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

Студент гр. 8381 _____ Муковский Д.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Муковский Д.В.

Группа 8381

Тема работы: Динамическое кодирование и декодирование по Хаффману

Исходные данные: необходимо выполнить визуализацию двух алгоритмов: динамического кодирования и декодирования по Хаффману. Должны быть реализована возможность пошагово выполнения алгоритмов.

Содержание пояснительной записки:
«Содержание», «Введение», «Задание», «Описание программы», «Теоретические положения», «Демонстрация», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:
Не менее 20 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент _____ Муковский Д.В.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

В ходе выполнения курсовой работы была разработана программа с GUI, позволяющая визуализировать динамические алгоритмы кодирования и декодирования по Хаффману с возможностью пошагового выполнения: доступны шаги как вперед, так и назад. Созданная программа обладает следующей функциональностью: пользователь может алгоритм из двух предложенных, файл, содержащий входную строку для выполнения алгоритма, а также режим его выполнения.

SUMMARY

During the course of the course work, a program was developed with a GUI that allows you to visualize dynamic Huffman coding and decoding algorithms with the possibility of step-by-step execution: steps are available both forward and backward. The created program has the following functionality: the user can choose from two proposed algorithms, a file containing the input string to execute the algorithm, as well as the mode of its execution.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью работы является реализация и демонстрация алгоритмов динамического кодирования и декодирования по Хаффману. Программа должна обладать возможностью пошагового выполнения с графической визуализацией каждого шага, функцией считывания из файла входной строки и записью в файл результата. Разработка программы велась на базе операционной системы Windows в интегрированной среде разработки *QtCreator*. Для создания графической оболочки были использованы стандартные библиотеки данного фреймворка. Бинарное дерево, которое визуализировалось, было реализовано в классе *BinTree*. Алгоритмы по кодированию и декодированию были написаны в функциональном стиле. Для вывода необходимой пользователю информации были реализованы классы окон *InfoAboutAlgorithm* и *Instruction*.

1. ЗАДАНИЕ

Необходимо продемонстрировать алгоритмы кодирования и декодирования.

Демонстрация должна:

1. Быть подробной и понятной (в том числе сопровождаться пояснениями);
2. Иметь возможность выполнения программы в пошаговом режиме;
3. Иметь возможность быть использованной в обучающих целях.

2. ОПИСАНИЕ ПРОГРАММЫ

2.1. Описание интерфейса пользователя

Интерфейс программы разделен на четыре части: панель ввода данных; кнопки отвечающие за выполнение программы; сцена, на которой изображается дерево и поле вывода результата. Основные виджеты и их назначение представлены в табл. 1.

Таблица 1 – Основные виджеты интерфейса

Класс объекта	Название виджета	Назначение
`QMenu`	`menuInfo`	Кнопка меню информации
`QComboBox`	`comboBox`	Поле со списком выбора алгоритма
`QLineEdit`	`inputStr`	Поле ввода входной строки
`QPushButton`	`startCodingButton`	Запуск алгоритма в моментальном режиме
`QPushButton`	`stepByStepStart`	Запуск алгоритма в пошаговом режиме
`QPushButton`	`readBit`	Считывание бита в пошаговом режиме декодирования
`QPushButton`	`addElement`	Добавление нового узла в бинарное дерево в пошаговом режиме
`QPushButton`	`upgradeTree`	Упорядочивание бинарного дерева в пошаговом режиме
`QPushButton`	`saveButton`	Кнопка сохранения результата в файл
`QPushButton`	`prevStep`	Возврат к предыдущему шагу
`QPushButton`	`readFileButton`	Кнопка считывания строки из файла
`QGraphicsView`	`graphicView`	Поле, где дерево визуализируется
`QTextBrowser`	`answer`	Поле вывода результата

2.2. Описание основных методов для бинарного дерева

Для реализации бинарного дерева была написана структура *Node*, описывающая один узел дерева и класс *BinTree*, хранящий корень дерева. Основные методы класса *BinTree* представлены в табл. 2.

Таблица 2 – Основные методы класса *BinTree*

Метод	Назначение
`bool needUpdateTree();`	Возвращает true или false в зависимости от того, нужно ли упорядочить дерево.
`void getZeroNode(Node* tree, Node*& node) const;`	Находит нулевой элемент
`bool findNodeSymb(Node* tree, Node*& nodeSymb, char symb,bool& flag) const;`	Находит элемент по символу, который уже есть в дереве, иначе возвращает false
`int getMaxTreeDepth(Node* node) const`	Возвращает максимальную глубину
`Node* getRoot() const;`	Возвращает корень дерева
`void freeMem(Node* node = nullptr);`	Метод высвобождение памяти для деструктора
`void setOrdinaryNodeColor(Node* tree);`	Метод установки всех флагов окраски в положение false
`BinTree* copyTree(BinTree* tree);`	Метод копирования дерева
`void swapNodesForOrdering();`	Метод, меняющий два узла местами для упорядочивания

2.3. Описание реализации алгоритмов кодирования и декодирования

Алгоритмы кодирования и декодирования были написаны в функциональном стиле. Основные функции, реализующие алгоритм кодирования представлены в табл.3.

Таблица 3 – Функции для реализации алгоритма кодирования

Функция	Назначение
`int* codeOfNode(Node *node, int *n);`	Проходит от корня к нулевому элементу и записывает код
`void reverseCode(int* code, int codeSize);`	Реверсирует полученный код
`void addCodeToOut(std::string& outp,std::string& resultCode, int codeSize, int* symbCode, char byte, bool flag)`	Записывает код элемента в выходную строку
`void addAsciiToOut(std::string& outp,std::string& resultCode, int byte)`	В случае, если элемент новый записывает *ASKII* код символа
`Node* addSymbol(char symbol, Node** zeroNode);`	Добавляет *escape*-символ и новый узел в дерево
`void recalculationNodeValue(Node*currNode);`	Пересчитывает вес узлов для всех предков добавленного узла.
`void encode(char* input, std::string& output, int inputSize, BinTree* tree,std::string& resultCode);`	Полностью выполняет алгоритм кодирования
`void addNewNode(BinTree* tree,Node*& zeroNode,char symbol,std::string& output,std::string& resultCode)`	Добавляет следующий символ в строку, используется для пошагового режима

Функции для реализации алгоритма декодирования представлены в табл. 4.

Таблица 4 – Функции для реализации алгоритма кодирования

Функция	Назначение
`bool decode(char* input, std::string& output, int inputSize, BinTree* tree,std::string& resultCode)`	Полностью выполняет алгоритм декодирования

`bool readCurNode(char* input,int& currentInputLen,int inputLen,std::string& output,std::string& resultCode,Node*& curNode,BinTree* tree, int& diffCurrentInLen)`	Считывает символ и записывает его в выходную строку
`void readOneBit(Node*& curNode,std::string& output,char* input,int& currentInputLen)`	Считывает 1 бит и переходит к следующему узлу
`char readByte(char* input,int& curIndex, int inputSize,std::string& output)`	Считывает 1 байт и записывает в выходную строку

2.4. Описание методов демонстрации

Все методы визуализации располагаются в классе *mainwindow*. Ниже перечислены основные из них представлены в табл. 5.

Таблица 5 – методы для визуализации дерева

Метод	Назначение
`void MainWindow::updateScene();`	Обновляет текущую сцену
`DrawNode(Node * n,int maxdepth,int depth,int x,int y);`	Рекурсивно рисует дерево
`void MainWindow::setOrdinaryMode();`	Возвращает программу в стартовый режим
`void MainWindow::setStepEncodeMode()`	Устанавливает программу в пошаговый режим кодирования
`void MainWindow::setUpgrTreeMode();`	Устанавливает программу в пошаговый режим упорядочивания дерева
`void MainWindow::setReadBitMode();`	Устанавливает программу в пошаговый режим считывания бита

2.5. Описание реализации пошагового режима

Пошаговый режим был реализован следующим образом: была написана функция *dataPhotography*()(см. Приложение К), которая копировала дерево,

текущий совершаемый шаг вперед и строку вывода. Она вызывалась каждый раз, когда пользователь совершал шаг вперед. Но чтобы избежать засорение памяти от большого количества деревьев была написана функция *freeSteps*(), которая высвобождала память всех шагов, кроме пяти последних. Таким образов в любой момент выполнения программы в ней хранилось максимум 5 копий дерева. По кнопке предыдущий шаг алгоритм откатывал состояние всех текущих данных и состояний на предыдущие. В случае, если пользователь хотел вернуть программу на пять шагов назад программа выводила предупреждающее сообщение.

3. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ РАБОТЫ

3.1 Адаптивный алгоритм Хаффмана

Адаптивный алгоритм Хаффмана является модификацией обычного алгоритма Хаффмана сжатия сообщений. Он позволяет не передавать таблицу кодов и ограничиться одним проходом по сообщению, как при кодировании, так и при декодировании. Суть адаптивного алгоритма состоит в том, что при каждом сопоставлении символу кода изменяется внутренний ход вычислений так, что в следующий раз этому же символу может быть сопоставлен другой код, т.е. происходит адаптация алгоритма к поступающим для кодирования символам.

В адаптивном алгоритме сжатия Хаффмана используется упорядоченное бинарное дерево. Бинарное дерево называется упорядоченным, если его узлы могут быть перечислены в порядке не убывания веса. Перечисление узлов происходит по ярусам снизу-вверх и слева-направо в каждом ярусе. Узлы, имеющие общего родителя, находятся рядом на одном ярусе.

3.2. Правила кодирования

1. Элементы входного сообщения считываются побайтно.

2. Если входной символ присутствует в дереве, в выходной поток записывается код, соответствующий последовательности нулей и единиц, которыми помечены ветки дерева, при проходе от корня дерева к данному листу. Вес данного листа увеличивается на 1. Веса узлов-предков корректируются. Если дерево становится неупорядоченным - упорядочивается.

3. Если очередной символ, считанный из входного сообщения при сжатии, отсутствует в дереве, в выходной поток записывается набор нулей и единиц, которыми помечены ветки бинарного дерева при движении от корня к *escape*-символу, а затем 8 бит *ASCII*-кода нового символа. В дерево вместо *escape*-символа добавляется ветка: родитель, два потомка. Левый потомок становится *escape*-

символом, правый - новым добавленным в дерево символом. Веса узлов-предков корректируются, а дерево при необходимости упорядочивается.

3.3 Правила декодирования

1. Элементы входного сообщения считываются побитно.

2. Каждый раз при считывании 0 или 1 происходит перемещение от корня вниз по соответствующей ветке бинарного дерева Хаффмана, до тех пор, пока не будет достигнут какой-либо лист дерева.

3. Если достигнут лист, соответствующий символу, в выходное сообщение записывается *ASCII*-код данного символа. Вес листа увеличивается на 1, веса узлов-предков корректируются, дерево при необходимости упорядочивается.

4. Если же достигнут *escape*-символ, из входного сообщения считываются 8 следующих бит, соответствующих *ASCII*-коду нового символа. В выходное сообщение записывается *ASCII*-код данного символа. В дерево добавляется новый символ, веса узлов-предков корректируются, затем при необходимости производится его упорядочивание.

4. ДЕМОНСТРАЦИЯ

4.1. Вид программы

Программа запускается в полноэкранном режиме и представляет собой окно с графическим интерфейсом. Вид программы после запуска представлен на рис. 2.

Рисунок 2 – Вид программы после запуска

В данном окне пользователь может выбрать алгоритм, считать входную строку из файла и запустить программу в двух режимах исполнения.

4.2. Демонстрация алгоритма кодирования

На рис.3 представлен вид программы после моментального выполнения алгоритма динамического кодирования Хаффмана.

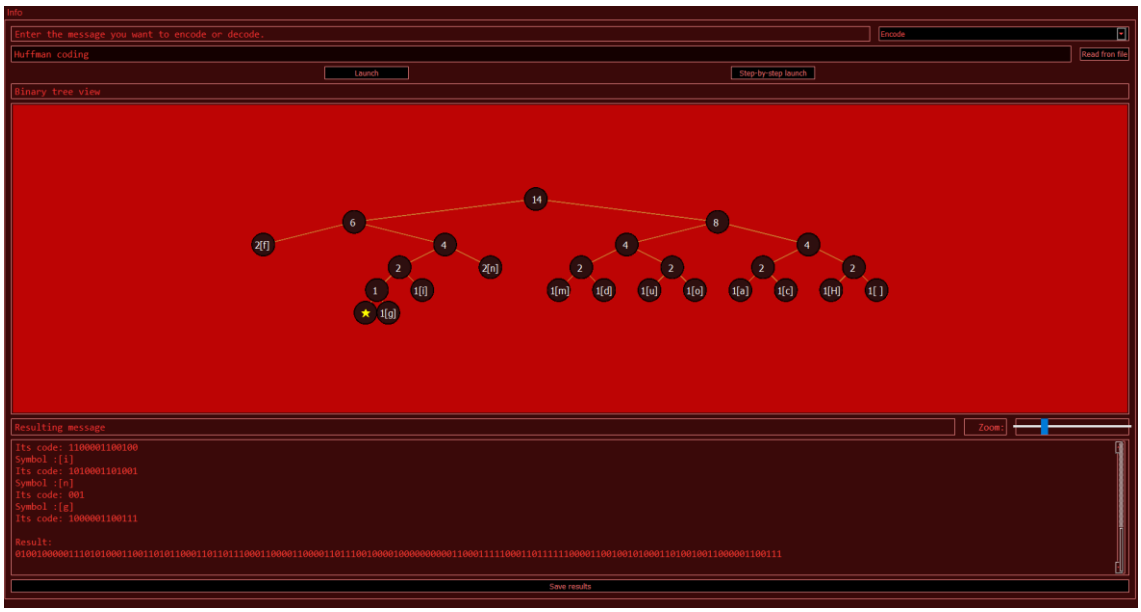

Рисунок 3 – Программа после моментального кодирования сообщения

После выполнения пользователю предоставляется возможность сохранить результирующую строку с промежуточными выводами в файл.

На рис. 4 представлен вид программы в пошаговом режиме после добавления нового элемента.

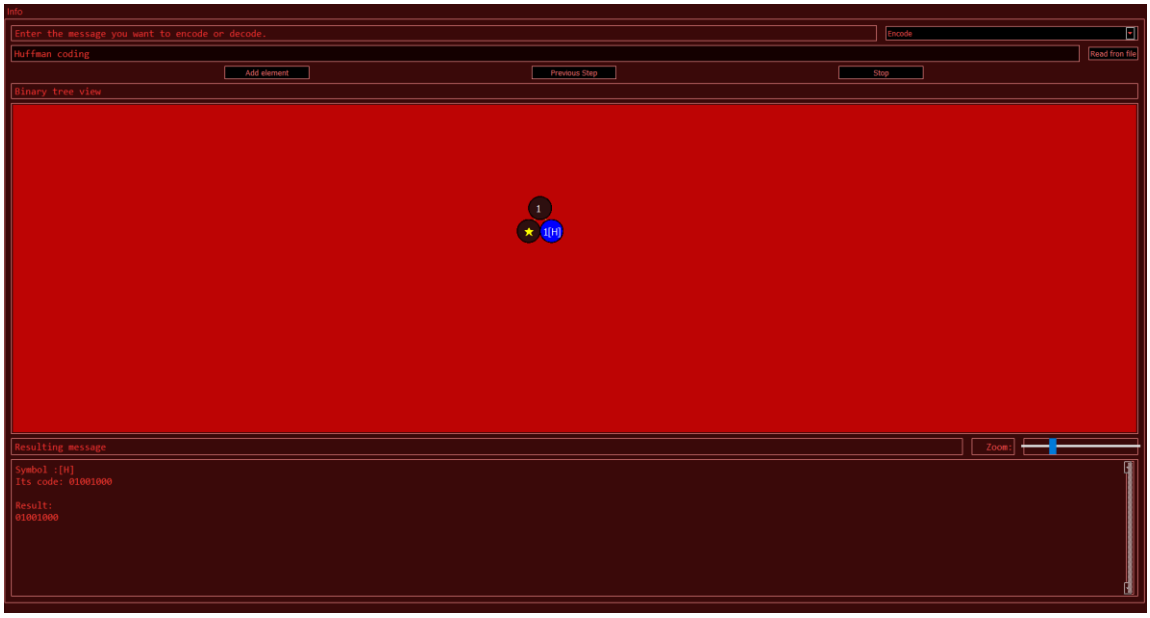

Рисунок 4 – дерево после добавления нового элемента

Последний добавленный элемент, в нашем случае самый первый, выделен синим цветом. Промежуточные выводы также обновлены.

На рис. 5 и 6 представлены виды программы до упорядочивания дерева и после соответственно.

Рисунок 5 – Дерево до упорядочивания

Рисунок 6 – Дерево после упорядочивания

Узлы дерева, которые поменялись местами выделяются фиолетовым цветом. Промежуточные выводы при этом никак не обновляются.

4.3. Демонстрация алгоритма декодирования

На рис.7 представлен вид программы после моментального выполнения алгоритма динамического декодирования Хаффмана.

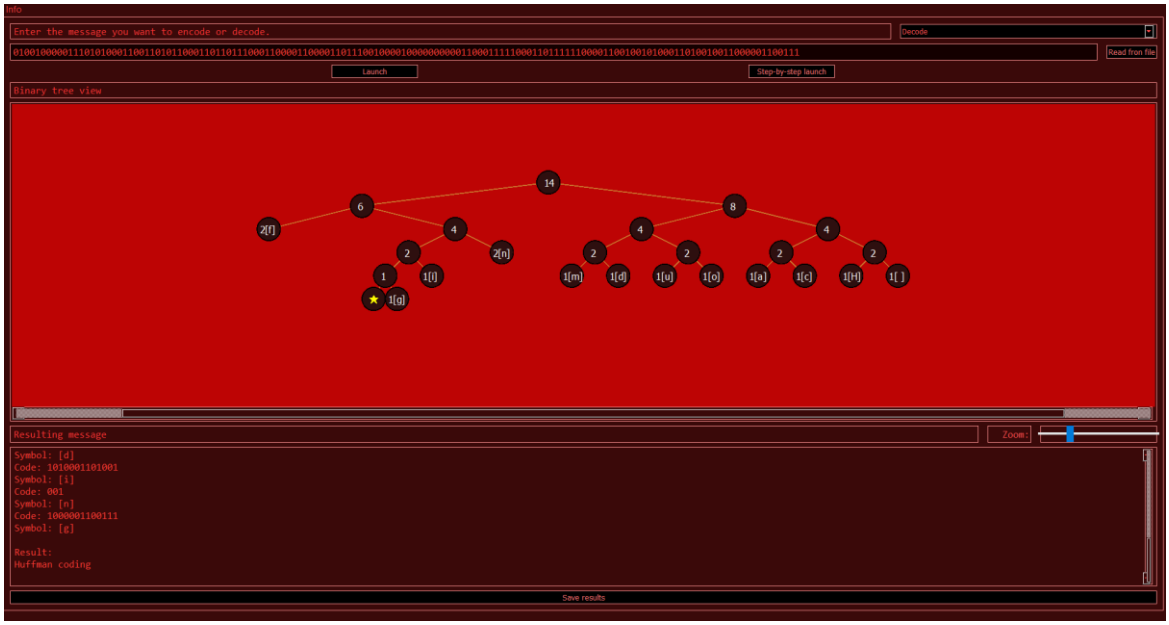

Рисунок 7 – Программа после моментального декодирования

После выполнения пользователю также предоставляется возможность сохранить выходную строку в файл.

На рис. 8 представлен вид программы в пошаговом режиме во время считывания очередного бита из входного кода.

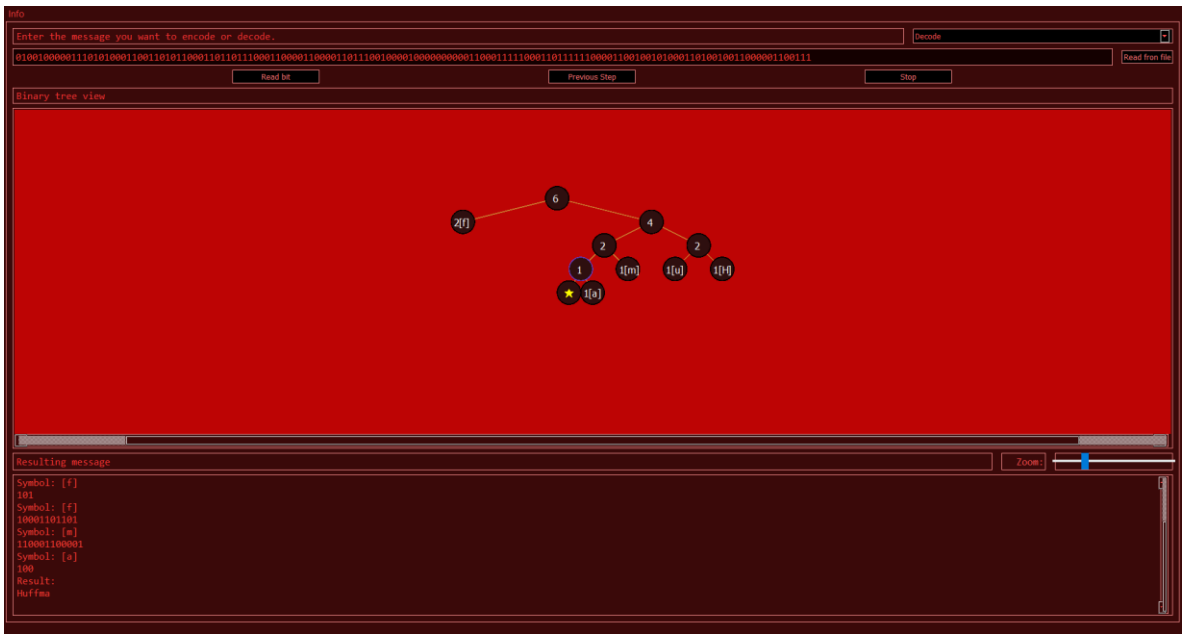

Рисунок 8 – Считывание бита во время декодирования

Синим контуром подсвечен текущий узел дерева, на котором остановился алгоритм при считывании бита. Данный шаг выполняется до тех пор, пока очередной считанный бит не приведет к листу дерева. В случае если строка закончилась и текущий элемент не лист, то программа обрывается с предупреждающим сообщением.

Шаги добавления и упорядочивания дерева аналогично выглядят как в алгоритме кодирования и представлены на рис.4, 5 и 6.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана программа, которая обладает следующей функциональностью: демонстрирует алгоритмы Хаффмана динамического кодирования и декодирования в двух режимах: моментальном и пошаговом. Данные, по которым выполняется алгоритм можно ввести разными способами. Все операции в пошаговом режиме сопровождаются соответствующей цветовой маркировкой, пошаговым выводом и указанием, что в данный момент делает алгоритм. В ходе работы возникали сложности с созданием корректного возврата к предыдущему шагу, также возникали сложности с реализацией шага вперед, а именно: какой период выполнения алгоритма считать за один шаг. Результативную программу можно использовать в обучающих целях, для изучения алгоритмов Хаффмана.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Bjarne Stroustrup. A Tour of C++. M.: Addison-Wesley, 2018. 217 с.
2. Макс Шлее. Qt5.10. Профессиональное программирование на C++. М.: BHV-СПб, 2018, 513 с.
3. Адаптивный алгоритм Хаффмана сжатия информации/ Кудрина М.А., Кудрин К.А., Дегтярева О.А., Сопченко Е.В. //Труды Международного симпозиума «Надежность и качество», 2015, том 1 2015. 134 с.
4. Перевод и дополнение документации QT // CrossPlatform.RU. URL: http://doc.crossplatform.ru (дата обращения: 17.12.2019).
5. Adaptive Huffman Coding. URL: https://www2.cs.duke.edu/csed/curious/compression/adaptivehuff.html (дата обращения: 15.12.2019).
6. Qt Documentation // Qt. URL: https://doc.qt.io/qt-5/index.html (дата обращения: 15.12.2019).

ПРИЛОЖЕНИЕ А.

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();
}
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ. BINTREE.H

```
#ifndef BINTREE_H
#define BINTREE_H
#include <memory>
#include <iostream>
#include <string>
#include <QString>
#include <sstream>
#include <QVector>
#include <QQueue>
#include <algorithm>

#define INVALID -1 /* setted as the symbol of the non-leaf nodes */
#define HOW_MANY_POSSIBLE_SYMBOLS 256 /* how many possible symbols */

struct Node{
    Node *left = nullptr;
    Node *right =nullptr;
    Node *parent;
    int value;
    bool isZero;
    char symbol;
    bool isRoot;
    bool isLeaf;
    int level = 1;
    bool isLast;
    bool isHere;
    bool isSwapped;
};


class BinTree{
private:
    Node* root = nullptr;
    Node* firstSwap = nullptr;
    Node* secondSwap = nullptr;
    QVector<Node*> makeArray();
    void swapNodes();
    Node* copyRoot(Node* tree, Node*parent) ;
    Node* copyNode(Node* node);
public:
    BinTree(){
        root = new Node;
        root->isZero = true;
        root->isRoot = true;
        root->isLeaf = true;
        root->isSwapped = false;
        root->isLast = false;
        root->isHere = true;
        root->parent = nullptr;
        root->left = nullptr;
        root->right = nullptr;
```

```
        root->symbol = '\n';
        root->value = 0;
    }
    ~BinTree(){
        this->freeMem(root);
    }
    bool needUpdateTree();
    void getCurNode(Node* tree,Node*& curNode) const;
    void getZeroNode(Node* tree, Node*& node) const;
    bool findNodeSymb(Node* tree, Node*& nodeSymb, char symb,bool& flag) const;
    int getMaxTreeDepth(Node* node) const;
    Node* getRoot() const;
    void freeMem(Node* node = nullptr);
    void setOrdinaryNodeColor(Node* tree);
    BinTree* copyTree(BinTree* tree);
    void swapNodesForOrdering();
};

void recalculationNodeValue(Node* node);


#endif // BINTREE_H
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ.

DECODINGANDCODINGHUUFMANALGORITHM.H

```
#ifndef DECODINGANDCODINGHUUFMANALGORITHM_H
#define DECODINGANDCODINGHUUFMANALGORITHM_H
#include "bintree.h"
//Node* getTreeFromSymbol(unsigned char symbol, Symbol **symbols);
void reverseCode(int *code,int codeSize);
int* codeOfNode(Node *node, int *n);
Node* addChild(Node *parent, bool isZero, bool isRoot, char symbol, int value,
bool isLast);
Node* addSymbol(char symbol, Node** zeroNode);
void addCodeToOut(std::string&outp,std::string& resultCode,int
codeSize,int*symbCode, char byte,bool flag);
void addAsciiToOut(std::string&outp,std::string& resultCode,int byte);
void encode(char* input, std::string& output, int inputSzie,BinTree*
root,std::string& resultCode);
void addNewNode(BinTree* tree,Node*& zeroNode,char symbol,std::string& out-
put,std::string& resultCode);
bool decode(char* input, std::string& output, int inputSize, BinTree*
tree,std::string& resultCode);
char readByte(char* input,int& curIndex, int inputSize,std::string& output);
void readOneBit(Node*& curNode,std::string& output,char* input,int& currentIn-
putLen);
bool readCurNode(char* input,int& currentInputLen,int inputLen,std::string&
output,std::string& resultCode,Node*& curNode,BinTree* tree,int& diffCur-
rentInLen);
#endif // DECODINGANDCODINGHUUFMANALGORITHM_H
```

ПРИЛОЖЕНИЕ Г

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.H

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QObject>
#include <QMessageBox>
#include <QDebug>
#include <QString>
#include <QFileDialog>
#include <QGraphicsItem>
#include <QtGui>
#include <QDialog>
#include <QColorDialog>
#include <QString>
#include <QDebug>
#include <QPainter>
#include <QComboBox>
#include <QLabel>
#include <QPushButton>
#include <QFile>
#include <QWidget>
#include <QVBoxLayout>
#include <QPushButton>
#include <QLabel>
#include <QLineEdit>
#include <QGroupBox>
#include <QRadioButton>
#include <QTextEdit>
#include <QEventLoop>
#include <QTimer>
#include <QColor>
#include <QDebug>
#include <QGraphicsView>
#include <QFormLayout>
#include <QScrollBar>
#include "decodingandcodinghuufmanalgorithm.h"
#include <fstream>
#include "infoaboutalgorithm.h"
#include "instruction.h"

#define RUSSIAN
"АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюя"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT
```

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

private slots:
    void on_startCodingButton_clicked();

    void on_stepByStepStart_clicked();

    void on_stopButton_clicked();

    void on_readFileButton_clicked();

    void on_saveButton_clicked();

    void on_addElement_clicked();

    void on_upgradeTree_clicked();

    void on_prevStep_clicked();

    void on_readBit_clicked();

    void on_addElemDecode_clicked();

    void on_horizontalSlider_sliderMoved(int position);

    void on_actionAbout_algorithm_triggered();

    void on_actionInstruction_triggered();

private:
    Ui::MainWindow *ui;
    QGraphicsScene *mainGraphicsScene;
    char* input;
    int currentInputLen = 0;
    int inputLen = 0;
    std::string output;
    std::string resultCode;
    std::string realOutput;
    QString answ;
    QPen pen;
    QColor color;
    QBrush brush;
    QFont font;
    QScrollBar* scrollBar;
    int algorithm;
    enum {decoding = 0,coding = 1};
    BinTree* tree;
    Node* curNode;
    Node* zeroNode;
    void setMode(bool isMode);
    void DrawNode(Node* n, int maxdepth, int depth = 0, int x = 0, int y = 0);
    void updateScene();
    bool readInput(bool isAlg);
```

```
    void setStepEncodeMode();
    void setOrdinaryMode();
    void setAddElemDecodeMode();
    void setReadBitMode();
    void setUpgrTreeMode();
    void freeSteps();
    void allClear();
    bool decodeCheck(QString code);
    void setLogs(std::string output,std::string resultCode );
    QString deleteSpaces(QString& str);
    enum {upgTree = 0,addSymb=1,readBit=2 , addEl=3};
    int maxTreeArrayLen = 0;
    QVector <BinTree*> treeArray;
    QVector <int> isPrevStep;
    QVector <int> diffInputLen;
    QVector <std::string> stepCode;
    QVector <std::string> stepOutput;
    QString inputString;
    InfoAboutAlgorithm* infoForm1;
    Instruction* infoForm2;
};
bool checkRussian(QString str);

#endif // MAINWINDOW_H
```

ПРИЛОЖЕНИЕ Д

ИСХОДНЫЙ КОД ПРОГРАММЫ. INFOABOUTALGORITHM.H

```
#ifndef INFOABOUTALGORITHM_H
#define INFOABOUTALGORITHM_H

#include <QWidget>

namespace Ui {
class InfoAboutAlgorithm;
}

class InfoAboutAlgorithm : public QWidget
{
    Q_OBJECT

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

private:
    Ui::InfoAboutAlgorithm *ui;
};

#endif // INFOABOUTALGORITHM_H
```

ПРИЛОЖЕНИЕ Е

ИСХОДНЫЙ КОД ПРОГРАММЫ. INSTRUCTION.H

```
#ifndef INSTRUCTION_H
#define INSTRUCTION_H

#include <QWidget>

namespace Ui {
class Instruction;
}

class Instruction : public QWidget
{
    Q_OBJECT

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

private:
    Ui::Instruction *ui;
};

#endif // INSTRUCTION_H
```

ПРИЛОЖЕНИЕ Ж

ИСХОДНЫЙ КОД ПРОГРАММЫ. BINTREE.CPP

```
#include "bintree.h"


int BinTree::getMaxTreeDepth(Node* tree) const{
    if(tree == nullptr) return 0;
        else{
            int lDepth = this->getMaxTreeDepth(tree->left);
            int rDepth = this->getMaxTreeDepth(tree->right);
            if (lDepth > rDepth)  return(lDepth + 1);
            else return(rDepth + 1);
        }
}

QVector<Node*> BinTree::makeArray(){
    QQueue<Node*> queue;
    QVector<Node*> result ;
    queue.enqueue(root);
    for(;;){
        Node* current = queue.dequeue();
       (result).push_back(current);
        if (current->left==nullptr && current->right == nullptr &&
queue.size()==0){
            break;
        }
        if (current->right!=nullptr){
            queue.enqueue(current->right);
            current->right->level = current->level +1;
        }
        if (current->left!=nullptr){
            queue.enqueue(current->left);
            current->left->level = current->level +1;
        }
    }
    std::reverse( result.begin(), result.end());
    return result;
}

Node* BinTree::copyNode(Node* node_) {
   Node* node = new Node;
   node->isZero = node_->isZero;
   node->isRoot = node_->isRoot;
   node->isLeaf = node_->isLeaf;
   node->isSwapped = node_->isSwapped;
   node->isLast = node_->isLast;
   node->symbol = node_->symbol;
   node->value = node_->value;
   node->isHere = node_->isHere;
   return node;
}

void BinTree::freeMem(Node* tree){
```

```
    if(!tree)
        return;
    if (tree->right)
        this->freeMem(tree->right);
    if (tree->left)
        this->freeMem(tree->left);
    delete tree;
    return;
}

Node* BinTree::getRoot() const{
    return root;
}



void recalculationNodeValue(Node*currNode){
    while(!(currNode->isRoot)){
        (currNode->value)++;
         currNode = currNode->parent;
    }
    if (currNode->isRoot)
         (currNode->value)++;
}


Node* BinTree::copyRoot(Node* node, Node* parent) {
    if (node == nullptr)
        return nullptr;
    Node* newnode = copyNode(node);
    if(node!=parent)
        newnode->parent = parent;
    newnode->left = copyRoot(node->left,newnode);
    newnode->right = copyRoot(node->right,newnode);
    return newnode;
}

BinTree* BinTree::copyTree(BinTree* tree){
    Node* newroot = copyRoot(tree->root,tree->root);
    BinTree* newTree = new (BinTree);
    newTree->root = newroot;
    newTree->firstSwap = nullptr;
    newTree->secondSwap = nullptr;
    return newTree;
}



bool BinTree::needUpdateTree(){
    QVector<Node*> arr;
    Node* first;
    Node* second;
    int i =0;
    first = nullptr;
    second = nullptr;
```

```
    arr= this->makeArray();
    int min=0;
    bool flag = false;

    for (i =0;i<arr.size();i++){
        min = arr[i]->value;
        for (int j = i+1;j<arr.size()-1;j++){
            if(arr[j]->value<min){
                first = arr[i];
                flag = true;
                break;
            }
        }
        if (flag)break;
    }
    if(!first)return false;

    int fir = i;
    int max = first->value;
    for(i = fir+1;i<arr.size();i++){
        if (arr[i]->value<max){
            min = arr[i]->value;
            second = arr[i];
            break;
        }
    }
    if(!second)return false;

    for(i = fir+1;i<arr.size();i++){
        if (max>arr[i]->value && min>=arr[i]->value){
            min = arr[i]->value;
            second = arr[i];
        }
     }
    if(!second)return false;
    firstSwap = first;
    secondSwap = second;

    return true;
}

void BinTree::getZeroNode(Node* tree,Node*& zeroNode) const{
    if (tree == nullptr)
        return;
    if (tree->isZero){
        zeroNode = tree;
        return;
    }
    getZeroNode(tree->left, zeroNode);

    getZeroNode(tree->right, zeroNode);
    return;
}

void BinTree::getCurNode(Node* tree,Node*& curNode) const{
```

```
    if (tree == nullptr)
        return;
    if (tree->isHere){
        curNode = tree;
        return;
    }
    getCurNode(tree->left, curNode);

    getCurNode(tree->right, curNode);
    return;
}

bool BinTree::findNodeSymb(Node* tree, Node*& nodeSymb, char symb,bool& flag)
const{
    if (nodeSymb)
        return flag;
    if (tree == nullptr)
        return flag;
    if (tree->symbol == symb ){
        nodeSymb = tree;
        flag = true;
        return flag;
    }
    findNodeSymb(tree->left, nodeSymb,symb,flag);
    findNodeSymb(tree->right, nodeSymb,symb,flag);
    return flag;
}



void BinTree::swapNodesForOrdering() {
    firstSwap->isSwapped = true;
    secondSwap->isSwapped = true;
    int diff = firstSwap->value-secondSwap->value;
    if (firstSwap==secondSwap) return;
    if (firstSwap->parent!=secondSwap->parent){
        if (firstSwap->parent->left == firstSwap) {
            firstSwap->parent->left = secondSwap;
        }
        else {
            firstSwap->parent->right = secondSwap;
        }
        if (secondSwap->parent->left == secondSwap) {
            secondSwap->parent->left = firstSwap;
        }
        else {
            secondSwap->parent->right = firstSwap;
        }
        Node* temp = firstSwap->parent;
        firstSwap->parent = secondSwap->parent;
        secondSwap->parent = temp;
        Node* s2=secondSwap->parent;
        Node *s1=firstSwap->parent;

        while(!(s2->isRoot)){
```

```
            (s2->value)-=diff;
            s2 = s2->parent;
        }
        while(!(s1->isRoot)){
            (s1->value)+=diff;
            s1 = s1->parent;
        }
    }
    else {
        if (firstSwap->parent->left ==firstSwap){
            firstSwap->parent->left = secondSwap;
            secondSwap->parent->right = firstSwap;
        }
        else{
            firstSwap->parent->right = secondSwap;
            secondSwap->parent->left = firstSwap;
        }
    }
    firstSwap = nullptr;
    secondSwap = nullptr;
    return;
}


void BinTree::setOrdinaryNodeColor(Node* tree){
    if(!tree)
        return;
    if (tree->isLast){
        tree->isLast = false;
    }
    if (tree->isSwapped){
        tree->isSwapped = false;
    }
    if (tree->isHere){
        tree->isHere = false;
    }
    if (tree->right)
        this->setOrdinaryNodeColor(tree->right);
    if (tree->left)
        this->setOrdinaryNodeColor(tree->left);
    return;
}
```

ПРИЛОЖЕНИЕ И

ИСХОДНЫЙ КОД ПРОГРАММЫ.

DECODINGANDCODINGHUUFMANALGORITHM.CPP

```
#include "decodingandcodinghuufmanalgorithm.h"


void reverseCode(int* code, int codeSize) {
    if (code == nullptr) {
        return;
    }
    int* start = code;
    int* end = code + (codeSize - 1);
    while (start < end) {
        int temp = *start;
        *start = *end;
        *end = temp;
        start++;
        end--;
    }
}

int* codeOfNode(Node* node, int* n) {
    Node* current = node;
    int* code = new int[HOW_MANY_POSSIBLE_SYMBOLS * 2];
    int i = 0;

    while (!current->isRoot) {
        Node* parent = current->parent;
        code[i] = (parent->left == current) ? 0 : 1;
        current = current->parent;
        i++;
    }
    reverseCode(code, i);
    *n = i;
    return code;
}

Node* addChild(Node* parent, bool isZero, bool isRoot, char symbol, int value)
{
    Node* node = new Node;
    node->isZero = isZero;
    node->isRoot = isRoot;
    node->isHere = false;
    node->isLast = false;
    node->isSwapped = false;
    node->isLeaf = true;
    node->parent = parent;
    node->left = nullptr;
    node->right = nullptr;
    node->symbol = symbol;
    node->value = value;
    return node;
}
```

```
Node* addSymbol(char symbol, Node** zeroNode) {
    Node* leftNode = addChild(*zeroNode, true, false, '\n', 0);
    Node* rightNode = addChild(*zeroNode, false, false, symbol, 0);

    (*zeroNode)->isZero = false;
    (*zeroNode)->isLeaf = false;
    (*zeroNode)->isLast = false;
    (*zeroNode)->isSwapped = false;
    (*zeroNode)->isHere = false;
    (*zeroNode)->left = leftNode;
    (*zeroNode)->right = rightNode;

    *zeroNode = leftNode;
    return rightNode;
}



template <typename T>
std::string NumberToString(T Number)
{
    std::ostringstream ss;
    ss << Number;
    return ss.str();
}


void addCodeToOut(std::string& outp,std::string& resultCode, int codeSize, int*
symbCode, char byte, bool flag) {
    outp =outp+ "Symbol :["+byte+"]\nIts code: ";
    for (int i = 0; i < codeSize; i++) {
        outp += NumberToString(symbCode[i]);
        resultCode += NumberToString(symbCode[i]);
    }
    if (flag){
        addAsciiToOut(outp,resultCode,byte);
    }
    outp+="\n";
}


void addAsciiToOut(std::string& outp,std::string& resultCode, int byte) {
    for (int i = 0; i < 8; i++) {
        int some = byte;
        some = some & 128;
        (some == 128) ? outp += "1" : outp += "0";
        (some == 128) ? resultCode += "1" : resultCode+= "0";
        byte <<= 1;
    }
}
```

```
void addNewNode(BinTree* tree,Node*& zeroNode,char symbol,std::string& out-
put,std::string& resultCode){
    Node* symbolTree = nullptr;
    Node* newNode = nullptr;
    bool flag = false;
    if(tree->findNodeSymb(tree->getRoot(),symbolTree,symbol,flag)){
        int codeSize;
        int* symbolCode = codeOfNode(symbolTree, &codeSize);
        addCodeToOut(output,resultCode, codeSize, symbolCode,symbol,false);
        recalculationNodeValue(symbolTree);
        delete symbolCode;
    }
    else{
        int codeSize;
        int* zeroCode = codeOfNode(zeroNode, &codeSize);
        addCodeToOut(output,resultCode, codeSize, zeroCode,symbol,true);
        newNode = addSymbol(symbol, &zeroNode);
        recalculationNodeValue(newNode);
        delete zeroCode;
    }
    if (!flag && newNode){
        newNode->isLast = true;
    }
    else if (flag && symbolTree){
         symbolTree->isLast = true;
    }
}



void encode(char* input, std::string& output, int inputSize, BinTree*
tree,std::string& resultCode) {
    Node* zeroNode = tree->getRoot();
    bool flag = false;
    char currByte;
    Node* newNode = nullptr;
    Node* symbolTree = nullptr;
    for (int i = 0; i < inputSize; i++) {
        currByte = input[i];
        symbolTree = nullptr;
        flag = false;
        if (tree->findNodeSymb(tree->getRoot(),symbolTree,currByte,flag)) {
            int codeSize;
            int* symbolCode = codeOfNode(symbolTree, &codeSize);
            addCodeToOut(output,resultCode, codeSize, sym-
bolCode,currByte,false);
            recalculationNodeValue(symbolTree);
            while(tree->needUpdateTree()){
                tree->swapNodesForOrdering();
            }
            delete symbolCode;
        }
        else {
            int codeSize;
            int* zeroCode = codeOfNode(zeroNode, &codeSize);
```

```
            addCodeToOut(output,resultCode, codeSize, zeroCode,currByte,true);
            newNode = addSymbol(currByte, &zeroNode);
            recalculationNodeValue(newNode);
            while(tree->needUpdateTree()){
                tree->swapNodesForOrdering();
            }
            delete zeroCode;
        }

    }

    if (!flag && newNode){
        newNode->isLast = true;
    }
    else if (flag && symbolTree){
         symbolTree->isLast = true;
    }

}

char readByte(char* input,int& curIndex, int inputSize,std::string& output){
    int j =0;
    int result = 0;
    int multi = 128;
    for(int i = curIndex; i<inputSize&&j<8;i++){
        j++;
        if(input[i]=='1'){
            result+=multi;
            output+="1";
        }
        else{
            output+="0";
        }
        multi/=2;
    }
    curIndex+=8;
    char symb = result;
    return symb;
}

bool decode(char* input, std::string& output, int inputSize, BinTree*
tree,std::string& resultCode){
    Node* node = tree->getRoot();
    Node* zeroNode;
    int i =0;
    char symb;
    while(i!=inputSize){
        node = tree->getRoot();
        output +="Code: ";
        while(!node->isLeaf && i<=inputSize){
            if (input[i] == '0'){
                node = node->left;
                output+="0";
            }else if(input[i] == '1'){
                node = node->right;
```

```
                output+="1";
            }
            if (i == inputSize) return false;
            i++;
        }

        if (!node->isLeaf) return false;
        if (node->isZero){
            if (inputSize<i+8) return false;
            symb = readByte(input,i,inputSize,output);
            tree->getZeroNode(tree->getRoot(),zeroNode);
            node = addSymbol(symb, &zeroNode);
            recalculationNodeValue(node);
        }
        else{
            symb = node->symbol;
            recalculationNodeValue(node);
        }

        output= output+"\nSymbol: ["+symb+"]\n";
        resultCode+=symb;
        while(tree->needUpdateTree()){
            tree->swapNodesForOrdering();
        }
    }
    return true;
}
bool readCurNode(char* input,int& currentInputLen,int inputLen,std::string&
output,std::string& resultCode,Node*& curNode,BinTree* tree, int& diffCur-
rentInLen){
    char symb;
    diffCurrentInLen = currentInputLen;
    if (curNode->isZero){
        if (inputLen<currentInputLen+8){
            return false;
        }
        symb = readByte(input,currentInputLen,inputLen,output);
        Node* zeroNode = nullptr;
        tree->getZeroNode(tree->getRoot(),zeroNode);
        curNode = addSymbol(symb, &zeroNode);
        recalculationNodeValue(curNode);
        int some = currentInputLen - diffCurrentInLen;
        diffCurrentInLen = some;
    }
    else{
        symb = curNode->symbol;
        recalculationNodeValue(curNode);
        diffCurrentInLen = 0;
    }
    output= output+"\nSymbol: ["+symb+"]\n";
    resultCode+=symb;
    return true;
}
```

```
void readOneBit(Node*& curNode,std::string& output,char* input,int& currentIn-
putLen){
    if (input[currentInputLen] == '0'){
        curNode = curNode->left;
        curNode->isHere = true;
        output+="0";
    }
    else{
        curNode = curNode->right;
        curNode->isHere = true;
        output+="1";
    }
    currentInputLen++;
    return;
}
```

ПРИЛОЖЕНИЕ К

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.CPP

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <string>
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent),
    ui(new Ui::MainWindow),
    mainGraphicsScene(new QGraphicsScene())
{
    ui->setupUi(this);
    ui->graphicsView->setScene(mainGraphicsScene);
    this->setWindowTitle("HUFFMAN CODING/DECODING");
    this->setWindowFlags(Qt::CustomizeWindowHint);
    QMainWindow::showMaximized();
    QColor color(203,119,47);
    pen.setColor (color);
    brush.setColor(color);
    font.setFamily("Roboto");
    pen.setWidth(3);
    ui->horizontalSlider->setMinimum(1);
    ui->horizontalSlider->setValue(2);
    ui->horizontalSlider->setMaximum(5);
    ui->graphicsView->resetTransform();
    ui->graphicsView->scale(1.2/2,1.2/2);
    ui->addElement->hide();
    ui->upgradeTree->hide();
    ui->prevStep->hide();
    ui->stopButton->hide();
    ui->comboBox->addItem("Encode");
    ui->comboBox->addItem("Decode");
    ui->readBit->hide();
    ui->addElemDecode->hide();
    scrollBar =  ui->answer->verticalScrollBar();
    ui->inputStr-
>setText("01101000001110101000110011010110001101101110001100001100001101110010000100000000001100100111000110010111000011000111010001101111110110000011010011101010000001100111");
}

MainWindow::~MainWindow()
{
    delete ui;
}
void MainWindow::updateScene(){
    Node* root = tree->getRoot();
    mainGraphicsScene->clear();
    if (!root) return;
    DrawNode(root,tree->getMaxTreeDepth(root));
}
```

```
void MainWindow::DrawNode(Node * n,int maxdepth,int depth,int x,int y){
    if (n==nullptr) return;
    int offset = pow(2,maxdepth+3)/pow(2,depth);
    if (n->left)
        mainGraphicsScene->addLine(x+32,y+32,x-offset+32,y+64+32,pen);
    if (n->right)
        mainGraphicsScene->addLine(x+32,y+32,x+offset+32,y+64+32,pen);
    QColor color_c(46,15,15);
    QColor colorSwap(96,63,196);
    QBrush brush(color_c);
    QPen pen(color,3);
    if (n->isLast){
        brush.setColor(Qt::blue);
        mainGraphicsScene->addEllipse(x,y,64,64,pen,brush);
    }
    else if (n->isSwapped){
        brush.setColor(colorSwap);
        mainGraphicsScene->addEllipse(x,y,64,64,pen,brush);
    }
    else if(n->isHere){
        pen.setColor(colorSwap);
        mainGraphicsScene->addEllipse(x,y,64,64,pen,brush);
    }
    else{
        pen.setColor(color);
        brush.setColor(color_c);
        mainGraphicsScene->addEllipse(x,y,64,64,pen,brush);
    }

    QGraphicsTextItem *numb = new QGraphicsTextItem();
    numb->setDefaultTextColor(Qt::white);
    numb->setPos(x+13,y+10);
    QString textRes;
    if (QChar(n->symbol)!='\n'){
        numb->setPos(x,y+10);
        textRes = QString::number(n->value)+"["+QChar(n->symbol)+"]";
    }
    else if (n->value==0){
        numb->setDefaultTextColor(Qt::yellow);
        textRes = "★";
    }
    else
        textRes = QString::number(n->value);
    numb->setPlainText(textRes);
    numb->setScale(2);
    mainGraphicsScene->addItem(numb);
    DrawNode(n->left,maxdepth,depth+1,x-offset,y+64);
    DrawNode(n->right,maxdepth,depth+1,x+offset,y+64);

}

bool MainWindow::decodeCheck(QString code){
    QRegExp reg( "^(1*0*)*$" );
    return reg.exactMatch(code);
}
```

```
QString MainWindow::deleteSpaces(QString& input){
    QString out="";
        for (auto i = 0;i < input.length();i++){
            if (input[i]!=' ' && input[i]!='\n' && input[i]!='\t' )
                out.push_back(input[i]);
        }
    return out;
}

bool MainWindow::readInput(bool isAlg){
    inputString = ui->inputStr->text();
    if (inputString.size()==0){
        QMessageBox::critical(this,"ERROR!","No massage");
        return false;
    }
    if (!isAlg){
        inputString = deleteSpaces(inputString);
        if(!decodeCheck(inputString)){
            QMessageBox::critical(this,"ERROR!","Incorrect code");
            return false;
        }
    }
    if (checkRussian(inputString)){
        QMessageBox::critical(this,"ERROR!","Text contains Russian");
        return false;
    }
    int i;
    inputLen = inputString.size();
    input = new char[inputLen+1];
    for (i =0;i<inputLen;i++){
        input[i] = inputString[i].toLatin1();
    }
    input[inputLen] = '\0';
    ui->inputStr->setText(inputString);
    return true;
}



void MainWindow::on_startCodingButton_clicked()
{
    int value = ui->comboBox->currentIndex();
    allClear();

    if(!value){
        setOrdinaryMode();
        if (!readInput(true)) return;
        tree = new(BinTree);
        encode(input, output, inputLen, tree,resultCode);
        delete[] input;
        tree->setOrdinaryNodeColor(tree->getRoot());
        updateScene();
        setLogs(output,resultCode);
        ui->saveButton->show();
```

```
        tree->freeMem(tree->getRoot());
    }
    else{
        setOrdinaryMode();
        if(!readInput(false)) return;

        tree = new(BinTree);
        if(!decode(input, output, inputLen, tree,resultCode)){
            QMessageBox::critical(this,"ERROR!","Wrong code!");
            on_stopButton_clicked();
            return;
        }
        delete[]input;
        tree->setOrdinaryNodeColor(tree->getRoot());
        updateScene();
        ui->saveButton->show();
        setLogs(output,resultCode);
        tree->freeMem(tree->getRoot());
    }
}

void MainWindow::allClear(){
    output.clear();
    resultCode.clear();
    ui->answer->clear();
    mainGraphicsScene->clear();
}

void MainWindow::setReadBitMode(){
    ui->readFileButton->setEnabled(false);
    ui->inputStr->setEnabled(false);
    ui->startCodingButton->hide();
    ui->stepByStepStart->hide();
    ui->addElement->hide();
    ui->stopButton->show();
    ui->upgradeTree->hide();
    ui->prevStep->show();
    ui->readBit->show();
    ui->addElemDecode->hide();
    ui->saveButton->hide();
}

void MainWindow::setAddElemDecodeMode(){
    ui->readFileButton->setEnabled(false);
    ui->inputStr->setEnabled(false);
    ui->startCodingButton->hide();
    ui->stepByStepStart->hide();
    ui->addElement->hide();
    ui->stopButton->show();
    ui->upgradeTree->hide();
    ui->prevStep->show();
    ui->readBit->hide();
    ui->addElemDecode->show();
    ui->saveButton->hide();
}
```

```
void MainWindow::setOrdinaryMode(){
    ui->readFileButton->setEnabled(true);
    ui->inputStr->setEnabled(true);
    currentInputLen = 0;
    ui->startCodingButton->show();
    ui->stepByStepStart->show();
    ui->addElement->hide();
    ui->readBit->hide();
    ui->stopButton->hide();
    ui->upgradeTree->hide();
    ui->prevStep->hide();
    ui->addElemDecode->hide();
    ui->saveButton->show();
}

void MainWindow::setStepEncodeMode(){
    ui->readFileButton->setEnabled(false);
    ui->inputStr->setEnabled(false);
    ui->startCodingButton->hide();
    ui->stepByStepStart->hide();
    ui->addElement->show();
    ui->stopButton->show();
    ui->upgradeTree->hide();
    ui->prevStep->show();
    ui->readBit->hide();
    ui->saveButton->hide();
}

void MainWindow::setUpgrTreeMode(){
    ui->upgradeTree->show();
    ui->addElement->hide();
    ui->readBit->hide();
    ui->addElemDecode->hide();
}


void MainWindow::on_stepByStepStart_clicked()
{
    currentInputLen = 0;
    int value = ui->comboBox->currentIndex();
    allClear();
    if(!value){
        if (!readInput(true)) return;
        setStepEncodeMode();
        tree = new(BinTree);
        updateScene();
        algorithm = coding;
    }
    else{
        algorithm = decoding;
        if (!readInput(false)) return;
        setReadBitMode();
        tree = new(BinTree);
        updateScene();
```

```
        curNode = tree->getRoot();
    }
}



void MainWindow::on_stopButton_clicked()
{
    setOrdinaryMode();
    if (tree->getRoot())
        tree->freeMem(tree->getRoot());
    delete[]input;
    while (!treeArray.isEmpty()){
        treeArray.pop_back();
    }
    while(!isPrevStep.isEmpty()){
        isPrevStep.pop_back();
    }
    while(!stepOutput.isEmpty()){
        stepOutput.pop_back();
    }
    while(!stepCode.isEmpty()){
        stepCode.pop_back();
    }
    while (!diffInputLen.isEmpty())
        diffInputLen.pop_back();
    maxTreeArrayLen = 0;
    currentInputLen = 0;
    ui->saveButton->show();
}



void MainWindow::on_readFileButton_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this, tr("load"),
QDir::homePath(), tr("*.txt"));

    if (QString::compare(fileName, QString()) != 0)
       {
          std::ifstream f(qPrintable(fileName), std::ios::in);
          std::string out;
          getline(f, out);
          f.close();
          ui->inputStr->setText(QString::fromStdString(out));
       }
}

bool checkRussian(QString string){
    static QString russian =RUSSIAN;
    foreach(const QChar & ch, russian) {
            if(string.contains(ch)) {
                return true;
            }
        }
```

```
        return false;

}
void MainWindow::setLogs(std::string output,std::string resultCode ){
    realOutput = output +"\nResult:\n"+resultCode+"\n";
    answ = QString::fromStdString(realOutput);
    ui->answer->setText(answ);
    scrollBar->setValue(scrollBar->maximum());
}


void MainWindow::on_saveButton_clicked()
{
    QString saveString = "Source string:\n\n"+inputString+"\n"+ui->answer->toP-
lainText();
    if (ui->answer->toPlainText().isEmpty()){
        QMessageBox::critical(this,"ERROR!","Nothing to save");
        return;
    }
    QString filePath = QFileDialog::getSaveFileName(this, tr("save"),
QDir::homePath(), tr("*.txt"));
    if (QString::compare(filePath, QString()) != 0)
     {
        std::ofstream ff(qPrintable(filePath));
        ff << qPrintable(saveString);
        ff.close();
     }
}

void MainWindow::freeSteps(){
    BinTree* curr;
    while(maxTreeArrayLen>=5){
        curr = treeArray.last();
        curr->freeMem(curr->getRoot());
        treeArray.pop_back();
        if (!stepCode.isEmpty())
            stepCode.pop_back();
        if (!stepOutput.isEmpty())
            stepOutput.pop_back();
        if (!isPrevStep.isEmpty())
            isPrevStep.pop_back();
        if (!diffInputLen.isEmpty())
            diffInputLen.pop_back();
        maxTreeArrayLen--;
    }
}


void MainWindow::on_addElement_clicked()
{
    freeSteps();
    if (tree->needUpdateTree()){
         setUpgrTreeMode();
         return;
    }
```

```
    if (currentInputLen == inputLen){
        QMessageBox::about(this,"","Message is encoded");
        setOrdinaryMode();
        ui->saveButton->show();
        on_stopButton_clicked();
        return;
    }

    BinTree* newTree = tree->copyTree(tree);
    treeArray.push_front(newTree);
    isPrevStep.push_front(addSymb);
    stepOutput.push_front(output);
    stepCode.push_front(resultCode);

    maxTreeArrayLen++;
    char symbol = input[currentInputLen++];
    tree->getZeroNode(tree->getRoot(),zeroNode);
    addNewNode(tree,zeroNode,symbol,output,resultCode);
    setLogs(output,resultCode);
    updateScene();
    tree->setOrdinaryNodeColor(tree->getRoot());
    if (tree->needUpdateTree()){
         setUpgrTreeMode();
         return;
    }
}

void MainWindow::on_upgradeTree_clicked()
{
    freeSteps();
    if (!tree->needUpdateTree()){
        if (algorithm == coding)
            setStepEncodeMode();
        else{
            curNode = tree->getRoot();
            curNode->isHere = true;
            setReadBitMode();
        }
        return;
    }
    BinTree* newTree = tree->copyTree(tree);

    treeArray.push_front(newTree);
    isPrevStep.push_front(upgTree);
    stepOutput.push_front("");
    stepCode.push_front("");
    diffInputLen.push_front(0);
    maxTreeArrayLen++;

    tree->swapNodesForOrdering();
    updateScene();
    tree->setOrdinaryNodeColor(tree->getRoot());

    if (!tree->needUpdateTree()){
        if (algorithm == coding)
```

```
            setStepEncodeMode();
        else{
            curNode = tree->getRoot();
            curNode->isHere = true;
            setReadBitMode();
        }
        return;
    }
}


void MainWindow::on_prevStep_clicked()
{
    if(treeArray.isEmpty()){
        QMessageBox::about(this,"","Only five steps back");
        return;
    }
    BinTree* prevTree = treeArray.first();
    tree->freeMem(tree->getRoot());
    tree = prevTree;
    maxTreeArrayLen--;
    int diff = 0;
    if (!diffInputLen.isEmpty()){
        diff = diffInputLen.first();
        diffInputLen.pop_front();
    }
    int prevStep = isPrevStep.first();
    switch (prevStep) {
        case 1://addSymb
            setStepEncodeMode();
            currentInputLen--;
            output = stepOutput.first();
            resultCode = stepCode.first();
            setLogs(output,resultCode);
        break;
        case 0://upgTree
             setUpgrTreeMode();
        break;
        case 2://readBit
            setReadBitMode();
            currentInputLen-=diff;
            output = stepOutput.first();
            resultCode = stepCode.first();
            tree->getCurNode(tree->getRoot(),curNode);
            setLogs(output,resultCode);
        break;
        case 3://addEl
            setAddElemDecodeMode();
            tree->getCurNode(tree->getRoot(),curNode);
            currentInputLen-=diff;
            output = stepOutput.first();
            resultCode = stepCode.first();
            setLogs(output,resultCode);
        break;
    }
```

```
    updateScene();
    isPrevStep.pop_front();
    treeArray.pop_front();
    stepOutput.pop_front();
    stepCode.pop_front();
}

void MainWindow::on_readBit_clicked()
{
    freeSteps();
    if (currentInputLen == inputLen){
        QMessageBox::about(this,"","Message is decoded");
        setOrdinaryMode();
        on_stopButton_clicked();
        return;
    }
    if (curNode->isLeaf&&curNode->isZero&&curNode->isRoot){
        setAddElemDecodeMode();
        tree->setOrdinaryNodeColor(tree->getRoot());
        return;
    }
    if (curNode->isLeaf){
        setLogs(output,resultCode);
        setAddElemDecodeMode();
        return;
    }

    BinTree* newTree = tree->copyTree(tree);
    treeArray.push_front(newTree);
    isPrevStep.push_front(readBit);
    stepOutput.push_front(output);
    stepCode.push_front(resultCode);
    maxTreeArrayLen++;

    if(!curNode->isLeaf && currentInputLen<=inputLen){
        tree->setOrdinaryNodeColor(tree->getRoot());
        readOneBit(curNode,output,input,currentInputLen);
        diffInputLen.push_front(1);
        setLogs(output,resultCode);
        updateScene();
        if (curNode->isLeaf){
            setAddElemDecodeMode();
        }
    }
    else{
        QMessageBox::about(this,"","Incorrect code, last isn't leaf");
        on_stopButton_clicked();
    }
}

void MainWindow::on_addElemDecode_clicked()
{
    freeSteps();
    BinTree* newTree = tree->copyTree(tree);
    treeArray.push_front(newTree);
```

```
    isPrevStep.push_front(addEl);
    stepOutput.push_front(output);
    stepCode.push_front(resultCode);
    maxTreeArrayLen++;

    int diffCurentInputLen=0;
    if (!readCurNode(input,currentInputLen,inputLen,output,resultCode,cur-
Node,tree,diffCurentInputLen)){
        QMessageBox::about(this,"","Incorrect code, need 8 bits for ASKII");
        on_stopButton_clicked();
        return;
    }
    diffInputLen.push_front(diffCurentInputLen);
    setLogs(output,resultCode);
    updateScene();
    if (currentInputLen == inputLen){
        QMessageBox::about(this,"","Message is decoded");
        setOrdinaryMode();
        on_stopButton_clicked();
        return;
    }
    if (tree->needUpdateTree()){
         setUpgrTreeMode();
         return;
    }
    setReadBitMode();
    tree->setOrdinaryNodeColor(tree->getRoot());
    curNode = tree->getRoot();
    curNode->isHere = true;
}

void MainWindow::on_horizontalSlider_sliderMoved(int position)
{
    double off = position/2.0+1;
    ui->graphicsView->resetTransform();
    ui->graphicsView->scale(1.2 / off, 1.2 / off);
}

void MainWindow::on_actionAbout_algorithm_triggered()
{
    infoForm1 = new InfoAboutAlgorithm();
    infoForm1->show();
}

void MainWindow::on_actionInstruction_triggered()
{
    infoForm2 = new Instruction();
    infoForm2->show();
}
```

ПРИЛОЖЕНИЕ Л

ИСХОДНЫЙ КОД ПРОГРАММЫ.INSTRUCTIOM.CPP

```
#include "instruction.h"
#include "ui_instruction.h"

Instruction::Instruction(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Instruction)
{
    ui->setupUi(this);
    this->setWindowTitle("Instruction");
    setAttribute(Qt::WA_DeleteOnClose);
}

Instruction::~Instruction()
{
    delete ui;
}
```

ПРИЛОЖЕНИЕ М

ИСХОДНЫЙ КОД ПРОГРАММЫ.INFOABOUTALGORITHM.CPP

```
#include "infoaboutalgorithm.h"
#include "ui_infoaboutalgorithm.h"

InfoAboutAlgorithm::InfoAboutAlgorithm(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::InfoAboutAlgorithm)
{
    ui->setupUi(this);
    this->setWindowTitle("Info about algorithm");
    setAttribute(Qt::WA_DeleteOnClose);
}

InfoAboutAlgorithm::~InfoAboutAlgorithm()
{
    delete ui;
}
```

ПРИЛОЖЕНИЕ Н

ИСХОДНЫЙ КОД ПРОГРАММЫ. INFOABOUTALGORITHM.UI

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>InfoAboutAlgorithm</class>
 <widget class="QWidget" name="InfoAboutAlgorithm">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>1496</width>
    <height>900</height>
   </rect>
  </property>
  <property name="minimumSize">
   <size>
    <width>700</width>
    <height>500</height>
   </size>
  </property>
  <property name="maximumSize">
   <size>
    <width>1496</width>
    <height>922</height>
   </size>
  </property>
  <property name="windowTitle">
   <string>Form</string>
  </property>
  <property name="styleSheet">
   <string notr="true">border-color: rgb(255, 20, 20);
border-width : 1.2px;
border-style: ridge;
margin-left: 4px;
margin-right: 4px;
padding:2px;
color: rgb(255, 76, 76);
background-color: rgb(58, 9, 9);
color: rgb(255, 60, 60);
</string>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout">
   <item>
    <widget class="QTextBrowser" name="textBrowser">
     <property name="minimumSize">
      <size>
       <width>600</width>
       <height>400</height>
      </size>
     </property>
     <property name="maximumSize">
      <size>
       <width>1500</width>
```

```
        <height>900</height>
       </size>
      </property>
      <property name="styleSheet">
       <string notr="true">color: rgb(255, 58, 51);</string>
      </property>
      <property name="html">
       <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; con-
tent=&quot;1&quot; /&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'MS Shell Dlg 2';
font-size:7.8pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;h2 align=&quot;center&quot; style=&quot; margin-top:16px; margin-bot-
tom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-in-
dent:0px;&quot;&gt;&lt;span style=&quot; font-family:'MS Shell Dlg 2'; font-
size:8pt; font-weight:600;&quot;&gt;  &lt;/span&gt;&lt;span style=&quot; font-
family:'MS Shell Dlg 2'; font-size:14pt; font-weight:600;&quot;&gt;Адаптивный
алгоритм Хаффмана&lt;/span&gt;&lt;/h2&gt;
&lt;p align=&quot;justify&quot; style=&quot; margin-top:12px; margin-bot-
tom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-in-
dent:0px;&quot;&gt;&lt;span style=&quot; font-family:'MS Shell Dlg 2'; font-
size:12pt;&quot;&gt;  В адаптивном алгоритме сжатия Хаффмана используется упо-
рядоченное бинарное дерево. Бинарное дерево называется упорядоченным, если его
узлы могут быть перечислены в порядке не убывания веса. Перечисление узлов про-
исходит по ярусам снизу-вверх и слева-направо в каждом ярусе. Узлы, имеющие об-
щего родителя, находятся рядом на одном ярусе. &lt;/span&gt;&lt;/p&gt;
&lt;p align=&quot;justify&quot; style=&quot; margin-top:12px; margin-
bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-
indent:0px;&quot;&gt;&lt;span style=&quot; font-family:'MS Shell Dlg 2'; font-
size:12pt;&quot;&gt;  Адаптивный алгоритм Хаффмана является модификацией обыч-
ного алгоритма Хаффмана сжатия сообщений. Он позволяет не передавать таблицу
кодов и ограничиться одним проходом по сообщению, как при кодировании, так и
при декодировании. Суть адаптивного алгоритма состоит в том, что при каждом со-
поставлении символу кода изменяется внутренний ход вычислений так, что в следу-
ющий раз этому же символу может быть сопоставлен другой код, т.е. происходит
адаптация алгоритма к поступающим для кодирования символам.
&lt;/span&gt;&lt;/p&gt;
&lt;p align=&quot;justify&quot; style=&quot; margin-top:12px; margin-bot-
tom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-in-
dent:0px;&quot;&gt;&lt;span style=&quot; font-family:'MS Shell Dlg 2'; font-
size:12pt; font-weight:600;&quot;&gt;Правила кодирования:
&lt;/span&gt;&lt;/p&gt;
&lt;p align=&quot;justify&quot; style=&quot; margin-top:12px; margin-bot-
tom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-in-
dent:0px;&quot;&gt;&lt;span style=&quot; font-family:'MS Shell Dlg 2'; font-
size:12pt;&quot;&gt;1.Элементы входного сообщения считываются побайтно.
&lt;/span&gt;&lt;/p&gt;
&lt;p align=&quot;justify&quot; style=&quot; margin-top:12px; margin-bot-
tom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-in-
dent:0px;&quot;&gt;&lt;span style=&quot; font-family:'MS Shell Dlg 2'; font-
size:12pt;&quot;&gt;2.Если входной символ присутствует в дереве, в выходной по-
ток записывается код, соответствующий последовательности нулей и единиц, кото-
рыми помечены ветки дерева, при проходе от корня дерева к данному листу. Вес
```

данного листа увеличивается на 1. Веса узлов-предков корректируются. Если дерево становится неупорядоченным - упорядочивается. </p>
<p align="justify" style=" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">3.Если очередной символ, считанный из входного сообщения при сжатии, отсутствует в дереве, в выходной поток записывается набор нулей и единиц, которыми помечены ветки бинарного дерева при движении от корня к escape-символу, а затем 8 бит ASCII-кода нового символа. В дерево вместо escape-символа добавляется ветка: родитель, два потомка. Левый потомок становится escape-символом, правый - новым добавленным в дерево символом. Веса узлов-предков корректируются, а дерево при необходимости упорядочивается. </p>
<p align="justify" style=" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">Правила декодирования:</p>
<p align="justify" style=" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">1.Элементы входного сообщения считываются побитно. </p>
<p align="justify" style=" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">2.Каждый раз при считывании 0 или 1 происходит перемещение от корня вниз по соответствующей ветке бинарного дерева Хаффмана, до тех пор, пока не будет достигнут какой-либо лист дерева.</p>
<p align="justify" style=" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">3.Если достигнут лист, соответствующий символу, в выходное сообщение записывается ASCII-код данного символа. Вес листа увеличивается на 1, веса узлов-предков корректируются, дерево при необходимости упорядочивается. </p>
<p align="justify" style=" margin-top:12px; margin-bottom:12px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">4.Если же достигнут escape-символ, из входного сообщения считываются 8 следующих бит, соответствующих ASCII-коду нового символа. В выходное сообщение записывается ASCII-код данного символа. В дерево добавляется новый символ, веса узлов-предков корректируются, затем при необходимости производится его упорядочивание.</p></body></html>*</string>*

```
    </property>
   </widget>
  </item>
 </layout>
</widget>
<resources/>
<connections/>
     </ui>
```

ПРИЛОЖЕНИЕ П

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.UI

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Instruction</class>
 <widget class="QWidget" name="Instruction">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>881</width>
    <height>568</height>
   </rect>
  </property>
  <property name="minimumSize">
   <size>
    <width>700</width>
    <height>500</height>
   </size>
  </property>
  <property name="maximumSize">
   <size>
    <width>1496</width>
    <height>922</height>
   </size>
  </property>
  <property name="windowTitle">
   <string>Form</string>
  </property>
  <property name="styleSheet">
   <string notr="true">border-color: rgb(255, 20, 20);
border-width : 1.2px;
border-style: ridge;
margin-left: 4px;
margin-right: 4px;
padding:2px;
color: rgb(255, 76, 76);
background-color: rgb(58, 9, 9);
color: rgb(255, 60, 60);
</string>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout">
   <item>
    <widget class="QTextBrowser" name="textBrowser">
     <property name="minimumSize">
      <size>
       <width>600</width>
       <height>400</height>
      </size>
     </property>
     <property name="maximumSize">
      <size>
       <width>1500</width>
```

```
       <height>900</height>
      </size>
     </property>
     <property name="html">
      <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; con-
tent=&quot;1&quot; /&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'MS Shell Dlg 2';
font-size:7.8pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-
right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot;
font-family:'MS Shell Dlg 2'; font-size:12pt;&quot;&gt;Программа визуализирует
динамическое кодирование и декодирование Хаффмана.&lt;/span&gt;&lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-
right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot;
font-family:'MS Shell Dlg 2'; font-size:12pt;&quot;&gt;При обычном запуске про-
граммы, дерево отрисовывается полностью. На узлах дерева указан вес, если узел
- лист, то также указан символ, который он хранит, escape-символ отмечен звез-
дой. &lt;/span&gt;&lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-
right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot;
font-family:'MS Shell Dlg 2'; font-size:12pt;&quot;&gt; &lt;/span&gt;&lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-
right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot;
font-family:'MS Shell Dlg 2'; font-size:12pt;&quot;&gt;При запуске в пошаговом
режиме алгоритма кодирования синим цветом выделяются последние добавленные
узлы, а фиолетовым цветом выделяются узлы которые поменялись местами при упоря-
довачивании дерева.&lt;/span&gt;&lt;/p&gt;
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-
family:'MS Shell Dlg 2'; font-size:12pt;&quot;&gt;&lt;br /&gt;&lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-
right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot;
font-family:'MS Shell Dlg 2'; font-size:12pt;&quot;&gt;При запуске в пошаговом
режиме алгоритма декодирования синим контуром выделяется считанный бит, синим
цветом выделяются последние добавленные узлы, а фиолетовым цветом выделяются
узлы которые поменялись местами при упорядовачивании де-
рева.&lt;/span&gt;&lt;/p&gt;
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-
family:'MS Shell Dlg 2'; font-size:12pt;&quot;&gt;&lt;br /&gt;&lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-
right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot;
font-family:'MS Shell Dlg 2'; font-size:12pt;&quot;&gt;В пошаговом режиме вся
информация обновляется с каждым шагом и выводится в поле тек-
ста.&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
     </property>
    </widget>
   </item>
  </layout>
 </widget>
 <resources/>
 <connections/>
```

```
</ui>
```

ПРИЛОЖЕНИЕ Р

ИСХОДНЫЙ КОД ПРОГРАММЫ.

ADAPTIVEHUFFMANCODINGANDDECODING.PRO

```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

# The following define makes your compiler emit warnings if you use
# any Qt feature that has been marked deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version
of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs
deprecated before Qt 6.0.0

SOURCES += \
    bintree.cpp \
    decodingandcodinghuufmanalgorithm.cpp \
    infoaboutalgorithm.cpp \
    instruction.cpp \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    bintree.h \
    decodingandcodinghuufmanalgorithm.h \
    infoaboutalgorithm.h \
    instruction.h \
    mainwindow.h

FORMS += \
    infoaboutalgorithm.ui \
    instruction.ui \
    mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

ПРИЛОЖЕНИЕ С

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.UI

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>902</width>
    <height>689</height>
   </rect>
  </property>
  <property name="minimumSize">
   <size>
    <width>902</width>
    <height>689</height>
   </size>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <property name="styleSheet">
   <string notr="true">background-color: rgb(58, 9, 9);
color: rgb(255, 60, 60);
</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <property name="styleSheet">
    <string notr="true">border-color: rgb(255, 20, 20);
border-width : 1.2px;
border-style: ridge;
margin-left: 4px;
margin-right: 4px;
padding:2px;
color: rgb(255, 76, 76);</string>
   </property>
   <layout class="QVBoxLayout" name="verticalLayout">
    <item>
     <layout class="QHBoxLayout" name="horizontalLayout_4">
      <item>
       <widget class="QLabel" name="label">
        <property name="maximumSize">
         <size>
          <width>121212</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="font">
         <font>
          <family>Consolas</family>
          <pointsize>8</pointsize>
```

```
         </font>
        </property>
        <property name="styleSheet">
         <string notr="true">color: rgb(253, 51, 51);</string>
        </property>
        <property name="text">
         <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span
style=&quot; font-size:10pt;&quot;&gt;Enter the message you want to encode or
decode.&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QComboBox" name="comboBox">
        <property name="minimumSize">
         <size>
          <width>431</width>
          <height>0</height>
         </size>
        </property>
        <property name="maximumSize">
         <size>
          <width>200</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="styleSheet">
         <string notr="true">background-color: rgb(0, 0, 0);</string>
        </property>
       </widget>
      </item>
     </layout>
    </item>
    <item>
     <layout class="QHBoxLayout" name="horizontalLayout">
      <item>
       <widget class="QLineEdit" name="inputStr">
        <property name="font">
         <font>
          <family>Consolas</family>
          <pointsize>10</pointsize>
          <weight>50</weight>
          <italic>false</italic>
          <bold>false</bold>
         </font>
        </property>
        <property name="styleSheet">
         <string notr="true">
color: rgb(255, 58, 51);

background-color: rgb(20, 0, 0);</string>
        </property>
        <property name="maxLength">
         <number>1000</number>
        </property>
```

```
       </widget>
      </item>
      <item>
       <widget class="QPushButton" name="readFileButton">
        <property name="maximumSize">
         <size>
          <width>110</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="styleSheet">
         <string notr="true">
background-color: rgb(0, 0, 0);
color: rgb(255, 110, 110);</string>
        </property>
        <property name="text">
         <string>Read fron file</string>
        </property>
       </widget>
      </item>
     </layout>
    </item>
    <item>
     <layout class="QHBoxLayout" name="horizontalLayout_2">
      <item>
       <widget class="QPushButton" name="startCodingButton">
        <property name="maximumSize">
         <size>
          <width>150</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="styleSheet">
         <string notr="true">
background-color: rgb(0, 0, 0);
color: rgb(255, 110, 110);</string>
        </property>
        <property name="text">
         <string>Launch</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QPushButton" name="stepByStepStart">
        <property name="maximumSize">
         <size>
          <width>150</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="styleSheet">
         <string notr="true">
background-color: rgb(0, 0, 0);
color: rgb(255, 110, 110);</string>
        </property>
```

```
        <property name="text">
         <string>Step-by-step launch</string>
        </property>
       </widget>
      </item>
     </layout>
    </item>
    <item>
     <layout class="QHBoxLayout" name="horizontalLayout_3">
      <item>
       <widget class="QPushButton" name="readBit">
        <property name="maximumSize">
         <size>
          <width>150</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="styleSheet">
         <string notr="true">
background-color: rgb(0, 0, 0);
color: rgb(255, 110, 110);</string>
        </property>
        <property name="text">
         <string>Read bit</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QPushButton" name="addElemDecode">
        <property name="maximumSize">
         <size>
          <width>150</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="styleSheet">
         <string notr="true">background-color: rgb(42, 0, 79);
color: rgb(255, 110, 110);</string>
        </property>
        <property name="text">
         <string>Add element</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QPushButton" name="addElement">
        <property name="maximumSize">
         <size>
          <width>150</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="styleSheet">
         <string notr="true">
background-color: rgb(0, 0, 0);
```

```
color: rgb(255, 110, 110);</string>
        </property>
        <property name="text">
         <string>Add element</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QPushButton" name="upgradeTree">
        <property name="maximumSize">
         <size>
          <width>150</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="styleSheet">
         <string notr="true">background-color: rgb(95, 0, 42);
color: rgb(255, 110, 110);</string>
        </property>
        <property name="text">
         <string>Update tree</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QPushButton" name="prevStep">
        <property name="maximumSize">
         <size>
          <width>150</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="styleSheet">
         <string notr="true">
background-color: rgb(0, 0, 0);
color: rgb(255, 110, 110);</string>
        </property>
        <property name="text">
         <string>Previous Step</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QPushButton" name="stopButton">
        <property name="maximumSize">
         <size>
          <width>150</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="styleSheet">
         <string notr="true">
background-color: rgb(0, 0, 0);
color: rgb(255, 110, 110);</string>
        </property>
```

```
        <property name="text">
         <string>Stop</string>
        </property>
       </widget>
      </item>
     </layout>
    </item>
    <item>
     <widget class="QLabel" name="label_2">
      <property name="maximumSize">
       <size>
        <width>121212</width>
        <height>16777215</height>
       </size>
      </property>
      <property name="font">
       <font>
        <family>Consolas</family>
       </font>
      </property>
      <property name="styleSheet">
       <string notr="true">color: rgb(253, 51, 51);</string>
      </property>
      <property name="text">
       <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span
style=&quot; font-size:10pt;&quot;&gt;Binary tree
view&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QGraphicsView" name="graphicsView">
      <property name="backgroundBrush">
       <brush brushstyle="SolidPattern">
        <color alpha="255">
         <red>189</red>
         <green>4</green>
         <blue>4</blue>
        </color>
       </brush>
      </property>
     </widget>
    </item>
    <item>
     <layout class="QHBoxLayout" name="horizontalLayout_5">
      <item>
       <widget class="QLabel" name="label_3">
        <property name="minimumSize">
         <size>
          <width>600</width>
          <height>0</height>
         </size>
        </property>
        <property name="maximumSize">
         <size>
```

```
          <width>10000</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="font">
         <font>
          <family>Consolas</family>
         </font>
        </property>
        <property name="styleSheet">
         <string notr="true">color: rgb(253, 51, 51);</string>
        </property>
        <property name="text">
         <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span
style=&quot; font-size:10pt;&quot;&gt;Resulting mes-
sage&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QLabel" name="label_4">
        <property name="minimumSize">
         <size>
          <width>70</width>
          <height>0</height>
         </size>
        </property>
        <property name="maximumSize">
         <size>
          <width>80</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="font">
         <font>
          <family>Consolas</family>
         </font>
        </property>
        <property name="text">
         <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p
align=&quot;right&quot;&gt;&lt;span style=&quot; font-
size:10pt;&quot;&gt;Zoom:&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</str
ing>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QSlider" name="horizontalSlider">
        <property name="minimumSize">
         <size>
          <width>200</width>
          <height>0</height>
         </size>
        </property>
        <property name="maximumSize">
```

```
         <size>
          <width>200</width>
          <height>16777215</height>
         </size>
        </property>
        <property name="orientation">
         <enum>Qt::Horizontal</enum>
        </property>
       </widget>
      </item>
     </layout>
    </item>
    <item>
     <widget class="QTextBrowser" name="answer">
      <property name="minimumSize">
       <size>
        <width>0</width>
        <height>40</height>
       </size>
      </property>
      <property name="maximumSize">
       <size>
        <width>16777215</width>
        <height>230</height>
       </size>
      </property>
      <property name="font">
       <font>
        <family>Consolas</family>
        <pointsize>10</pointsize>
       </font>
      </property>
      <property name="styleSheet">
       <string notr="true">color: rgb(255, 58, 51);</string>
      </property>
      <property name="verticalScrollBarPolicy">
       <enum>Qt::ScrollBarAlwaysOn</enum>
      </property>
      <property name="sizeAdjustPolicy">
       <enum>QAbstractScrollArea::AdjustIgnored</enum>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QPushButton" name="saveButton">
      <property name="styleSheet">
       <string notr="true">
background-color: rgb(0, 0, 0);
color: rgb(255, 110, 110);</string>
      </property>
      <property name="text">
       <string>Save results</string>
      </property>
     </widget>
    </item>
```

```
   </layout>
  </widget>
  <widget class="QMenuBar" name="menubar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>902</width>
     <height>25</height>
    </rect>
   </property>
   <widget class="QMenu" name="menuInfo">
    <property name="title">
     <string>Info</string>
    </property>
    <addaction name="actionAbout_algorithm"/>
    <addaction name="actionInstruction"/>
   </widget>
   <addaction name="menuInfo"/>
  </widget>
  <widget class="QStatusBar" name="statusbar"/>
  <action name="actionAbout_algorithm">
   <property name="checkable">
    <bool>false</bool>
   </property>
   <property name="text">
    <string>About algorithm</string>
   </property>
  </action>
  <action name="actionInstruction">
   <property name="checkable">
    <bool>true</bool>
   </property>
   <property name="text">
    <string>Instruction</string>
   </property>
  </action>
 </widget>
 <resources/>
 <connections/>
</ui>
```