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»

«Статическое кодирование и декодирование текстового файла методами Хаффмана и Фано-Шеннона»

Студент гр. 8381 _____ Гоголев Е.Е.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Гоголев Е.Е.

Группа 8381

Тема работы: Статическое кодирование и декодирование текстового файла методами Хаффмана и Фано-Шеннона

Исходные данные: необходимо создать программу для генерации заданий с ответами к ним для проведения текущего контроля среди студентов.

Содержание пояснительной записки:
«Содержание», «Введение», «Задание», «Описание программы», «Текущий контроль», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Дата выдачи задания: 12.10.2019

Дата сдачи реферата: 26.12.2019

Дата защиты реферата: 26.12.2019

Студент Гоголев Е.Е.

Преподаватель Жангиров Т.Р.

АННОТАЦИЯ

В ходе выполнения курсовой работы была написана программа, которая может генерировать наборы вариантов для проведения текущего контроля несколькими способами, выводить их на экран или записывать в файл. Возможна настройка сложности заданий. Доступна генерация вариантов для кодирования и декодирования Хаффмана, а также кодирования и декодирования Фано-Шеннона. Доступен выбор количества генерируемых файлов.

SUMMARY

During the course of the course work, a program was written that can generate testing variants using various ways. Also it can be showed on the screen or in the file. There is ability exist to choose what number of variants to generate. Encoding and decoding Haffman and encoding and decoding Fano Shennon types of testing included.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы

Необходимо разработать программу для генерации заданий с ответами к ним для проведения текущего контроля среди студентов. Задания и ответы должны выводиться в файл в удобной форме: тексты заданий должны быть готовы для передачи студентам, проходящим ТК; ответы должны позволять удобную проверку правильности выполнения заданий.

Разработка программы велась на базе операционной системы Windows 10 в среде разработки QtCreator [1]. Для создания графической оболочки использовался редактор интерфейса в QtCreator и система сигналов-слотов Qt [2]. Для реализации текущего контроля был создан набор функций в файлах coding.h и coding.cpp.

1. ОПИСАНИЕ ПРОГРАММЫ

1.1. Описание интерфейса пользователя

Интерфейс программы [3] разделен на две части: левая панель ввода данных и генерации данных, правая панель вывода сгенерированного варианта. Основные виджеты и их назначение представлены в табл. 1 и табл. 2

Таблица 1 – Основные виджеты левой панели программы

Класс объекта	Название виджета	Назначение
QComboBox	type	Список выбора типа алгоритма
QComboBox	method	Список выбора кодирования/декодирования
QSpinBox	length	Окно выбора длины кодируемой строки
QPushButton	generate	Кнопка генерации варианта на экран
QSpinBox	variants	Окно выбора количество генерируемых вариантов
QPushButton	file	Кнопка генерации набора вариантов в файл

Таблица 2 – Основные виджеты правой панели программы

Класс объекта	Название виджета	Назначение
QTextBrowser	issue	Окно вывода задания
QTextBrowser	answer	Окно вывода ответа

1.2. Описание основных функций текущего контроля

Для реализации текущего контроля были созданы функции [4] для кодирования и декодирования методами Хаффмана и Фано-Шеннона. Также

был создан ряд функций для рандомизации входных данных. Данные функции описаны в табл. 3.

Таблица 3 – Основные функции файла coding.h

Название функции	Назначение
`generate_pack`	Генерирует набор вариантов и записывает отдельные части в строки
`generate_var`	Генерирует один вариант для вывода на экран
`encode_haffman` `encode_fano_shennon`	Соответственно кодирует строку методами Хаффмана и Фано-Шеннона
`write_to_file`	Записывает строку в указанный файл
`calc_freqs`	Считает частоту встречаемости символов в заданной строке. Требуется для обоих алгоритмов кодирования
`shannon`	Кодирует строку методом Фано-Шеннона

`NewNode`
`createMinHeap`
`swapMinHeapNode`
`minHeapify`
`extractMin`
`insertMinHeap`
`buildMinHeap`
`createAndBuildMinHeap`
`buildHuffmanTree`
	Вспомогательные функции для кодирования методом Хаффмана. Основаны на том, что для кодирования требуется сначала создать минимальную кучу, которая учитывает частоту каждого символа, после чего посчитать путь до каждого символа

`PrintArr` `printCodes`	Считывает код символа из массива, полученного проходом по минимальной куче
`HuffmanCodes`	Основной алгоритм кодирования Хаффмана

1.3. Описание алгоритма Хаффмана

Алгоритм Хаффмана[4] – жадный алгоритм оптимального префиксного кодирования алфавита с минимальной избыточностью.

Классический алгоритм Хаффмана на входе получает таблицу частот встречаемости символов в сообщении. Далее на основании этой таблицы строится дерево кодирования Хаффмана:

- Символы входного алфавита образуют список свободных узлов. Каждый лист имеет вес, который может быть равен либо вероятности, либо количеству вхождений символа в сжимаемое сообщение.
- Выбираются два свободных узла дерева с наименьшими весами.
- Создается их родитель с весом, равным их суммарному весу.
- Родитель добавляется в список свободных узлов, а два его потомка удаляются из этого списка.
- Одной дуге, выходящей из родителя, ставится в соответствие бит 1, другой — бит 0. Битовые значения ветвей, исходящих от корня, не зависят от весов потомков.
- 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.

1.4. Описание алгоритма Фано-Шеннона

Алгоритм Фано-Шеннона [5] использует коды переменной длины: часто встречающийся символ кодируется кодом меньшей длины, редко встречающийся – кодом большей длины. Этапы работы алгоритма:

- Символы первичного алфавита m1 выписывают по убыванию вероятностей.
- Символы полученного алфавита делят на две части, суммарные вероятности символов которых максимально близки друг другу.

- В префиксном коде для первой части алфавита присваивается двоичная цифра «0», второй части — «1».
- Полученные части рекурсивно делятся и их частям назначаются соответствующие двоичные цифры в префиксном коде.

1.5 Описание функций демонстрации текущего контроля

Все методы вывода [6] располагаются в классе mainwindow. Ниже перечислены основные из них представлены в табл. 4.

Таблица 4 – Основные методы вывода сгенерированных данных

Название функции	Описание функции
`on_type_currentIndexChanged`	Обновляет тип генерируемых вариантов
`on_method_currentIndexChanged`	Обновляет метод кодирования генерируемых вариантов
`on_generate_clicked`	Генерирует вариант на экран
`on_variants_valueChanged`	Обновляет количество генерируемых вариантов
`on_file_clicked`	Генерирует набор вариантов для текущего контроля в файлы issue.txt и answer.txt
`on_length_valueChanged`	Обновляет количество символов в генерируемых вариантах

2. ТЕКУЩИЙ КОНТРОЛЬ

2.1. Вид программы

Программа представляет собой окно с графическим интерфейсом. Вид программы после запуска представлен на рис. 1.

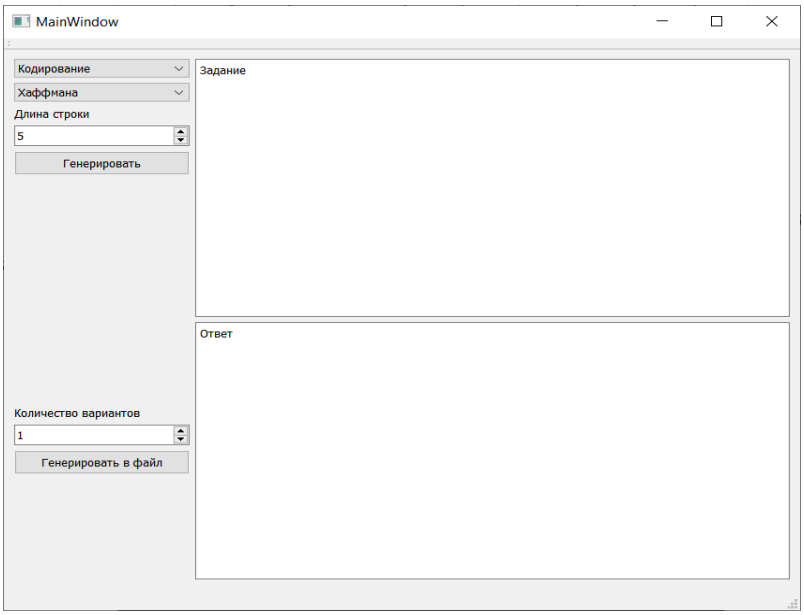

Рисунок 1 – Вид программы после запуска

На рис. 2 представлен сгенерированный вариант

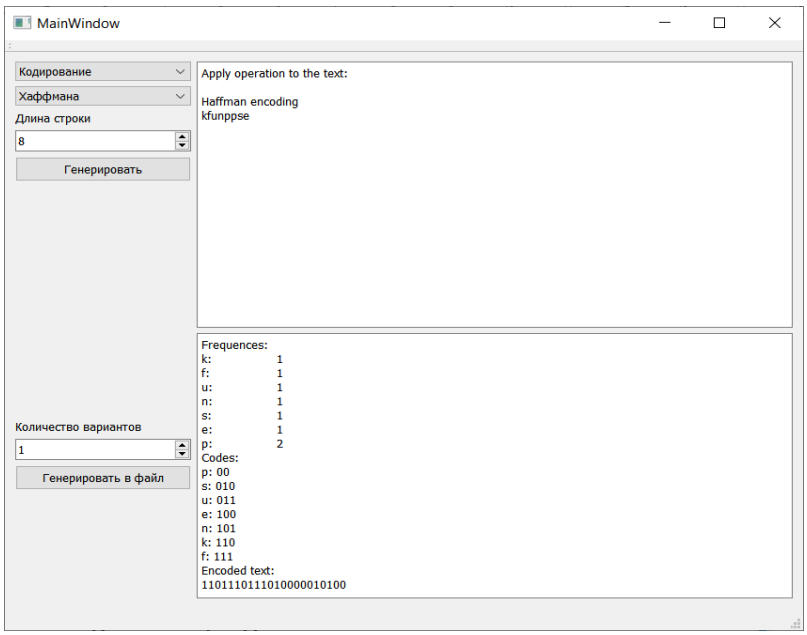

Рисунок 2 – Вид программы после генерации варианта

2.2. Генерация кодирования Хаффмана

После ввода количества символов производится нажатие кнопки «Генерировать» (по умолчанию длина строки равна 5), как представлено на рис. 3.

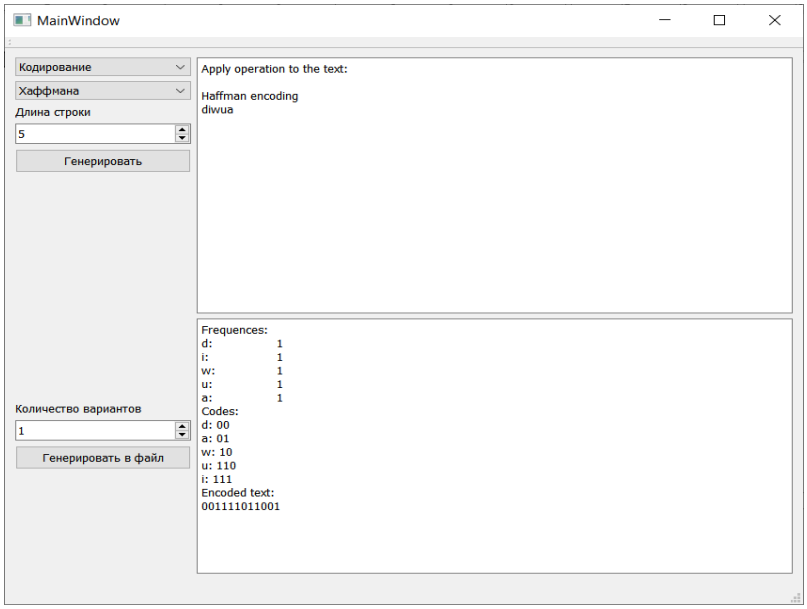

Рисунок 3 – Вид программы после генерации задания на кодирование Хаффмана

2.3. Генерация декодирования Хаффмана

На рис. 4 представлен сгенерированный вариант на декодирование Хаффмана.

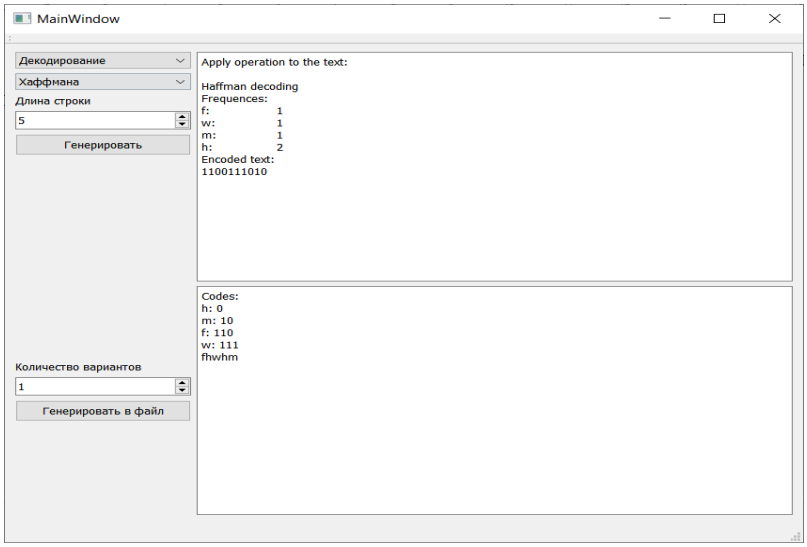

Рисунок 4 – Проход вниз при вставке элемента

2.4. Генерация кодирования Фано-Шеннона

На рис. 5 представлен сгенерированный вариант на кодирование Фано-Шеннона.

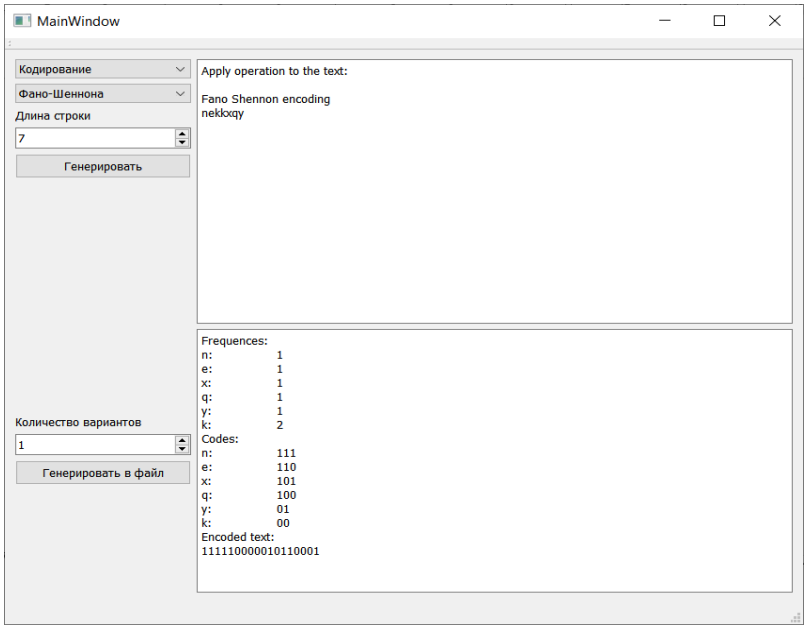

Рисунок 5 – Вид программы после генерации варианта на кодирования Фано-Шеннона

2.5. Генерация декодирования Фано-Шеннона

На рис. 6 представлен сгенерированный вариант на декодирование Фано-Шеннона.

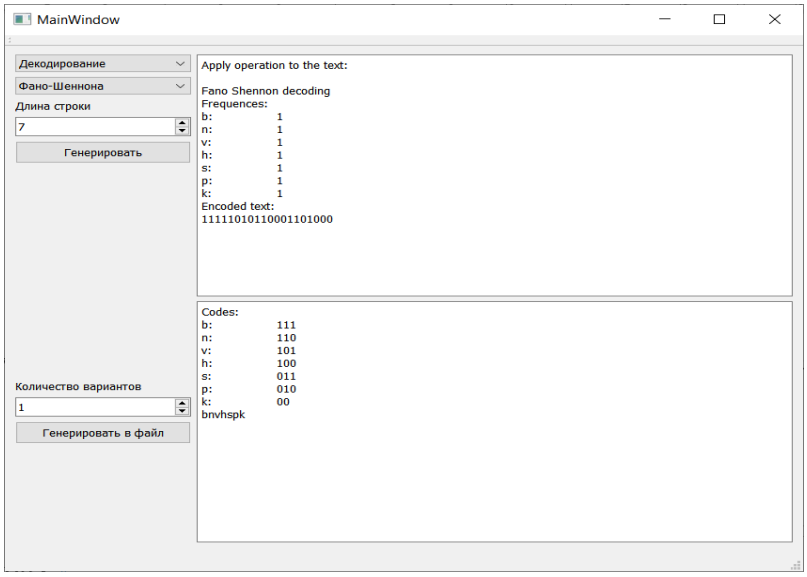

Рисунок 6 – Поиск требуемого элемента

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана программа, которая обладает следующей функциональностью: генерация вариантов для проведения текущего контроля по таким темам, как кодирование Хаффмана, декодирование Хаффмана, кодирование Фано-Шеннона и декодирование Фано-Шеннона; Запись сгенерированных вариантов возможна как в файл, так и на экран. Ко всем вариантам генерируется соответствующее решение. В ходе работы возникали сложности с необходимостью вывода кода каждого символа в алгоритме Хаффмана, необходимость в затирании файла при каждой перезаписи. Также возникали сложности с сохранением всего текста задания и ответа для каждого режима работы, что было решено использованием раздельных строк для хранения каждого типа данных, и конкатенации в требуемой комбинации после возврата из функции кодирования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Qt Documentation // Qt. URL: https://doc.qt.io/qt-5/index.html (дата обращения: 20.12.2019)
2. Перевод и дополнение документации QT // CrossPlatform.RU. URL: http://doc.crossplatform.ru/ (дата обращения: 20.12.2019)
3. Bjarne Stroustrup. A Tour of C++. M.: Addison-Wesley, 2018. 217 с.
4. Всё о сжатии данных, изображений и видео. // Compression URL: http://compression.ru/download/articles/huff/simakov_2002_huffcode.html (дата обращения: 20.12.2019)
5. Всё о сжатии данных, изображений и видео. // Compression URL: http://compression.ru/download/articles/huff/tiger_shannon-fano.html (дата обращения: 20.12.2019)
6. Макс Шлее. Qt5.10. Профессиональное программирование на C++. M.: BHV-СПб, 2018, 513 с.

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. MAINWINDOW.H

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

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

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

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

    types type;
    methods method;
    int variant_count;
    int length;

private slots:
    void on_type_currentIndexChanged(int index);

    void on_method_currentIndexChanged(int index);

    void on_generate_clicked();

    void on_variants_valueChanged(int arg1);

    void on_file_clicked();

    void on_length_valueChanged(int arg1);

    void on_hand_textChanged(const QString &arg1);

    void on_lr5_clicked();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ. CODING.H

```
#ifndef CODING_H
#define CODING_H

#include <iostream>
#include <fstream>
#include <string>
#include <algorithm>
#include <ctime>
#include <cstdlib>

using namespace std;

enum methods {
    haffman,
    fano_shennon
};

enum types {
    coding,
    decoding
};

void generate_pack(types t, methods m, int length, int count, string& issue,
string& answer);
void generate_var(types t, methods m, int length, string& issue, string&
answer);
void encode_haffman(string &input, string& res_freqs, string& res_codes,
string& res);
void encode_fano_shennon(string& input, string& res_freqs, string& res_codes,
string& res);
void write_to_file(string filename, string& data);

void calc_freqs(string& input, char chars[], int freqs[], int& count);

// Haffman

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

struct MinHeap {
    int size;
    int capacity;
    MinHeapNode** array;
```

```
};

MinHeapNode* newNode(char data, unsigned freq);
MinHeap* createMinHeap(unsigned capacity);
void swapMinHeapNode(MinHeapNode** a, MinHeapNode** b);
void minHeapify(MinHeap* minHeap, int idx);
MinHeapNode* extractMin(MinHeap* minHeap);
void insertMinHeap(MinHeap* minHeap, MinHeapNode* minHeapNode);
void buildMinHeap(MinHeap* minHeap);
MinHeap* createAndBuildMinHeap(char data[], int freq[], int size);
MinHeapNode* buildHuffmanTree(char data[], int freq[], int size);
void printArr(string& code, int arr[], int n);
void printCodes(string& result, char codes_chars[], string* codes, int& ind,
MinHeapNode* root, int arr[], int top);
void HuffmanCodes(string& result, char codes_chars[], string result_codes[],
char data[], int freq[], int size);

// Fano-Shennon

struct node {
    char sym; //char
    int pro; //freq
    int arr[20];
    int top;
    string code;
};

void shannon(int l, int h, node p[]);

#endif // CODING_H
```

ПРИЛОЖЕНИЕ Г

ИСХОДНЫЙ КОД ПРОГРАММЫ. CODING.CPP

```
#include "coding.h"

void generate_pack(types t, methods m, int length, int count, string &issue,
string &answer) {
    for (int i = 0; i < count; i++) {
        // TODO описание задания и форматирование строк, чтоб красиво
        string current = "Variant " + to_string(i) + "\n";
        string one_issue = current;
        string one_answer = current;
        generate_var(t, m, length, one_issue, one_answer);
        issue += one_issue + "\n----------------\n";
        answer += one_answer + "\n----------------\n";
    }
}

void generate_var(types t, methods m, int length, string &issue, string
&answer) {
    string current_issue = "";
    issue += "Apply operation to the text:\n";
    for (int i = 0; i < length; i++) {
        current_issue += 'a' + rand() % ('z' - 'a');
    }
    string res_freqs = "Frequences:\n";
    string res_codes = "Codes:\n";
    string res = "Encoded text:\n";
    switch (m) {
    case haffman:
        encode_haffman(current_issue, res_freqs, res_codes, res);
        switch (t) {
        case coding:
            issue += "\nHaffman encoding\n";
            issue += current_issue;
            answer += res_freqs + res_codes + res;
            break;
        case decoding:
            issue += "\nHaffman decoding\n";
            issue += res_freqs + res;
            answer += res_codes + current_issue;
            break;
        }
        break;
    case fano_shennon:
        encode_fano_shennon(current_issue, res_freqs, res_codes, res);
        switch (t) {
        case coding:
            issue += "\nFano Shennon encoding\n";
```

```
            issue += current_issue;
            answer += res_freqs + res_codes + res;
            break;
        case decoding:
            issue += "\nFano Shennon decoding\n";
            issue += res_freqs + res;
            answer += res_codes + current_issue;
            break;
        }
        break;
    }
}

void encode_haffman(string &input, string& res_freqs, string& res_codes,
string& res) {
    //Freqs
    char* chars = new char[input.length()];
    int* freqs = new int[input.length()];
    int count = 0;
    calc_freqs(input, chars, freqs, count);
    for (int i = 0; i < count; i++) {
        string s(1, chars[i]);
        res_freqs += s + ":\t" + to_string(freqs[i]) + "\n";
    }
    //Encoding
    string* result_codes = new string[count];
    char* result_chars = new char[count];
    HuffmanCodes(res_codes, result_chars, result_codes, chars, freqs, count);
    //Result
    for (int i = 0; i < input.length(); i++) {
        for (int k = 0; k < count; k++) {
            if (result_chars[k] == input[i]) {
                res += result_codes[k];
                break;
            }
        }
    }
}

void encode_fano_shennon(string &input, string& res_freqs, string& res_codes,
string& res) {
    //Freqs
    char* chars = new char[input.length()];
    int* freqs = new int[input.length()];
    int count = 0;
    calc_freqs(input, chars, freqs, count);
    node* nodes = new node[count];
    for (int i = 0; i < count; i++) {
        string s(1, chars[i]);
```

```
        res_freqs += s + ":\t" + to_string(freqs[i]) + "\n";
        nodes[i].sym = chars[i];
        nodes[i].pro = freqs[i];
        nodes[i].top = -1;
    }
    //Encoding
    shannon(0, count - 1, nodes);
    for (int i = 0; i < count; i++) {
        string s(1, nodes[i].sym);
        res_codes += s + ":\t";
               for (int j = 0; j <= nodes[i].top; j++) nodes[i].code +=
to_string(nodes[i].arr[j]);
        res_codes += nodes[i].code + "\n";
    }
    //Result
    for (int i = 0; i < input.length(); i++) {
        for (int k = 0; k < count; k++) {
            if (nodes[k].sym == input[i]) {
                res += nodes[k].code;
                break;
            }
        }
    }
}

void write_to_file(string filename, string &data) {
    cout << data;
    cout << &data;
    ofstream myfile;
    myfile.open(filename);
    myfile << data;
    myfile.close();
}

void calc_freqs(string &input, char chars[], int freqs[], int& count)
{
    for (int i = 0; i < input.length(); i++) {
        int k;
        for (k = 0; k < count; k++) {
            if (chars[k] == input[i]) {
                freqs[k]++;
                break;
            }
        }
        if (k == count) {
            chars[count] = input[i];
            freqs[count] = 1;
            count++;
        }
```

```
    }
    //bubble
    for (int i = 0; i < count - 1; i++) {
        for (int k = 0; k < count - i - 1; k++) {
            if (freqs[k] > freqs[k+1]) {
                int temp = freqs[k];
                freqs[k] = freqs[k+1];
                freqs[k+1] = temp;
                char temp2 = chars[k];
                chars[k] = chars[k+1];
                chars[k+1] = temp2;
            }
        }
    }
}


// Haffman encoding

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

MinHeap* createMinHeap(unsigned capacity) {
    MinHeap* minHeap = new MinHeap;
    minHeap->size = 0;
    minHeap->capacity = capacity;
    minHeap->array = new MinHeapNode*[minHeap->capacity];
    return minHeap;
}

void swapMinHeapNode(MinHeapNode** a, MinHeapNode** b) {
    MinHeapNode* t = *a;
    *a = *b;
    *b = t;
}

void minHeapify(MinHeap* minHeap, int idx) {
    int smallest = idx;
    int left = 2 * idx + 1;
    int right = 2 * idx + 2;

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

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

    if (smallest != idx) {
        swapMinHeapNode(&minHeap->array[smallest], &minHeap->array[idx]);
        minHeapify(minHeap, smallest);
    }
}

MinHeapNode* extractMin(MinHeap* minHeap) {
    MinHeapNode* temp = minHeap->array[0];
    minHeap->array[0] = minHeap->array[minHeap->size - 1];
    --minHeap->size;
    minHeapify(minHeap, 0);
    return temp;
}

void insertMinHeap(MinHeap* minHeap, MinHeapNode* minHeapNode) {
    ++minHeap->size;
    int i = minHeap->size - 1;
    while (i && minHeapNode->freq < minHeap->array[(i - 1) / 2]->freq) {
        minHeap->array[i] = minHeap->array[(i - 1) / 2];
        i = (i - 1) / 2;
    }
    minHeap->array[i] = minHeapNode;
}

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

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

MinHeapNode* buildHuffmanTree(char data[], int freq[], int size) {
    MinHeapNode *left, *right, *top;
    MinHeap* minHeap = createAndBuildMinHeap(data, freq, size);
    while (minHeap->size != 1) {
        left = extractMin(minHeap);
```

```
        right = extractMin(minHeap);
        // '$' is a special value for internal nodes, not used
        top = newNode('$', left->freq + right->freq);
        top->left = left;
        top->right = right;
        insertMinHeap(minHeap, top);
    }
    return extractMin(minHeap);
}

void printArr(string& code, int arr[], int n) {
    for (int i = 0; i < n; ++i) code += to_string(arr[i]);
}

void printCodes(string& result, char codes_chars[], string* codes, int& ind,
MinHeapNode* root, int arr[], int top) {
    if (root->left) {
        arr[top] = 0;
        printCodes(result, codes_chars, codes, ind, root->left, arr, top + 1);
    }
    if (root->right) {
        arr[top] = 1;
        printCodes(result, codes_chars, codes, ind, root->right, arr, top + 1);
    }
    if (!(root->left) && !(root->right)) {
        string s(1, root->data);
        result += s + ": ";
        codes_chars[ind] = root->data;
        printArr(codes[ind], arr, top);
        result += codes[ind] + "\n";
        ind++;
    }
}

void HuffmanCodes(string& result, char result_chars[], string result_codes[],
char data[], int freq[], int size) {
    MinHeapNode* root = buildHuffmanTree(data, freq, size);
    int arr[256];
    int top = 0;
    int ind = 0;
    printCodes(result, result_chars, result_codes, ind, root, arr, top);
}

// Fano-Shennon

void shannon(int l, int h, node p[]) {
    int pack1 = 0, pack2 = 0, diff1 = 0, diff2 = 0;
    int i, d, k, j;
    if ((l + 1) == h || l == h || l > h) {
```

```
        if (l == h || l > h)
            return;
        p[h].arr[++(p[h].top)] = 0;
        p[l].arr[++(p[l].top)] = 1;
        return;
    }
    else {
        for (i = l; i <= h - 1; i++) pack1 = pack1 + p[i].pro;
        pack2 = pack2 + p[h].pro;
        diff1 = pack1 - pack2;
        if (diff1 < 0) diff1 = diff1 * -1;
        j = 2;
        while (j != h - l + 1) {
            k = h - j;
            pack1 = pack2 = 0;
            for (i = l; i <= k; i++) pack1 = pack1 + p[i].pro;
            for (i = h; i > k; i--) pack2 = pack2 + p[i].pro;
            diff2 = pack1 - pack2;
            if (diff2 < 0) diff2 = diff2 * -1;
            if (diff2 >= diff1) break;
            diff1 = diff2;
            j++;
        }
        k++;
        for (i = l; i <= k; i++) p[i].arr[++(p[i].top)] = 1;
        for (i = k + 1; i <= h; i++) p[i].arr[++(p[i].top)] = 0;

        shannon(l, k, p);
        shannon(k + 1, h, p);
    }
}
```

ПРИЛОЖЕНИЕ Д

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.CPP

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

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    srand(time(nullptr));
    type = coding;
    method = haffman;
    variant_count = 1;
    length = 5;
}

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

void MainWindow::on_type_currentIndexChanged(int index)
{
    type = (types)index;
}

void MainWindow::on_method_currentIndexChanged(int index)
{
    method = (methods)index;
}

void MainWindow::on_generate_clicked()
{
    string issue;
    string answer;
    generate_var(type, method, length, issue, answer);
    ui->issue->setText(QString::fromStdString(issue));
    ui->answer->setText(QString::fromStdString(answer));
}

void MainWindow::on_variants_valueChanged(int arg1)
{
    variant_count = arg1;
}

void MainWindow::on_length_valueChanged(int arg1)
{
```

```
    length = arg1;
}

void MainWindow::on_file_clicked()
{
    string issue = "";
    string answer = "";
    generate_pack(type, method, length, variant_count, issue, answer);
    write_to_file("issue.txt", issue);
    write_to_file("answer.txt", answer);
}


void MainWindow::on_hand_textChanged(const QString &arg1)
{

}

void MainWindow::on_lr5_clicked()
{

}
```

ПРИЛОЖЕНИЕ Е

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>849</width>
    <height>670</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <layout class="QHBoxLayout" name="horizontalLayout" stretch="1,4">
    <item>
     <layout class="QVBoxLayout" name="verticalLayout">
      <item>
       <widget class="QComboBox" name="type">
        <item>
         <property name="text">
          <string>Кодирование</string>
         </property>
        </item>
        <item>
         <property name="text">
          <string>Декодирование</string>
         </property>
        </item>
       </widget>
      </item>
      <item>
       <widget class="QComboBox" name="method">
        <item>
         <property name="text">
          <string>Хаффмана</string>
         </property>
        </item>
        <item>
         <property name="text">
          <string>Фано-Шеннона</string>
         </property>
        </item>
       </widget>
```

```
</item>
<item>
 <widget class="QLabel" name="label_2">
  <property name="text">
   <string>Длина строки</string>
  </property>
 </widget>
</item>
<item>
 <widget class="QSpinBox" name="length">
  <property name="minimum">
   <number>2</number>
  </property>
  <property name="maximum">
   <number>100</number>
  </property>
  <property name="value">
   <number>5</number>
  </property>
 </widget>
</item>
<item>
 <widget class="QPushButton" name="generate">
  <property name="text">
   <string>Генерировать</string>
  </property>
 </widget>
</item>
<item>
 <spacer name="verticalSpacer_3">
  <property name="orientation">
   <enum>Qt::Vertical</enum>
  </property>
  <property name="sizeHint" stdset="0">
   <size>
    <width>20</width>
    <height>40</height>
   </size>
  </property>
 </spacer>
</item>
<item>
 <widget class="QLabel" name="label_3">
  <property name="maximumSize">
   <size>
    <width>16777215</width>
    <height>0</height>
   </size>
  </property>
```

```
  <property name="text">
   <string>Ручной ввод</string>
  </property>
 </widget>
</item>
<item>
 <widget class="QLineEdit" name="hand">
  <property name="maximumSize">
   <size>
    <width>16777215</width>
    <height>0</height>
   </size>
  </property>
 </widget>
</item>
<item>
 <widget class="QPushButton" name="lr5">
  <property name="maximumSize">
   <size>
    <width>16777215</width>
    <height>0</height>
   </size>
  </property>
  <property name="text">
   <string>Декодировать Фано-Шеннона</string>
  </property>
 </widget>
</item>
<item>
 <spacer name="verticalSpacer_2">
  <property name="orientation">
   <enum>Qt::Vertical</enum>
  </property>
  <property name="sizeHint" stdset="0">
   <size>
    <width>20</width>
    <height>20</height>
   </size>
  </property>
 </spacer>
</item>
<item>
 <widget class="QLabel" name="label">
  <property name="sizePolicy">
   <sizepolicy hsizetype="Preferred" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
```

```
    <property name="text">
     <string>Количество вариантов</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QSpinBox" name="variants">
    <property name="sizePolicy">
     <sizepolicy hsizetype="Preferred" vsizetype="Fixed">
      <horstretch>0</horstretch>
      <verstretch>0</verstretch>
     </sizepolicy>
    </property>
    <property name="minimum">
     <number>1</number>
    </property>
    <property name="maximum">
     <number>1000</number>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QPushButton" name="file">
    <property name="text">
     <string>Генерировать в файл</string>
    </property>
   </widget>
  </item>
  <item>
   <spacer name="verticalSpacer">
    <property name="orientation">
     <enum>Qt::Vertical</enum>
    </property>
    <property name="sizeHint" stdset="0">
     <size>
      <width>20</width>
      <height>40</height>
     </size>
    </property>
   </spacer>
  </item>
 </layout>
</item>
<item>
 <layout class="QVBoxLayout" name="verticalLayout_2">
  <item>
   <widget class="QTextBrowser" name="issue">
    <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;
content=&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;Задание&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;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QTextBrowser" name="answer">
        <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;
content=&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;Ответ&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
        </property>
       </widget>
      </item>
     </layout>
    </item>
   </layout>
  </widget>
  <widget class="QMenuBar" name="menuBar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>849</width>
     <height>25</height>
    </rect>
   </property>
  </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>
```

ПРИЛОЖЕНИЕ Ж

ИСХОДНЫЙ КОД ПРОГРАММЫ. GOGOLEV_CW.PRO

```
#-------------------------------------------------
#
# Project created by QtCreator 2019-12-20T20:45:05
#
#-------------------------------------------------

QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = gogolev_cw
TEMPLATE = app

# The following define makes your compiler emit warnings if you use
# any feature of Qt which has been marked as 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 you use 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

CONFIG += c++11

SOURCES += \
        main.cpp \
        mainwindow.cpp \
    coding.cpp

HEADERS += \
        mainwindow.h \
    coding.h

FORMS += \
        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
```