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: «Кодирование и декодирование»

Студентка гр. 9381 _____ Андрух И.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2020

Цель работы

Ознакомиться с алгоритмами кодирования и декодирования и применить их на практике.

Задание

Декодирование: статическое, коды символов хранятся в бинарном дереве

Описание алгоритма.

На вход программе подается бинарное дерево Хаффмана для алфавита, по нему определяются коды символов. Происходит чтение первого бита файла. Если это «0», то следует двигаться по левой ветке дерева, начиная с корня, а если «1», то по правой. Далее читается второй бит и происходит движение по одной из следующих двух веток, по направлению к листьям. Когда декодер достигнет листа дерева, он узнает код первого несжатого символа(обычно это символ ASCII). Процедура повторяется для следующего бита, начиная опять из корня дерева.

Далее по полученному набору кодов декодируем закодированный текст из файла.

Описание структур данных и функций.

Реализован класс class HuffmanTree. Полями класса являются:

HuffmanTree* left – указатель на левое поддерево;

HuffmanTree* right – указатель на правое поддерево;

bool flag = false – флаг, показывающий, был ли встречен лист дерева;

T value – символ словаря;

Функции:

1) HuffmanTree<T>::HuffmanTree(ifstream& s) – конструктор класса, инициализирует дерево строкой, находящейся в начале файла;
2) bool isLeaf() – возвращает true, если элемент является листом дерева;
3) HuffmanTree *get_left() - возвращает указатель на левый элемент;
4) HuffmanTree *get_right() - возвращает указатель на правый элемент;

5) T val() – возвращает символ декодируемого текста.

Тестирование программы.

Было создано несколько тестов для проверки работы программы. Помимо тестов, демонстрирующих работу алгоритма, были написаны тесты, содержащие некорректные данные, для демонстрации вывода сообщений об ошибках введенных данных (см. Приложение А).

Вывод.

В ходе выполнения работы был изучен алгоритм статического декодирования с использованием бинарного дерева.

ПРИЛОЖЕНИЕ А

Тестирование.

Демонстрация работы для введенной строки «((((a)(b))())(((c)())(d)))0000011110010011001100»:

Нажмите 1, чтобы декодировать файл
Нажмите 2, чтобы выйти
1
Введите файл с данными: /home/ira/CLionProjects/AiSD_lb5/input
Введите файл для записи: /home/ira/CLionProjects/AiSD_lb5/output
скобка: (
скобка: (
скобка: (
скобка: (
элемент словаря: a
скобка:)
скобка:)
скобка: (
элемент словаря: b
скобка:)
скобка: (
скобка: (
элемент словаря:)
скобка: (
скобка: (
скобка: (
скобка: (
элемент словаря: c
скобка:)
скобка:)
скобка: (
элемент словаря:)
скобка: (
скобка: (
элемент словаря: d
скобка:)
символ кода: 0
символ кода: 0
символ кода: 0
символ декодированного текста: a
символ кода: 0
символ кода: 0
символ кода: 1
символ декодированного текста: b

символ кода: 1
символ кода: 1
символ декодированного текста: d
символ кода: 1
символ кода: 0
символ кода: 0
символ декодированного текста: c
символ кода: 1
символ кода: 0
символ кода: 0
символ декодированного текста: c
символ кода: 1
символ кода: 1
символ декодированного текста: d
символ кода: 0
символ кода: 0
символ кода: 1
символ декодированного текста: b
символ кода: 1
символ кода: 0
символ кода: 0
символ декодированного текста: c
/home/ira/CLionProjects/AiSD_lb5/input был декодирован в /home/ira/CLionProjects/AiSD_lb5/output успешно.
Нажмите 1, чтобы декодировать файл
Нажмите 2, чтобы выйти

Входные данные	Выходные данные
((a)(b))010101	ababab
(((b)(c))(a))01011000	ccab
(a)0100101	введенный файл не корректен
(((a)((b)(c))) ())011000001001100	caabca
()10010	/home/ira/CLionProjects/AiSD_lb5/input is broken.
((a)(b))320101	введенный файл не корректен
((((a)(b))())(((c)())	abdccdbc

(d)))0000011110010011001100	
(((a)(b))(c))00000110010101	aabcacbb
(((a)(b)))034	введенный файл не корректен

ПРИЛОЖЕНИЕ Б

Исходный код программы.

```
#include <iostream>
#include <fstream>
#include <cctype>
#include <string>
using namespace std;
template <class T>
class HuffmanTree{
private:
    HuffmanTree* left; // 0
    HuffmanTree* right; // 1
    bool flag = false; //true if leaf
    T value;
public:
    HuffmanTree(ifstream& s);
    bool isLeaf(){ //возвращает true, если элемент является листом дерева
        return flag;
    }
    HuffmanTree *get_left() { //возвращает указатель на левый элемент
        return left;
    }
    HuffmanTree *get_right() { //возвращает указатель на првый элемент
        return right;
    }
    T val() { //возвращает кодируемый элемент
        cout << "символ декодированного текста: " << value << endl;
        return value;
    }
    ~HuffmanTree();
};
template <class T>
HuffmanTree<T>::HuffmanTree(ifstream& s/*элемент класса ifstream*/)  {
    flag = false;
    left = NULL;
    right = NULL;
    char ch;
    if (s.peek() == '('){ //.peek() Считывает байт из файла, но не
переходит к следующему
        s >> ch; // извлекаем '('
        cout << "скобка: " << ch << endl;
    }
```

```
    else
        return;
    if (s.peek() == '('){
        left = new HuffmanTree(s);
        if (s.peek() == ')'){
            s >> ch; // извлекаем ')'
            cout << "скобка: " << ch << endl;
        }
    }
    else{
        flag = true;
        s >> value;
        cout << "элемент словаря: " << value << endl;
        return;
    }
    if (s.peek() == '('){
        cout << "скобка: " << ch << endl;
        right = new HuffmanTree(s);
        s >> ch; // извлекаем ')'
        if(ch != ')')
            cout << "буква кода: " << ch << endl;
    }
    if (s.peek() == ')'){
        s >> ch; // извлекаем ')'
        cout << "скобка: " << ch << endl;
    }
}
template <class T>
HuffmanTree<T>::~HuffmanTree(){//деструктор
    if (left)
        delete left;
    if (right)
        delete right;
}
int main(){
    int  c;
    string str_i, str_o;//названия входного и выходного файлов
    while(true){
        cout << "Нажмите 1, чтобы декодировать файл\n" << "Нажмите 2, чтобы
выйти" << endl;
        cin >> str_i;
        if (!isdigit(str_i[0])) //проверка
            continue;
        c = stoi(str_i);
        str_i.clear();
        switch (c)
        {
            case 1:
                break;
            case 2:
                return 0;
            default:
                cout << "Введите 1 или 2" << endl;
                continue;
        }
        cout << "Введите файл с данными: ";
        cin >> str_i;
        cout << "Введите файл для записи: ";
        cin >> str_o;
        if (str_i == str_o){
            cout << "Названия файлов не могут совпадать" <<endl;
            continue;
        }
        ifstream f;
        ofstream o;
        char b;
        f.open(str_i);
```

```
        o.open(str_o);
        if (!f){//проверка на существование входного файла
            cout << "Невозможно открыть файл с данными" << endl;
            continue;
        }
        if (!o)//проверка на существование выходного файла
        {
            cout <<"Невозможно открыть файл для записи" << endl;
            continue;
        }
        HuffmanTree<char> *Dictionary = new HuffmanTree<char>(f);
//создание и выделение памяти под словарь
        HuffmanTree<char> *tmp = Dictionary;
        while (!f.eof()){//пока не достигнут конец файла
            b = f.get();//считываем символ из файла f и возвращаем его
значение
            if ((b!='0')&&(b!='1')) {//если в закодированном сообщении
встречен недопустимый символ, выводим в файл предупреждение
                o << "введенный файл не корректен";
                break;
            }
            else {
                cout << "символ кода: " << b << endl;
                if (f.peek() == EOF)
                    if (b == '\n')
                        break;
                if (tmp)
                    tmp = ((b & 1) == 0 ? tmp->get_left(): tmp-
>get_right()); //если 1, то спускаемся к правому поддереву, если 0, то к
левому
                if (tmp && tmp->isLeaf()) {
                    o << tmp->val();//если дошли до листа, то запись
раскодированного символа
                    tmp = Dictionary;
                }
            }
        }
        f.close();
        o.close();
        if (tmp)
            cout << str_i << " был декодирован в " << str_o << " успешно."
<< endl;
        else
        cout << str_i << " не корректен." << endl;
        delete Dictionary;
    }
}
```