МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г.
ШУХОВА» (БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №3
по дисциплине: Теория информации
тема: «Исследование возможности применения методов энтропийного кодирования для обработки двоичных последовательностей»

Выполнил: ст. группы ПВ-223
Дмитриев Андрей Александрович

Проверил:
Твердохлеб Виталий Викторович

Белгород 2024 г.

Содержание:

1. Тема лабораторной работы.
2. Цель лабораторной работы.
3. Текст задания к работе.
4. Выполнение задания.
    a. Аналитика касательно построения кодов для исходной двоичной последовательности
    b. Примеры кодовой реализации п.3, п.3, п.4 и п.6.
    c. Результаты обработки кодов, полученных в п.5.
    d. Текстовая последовательность, восстановленная к читаемомувиду.
    e. Общие выводы.
5. Вывод по работе.

Цель лабораторной работы: изучить возможность применения методов энтропийного кодирования для обработки двоичных последовательностей.Написать и отладить программу составления кода для каждого символа методом Хаффмана и методом Шеннона-Фано, кодирования и декодирования двоичной последовательности. Сравнить время работы программы и коэффициенты сжатия при использовании метода Хаффманаи метода Шеннона-Фано.

Текст задания к работе:

1. Открыть файл Лабораторная работа 3 (задание).txt. Рассмотреть возможность построения кода по методам Хаффмана и Шеннона-Фано для бинарной последовательности. Сделать выводы.
2. Рассмотреть варианты обработки цепочек символов, а именно:

- 2 символа;
- 4 символа;
- 8 символов.

Для этого разработать консольное приложение, разбивающее сплошной массив символов на цепочки заданной длины.

3. Рассматривая каждую цепочку (2, 4 и 8 символов длиной) как отдельный символ, построить коды по методу Хаффмана и Шеннона-Фано.
4. Составить последовательности из полученных кодов символов для каждого случая.
5. По результатам работы в п.3 сделать выводы по поводу полученных результатов для каждого из методов (простота, скорость, полученные результаты (рассчитать коэффициенты сжатия)).
6. Написать программу, восстанавливающую последовательности, полученные в п.3 в исходный вид согласно вариантам, приведенным в п.2.
7. Восстановить исходный текст из полученных последовательностей, пользуясь сервисом https://onlineutf8tools.com/convert-binary-to-utf8.

Ход работы:

№1

В файле содержится бинарная последовательность (0 и 1). Если строить код для каждого символа по отдельности, то каждому символу будет соответствовать код длиною 1 символ. Мы можем составлять код, взяв за символ последовательность нулей и единиц определенной длины. Если мы возьмём длину 2, то алфавит будет содержать максимум 4 символа, если длина будет равна 4, то алфавит будет состоять не больше, чем из 16 символов. Т. е. если длина последовательности из нулей и единиц, учитываемой как один символ, будет равна n, то алфавит сообщения будет состоять из 2^n символов или меньше, если некоторые символы (последовательности нулей и единиц) не встретятся в сообщении.

№2

Код программы:

```
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>

int getNumberFromCharVector(const std::vector<char> &a)
{
    int i = a.size() - 1;
    int number = 0;
    for (auto &x : a)
    {
        if (x == '1')
            number += pow(2, i);
        i--;
    }
    return number;
}

std::vector<int> getSequencesOfNCharactersEach(const std::string &s, const int
n)
{
    std::vector<int> sequences;
    std::vector<char> a(n);
    int i = 1;
    for (auto &x : s)
    {
        a[i] = x;
        if ((i + 1) % n == 0)
        {
            int p = getNumberFromCharVector(a);
            sequences.push_back(p);
```

```
            i = 0;
        }
        else
            i++;
    }

    return sequences;
}

void outputVector(const std::vector<int> &r)
{
    for (auto &x : r)
        std::cout << x << ' ';
    std::cout << '\n';
}

std::vector<int> getBinaryNumberNotation(std::vector<int> &a, int i, int n)
{
    int digit = n & 1;
    if (n == 0)
        return a;
    else
    {
        getBinaryNumberNotation(a, i + 1, n >> 1);
        a[i] = digit;
    }
    return a;
}

void outputVectorSequencesForReading(const std::vector<int> &r, const int
&length)
{
    for (auto &x : r)
    {
        std::vector<int> a(length, 0);
        a = getBinaryNumberNotation(a, 0, x);
        for (auto &y : a)
        {
            std::cout << y;
        }
        std::cout << ' ';
    }
    std::cout << '\n';
}

int main()
{
    std::string s;
    std::ifstream f0("T:\\2kurs2sem\\InformTheor\\lab3\\TI_3.txt");
    f0 >> s;
    f0.close();
```

```
    int length = 2;
    std::vector<int> r = getSequencesOfNCharactersEach(s, length);
    std::cout << "In computer memory:\n";

    outputVector(r);

    std::cout << "\nFor reading:\n";
    outputVectorSequencesForReading(r, length);

    return 0;
}
```

Результат работы программы:

По 2 символа:

```
In computer memory:
1 2 2 0 1 0 2 1 1 2 2 0 1 1 2 2 3 2 2 0 3 0 0 1 1 2 2 0 1 1 2 2 3 2 2 0 3 0 0 0
0 1 0 0 1 2 2 0 3 0 0 0 3 2 2 0 1 1 2 1 1 2 2 0 1 1 3 0 1 2 2 0 3 0 0 0 3 2 2 0
3 0 0 1 1 2 2 0 1 1 2 2 3 2 2 0 1 1 3 1 2 1 1 2 0 1 0 0 1 2 2 0 1 1 2 1 1 2 2 0
1 1 3 0 1 2 2 0 1 1 2 3 3 2 2 0 1 1 2 3 1 2 2 0 1 1 2 0 1 2 2 0 1 1 3 1 2 1 1 2
0 1 0 0 1 2 2 0 1 1 3 1 1 2 2 0 3 0 0 0 1 2 2 0 3 0 1 3 3 2 2 0 3 0 0 2 3 2 2 0
3 0 0 1 1 2 2 0 1 1 2 2 3 2 2 0 1 1 3 1 2 1 0 0 1 2 2 0 1 1 3 0 0 1 0 0 1 2 2 0
1 1 2 1 3 2 2 0 3 0 0 1 3 2 2 0 1 1 2 2 1 2 2 0 1 1 2 2 3 2 2 0 1 1 3 1 2 1 0 0
1 2 2 0 1 1 3 2 3 2 2 0 1 1 2 0 0 1 0 0 1 2 2 0 3 0 0 0 1 2 2 0 1 1 2 0 1 2 2 0
1 1 2 3 3 2 2 0 1 1 3 2 3 2 2 0 3 0 1 1 3 2 2 0 1 1 2 2 2 1 0 0 1 2 2 0 1 1 3 1
3 2 2 0 1 1 2 0 1 2 2 0 1 1 2 2 1 2 2 0 3 0 1 1 2 1 1 3 0 1 0 0 1 2 2 0 1 1 0 1
1 2 2 0 1 1 3 3 0 1 0 0 1 2 2 0 1 1 2 3 1 2 2 0 1 1 2 0 1 2 2 0 1 1 3 1 3 2 2 0
1 1 3 3 1 2 2 0 1 1 2 0 3 2 2 0 1 1 3 2 3 2 2 0 3 0 1 1 3 2 2 0 1 1 3 2 0 1 0 0
1 2 2 0 3 0 0 1 1 2 2 0 1 1 3 3 1 2 2 0 1 1 3 2 3 2 2 0 1 1 2 2 3 2 2 0 1 1 3 2
3 2 2 0 3 0 1 2 1 2 2 0 1 1 3 1 1 2 2 0 1 1 3 0 1 2 2 0 1 1 3 2 0 1 0 0 1 2 2 0
1 1 2 1 3 2 2 0 1 1 3 3 1 2 2 0 1 1 3 1 3 2 2 0 1 1 3 3 1 2 2 0 3 0 0 0 3 2 2 0
1 1 3 1 1 2 2 0 1 1 3 3 1 2 2 0 1 1 3 2 0 1 1 2 0 1 0 0 1 2 2 0 3 0 0 1 1 2 2 0
1 1 3 3 0 1 0 0 1 2 2 0 1 1 2 1 3 2 2 0 3 0 0 0 1 2 2 0 3 0 0 1 3 2 2 0 1 1 2 0
3 2 2 0 3 0 1 1 3 2 2 0 1 1 3 2 0 1 0 0 1 2 2 0 1 1 2 0 3 2 2 0 1 1 2 0 1 2 2 0
3 0 0 0 3 2 2 0 1 1 3 3 1 2 2 0 1 1 2 1 1 2 2 0 3 0 1 1 3 2 2 0 1 1 3 2 0 1 0 0
1 2 2 0 3 0 0 0 1 2 2 0 1 1 2 0 1 2 2 0 3 0 0 0 3 2 2 0 1 1 3 1 1 2 2 0 1 1 2 0
1 2 2 0 3 0 0 1 1 2 2 0 1 1 3 3 1 2 2 0 1 1 3 2 0 1 0 0 1 2 2 0 3 0 0 0 1 2 2 0
1 1 2 0 1 2 2 0 3 0 0 0 3 2 2 0 1 1 3 3 3 2 2 0 1 1 2 2 3 2 2 0 1 1 2 1 1 2 2 0
1 1 2 0 1 2 2 0 1 1 3 1 2 1 0 0 1 2 2 0 1 1 3 3 1 2 2 0 1 1 3 2 2 1 0 0 1 2 2 0
3 0 0 0 3 2 2 0 1 1 2 1 1 2 2 0 1 1 3 3 1 2 2 0 3 0 1 3 0 1 0 0 1 2 2 0 1 1 2 0
3 2 2 0 1 1 3 3 1 2 2 0 1 1 2 2 3 2 2 0 1 1 2 1 1 2 2 0 3 0 0 1 3 2 2 0 3 0 1 3
0 1 0 0 1 2 2 0 1 1 3 3 3 2 2 0 1 1 2 2 3 2 2 0 3 0 0 0 3 2 2 0 1 1 2 2 3 2 2 0
1 1 3 2 3 2 2 0 1 1 3 1 1 2 2 0 3 0 0 1 2 1 1 3 0 1 0 0 1 2 2 0 1 1 0 2 1 2 2 0
1 1 3 3 1 2 2 0 1 1 3 2 3 2 2 0 1 1 2 0 1 2 2 0 3 0 0 0 1 2 2 0 1 1 3 0 0 1 0 0
1 2 2 0 3 0 0 3 3 2 2 0 3 0 0 1 3 2 2 0 3 0 0 1 1 2 2 0 3 0 1 2 0 1 0 0 1 2 2 0
1 1 2 3 3 2 2 0 1 1 2 0 1 2 2 0 1 1 3 2 1 2 2 0 1 1 2 2 3 2 2 0 3 0 0 1 1 2 2 0
1 1 3 2 3 2 2 0 1 1 3 3 0 1 0 0 1 2 2 0 1 1 3 2 1 2 2 0 1 1 3 0 1 2 2 0 1 1 2 1
3 2 2 0 1 1 2 0 1 2 2 0 1 1 3 1 3 2 2 0 1 1 3 0 0 1 0 0 1 2 2 0 3 0 0 0 3 2 2 0
1 1 3 1 1 2 2 0 1 1 2 1 1 2 2 0 1 1 3 3 1 2 2 0 1 1 2 3 3 2 2 0 3 0 1 2 0 1 0 0
1 2 2 0 1 1 3 3 1 2 2 0 1 1 2 1 3 2 2 0 3 0 0 0 1 2 2 0 1 1 3 3 1 2 2 0 1 1 3 2
1 2 2 0 1 1 3 2 3 2 2 0 3 0 1 1 3 2 2 0 1 1 2 2 2 1 0 0 1 2 2 0 1 1 2 0 3 2 2 0
1 1 2 2 3 2 2 0 1 1 3 1 3 2 2 0 3 0 1 1 3 2 2 0 1 1 2 2 2 1 0 0 1 2 2 0 3 0 0 2
```

```
3 2 2 0 1 1 3 1 3 2 2 0 1 1 3 3 1 2 2 0 1 1 3 3 3 2 2 0 3 0 1 2 1 2 2 0 3 0 1 3
2 1 0 0 1 2 2 0 3 0 0 0 3 2 2 0 1 1 3 2 3 2 2 0 1 1 2 2 3 2 2 0 1 1 2 1 3 2 2 0
1 1 2 0 0 1 1 2 0 1 0 0 1 2 2 0 1 1 3 3 1 2 2 0 1 1 2 0 3 2 2 0 1 1 3 0 1 2 2 0
1 1 3 1 3 2 2 0 3 0 1 2 1 2 2 0 1 1 3 2 3 2 2 0 1 1 3 3 0 1 0 0 1 2 2 0 3 0 0 0
3 2 2 0 3 0 1 1 3 2 2 0 1 1 3 3 3 2 2 0 1 1 2 0 1 2 2 0 1 1 2 1 1 2 2 0 3 0 1 0
1 2 2 0 1 1 3 0 1 2 2 0 1 1 2 2 3 2 2 0 3 0 0 0 3 2 2 0 3 0 1 3 2 1 0 0 1 2 2 0
1 1 3 2 3 2 2 0 1 1 2 0 0 1 0 0 1 2 2 0 3 0 0 1 1 2 2 0 3 0 0 0 1 2 2 0 1 1 3 3
1 2 2 0 3 0 0 1 1 2 2 0 3 0 0 1 3 2 2 0 1 1 2 0 1 2 2 0 3 0 0 0 1 2 2 0 3 0 1 1
2 1 1 2 0 1 0 0 1 2 2 0 1 1 3 2 3 2 2 0 1 1 2 0 0 1 0 0 1 2 2 0 3 0 0 1 3 2 2 0
1 1 3 1 3 2 2 0 1 1 3 0 1 2 2 0 3 0 0 3 1 2 2 0 3 0 0 1 2 1 1 2 0 1 0 0 1 2 2 0
1 1 3 2 3 2 2 0 1 1 2 0 0 1 0 0 1 2 2 0 3 0 1 2 3 2 2 0 1 1 3 1 1 2 2 0 1 1 3 0
1 2 2 0 1 1 3 3 3 2 2 0 1 1 2 0 1 2 2 0 1 1 2 3 1 2 2 0 1 1 3 0 0 1 1 2 0 1 0 0
1 2 2 0 1 1 3 1 3 2 2 0 1 1 3 3 1 2 2 0 3 0 1 0 1 2 2 0 1 1 2 0 1 2 2 0 1 1 2 2
1 2 2 0 1 1 2 2 3 2 2 0 1 1 3 0 2 1 0 0 1 2 2 0 1 1 3 0 0 1 0 0 1 2 2 0 1 1 3 3
3 2 2 0 3 0 0 0 1 2 2 0 1 1 3 3 1 2 2 0 3 0 0 2 3 2 2 0 1 1 3 3 1 2 2 0 1 1 2 3
1 2 2 0 1 1 3 0 1 2 2 0 3 0 0 2 2 1 1 3
```

For reading:

```
10 01 01 00 10 00 01 10 10 01 01 00 10 10 01 01 11 01 01 00 11 00 00 10 10 01 01
00 10 10 01 01 11 01 01 00 11 00 00 00 00 10 00 00 10 01 01 00 11 00 00 00 11 01
01 00 10 10 01 10 10 01 01 00 10 10 11 00 10 01 01 00 11 00 00 00 11 01 01 00 11
00 00 10 10 01 01 00 10 10 01 01 11 01 01 00 10 10 11 10 01 10 10 01 00 10 00 00
10 01 01 00 10 10 01 10 10 01 01 00 10 10 11 00 10 01 01 00 10 10 01 11 11 01 01
00 10 10 01 11 10 01 01 00 10 10 01 00 10 01 01 00 10 10 11 10 01 10 10 01 00 10
00 00 10 01 01 00 10 10 11 10 10 01 01 00 11 00 00 00 10 01 01 00 11 00 10 11 11
01 01 00 11 00 00 01 11 01 01 00 11 00 00 10 10 01 01 00 10 10 01 01 11 01 01 00
10 10 11 10 01 10 00 00 10 01 01 00 10 10 11 00 00 10 00 00 10 01 01 00 10 10 01
10 11 01 01 00 11 00 00 10 11 01 01 00 10 10 01 01 10 01 01 00 10 10 01 01 11 01
01 00 10 10 11 10 01 10 00 00 10 01 01 00 10 10 11 01 11 01 01 00 10 10 01 00 00
10 00 00 10 01 01 00 11 00 00 00 10 01 01 00 10 10 01 00 10 01 01 00 10 10 01 11
11 01 01 00 10 10 11 01 11 01 01 00 11 00 10 10 11 01 01 00 10 10 01 01 01 10 00
00 10 01 01 00 10 10 11 10 11 01 01 00 10 10 01 00 10 01 01 00 10 10 01 01 10 01
01 00 11 00 10 10 01 10 10 11 00 10 00 00 10 01 01 00 10 10 00 10 10 01 01 00 10
10 11 11 00 10 00 00 10 01 01 00 10 10 01 11 10 01 01 00 10 10 01 00 10 01 01 00
10 10 11 10 11 01 01 00 10 10 11 11 10 01 01 00 10 10 01 00 11 01 01 00 10 10 11
01 11 01 01 00 11 00 10 10 11 01 01 00 10 10 11 01 00 10 00 00 10 01 01 00 11 00
00 10 10 01 01 00 10 10 11 11 10 01 01 00 10 10 11 01 11 01 01 00 10 10 01 01 11
01 01 00 10 10 11 01 11 01 01 00 11 00 10 01 10 01 01 00 10 10 11 10 10 01 01 00
10 10 11 00 10 01 01 00 10 10 11 01 00 10 00 00 10 01 01 00 10 10 01 10 11 01 01
00 10 10 11 11 10 01 01 00 10 10 11 10 11 01 01 00 10 10 11 11 10 01 01 00 11 00
00 00 11 01 01 00 10 10 11 10 10 01 01 00 10 10 11 11 10 01 01 00 10 10 11 01 00
10 10 01 00 10 00 00 10 01 01 00 11 00 00 10 10 01 01 00 10 10 11 11 00 10 00 00
10 01 01 00 10 10 01 10 11 01 01 00 11 00 00 00 10 01 01 00 11 00 00 10 11 01 01
00 10 10 01 00 11 01 01 00 11 00 10 10 11 01 01 00 10 10 11 01 00 10 00 00 10 01
01 00 10 10 01 00 11 01 01 00 10 10 01 00 10 01 01 00 11 00 00 00 11 01 01 00 10
10 11 11 10 01 01 00 10 10 01 10 10 01 01 00 11 00 10 10 11 01 01 00 10 10 11 01
00 10 00 00 10 01 01 00 11 00 00 00 10 01 01 00 10 10 01 00 10 01 01 00 11 00 00
00 11 01 01 00 10 10 11 10 10 01 01 00 10 10 01 00 10 01 01 00 11 00 00 10 10 01
01 00 10 10 11 11 10 01 01 00 10 10 11 01 00 10 00 00 10 01 01 00 11 00 00 00 10
01 01 00 10 10 01 00 10 01 01 00 11 00 00 00 11 01 01 00 10 10 11 11 11 01 01 00
10 10 01 01 11 01 01 00 10 10 01 10 10 01 01 00 10 10 01 00 10 01 01 00 10 10 11
10 01 10 00 00 10 01 01 00 10 10 11 11 10 01 01 00 10 10 11 01 01 10 00 00 10 01
```

```
01 00 11 00 00 00 11 01 01 00 10 10 01 10 10 01 01 00 10 10 11 11 10 01 01 00 11
00 10 11 00 10 00 00 10 01 01 00 10 10 01 00 11 01 01 00 10 10 11 11 10 01 01 00
10 10 01 01 11 01 01 00 10 10 01 10 10 01 01 00 11 00 00 10 11 01 01 00 11 00 10
11 00 10 00 00 10 01 01 00 10 10 11 11 11 01 01 00 10 10 01 01 11 01 01 00 11 00
00 00 11 01 01 00 10 10 01 01 11 01 01 00 10 10 11 01 11 01 01 00 10 10 11 10 10
01 01 00 11 00 00 10 01 10 10 11 00 10 00 00 10 01 01 00 10 10 00 01 10 01 01 00
10 10 11 11 10 01 01 00 10 10 11 01 11 01 01 00 10 10 01 00 10 01 01 00 11 00 00
00 10 01 01 00 10 10 11 00 00 10 00 00 10 01 01 00 11 00 00 11 11 01 01 00 11 00
00 10 11 01 01 00 11 00 00 10 10 01 01 00 11 00 10 01 00 10 00 00 10 01 01 00 10
10 01 11 11 01 01 00 10 10 01 00 10 01 01 00 10 10 11 01 10 01 01 00 10 10 01 01
11 01 01 00 11 00 00 10 10 01 01 00 10 10 11 01 11 01 01 00 10 10 11 11 00 10 00
00 10 01 01 00 10 10 11 01 10 01 01 00 10 10 11 00 10 01 01 00 10 10 01 10 11 01
01 00 10 10 01 00 10 01 01 00 10 10 11 10 11 01 01 00 10 10 11 00 00 10 00 00 10
01 01 00 11 00 00 00 11 01 01 00 10 10 11 10 10 01 01 00 10 10 01 10 10 01 01 00
10 10 11 11 10 01 01 00 10 10 01 11 11 01 01 00 11 00 10 01 00 10 00 00 10 01 01
00 10 10 11 11 10 01 01 00 10 10 01 10 11 01 01 00 11 00 00 00 10 01 01 00 10 10
11 11 10 01 01 00 10 10 11 01 10 01 01 00 10 10 11 01 11 01 01 00 11 00 10 10 11
01 01 00 10 10 01 01 01 10 00 00 10 01 01 00 10 10 01 00 11 01 01 00 10 10 01 01
11 01 01 00 10 10 11 10 11 01 01 00 11 00 10 10 11 01 01 00 10 10 01 01 01 10 00
00 10 01 01 00 11 00 00 01 11 01 01 00 10 10 11 10 11 01 01 00 10 10 11 11 10 01
01 00 10 10 11 11 11 01 01 00 11 00 10 01 10 01 01 00 11 00 10 11 01 10 00 00 10
01 01 00 11 00 00 00 11 01 01 00 10 10 11 01 11 01 01 00 10 10 01 01 11 01 01 00
10 10 01 10 11 01 01 00 10 10 01 00 00 10 10 01 00 10 00 00 10 01 01 00 10 10 11
11 10 01 01 00 10 10 01 00 11 01 01 00 10 10 11 00 10 01 01 00 10 10 11 10 11 01
01 00 11 00 10 01 10 01 01 00 10 10 11 01 11 01 01 00 10 10 11 11 00 10 00 00 10
01 01 00 11 00 00 00 11 01 01 00 11 00 10 10 11 01 01 00 10 10 11 11 11 01 01 00
10 10 01 00 10 01 01 00 10 10 01 10 10 01 01 00 11 00 10 00 10 01 01 00 10 10 11
00 10 01 01 00 10 10 01 01 11 01 01 00 11 00 00 00 11 01 01 00 11 00 10 11 01 10
00 00 10 01 01 00 10 10 11 01 11 01 01 00 10 10 01 00 00 10 00 00 10 01 01 00 11
00 00 10 10 01 01 00 11 00 00 00 10 01 01 00 10 10 11 11 10 01 01 00 11 00 00 10
10 01 01 00 11 00 00 10 11 01 01 00 10 10 01 00 10 01 01 00 11 00 00 00 10 01 01
00 11 00 10 10 01 10 10 01 00 10 00 00 10 01 01 00 10 10 11 01 11 01 01 00 10 10
01 00 00 10 00 00 10 01 01 00 11 00 00 10 11 01 01 00 10 10 11 10 11 01 01 00 10
10 11 00 10 01 01 00 11 00 00 11 10 01 01 00 11 00 00 10 01 10 10 01 00 10 00 00
10 01 01 00 10 10 11 01 11 01 01 00 10 10 01 00 00 10 00 00 10 01 01 00 11 00 10
01 11 01 01 00 10 10 11 10 10 01 01 00 10 10 11 00 10 01 01 00 10 10 11 11 11 01
01 00 10 10 01 00 10 01 01 00 10 10 01 11 10 01 01 00 10 10 11 00 00 10 10 01 00
10 00 00 10 01 01 00 10 10 11 10 11 01 01 00 10 10 11 11 10 01 01 00 11 00 10 00
10 01 01 00 10 10 01 00 10 01 01 00 10 10 01 01 10 01 01 00 10 10 01 01 11 01 01
00 10 10 11 00 01 10 00 00 10 01 01 00 10 10 11 00 00 10 00 00 10 01 01 00 10 10
11 11 11 01 01 00 11 00 00 00 10 01 01 00 10 10 11 11 10 01 01 00 11 00 00 01 11
01 01 00 10 10 11 11 10 01 01 00 10 10 01 11 10 01 01 00 10 10 11 00 10 01 01 00
11 00 00 01 01 10 10
```

По 4 символа:

```
In computer memory:
6 8 4 9 6 8 5 10 14 8 12 1 6 8 5 10 14 8 12 0 1 0 6 8 12 0 14 8 5 9 6 8 5 12 6 8
12 0 14 8 12 1 6 8 5 10 14 8 5 13 9 6 1 0 6 8 5 9 6 8 5 12 6 8 5 11 14 8 5 11 6
8 5 8 6 8 5 13 9 6 1 0 6 8 5 13 6 8 12 0 6 8 12 7 14 8 12 2 14 8 12 1 6 8 5 10
14 8 5 13 9 0 6 8 5 12 1 0 6 8 5 9 14 8 12 1 14 8 5 10 6 8 5 10 14 8 5 13 9 0 6
8 5 14 14 8 5 8 1 0 6 8 12 0 6 8 5 8 6 8 5 11 14 8 5 14 14 8 12 5 14 8 5 10 9 0
6 8 5 13 14 8 5 8 6 8 5 10 6 8 12 5 9 7 1 0 6 8 5 1 6 8 5 15 1 0 6 8 5 11 6 8 5
```

8 6 8 5 13 14 8 5 15 6 8 5 8 14 8 5 14 14 8 12 5 14 8 5 14 1 0 6 8 12 1 6 8 5 15
6 8 5 14 14 8 5 10 14 8 5 14 14 8 12 6 6 8 5 13 6 8 5 12 6 8 5 14 1 0 6 8 5 9 14
8 5 15 6 8 5 13 14 8 5 15 6 8 12 0 14 8 5 13 6 8 5 15 6 8 5 14 1 6 1 0 6 8 12 1
6 8 5 15 1 0 6 8 5 9 14 8 12 0 6 8 12 1 14 8 5 8 14 8 12 5 14 8 5 14 1 0 6 8 5 8
14 8 5 8 6 8 12 0 14 8 5 15 6 8 5 9 6 8 12 5 14 8 5 14 1 0 6 8 12 0 6 8 5 8 6 8
12 0 14 8 5 13 6 8 5 8 6 8 12 1 6 8 5 15 6 8 5 14 1 0 6 8 12 0 6 8 5 8 6 8 12 0
14 8 5 15 14 8 5 10 14 8 5 9 6 8 5 8 6 8 5 13 9 0 6 8 5 15 6 8 5 14 9 0 6 8 12 0
14 8 5 9 6 8 5 15 6 8 12 7 1 0 6 8 5 8 14 8 5 15 6 8 5 10 14 8 5 9 6 8 12 1 14 8
12 7 1 0 6 8 5 15 14 8 5 10 14 8 12 0 14 8 5 10 14 8 5 14 14 8 5 13 6 8 12 1 9 7
1 0 6 8 5 2 6 8 5 15 6 8 5 14 14 8 5 8 6 8 12 0 6 8 5 12 1 0 6 8 12 3 14 8 12 1
14 8 12 1 6 8 12 6 1 0 6 8 5 11 14 8 5 8 6 8 5 14 6 8 5 10 14 8 12 1 6 8 5 14 14
8 5 15 1 0 6 8 5 14 6 8 5 12 6 8 5 9 14 8 5 8 6 8 5 13 14 8 5 12 1 0 6 8 12 0 14
8 5 13 6 8 5 9 6 8 5 15 6 8 5 11 14 8 12 6 1 0 6 8 5 15 6 8 5 9 14 8 12 0 6 8 5
15 6 8 5 14 6 8 5 14 14 8 12 5 14 8 5 10 9 0 6 8 5 8 14 8 5 10 14 8 5 13 14 8 12
5 14 8 5 10 9 0 6 8 12 2 14 8 5 13 14 8 5 15 6 8 5 15 14 8 12 6 6 8 12 7 9 0 6 8
12 0 14 8 5 14 14 8 5 10 14 8 5 9 14 8 5 8 1 6 1 0 6 8 5 15 6 8 5 8 14 8 5 12 6
8 5 13 14 8 12 6 6 8 5 14 14 8 5 15 1 0 6 8 12 0 14 8 12 5 14 8 5 15 14 8 5 8 6
8 5 9 6 8 12 4 6 8 5 12 6 8 5 10 14 8 12 0 14 8 12 7 9 0 6 8 5 14 14 8 5 8 1 0 6
8 12 1 6 8 12 0 6 8 5 15 6 8 12 1 6 8 12 1 14 8 5 8 6 8 12 0 6 8 12 5 9 6 1 0 6
8 5 14 14 8 5 8 1 0 6 8 12 1 14 8 5 13 14 8 5 12 6 8 12 3 6 8 12 1 9 6 1 0 6 8 5
14 14 8 5 8 1 0 6 8 12 6 14 8 5 13 6 8 5 12 6 8 5 15 14 8 5 8 6 8 5 11 6 8 5 12
1 6 1 0 6 8 5 13 14 8 5 15 6 8 12 4 6 8 5 8 6 8 5 10 6 8 5 10 14 8 5 12 9 0 6 8
5 12 1 0 6 8 5 15 14 8 12 0 6 8 5 15 6 8 12 2 14 8 5 15 6 8 5 11 6 8 5 12 6 8 12
2 9 7

For reading:
0110 0001 0010 1001 0110 0001 1010 0101 0111 0001 0011 1000 0110 0001 1010 0101
0111 0001 0011 0000 1000 0000 0110 0001 0011 0000 0111 0001 1010 1001 0110 0001
1010 0011 0110 0001 0011 0000 0111 0001 0011 1000 0110 0001 1010 0101 0111 0001
1010 1011 1001 0110 1000 0000 0110 0001 1010 1001 0110 0001 1010 0011 0110 0001
1010 1101 0111 0001 1010 1101 0110 0001 1010 0001 0110 0001 1010 1011 1001 0110
1000 0000 0110 0001 1010 1011 0110 0001 0011 0000 0110 0001 0011 1110 0111 0001
0011 0100 0111 0001 0011 1000 0110 0001 1010 0101 0111 0001 1010 1011 1001 0000
0110 0001 1010 0011 1000 0000 0110 0001 1010 1001 0111 0001 0011 1000 0111 0001
1010 0101 0110 0001 1010 0101 0111 0001 1010 1011 1001 0000 0110 0001 1010 0111
0111 0001 1010 0001 1000 0000 0110 0001 0011 0000 0110 0001 1010 0001 0110 0001
1010 1101 0111 0001 1010 0111 0111 0001 0011 1010 0111 0001 1010 0101 1001 0000
0110 0001 1010 1011 0111 0001 1010 0001 0110 0001 1010 0101 0110 0001 0011 1010
1001 1110 1000 0000 0110 0001 1010 1000 0110 0001 1010 1111 1000 0000 0110 0001
1010 1101 0110 0001 1010 0001 0110 0001 1010 1011 0111 0001 1010 1111 0110 0001
1010 0001 0111 0001 1010 0111 0111 0001 0011 1010 0111 0001 1010 0111 1000 0000
0110 0001 0011 1000 0110 0001 1010 1111 0110 0001 1010 0111 0111 0001 1010 0101
0111 0001 1010 0111 0111 0001 0011 0110 0110 0001 1010 1011 0110 0001 1010 0011
0110 0001 1010 0111 1000 0000 0110 0001 1010 1001 0111 0001 1010 1111 0110 0001
1010 1011 0111 0001 1010 1111 0110 0001 0011 0000 0111 0001 1010 1011 0110 0001
1010 1111 0110 0001 1010 0111 1000 0110 1000 0000 0110 0001 0011 1000 0110 0001
1010 1111 1000 0000 0110 0001 1010 1001 0111 0001 0011 0000 0110 0001 0011 1000
0111 0001 1010 0001 0111 0001 0011 1010 0111 0001 1010 0111 1000 0000 0110 0001
1010 0001 0111 0001 1010 0001 0110 0001 0011 0000 0111 0001 1010 1111 0110 0001
1010 1001 0110 0001 0011 1010 0111 0001 1010 0111 1000 0000 0110 0001 0011 0000
0110 0001 1010 0001 0110 0001 0011 0000 0111 0001 1010 1011 0110 0001 1010 0001
0110 0001 0011 1000 0110 0001 1010 1111 0110 0001 1010 0111 1000 0000 0110 0001

```
0011 0000 0110 0001 1010 0001 0110 0001 0011 0000 0111 0001 1010 1111 0111 0001
1010 0101 0111 0001 1010 1001 0110 0001 1010 0001 0110 0001 1010 1011 1001 0000
0110 0001 1010 1111 0110 0001 1010 0111 1001 0000 0110 0001 0011 0000 0111 0001
1010 1001 0110 0001 1010 1111 0110 0001 0011 1110 1000 0000 0110 0001 1010 0001
0111 0001 1010 1111 0110 0001 1010 0101 0111 0001 1010 1001 0110 0001 0011 1000
0111 0001 0011 1110 1000 0000 0110 0001 1010 1111 0111 0001 1010 0101 0111 0001
0011 0000 0111 0001 1010 0101 0111 0001 1010 0111 0111 0001 1010 1011 0110 0001
0011 1000 1001 1110 1000 0000 0110 0001 1010 0100 0110 0001 1010 1111 0110 0001
1010 0111 0111 0001 1010 0001 0110 0001 0011 0000 0110 0001 1010 0011 1000 0000
0110 0001 0011 1100 0111 0001 0011 1000 0111 0001 0011 1000 0110 0001 0011 0110
1000 0000 0110 0001 1010 1101 0111 0001 1010 0001 0110 0001 1010 0111 0110 0001
1010 0101 0111 0001 0011 1000 0110 0001 1010 0111 0111 0001 1010 1111 1000 0000
0110 0001 1010 0111 0110 0001 1010 0011 0110 0001 1010 1001 0111 0001 1010 0001
0110 0001 1010 1011 0111 0001 1010 0011 1000 0000 0110 0001 0011 0000 0111 0001
1010 1011 0110 0001 1010 1001 0110 0001 1010 1111 0110 0001 1010 1101 0111 0001
0011 0110 1000 0000 0110 0001 1010 1111 0110 0001 1010 1001 0111 0001 0011 0000
0110 0001 1010 1111 0110 0001 1010 0111 0110 0001 1010 0111 0111 0001 0011 1010
0111 0001 1010 0101 1001 0000 0110 0001 1010 0001 0111 0001 1010 0101 0111 0001
1010 1011 0111 0001 0011 1010 0111 0001 1010 0101 1001 0000 0110 0001 0011 0100
0111 0001 1010 1011 0111 0001 1010 1111 0110 0001 1010 1111 0111 0001 0011 0110
0110 0001 0011 1110 1001 0000 0110 0001 0011 0000 0111 0001 1010 0111 0111 0001
1010 0101 0111 0001 1010 1001 0111 0001 1010 0001 1000 0110 1000 0000 0110 0001
1010 1111 0110 0001 1010 0001 0111 0001 1010 0011 0110 0001 1010 1011 0111 0001
0011 0110 0110 0001 1010 0111 0111 0001 1010 1111 1000 0000 0110 0001 0011 0000
0111 0001 0011 1010 0111 0001 1010 1111 0111 0001 1010 0001 0110 0001 1010 1001
0110 0001 0011 0010 0110 0001 1010 0011 0110 0001 1010 0101 0111 0001 0011 0000
0111 0001 0011 1110 1001 0000 0110 0001 1010 0111 0111 0001 1010 0001 1000 0000
0110 0001 0011 1000 0110 0001 0011 0000 0110 0001 1010 1111 0110 0001 0011 1000
0110 0001 0011 1000 0111 0001 1010 0001 0110 0001 0011 0000 0110 0001 0011 1010
1001 0110 1000 0000 0110 0001 1010 0111 0111 0001 1010 0001 1000 0000 0110 0001
0011 1000 0111 0001 1010 1011 0111 0001 1010 0011 0110 0001 0011 1100 0110 0001
0011 1000 1001 0110 1000 0000 0110 0001 1010 0111 0111 0001 1010 0001 1000 0000
0110 0001 0011 0110 0111 0001 1010 1011 0110 0001 1010 0011 0110 0001 1010 1111
0111 0001 1010 0001 0110 0001 1010 1101 0110 0001 1010 0011 1000 0110 1000 0000
0110 0001 1010 1011 0111 0001 1010 1111 0110 0001 0011 0010 0110 0001 1010 0001
0110 0001 1010 0101 0110 0001 1010 0101 0111 0001 1010 0011 1001 0000 0110 0001
1010 0011 1000 0000 0110 0001 1010 1111 0111 0001 0011 0000 0110 0001 1010 1111
0110 0001 0011 0100 0111 0001 1010 1111 0110 0001 1010 1101 0110 0001 1010 0011
0110 0001 0011 0100 1001 1110
```

По 8 символов:

```
In computer memory:
104 73 104 90 232 193 104 90 232 192 16 104 192 232 89 104 92 104 192 232 193
104 90 232 93 150 16 104 89 104 92 104 91 232 91 104 88 104 93 150 16 104 93 104
192 104 199 232 194 232 193 104 90 232 93 144 104 92 16 104 89 232 193 232 90
104 90 232 93 144 104 94 232 88 16 104 192 104 88 104 91 232 94 232 197 232 90
144 104 93 232 88 104 90 104 197 151 16 104 81 104 95 16 104 91 104 88 104 93
232 95 104 88 232 94 232 197 232 94 16 104 193 104 95 104 94 232 90 232 94 232
198 104 93 104 92 104 94 16 104 89 232 95 104 93 232 95 104 192 232 93 104 95
104 94 22 16 104 193 104 95 16 104 89 232 192 104 193 232 88 232 197 232 94 16
104 88 232 88 104 192 232 95 104 89 104 197 232 94 16 104 192 104 88 104 192 232
93 104 88 104 193 104 95 104 94 16 104 192 104 88 104 192 232 95 232 90 232 89
```

104 88 104 93 144 104 95 104 94 144 104 192 232 89 104 95 104 199 16 104 88 232 95 104 90 232 89 104 193 232 199 16 104 95 232 90 232 192 232 90 232 94 232 93 104 193 151 16 104 82 104 95 104 94 232 88 104 192 104 92 16 104 195 232 193 232 193 104 198 16 104 91 232 88 104 94 104 90 232 193 104 94 232 95 16 104 94 104 92 104 89 232 88 104 93 232 92 16 104 192 232 93 104 89 104 95 104 91 232 198 16 104 95 104 89 232 192 104 95 104 94 104 94 232 197 232 90 144 104 88 232 90 232 93 232 197 232 90 144 104 194 232 93 232 95 104 95 232 198 104 199 144 104 192 232 94 232 90 232 89 232 88 22 16 104 95 104 88 232 92 104 93 232 198 104 94 232 95 16 104 192 232 197 232 95 232 88 104 89 104 196 104 92 104 90 232 192 232 199 144 104 94 232 88 16 104 193 104 192 104 95 104 193 104 193 232 88 104 192 104 197 150 16 104 94 232 88 16 104 193 232 93 232 92 104 195 104 193 150 16 104 94 232 88 16 104 198 232 93 104 92 104 95 232 88 104 91 104 92 22 16 104 93 232 95 104 196 104 88 104 90 104 90 232 92 144 104 92 16 104 95 232 192 104 95 104 194 232 95 104 91 104 92 104 194 151

For reading:
00010110 10010010 00010110 01011010 00010111 10000011 00010110 01011010 00010111 00000011 00001000 00010110 00000011 00010111 10011010 00010110 00111010 00010110 00000011 00010111 10000011 00010110 01011010 00010111 10111010 01101001 00001000 00010110 10011010 00010110 00111010 00010110 11011010 00010111 11011010 00010110 00011010 00010110 10111010 01101001 00001000 00010110 10111010 00010110 00000011 00010110 11100011 00010111 01000011 00010111 10000011 00010110 01011010 00010111 10111010 00001001 00010110 00111010 00001000 00010110 10011010 00010111 10000011 00010111 01011010 00010110 01011010 00010111 10111010 00001001 00010110 01111010 00010111 00011010 00001000 00010110 00000011 00010110 00011010 00010110 11011010 00010111 01111010 00010111 10100011 00010111 01011010 00001001 00010110 10111010 00010111 00011010 00010110 01011010 00010110 10100011 11101001 00001000 00010110 10001010 00010110 11111010 00001000 00010110 11011010 00010110 00011010 00010110 10111010 00010111 11111010 00010110 00011010 00010111 01111010 00010111 10100011 00010111 01111010 00001000 00010110 10000011 00010110 11111010 00010110 01111010 00010111 01011010 00010111 01111010 00010111 01100011 00010110 10111010 00010110 00111010 00010110 01111010 00001000 00010110 10011010 00010111 11111010 00010110 10111010 00010111 11111010 00010110 00000011 00010111 10111010 00010110 11111010 00010110 01111010 01101000 00001000 00010110 10000011 00010110 11111010 00001000 00010110 10011010 00010111 00000011 00010110 10000011 00010111 00011010 00010111 10100011 00010111 01111010 00001000 00010110 00011010 00010111 00011010 00010110 00000011 00010111 11111010 00010110 10011010 00010110 10100011 00010111 01111010 00001000 00010110 00000011 00010110 00011010 00010110 00000011 00010111 10111010 00010110 00011010 00010110 10000011 00010110 11111010 00010110 01111010 00001000 00010110 00000011 00010110 00011010 00010110 00000011 00010111 11111010 00010111 01011010 00010111 10011010 00010110 00011010 00010110 10111010 00001001 00010110 11111010 00010110 01111010 00001001 00010110 00000011 00010111 10011010 00010110 11111010 00010110 11100011 00001000 00010110 00011010 00010111 11111010 00010110 01011010 00010111 10011010 00010110 10000011 00010111 11100011 00001000 00010110 11111010 00010111 01011010 00010111 00000011 00010111 01011010 00010111 01111010 00010111 10111010 00010110 10000011 11101001 00001000 00010110 01001010 00010110 11111010 00010110 01111010 00010111 00011010 00010110 00000011 00010110 00111010 00001000 00010110 11000011 00010111 10000011 00010111 10000011 00010110 01100011 00001000 00010110 11011010 00010111 00011010 00010110 01111010 00010110 01011010 00010111 10000011 00010110 01111010 00010111 11111010 00001000 00010110 01111010 00010110 00111010 00010110 10011010 00010111 00011010 00010110 10111010 00010111 00111010 00001000 00010110 00000011 00010111 10111010 00010110 10011010 00010110

```
11111010 00010110 11011010 00010111 01100011 00001000 00010110 11111010 00010110
10011010 00010111 00000011 00010110 11111010 00010110 01111010 00010110 01111010
00010111 10100011 00010111 01011010 00001001 00010110 00011010 00010111 01011010
00010111 10111010 00010111 10100011 00010111 01011010 00001001 00010110 01000011
00010111 10111010 00010111 11111010 00010110 11111010 00010111 01100011 00010110
11100011 00001001 00010110 00000011 00010111 01111010 00010111 01011010 00010111
10011010 00010111 00011010 01101000 00001000 00010110 11111010 00010110 00011010
00010111 00111010 00010110 10111010 00010111 01100011 00010110 01111010 00010111
11111010 00001000 00010110 00000011 00010111 10100011 00010111 11111010 00010111
00011010 00010110 10011010 00010110 00100011 00010110 00111010 00010110 01011010
00010111 00000011 00010111 11100011 00001001 00010110 01111010 00010111 00011010
00001000 00010110 10000011 00010110 00000011 00010110 11111010 00010110 10000011
00010110 10000011 00010111 00011010 00010110 00000011 00010110 10100011 01101001
00001000 00010110 01111010 00010111 00011010 00001000 00010110 10000011 00010111
10111010 00010111 00111010 00010110 11000011 00010110 10000011 01101001 00001000
00010110 01111010 00010111 00011010 00001000 00010110 01100011 00010111 10111010
00010110 00111010 00010110 11111010 00010111 00011010 00010110 11011010 00010110
00111010 01101000 00001000 00010110 10111010 00010111 11111010 00010110 00100011
00010110 00011010 00010110 01011010 00010110 01011010 00010111 00111010 00001001
00010110 00111010 00001000 00010110 11111010 00010111 00000011 00010110 11111010
00010110 01000011 00010111 11111010 00010110 11011010 00010110 00111010 00010110
01000011 11101001
```

№3

1) Метод Хаффмана:

Код программы:

```
#include "2.h"
#include "3.h"



// возвращает позицию элемента n в векторе a, если элемента нет в векторе, то
возвращае т -1.
// позиция элемента n - позиция элемента в векторе a, полем symbol которого
равно n
int getSymbolPosition(const std::vector<character> &a, const int n)
{
    for (int i = 0; i < a.size(); i++)
        for (int j = 0; j < a[i].symbol.size(); j++)
            if (a[i].symbol.at(j) == n)
                return i;
    return -1;
}

bool comp(const character &a, const character &b)
{
    return a.numbers > b.numbers;
}
```

```
// возвращает отсортированный по неувозрастанию вектор содержащий
// структуру character, полученную после обработки вектора table,
// содержащего двоичные последовательности, записанные целыми числами
// поле symbol - двоичная последовательность, записанная целыми числами
// поле numbers - количество раз, сколько последовательность встречается
// в векторе table, поле code - пустое
std::vector<character> getTable(const std::vector<int> &table)
{
    std::vector<character> res;
    for (auto &x : table)
    {
        int pos = getSymbolPosition(res, x);
        if (pos == -1)
            res.push_back(character(std::vector<int>{x}, 1,
std::vector<int>()));
        else
            res[pos].numbers++;
    }
    std::sort(res.begin(), res.end(), comp);

    return res;
}

void reverseVector(std::vector<int> &v)
{

    for (int i = 0; i < v.size() / 2; i++)
    {
        bool c = v[i];
        v[i] = v[v.size() - 1 - i];
        v[v.size() - 1 - i] = c;
    }
}

// метод Хаффмана
// возвращает таблицу, содержащую символ, его количество повторений, код
std::vector<character> theHuffmanMethod(const std::vector<int> &table)
{
    std::vector<character> res = getTable(table);
    std::vector<character> p = res;

    while (p.size() > 1)
    {
        int n = p.size() - 1;
        for (int i = 0; i < p[n].symbol.size(); i++)
        {
            int k = getSymbolPosition(res, p[n].symbol[i]);
            res[k].code.push_back(0);
        }

        n--;
        for (int i = 0; i < p[n].symbol.size(); i++)
```

```
        {
            res[getSymbolPosition(res, p[n].symbol[i])].code.push_back(1);
        }

        p[n].numbers += p[n + 1].numbers;
        for (int i = 0; i < p[n + 1].symbol.size(); i++)
            p[n].symbol.push_back(p[n + 1].symbol[i]);
        p.erase(p.cend());
        std::sort(p.begin(), p.end(), comp);
    }
    for (auto &x : res)
        reverseVector(x.code);

    return res;
}

void outputSymbolCodes(const std::vector<character> &a, int codeLength)
{
    for (character element : a)
    {
        std::cout << "Symbol: " << element.symbol[0] << " / ";
        std::vector<int> a(codeLength, 0);
        a = getBinaryNumberNotation(a, 0, element.symbol[0]);
        for (auto &x : a)
        {
            std::cout << x;
        }
        std::cout << "\nCode: ";

        for (int x : element.code)
        {
            std::cout << x;
        }
        std::cout << "\n\n";
    }
}

int main()
{
    std::string s;
    std::ifstream f0("T:\\2kurs2sem\\InformTheor\\lab3\\TI_3.txt");
    f0 >> s;
    f0.close();

    int length = 8;
    std::vector<int> r = getSequencesOfNCharactersEach(s, length);
    std::vector<character> res = theHuffmanMethod(r);
    outputSymbolCodes(res, length);
    return 0;
}
```

Результат работы программы:

По 2 символа:

```
Symbol: 2 / 01
Code: 11

Symbol: 1 / 10
Code: 10

Symbol: 0 / 00
Code: 01

Symbol: 3 / 11
Code: 00
```

По 4 символа:

```
Symbol: 8 / 0001
Code: 10

Symbol: 5 / 1010
Code: 111

Symbol: 6 / 0110
Code: 110

Symbol: 14 / 0111
Code: 010

Symbol: 12 / 0011
Code: 000

Symbol: 0 / 0000
Code: 0110

Symbol: 1 / 1000
Code: 0010

Symbol: 9 / 1001
Code: 01110

Symbol: 15 / 1111
Code: 00111

Symbol: 10 / 0101
Code: 011111

Symbol: 13 / 1011
Code: 011110

Symbol: 11 / 1101
Code: 0011011

Symbol: 7 / 1110
Code: 0011010

Symbol: 2 / 0100
Code: 0011001

Symbol: 4 / 0010
Code: 00110001

Symbol: 3 / 1100
Code: 00110000
```

По 8 символов:

```
Symbol: 104 / 00010110
Code: 10

Symbol: 232 / 00010111
Code: 111

Symbol: 16 / 00001000
Code: 0111

Symbol: 95 / 11111010
Code: 0110

Symbol: 88 / 00011010
Code: 0100

Symbol: 94 / 01111010
Code: 0011

Symbol: 192 / 00000011
Code: 0001

Symbol: 90 / 01011010
Code: 11011

Symbol: 93 / 10111010
Code: 0000

Symbol: 193 / 10000011
Code: 11001

Symbol: 92 / 00111010
Code: 01011

Symbol: 89 / 10011010
Code: 01010

Symbol: 144 / 00001001
Code: 110101

Symbol: 197 / 10100011
Code: 110100

Symbol: 91 / 11011010
Code: 110001

Symbol: 198 / 01100011
Code: 001010

Symbol: 199 / 11100011
Code: 001000
```

```
Symbol: 194 / 01000011
Code: 1100001

Symbol: 150 / 01101001
Code: 1100000

Symbol: 151 / 11101001
Code: 0010110

Symbol: 22 / 01101000
Code: 0010011

Symbol: 196 / 00100011
Code: 00101111

Symbol: 195 / 11000011
Code: 00101110

Symbol: 82 / 01001010
Code: 00100100

Symbol: 81 / 10001010
Code: 001001011

Symbol: 73 / 10010010
Code: 0010010
```

2) Метод Шеннона-Фано
Код программы:

```
#include "3_2.h"

std::vector<character> theShannonFanoMethod_(std::vector<character> a, int from,
int to)
{
    if (to - from > 1)
    {
        int mid;
        if (to - from == 2)
            mid = from + 1;

        else
        {
            int sum = 0;
            for (int i = from; i < to; ++i)
                sum += a[i].numbers;
            int halfOfSum = sum / 2;
            sum = 0;
            int i = from;
            while (sum < halfOfSum && i < to)
            {
                sum += a[i].numbers;
```

```
                i++;
            }
            int k1 = abs(sum - a[i].numbers - halfOfSum);
            int k2 = abs(sum - halfOfSum);

            if (k1 < k2)
                mid = i;
            else
                mid = i + 1;
        }

        for (int i = from; i < mid; i++)
            a[i].code.push_back(0);
        a = theShannonFanoMethod_(a, from, mid);

        for (int i = mid; i < to; i++)
            a[i].code.push_back(1);
        a = theShannonFanoMethod_(a, mid, to);
    }

    return a;
}

// метод Шеннона-Фано
// возвращает таблицу, содержащую символ, его количество повторений, код
std::vector<character> theShannonFanoMethod(const std::vector<int> &table)
{
    std::vector<character> res = getTable(table);
    res = theShannonFanoMethod_(res, 0, res.size());

    return res;
}

int main()
{
    std::string s;
    std::ifstream f0("T:\\2kurs2sem\\InformTheor\\lab3\\TI_3.txt");
    f0 >> s;

    f0.close();

    int length = 2;
    std::vector<int> r = getSequencesOfNCharactersEach(s, length);
    std::vector<character> res = theShannonFanoMethod(r);
    outputSymbolCodes(res, length);
    return 0;
}
```

Результат работы программы:

По 2 символа:

```
Symbol: 2 / 01
Code: 000

Symbol: 1 / 10
Code: 001

Symbol: 0 / 00
Code: 01

Symbol: 3 / 11
Code: 1
```

По 4 символа:

```
Symbol: 8 / 0001
Code: 000

Symbol: 5 / 1010
Code: 001

Symbol: 6 / 0110
Code: 01

Symbol: 14 / 0111
Code: 1000

Symbol: 12 / 0011
Code: 1001

Symbol: 0 / 0000
Code: 101

Symbol: 1 / 1000
Code: 11000

Symbol: 9 / 1001
Code: 11001

Symbol: 15 / 1111
Code: 1101

Symbol: 10 / 0101
Code: 11100

Symbol: 13 / 1011
Code: 11101

Symbol: 11 / 1101
Code: 111100

Symbol: 7 / 1110
Code: 111101
```

```
Symbol: 2 / 0100
Code: 1111100

Symbol: 4 / 0010
Code: 1111101

Symbol: 3 / 1100
Code: 111111
```

По 8 символов:

```
Symbol: 104 / 00010110
Code: 000

Symbol: 232 / 00010111
Code: 001

Symbol: 16 / 00001000
Code: 010

Symbol: 95 / 11111010
Code: 011

Symbol: 88 / 00011010
Code: 100000

Symbol: 94 / 01111010
Code: 100001

Symbol: 192 / 00000011
Code: 10001

Symbol: 90 / 01011010
Code: 1001

Symbol: 93 / 10111010
Code: 1010

Symbol: 193 / 10000011
Code: 1011

Symbol: 92 / 00111010
Code: 110000

Symbol: 89 / 10011010
Code: 110001

Symbol: 144 / 00001001
Code: 11001

Symbol: 197 / 10100011
Code: 11010

Symbol: 91 / 11011010
```

```
Code: 11011

Symbol: 198 / 01100011
Code: 1110000

Symbol: 199 / 11100011
Code: 1110001

Symbol: 194 / 01000011
Code: 111001

Symbol: 150 / 01101001
Code: 11101

Symbol: 151 / 11101001
Code: 1111000

Symbol: 22 / 01101000
Code: 1111001

Symbol: 196 / 00100011
Code: 111101

Symbol: 195 / 11000011
Code: 1111100

Symbol: 82 / 01001010
Code: 1111101

Symbol: 81 / 10001010
Code: 1111110

Symbol: 73 / 10010010
Code: 111111
```

№4

```
#include "4.h"

std::string
replaceCharactersWithTheirCodes(const std::vector<character> &table, const
std::vector<int> &s)
{
    std::vector<char> a;
    std::string res;
    for (auto &x : s)
    {
        int pos = getSymbolPosition(table, x);
        if (pos >= 0)
            for (auto &y : table[pos].code)
                res.push_back('0' + y);
    }
    return res;
}

int main()
{
    std::string s;

    std::ifstream f0("T:\\2kurs2sem\\InformTheor\\lab3\\TI_3.txt");
    f0 >> s;
    f0.close();

    int length = 2;
    std::vector<int> r = getSequencesOfNCharactersEach(s, length);
    std::vector<character> res = theHuffmanMethod(r);
    std::string sCode = replaceCharactersWithTheirCodes(res, r);
    std::cout << sCode;
    return 0;
}
```

Результат работы программы:

Метод Хаффмана

По 2 символа:

```
10111101100111101011110110101111001111010001011010111101101011110011110100101010
01100101101111010001010100111101101011101011110110100001101111010001010100111101
00010110101111011010111100111101101000101110101101100101101111011010111010111101
10100001101111011010110000111101101011001011110110101101101111011010001011101011
01100101101111011010001010111101000101010111101000110000011110100010111001111101
00010110101111011010111100111101101000101110010110111101101000010110010110111101
10101110001111010001011000111101101011111011110110101110011111011010001011100101
10111101101000110011110110101101011001011011110100010101101111011010110110111101
10101100001111011010001100111101000110100011110110101111111001011011110110100010
00111101101011011011110110101111101111010001101011101000011001011011110110100110
10111101101000000110010110111101101011001011110110101101101111011010001000111101
10100000101111011010110100111101101000110011110100011010001111011010001101100101
10111101000101101011110110100000101111011010001100111101101011110011110110100011
00111101000110111011110110100010101111011010000110111101101000110110010110111101
```

```
1010111000111101101000001011110110100010001111011010000010111101000101010011110
1101000101011110110100000101111011010001101101011011001011011110100010110101111
1011010000001100101101111011010111000111101000101011011110100010110001111011010
1101001111010001101000111101101000110110010110111101101011010011110110101101101
11101000101010011110110100000101111011010111010111101000110100011110110100011
01100101101111010001010110111101101011011011110100010101001111011010001010111
10110101101101111010001011010111101101000001011110110100011011001011011110100
01010110111101101011011011110100010101001111011010000000111101101011110011110
11010111010111101101011011011110110100010111001011011110110100000101111011010
00111110010110111101000101010011110110101110101111011010000010111101000110000
11001011011110110101101001111011010000010111101101011110011110110101110101111
01000101100011110100011000011001011011110110100000001111011010111100111101000
1010100111101101011110011110110100011001111011010001010111101000101101110100
00110010110111101101001111011110110100000101111011010001100111101101011011011
110100010101101111011010000101100101101111010001010000111101000101100011110
10001011010111101000110110110010110111101101011000011110110101101101111011010
00111011110110101111001111010001011010111101101000110011110110100000011001011
01111011010001110111101101000011011110110101110001111011010110110111101101000
10001111011010000101100101101111010001010100111101101000101011110110101110101
11101101000001011110110101100001111010001101101100101101111011010000010111101
10101110001111010001010110111101101000000101111011010001110111101101000110011
11010001101000111101101011111110010110111101101011010011110110101111001111011
010001000111101000110100011110110101111111001011011110100010111001111011010001
0001111011010000010111101101000000011110100011011101111010001100011100101101111
0100010101001111011010001100111101101011110011110110101110001111011010110101101
01101100101101111011010000010111101101011010011110110100001101111011010001000111
10100011011101111011010001100111101101000000110010110111101000101010011110100011
01000111101101000000001111011010110110111101101011101011110100011001101111011010
00011011110110101111001111010001010100111101000110001110010110111101101000110011
11011010110101100101101111010001011010111101000101011011110110100000101111010001
01101011110100010110001111011010110110111101000101011011110100011010111010110110
01011011110110100011001111011010110101100101101111010001011000111101101000100011
11011010000110111101000101001011110100010110111010110110010110111101101000110011
11011010110101100101101111010001101100111101101000101011110110100001101111011010
00000011110110101101101111011010110010111101101000001011010110110010110111101101
00010001111011010000010111101000110011011110110101101101111011010111110111101101
01111100111101101000011110010110111101101000001011001011011110110100000001111010
00101011011110110100000101111010001011100111101101000001011110110101100101111011
01000011011110100010111111101000
```

По 4 символа:

```
11010001100010111011010111011111010100000010110101110111110101000001100010011011
01000001100101011101111011010111000110100000110010100000001011010111011111010101
11011110011101100010011011010111011101101011100011010111001101101010111001101111
10101111011010111011110011101100010011011010111011110110100000110110100000011010
01010000001100101010000000101101011101111101010111011110011100110110101110000010
01101101011101110010100000010010101110111111101011101111101010111011110011100110
11010111101001010111100010011011010000011011010111101101011100110110101011101001
01000011101010111011111011100110110101110111100101011110110101110111111101000011
10111000110100010011011010111001011010111001110010011011010111001101111010111101
10101110111100101011100111110101111001010111010010100001110101011101000100110110
10000001011010111001111101011101001010111011111010101110100101000011011010111011
11011010111000110101110100010011011010111011100101011100111110101110111100101011
10011111010000001100101011101111101101011100111110101110100010110001001101101000
00010110110101110100010110001001101101000001011
```

```
01011100111001001101101011101110010100000110110100000010010101111001010000111010
10111010001001101101011110010101111011010000011001010111001111101011101110110100
00111010101110100010011011010000011011010111101101000001100101011101111011010111
10110100000010110101110011111010111010001001101101000001101101011110110100000110
01010111001110101011101111101010111011101101011110101011011110011100110110101011
10011111010111010011100110110100000110010101110111011010111001111101000000110100
01001101101011110010101110011111010111011111010101110111011010000001001010000001
10100010011011010111001110101011101111101010000011001010111011111010101110100101
01110111101101000000100111000110100010011011010111001100111010111001111010111101
00101011110111010000011011010111000001001101101000000110000010100000010010100000
10110100001100010011011010111001101101010111011010111010110101110111110101000000
01011010111010010101110011100100110110101110101101110101110001101011101110010101110
11010111011110010101110000010011011010000011001010111011111011010111011011010111
00111110101110011011010100001100010011011010111001111101011101110010100000110110
10111001111101011101011010111010010100001110101011101111101110011011010111100101
01110111110101011101111001010000111010101110111111011100110110100000011001010101
10111100101011100111110101110011101010000110110100000011010011100110110100000110
01010111010010101110111110101011101110010101111000101100010011011010111001111101
01111001010111000110101110111100101000011011010111010010101110011100100110110100
00011001010000111010101110011101010111101101011101110110100000011000111010111000
11010111011111010100000110010100000011010011100110110101110100101011110001001101
10100000010110100000110110101110011111010000001011010000001001010111101101000001
10110100001110111011000100110110101110100101011110001001101101000000100101011101
11100101011100011010000001100001101000000100111011000100110110101110100101011110
00100110110100001100101011101111011010111000110101110011101010111101101011100110
11110101110000010110001001101101011101111001010111001111101000000110001110101111
01101011101111111010111011111010101110000111001101101011100000100110110101110011
10101000001101101011100111110100000011001010101110011111010111001101111010111000
11010000001100101110001101
```

По 8 символов:

```
10001001010101101111111001101101111100010111100001111010101001011100001111110011
01101111100001100000011110010101001011101100011111100011001001000001100000011110
00001000011000100011111000011111100110110111110000110101100101101111001010111110
01111110111011011111000011010110001111101000111100001100100101100011110011111110
10011111011110101100000111010010110111011010000101100111100010010111001100111101
10001100100100000111011010010011100111111101001110011011110110011001101000111111
10111110011111001010100000100101110001101111001010111011010000011101101000011110
00010011010001100100110111110110011001100111100101011100011011001111010011111010
11100110111100100111010010000111101101001010101101001110011011110000110010010000
11110000100100101100110011010001101111000011001001000011111011011111011111010101
01001000001101011001101000111101011000011110101010011010001000011110010011101101
01101111101010101100111100100001111001101111101111100011111101111001111100001010
10010010110011110001001001001101000111110100100001100101101111000101110111110011
11110011000101001111011000111101001000111011011111110011000111111011001111000111
01011100101011101001000001110101101111000011110000100101010011010110001111001010
01111001101001010111000110011010001110001111111010011111011110101100100111110111
11000011111010011111011110101101100001111000011101101001101110010101000100011010
11000011110011111110111101010111010000100110111100110100100111010111000000111001
01010001111101100111100001111111010011101101110100100101010001011111001011101011
11100011110010001101011000111111010001111011001100001100110101100110110011110100
00001101101001100000011110001111101000111101100111100001110101110001011101011001
11000000111100011111010001111000101011100001001011100110111010010110001100101100
```

```
1001101111000001110110100010111110010010110111011011111010111101011001011011110
0110111000110011010110000111101101011000110010111011000010010110
```

1) Метод Шеннона-Фано

По 2 символа:

```
00100000001001010000010010000000100100100000001000000011010100100100000000100100
10000001000000011010101010010101001000000011010101100000001001001000001001000000001
00100110100100000001101010110000000110101001001000000010010010000000100000000100
10011001000000100100001001010100100000000100100100000100100000000100100110100100000000
10010010001100000001001001000100100000000100100100001001000000001001001100100000010
01000010010101001000000010010011001001000000011010101001000000011010011100000001
10101000100000001101010010010000000100100100000001000000010010011001000000101010010
00000001001001101010010101001000000010010010000011000000011010100110000000100100
10000000010000000100100100000001000000010010011001000000101010010000000100100110000
10000000100100100001010010101001000000011010101001000000010010010000010010000000001
00100100011000000010010011000100000001101001001100000001001001000000000001010100
10000000100100110011000000010010010000100100000000100100100000000100000000110100100
01000000100110100101010010000000100100101001001000000010010011101001010100100000000
01001001000100100000000100100100001001000000010010011001100000001001001110010000000
00100100100001100000001001001100010000000110100100110000000100100110000100101010
01000000011010100100100000000100100111001000000010010011000100000000100100100000001
00000000100100110001000000011010010000010000000010010011001001000000010010011010010
0000000100100110000100101010010000000010010010000011000000010010011100100000000100
10011001100000001001001110010000000110101011000000010010011001001000000010010011
10010000000100100110000100100100001001010100100000001101010010010000000100100111
01001010100100000000100100100000110000000110101010010000000110101001100000000100100
01000011000000011010010011000000010010011000010010101001000000010010010000110000
00010010010000100100000001101010110000000100100111001000000010010010000010010000
00011010010011000000010010011000010010101001000000011010101001000000010010010000
10010000000110101011000000010010011001001000000010010010000100100000000110101001
01000000010010011100100000001001001100001001010100100000001101010100100000001001
00100000100100000001101010110000000100100111100000001001001000000010000000010010010
00001001000000010010010000100100000001001001100100000010101001000000001001001110010
00000001001001100000000101010010000000110101011000000010010010000010010000000100
10011100100000001101001101001010100100000001001001000011000000010010011100100000
00100100100000010000000010010010000010010000000110101001100000001101001101001010101
00100000000100100111100000001001001000000010000000110101011000000010010010000000100
00000100100110001000000010010011001001000000011010100100000100110100101010010000
00010010010100000010000000100100111001000000010010011000100000000100100100000100100
00000110101010010000000010010011010100101010010000000110101110000000110101001100
00001101010010010000000110100100001001010100100000001001001000110000000100100100
00100100000000100100110000010000000100100100000001000000011010100100100000000100100
11000100000000100100111010010101001000000010010011000001000000010010011010010000000
00100100100000110000000100100100001001000000001001001100110000000100100110101001
10100100000000110101011000000010010011001001000000010010010000010010000000010010011
11001000000000100100100011000000011010010000100101010010000000010010011100100000001
00100100000011000000011010101001000000010010011100100000000100100110000010000000010
01001100010000000110100100110000000100100100000000000101010010000000010010010000
10000000100100100000001000000010010011001100000001101001001100000000100100100000000
00001010100100000001101010001000000010010011001100000001001001110010000000100100
11110000000110100100000100000000110100110000010101001000000011010101100000000100100
01100010000000100100100000001000000010010010000011000000010010010000101001001000000
```

```
10010101001000000010010011100100000001001001000011000000010010011010010000000100
10011001100000001101001000001000000010010011000100000001001001110100101010010000
00011010101100000001101001001100000001001001111000000010010010000100100000001001
00100000100100000001101001010010000000100100110100100000001001001000000100000001
10101011000000011010011000001010100100000010010011000100000001001001000010100010
10100100000001101010010010000000110101010010000000100100111001000000011010100100
10000000110101001100000001001001000010010000000110101010010000000110100100100000
10010000100101010010000000100100110001000000010010010000101001010100100000001101
01001100000001001001100110000000100100111010010000000110101100100000001101010100
00010010000100101010010000000100100110001000000010010010000101001010100100000001
10100100010000000100100110010010000000100100111010010000000100100111100000001001
01000010010000000100100100010010000000100100110101001001000010010101001000000010
01001100110000000100100111001000000011010010100100000001001001000010010000000100
10010000000010000000100100100000010000000100100111010000010101001000000100100110
10100101010010000000100100111100000001101010100100000001001001110010000000110101
00010000000100100111001000000010010010001001000000010010011010010000000110101000
0000010011
```

По 4 символа:

```
01000111110111001010000011110010000001001110000100000111100100000010011011100010
10100010011011000000001110010100000110010100010011011000000100111000010000011110
01000000001111011100101110001010100000111001010000011001010000011111001000000001
11110001000001000010000011110111001011100010101000001111010100010011010100010011
11101100000010011111100100000010011100001000001111001000000001111011100110101000
00110011100010101000001110011000000100111000100000000111100010000011110010000000
01111011100110101000001100010000000010001100010101000100110101000001000010000011
11100100000000110001000000100100110000000011110011001101010000011110110000000010
00010000011110001000100100111001111011100010101000001110000100000111011100010100
10000011111000100000100001000001111011000000001110101000001000100000000110001000
00010010011000000001100011000101010001001110000100000111010100000110001000000001
11100100000000110001000000100101010000011110101000001100101000001100011000101010
00001110011000000001110101000001111011000000001110101000100110111000000000111101
00000111010100000110001100001110001010100010011100001000001110111000101010000011
10011000000100110101000100111000100000000100010000001001001100000000110001100010
10100000100010000000010000100010011011000000001110101000001110010100010010011000
00000110001100010101000100110101000001000010001001101100000000111010100000100000
10001001110000100000111010100000110001100010101000100110101000001000010001001101
10000000011101100000000111100100000000111001010000010000100000111101110011010100
00011101010000011000110011010100010011011000000001110010100000111010100010011111
01110001010100000100010000000011101010000011110010000000011100101000100111000100
00001001111101110001010100000111011000000001111001000000100110110000000011110010
00000001100010000000011110101000100111000110011111011100010101000001111110001000
00111010100000110001000000010000100010011010100000110011100010101000100111111111
00000010011100010000001001110000100010010111000101010000011111001000000001000010
00001100001000001111001000000100111000010000011000100000000111011100010101000001
10000100000110010100000111001100000000100001000001111011000000001100111000101010
00100110110000000011111010100000111001010000011101010000011111001000000100101110
01010100000111010100000111001100000010011010100000111010100000110000100000110001
00000010010011000000001111001100110101000010011111100100000000111101100000000111010
00001001001100000000111100110011010100010011111100100000000111101100000000111010
10000011101100000010010101000100111110111001101010001001101100000000110001000000
```

```
00111100100000000111001100000000100011000011100010101000001110101000001000100000
00011001010000011110110000001001010100000110001000000011101110001010100010011101
10000001001001100000000111011000000001000010000011100101000100111111010100000110
01010000011110010000010011011000000100111110111001101010000011000100000000100001
10001010100010011100001000100110101000001110101000100111000010001001110001000000
00100001000100110101000100100111001011100010101000001100010000000010001100010101
00010011100010000000011110110000000001100101000100111111101000100111000110010111
00101010000011000100000000100011000101010001001011000000001111010100000110010100
00011101100000000100001000001111100010000011001110000111000101010000011110110000
00001110101000100111111010100000100001000001111000100000111100100000000110011100
11010100000110011100010101000001110110000001001101010000011101010001001111110010
00000001110101000001111100010000011001010001001111110011001111101
```

По 8 символов:

```
00011111110001001001101100010010011000101000010001001110001000110000000100010011
01100010010011010111010100001100010001100000001101100111011000100000000101011101
01000010100001000100011100010011110010011011000100100110101100100011000001000011
00010011011001100100010010011010110010001000010011000000100001000100010000000011
01100110000100111010001100111001000101000110000000010010001101011110000100001111
11000001101000011011000100000000101000101100010000000110000100111010001100001010
00010110000110001000010011001001100001001111000000010100001100000001000010100001
10001001011000101000101100010001001101000011000100001111100101000010110000110100
00011000100110001000101100110000000111010001100001010000100000011000000001000100
01011000110001000110100011000010100001000100010000000010001001101000010000000010
11000011000100001010000100010001000000001000100101100110010011100010001000000001
01011001000011000100001110010001000100111000100001100011100010100001000000010110
00100100111000100010110011111000101000001100110010011000100110010011000010011010
00101111110000100001111101000011000100001001100000000100010001100000100001111100
00110110011011000111000001000011011001100000000100001000100100110110001000010010
11010000100001000110000000110001001100000000010100011100000100001000100110100011
00010000110001101100111100000100000110001100010011000100001100010000100010000100
11101000110011100100010000000110010011010001110100011001110010001110010011010001
01100001100111100000001110001110010001000100110000100110010011100010011000001111
00101000001100010000000111000000010100011110000000100001001011010000100010011101
00010110011000000001100010001111010001100000001001001100010011110001110010001000
01001100000010000101100010001000011000101100010110011000000001000100011010111010
10000100001001100000010000101100110100011100000001111100000101111101010000100001
00110000001000011100000001101000011000000001100110000000011011000110000111100101
00010100010110001111010001000000001001000100100111000011001000110000010000011001
10001000011000111001001011000110110001100000001110011111000
```

№5

Код программы:

```
#include "5.h"


double getCompressionRatio(std::string s, const std::vector<character> &table)
{
```

```
    int numberOfSymbols = s.size();
    int b = numberOfSymbols;

    int b0 = 0;
    for (auto &x : table)
        b0 += x.numbers * x.code.size();
    double res = (double)b / b0;
    return res;
}

int main()
{
    std::string s;
    std::ifstream f0("T:\\2kurs2sem\\InformTheor\\lab3\\TI_3.txt");
    f0 >> s;
    f0.close();

    int length = 2;
    std::vector<int> r = getSequencesOfNCharactersEach(s, length);

    clock_t start_time = clock();
    std::vector<character> res = theHuffmanMethod(r);
    clock_t end_time = clock();
    clock_t work_time = end_time - start_time;

    std::cout << "Time: " << (double)work_time;
    std::cout << "\nCompression ratio: " << getCompressionRatio(s, res);

    return 0;
}
```

Результат работы программы:

1) Метод Хаффмана

По 2 символа:

```
Time: 0
Compression ratio: 1
```

По 4 символа:

```
Time: 0
Compression ratio: 1.2458
```

По 8 символов:

```
Time: 0
Compression ratio: 2.165
```

2) Метод Шеннона-Фано

По 2 символа:

```
Time: 0
Compression ratio: 0.815842
```

По 4 символа:

```
Time: 0
Compression ratio: 1.17546
```

По 8 символов:

```
Time: 0
Compression ratio: 2.00097
```

Время работы программы:

Метод построения кода	Количество символов в последовательности, взятой в качестве кодируемого символа		
	2	4	8
Метод Хаффмана	0	0	0
Метод Шеннона-Фано	0	0	0

Коэффициент сжатия:

Метод построения кода	Количество символов в последовательности, взятой в качестве кодируемого символа		
	2	4	8
Метод Хаффмана	1	1,245	2.165
Метод Шеннона-Фано	0.815	1,175	2

Вывод:

По результатам, полученным в ходе работы программы и приведенным в таблицах выше, можно сделать следующие выводы. При анализе времени выполнения явного преобладания в скорости нет, так как язык C++ достаточно быстрый, но, если судить по сложности кода алгоритмов, алгоритм Шеннона-Фано будет эффективнее. Также мы сравнили коэффициенты сжатия. По данному показателю метод Хаффмана более эффективен, коэффициенты сжатия методом Хаффмана - больше, чем при использовании метода Шеннона-Фано. Сложность программной реализации обоих алгоритмов примерно одинаковая - средняя. Вручную же алгоритм Шеннона-Фано выполняется в более компактном и, следовательно, более удобном виде. Сложности выполнения этих алгоритмов вручнуютакже примерно одинаковы. Таким образом, можно сделать вывод, что алгоритм Хаффмана более эффективен, чем метод Шеннона-Фано. А значит, лучше использоватьметод Хаффмана, т.к. по скорости и простоте выполнения алгоритмы очень схожи.

№6

Код программы:

```
#include "6.h"

bool areVectorsEqual(std::vector<int> a, std::vector<int> b)
{
    if (a.size() != b.size())
        return false;
    for (int i = 0; i < a.size(); i++)
        if (a[i] != b[i])
            return false;
    return true;
}

int getPosOfTheVector(const std::vector<character> &table, const
std::vector<int> &a)
{
    for (int i = 0; i < table.size(); i++)
        if (areVectorsEqual(table[i].code, a))
            return i;
    return -1;
}
```

```
std::string decoding(std::string codingS, std::vector<character> table, int
length)
{
    std::string res;
    std::vector<int> a;
    for (auto &x : codingS)
    {
        a.push_back(x - '0');
        int pos = getPosOfTheVector(table, a);
        if (pos >= 0)
        {
            std::vector<int> b(length, 0);
            b = getBinaryNumberNotation(b, 0, table[pos].symbol[0]);
            reverseVector(b);
            for (auto &y : b)
                res.push_back(y + '0');
            a.clear();
        }
    }
    return res;
}

int main()
{
    std::string s;
    std::ifstream f0("T:\\2kurs2sem\\InformTheor\\lab3\\TI_3.txt");
    f0 >> s;
    f0.close();

    int length = 2;
    std::vector<int> r = getSequencesOfNCharactersEach(s, length);

    std::vector<character> res1 = theShannonFanoMethod(r);
    std::string sCode1 = replaceCharactersWithTheirCodes(res1, r);
    std::string s1 = decoding(sCode1, res1, length);

    std::vector<character> res2 = theHuffmanMethod(r);
    std::string sCode2 = replaceCharactersWithTheirCodes(res2, r);
    std::string s2 = decoding(sCode2, res2, length);
    if (s1 == s2)
        std::cout << "YES!!!\n";
    else
        std::cout << "NO!!\n";

    return 0;
}
```

Результат работы программы:

По 2 символа:

```
YES!!!
```

По 4 символа:

```
YES!!!
```

По 8 символов:

```
YES!!!
```

№7

Расшифровка кода через сайт:

Ветер свистел, визжал, кряхтел и гудел на разные лады. То жалобным тоненьким голоском, то грубым басовым раскатом распевал он свою боевую песенку. Фонари чуть заметно мигали сквозь огромные белые хлопья снега, обильно сыпавшиеся на тротуары, на улицу, на экипажи, лошадей и прохожих.

Вывод: в ходе работы изучены возможности применения методов энтропийного кодирования для обработки двоичных последовательностей. Получены навыки написания и отладки программы составления кода для каждого символа сообщения методом Хаффмана и методом Шеннона-Фано, кодирования и декодирования двоичной последовательности. Сравнены время работы программы и коэффициенты сжатия.