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
Игнатьев Артур Олегович

Проверил:
Твердохлеб Виталий Викторович

Белгород 2024г.

Лабораторная работа №3

«Исследование возможности применения методов энтропийного кодирования для обработки двоичных последовательностей»

Цель 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 Хаффмана и метода Шеннона-Фано.

Задания

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

Задание №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.

Задание №2

Код программы:

task_2.txt

```
11010000100100101101000010110101110100011000001011010000101101011101000110000
00000100000110100011000000111010000101100101101000010111000110100011000000111
01000110000010110100000101101011101000010111011001011000010000011010000101100 1
01101000010111000110100000101101111101000010110110110100000101100001101000010111
10110010110000100000110100001011101011010001100000001101000110001111110100011
00001011101000110000010110100000101101011101000010111011001000001101000010111 0
00001000001101000010110011110100011000001111010000101101001101000010110101110
10000101110110010000011010000101111011101000010110000001000001101000110000000
11010000101100001101000010110111110100001011110111010001100010111010000101100
10100100000110100001011101111010000101100001101000010110100110100011000101100
10111000100000110100001010001011010000101111100010000011010000101101101101000
01011000011010000101110111110100001011111011010000101100011101000010111101110 1
00011000101111010000101111000010000011010001100000101101000010111110110100001
01111011101000010110101110100001011110111010001100011001101000010111010110100
00101110001101000010111100001000001101000010110011110100001011111011010000101
11011110100001011111011010001100000011101000010111010110100001011111011010000
10111100001011000010000011010001100000101101000010111110001000001101000010110
01111010001100000001101000110000011110100001011000111010001100010111101000010
11110000100000110100001011000111010000101100001101000110000001110100001011111
01101000010110010110100011000101111010000101111000010000011010001100000001101
00001011000011010001100000011101000010111010110100001011000011010001100000101
10100001011111011010000101111000010000011010001100000001101000010110000110100
01100000011101000010111111110100001011010111010000101100101101000010110000110
10000101110110010000011010000101111011010000101111010010000011010001100000001
11010000101100101101000010111110110100011000111000100000110100001011000111010
00010111110110100001011010111010000101100101101000110000011110100011000111000
10000011010000101111111101000010110101110100011000000111010000101101011101000
01011110111010000101110101101000110000011001011100010000011010000101001001101
00001011111011010000101111011101000010110000110100011000000011010000101110000
01000001101000110000111110100011000001111010001100000101101000110001100001000
00110100001011011111010000101100001101000010111100110100001011010111010001100
00010110100001011110111010000101111100010000011010000101111001101000010111000
```

```
110100001011001111010000101100001101000010111011110100001011100000100000011010
001100000001110100001011101011010000101100101101000010111110110100001011011111
010001100011000010000011010000101111101101000010110011110100011000000001101000
010111110110100001011110011010000101111011101000110001011110100001011010100010
000011010000101100011101000010110101110100001011101111010001100010111101000001
011010100100000110100011000010111010000101110111101000010111110110100001011111
111101000110001100110100011000111100100000110100011000000111010000101111101110
100001011010111010000101100111101000010110000000101100001000001101000010111110
110100001011000111010000101110001101000010111011110100011000110011010000010111
101110100001011111100010000011010001100000011101000110001011110100001011111111
010000101100001101000010110010110100011000100011010000101110001101000010110101
111010001100000001110100011000111100100000110100001011110111010000101100000010
000011010001100000101101000110000000110100001011111011010001100000101101000111
000001111010000101100001101000110000000110100011000101100101100001000001101000
001011110111010000101100000001000001101000110000011110100001011101111010000101
110001101000110000110110100011000001100101100001000001101000010111101110100000
101100000001000001101000110001101110100001011101011010000101110001101000010111
111110100001011000011010000101101101101000010111000000101100001000001101000010
111011110100001011111011010001100010001101000010110000110100001011010011010000
010110101110100001011100100100000110100001011100000010000011010000101111111101
000110000000011010000101111101101000110000101110100001011111011010000101101101
1010000101110001101000110000101001011110
```

task2.h

```
#ifndef INFORMATION_THEORY_TASK2_H
#define INFORMATION_THEORY_TASK2_H

#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
#include <windows.h>

void outputVectorSequencesForReading(const std::vector<int> &r, const int
&length);

std::vector<int> getBinaryNumberNotation(std::vector<int> &a, int i, int n);

void outputVector(const std::vector<int> &r);

std::vector<int> getSequencesOfNCharactersEach(const std::string &s, const
int n);

int getNumberFromCharVector(const std::vector<char> &a);

#endif //INFORMATION_THEORY_TASK2_H
```

tast2.cpp

```
#include "task2.h"

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';
}
```

main2.cpp

```
#include "../../libs/alg/labs/lab3/task2.h"

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(CP_UTF8);
    setlocale(LC_ALL, "Russian");

    std::string s;
    std::ifstream f0("C:\\Users\\Artur\\Projects\\C++\\information_the-
ory\\labs\\lab3\\task_2.txt");
    f0 >> s;
```

```
    f0.close();

    int length = 2;
    std::vector<int> r = getSequencesOfNCharactersEach(s, length);
    std::cout << "В памяти компьютера:\n";

    outputVector(r);

    std::cout << "\nДля чтения:\n";
    outputVectorSequencesForReading(r, length);

    return 0;
}
```

Результат работы по 2 символа:

```
В памяти компьютера:
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

Для чтения:
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 11
```

Результат работы по 4 символа:

```
В памяти компьютера:
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 1
4 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

Для чтения:
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
```

Результат работы по 8 символов:

```
В памяти компьютера:
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 1
93 104 193 232 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

Для чтения:
00010110 10010010 00010110 01011010 00010111 10000011 00010110 01011010
00010111 00000011 00001000 00010110 00000011 000
10111 10011010 00010110 00111010 00010110 00000011 00010111 10000011 00010110
01011010 00010111 10111010 01101001 000010
00 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 000
10111 10111010 00001001 00010110 00111010 00001000 00010110 10011010 00010111
10000011 00010111 01011010 00010110 010110
10 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 010
11010 00010110 10100011 11101001 00001000 00010110 10001010 00010110 11111010
00001000 00010110 11011010 00010110 000110
10 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 101
11010 00010110 00111010 00010110 01111010 00001000 00010110 10011010 00010111
11111010 00010110 10111010 00010111 111110
10 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 011
11010 00001000 00010110 00011010 00010111 00011010 00010110 00000011 00010111
11111010 00010110 10011010 00010110 101000
11 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 000
10111 11111010 00010111 01011010 00010111 10011010 00010110 00011010 00010110
10111010 00001001 00010110 11111010 000101
10 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 000
10111 01011010 00010111 00000011 00010111 01011010 00010111 01111010 00010111
10111010 00010110 10000011 11101001 000010
00 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 000
10110 01111010 00010110 01011010 00010111 10000011 00010110 01111010 00010111
11111010 00001000 00010110 01111010 000101
10 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 100
11010 00010111 00000011 00010110 11111010 00010110 01111010 00010110 01111010
00010111 10100011 00010111 01011010 000010
01 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 000
10111 01111010 00010111 01011010 00010111 10011010 00010111 00011010 01101000
00001000 00010110 11111010 00010110 000110
10 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 010
11010 00010111 00000011 00010111 11100011 00001001 00010110 01111010 00010111
00011010 00001000 00010110 10000011 000101
10 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 000
10110 11000011 00010110 10000011 01101001 00001000 00010110 01111010 00010111
00011010 00001000 00010110 01100011 000101
11 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 001
11010 00001001 00010110 00111010 00001000 00010110 11111010 00010111 00000011
00010110 11111010 00010110 01000011 000101
11 11111010 00010110 11011010 00010110 00111010 00010110 01000011 11101001
```

Задание №3

Метод Хаффмана

Код программы:

task3_1.h

```
#ifndef INFORMATION_THEORY_TASK3_1_H
#define INFORMATION_THEORY_TASK3_1_H

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

#include "task2.h"

class character {
public:
    std::vector<int> symbol;
    int numbers;
    std::vector<int> code;

    character() {
        symbol = std::vector<int>();
        code = std::vector<int>();
    }

    character(std::vector<int> symbol, int numbers, std::vector<int> code) {
        this->symbol = symbol;
        this->numbers = numbers;
        this->code = code;
    }
};

void outputSymbolCodes(const std::vector<character> &a, int codeLength);

std::vector<character> theHuffmanMethod(const std::vector<int> &table);

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

std::vector<character> getTable(const std::vector<int> &table);

int getSymbolPosition(const std::vector<character> &a, const int n);
// std::vector<int> getBinaryNumberNotation(std::vector<int> &a, int i, int
n);

#endif //INFORMATION_THEORY_TASK3_1_H
```

task3_1.cpp

```
#include "task3_1.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::vec-
tor<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 << "Символ: " << 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 << "\nКод: ";

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

main3_1.cpp

```
#include "../../libs/alg/labs/lab3/task3_1.h"

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(CP_UTF8);
    setlocale(LC_ALL, "Russian");

    std::string s;
    std::ifstream f0("C:\\Users\\Artur\\Projects\\C++\\information_the-
ory\\labs\\lab3\\task_2.txt");
    f0 >> s;
    f0.close();

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

Работа программы по 2 символа:

```
Символ: 2 / 01
Код: 11

Символ: 1 / 10
Код: 10

Символ: 0 / 00
Код: 01

Символ: 3 / 11
Код: 00
```

Работа программы по 4 символа:

```
Символ: 8 / 0001
Код: 10

Символ: 5 / 1010
Код: 111

Символ: 6 / 0110
Код: 110

Символ: 14 / 0111
Код: 010

Символ: 12 / 0011
Код: 000

Символ: 0 / 0000
Код: 0110

Символ: 1 / 1000
Код: 0010

Символ: 9 / 1001
Код: 01110

Символ: 15 / 1111
Код: 00111

Символ: 10 / 0101
Код: 011111

Символ: 13 / 1011
Код: 011110

Символ: 11 / 1101
Код: 0011011

Символ: 7 / 1110
Код: 0011010

Символ: 2 / 0100
Код: 0011001

Символ: 4 / 0010
Код: 00110001

Символ: 3 / 1100
Код: 00110000
```

Работа программы по 8 символов:

```
Символ: 104 / 00010110
Код: 10

Символ: 232 / 00010111
Код: 111

Символ: 16 / 00001000
Код: 0111

Символ: 95 / 11111010
Код: 0110

Символ: 88 / 00011010
```

```
Код: 0100

Символ: 94 / 01111010
Код: 0011

Символ: 192 / 00000011
Код: 0001

Символ: 90 / 01011010
Код: 11011

Символ: 93 / 10111010
Код: 0000

Символ: 193 / 10000011
Код: 11001

Символ: 92 / 00111010
Код: 01011

Символ: 89 / 10011010
Код: 01010

Символ: 144 / 00001001
Код: 110101

Символ: 197 / 10100011
Код: 110100

Символ: 91 / 11011010
Код: 110001

Символ: 198 / 01100011
Код: 001010

Символ: 199 / 11100011
Код: 001000

Символ: 194 / 01000011
Код: 1100001

Символ: 150 / 01101001
Код: 1100000

Символ: 151 / 11101001
Код: 0010110

Символ: 22 / 01101000
Код: 0010011

Символ: 196 / 00100011
Код: 00101111

Символ: 195 / 11000011
Код: 00101110

Символ: 82 / 01001010
Код: 00100100

Символ: 81 / 10001010
Код: 001001011

Символ: 73 / 10010010
Код: 001001010
```

Метод Шенона-Фано

Код программы:

task3_2.h

```
#ifndef INFORMATION_THEORY_TASK3_2_H
#define INFORMATION_THEORY_TASK3_2_H

#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>
#include <cmath>
#include <windows.h>

#include "task3_1.h"

std::vector<character> theShannonFanoMethod(const std::vector<int> &table);

#endif //INFORMATION_THEORY_TASK3_2_H
```

task3_2.cpp

```
#include "task3_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;
}
```

main3_2.cpp

```
#include "../../libs/alg/labs/lab3/task3_2.h"

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(CP_UTF8);
    setlocale(LC_ALL, "Russian");

    std::string s;
    std::ifstream f0("C:\\Users\\Artur\\Projects\\C++\\information_the-
ory\\labs\\lab3\\task_2.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 программ по 2 символа

```
Символ: 2 / 01
Код: 000

Символ: 1 / 10
Код: 001

Символ: 0 / 00
Код: 01

Символ: 3 / 11
Код: 1
```

Результат работы программы по 4 символа:

```
Символ: 8 / 0001
Код: 000

Символ: 5 / 1010
Код: 001

Символ: 6 / 0110
```

```
Код: 01

Символ: 14 / 0111
Код: 1000

Символ: 12 / 0011
Код: 1001

Символ: 0 / 0000
Код: 101

Символ: 1 / 1000
Код: 11000

Символ: 9 / 1001
Код: 11001

Символ: 15 / 1111
Код: 1101

Символ: 10 / 0101
Код: 11100

Символ: 13 / 1011
Код: 11101

Символ: 11 / 1101
Код: 111100

Символ: 7 / 1110
Код: 111101

Символ: 2 / 0100
Код: 1111100

Символ: 4 / 0010
Код: 1111101

Символ: 3 / 1100
Код: 111111
```

Результат работы программы по 8 символов:

```
Символ: 104 / 00010110
Код: 000

Символ: 232 / 00010111
Код: 001

Символ: 16 / 00001000
Код: 010

Символ: 95 / 11111010
Код: 011

Символ: 88 / 00011010
Код: 100000

Символ: 94 / 01111010
Код: 100001

Символ: 192 / 00000011
Код: 10001
```

```
Символ: 90 / 01011010
Код: 1001

Символ: 93 / 10111010
Код: 1010

Символ: 193 / 10000011
Код: 1011

Символ: 92 / 00111010
Код: 110000

Символ: 89 / 10011010
Код: 110001

Символ: 144 / 00001001
Код: 11001

Символ: 197 / 10100011
Код: 11010

Символ: 91 / 11011010
Код: 11011

Символ: 198 / 01100011
Код: 1110000

Символ: 199 / 11100011
Код: 1110001

Символ: 194 / 01000011
Код: 111001

Символ: 150 / 01101001
Код: 11101

Символ: 151 / 11101001
Код: 1111000

Символ: 22 / 01101000
Код: 1111001

Символ: 196 / 00100011
Код: 111101

Символ: 195 / 11000011
Код: 1111100

Символ: 82 / 01001010
Код: 1111101

Символ: 81 / 10001010
Код: 1111110

Символ: 73 / 10010010
Код: 1111111
```

Задание №4

Код программы:

task4.h

```
#ifndef INFORMATION_THEORY_TASK4_H
#define INFORMATION_THEORY_TASK4_H

#include <string>

#include "task3_1.h"
#include "task3_2.h"

std::string replaceCharactersWithTheirCodes(const std::vector<character> &ta-
ble, const std::vector<int> &s);

#endif //INFORMATION_THEORY_TASK4_H
```

task4.cpp

```
#include "task4.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;
}
```

main4.cpp

```
#include "../../libs/alg/labs/lab3/task4.h"

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(CP_UTF8);
    setlocale(LC_ALL, "Russian");

    std::string s;

    std::ifstream f0("C:\\Users\\Artur\\Projects\\C++\\information_the-
ory\\labs\\lab3\\task_2.txt");
    f0 >> s;
    f0.close();

    int length = 8;
    std::vector<int> r = getSequencesOfNCharactersEach(s, length);
    std::vector<character> res = theShannonFanoMethod(r);
    std::string sCode = replaceCharactersWithTheirCodes(res, r);
    std::cout << sCode;
```

```
    return 0;
}
```

Метод Хаффмана

Результат работы программы по 2 символа:

```
1011110110011110101111011010111100111101000101101011110110101111001111010001
1010110010110111101000101010011110110101110
1011110110100001101111010001010100111101000101101011110110101111001111011010
0101110101101100101101111011010111010111101
1010000110111101101011000011110110101100101111011010110111101101000101110 1
0110110010110111101101000101011110100010101
1011110100011000001111010001011100111101000101101011110110101111001111011010
0101110010110111101101000010110010110111101
1010111000111101000101100011110110101111011110110101111001111011010001011100
1011011110110100011001111011010110101100101
1011110100010101101111011010110110111101101011000011110110100011001111010001 1
0100011110110101111111001011011110110100010
0011110110101101101111011010111110111010001101011101000011001011011110110100
1101011110110100000011001011011110110101100
1011110110101101101111011010001000111101101000001011110110101101001111011010
0110011110100011010001111011010001101100101
1011110100010110101111011010000010111101101000110011110110101111001111011010
0110011110100011011101111011010001010111101
1010000110111101101000110110010110111101101011000111101101000001011110110100
0100011110110100000101111010001010100111101
1010001010111101101000001011110110100011011010110110010110111101000101101011
1011010000001100101101111011010111000111101
0001010110111101000101100011110110101101001111010001101000111101101000110110 0
1011011110110101101001111011010110110111101
0001010100111101101000001011110110101110101111010001101000111101101000110110 0
1011011110100010101101111011010110110111101
0001010100111101101000101011110110101110101111010001011010111101101000001011
1011010001101100101101111010001010110111101
1010110110111101000101010011110110100000011110110101111001111011010111010111
1011010110110111101101000101110010110111101
1010000010111101101000111110010110111101000101010011110110101110101111011010
0001011110100011000011001011011110110101100
0011110110100000101111011010111000111101000101011011110110100000101111011010
0001110010110111101000101010011110110100011
0011110110101111001111011010111000111101101011010110101100101101111011010 0
0001011110110101101001111011010000110111101
1010001000111101000110111011110110100011001111011010000001100101101111010001 0
1010011110100011010001111011010000000111101
1010110110111101101011101011110100011001101111011010000110111101101011110011 1
```

```
10100010101001111010001100011100101101111010
10100011001111011010110101100101101111010001011010111101000101011011110110100
00010111101000101101011111010001011000111101
10101101101111010001010110111101000110101110101101100101101111011010001100111
10110101101011000101101111010001011000111101
10100010001111011010000110111101000101001011110100010110111010110110010110111
10110100011001111011010110101100101101111010
00011011001111011010001010111101101000011011110110100000001111011010110110111
10110101100101111011010000101101011011001010
10111101101000100011110110100000101111010001100110111101101011011011110110101
11110111101101011110011110110100000111100101
10111101101000010110010110111101101000000011110100010101101111011010000010111
10100010111001111011010000010111101101011000
10111101101000011011110100010111101101011100
```

Результат работы программы по 4 символа:

```
11010001100010111011010111011111010100000010110101110111111010100000110001001 1
01101000001100101011101110110101110000110100
00011001010000001011010111011111010101110111100111011000100110110101110111011
01011100011010111001101101010111001101111010
01111011010111011110011101100010011011010111011111011010000011011010000001101 0
01010000001100101010000001011010111011110101
01011101111001110011011010111000001001101101011101110010100000010010101110111
11110101110111111010101110111110011001101101
01110100101011110001001101101000001101101011111011010111001101101010111010010 1
00001110101011101111101110011011010111101111
00101011110110101110111111101000011101110001101000100110110101110010110101110
01110010011011010111001101111010111001110110101
11011110010101110011111010111100101011101001010000111010101110100010011011010
00000101101011100111110101110100101011101111
11010101110100101000011011010111011110110101110001101011101000100110110101110
11100101011100111110101110111110010101110011
11101000001100101011101111011010111001111101011101000101100010011011010000001 0
11010111001110010011011010111011001010000
01101101000000100101011110010100001110101011101000100110110101111001010111101
10100000110010101110011111010111011101101000
00111010101110100010011011010000011011010111011010000011001010111011111011010
11110110100000010110101110011111010111010000
10011011010000011011010111011010000011001010111001110101011101111010101110101
11011010111110110101110111100111001101101011
10011111010111010011100110110100000110010101110111011010111001111010000000110
10001001101101011110010101110011111010111101
11110101011101110110100000010010100000011010001001101101011100111010101110111
11010100000110010101110111110101011101001010
01110111101101000000100111000110100010011011010111001100111010111001111010111
10100101011110110100000011011010111000001001
10110100000011000001010000001001010000001011010000110001001101101011100110110
10101111011010111010110101110111110100000000
01011010111010010101110011100100110110101110101101011100011010111011100101011
11011010111011110010101110000010011011010000
00110010101110111110110101110111011010111001111101011100110110101000011000100 1
10110101110011111010111011100101000001101 0
10111001111101011101011010111010001010000111010101110111111011100110110101111 00
10101110111111010101110111100101000011101010
11101111101110011011010000001100101010111011110010101110011111010111001110101
00001101101000000011010011100110110100000110
01010111010010101110111111010101110111001010111100010110001001101101011100111 1
10101111001010111000110101110111100101000010
10110101110100101011100111001001101101000001100101000011101010111001110101011
11011010111011101101000000110001110101110000
11010111011111101010000011001010000001101001110011011010111010010101111000100 1
10110100000010110100000110110101110011111101
```

```
0000001011010000001001010111011010000011011010000111011101100010011011010111
0100101011110001001101101000001001010111011
1110010101110001101000000110000110100000010011101100010011011010111010010101
1100010011011010000110010101110111101101011
1000110101110011101010111101101110011011110101110000010110001001101101011101
1111001010111001111101000000110001110101111
0110101110111111101011101111101010111000011100110110101110000010011011010111
0
0111010100000110110101110011111010000001100
1010101110011111010111001101111010111000110100000011001011100011010
```

Результат работы программы по 8 символов:

```
1000100101010110111111110011011011111000101111000011110101010010111000011111110
0110110111110000110000001111001010100101110
1100011111100011001001000001100000011110000010000110001000111110000111111001
0110111110000110101100101101111001010111110
0111111011101101111100001101011000111110100011110000110010010110001111001111
1101001111101111010110000011101001011011101
1010000101100111100010010111001100111101100011001001000001110110100100111001
1111101001110011011110110011001101000111111
1011111001111100101010000010010111000110111100101011101101000001110110100001
1
1100001001101000110010011011110110011001100
1111001010111000110110011111010011111010011100110111100100111010010000111101
1001010101101001110011011110000110010010000
1111000010010010110011001100110100011011110000110010010000111101101111101111
0100100100000110101100110100011110101100001
1110101010011010001000011110010011101101011011111010101011001111001000011110
0
1101111101111100011111101111001111000001010
1001001011001111000100100100110100011111010010000110010111011110001011101111
0111111001100010100111101100011110100100011
1011011111110011000111110110011110001110010111001010111010010000011101011011
1
1000011110000100101010011010110001111001010
0111100110100101011100011001101000111000111111101001111101111010110010011111
1111100001111101001111101111010110110000111
1000011101101001101110010101000100011010110000111100111111101111101010111010
0
0010011011110011010010011101011100000111001
0101000111110110011110000111111010011101101110100100101010001011111001011101
1
0111110001111001000110101100011111010001111
0110011000011001101011001101100111101001000011011010011000000111100011111010
0
0111101100111100001110101110001011101011001
1100000011110001111101000111100010101110000100101110011011101001011000110010
1
1001001101111000001110110100010111110010010
1101110110111110101111010110010110111100110111000110011010110000111101101011
0
0011001011101100001001011
```

Метод Шеннона-Фано

Результат работы программы по 2 символа:

```
0010000000100101000001001000000010010010000001000000011010100100100000000100
1
0100000010000000110101010100101010010000000110101011000000010010010000010010
0
0000010010011010010000000110101011000000011010100100100000000100100100000001000
0
0000100100110010000010010000100101010010000000010010010000001001000000010010011
0
0100100000001001001000110000000100100100010010000000010010010000010010000000100
1
1001100100000010010000100101010010000000010010011001001000000011010101001000000
0
0110100111000000011010100010000000110101001001000000001001001000000100000000100
1
1001100100000010101001000000010010011010100101010010000000010010010000011000000
0
0110101001100000001001001000000001000000010010010000001000000010010011001000000
0
0101010010000000100100110001000000010010010000101001010100100000001101010100
1
0000000100100100001001000000010010010001100000001001001100010000000110100100
1
1000000010010010000000000010101001000000010010011001100000001001001000010010
0
```

```
000001001001000000000100000001101001001000001001101001010100100000001001001010
0100100000000100100111010010101001000000010010010001001000000001001001000010010
00000010010011001100000000100100111001000000001001001000011000000001001001100010
00000011010010011000000001001001100001001010100100000000110101001001000000001001
00111001000000010010011000100000000100100100000010000000100100110001000000000110
1001000001000000010010011001001000000001001001101001000000001001001100001001010
10010000000100100100000110000000100100111001000000001001001100110000000001001001
1100100000001101010110000000100100110010010000000010010011100100000000100100110
000100100100001001010100100000001101010010010000000010010011101001010100100000
0010010010000011000000011010101001000000011010100110000000010010010000110000000
01101001001100000000100100110000100101010010000000010010010000110000000010010010
0001001000000011010101100000001001001110010000000010010010000010010000000011010
0100110000000010010011000010010101001000000011010101001000000001001001000010010
0000001101010110000000100100110010010000000010010010000100100000000110101001001
00000001001001110010000000100100110000100101010010000000110101010010000000100
1001000010010000000110101011000000010010011110000000100100100000001000000001001
00100000010010000000100100100001001000000001001001100100000101010010000000010010
011100100000001001001100000000101010010000000110101011000000010010010000001001
00000001001001110010000000011010011010010101001000000001001001000011000000001001
0011100100000001001001000000010000000100100100000100100000000110101001100000001
101001101001010100100000000100100111100000000100100100000001000000001101010110000
00010010010000000100000001001001100010000000010010011001001000000001101010010000
0100110100101010010000000010010010100000100000001001001110010000000010010011000
1000000010010010000100100000000110101010010000000010010011010100101010010000000
110101110000000110101001100000001101010010010000000011010010000100101010010000
0001001001000110000000010010010000100100000000100100110000010000000100100100000
0100000001101010010010000000010010011000100000000100100111010010101001000000010
0100110000010000000010010011010010000000010010010000011000000010010010000100100
000001001001100110000000010010011010100101010010000000110101011000000010010011
00100100000000100100100000100100000000100100111001000000001001001000110000000110
10010000100101010010000000010010011100100000000100100100000110000000011010101001
00000001001001110010000000010010011000001000000001001001100010000000011010010011
000000010010010000000000001010100100000001001001000011000000010010010000001000
00001001001100110000000011010010011000000010010010000000000000101010010000000110
101000100000000100100110011000000001001001110010000000010010011110000000011010010
0000100000001101001100000101010010000000011010101100000001001001100010000000010
010010000000100000001001001000001100000001001001000010100100100001001010100100
0000010010011100100000000100100100001100000001001001101001000000001001001100110
0000001101001000001000000010010011000100000000100100111010010101001000000001101
01011000000001101001001100000000100100111100000001001001000010010000000010010010
000010010000000011010010100100000000100100110100100000000100100100000001000000011
01010110000000011010011000001010100100000001001001100010000000010010010000 10100
10101001000000011010100100100000000110101010010000000010010011100100000000110101
001001000000001101010011000000001001001000010010000000011010101001000000001101001
00100000010010000100101010010000000010010011000100000000100100100001010010101001
00000001101010011000000001001001100110000000010010011010010000000011010110010000
0001101010010000010010000100101010010000000010010011000100000000100100100001010
0101010010000000011010010001000000001001001100100100000000100100110100100000000010
0100111100000001001001000010010000000010010010001001000000001001001101010010010
0001001010100100000000100100110011000000001001001110010000000110100101001000000
01001001000010010000000100100100000000010000000100100100000001000000001001001101
0000010101001000000001001001101010010101001000000001001001111000000011010101001
0000000100100111001000000001101010001000000010010011100100000000100100100010010
000000100100110100100000000110101000000000010011
```

Результат работы программы по 4 символа:

```
010001111101110010100000111100100000010011100001000001111001000000010011011100
0101010001001101100000000111001010000011001
01000100110110000001001110000100000111100100000000111101110010111000101010000
0111001010000011001010000011111001000000001
11110001000001000010000011110111001011100010101000001111010100010011010100010
0111110110000000100111111001000000001001110000
```

```
1000001111001000000001111011100110101000001100111000101010000011100110000001001111000100000000111101110011010100000110001000000010001100010101000010011010100000100001000001111000100000010011000000
0011110011001101010000011110110000000010000100000111100010001001001110011111000001111000100000100001111000100000100001000010000111000010000011101110001010000011100001000001110111000101010
1000001111100010000010000100000111101100000000111010100000100010000000110001000000000111010100000100010000000100101010000011100001000001110111000101010000011100001000001110111000101010
0000011001100000000111010100000111101100000000111010100010011011000000000111100000000111010100010011011000000000111100000000111010100010011011000000000111101010000011100011000011100010101
1010000011101010000011000110000111000101010000011101010000111000101010000011101010000111000101010000011101010000111000101010000011101010000111000101010
0010011100001000001110111000101010000011100110000001001101010001001110001000000100010010000100000100010000010010011000000000110001100010101000100010
1010000010001000000001000010001001101100000000111010100000111001010001001001100000100010000011101110011010000000111000110011010100010010001101000010
0000000011000110001010100010011010100000100001000100111000010000011101010000011000101000010000111010100000110001100011010100010011010100000100001000
0100010011011000000001111010100000100001000100111000010000011101010000011000101000010000111010100000110001100011010100010011010100000100001000
1100010101000100110101000001000010001001101
1000000001110110000000011110010000000011100101000001000010000011110111001101000000111011100110100
1000001110101000001100011001101010001001101
1000000001110010100000111010100010011111011100010101000001000100000001110101000000111011100110100
0000011110010000000111001010001001111000100001110101000001100010000000100001000010000111010111001101
0100000001100010000000011110101000100111000100
0110011111011100010101000001111111000100000111010100000110001000000001000010000
1001101010000011001110001010100010011111111
0000001001110001000000100111000010001001011100010101000001111100100000000100001000
0100000110000100000111100100000010011100001
0000011000100000000111011100010101000001100001000001100101000001110011000000000
0100001000001111011000000001100111000101010
0010011011000000000111010100000111001010000011101010000011111001000000100101100
1000101010000011101010000011100110000001001
1010100000111010100000110000100000110001000000100100110000000001111001100110100
1000001000100000000111100100000000111101100
0000100100110000000001111001100110101000100111111100100000000111101100000000111
0101000001110110000001001010100010011111011
1001101010001001101100000000011000100000000111100100000000011100110000000010001
1000011100010101000001110101000001000100000
0001100101000000111101100000010010101000001100010000000011101110001010100010010
1011000000100100110000000011101100000000100
0010000011100101000100111111101010000011001010000011110010000001001101100000001
0011111011100110101000001100010000000010001
1000101010001001110000100010011010100000111010100010011100001000100111000100000
0000010000100010011010100010010011100101110
0010101000001100010000000010001100010101000100111000100000000111101100000000100
1001010001001111111010001001110001100101110
0010101000001100010000000010001100010101000100101100000000111101010000011001000
1000001110110000000010000100000111110001000
0011001110000111000101010000011110110000000011101010001001111110101000001000000
1000001111000100000111100100000000110011100
1101010000011001110001010100000111011000000100110101000001110101000100111111000
0100000000111010100000111111000100000110010100
0001001111111001100111110
```

Результат работы программы по 8 символов:

```
0001111111000100100110110001001001100010100001000100111000100011000000010001001101100011101100010100001000100111000100011000000010001001111001100110101110101000011000100011000000100010000111000100111100100110101100100010000100110000001000010001000100001000000001101100111011000100000101011100101000001010000100010011100010001100000001000100001111001001111001100110101100100011000001000011
0001001101100110010001001001101011001000100001001100000010000100010001000010000001101100111011000100000101011100101000001010000100010011100010001100000001000100001111001001111001100110101100100011000001000011
0110110011000010011110100011001110010001000100
0110000000010010001101011110000100001111110000011010000110110001000000001010000
```

```
0101100010000000110000100111010001100001010
000101100001100010000100110010011000010011110000000101000011000000010000101000
001100010010110001010001011000100010011010
000110001000011111001010000101100001101000011000100110001000101100110000000011
101000110000101000010000000110000000100010
010110001100010001101000110000101000010001000100000000100010011010000100000000
010110000110001000010100001000100010000000
100010010110011001001110001000100000000101011001000011000100001110010001000100010
011100010000110001110001010000100000010110
001001001110001000101100111100010100000110011001001100010011001001100001001100
100001011111100001000011111010000110001000
100110000000010001000110000010000111110000110110011011000111000001000011011000
110000000010000100010010011011000100010010
110100001000010001100000001100010011000000001010001110000010000100010011010000
011000100001100011011001111000001000011000
110001001100010000110001000010001000010011101000110011100100010000000110010010
101000111010001100111001000111001011010001
011000011001111000000011100011100100010001001100001001100100111000100110000001
111001010000011000100000001110000000101000 1
111000000010000100101101000010001001110100010110011000000001100010001111010000
110000000100100110001001111000111001000 1000
010011000000100001011000100010000110001011000101100110000000010001000110101111
010100001000010011000000100001011001101000 1
110000000111110000010111111010100001000010011000000100001110000000110100001100 0
000001100110000000011011000110000111100 1010
000101000101100011110100010000000010010001001001111000011001000110000010000011
001100010000110001110010010110001101100011 0
0000001110011111000
```

Задание №5

Код программы:

task5.h

```
#ifndef INFORMATION_THEORY_TASK5_H
#define INFORMATION_THEORY_TASK5_H

#include "task3_1.h"
#include "task3_2.h"

#include <ctime>
#include <time.h>

double getCompressionRatio(std::string s, const std::vector<character> &ta-
ble);

#endif //INFORMATION_THEORY_TASK5_H
```

task5.cpp

```
#include "task5.h"

double getCompressionRatio(std::string s, const std::vector<character> &ta-
ble) {
    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;
}
```

main5.cpp

```
#include "../../libs/alg/labs/lab3/task5.h"

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(CP_UTF8);
    setlocale(LC_ALL, "Russian");

    std::string s;
    std::ifstream f0("C:\\Users\\Artur\\Projects\\C++\\information_the-
ory\\labs\\lab3\\task_2.txt");
    f0 >> s;
    f0.close();

    int length = 8;
    std::vector<int> r = getSequencesOfNCharactersEach(s, length);

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

    std::cout << "Время: " << (double) work_time;
    std::cout << "\nКоэффициент сжатия: " << getCompressionRatio(s, res);

    return 0;
}
```

Метод Хаффмана

Результат работы программы по 2 символам:

```
Время: 0
Коэффициент сжатия: 1
```

Результат работы программы по 4 символам:

```
Время: 0
Коэффициент сжатия: 1.24584
```

Результат работы программы по 8 символам:

```
Время: 0
Коэффициент сжатия: 2.165
```

Метод Шеннона-Фано

Результат работы программы по 2 символам:

```
Время: 0
Коэффициент сжатия: 0.815842
```

Результат выполнения программы по 4 символам:

```
Время: 0
Коэффициент сжатия: 1.17546
```

Результат работы программы по 8 символам:

```
Время: 0
Коэффициент сжатия: 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 вручную также примерно одинаковы. Таким образом, можно сделать вывод, что алгоритм Хаффмана более эффективен, чем метод Шеннона-Фано. А значит, лучше использовать метод Хаффмана, т.к. по скорости и простоте выполнения алгоритмы очень схожи.

Задание №6

Код программы:

task6.h

```
#ifndef INFORMATION_THEORY_TASK6_H
#define INFORMATION_THEORY_TASK6_H

#include "task3_1.h"
#include "task3_2.h"
#include "task4.h"

bool areVectorsEqual(std::vector<int> a, std::vector<int> b);

int getPosOfTheVector(const std::vector<character> &table, const std::vec-
tor<int> &a);

std::string decoding(std::string codingS, std::vector<character> table, int
length);

#endif //INFORMATION_THEORY_TASK6_H
```

task6.cpp

```
#include "task6.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::vec-
tor<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;
}
```

main6.cpp

```
#include "../../libs/alg/labs/lab3/task6.h"

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(CP_UTF8);
    setlocale(LC_ALL, "Russian");

    std::string s;
    std::ifstream f0("C:\\Users\\Artur\\Projects\\C++\\information_the-
ory\\labs\\lab3\\task_2.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 << "ДА!!!\n";
    else
        std::cout << "НЕТ!!\n";

    return 0;
}
```

Результат работы программы по 2 символов:

```
ДА!!!
```

Результат работы программы по 4 символов:

```
ДА!!!
```

Результат работы программы по 8 символов:

```
ДА!!!
```

Задание №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.