

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

КУРСОВАЯ РАБОТА
по дисциплине «Алгоритмы и структуры данных»
Тема: Динамическое кодирование и декодирование по Хаффману

Студент гр. 7383

Зуев Д.В.

Преподаватель

Размочаева Н.В.

Санкт-Петербург

2018

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Зуев Д.В.

Группа 7383

Тема работы: Динамическое кодирование и декодирование по Хаффману

Исходные данные:

Написать программу, генерирующую задания по динамическому кодированию и декодированию по Хаффману

Содержание пояснительной записки:

- Содержание
- Введение
- Формулировка задачи
- Теоретические сведения
- Реализация задачи
- Тестирование
- Заключение
- Приложение А. Исходный код программы

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент

Зуев Д.В.

Преподаватель

Размочаева Н.В.

АННОТАЦИЯ

В данной курсовой работе была разработана программа на языке программирования C++, генерирующая несколько вариантов заданий на динамическое кодирование или декодирование по Хаффману. В данной программе пользователь должен вводить исходные данные с помощью графического интерфейса. Результаты и примеры работы программы представлены в разделе Тестирование.

SUMMARY

In this course work, a program was developed in the C++ programming language, generating several variants of tasks for dynamic coding or decoding by Huffman. In this program, the user must enter the original data using the graphical interface. The results and examples of the program are presented in the Testing section.

СОДЕРЖАНИЕ

	Введение	5
1.	Формулировка задачи	6
2.	Теоретические сведения	7
3.	Реализация задачи	8
3.1.	Структура bin_tree	8
3.2.	Функция create_node	8
3.3	Функция path_redefenition	8
3.4	Функция comparision	8
3.5	Функция get_new	8
3.6	Функция code	8
3.7	Функция generate	9
4.	Тестирование	10
	Заключение	15
	Список использованных источников	16
	Приложение А. Исходный код программы	17

ВВЕДЕНИЕ

Целью работы является освоение на практике алгоритма динамического кодирования и декодирования по Хаффману.

Задачей работы является разработка программы на языке C++, генерирующей задания и ответы на динамическое кодирование или декодирование по Хаффману в отдельный файл, и создание в среде разработки Qt графического интерфейса, с помощью которой пользователь должен взаимодействовать с программой.

1. ФОРМУЛИРОВКА ЗАДАЧИ

Написать программу на языке C++, которая генерирует несколько вариантов заданий на динамическое кодирование или декодирование и записывает задания и ответы к ним в отдельные текстовые файлы. Обеспечить с помощью графического интерфейса ввод входных данных, таких как длина слова, максимальное количество используемых символов, выбор опции закодировать или декодировать.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Динамическое кодирование Хаффмана — адаптивный метод, основанный на кодировании Хаффмана. Он позволяет строить кодовую схему в поточном режиме (без предварительного сканирования данных), не имея никаких начальных знаний из исходного распределения, что позволяет за один проход сжать данные. Преимуществом этого способа является возможность кодировать на лету.

В данной работе метод был реализован с помощью ФГК (Фоллер, Галлагер и Кнут) алгоритма.

Он позволяет динамически регулировать дерево Хаффмана, не имея начальных частот. В ФГК дереве Хаффмана есть особый внешний узел, называемый *0-узел*, используемый для идентификации входящих символов. То есть, всякий раз, когда встречается новый символ — его путь в дереве начинается с нулевого узла. Самое важное — то, что нужно усекать и балансировать ФГК дерево Хаффмана при необходимости, и обновлять частоту связанных узлов. Как только частота символа увеличивается, частота всех его родителей должна быть тоже увеличена. Это достигается путём последовательной перестановки узлов, поддеревьев или и тех и других.

Важной особенностью ФГК дерева является принцип братства (или соперничества): каждый узел имеет два потомка (узлы без потомков называются листьями) и веса идут в порядке убывания. Благодаря этому свойству дерево можно хранить в обычном массиве, что увеличивает производительность.

3. РЕАЛИЗАЦИЯ ЗАДАЧИ

В данной работе была разработана программа, которая генерирует строку, кодирует ее с помощью функций, описанных ниже, записывает строку и код в отдельные файлы, в зависимости от опции, и повторяющая это действие несколько раз, для каждого варианта.

3.1. Структура `bin_tree`

В данной структуре содержатся данные об узле дерева, такие как указатели на сыновей и родителя, вес узла, символ, и код пути от корня до данного узла.

3.2. Функция `create_node`

Для данного листа создает правое и левое поддеревья, в левое поддерево записывает терминальный символ, в правое поддерево записывает символ, который подан на вход функции.

3.3. Функция `path_redefenition`

Проходит от узла до корня и строит код пути от корня до этого узла.

3.4. Функция `comparision`

Проходит по массиву узлов дерева, сравнивает веса соседних элементов и, если необходимо, перевешивает узлы в дереве так, чтобы данный массив был отсортирован по неубыванию весов узлов.

3.5. Функция `get_new`

Создает новый лист с помощью функции `create_node` и добавляет его в дерево и в массив, обновляя пути и порядок элементов в массиве с помощью функций `path_redefenition` и `comparision` соответственно.

3.5. Функция `code`

Кодирует поступающую на вход строку из символов, являющихся латинскими буквами в нижнем регистре, с помощью динамического алгоритма Хаффмана. Если считанный символ уже встречался, то вес узла, в котором находится символ, увеличивается на один, иначе вызывается функция `get_new`. После этого производится балансировка дерева с помощью функции

comparision и переопределение путей от корня до узлов с помощью функции path_redefenition.

3.5. Функция generate

Генерирует строку заданной длины из диапазона символов заданного максимальным числом используемых символов.

Код программы представлен в приложении А.

3. ТЕСТИРОВАНИЕ

Тестирование программы проводилось в операционной системе Linux Ubuntu с помощью среды разработки Qt.

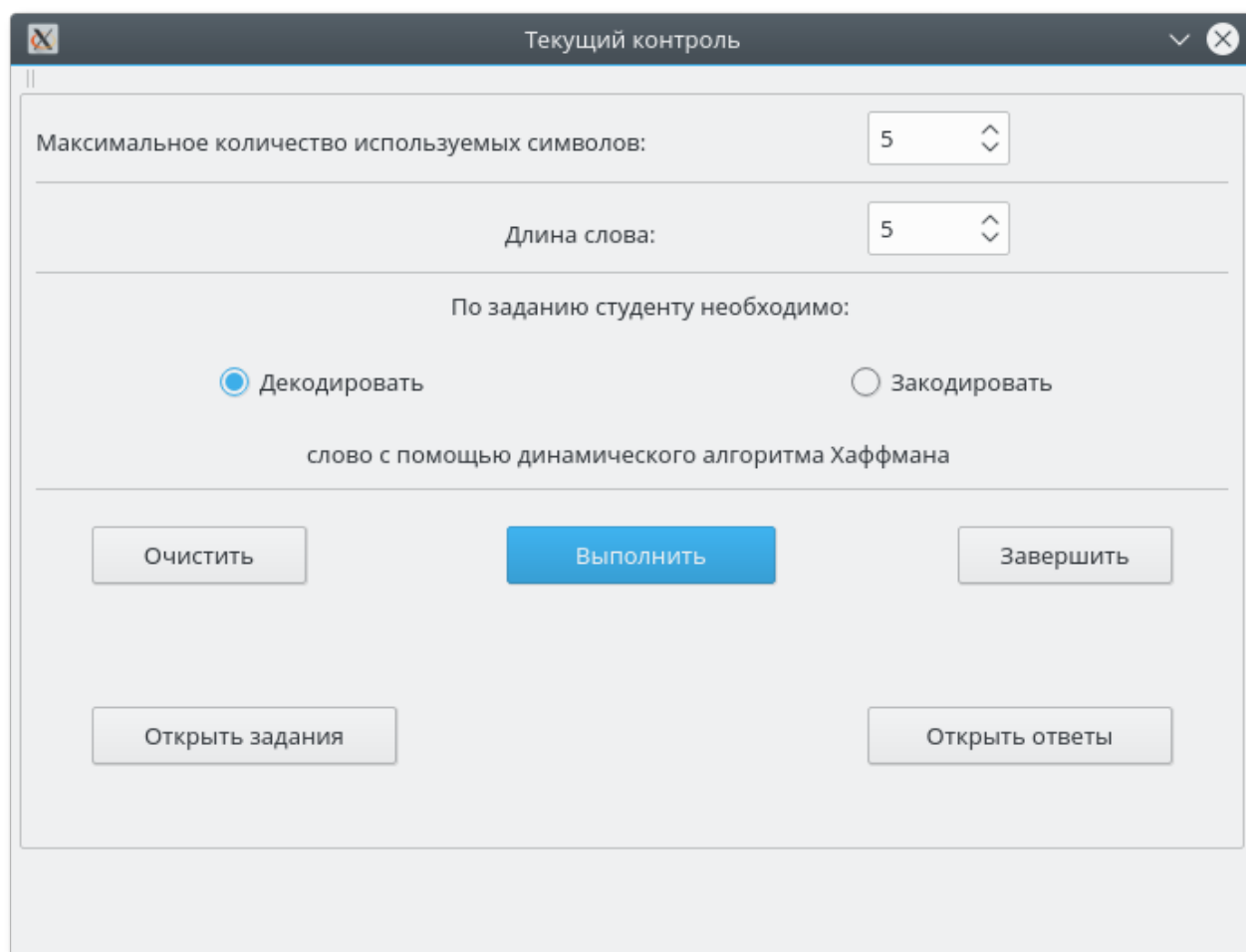


Рисунок 1 - Пользовательский интерфейс

При запуске программы появляется окно, представленное на рис. 1. Это окно является пользовательским интерфейсом, с помощью которого вводятся входные данные.

Пользователь имеет возможность выбрать максимальное количество используемых символов ограниченное от 1 до 26 — количество букв в английском алфавите, длину слова ограниченную от 1 до 22 и то что студенту или иному лицу необходимо сделать по заданию — закодировать слово или декодировать. Данные опции изначально установлены по умолчанию:

максимальное количество используемых символов равно 5, длина слова равна 5, действие — закодировать..

Пример выбора опций представлен на рис. 2.

The screenshot shows a window titled "Текущий контроль" (Current Control). Inside, there are several configuration options:

- "Максимальное количество используемых символов:" (Maximum number of symbols used) with a value of 9.
- "Длина слова:" (Word length) with a value of 22.
- A section titled "По заданию студенту необходимо:" (According to the assignment, the student needs to:) with two radio buttons: "Декодировать" (Decode) and "Закодировать" (Encode). The "Закодировать" option is selected.
- Below the radio buttons, the text "слово с помощью динамического алгоритма Хаффмана" (word using the dynamic Huffman algorithm) is displayed.
- At the bottom, there are five buttons: "Очистить" (Clear), "Выполнить" (Execute), "Завершить" (Finish), "Открыть задания" (Open tasks), and "Открыть ответы" (Open answers).

Рисунок 2 - Выбор опций

После выбора необходимых опций нужно нажать кнопку «Выполнить», что запускает генерацию слов и их кодов и записывает это все в текстовые файлы с заданиями и ответами.

С помощью кнопок «Открыть задания» и «Открыть ответы» пользователь может открыть текстовые файлы с заданиями и ответами соответственно.

Содержимое файла с заданиями task.txt представлено на рис. 3.

```
task.txt
~/programming/TA/build-COURSEWORK-Desktop-Debug
Открыть Сохранить
Задание на кодирование динамическим алгоритмом Хаффмана

Вариант 1
Задание №1
Используемые символы:
b - 01100010
c - 01100011
d - 01100100
e - 01100101
f - 01100110
g - 01100111
i - 01101001
dgbefgcceiefibecbcgdgg;

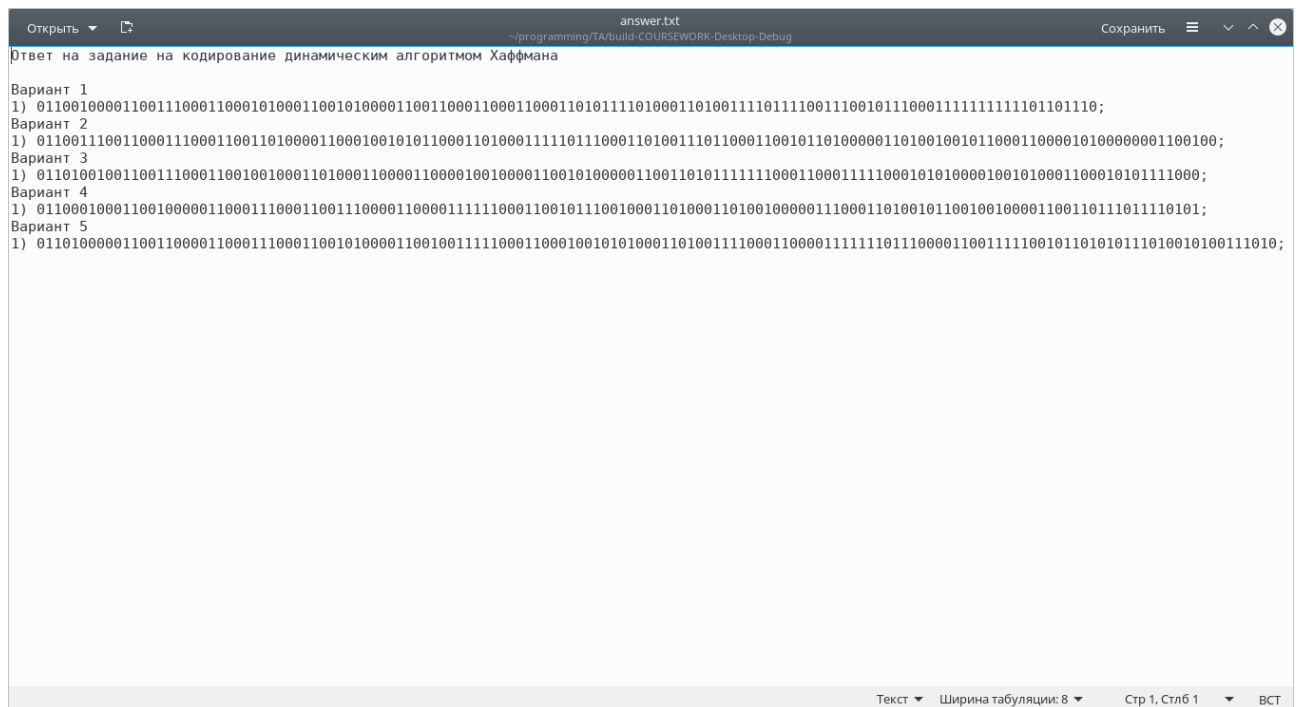
Вариант 2
Задание №1
Используемые символы:
a - 01100001
b - 01100010
c - 01100011
d - 01100100
e - 01100101
f - 01100110
g - 01100111
h - 01101000
i - 01101001
gcfgbbghcbigefhfhaccd;

Вариант 3
Задание №1
Используемые символы:
a - 01100001
b - 01100010
c - 01100011
d - 01100100
e - 01100101
f - 01100110
- 01100111

Текст Ширина табуляции: 8 Стр 1, Стлб 1 ВСТ
```

Рисунок 3 - Файл с заданиями

Содержимое файла answer.txt с ответами к представленным выше заданиям представлено на рис. 4.



```
answer.txt
~/programming/TA/build-COURSEWORK-Desktop-Debug

Ответ на задание на кодирование динамическим алгоритмом Хаффмана

Вариант 1
1) 011001000011001110001100010100011001010000110011000110001100111101000110100111101111001110010111000111111111101101110;
Вариант 2
1) 011001110011000111000110011010000110001001010110001101000111110111000110100111011000110010110100000110100100110001100000001100100;
Вариант 3
1) 011010010011001110001100100100011010001100001001000011001010000011001101011111100011000111110001010100001001010001100010101111000;
Вариант 4
1) 0110001000110010000011000111000110011100001100001111110001100101110010001101000110100100000111000110100100100100001100110111011110101;
Вариант 5
1) 011010000011001100001100011100011001010000110010011111000110001001010100011010011110001100001111110111000011001111100101101010111010010100111010;
```

Рисунок 4 - Файл с ответами

Нажатие кнопки «Очистить» возвращает все опции к начальным значениям и очищает текстовые файлы с заданиями и ответами.

Вид окна после нажатия кнопки «Очистить» представлен на рис. 5.

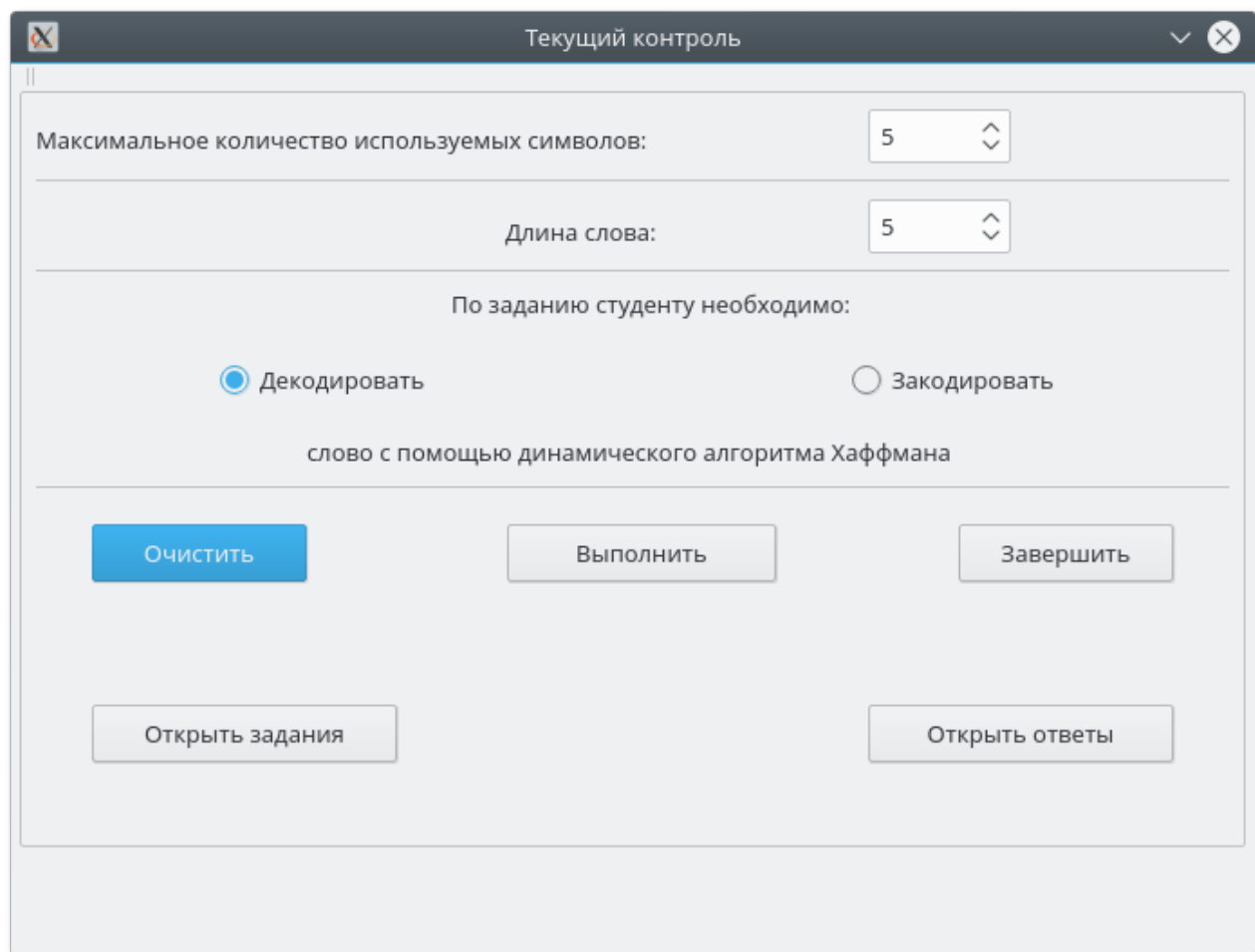


Рисунок 5 - Откат к начальному состоянию

Нажатие кнопки «Завершить» завершает выполнение программы.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была реализована программа на языке C++ для генерации заданий и ответов к ним на динамическое кодирование и декодирование по Хаффману. Для взаимодействия с программой был реализован графический интерфейс. По результатам тестирования видно, что поставленная задача выполнена верно

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Адаптивный алгоритм Хаффмана — Википедия // Википедия.
URL: https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0%D0%BF%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A5%D0%B0%D1%84%D1%84%D0%BC%D0%B0%D0%BD%D0%B0

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

main.cpp:

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

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

    return a.exec();
}
```

haffman_dinamic.cpp:

```
#include <iostream>
#include <cmath>
#include <cstring>
#include <sstream>
#include <fstream>
#include "haffman_dinamic.hpp"

using namespace std;

bin_tree* create()
{
    bin_tree* bt;
    bt = new bin_tree;
    bt->parent = NULL;
    bt->left = NULL;
    bt->right = NULL;
    bt->weight = 0;
    bt->el = '\n';
    bt->path = "";
    return bt;
}

bin_tree* create_node(bin_tree* btnul, char x)
{
    bin_tree* bt_left;
    bt_left = create();
    bt_left->parent = btnul;

    bin_tree* bt_right;
    bt_right = create();
    bt_right->el = x;
    bt_right->parent = btnul;

    btnul->left = bt_left;
    btnul->right = bt_right;
    btnul->right->parent = btnul;
    btnul->el = char(0);
    return btnul;
}
```

```

}
string path_redefenition(bin_tree* bt)
{
    string a;
    if(bt->parent == NULL)
        return "";
    a = path_redefenition(bt->parent);
    if(bt->parent->left == bt)
        a = a + '0';
    else
        a = a + '1';
    return a;
}
string char_to_bin(char x)
{
    int a = (int)x;
    string bin;
    while(a)
    {
        char b;
        b = '0' + a%2;
        a = a/2;
        bin = b + bin;
    }
    bin = '0' + bin;
    return bin;
}
bin_tree** comparison(bin_tree* list[], int& size, bin_tree* bt)
{
    bt->weight+=1;
    for(int i = 0; i<size-2;i++)
    {
        if(list[i]->weight > list[i+1]->weight)
        {
            int j = i+1;
            while(list[i]->weight > list[j]->weight)
                j++;
            j--;
            bin_tree* tmpl;
            bin_tree* tmpr;
            int tmpi;
            char tmpel;
            if(list[i]->parent == list[j])
                return comparison(list, size, list[j]);
            tmpl = list[j]->left;
            tmpr = list[j]->right;
            tmpi = list[j]->weight;
            tmpel = list[j]->el;
            list[j]->left = list[i]->left;
            list[j]->right = list[i]->right;
            list[j]->weight = list[i]->weight;
            list[j]->el = list[i]->el;
            list[i]->left = tmpl;
            list[i]->right = tmpr;
            list[i]->weight = tmpi;
            list[i]->el = tmpel;
            if(list[i]->left != NULL)
                list[i]->left->parent = list[i];
            if(list[j]->left != NULL)
                list[j]->left->parent = list[j];
            if(list[i]->right != NULL)

```

```

        list[i]->right->parent = list[i];
        if(list[j]->right != NULL)
            list[j]->right->parent = list[j];
        return comparison(list, size, list[j]->parent);
    }
}
if(bt->parent!=NULL)
{

    list = comparison(list, size, bt->parent);
}
return list;
}
bin_tree** get_new(char a, bin_tree* btnul, bin_tree* list[], int &size)
{
    btnul = create_node(btnul, char(a));

    bin_tree** list1;
    list1 = new bin_tree*[size+2];
    list1[0] = btnul->left;
    list1[1] = btnul->right;
    size+=2;
    for(int i = 2; i<size; i++)
    {
        list1[i] = list[i-2];
    }
    return comparison(list1, size, btnul->right);
}
string code(stringstream& xstream, string* accordance)
{
    string a;
    int j = 0;
    string bin_code;
    char x;
    bin_tree** list;
    list = new bin_tree*[1];
    int size = 1;
    bin_tree* bt;
    bt = create();
    bt->parent = NULL;
    list[0] = bt;
    while(xstream>>x)
    {
        int i;
        for(i = size-1; i>=0; i--)
        {
            if(list[i]->el == x)
            {
                i--;
                break;
            }
        }
        i++;
        a+=list[i]->path;
        if(list[i]->el == '\n')
        {
            bin_code = char_to_bin(x);
            accordance[j]+=x;
            accordance[j]+=" - ";
            accordance[j]+=bin_code;
            j++;
        }
    }
}

```

```

        a+=bin_code;
        list = get_new(x,list[i],list,size);
    }
    else
    {
        list = comparison(list, size, list[i]);
    }
    for(i = 0;i<size;i++)
        list[i]->path = path_redefenition(list[i]);
    }
    return a;
}
string generate(int count, int length)
{
    string str;
    int a;
    char c;
    for(int i = 0;i<length;i++)
    {
        a = rand()%count;
        c = 'a'+a;
        str+=c;
    }
    cout<<str<<endl;
    return str;
}

```

haffman_dinamic.hpp:

```

#include <cstring>

#ifndef HAFFMAN_DINAMIC_HPP
#define HAFFMAN_DINAMIC_HPP

using namespace std;

struct bin_tree{
    bin_tree* parent;
    bin_tree* left;
    bin_tree* right;
    int weight;
    char el;
    string path;
};

bin_tree* create();
bin_tree* create_node(bin_tree* btnul,char x);
string path_redefenition(bin_tree* bt);
bin_tree** comparison(bin_tree* list[], int& size, bin_tree* bt);
bin_tree** get_new(char a, bin_tree* btnul, bin_tree* list[], int &size);
string code(stringstream& xstream, string* accordance);
string generate(int count, int length);

#endif // HAFFMAN_DINAMIC_HPP

```

mainwindow.cpp:

```

#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "haffman_dinamic.hpp"
#include <random>
#include <fstream>

```

```

#include <cstdio>
#include <cstring>
#include <sstream>
#include <unistd.h>

using namespace std;

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    count = 5;
    length = 5;
    tmp = 2;
}
MainWindow::~MainWindow()
{
    delete ui;
}
void MainWindow::on_pushButton_3_clicked()
{
    exit(0);
}
void MainWindow::on_pushButton_clicked()
{
    srand(time(NULL));
    ofstream task;
    task.open("task.txt");
    ofstream answer;
    answer.open("answer.txt");
    stringstream xstream;
    switch(tmp){
    case 1:
    {
        task<<"Задание на кодирование динамическим алгоритмом
Хаффмана"<<endl<<endl;
        answer<<"Ответ на задание на кодирование динамическим алгоритмом
Хаффмана"<<endl<<endl;
        for(int i = 1; i<=5; i++)
        {
            task<<"Вариант " <<i<<endl;
            answer<<"Вариант " <<i<<endl;
            for(int j = 1; j<=4;j++)
            {
                xstream.str("");
                xstream.clear();
                string str_task;
                string str_answer;
                string* accordance;
                accordance = new string[26];
                task<<"Задание №" <<j<<endl;
                answer<<j<<" ";
                task<<"Используемые символы:"<<endl;
                str_task = generate(count, length);
                xstream << str_task;
                str_answer = code(xstream, accordance);
                for(j = 0; j<26;j++)
                    for(int i = 0;i<26;i++)
                        if(char('a'+j) == accordance[i][0])
                            task<<accordance[i]<<endl;
            }
        }
    }
}

```

```

        task<<str_task<<';'<<endl;
        answer<<str_answer<<';'<<endl;
    }
    task<<endl;
}
break;
}
case 2:
{
    task<<"Задание на декодирование динамическим алгоритмом
Хаффмана"<<endl<<endl;
    answer<<"Ответ на задание на декодирование динамическим алгоритмом
Хаффмана"<<endl<<endl;
    for(int i = 1; i<=5; i++)
    {
        task<<"Вариант "<<i<<endl;
        answer<<"Вариант "<<i<<endl;
        for(int j = 1; j<=4;j++)
        {
            xstream.str("");
            xstream.clear();
            string str_task;
            string str_answer;
            string* accordance;
            accordance = new string[26];
            task<<"Задание №"<<j<<endl;
            answer<<j<<" ";
            task<<"Используемые символы:"<<endl;
            str_task = generate(count, length);
            xstream << str_task;
            str_answer = code(xstream, accordance);
            for(j = 0; j<26;j++)
                for(int i = 0;i<26;i++)
                {
                    if(char('a'+j) == accordance[i][0])
                        task<<accordance[i]<<endl;
                }
            task<<str_answer<<';'<<endl;
            answer<<str_task<<';'<<endl;
        }
        task<<endl;
    }
    break;
}
task.close();
answer.close();
}
}

void MainWindow::on_pushButton_4_clicked()
{
    system("gedit task.txt");
}

void MainWindow::on_pushButton_5_clicked()
{
    system("gedit answer.txt");
}

void MainWindow::on_radioButton_clicked()
{
    tmp = 1;
}

```

```

void MainWindow::on_spinBox_valueChanged(int arg1)
{
    count = arg1;
}
void MainWindow::on_spinBox_2_valueChanged(int arg1)
{
    length = arg1;
}
void MainWindow::on_pushButton_2_clicked()
{
    MainWindow::on_spinBox_valueChanged(5);
    ui->spinBox->setValue(5);
    MainWindow::on_spinBox_2_valueChanged(5);
    ui->spinBox_2->setValue(5);
    ui->radioButton_2->setChecked(true);
    ofstream task;
    task.open("task.txt");
    ofstream answer;
    answer.open("answer.txt");
    task.close();
    answer.close();
}
void MainWindow::on_radioButton_2_clicked()
{
    tmp = 2;
}

```

mainwindow.h:

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
private slots:
    void on_pushButton_3_clicked();
    void on_pushButton_clicked();
    void on_pushButton_4_clicked();
    void on_pushButton_5_clicked();
    void on_radioButton_clicked();
    void on_spinBox_valueChanged(int arg1);
    void on_spinBox_2_valueChanged(int arg1);
    void on_pushButton_2_clicked();
    void on_radioButton_2_clicked();
private:
    Ui::MainWindow *ui;
    int count;
    int length;
    int tmp;
};
#endif // MAINWINDOW_H

```