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: Статическое кодирование и декодирование текстового сообщения методами Хаффмана и Фано-Шеннона – Демонстрация

Студентка гр. 7383 Чемова К. А.

Преподаватель Размочаева Н.В.

Санкт-Петербург

2018

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студентка Чемова К.А.

Группа 7383

Тема работы: Статическое кодирование и декодирование текстового сообщения методами Хаффмана и Фано-Шеннона – Демонстрация

Содержание пояснительной записки:

- Содержание
- Введение
- Теоретические сведения
- Решение задачи
- Примеры работы программы
- Заключение
- Приложение А. Исходный код программы

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 19.10.2018

Дата сдачи курсовой работы:

Дата защиты курсовой работы:

Студентка гр. 7383 Чемова К. А.

Преподаватель Размочаева Н.В.

АННОТАЦИЯ

В курсовой работе была реализована программа с графическим интерфейсом. Программа кодирует и декодирует заданный файл, а также выводит на экран деревья, соответствующие алгоритмам кодирования Фано-Шеннона и Хаффмана.

SUMMARY

In the curse work was implemented a program with graphic interface. The program encodes and decodes the specified file as well displays a tree corresponding to the encoding algorithms Fano-Shannon and Huffman.

СОДЕРЖАНИЕ

Задание на курсовую работу 2
Аннотация 3
Summary 3
Введение 5
Теоретические сведения 6
Решение задачи 7
Пример работы программы 8
Заключение 10
Список использованных источников 11
ПРИЛОЖЕНИЕ А Исходный код программы 12

ВВЕДЕНИЕ

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

Задачей является написание программы, которая кодирует и декодирует текст файла, а также выводит на экран деревья, построенные по алгоритмам кодирования Фано-Шеннона и Хаффмана.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1. Алгоритм Фано-Шеннона

Кодирование Фано-Шеннона представляет собой алгоритм префиксного кодирования с кодом переменной длины.

Этапы выполнения кодирования:

1.1. Сортировка символов данного текста по убыванию частоты встречаемости;

1.2. Все символы делят на две части, их суммарные частоты должны быть примерно равны;

1.3. Первой части символов присваивается код 0, правой части – 1;

1.4. Для каждой части выполняют действия с пп. 1.2, дописывая в конец код соответствующие нули и единицы.

2. Алгоритм Хаффмана

Алгоритм Хаффмана является жадным алгоритмом оптимального префиксного кодирования с минимальной избыточностью.

Этапы выполнения кодирования:

2.1. Символы входного алфавита образуют список свободных узлов. Каждый лист имеет вес, который может быть равен либо вероятности, либо количеству вхождений символа в сжимаемое сообщение.

2.2. Выбираются два свободных узла дерева с наименьшими весами.

2.3. Создается их родитель с весом, равным их суммарному весу.

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

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

2.6. 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.

РЕШЕНИЕ ЗАДАЧИ

Для кодирования Фано-Шеннона были написаны `struct Elem`, `struct Node`. Во время работы программы строится дерево, которое обходится рекурсивно, сначала левое поддерево, потом правое. К коду всех элементов левого поддерева добавляется 0, а к коду символов правого поддерева – 1.

Для кодирования Хаффмана были написаны `struct Huff`, `class QueueEl`, `class Nod`. Поскольку дерево строится с конца, то создается очередь для сортировки элементов. Отличие кодирования Фано-Шеннона для пошаговой визуализации кодирования Хаффмана приходится хранить строку, содержащую в себе символы, на данном этапе кодирования.

Функция `void MainWindow::ChooseANDencode()` кодирует выбранный файл (при нажатии на эту кнопку открывается новое окно для выбора файла формата .txt). Функция `void MainWindow::on_close()` закрывает приложение. Функция `void MainWindow::MakeDecode()` декодирует предварительно закодированный текст. Функция `void MainWindow::PrintTree()` печатает деревья в выделенных рамках.

При выборе пустого файла, при попытке декодировать сообщение перед кодированием выводится сообщение об соответствующей ошибке.

ПРИМЕР РАБОТЫ ПРОГРАММЫ

Программа была написана в QtCreator 5.7.0 MinGW 32bit в операционной системе Windows 10. В других системах тестирование не проводилось.

На рис. 1 представлено удачное выполнение кодирования и декодирования.

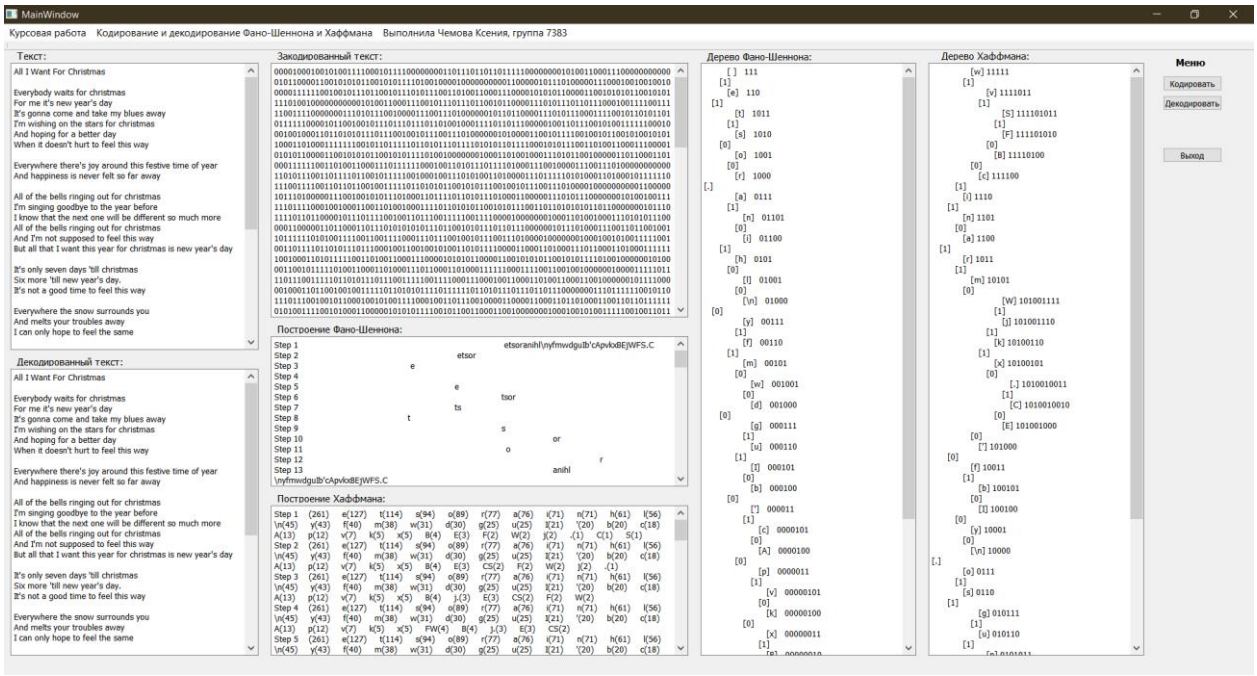

Рисунок 1 – Правильная работа программы

На рис. 2 показано окно выбора файла, текст которого необходимо закодировать

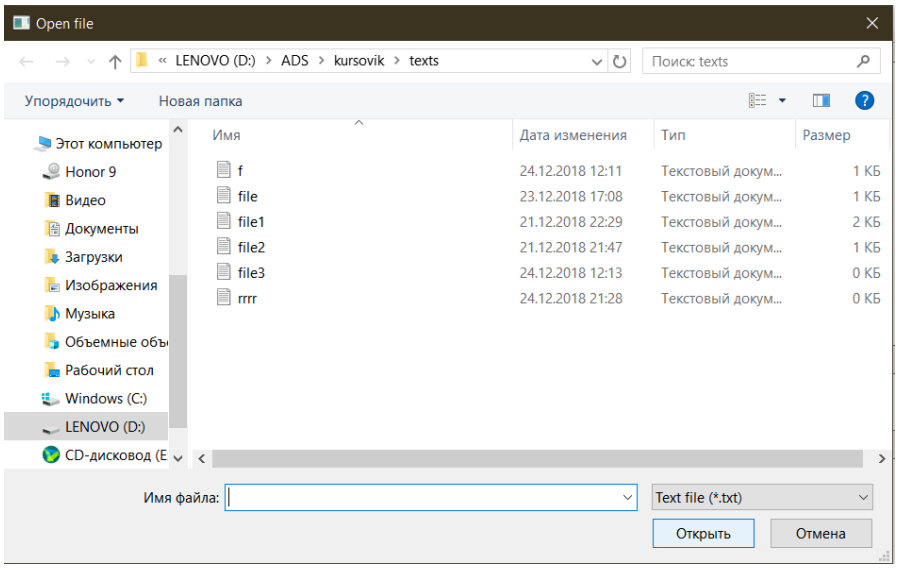

Рисунок 2 – Окно выбора файла

На рис. 3 представлены сообщения об ошибках, которые выводятся в отдельных окнах.

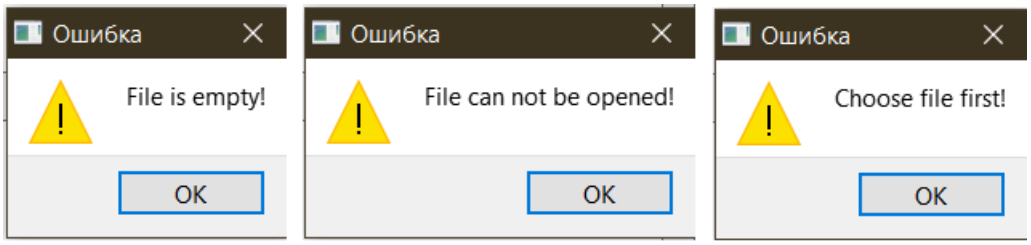

Рисунок 3 – Сообщения об ошибках

ЗАКЛЮЧЕНИЕ

В ходе работы была реализована программа на языке программирования C++ с использованием программы Qt Creator для кодирования и декодирования текста с использованием алгоритмов Хаффмана и Фано-Шеннона. Также было реализовано пошаговая реализация обоих алгоритмов.

Для взаимодействия с программой был реализован графический интерфейс, упрощающий работу пользователя.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

http://doc.qt.io/

Статический алгоритм Хаффмана

Алгоритм Шеннона-Фано

Динамический алгоритм Хаффмана

ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ

curse.pro:

```
#-------------------------------------------------
#
# Project created by QtCreator 2018-10-29T19:58:45
#
#-------------------------------------------------

QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = kursovik
TEMPLATE = app


SOURCES += main.cpp\
        mainwindow.cpp

HEADERS  += mainwindow.h

FORMS    += mainwindow.ui

RESOURCES +=
```

mainwindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <iostream>
#include <cstdlib>
#include <vector>
#include <list>
#include <QFileDialog>
#include <QMessageBox>
#include <QDebug>
#include <QMainWindow>

using namespace std;

struct Elem{
    QChar c;
    QString value;
    Elem* parent;
```

```
    Elem* left;
    Elem* right;
};

struct Node{
    int num;
    Elem* ptr;
    QChar value;
};

struct Huff{
    int num;
    Elem* ptr;
    QString val;
    QString code;
};

class QueueEl{
    public:
    QChar symbol;
    int count;

    QueueEl(QChar s, int n) {
        symbol = s;
        count = n;
    }

    bool operator<(QueueEl t) const {
        if(count != t.count)
            return count < t.count;
        return symbol < t.symbol;
    }
};

class Nod{
    public:
    int count;
    QChar value;
    Nod* left, *right;

    Nod(Nod* l, Nod* r) {
        left = l;
        right = r;
        value = NULL;
        count = l->count + r->count;
    }

    Nod() {
        left = NULL;
        right = NULL;
        count = 0;
```

```
        value = NULL;
    }
    bool operator<(Nod* t) const {
            return count < t->count;
        }
    };

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    void MakeList(Elem* t);
    void MakeHuffList(Nod* t);

private slots:
    void ChooseANDencode();

    void on_close();

    void MakeDecode();

    void PrintTree();

    void PrintBinary();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

main.cpp:

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

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

mainwindow.cpp:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QString>
#include <list>
#include <QMessageBox>
using namespace std;

Node* arr = NULL;
Elem* head = NULL;
Nod* he = NULL;
int index = 0;
QString binary = "";

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow){
    ui->setupUi(this);
    connect(ui-
>ChooseEncode,SIGNAL(clicked()),this,SLOT(ChooseANDencode()));
    connect(ui->MakeDecode,SIGNAL(clicked()),this,SLOT(MakeDecode()));
    //connect(ui->PrintTree,SIGNAL(clicked()),this,SLOT(PrintTree()));
    connect(ui->on_close,SIGNAL(clicked()),this,SLOT(on_close()));
}

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

bool compare_node(const Nod* first,const Nod* second){
    return first->count < second->count;
}

int cmp(const void* a,const void* b){
    return (*(Node*)b).num-(*(Node*)a).num;
}

int comp(const void* a,const void* b){
    if((*(Huff*)b).num == (*(Huff*)a).num){
        if ((*(Huff*)b).val.size() == (*(Huff*)a).val.size()){
            for (int i=0;i<(*(Huff*)b).val.size();i++){
                if ((*(Huff*)b).val[i] > (*(Huff*)a).val[0])
                    return -1;
                else if ((*(Huff*)b).val[i] < (*(Huff*)a).val[0])
                        return 1;
                     else continue;
            }
            return 0;
        }
        else return (*(Huff*)b).val.size() - (*(Huff*)a).val.size();
```

```
    }
    else return (*(Huff*)b).num-(*(Huff*)a).num;
}

void iteration(Elem** root){
    if (*root){
        iteration(&(*root)->left);
        iteration(&(*root)->right);
                free(*root);
    }
}
 void iteration1(Nod** root) {
     if (*root){
         iteration1(&(*root)->left);
         iteration1(&(*root)->right);
                 delete[] *root;
     }
 }

Elem* buildTree(QChar el){
    Elem* head = new Elem;
    head->c = el;
    head->parent = NULL;
    head->value = "";
    head->left = NULL;
    head->right = NULL;
    return head;
}

void FillTheTree(Elem* head, QString str,int &count){
    int deep = 0;

    for(unsigned int i = 1; i < str.size(); i++){
        if(str[i] == '('){
        deep++;
        }
        if(str[i] == ')'){
        deep--;
        }
        if(str[i] != '(' && str[i] != ')'){
            Elem* tree = head;
            for(int j = 1; j < deep; j++){
               if(tree->right == NULL){
                   tree = tree->left;
               } else {
                   tree = tree->right;
                   }
         }
        Elem* newElem = new Elem;
        if(tree->left == NULL){
            tree->left = newElem;
```

```
        }else {
            tree->right = newElem;
            }
        newElem->value = "";
        newElem->c = str[i];
        if (newElem->c == '1' || newElem->c == '0')
            count--;
        count++;
        newElem->left = NULL;
        newElem->right = NULL;
        }
    }
}

void Huffman(Elem* t,Huff* h,int i,QString &str,int ind,int cycle){
    char st[10];
    str = str + "Step " + itoa(ind++,st,9) + "   ";
    for (int m=0;m<cycle;m++){
        for (int j=0;j<h[m].val.size();j++){
            if ((h[m].val)[j] == '\n')
                str = str + "\\n";
            else str += (h[m].val)[j];
        }
        str = str + "(" + itoa(h[m].num,st,10) + ")     ";
    }
    str+="\n";
    if (i == 0){
        h[i].code = h[i].code + ".";
        return;
    }
    h[i+1].num = h[i].num;
    h[i+1].code.clear();
    h[i+1].code = h[i+1].code + "1";
    h[i+1].val.clear();
    h[i+1].val += h[i].val;
    h[i].num = h[i-1].num;
    h[i].code.clear();
    h[i].code = h[i].code + "0";
    h[i].val.clear();
    h[i].val = h[i-1].val;
    h[i-1].num = h[i].num + h[i+1].num;
    h[i-1].val += h[i+1].val;
    qsort(h,i,sizeof(Huff),comp);
    Huffman(t,h,i-1,str,ind,cycle-1);
}

void SearchTree(Elem** t,Node* el, QString &branch, QString
&fullBranch, int start, int end,QString &steps,QChar uzel,int &ind,int
tab){
    char st[10];
    steps = steps + "Step " + itoa(ind++,st,10) + "   ";
```

```
    for (int i=0;i<tab;i++)
        steps = steps + "   ";
    for (int i = start;i<=end;i++){
        if (el[i].value == '\n')
            steps = steps + "\\n";
        else steps = steps + el[i].value;
    }
    steps = steps + "\n";
    if (*t == NULL){
        *t = new Elem;
        (*t)->parent = NULL;
        (*t)->left = NULL;
        (*t)->right = NULL;
        (*t)->value = "";
        (*t)->c = uzel;
    }
    double dS=0;
    int i,S=0;
    QString cBranch="";
    cBranch=fullBranch+branch;
    if (start==end){
        (*t)->value += cBranch;
        (*t)->c = el[start].value;
        el[start].ptr = *t;
        return;
    }
    for (i=start;i<=end;i++)
        dS+=el[i].num;
    dS/=2.;
    i=start+1;
    S+=el[start].num;
    while (fabs(dS-(S+el[i].num))<fabs(dS-S) && (i<end)){
        S+=el[i].num;
        i++;
    }
    QChar o = '1';
    QChar z = '0';
    QString zero="0";
    QString one="1";
    SearchTree(&(*t)->left,el,one,cBranch,start,i-1,steps,o,ind,tab-
6);
    SearchTree(&(*t)->right,el,zero,cBranch,i,end,steps,z,ind,tab+6);
}

void Encode(Node* arr,int index,QString copy,QString &binary){
    for (int i=0;copy[i]!='\0';i++){
        for (int j=0;j<index;j++){
            if (copy[i] == arr[j].value){
                binary += arr[j].ptr->value;
            }
        }
```

```
    }
}

void print(Elem *t,QString &out,int count){
   if (t != NULL)   {
       print(t->left,out,count+1);
       for (int i = 0;i < count;i++)
           out = out + "    ";
       if (t->c == '\n')
           out = out + "[\\n]" + "    " + t->value + "\n";
       else out = out + "[" + t->c + "]    " + t->value + "\n";
       print(t->right,out,count+1);
   }
}

void print1(Nod *t,QString &out,int count,char uzel,QString code){
   if (t != NULL)   {
       if (t->left != NULL)
        t->value = uzel;
       uzel = '1';
       print1(t->left,out,count+1,uzel,code+"1");
       for (int i = 0;i < count;i++)
           out = out + "    ";
       if (t->value == '0' || t->value == '1'){
           out = out + "[" + t->value + "]" + "\n";
       }
       else if (t->value == '\n')
                out = out + "[\\n] " + code + "\n";
            else out = out + "[" + t->value + "] " + code + "\n";
       uzel = '0';
       print1(t->right,out,count+1,uzel,code+"0");
   }
}

void MainWindow::MakeHuffList(Nod* t){
    if (t == NULL){
        QMessageBox::warning(this, "Ошибка", "Tree is empty!");
    }
    else{
        QString tr;
        char uzel = '.';
        QString cod = "";
        print1(t,tr,0,uzel,cod);
        ui->HuffTree->setText(QString(tr.data()));
    }
}

void MainWindow::MakeList(Elem* t){
    if (t == NULL){
        QMessageBox::warning(this, "Ошибка", "Tree is empty!");
    }
```

```
    else{
        QString tr;
        print(t,tr,0);
        ui->ResultEdit->setText(QString(tr.data()));
    }
}

void Decode(Node* arr,QString &answer,QString binary,int index){
    int len = 0;
    QString ptr = "";
    while (len <= binary.length()){
        ptr += binary[len++];
        for (int i = 0; i < index;i++){
            if(ptr == arr[i].ptr->value) {
                answer += arr[i].value;
                ptr.clear();
            }
        }
    }
}

void MainWindow::ChooseANDencode(){
    int k;
    QString str = "",copy = "",all = "";
    QString filename = QFileDialog::getOpenFileName(this,tr("Open
file"),"D://ADS//kursovik//texts","Text file (*.txt)");
    QFile file(filename.toStdString().data());
    if (!file.open(QIODevice::ReadOnly)){
        QMessageBox::warning(this, "Ошибка", "File can not be
opened!");
    }
    else{
        free(arr);
        iteration(&head);
        head = NULL;
        iteration1(&he);
        he = NULL;
        binary.clear();
        ui->ResultEdit->clear();
        ui->BinaryEdit->clear();
        ui->CodesEdit->clear();
        ui->DecodeEdit->clear();
        ui->PreEdit->clear();
        index = 0;
        arr = (Node*) malloc(sizeof(Node));
        QTextStream in(&file);
        all = in.readLine();
        int c=0;
        str += all + "\n";
        while (!in.atEnd()){
            all = in.readLine();
```

```
            str += all + "\n";
        }
        if (str.size()==1 && (str[0]=='\n' || str[0] =='\0')){
            QMessageBox::warning(this, "Ошибка", "File is empty!");
            return;
        }

        ui->PreEdit->setText(str);
        copy += str;
        for (int j=0;str[j] != '\0';j++){
            k = 0;
            for (int i=j+1;str[i] != '\0';i++){
                if (str[j] == str[i]){
                    while (str[j] == str[i]){
                        str.remove(i,1);
                        k++;
                    }
                }
            }
            k++;
            index++;
            arr = (Node*) realloc (arr,index*sizeof(Node));
            if (arr != NULL){
                arr[index-1].num = k;
                arr[index-1].value = str[j];
                arr[index-1].ptr = NULL;
            }
            else {
                free(arr);
                QMessageBox::warning(this, "Ошибка", "Error with
allocation!");
                break;
            }
        }
        if (c != index){
            //ui->BinaryStroka->setText("Trees are not equal,deleting
your tree...");
            iteration(&head);
            head = NULL;
        }
        file.close();
        qsort(arr,index,sizeof(Node),cmp);
        Huff* mass = NULL;
        mass = new Huff[2*index-1];
        for (int i = 0;i<index;i++){
            mass[i].num = arr[i].num;
            mass[i].val = arr[i].value;
        }
        QString hu;
        int ind = 1;
        list<QueueEl> queue;
```

```
        for (int i=0;i<index;i++) {
            queue.push_back(QueueEl(mass[i].val[0],mass[i].num));
        }
        queue.sort();
        list<QueueEl>::iterator q_it;
        list<Nod*>listNode;
        for (q_it = queue.begin(); q_it != queue.end(); q_it++) {
            Nod* p = new Nod;
            p->count = q_it->count;
            p->value = q_it->symbol;
            listNode.push_back(p);
        }
        while (listNode.size() > 1) {
            listNode.sort(compare_node);
            Nod* L = listNode.front();
            listNode.pop_front();
            Nod* R = listNode.front();
            listNode.pop_front();
            Nod* parent = new Nod(L, R);
            listNode.push_back(parent);
        }
        he = listNode.front();
        qsort(mass,index,sizeof(Huff),comp);
        Huffman(head,mass,index-1,hu,ind,index);
        QChar sym = '.';
        ui->HuffmanStep->setText(hu);
        delete[] mass;
        QString a = "",b = "",symbols = "";
        ind = 1;
        int tab = 25;
        SearchTree(&head,arr,a,b,0,index-1,symbols,sym,ind,tab);
        ui->CodesEdit->setText(symbols);
        Encode(arr,index,copy,binary);
        if (binary.isEmpty())
            QMessageBox::warning(this, "Ошибка", "Binary is empty!");
        else ui->BinaryEdit->setText(binary);
        PrintTree();
    }
}

void MainWindow::on_close(){
    this->close();
}

void MainWindow::MakeDecode(){
    if (index == 0){
        QMessageBox::warning(this, "Ошибка", "Choose file first!");
        return;
    }
    QString answer = "";
    Decode(arr,answer,binary,index);
```

```
    ui->DecodeEdit->setText(answer);
}

void MainWindow::PrintTree(){
    MakeList(head);
    MakeHuffList(he);
}
```

mainwindow.ui:

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="enabled">
   <bool>true</bool>
  </property>
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>1910</width>
    <height>1006</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <property name="minimumSize">
    <size>
     <width>1533</width>
     <height>941</height>
    </size>
   </property>
   <property name="autoFillBackground">
    <bool>false</bool>
   </property>
   <widget class="QPushButton" name="ChooseEncode">
    <property name="geometry">
     <rect>
      <x>1780</x>
      <y>40</y>
      <width>91</width>
      <height>23</height>
```

```
  </rect>
 </property>
 <property name="text">
  <string>Кодировать</string>
 </property>
</widget>
<widget class="QTextEdit" name="ResultEdit">
 <property name="geometry">
  <rect>
   <x>1070</x>
   <y>20</y>
   <width>331</width>
   <height>911</height>
  </rect>
 </property>
 <property name="readOnly">
  <bool>true</bool>
 </property>
</widget>
<widget class="QTextEdit" name="DecodeEdit">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>490</y>
   <width>381</width>
   <height>441</height>
  </rect>
 </property>
 <property name="readOnly">
  <bool>true</bool>
 </property>
</widget>
<widget class="QPushButton" name="on_close">
 <property name="geometry">
  <rect>
   <x>1780</x>
   <y>150</y>
   <width>91</width>
   <height>23</height>
  </rect>
 </property>
 <property name="text">
  <string>Выход</string>
 </property>
```

```
</widget>
<widget class="QTextEdit" name="PreEdit">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>20</y>
   <width>381</width>
   <height>441</height>
  </rect>
 </property>
 <property name="readOnly">
  <bool>true</bool>
 </property>
</widget>
<widget class="QPushButton" name="MakeDecode">
 <property name="geometry">
  <rect>
   <x>1780</x>
   <y>70</y>
   <width>91</width>
   <height>23</height>
  </rect>
 </property>
 <property name="text">
  <string>Декодировать</string>
 </property>
</widget>
<widget class="QTextEdit" name="HuffTree">
 <property name="geometry">
  <rect>
   <x>1420</x>
   <y>20</y>
   <width>331</width>
   <height>911</height>
  </rect>
 </property>
 <property name="readOnly">
  <bool>true</bool>
 </property>
</widget>
<widget class="QLabel" name="label">
 <property name="geometry">
  <rect>
   <x>20</x>
```

```
      <y>0</y>
      <width>55</width>
      <height>16</height>
     </rect>
    </property>
    <property name="text">
     <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span
style=&quot; font-
size:9pt;&quot;&gt;Текст:&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html
&gt;</string>
    </property>
   </widget>
   <widget class="QLabel" name="label_4">
    <property name="geometry">
     <rect>
      <x>1080</x>
      <y>0</y>
      <width>171</width>
      <height>20</height>
     </rect>
    </property>
    <property name="text">
     <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span
style=&quot; font-size:9pt;&quot;&gt;Дерево Фано-
Шеннона:&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QLabel" name="label_5">
    <property name="geometry">
     <rect>
      <x>1440</x>
      <y>0</y>
      <width>141</width>
      <height>16</height>
     </rect>
    </property>
    <property name="text">
     <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span
style=&quot; font-size:9pt;&quot;&gt;Дерево
Хаффмана:&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QLabel" name="label_2">
    <property name="geometry">
```

```
     <rect>
      <x>20</x>
      <y>470</y>
      <width>171</width>
      <height>16</height>
     </rect>
    </property>
    <property name="text">
     <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span
style=&quot; font-size:9pt;&quot;&gt;Декодированный
текст:&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QLabel" name="label_3">
    <property name="geometry">
     <rect>
      <x>1800</x>
      <y>10</y>
      <width>55</width>
      <height>16</height>
     </rect>
    </property>
    <property name="text">
     <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span
style=&quot; font-size:9pt; font-
weight:600;&quot;&gt;Меню&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html
&gt;</string>
    </property>
   </widget>
   <widget class="QTextEdit" name="HuffmanStep">
    <property name="geometry">
     <rect>
      <x>410</x>
      <y>700</y>
      <width>641</width>
      <height>231</height>
     </rect>
    </property>
    <property name="readOnly">
     <bool>true</bool>
    </property>
   </widget>
   <widget class="QTextEdit" name="CodesEdit">
    <property name="geometry">
```

```
      <rect>
       <x>410</x>
       <y>440</y>
       <width>641</width>
       <height>231</height>
      </rect>
     </property>
     <property name="readOnly">
      <bool>true</bool>
     </property>
    </widget>
    <widget class="QTextEdit" name="BinaryEdit">
     <property name="geometry">
      <rect>
       <x>410</x>
       <y>20</y>
       <width>641</width>
       <height>391</height>
      </rect>
     </property>
     <property name="readOnly">
      <bool>true</bool>
     </property>
    </widget>
    <widget class="QLabel" name="label_6">
     <property name="geometry">
      <rect>
       <x>420</x>
       <y>0</y>
       <width>171</width>
       <height>16</height>
      </rect>
     </property>
     <property name="text">
      <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span
style=&quot; font-size:9pt;&quot;&gt;Закодированный
текст:&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
     </property>
    </widget>
    <widget class="QLabel" name="label_8">
     <property name="geometry">
      <rect>
       <x>420</x>
       <y>420</y>
```

```
      <width>191</width>
      <height>16</height>
     </rect>
    </property>
    <property name="text">
     <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span
style=&quot; font-size:9pt;&quot;&gt;Построение Фано-
Шеннона:&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QLabel" name="label_9">
    <property name="geometry">
     <rect>
      <x>420</x>
      <y>680</y>
      <width>171</width>
      <height>16</height>
     </rect>
    </property>
    <property name="text">
     <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span
style=&quot; font-size:9pt;&quot;&gt;Построение
Хаффмана:&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
  </widget>
  <widget class="QMenuBar" name="menuBar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>1910</width>
     <height>26</height>
    </rect>
   </property>
   <widget class="QMenu" name="menu">
    <property name="title">
     <string>Курсовая работа</string>
    </property>
   </widget>
   <widget class="QMenu" name="menu_2">
    <property name="title">
     <string>Кодирование и декодирование Фано-Шеннона и
Хаффмана</string>
```

```
    </property>
   </widget>
   <widget class="QMenu" name="menu_7383">
    <property name="title">
     <string>Выполнила Чемова Ксения, группа 7383 </string>
    </property>
   </widget>
   <addaction name="menu"/>
   <addaction name="menu_2"/>
   <addaction name="menu_7383"/>
  </widget>
  <widget class="QToolBar" name="mainToolBar">
   <attribute name="toolBarArea">
    <enum>TopToolBarArea</enum>
   </attribute>
   <attribute name="toolBarBreak">
    <bool>false</bool>
   </attribute>
  </widget>
  <widget class="QStatusBar" name="statusBar"/>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources/>
 <connections/>
</ui>
```