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: Декодирование

Студент гр. 9382 _____ Бочаров Г.С.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

Декодировать входные данные, используя хеш-таблицу в качестве хранилища для символов.

Задание.

Вариант 4:

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

Описание алгоритма.

1) В задании не указан тип кодирования, поэтому условимся, что кодирование каждого символа производилось двоичным кодом одинакового размера для каждого символа, при том минимально возможной длины.

2) Считывается кол-во символов в словаре. После чего считываются символы и их коды и заносятся в массив соответствующих структур.

3) Заводится хеш-таблица, представляющая собой массив символов. Размером данного массива будет минимальная степень двойки, не меньше размера словаря символов, что позволит определить уникальный хэш для каждого кода. В массиве хранятся символы.

4.1) Для символа по его коду определяется его ячейка в хэш-таблице с помощью хэш-функции.

4.2) Хэш-функция (getHash()) принимает на вход размер таблицы и строку из нулей и единиц, соответствующую коду символа и переводит ее в десятичное число. Отсутствие коллизий обеспечивается устанавливаемой длинной кода и хэш-таблицы. Число всевозможных кодов указанной длины не превышает размера хэш-таблица. Одинаковый хэш может быть получен только из одинаковых кодов.

5) Далее посимвольно считывается закодированное сообщение. Как только считывается часть сообщения, которая может быть кодом какого-либо символа, к ней применяется хэш-функция. Таким образом определяется

является ли эта часть кодом символа. Если является, то определяется, коду какого символа она соответствует.

Функции и структуры данных.

Структура Node хранит в себе значение типа char (символ) и строковое значение (код символа).

struct HashTable — Структура, содержащая в себе массив символов.

void HashTable::void addElement(const Node &n) — Функция принимает на вход структуру Node, содержащую символ и его код. Функция определяет номер символа в массиве, и добавляет этот символ в массив.

char HashTable:: getElement(std::string s) – Функция определяет символ по его коду. s – это строка, содержащая код символа. Возвращаемое значение – найденный символ.

int getHash(int tableSize, std::string s) — Функция принимает на вход размер хэш-таблицы и строковое значение соответствующее в данной программе двоичному коду. Функция возвращает целочисленное значение, которое является номером ячейки массива, в которую записывается символ с соответствующим кодом.

template<typename StreamT>
void readTable(StreamT &in, std::vector<Node> &table, int size)
 — Функция принимает на вход поток ввода, массив переменных и допустимый размер кода. Функция считывает переменные с их кодами в массив.

 template<typename StreamT>
void getCode(StreamT &in, std::string &res)- Функция считывает коды символов.

int getLog(int a) — Функция принимает на вход число и вычисляет его логарифм по основанию 2.

template<typename StreamT>
bool readCode(StreamT &in, int codeSize, std::string &temp) — Функция считывает очередной код из закодированного сообщения.

template<typename StreamT>
std::string deCode(StreamT &in, int codeSize, const HashTable &hTable) — Функция принимает на вход поток ввода, допустимый размер кода, и хэш-таблицу. Функция считывает закодированное сообщение и декодирует его.

void launch() - Функция запрашивает у пользователя формат ввода и вызывает функциии считывания и обработки входных данных.

Тестирование.

Результаты тестирования представлены в табл. 1.

(Скрины см. в приложении Б)

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	7 a 000 b 111 c 001 d 110 m 010 9 011 8 101 101000111110010	Вывод хэш-таблицы 0 : a 1 : c 2 : m 3 : 9 4 : 5 : 8 6 : d 7 : b Найден код : 101 Хэш : 5	Вывод хэш-таблицы и результата и процесса декодирования.

		Найден код : 000 Хэш : 0 Найден код : 111 Хэш : 7 Найден код : 110 Хэш : 6 Найден код : 010 Хэш : 2 res-->8abdm	
2.	8 1 000 b 111 3 001 d 110 (010 9 011 8 101 0 100 00101010101111000111111 01011	Вывод хэш-таблицы 0 : 1 1 : 3 2 : (3 : 9 4 : 0 5 : 8 6 : d 7 : b res-->3(89d3b89	
3.	2 1 0 0 1 10100011	Вывод хэш-таблицы 0 : 1 1 : 0 res-->01011100	
4.	3 b 01 c 11 d 10 011101101101	Вывод хэш-таблицы 0 : 1 : b 2 : d 3 : c res-->bcbdcb	
5.	7 a 020 b 111 c 001	Неверный код	Неверный код символа

	d 110 m 010 9 011 8 101 1010001111110010		
6.	3 a 11 b 11 c 01 1111	Введенный код уже используется	Попытка присвоить 1 код 2-м символам
7.	3 a 11 a 10 c 01 1111	Введенный символ уже имеет код	Попытка присвоить одному символу 2 кода
8.	2 a 1010 b 1111 10101111	Неверная длина кода	Программа не смогла считать очередной код из закодированного сообщения
9.	6 a 000 b 111 c 001 d 110 m 010 9 011	Вывод хэш-таблицы 0 : a 1 : c 2 : m 3 : 9 4 :	Встречен неизвестный код

	101000111110010	5 : 6 : d 7 : b Неопознанный код 101	

Выводы.

В ходе работы был разработан алгоритм декодирующий зашифрованное сообщение с помощью хэш-таблицы, в которой хранятся символы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <vector>
#include <cmath>
#include <fstream>

struct Node {
    char symbol_;
    std::string code_;
};

int getHash(int tableSize, std::string s) {
    int k = 0;
    for (int i = 0; i < s.size(); i++)
        k += (s.at(i) - '0') * pow(2, s.size() - i - 1);
    //std::cout << "Code = " << s << "Hash == " << k % tableSize << std::endl;
    return k % tableSize;
}

struct HashTable {
    std::vector<char> arr_;

    explicit HashTable(int size) {
        arr_ = std::vector<char>(size);
    }

    void addElement(const Node &n) {
        int k = getHash(arr_.size(), n.code_);
        if (arr_.at(k) != '\0') {
            throw std::runtime_error("Внимание коллизия");
        }
        arr_.at(k) = n.symbol_;
    }
    char getElement(std::string s)
    {
        int k = getHash(arr_.size(), s);
        std::cout<<"Найден код : "<< s <<"  Хэш : "<< k<<std::endl;
        return arr_.at(k);
    }
};

int getLog(int a);

template<typename StreamT>
void getCode(StreamT &in, std::string &res) {
    char c;
    res.clear();

    in >> std::noskipws;

    in >> c;
    if (c != ' ')
        throw std::runtime_error("Нехватает пробела");
    while (in >> c && c != '\n') {
        if (c != '0' && c != '1')
            throw std::runtime_error("Неверный код");
        res.push_back(c);
    }
}
```

```
template<typename StreamT>
void readTable(StreamT &in, std::vector<Node> &table, int size) {
    char symbol;
    std::string code;
    int codeSize = getLog(size);
    while (in >> symbol && symbol != '\n') {
        getCode(in, code);
        if (code.size() != codeSize)
            throw std::runtime_error("Неверная длина кода");
        //std::cout << symbol << "-->" << code << std::endl;
        for (auto &i:table) {
            if (i.code_ == code)
                throw std::runtime_error("Введенный код уже используется");
            if (i.symbol_ == symbol)
                throw std::runtime_error("Введенный символ уже имеет код");
            if (table.size() > size - 1)
                throw std::runtime_error("Введено неверное кол-во символов");
        }
        Node n{symbol, code};
        table.push_back(n);
    }
    if (table.size() != size)
        throw std::runtime_error("Введено неверное кол-во символов");
}

void printNode(const Node &n) {
    std::cout << "имя - " << n.symbol_ << "  Код - " << n.code_ << std::endl;
}

void printTable(const std::vector<Node> &table) {
    for (auto &i:table)
        printNode(i);
}

void printHashTable(const HashTable &t) {
    std::cout << "Вывод хэш-таблицы" << std::endl;
    for (int i = 0; i < t.arr_.size(); i++)
        std::cout << i << " : " << t.arr_.at(i) << std::endl;
}

int getLog(int k) {
    if (k == 1)
        return 1;
    int t = k - 1;
    int res = 0;
    while (t != 0) {
        t /= 2;
        res++;
    }
    return res;
}


template<typename StreamT>
bool readCode(StreamT &in, int codeSize, std::string &temp) {
    char c;
    temp.clear();
    for (int i = 0; i < codeSize; i++) {
        in >> c;
        if (c == '\n')
            break;
        if (c != '1' && c != '0')
            throw std::runtime_error("Неверный формат закодированного сообщения
c =" + c);
        temp.push_back(c);
    }
```

```
    if (temp.size() != 0 && temp.size() != codeSize)
        throw std::runtime_error("Неверный формат закодированного сообщения");
    return temp.size() == codeSize;
}

template<typename StreamT>
std::string deCode(StreamT &in, int codeSize,  HashTable &hTable) {
    std::string res;
    std::string temp;
    char c;
    int k;
    while (readCode(in, codeSize, temp)) {
        c = hTable.getElement(temp);
        if (c == '\0')
            throw std::runtime_error("Неопознанный код " + temp);
        res.push_back(c);
    }
    return res;
}

void launch() {
    int symbolCount;
    std::vector<Node> table;
    int readFormat;

    std::cout << "0 - считать из файла, 1 - считать с консоли" << std::endl;
    std::cin >> readFormat;
    std::cin.ignore();
    switch (readFormat) {
        case 0: {
            std::cout << "Введите имя файла : ";
            std::ifstream in;
            std::string fileName;
            std::cin >> fileName;
            in.open(fileName);
            if (in) {
                in >> symbolCount;
                HashTable hTable(pow(2, getLog(symbolCount)));
                readTable(in, table, symbolCount);

                for (auto &i:table) {
                    hTable.addElement(i);
                }
                printHashTable(hTable);
                std::string res = deCode(in, getLog(symbolCount), hTable);
                std::cout << "res-->" << res;
            } else
                throw std::runtime_error("Файл не найден!");
            in.close();
            break;
        }
        case 1: {
            std::cout
                    << "Введите количество символов в словаре"
                    << std::endl;

            std::cin >> symbolCount;
            HashTable hTable(pow(2, getLog(symbolCount)));
            std::cout
                    << "Введите символы и их коды через пробел, каждая пара
символ-код с новой строки."
                       " Конец ввода - пустая строка. Например: a 101"
                    << std::endl;
            std::cout << "Длина кода = " << getLog(symbolCount) << std::endl;
            readTable(std::cin, table, symbolCount);
            for (auto &i:table) {
```

```
                hTable.addElement(i);
            }

            printHashTable(hTable);
            std::cout
                    << "Введите закодированное сообщение"
                    << std::endl;
            std::string res = deCode(std::cin, getLog(symbolCount), hTable);
            std::cout << "res-->" << res;
            break;
        }
        default: {
            throw std::runtime_error("Неверное действие");
        }
    }
}

int main() {
    try {
        launch();
    } catch (std::exception &e) {
        std::cerr << e.what() << std::endl;
    }
    return 0;
}
```