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

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

Тема: Кодирование и декодирование Хаффмана и Шеннона-Фано

Студент гр. 7383 _____ Бергалиев М.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Бергалиев М.А.

Группа 7383

Тема работы: Кодирование и декодирование Хаффмана и Шеннона-Фано

Исходные данные:
Написать программу, генерирующую задания по кодированию и декодированию Хаффмана и Шеннона-Фано.

Содержание пояснительной записки:

- Содержание.
- Введение.
- Первый раздел: формулировка задачи.
- Второй раздел: решение задачи.
- Третий раздел: тестирование.
- Заключение.
- Приложение А. Исходный код программы.

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент _____ Бергалиев М.А.

Преподаватель _____ Размочаева Н.В.

АННОТАЦИЯ

В курсовой работе была реализована программа с графическим интерфейсом, генерирующая задания по кодированию и декодированию Хаффмана и Шеннона-Фано по опциям, задаваемым пользователем.

SUMMARY

In the course work was implemented a program with a graphical interface that generates tasks for encoding and decoding of Huffman and Shannon-Fano according to options set by the user.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью работы является практическое освоение стандартных алгоритмов статического кодирования и декодирования.

Задачей является создать программу, генерирующую задания по статическому кодированию и декодированию Хаффмана и Шеннона-Фано.

Тестирование будет проводится в Qt 5.10.1 в OS Linux Ubuntu 16.04 LTS.

1. ФОРМУЛИРОВКА ЗАДАЧИ И ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ.

Создать программу для генерации заданий с ответами к ним для проведения текущего контроля среди студентов. Задания и ответы должны выводиться в файл в удобной форме; ответы должны позволять удобную проверку правильности выполнения заданий.

Алгоритм Шеннона-Фано.

Кодирование Шеннона-Фано — алгоритм префиксного неоднородного кодирования.

Основные этапы:

1. Символы первичного алфавита выписывают по убыванию частоты.
2. Символы полученного алфавита делят на две части, суммарные частоты символов которых максимально близки друг другу.
3. В префиксном коде для первой части алфавита присваивается бит 1, второй части — бит 0.
4. Полученные части рекурсивно делятся и их частям назначаются соответствующие двоичные цифры в префиксном коде.

Алгоритм Хаффмана.

Алгоритм Хаффмана — жадный алгоритм оптимального префиксного кодирования алфавита с минимальной избыточностью.

Основные этапы:

1. Символы входного алфавита образуют список свободных узлов. Каждый лист имеет вес, который может быть равен либо вероятности, либо количеству вхождений символа в сжимаемое сообщение.
2. Выбираются два свободных узла дерева с наименьшими весами.
3. Создается их родитель с весом, равным их суммарному весу.

4. Родитель добавляется в список свободных узлов, а два его потомка удаляются из этого списка.

5. Одной дуге, выходящей из родителя, ставится в соответствие бит 1, другой — бит 0. Битовые значения ветвей, исходящих от корня, не зависят от весов потомков.

6. Шаги, начиная со второго, повторяются до тех пор, пока в списке свободных узлов не останется только один свободный узел. Он и будет считаться корнем дерева.

2. РЕШЕНИЕ ЗАДАЧИ.

Класс CodeTree представляет из себя дерево кодов. Класс содержит следующие поля:

- symbols — символы в данном дереве.
- left — указатель на левое поддерево.
- right — указатель на правое поддерево.

Статический метод ShannonFano принимает на вход множество символов с их частотами встречаемости. Все символы из данного множества записываются в данный узел. Создается два подмножества с примерно одинаковыми общими весами. Левое поддерево создается из подмножества с большим весом, а правое — с меньшим. Используемые переменные:

- weight — общий вес символов.
- sum — вес символов подмножества.
- res — созданное дереов кодов.

Статический метод Huffman принимает на вход множество символов с их частотами встречаемости. Для каждого символа создается свой узел. Далее на каждом шаге два узла с наименьшими весами присоединяются к новому узлу, который является их родителем, а эти два узла удаляются из множества. Левое поддерево имеет больший вес, а правое — меньший. Используемые переменные:

- weight — общий вес символов.
- sum — вес символов подмножества.
- nodes — множество узлов.
- res — созданное дерево кодов.

Метод encode принимает незакодированную строку. Проходит строку, параллельно проходя по кодовому дереву, идя в левое поддерево, если в его корне содержится символ из строки и в правое поддерево, если влево поддереве его нет, записывая в результирующую строку «1» или «0»

соответственно. Дойдя до листа, обход начинается сначала для следующего незакодированного символа. Используемые переменные:

- cur — текущий узел в обходе.
- res — закодированное сообщение.

Метод decode принимает закодированную строку. Проходит строку, параллельно проходя по кодовому дереву, идя в левое поддерево при встрече с 1 и в правое поддерево, если с 0. Дойдя до листа, записывает в результирующую строку символ из листа. Используемые переменные:

- cur — текущий узел в обходе.
- res — декодированное сообщение.

Графический интерфейс был реализован в Qt. Код программы представлен в приложении А.

3. ТЕСТИРОВАНИЕ ПРОГРАММЫ.

Компиляция была произведена в компиляторе GCC в OS Linux Ubuntu. Функции для работы с BMP-файлами были помещены в файл BMP.c, их прототипы — в BMP.h, a main и arg_parser — в main.c.

Запустили программу, главное окно показано на рис. 1.

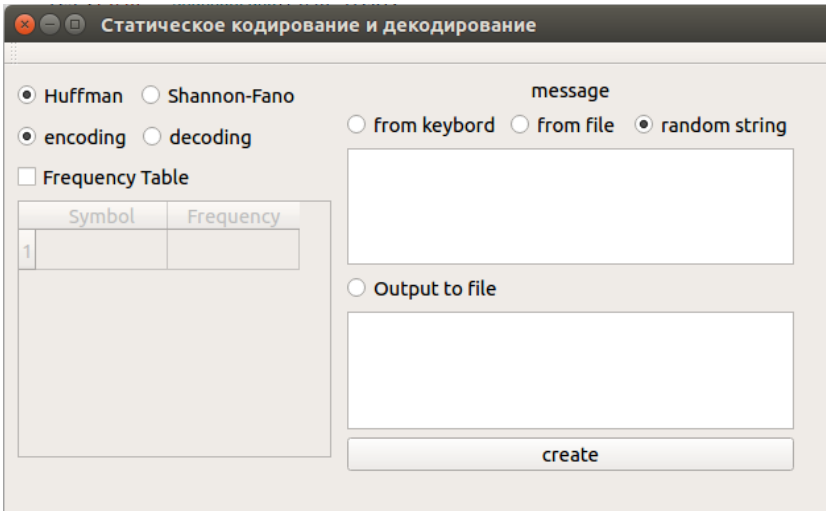

Рисунок 1 — Главное окно программы

Здесь можно выбрать настройки для создаваемой задачи: налгоритм, кодирование или декодирование, таблицу частот символов, сообщение, которое нужно закодировать или декодировать, файл, куда сохранить задание.

При вводе очередного символа и его частоты появляется поле для следующего символа, как показано на рис. 2.

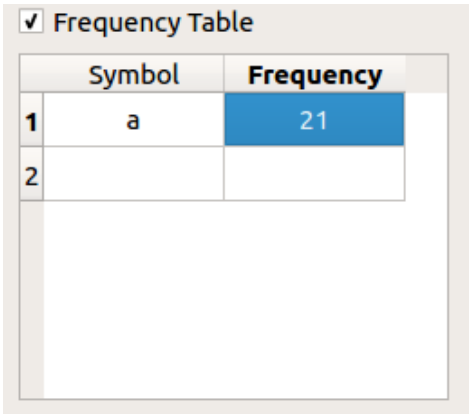

Рисунок 2 — Появление новой строки для следующего символа

При вводе нескольких символов в поле первого столбца или символа, не являющегося цифрой в поле второго столбца, программа сообщит о некорректных данных, как показано на рис. 3.

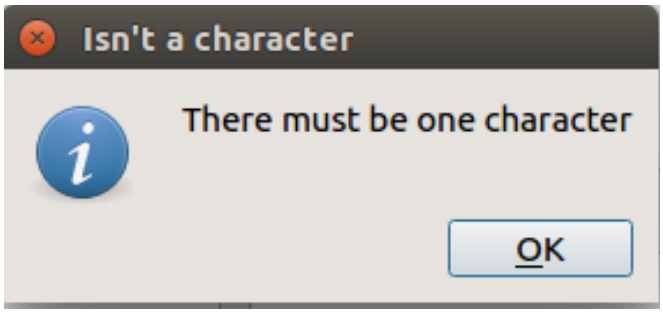

Рисунок 3 — Сообщение о вводе некорректных данных в таблицу

При вводе сообщения с клавиатуры программа открывает окно, где можно ввести сообщение, как показано на рис. 4.

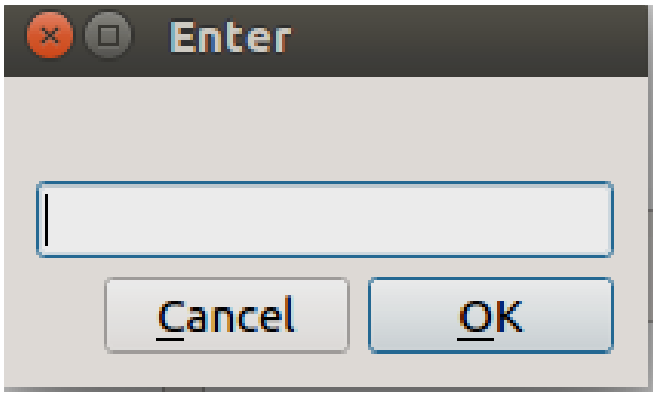

Рисунок 4 — Окно для ввода сообщения

При вводе из файла открывается окно файловой системы, где можно выбрать необходимый файл, как показано на рисунке 5.

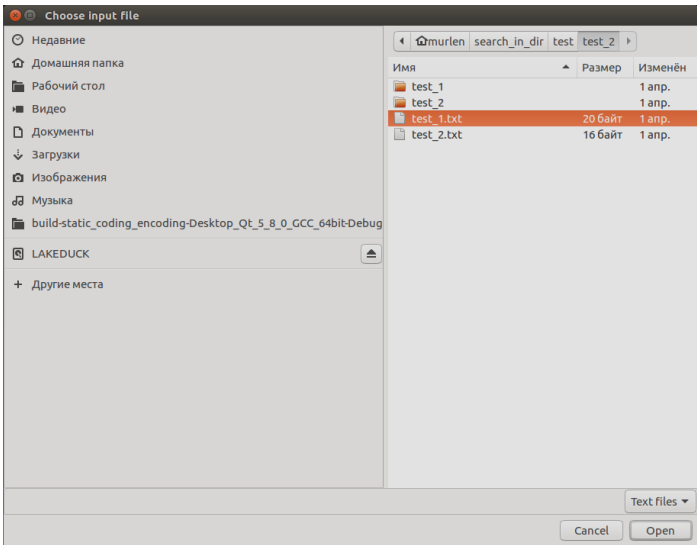

Рисунок 5 — Выбор файла для ввода сообщения

При нажатии кнопки «create» создается окно с заданием. Если была задана таблица, то она также будет присутствовать в окне. Пример показан на рис. 6.

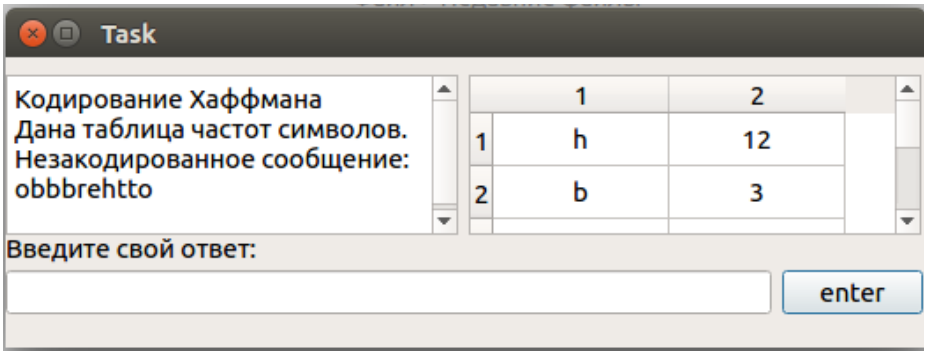

Рисунок 6 — Окно с заданием

При вводе ответа выводится является ли он верным или нет, как показано на рис. 7.

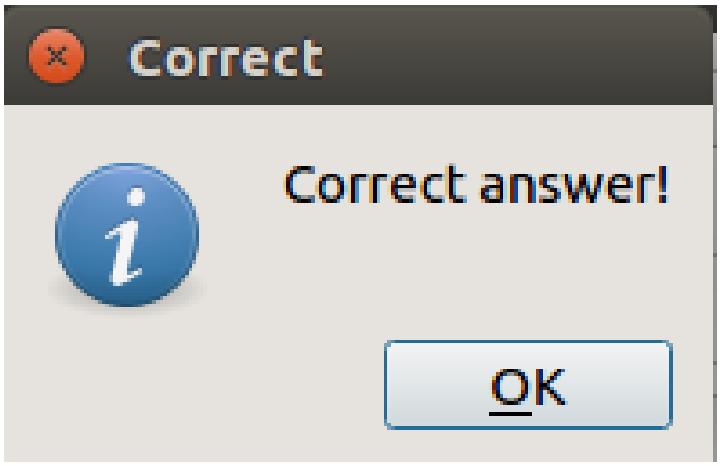

Рисунок 7 — Подтверждение корректности введенного ответа

Если выбрать вывод в файл, то при создании задания программа откроет окно, показанное на рис. 8.

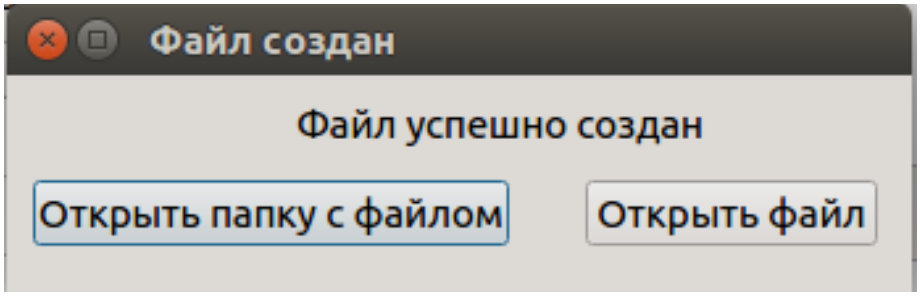

Рисунок 8 — Окно при создании задания с файлом вывода

Здесь можно открыть папку с файлом либо непосредственно файл.

ЗАКЛЮЧЕНИЕ

Были написаны класс кодового дерева, два метода для создания кодового дерева, кодирующие и декодирующие методы, графический интерфейс. Созданная программа предоставляет удобный выбор опций для создания заданий по кодированию и декодированию, сообщает об ошибках пользователя, позволяет простую проверку правильности ответа, выводит текст задания в простом и понятном формате.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл coding.h:

```
#pragma once
#include <set>
#include <utility>

class CodeTree{
public:
    static CodeTree* ShannonFano(std::set<std::pair<int,
char>>);
    static CodeTree* Huffman(std::set<std::pair<int, char>>);
    std::string encode(std::string&);
    std::string decode(std::string&);
    ~CodeTree();
private:
    CodeTree(std::string);
    CodeTree* left;
    CodeTree* right;
    std::string symbols;
};
```

Файл dialog.h:

```
#ifndef DIALOG_H
#define DIALOG_H

#include <QDialog>
#include <QTableWidget>
#include "mainwindow.h"

namespace Ui {
class Dialog;
}

class Dialog : public QDialog
{
    Q_OBJECT

public:
    explicit Dialog(std::vector<std::pair<int, char>>,
std::string given, std::string answer, Options options, QWidget
*parent = 0);
```

```
    ~Dialog();

private slots:
    void on_enterButton_clicked();

private:
    Ui::Dialog *ui;
    std::string answer;
};

#endif // DIALOG_H
```

Файл mainwindow.h:

```
#ifndef MAINWINDOW_H

#define MAINWINDOW_H

#include <QMainWindow>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:
    void on_FreqTableCheck_toggled(bool checked);

    void on_PlusButton_clicked();

    void on_MinusButton_clicked();

    void on_FreqTable_cellChanged(int row, int column);

    void on_encodingButton_clicked();

    void on_decodingButton_clicked();

    void on_createButton_clicked();
```

```
    void on_randomButton_clicked();

    void on_fromFileButton_clicked();

    void on_fromKeyboardButton_clicked();

    void on_toFileButton_clicked(bool checked);

private:
    Ui::MainWindow *ui;
    std::string ifilename;
    std::string ofilename;
    std::string message;
    std::string encoded;
};

struct Options
{
    Options() {}
    bool isEncode;
    bool isHuffman;
    bool isTable;

};
#endif // MAINWINDOW_H
```

Файл opendialog.h:

```
#ifndef OPENDIALOG_H
#define OPENDIALOG_H

#include <QDialog>

namespace Ui {
class OpenDialog;
}

class OpenDialog : public QDialog
{
    Q_OBJECT

public:
    explicit OpenDialog(std::string, QWidget *parent = 0);
    ~OpenDialog();
```

```
private slots:
    void on_openFileButton_clicked();

    void on_openDirButton_clicked();

private:
    Ui::OpenDialog *ui;
    std::string filepath;
};

#endif // OPENDIALOG_H
```

Файл task.h:

```
#ifndef TASK_H
#define TASK_H

#include <tuple>
#include "coding.h"

void create_task(std::vector<std::pair<int, char>> &freq,
std::string &message, std::string &encoded, CodeTree*(*coding)
(std::set<std::pair<int, char>>));

#endif
```

Файл coding.cpp:

```
#include <string>
#include <stdexcept>
#include "coding.h"
#include <iostream>
#include <tuple>

CodeTree::CodeTree(std::string syms){
    left = nullptr;
    right = nullptr;
    symbols = syms;
}

CodeTree* CodeTree::ShannonFano(std::set<std::pair<int, char>>
freq) {
    int weight = 0, sum = 0;
    std::string syms;
    for(auto it : freq) {
```

```
        weight += std::get<0>(it);
        syms.push_back(std::get<1>(it));
    }
    CodeTree* res = new CodeTree(syms);
    if(syms.length() == 1) {
        return res;
    }
    std::set<std::pair<int, char>> left_freq, right_freq;
    auto it = freq.rbegin();
    for(; it != freq.rend(); ++it) {
        if(2*(std::get<0>(*it) + sum)-weight <= 2*sum-weight) {
            sum += std::get<0>(*it);
            left_freq.insert(*it);
        }
        else
            break;
    }
    for(; it != freq.rend(); ++it){
        right_freq.insert(*it);
    }
    if(sum < weight - sum)
        left_freq.swap(right_freq);
    res->left = ShannonFano(left_freq);
    res->right = ShannonFano(right_freq);
    return res;
}

CodeTree* CodeTree::Huffman(std::set<std::pair<int, char>> freq)
{
    int weight = 0, sum = 0;
    std::set<std::tuple<int, std::string, CodeTree*>> nodes;
    std::string syms;
    for(auto it : freq) {
        std::tuple<int, std::string, CodeTree*>
l(std::get<0>(it), std::string(1, std::get<1>(it)), new
CodeTree(std::string(1, std::get<1>(it))));
        nodes.insert(l);
        syms.push_back(std::get<1>(it));
    }
    CodeTree* res = new CodeTree(syms);
    while(nodes.size() > 2){
        auto it = nodes.begin();
        auto r = *it;
        ++it;
        auto l = *it;
```

```
        CodeTree* new_node = new CodeTree(std::get<1>(l)
+std::get<1>(r));
        new_node->left = std::get<2>(l);
        new_node->right = std::get<2>(r);
        std::tuple<int, std::string, CodeTree*>
elem(std::get<0>(l)+std::get<0>(r), new_node->symbols,
new_node);
        nodes.erase(it);
        nodes.erase(nodes.begin());
        nodes.insert(elem);
    }
    res->right = std::get<2>(*nodes.begin());
    res->left = std::get<2>(*++nodes.begin());
    return res;
}

std::string CodeTree::decode(std::string& encoded) {
    CodeTree* cur = this;
    std::string res;
    for(auto it : encoded) {
        if(it == '1')
            cur = cur->left;
        else
            if(it == '0')
                cur = cur->right;
            else
                throw std::invalid_argument("Invalid encoded
string");
        if(cur->left == nullptr) {
            res += cur->symbols;
            cur = this;
        }
    }
    return res;
}

std::string CodeTree::encode(std::string& message){
    CodeTree* cur = this;
    std::string encoded;
    for(auto it : message){
        while(cur->left != nullptr){
            if(cur->left->symbols.find(it) != std::string::npos)
{
                encoded.append("1");
                cur = cur->left;
```

```
            }
            else{
                encoded.append("0");
                cur = cur->right;
            }
        }
        cur = this;
    }
    return encoded;
}

CodeTree::~CodeTree(){
    delete left;
    delete right;
}
```

Файл dialog.cpp:

```
#include "dialog.h"
#include "ui_dialog.h"
#include "mainwindow.h"
#include <set>
#include <cstdlib>
#include <QMessageBox>

Dialog::Dialog(std::vector<std::pair<int, char>> table,
std::string given, std::string answer, Options options, QWidget
*parent) :
    QDialog(parent),
    ui(new Ui::Dialog)
{
    ui->setupUi(this);
    this->answer = answer;
    if(options.isTable){
        ui->freqTableWidget-
>setHorizontalHeaderLabels(QStringList() << "Symbol" <<
"Frequency");
        ui->freqTableWidget->setColumnCount(2);
        ui->freqTableWidget->setRowCount(table.size());
        int j = 0;
        for(auto it : table){
            ui->freqTableWidget->setItem(j, 1, new
QTableWidgetItem(QString(std::to_string(std::get<0>(it)).c_str()
)));
```

```
            ui->freqTableWidget->setItem(j, 0, new
QTableWidgetItem(QString(std::string(1,
std::get<1>(it)).c_str())));
            ui->freqTableWidget->item(j,0)-
>setTextAlignment(Qt::AlignCenter);
            ui->freqTableWidget->item(j,1)-
>setTextAlignment(Qt::AlignCenter);
            ++j;
        }
    }
    else ui->freqTableWidget->hide();
    std::string text;
    if(options.isEncode)
        text = "Кодирование ";
    else text = "Декодирование ";
    if(options.isHuffman)
        text += "Хаффмана\n";
    else text += "Шеннона-Фано\n";
    if(options.isTable)
        text += "Дана таблица частот символов. ";
    if(options.isEncode)
        text += "Незакодированное сообщение:\n";
    else text += "Закодированное сообщение:\n";
    text += given;
    text += "\n";

    ui->taskText->setText(QString(text.c_str()));
}

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

void Dialog::on_enterButton_clicked()
{
    if(answer == ui->InputLine->text().toStdString())
        QMessageBox::information(this, tr("Correct"),
tr("Correct answer!"));
    else QMessageBox::information(this, tr("Incorrect"),
tr("Incorrect answer!"));
}
```

Файл main.cpp:

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

```
#include <QApplication>

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

    return a.exec();
}
```

Файл mainwindow.cpp:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "dialog.h"
#include "task.h"
#include "coding.h"
#include "opendialog.h"
#include <vector>
#include <set>
#include <fstream>
#include <QMessageBox>
#include <QFileDialog>
#include <QInputDialog>
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->FreqTable->setRowCount(1);
    ui->FreqTable->setColumnCount(2);
    QColor color;
    color.setRgb(239, 234, 234);
    ui->FreqTable->setHorizontalHeaderLabels(QStringList() <<
"Symbol" << "Frequency");
}

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

void MainWindow::on_FreqTableCheck_toggled(bool checked)
{
     ui->FreqTable->setEnabled(checked);
```

```
}

void MainWindow::on_FreqTable_cellChanged(int row, int column)
{
    auto item = ui->FreqTable->item(row, column);
    item->setTextAlignment(Qt::AlignCenter);
    if(item->text().length() == 0){
        if(ui->FreqTable->item(row+1, column) != nullptr){
            ui->FreqTable->setRowCount(row+1);
        }
        return;
    }
    if(column == 0){
        if(item->text().length() > 1){
            QMessageBox::information(this, tr("Isn't a
character"), tr("There must be one character"));
            item->setText(tr(""));
            return;
        }
        else{
            for(int i=0; i<ui->FreqTable->rowCount(); ++i)
                if(ui->FreqTable->item(i, 0) != nullptr && row !
= i && ui->FreqTable->item(i, 0)->text().length() != 0 && ui-
>FreqTable->item(i,0)->text() == item->text()){
                    QMessageBox::information(this, tr("Already
in table"), tr("This character is already in table"));
                    item->setText(tr(""));
                    return;
                }
        }
    }
    else{
        for(auto i : item->text().toStdString())
            if(!isdigit(i)){
                QMessageBox::information(this, tr("Not a
number"),tr("Frequency must be a number"));
                item->setText(tr(""));
                return;
            }
    }
    if(ui->FreqTable->item(row, 1-column) != nullptr && ui-
>FreqTable->item(row, 1-column)->text().length() != 0){
        QColor color;
        color.setRgb(255, 255, 255);
        if(ui->FreqTable->item(row+1, 0) == nullptr){
```

```
            ui->FreqTable->setRowCount(row+2);
            ui->FreqTable->setItem(row+1, 0, new
QTableWidgetItem());
            ui->FreqTable->setItem(row+1, 1, new
QTableWidgetItem());
        }
        ui->FreqTable->item(row+1, 0)-
>setFlags(Qt::ItemIsEditable | Qt::ItemIsEnabled |
Qt::ItemIsSelectable);
        ui->FreqTable->item(row+1, 0)->setBackground(color);
        ui->FreqTable->item(row+1, 1)-
>setFlags(Qt::ItemIsEditable | Qt::ItemIsEnabled |
Qt::ItemIsSelectable);
        ui->FreqTable->item(row+1, 1)->setBackground(color);
    }
}

void MainWindow::on_encodingButton_clicked()
{
    ui->FreqTableCheck->setEnabled(true);
}

void MainWindow::on_decodingButton_clicked()
{
    if(!ui->FreqTableCheck->isChecked())
        ui->FreqTableCheck->toggle();
    ui->FreqTable->setEnabled(true);
    ui->FreqTableCheck->setEnabled(false);
}

void MainWindow::on_toFileButton_clicked(bool checked)
{
    if(checked){
        ofilename = QFileDialog::getOpenFileName(this,
tr("Choose output file"), QDir::homePath(), "Text
Files(*.txt);;All Files(*.*)").toStdString();
        ui->OutputFilename->setText(tr(ofilename.c_str()));
    }else ui->OutputFilename->setText(tr(""));
}

void MainWindow::on_randomButton_clicked()
{
    ui->InputLine->setEnabled(false);
}
```

```
void MainWindow::on_fromFileButton_clicked()
{
    ifilename = QFileDialog::getOpenFileName(this, tr("Choose
input file"), QDir::homePath(), tr("Text files(*.txt);;All
files(*.*)")).toStdString();
    ui->InputLine->setText(tr(ifilename.c_str()));
}

void MainWindow::on_fromKeyboardButton_clicked()
{
    message = QInputDialog::getText(this, tr("Enter "),
tr("")).toStdString();
    for(auto it=message.begin(); it<message.end(); ++it)
        if(isspace(*it))
            message.erase(it);
    encoded = "";
    ui->InputLine->setText(tr(message.c_str()));
}

void MainWindow::on_createButton_clicked()
{
    Options options;
    options.isEncode = ui->encodingButton->isChecked();
    options.isHuffman = ui->HuffmanButton->isChecked();
    options.isTable = ui->FreqTableCheck->isChecked();
    if(ui->randomButton->isChecked()){
        encoded = "";
        message = "";
    }
    if(!options.isEncode){
        if(encoded == "")
            std::swap(message, encoded);
        for(auto c : encoded)
            if(c != '1' && c != '0'){
                QMessageBox::information(this, tr("Incorrect
encoded message"), tr("Encoded string must contain only '1' and
'0' characters"));
                return;
            }
    }

    std::vector<std::pair<int, char>> table;
    std::set<char> alphabet;
    auto freq = ui->FreqTable;
    if(options.isTable)
```

```
        for(int i=0; i<freq->rowCount(); ++i){
            if(!freq->item(i, 0) || !freq->item(i, 1) || freq-
>item(i, 0)->text() == tr("") || freq->item(i, 1)->text() ==
tr(""))
                continue;
            if(freq->item(i, 0)->text() != "" && freq->item(i,
1)->text() != "" && alphabet.find(freq->item(i, 0)-
>text().toStdString()[0]) == alphabet.end()){
                table.push_back(std::make_pair(std::stoi(freq-
>item(i, 1)->text().toStdString()), freq->item(i,0)-
>text().toStdString()[0]));
                alphabet.insert(freq->item(i, 0)-
>text().toStdString()[0]);
            }
        }
    if(options.isTable && table.size() < 2){
        QMessageBox::information(this, QString("Incomplete
table"), QString("Table must contain at least 2 symbols"));
        return;
    }
    if(ui->fromFileButton->isChecked()){
        std::filebuf file;
        if(file.open(ifilename, std::ios::in)){
            std::istream fin(&file);
            std::getline(fin, message);
        }else{
            QMessageBox::information(this, QString("File doesn't
exist"), QString("File doesn't exist"));
            return;
        }
    }
    if(options.isTable && options.isEncode)
        for(auto c : message)
            if(alphabet.find(c) == alphabet.end()){
                QMessageBox::information(this,
QString("Incomplete table"), QString("Message contains symbol
that table doesn't contain"));
                return;
            }
    CodeTree*(*coding)(std::set<std::pair<int, char>>);
    if(options.isHuffman)
        coding = CodeTree::Huffman;
    else coding = CodeTree::ShannonFano;
    create_task(table, message, encoded, coding);
    std::string given, answer;
```

```
    if(options.isEncode){
        given = message;
        answer = encoded;
    }else{
        given = encoded;
        answer = message;
    }
    if(!ui->toFileButton->isChecked()){
        Dialog d(table, given, answer, options);
        d.show();
        d.exec();
    }
    else{
        std::filebuf file;
        if(file.open(ofilename, std::ios::out)){
            std::ostream fout(&file);
            std::string text;
            if(options.isEncode)
                text = "Кодирование ";
            else text = "Декодирование ";
            if(options.isHuffman)
                text += "Хаффмана";
            else text += "Шеннона-Фано";
            fout << text << std::endl;
            if(options.isTable){
                fout << "Дана таблица частот символов:" <<
std::endl;
                for(auto i : table)
                    fout << std::get<1>(i) << " " <<
std::get<0>(i) << std::endl;
            }

            if(options.isEncode)
                text = "Незакодированное сообщение:\n";
            else text = "Закодированное сообщение:\n";
            text += given;
            text += "\nAnswer: ";
            text += answer;
            fout << text;
            fout.flush();
            OpenDialog dialog(ofilename, this);
            dialog.show();
            dialog.exec();
        }else{
```

```
            QMessageBox::information(this, QString("File doesn't
exist"), QString("File doesn't exist"));
            return;
        }
    }
    if(options.isEncode)
        encoded = "";
    else message = "";
}
```

Файл openwindow.cpp:

```
#include "opendialog.h"
#include "ui_opendialog.h"

OpenDialog::OpenDialog(std::string filepath, QWidget *parent) :
    QDialog(parent),
    ui(new Ui::OpenDialog)
{
    ui->setupUi(this);
    this->filepath = filepath;
}

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

void OpenDialog::on_openFileButton_clicked()
{
    std::string command("gedit ");
    command += filepath;
    system(command.c_str());
}

void OpenDialog::on_openDirButton_clicked()
{
    std::string command("nautilus ");
    command += filepath;
    system(command.c_str());
}
```

Файл task.cpp:

```
#include <chrono>
#include <random>
#include <tuple>
```

```
#include "coding.h"

void create_task(std::vector<std::pair<int, char>> &freq,
std::string &message, std::string &encoded, CodeTree*(*coding)
(std::set<std::pair<int, char>>)){
    unsigned seed =
std::chrono::system_clock::now().time_since_epoch().count();
    std::minstd_rand0 generator (seed);
    if(message == "" && encoded == ""){
        int message_len = generator() % 10 + 10;
        if(freq.size() != 0){
            std::string alphabet;
            for(auto i : freq)
                alphabet.append(1, std::get<1>(i));
            for(int i=0; i<message_len; ++i)
                message.append(1, alphabet[generator() %
alphabet.length()]);
        }
        else for(int i=0; i<message_len; ++i)
            message.append(1, ('a'+(generator()%26)));
    }
    if(freq.size() == 0){
        std::set<char> alphabet;
        for(auto it : message)
            alphabet.insert(it);
        int count = 0;
        for(char a : alphabet){
            for(char c : message)
                if(a == c)
                    ++count;
            freq.push_back(std::make_pair(count, a));
            count = 0;
        }
    }
    std::set<std::pair<int, char>> freq_table(freq.begin(),
freq.end());
    CodeTree* code = coding(freq_table);
    if(encoded == "")
        encoded = code->encode(message);
    else
        message = code->decode(encoded);
    delete code;
}
```