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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Алгоритмы и структуры данных»

Тема: Декодирование Хаффмана

Студент гр. 7383 _____ Зуев Д.В.

Преподаватель _____ Размочева Н.В.

Санкт-Петербург

2018

ОГЛАВЛЕНИЕ

Цель работы

Цель работы: познакомиться с алгоритмом динамического декодирования Хаффмана и реализовать его на языке программирования C++.

Формулировка задачи: На вход подаётся файл с закодированным содержимым. Требуется раскодировать содержимое файла определённым алгоритмом динамического декодирования Хаффмана.

Реализация задачи

В данной работе используется структура bin_tree — бинарное дерево.

- bin_tree* left — указатель на левое поддерево;
- bin_tree* right — указатель на правое поддерево;
- int weight — вес узла;
- char el — символ;

В функции main выводится приглашение выбрать способ ввода входных данных либо выйти из программы. В случае выбора файла, программа считывает текст из файла и записывает его в поток ввода. В случае ввода информации с консоли функция main считывает строку с консоли, записывает эту строку в этот же поток ввода.

Функция decode получает на вход поток входных данных. Проходит по текущему дереву в зависимости от получаемых битов. Вызывает функцию get_new или get_ex в зависимости от того куда она придет по дереву.

Функция print_Btree выводит бинарное дерево соответственно на экран с поворотом на 90 градусов против часовой стрелки. Введена для удобной проверки правильности перестроения дерева.

Функция get_new считывает двоичный код символа, добавляет этот символ в дерево и вызывает функцию comparison. выводит элементы леса при обходе его в ширину.

Функция get_ex добавляет единицу к весу существующего символа и вызывает функцию comparision.

Функция comparision перестраивает, если нужно, дерево.

Тестирование

Программа была собрана в компиляторе G++ в среде разработки Qt creator в OS Linux Ubuntu 16.04 LTS.

Корректные тестовые случаи представлены в приложении А.

По результатам тестирования выявлено, что поставленная задача была выполнена верно.

Вывод

В ходе работы была написана программа на языке C++, получающая на вход закодированное динамическим методом Хаффмана и декодирующая его. Был освоен динамический алгоритм кодирования Хаффмана.

ПРИЛОЖЕНИЕ А.

ТЕСТОВЫЕ СЛУЧАИ

Таблица 1 — Корректные случаи

Входные данные	Выходные данные
0111000000111001000011011111110001100111001	progg
0111000000111001000011011111100011001110001110000110000011000011011010101010001101001111000110111110010	programming
0110000100110001000011100100	abra
0110000100110001000011100100100010000110101100011001000110110	abrakadabra

ПРИЛОЖЕНИЕ Б.

ИСХОДНЫЙ КОД ПРОГРАММЫ

main.cpp:

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

using namespace std;

class error: public exception
{
public:
    explicit error(const char* a)
        {this->a=a; }

    virtual const char* what() const throw()
    { return "Wrong character.\n"; }
    void printErr()
    {
        cout<<"The character \""<<a<<"\" is not int.\n";
    }
private:
    const char* a;
};

struct bin_tree{
    bin_tree* parent;
    bin_tree* left;
    bin_tree* right;
    int weight;
    char el;
};

bin_tree* create()
{
    bin_tree* bt;
    bt = new bin_tree;
    bt->parent = NULL;
    bt->left = NULL;
    bt->right = NULL;
    bt->weight = 0;
    bt->el = '\n';
    return bt;
}

int print_BTree(bin_tree* BT, int t)
{
    if(BT != NULL)
```

```
    {
        t++;
        print_BTree(BT->right,t);
        for(int j = 0; j<t-1;j++)
            cout<<'\t';
        cout<<BT->weight<<' '<<BT->el<<endl;
        print_BTree(BT->left,t);
    }
    else
        cout<<endl;
    return 0;
}

int binary_to_int(char* bin)
{
    int a = atoi(bin);
    int s = 0;
    for(int i=0; a; i++)
    {
        s+=int(a%10*pow(2.0,i));
        a/=10;
    }
    return s;
}

bin_tree* create_node(bin_tree* btnul,char x)
{
    bin_tree* bt_left;
    bt_left = create();
    bt_left->parent = btnul;

    bin_tree* bt_right;
    bt_right = create();
    bt_right->el = x;
    bt_right->parent = btnul;

    btnul->left = bt_left;
    btnul->right = bt_right;
    btnul->right->parent = btnul;
    btnul->el = char(0);
    return btnul;
}

bin_tree** comparison(bin_tree* list[], int& size, bin_tree* bt)
{

    bt->weight+=1;
    for(int i = 0; i<size-2;i++)
    {
        if(list[i]->weight > list[i+1]->weight)
        {
            int j = i+1;
```

```
            while(list[i]->weight > list[j]->weight)
                j++;
            j--;
            bin_tree* tmpl;
            bin_tree* tmpr;
            int tmpi;
            char tmpel;
            if(list[i]->parent == list[j])
                return comparison(list, size, list[j]);
            tmpl = list[j]->left;
            tmpr = list[j]->right;
            tmpi = list[j]->weight;
            tmpel = list[j]->el;
            list[j]->left = list[i]->left;
            list[j]->right = list[i]->right;
            list[j]->weight = list[i]->weight;
            list[j]->el = list[i]->el;
            list[i]->left = tmpl;
            list[i]->right = tmpr;
            list[i]->weight = tmpi;
            list[i]->el = tmpel;
            if(list[i]->left != NULL)
                list[i]->left->parent = list[i];
            if(list[j]->left != NULL)
                list[j]->left->parent = list[j];
            if(list[i]->right != NULL)
                list[i]->right->parent = list[i];
            if(list[j]->right != NULL)
                list[j]->right->parent = list[j];
            return comparison(list, size, list[j]->parent);
        }
    }
    if(bt->parent!=NULL)
    {

        list = comparison(list, size, bt->parent);
    }
    return list;
}

bin_tree** get_new(stringstream& xstream, bin_tree* btnul,
bin_tree* list[], string& str, int &size)
{
    char* bin;
    bin = new char[8];
    for(int i = 0; i<8; i++)
    {
        if(!(xstream>>bin[i]))
            throw logic_error("Too few characters.\n");
    }
    int a;
    a = binary_to_int(bin);
```

```
    btnul = create_node(btnul, char(a));

    bin_tree** list1;
    list1 = new bin_tree*[size+2];
    list1[0] = btnul->left;
    list1[1] = btnul->right;
    size+=2;
    for(int i = 2; i<size; i++)
    {
        list1[i] = list[i-2];
    }
    str+=char(a);
    return comparison(list1, size, btnul->right);
}

bin_tree** get_ex(bin_tree* bt, bin_tree* list[], string& a, int
&size)
{
    a+=bt->el;
    return comparison(list, size, bt);
}

string decode(stringstream& xstream)
{
    string a;
    char x;
    bin_tree** list;
    list = new bin_tree*[1];
    int size = 1;
    bin_tree* bt1;
    bt1 = create();
    bt1->parent = NULL;
    bin_tree* bt;
    bt = create();

    list[0]=bt1;
    do
    {
        bt = list[size-1];
        print_BTree(bt, 0);// Для демонстрации работы

cout<<"_____________________________________________________________
_____________________"<<endl;
        char c = bt->el;
        while(int(bt->el) == 0)
        {
            if(!(xstream>>x))
                return a;
            if(x == '0')
            {
                bt = bt->left;
            }
```

```
            else
            {
                bt = bt->right;
            }
        }
        if(bt->el == '\n')
            list = get_new(xstream, bt, list, a, size);
        else
            list = get_ex(bt, list, a, size);
    }while(xstream.peek()!=EOF);
    print_BTree(bt1, 0);// Для демонстрации работы

cout<<"_____________________________________________________________
_____________________"<<endl;
    return a;
}

int main()
{
    stringstream xstream;
    short int tmp = 0;
    while(tmp != 3)
    {
        string str;
        string str0;
        string tmp1;
        try{
            xstream.str("");
            xstream.clear();
            cout<<"Введите 1, если желаете вводить выражение с
клавиатуры.\n"
              "Введите 2, если желаете брать выражение из файла
test.txt.\n"
              "Введите, 3 если хотите закончить работу."<<endl;
            getline(cin, tmp1);
            if(!atoi(tmp1.c_str() ))
                throw error(tmp1.c_str());
            else tmp = atoi(tmp1.c_str());
            switch(tmp){
                case 1:
                {
                    cout << "Введите формулу: \n";
                    getline(cin, str);
                    xstream << str;
                    break;
                }
                case 2:
                {
                    ifstream outfile;
                    outfile.open("test.txt");
                    if (!outfile)
                        throw runtime_error("File is not
```

```
open.\n");
                    getline(outfile, str);
                    outfile.close();
                    xstream << str;
                    break;
                }
                case 3:
                    continue;
                default:
                    throw invalid_argument("You entered wrong
number.\n");
            }
            str0 = decode(xstream);
            cout<<str0<<endl;
        }
        catch(error& e)
        {
            cout<<e.what();
            e.printErr();
            continue;
        }
        catch(exception& e)
        {
            cout<<e.what();
            continue;
        }
    }
    return 0;
}
```