T>
```

```
class HashTable{
public:
    T *array;
    int size;

    HashTable(int size){
        this->size = size;
        this->array = new T[size];
        for(int i=0 ; i < size ; i++)
            array[i] = NULL;
    }

    void append(T data){
        int key = (int)data % size;
        if(array[key] == NULL){
            array[key] = data;
        } else{
            int oldKey = key;
            while(array[key]!=NULL){
                if(array[key] == data){
                    return;
                }
                key++;
                key%=size;
                if(oldKey == key){
                    T *tmp = new T[size+1];
                    for(int i=0;i<size;i++) {
                        tmp[i] = array[i];
                    }
                    delete[] array;
                    this->array = tmp;
                    array[size] = data;
                    size+=1;
                    return;
                }
            }
            array[key] = data;
        }
    }
```

```
    void printHashTable(){
        for(int i= 0;i<size;i++){
            cout<<i<<": ";
            if(array[i] == NULL) cout<<"NULL"<<endl;
                                 else   cout<<array[i]<<"   (ascii   code
"<<int(array[i])<<')'<<endl;
        }
    }
};
#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
```