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

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

ТЕМА: Динамическое кодирование и декодирование по Хаффману

текущий контроль

Студент гр. 8381 _____ Нгуен Ш. Х.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Нгуен Ш. Х.

Группа 8381

Тема работы: Динамическое кодирование и декодирование по Хаффману

Содержание пояснительной записки:
«Содержание», «Введение», «Алгоритм Хаффмана», «Описание программы», «Тестирование», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:
Не менее 40 страниц.

Дата выдачи задания: 11.10.2019

Дата сдачи реферата:

Дата защиты реферата:

Студент _____ Нгуен Ш. Х.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

В данной работе была создана программа на языке программирования C++, которая сочетает в себе несколько функций: кодирования/декодирования текста. Были использованы преимущества C++ для минимизации кода. Для лучшего понимания кода было в нем приведено большое кол-во отладочных выводов. Также была проведена его оптимизация с целью экономии выделяемой в процессе работы памяти и улучшения быстродействия программы.

SUMMARY

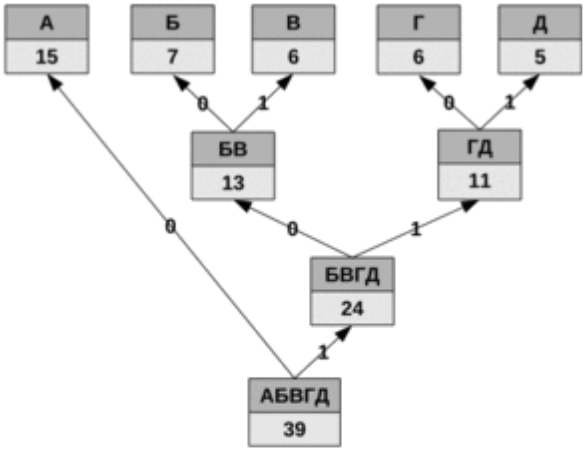
In this work, a program was created in the C ++ programming language, which combines several functions: encoding / decoding text. The benefits of C ++ were used to minimize code. For a better understanding of the code, it contained a large number of debugging outputs. Also, its optimization was carried out in order to save the memory allocated in the process of working and improve the speed of the program.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной курсовой работы является реализация алгоритма кодирования/декодирования текста методом Хаффмана и текущий контроль. Алгоритм использует коды переменной длины: часто встречающийся символ кодируется кодом меньшей длины, редко встречающийся — кодом большей длины. Кодирование Хаффмана позволяет строить кодовую схему в поточном режиме (без предварительного сканирования данных), не имея никаких начальных знаний из исходного распределения, что позволяет за один проход сжать данные. Преимуществом этого способа является возможность кодировать на лету. Данный метод сжатия имеет большое сходство с Фано-Шеннона, который появился на несколько лет раньше и является логическим продолжением алгоритма Шеннона. Кодирование Хаффмана широко применяется при сжатии данных, в том числе при сжатии фото- и видеоизображений (JPEG, MPEG), в популярных архиваторах (PKZIP, LZH и др.), в протоколах передачи данных HTTP (Deflate), MNP5 и MNP7 и других.

1. Алгоритм Хаффмана



Итого:

А	Б	В	Г	Д
0	100	101	110	111

Рисунок 1 – Пример алгоритма Хаффмана

Для решения поставленной подзадачи была написана функция HF_tree, которая строит дерево по алгоритму Хаффмана. Изначально во входной строке подсчитывается кол-во повторений каждого символа. При построении дерева на каждой итерации выбираются два символа с наименьшим кол-вом повторений во входной строке, для них создаётся новый элемент, который будет их родителем в дереве. Между элементами устанавливается связь, при этом элемент с большим кол-вом повторений помещается в левую ветку, а с наименьшим в правую. Затем по построенному дереву составляется словарь, в котором каждому символу сопоставляется его код, для последующей кодировки. Декодирование также происходит по построенному ранее дереву без помощи словаря.

Время работы : если сортировать элементы после каждого суммирования или использовать приоритетную очередь, то алгоритм будет работать за время массивы.

Описание программы

Структура данных.

Структура MinHeapNode и MinHeap :

```
typedef struct MinHeapNode{
   char data;
   unsigned freq;
   MinHeapNode *left, *right;
}MinHeapNode;


typedef struct MinHeap{
   unsigned size;
   unsigned capacity;
   MinHeapNode** array;
}MinHeap
```

Описание основных класса Huffman

Таблица 1 – Основные методы класса Huffman

Метод	Назначение
Minheapnode* newnode(char data, unsigned freq);	Создать новый узел
	Создать новый minheap
Void swapminheapnode(minheapnode** a, minheapnode** b);	Поменяйте местами содержимое двух minheap
Void minheapify(minheap* minheap, int idx);	Создать ветви дерева по порядку
	Проверьте size = 1

	Принимать minheap
Void insertminheap(minheap* minheap, minheapnode* minheapnode);	Вставить minheap
	Построить minheap
	Проверьте, является ли этот узел листом
Minheap* createandbuildminheap(char data[], int freq[], int size);	Инициализировать и построить minheap
	Построить дерево Huffman
String decode(minheapnode* root, string s,string youranswer);	Декодирование строки s
Void take_data_freq(string input, char* data, int* freq, int* size);	Возвращает данные, сохраненные в data
String printcodes(minheapnode* root, int arr[], int top);	Возвращает arr, содержащую строку символов 0 или 1 для отображения на экране
String printarr(int arr[], int n);	Возвращает arr, содержащую строку символов для отображения на экране
String take_encode(minheapnode* root, int *arr, int top,string input);	Возвращает arr, содержащую строку символов для отображения на экране
String take_encode_3(int *arr, int n);	Возвращает arr, содержащую строку символов для отображения на экране
String take_encode_2(minheapnode* root, int *arr, int top, char input);	Возвращает arr, содержащую строку символов для отображения на экране

Данные вопросов для пользавотеля

Decode

Easy:

+ Вар. 1

hello

110111100111

Результат : hoelo

+ Bap. 2

where

1110110100111

Результат : rewher

+ Bap. 3

letsgoo

11010010100010

Результат : logoto

- Normal:

+ Bap. 1

corlorfullsunshine

1100110100011101010011011000001000111010111010

Результат : finishonolulu

+ Bap. 2

whenthecitylightsup

011100111000010000011010101000100011101111110001100

Результат : yeuewnhiepctsu

+ Bap. 3

xmasnoitsexamatettest

000000010010001000000011001000101101110111100001101000101

Результат : nomonoxotseaitsot

Hard:

+ Bap. 1

jcuocsongnaycobietbaonhieuthucanphaithuchiend

1101100010011111011111000011111110110100001000101101011110101111110100111010000111011111101101111110

Результат : goodjobthanhsitdoyopipi

+ Bap. 2

vivaydunglangphithoigianvatuoitrecuabanokay

00110110001100011001110101011111011110111001001001110100110110100111011110111111111011100100110001100

Результат : rarrivotkabebobatoyourr

+ Bap. 3

dochilamaycauxamlethoijustdoititwillbefinepqzm

11111110111110101110011011110101110100110011100010111010100110001110001010000011010001110011000101

Результат : lzwhcqpudmoejxsitybn

2.3.1. Encode

- Easy:

+ Bap. 1

letsgoo

Результат : 110111000110101010

+ Bap. 2

where

Результат : 1101001110

+ Bap. 3

hello

Результат : 1101000111

- Normal :

+ Bap. 1

xmasnoitsexamatettest

Результат :

1001001111101000000011000011011101001111001111011100101110101010

+ Bap. 2

corlorfullsunshine

Результат :

10100000111110000111100010111111100010001100110111101000110 11

+ Вар. 3

whenthecitylightsup

Результат :

00001010010110111101001110110101110111100101011010101111100011000001

- Hard:

+ Вар. 1

vivaydunglangphithoigianvatuoitrecuabanokay

Результат :

10100111010110111110010010010001000111100110000100000101010101111110010110110111000011110000101011011101001101101111100011000111010001001110010011100001011111101110111111

+ Вар. 2

dochilamaycauxamlethoijustdoititwillbefinepqzm

Результат :

11000101011011111000111111000101100000111110110001100110001000010111111100101011100101001110001111100110000010110001010011010011010111010011111111110011010010010001100101100111010011010111101110111

+ Вар. 3

Р

е

з

у

л

ь

т

а

т

:

Тестирование

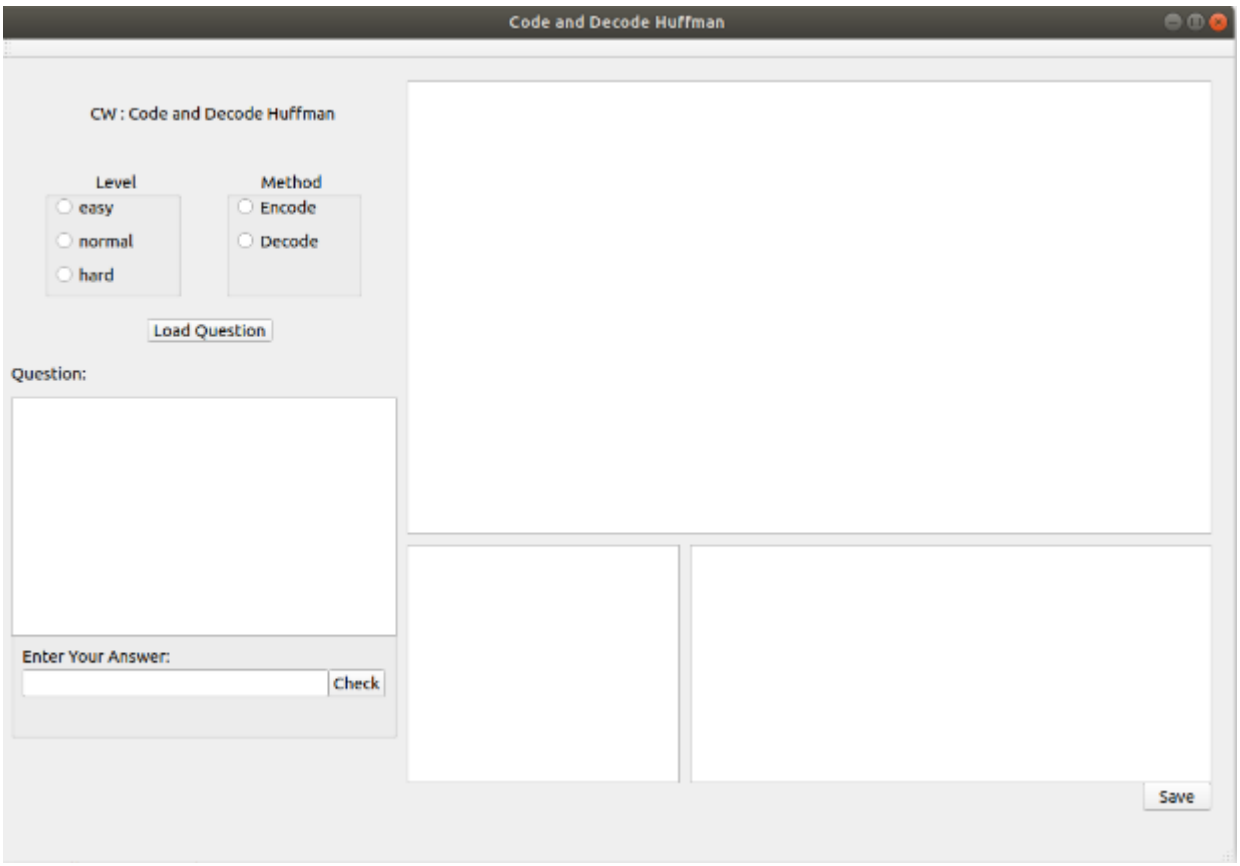


Рисунок 1 – Интерфейс программы

Пользователь должен выбрать один из уровней: легкий, средний или жесткий и один из двух методов: декодировать или кодировать. Затем будет отображаться проблема с просьбой ввести свои результаты, и вскоре после этого будут отображены результаты и действия по решению задачи.

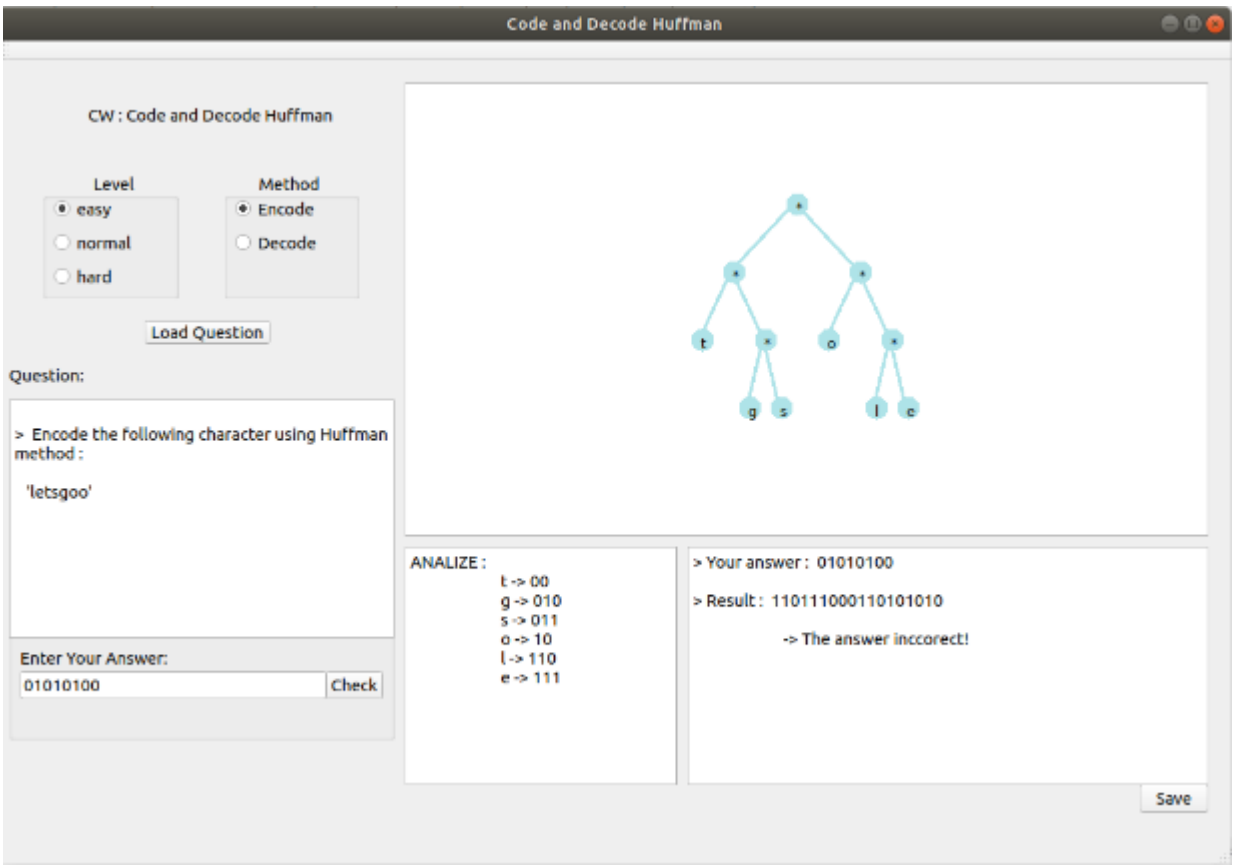


Рисунок 2 – Encode – Level : easy

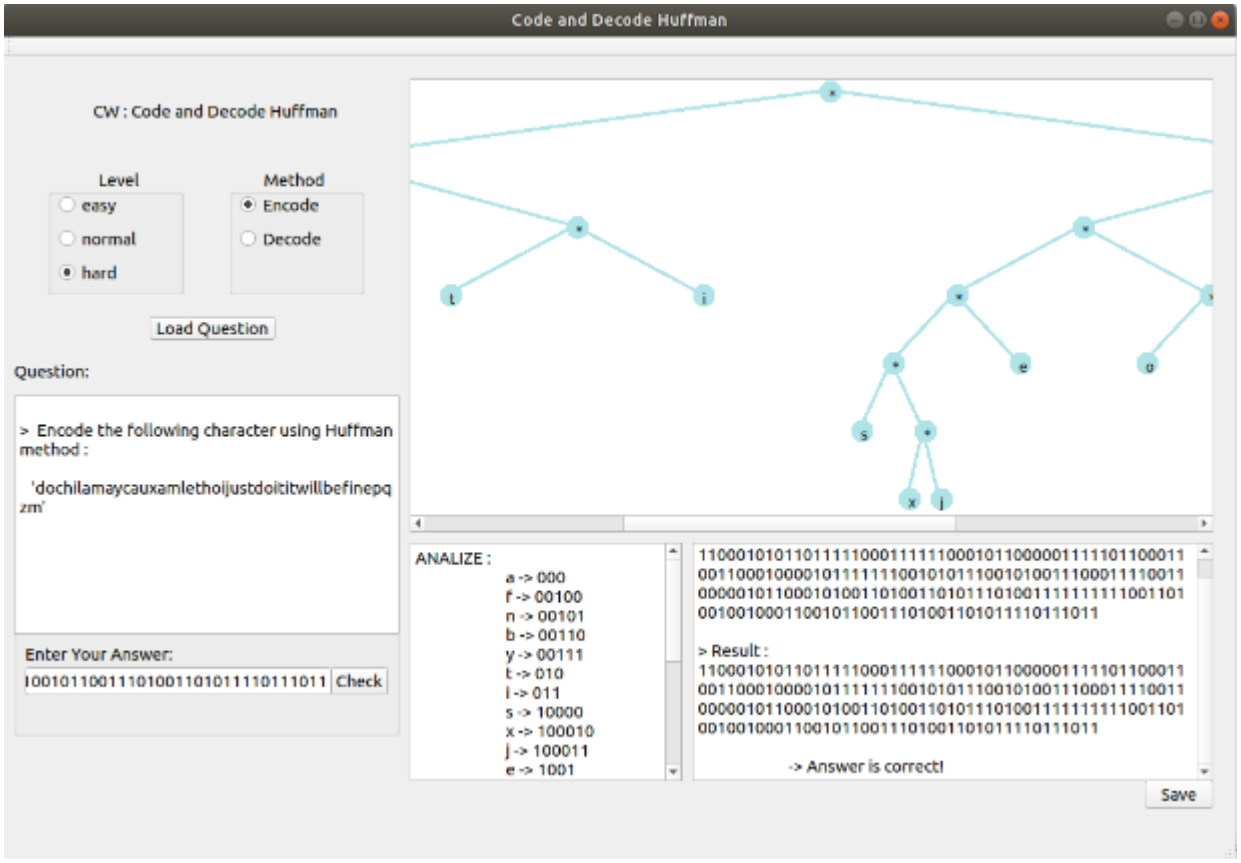


Рисунок 3 – Encode – Level : Hard

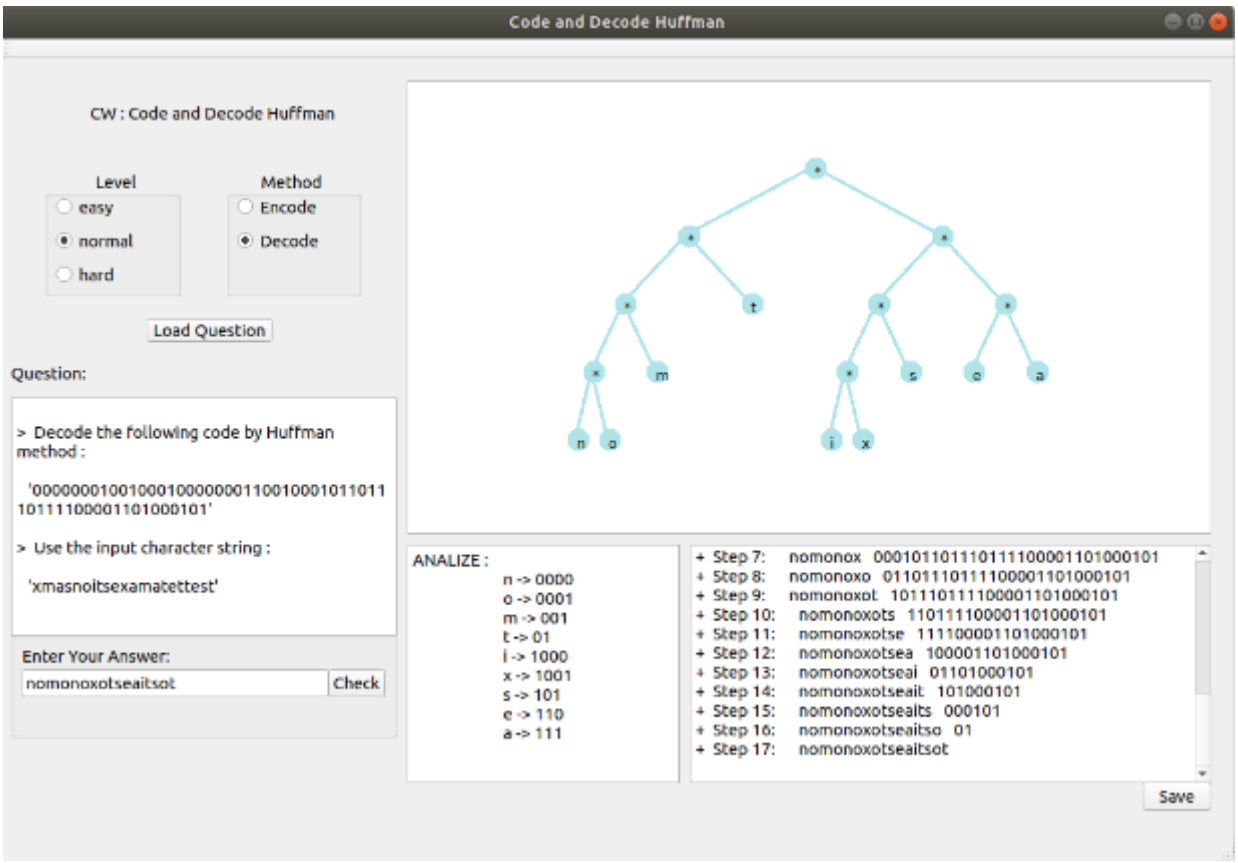


Рисунок 4 – Decode – Level : Normal

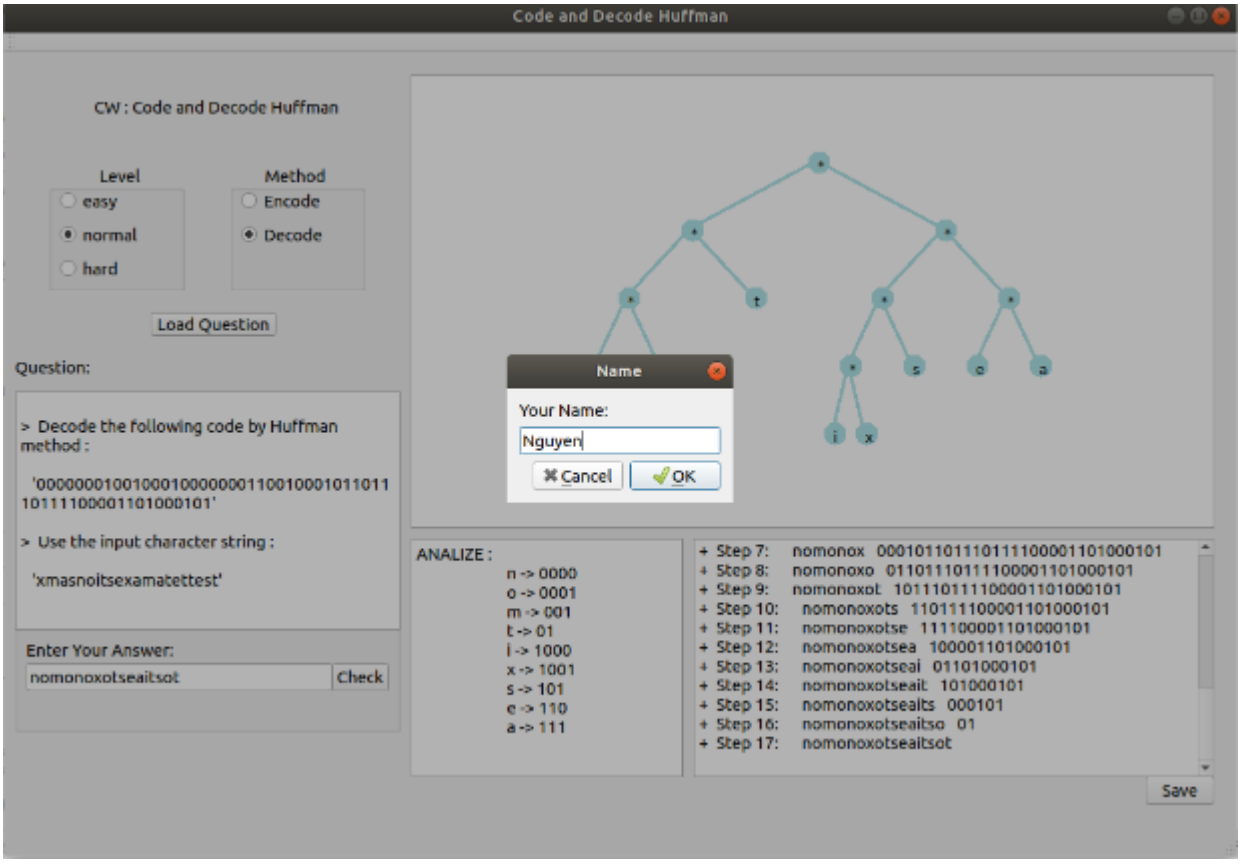


Рисунок 5 – Сохранит результат

После завершения задачи, если вы хотите сохранить результаты, выдается запрос на ввод имени

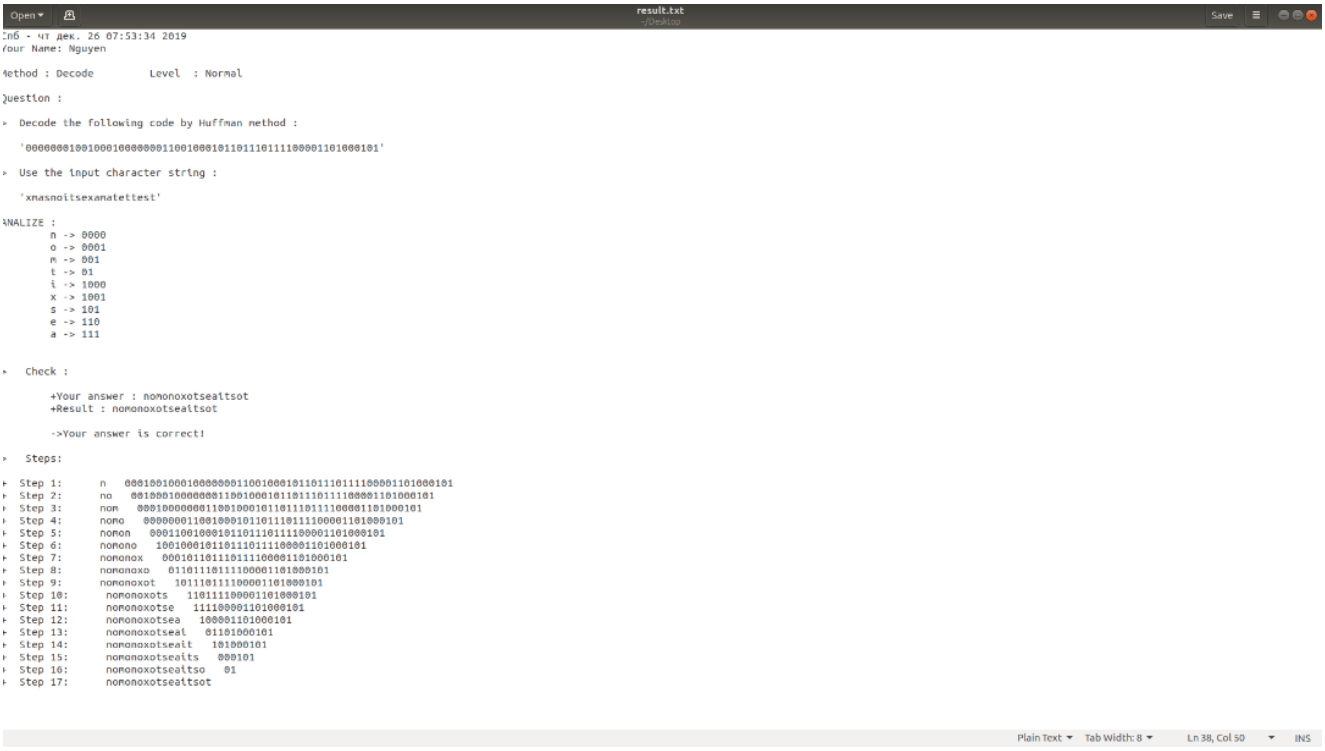


Рисунок 6 – результат в файле result.txt

Заключение

В ходе выполнения данной курсовой работы была изучена теория по основным понятиям и приёмам кодирования/декодирования информации алгоритмами Хаффмана, а также написана программа, выполняющая требуемые действия по кодированию информации. Создание программы для генерации заданий с ответами к ним для проведения текущего контроля среди студентов. Задания и ответы выводиться в файл в удобной форме.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Bjarne Stroustrup. A Tour of C++. M.: Addison-Wesley, 2018. 217 с.
2. П
3. Qt Documentation // Qt. URL: https://doc.qt.io/qt-5/index.html

р

е

в

о

д

и

д

о

п

о

л

н

е

н

и

е

д

о

к

у

м

е

н

т

а

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

Исходный код программы. Main.c

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

Исходный код программы. Mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <fstream>
#include <string>
#include <QGraphicsScene>
#include <QPainter>
#include <QFileDialog>
#include <QMessageBox>
#include <QString>
#include <QTextEdit>
#include <QInputDialog>
#include <QDateTime>
```

```
#include "console.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_radioButton_4_clicked();

    void on_radioButton_5_clicked();

    void on_radioButton_clicked();

    void on_radioButton_2_clicked();

    void on_radioButton_3_clicked();

    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_pushButton_3_clicked();
```

```
    private:
        Ui::MainWindow *ui;
        QGraphicsScene *scene;

        bool flagEncode = false;
        bool flagDecode = false;
        bool flagEasy = false;
        bool flagNormal = false;

        console cuaso;
        string input;
        string encode;
        string question_text;
        string analize;
        string result;

    };
```

e
n
d
i
f
M
A
I
N
W
I
N
D
O

Исходный код программы. Mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
```

```
    //resize(360,332);
    ui->setupUi(this);
    this->setWindowTitle("Code and Decode Huffman");
    //ui->groupBox_3->hide();
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_radioButton_4_clicked()
{
    flagEncode = true;
    flagDecode = false;
}

void MainWindow::on_radioButton_5_clicked()
{
    flagDecode = true;
    flagEncode = false;
}

void MainWindow::on_radioButton_clicked()
{
    flagEasy = true;
    flagNormal = false;
    flagHard = false;
}
```

```
void MainWindow::on_radioButton_2_clicked()
{
    flagNormal = true;
    flagHard = false;
    flagEasy = false;
}

void MainWindow::on_radioButton_3_clicked()
{
    flagHard = true;
    flagNormal = false;
    flagEasy = false;
}

void MainWindow::on_pushButton_clicked()
{
    if(flagDecode == false && flagEncode == false){
        QMessageBox::critical(this,"ERROR","Please select a method");
        return;
    }
    if(flagEasy == false && flagNormal == false && flagHard == false){
        QMessageBox::critical(this,"ERROR","Please select a level");
        return;
    }
    resize(360,640);
    int flag;
    int temp;
    if(flagEncode){
        flag = 1;
```

```
    if(flagEasy){
        temp = 11;
        input = cuaso.take_string(&temp);
    }
    else if (flagNormal){
        temp = 12;
        input = cuaso.take_string(&temp);
    }
    else{
        temp = 13;
        input = cuaso.take_string(&temp);
    }
    question_text = cuaso.create_question_text(input,"",flag);
}

else {
    flag = 2;
    if(flagEasy){
        temp = 11;
        int num = temp;
        input = cuaso.take_string(&temp);
        encode = cuaso.take_encode(num,temp);
    }
    else if (flagNormal) {
        temp = 12;
        int num = temp;
        input = cuaso.take_string(&temp);
        encode = cuaso.take_encode(num,temp);
    }
    else {
```

```
            temp = 13;
            int num = temp;
            input = cuaso.take_string(&temp);
            encode = cuaso.take_encode(num,temp);
        }
        question_text = cuaso.create_question_text(input,encode,flag);
    }
    ui->textEdit->setText(QString::fromStdString(question_text));

}



void MainWindow::on_pushButton_2_clicked()
{
    string youranswer =  (ui->lineEdit->text()).toStdString();
    if(youranswer == ""){
        QMessageBox::critical(this,"ERROR","Please enter your answer.");
        return;
    }
    resize(1091,728);


    //string result;
    scene = cuaso.Paint(input);
    ui->graphicsView->setScene(scene);

    analize = cuaso.analize(input);
    ui->textEdit_2->setText(QString::fromStdString(analize));
    if(flagEncode){
```

```
        result = cuaso.take_encode(input,youranswer);
        ui->textEdit_3->setText(QString::fromStdString(result));
    }
    else if(flagDecode){
        result = cuaso.result(input,encode,youranswer);
        ui->textEdit_3->setText(QString::fromStdString(result));
    }

}

void MainWindow::on_pushButton_3_clicked()
{
    string content ="";
    QString name = QInputDialog::getText(this, tr("Name"),
                                         tr("Your Name:"), QLineEdit::Normal,
                                         QDir::home().dirName());
    QDateTime dt = QDateTime::currentDateTime();
    content += "Спб - ЛЕТИ - " + (dt.toString()).toStdString();
    content += "\nYour Name: ";
    content += name.toStdString() + "\n\n";
    content += "Method : ";
    if(flagDecode)
        content += "Decode\t\tLevel  : ";
    else
        content += "Encode\t\tLevel  : ";
    if(flagEasy)
        content += "Easy\n\n";
    else if(flagNormal)
        content += "Normal\n\n";
    else
```

```
    content += "Hard\n\n";
content+= "Question :\n" + question_text + "\n\n";
content += analize + "\n\n";
content += result + "\n\n";
```

```
QString filePath = QFileDialog::getSaveFileName(this, tr("save"), "/home/nguyenhai/Desktop/result.txt", tr("*.txt"));
        if (QString::compare(filePath, QString()) != 0)
        {
            std::ofstream ff(qPrintable(filePath));
            ff << qPrintable(QString::fromStdString(content));// text chua noi dung cua cai can ghi
```

Исходный код программы. Console.h

```
#ifndef CONSOLE_H
#define CONSOLE_H
#include "huffman.h"

#include <QGraphicsScene>
#include <QGraphicsView>
#include <cmath>
#include <QGraphicsTextItem>
#include <QString>
```

```
#include <QMessageBox>
#include <QLineEdit>

#define RUSSIAN "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюя"

class console
{
public:
    console();
    string take_string(int *flag);
    string take_encode(int num, int random);
    string create_question_text(string str1, string str2, int fl);

    QGraphicsScene *Paint(string data);
    void treePainter(QGraphicsScene *&scene, MinHeapNode *root, int w, int h, int wDelta, int hDelta, QPen &pen, QBrush &brush, QFont &font, int depth);
    int checkErr(QString input, QString encode);
    int CountDeep(MinHeapNode *root);
    string analize(string data);
    string result(string input,string encode, string youranswer);
    void free(MinHeapNode* minheap);
    string take_encode(string input,string youranswer);

    huffman huffman;

    void free(MinHeap * a, MinHeapNode* b);
};
```

```
#endif // CONSOLE_H
```

Исходный код программы. Console.cpp

```
#include "console.h"

console::console()
{

}

string console::take_string(int *flag){
    string question ;
    int num = *flag;
    srand (time(NULL));
    int random = rand()%3 +1;

    if(num == 11){
        if(random == 1)
            question = "hello";
        else if (random == 2)
            question = "where";
        else
            question = "letsgoo";
    }

    else if(num == 12){
        if(random == 1)
            question = "corlorfullsunshine";
        else if(random == 2)
```

```
            question = "whenthecitylightsup";
        else
            question = "xmasnoitsexamatettest";
    }

    else if (num == 13) {
        if(random == 1)
            question = "jcuocsongnaycobietbaonhieuthucanphaithuchiend";
        else if(random == 2)
            question = "vivaydunglangphithoigianvatuoitrecuabanokay";
        else
            question = "dochilamaycauxamlethoijustdoititwillbefinepqzm";
    }

    *flag = random;
    return question;
}

string console::take_encode(int num, int random){
    string encode ;
    if (num == 11) {
        if(random == 1)
            encode = "110111100111";
        else if(random == 2)
            encode = "1110110100111";
        else
            encode = "110100101000010";
    }

    else if (num == 12) {
```

```
            if(random == 1)
                encode = "1100110100011101010011011000001000111010111010";
            else if(random == 2)
                encode = "011100111000010000011010101000100011101111111000110";
            else
                encode =
"000000010010001000000011001000101101110111100001101000101";
        }

        else{
            if(random == 1)
                encode =
"110110001001111101111100001111111011010000100010110101111010111111010011
101000011101111110110111111O";
            else if(random == 2)
                encode =
"001101100011000110011101010111110111101110010010011101001101101001110111
101011111111011100100110001l0";
            else
                encode =
"111111101111101011100110111101011101001100111000101110101001100011100010
010000011010001110011000101";
        }
        return encode;
    }

    string console::create_question_text(string str1, string str2, int fl){
        string question = "";

        if(fl == 1){
```

```
        question += "\n> Encode the following character using Huffman method :";
        question += "\n\n";
        question += "   '" + str1 + "'";
    }
    else {
        question += "\n> Decode the following code by Huffman method :";
        question += "\n\n";
        question += "   '" + str2 + "'";
        question += "\n";
        question += "\n> Use the input character string :";
        question += "\n\n";
        question += "   '" + str1 + "'";
    }
    return question;
}

int console::checkErr(QString input, QString encode){
    if(input.size() == 0 || encode.size() == 0)
        return 3;
    static QString russian = RUSSIAN;
    foreach(const QChar & ch, russian) {
            if(input.contains(ch)) {
                return 1;
            }
    }
    for(int i = 0; i < encode.length(); i++){
        if(encode[i] != '0' && encode[i] != '1')
            return 2;
    }
    return 0;
```

```
}
int console::CountDeep(MinHeapNode *root)
{
    if (root == nullptr)
        return 0;
    int cl = CountDeep(root->left);
    int cr = CountDeep(root->right);
    return 1 + ((cl>cr)?cl:cr);
}

QGraphicsScene* console::Paint(string data){
    QGraphicsScene *scene = new QGraphicsScene;
    string input = data;
    MinHeapNode *root = huffman.buildHuffmanTree(input);
    int deeptree = CountDeep(root);

    scene->clear();
    QPen pen;
    QColor color;
    color.setRgb(174, 227, 232);
    pen.setColor(color);
    QBrush brush (color);
    QFont font("Helvetica [Cronyx]", 9, 10,false);
    pen.setWidth(3);


    int wDeep = static_cast<int>(pow(2, deeptree));
    int hDelta = 60;
    int wDelta = 7;
    int width = (wDelta*wDeep)/2;
```

```
        treePainter(scene, root, width/2, hDelta, wDelta, hDelta, pen, brush, font, wDeep);
        free(root);
        return scene;



    }

    void console::treePainter(QGraphicsScene *&scene, MinHeapNode *root, int w, int h, int wDelta, int hDelta, QPen &pen, QBrush &brush, QFont &font, int depth)
    {

        if (root == nullptr)
            return ;
        string out;
        out += root->data;
        QGraphicsTextItem *textItem = new QGraphicsTextItem;
        textItem->setPos(w, h);// set toa do (x;y) cua nut
        textItem->setPlainText(QString::fromStdString(out));
        textItem->setFont(font);
        scene->addEllipse(w-wDelta/2, h, wDelta*5/2, wDelta*5/2, pen, brush); // Tạo hình tròn của các nút
        if (root->left != nullptr)
            scene->addLine(w+wDelta/2,  h+wDelta,  w-(depth/2)*wDelta+wDelta/2, h+hDelta+wDelta, pen);
        if (root->right != nullptr)
            scene->addLine(w+wDelta/2,  h+wDelta,  w+(depth/2)*wDelta+wDelta/2, h+hDelta+wDelta, pen);
        scene->addItem(textItem);
```

```
        treePainter(scene, root->left, w-(depth/2)*wDelta, h+hDelta, wDelta, hDelta, pen, brush, font, depth/2);
        treePainter(scene, root->right, w+(depth/2)*wDelta, h+hDelta, wDelta, hDelta, pen, brush, font, depth/2);
        return ;
    }

    string console::analize(string data){
        string out;
        out += "ANALIZE : \n";
        int ch[MAX_TREE_HT], top = 0;
        MinHeapNode *minheap = huffman.buildHuffmanTree(data);
        out += huffman.printCodes(minheap,ch,top);
        free(minheap);
        return out;
    }

    string  console::result(string input,string encode,string youranswer){
        return huffman.decode(huffman.buildHuffmanTree(input),encode,youranswer);
    }

    string console::take_encode(string input,string youranswer){
            string out;
            int ch[MAX_TREE_HT], top = 0;
            MinHeapNode *minheap = huffman.buildHuffmanTree(input);
            out += huffman.take_encode(minheap,ch,top,input);
            string buffer = "";
            buffer += "> Your answer :  ";
            buffer += youranswer + "\n\n";
            buffer += "> Result :  ";
```

```
        buffer += out + "\n\n";
        if(out == youranswer)
            buffer += "\t-> Answer is correct!";
        else
            buffer += "\t-> The answer inccorect!";
```

```
void console::free(MinHeapNode* minheap){
    if(minheap == nullptr)
        return;

    free(minheap->left);
    free(minheap->right);

    delete minheap;
```

Исходный код программы. Huffman.h

```
#ifndef HUFFMAN_H
#define HUFFMAN_H
#include <iostream>
#include <cstdlib>
#include <string>
#include <algorithm>
#include <stdlib.h>
#include <time.h>
```

```
using namespace std;
#define MAX_TREE_HT 100


typedef struct MinHeapNode{
   char data;
   unsigned freq;
   MinHeapNode *left, *right;
}MinHeapNode;

typedef struct MinHeap{
   unsigned size;
   unsigned capacity;
   MinHeapNode** array;
}MinHeap;

class huffman
{
public:
   huffman();
   MinHeapNode* newNode(char data, unsigned freq);
   MinHeap* createMinHeap(unsigned capacity);
   void swapMinHeapNode( MinHeapNode** a, MinHeapNode** b);
   void minHeapify(MinHeap* minHeap, int idx);

   MinHeapNode* extractMin(MinHeap* minHeap) ;
   void insertMinHeap( MinHeap* minHeap, MinHeapNode* minHeapNode);
   void buildMinHeap( MinHeap* minHeap);
   int isLeaf( MinHeapNode* root);
   MinHeap* createAndBuildMinHeap(char data[], int freq[], int size);
```

```
    MinHeapNode* buildHuffmanTree(string input);
    string decode(MinHeapNode* root, string s,string youranswer);
    void take_data_freq(string input, char* data, int* freq, int* size);
    string printCodes( MinHeapNode* root, int arr[], int top);
    string printArr(int arr[], int n);
    string take_encode(MinHeapNode* root, int *arr, int top,string input);
    string take_encode_3(int *arr, int n);
    string take_encode_2(MinHeapNode* root, int *arr, int top, char input);
};

#endif // HUFFMAN_H
```

Исходный код программы. Huffman.cpp

```
#include "huffman.h"

huffman::huffman()
{

}

MinHeapNode* huffman::newNode(char data, unsigned freq)
{
    MinHeapNode* temp = new MinHeapNode;
    temp->left = temp->right = nullptr;
    temp->data = data;
    temp->freq = freq;
    return temp;
}

MinHeap* huffman::createMinHeap(unsigned capacity) {
```

```
        MinHeap* minHeap = new MinHeap;
        // current size is 0
        minHeap->size = 0;
        minHeap->capacity = capacity;
        minHeap->array = (MinHeapNode**)malloc(capacity * sizeof(MinHeapNode*));
        return minHeap;
    }
    void huffman::swapMinHeapNode( MinHeapNode** a, MinHeapNode** b){
        MinHeapNode* t = *a;
        *a = *b;
        *b = t;
    }

    void huffman::minHeapify(MinHeap* minHeap, int idx){

        int smallest = idx;
        int left = 2 * idx + 1;
        int right = 2 * idx + 2;

        if (left <static_cast<int>(minHeap->size) && minHeap->array[left]->freq < minHeap->array[smallest]->freq)
            smallest = left;

        if (right < static_cast<int>(minHeap->size) && minHeap->array[right]->freq < minHeap->array[smallest]->freq)
            smallest = right;

        if (smallest != idx) {
```

```
        swapMinHeapNode(&minHeap->array[smallest], &minHeap->array[idx]);
        minHeapify(minHeap, smallest);
    }
}

int huffman::isSizeOne(MinHeap* minHeap){
    return (minHeap->size == 1);
}

MinHeapNode* huffman::extractMin(MinHeap* minHeap) {
    MinHeapNode* temp = minHeap->array[0];
    minHeap->array[0]
        = minHeap->array[minHeap->size - 1];

    --minHeap->size;
    minHeapify(minHeap, 0);

    return temp;
}

void huffman::insertMinHeap( MinHeap* minHeap, MinHeapNode*
minHeapNode) {

    ++minHeap->size;
    int i = minHeap->size - 1;

    while (i && minHeapNode->freq < minHeap->array[(i - 1) / 2]->freq) {

        minHeap->array[i] = minHeap->array[(i - 1) / 2];
        i = (i - 1) / 2;
```

```
    }
    minHeap->array[i] = minHeapNode;
}

void huffman::buildMinHeap( MinHeap* minHeap){
    int n = minHeap->size - 1;
    int i;
    for (i = (n - 1) / 2; i >= 0; --i)
        minHeapify(minHeap, i);
}

int huffman::isLeaf( MinHeapNode* root){
    return !(root->left) && !(root->right);
}

MinHeap* huffman::createAndBuildMinHeap(char data[], int freq[], int size){

    MinHeap* minHeap = createMinHeap(size);
    for (int i = 0; i < size; ++i)
        minHeap->array[i] = newNode(data[i], freq[i]);
    minHeap->size = size;
    buildMinHeap(minHeap);

    return minHeap;
}

MinHeapNode* huffman::buildHuffmanTree(string input){
    int len = static_cast<int>(input.length());
    char *data = new char[len];
    int *freq = new int[len];
```

```
int size = 0;
take_data_freq(input,data,freq, &size);

MinHeapNode *left, *right, *top;

// Step 1: Create a min heap of capacity
// equal to size. Initially, there are
// modes equal to size.
 MinHeap* minHeap = createAndBuildMinHeap(data, freq, size);

// Iterate while size of heap doesn't become 1
while (!isSizeOne(minHeap)) {

    // Step 2: Extract the two minimum
    // freq items from min heap
    left = extractMin(minHeap);
    right = extractMin(minHeap);

    // Step 3: Create a new internal
    // node with frequency equal to the
    // sum of the two nodes frequencies.
    // Make the two extracted node as
    // left and right children of this new node.
    // Add this node to the min heap
    // '$' is a special value for internal nodes, not used
    top = newNode('*', left->freq + right->freq);

    top->left = left;
    top->right = right;
```

```
        insertMinHeap(minHeap, top);
    }

    // Step 4: The remaining node is the
    // root node and the tree is complete.
    return extractMin(minHeap);
}

string huffman::printArr(int *arr, int n)
{
    string out;
    int i;
    for (i = 0; i < n; ++i){
        if(arr[i] == 1 )
            out.append(1,'1');
        else if(arr[i] == 0)
            out.append(1,'0');
    }
    out+= "\n";
    return out;
}

string huffman::printCodes( MinHeapNode* root, int *arr, int top){

    string out;

    if (root->left) {
        arr[top] = 0;
        out += printCodes(root->left, arr, top + 1);
    }
```

```
    if (root->right) {

        arr[top] = 1;
        out += printCodes(root->right, arr, top + 1);
    }

    if (isLeaf(root)) {
        out+= '\t';
        out += root->data;
        out += " -> ";
        out += printArr(arr, top);
    }
    return out;
}

string huffman::take_encode(MinHeapNode* root, int *arr, int top, string input){
    string out;
    int len = input.size();
    for(int i = 0; i < len; i++){
        out += take_encode_2(root,arr,top,input[i]);
    }
    return out;
}

string huffman::take_encode_2(MinHeapNode* root, int *arr, int top, char input){
    string out;
    if (root->left) {
        arr[top] = 0;
        out += take_encode_2(root->left, arr, top + 1,input);
```

```
    }

    if (root->right) {

        arr[top] = 1;
        out += take_encode_2(root->right, arr, top + 1,input);
    }

    if (isLeaf(root) && input == root->data) {
        out += take_encode_3(arr,top);
    }
    return out;
}

string huffman::take_encode_3(int *arr, int n)
{
    string out;
    int i;
    for (i = 0; i < n; ++i){
        if(arr[i] == 1 )
            out.append(1,'1');
        else if(arr[i] == 0)
            out.append(1,'0');
    }
    return out;
}
string huffman::decode(MinHeapNode* root, string s,string youranswer)
{
    string ans = "";
    string step ;
```

```
int id = 1;
struct MinHeapNode* curr = root;
int len = static_cast<int>(s.size());
for (int i=0; i < len; i++)
{
   if (s[i] == '0')
      curr = curr->left;
   else
      curr = curr->right;
   if (curr->left==nullptr and curr->right==nullptr)
   {
      step += "+  Step " + std::to_string(id) + ":      ";
      id++;
      ans += curr->data;
      step += ans;
      step += "    ";
      step.append(s,i+1,len-i-1);
      step += '\n';
      curr = root;
   }
}
string buffer = ">   Check :   \n\n";
buffer += "\t+Your answer : ";
buffer += youranswer;
buffer += "\n\t+Result : ";
buffer += ans;
if(ans == youranswer)
   buffer += "\n\n\t->Your answer is correct!\n\n";
else
   buffer += "\n\n\t->Your answer inccorect!\n\n";
```

```
    buffer += ">   Steps:\n\n";
    buffer += step;
    return buffer;
}

void huffman::take_data_freq(string input, char* data, int* freq, int* size){
    int id = 1;
    data[0] = input[0];
    for(int i = 0; i <static_cast<int>(input.length()) ; i++){
        int flag = 0;
        for(int j = 0; j < id; j++){
            if(data[j] == input[i]){
                flag = 1;
                break;
            }
        }
        if(flag == 0){
            data[id] = input[i];
            id++;
        }
    }
    *size = id;
    data[id] = '\0';
    freq[id] = '\0';
    for(int i = 0; i < id; i++){
        freq[i] = count(input.begin(),input.end(),data[i]);
    }
    return;
}
```