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

1101000010010010110100001011010111010001100000101101000010110101110100011000000000100000110100011000000111010000101100101101000010111000110100011000000111010001100000101101000010110101110100001011101100101100001000001101000010110010110100001011100011010000101101111101000010110110110100001011000011010000101110110010110000100000110100001011101011010001100000001101000110001111110100011000010111010001100000101101000010110101110100001011101100100000110100001011100000100000110100001011001111010001100000111101000010110100110100001011010111010000101110110010000011010000101111011101000010110000001000001101000110000000110100001011000011010000101101111101000010111101110100011000101111010000101101010010000011010000101110111101000010110000110100001011010011010001100010110010111000100000110100001010001011010000101111100010000011010000101101101101000010110000110100001011101111010000101111101101000010110001110100001011110111010001100010111101000010111100001000001101000110000010110100001011111011010000101111011101000010110101110100001011110111010001100011001101000010111010110100001011100011010000101111000010000011010000101100111101000010111110110100001011101111010000101111101101000110000001110100001011101011010000101111101101000010111100001011000010000011010001100000101101000010111110001000001101000010110011110100011000000011010001100000111101000010110001110100011000101111010000101111000010000011010000101100011101000010110000110100011000000111010000101111101101000010110010110100011000101111010000101111000010000011010001100000001101000010110000110100011000000111010000101110101101000010110000110100011000001011010000101111101101000010111100001000001101000110000000110100001011000011010001100000011101000010111111110100001011010111010000101100101101000010110000110100001011101100100000110100001011111011010000101111010010000011010001100000011101000010110010110100001011111011010001100011100010000011010000101100011101000010111110110100001011010111010000101100101101000110000011110100011000111000100000110100001011111111010000101101011101000110000001110100001011010111010000101111011101000010111010110100011000001100101110001000001101000010100100110100001011111011010000101111011101000010110000110100011000000011010000101110000010000011010001100001111101000110000011110100011000001011010001100011000010000011010000101101111101000010110000110100001011110011010000101101011101000110000010110100001011110111010000101111100010000011010000101111001101000010111000110100001011001111010000101100001101000010111011110100001011100000100000110100011000000111010000101110101101000010110010110100001011111011010000101101111101000110001100001000001101000010111110110100001011001111010001100000001101000010111110110100001011110011010000101111011101000110001011110100001011010100100000110100001011000111010000101101011101000010111011110100011000101111010000101101010010000011010001100001011101000010111011110100001011111011010000101111111101000110001100110100011000111100100000110100011000000111010000101111011101000010110101110100001011001111010000101100000010110000100000110100001011111011010000101100011101000010111000110100001011101111010001100011001101000010111101110100001011111000100000110100011000000111010001100010111101000010111111110100001011000011010000101100101101000110001000110100001011100011010000101101011101000110000001110100011000111100100000110100001011110111010000101100000010000011010001100000101101000110000000110100001011111011010001100000101101000110000011110100001011000011010001100000001101000110001011001011000010000011010000101111011101000010110000001000001101000110000011110100001011101111010000101110001101000110000110110100011000001100101100001000001101000010111101110100001011000000100000110100011000110111010000101110101101000010111000110100001011111111010000101100001101000010110110110100001011100000101100001000001101000010111011110100001011111011010001100010001101000010110000110100001011010011010000101101011101000010111001001000001101000010111000001000001101000010111111110100011000000011010000101111101101000110000101110100001011111011010000101101101101000010111000110100011000010100101110

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\_theory\\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 а, если элемента нет в векторе, то возвращае т -1.  
// позиция элемента n - позиция элемента в векторе а, 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 << "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\_theory\\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\_theory\\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> &table, 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\_theory\\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:

101111011001111010111101101011110011110100010110101111011010111100111101000101010110010110111101000101010011110110101110  
101111011010000110111101000101010011110100010110101111011010111100111101101000101110101101100101101111011010111010111101  
101000011011110110101100001111011010110010111101101011011011110110100010111010110110010110111101101000101011110100010101  
101111010001100000111101000101110011110100010110101111011010111100111101101000101110010110111101101000010110010110111101  
101011100011110100010110001111011010111110111101101011110011110110100010111001011011110110100011001111011010110101100101  
101111010001010110111101101011011011110110101100001111011010001100111101000110100011110110101111111001011011110110100010  
001111011010110110111101101011111011110100011010111010000110010110111101101001101011110110100000011001011011110110101100  
101111011010110110111101101000100011110110100000101111011010110100111101101000110011110100011010001111011010001101100101  
101111010001011010111101101000001011110110100011001111011010111100111101101000110011110100011011101111011010001010111101  
101000011011110110100011011001011011110110101110001111011010000010111101101000100011110110100000101111010001010100111101  
101000101011110110100000101111011010001101101011011001011011110100010110101111011010000001100101101111011010111000111101  
000101011011110100010110001111011010110100111101000110100011110110100011011001011011110110101101001111011010110110111101  
000101010011110110100000101111011010111010111101000110100011110110100011011001011011110100010101101111011010110110111101  
000101010011110110100010101111011010110110111101000101101011110110100000101111011010001101100101101111010001010110111101  
101011011011110100010101001111011010000000111101101011110011110110101110101111011010110110111101101000101110010110111101  
101000001011110110100011111001011011110100010101001111011010111010111101101000001011110100011000011001011011110110101101  
001111011010000010111101101011110011110110101110101111010001011000111101000110000110010110111101101000000011110110101111  
001111010001010100111101101011110011110110100011001111011010001010111101000101101110100001100101101111011010011110111101  
101000001011110110100011001111011010110110111101000101011011110110100001011001011011110100010100001111010001011000111101  
000101101011110100011011011001011011110110101100001111011010110110111101101000111011110110101111001111010001011010111101  
101000110011110110100000011001011011110110100011101111011010000110111101101011100011110110101101101111011010001000111101  
101000010110010110111101000101010011110110100010101111011010111010111101101000001011110110101100001111010001101101100101  
101111011010000010111101101011100011110100010101101111011010000010111101101000111011110110100011001111010001101000111101  
101011111110010110111101101011010011110110101111001111011010001000111101000110100011110110101111111001011011110100010111  
001111011010001000111101101000001011110110100000001111010001101110111101000110001110010110111101000101010011110110100011  
001111011010111100111101101011100011110110101101011010110110010110111101101000001011110110101101001111011010000110111101  
101000100011110100011011101111011010001100111101101000000110010110111101000101010011110100011010001111011010000000111101  
101011011011110110101110101111010001100110111101101000011011110110101111001111010001010100111101000110001110010110111101  
101000110011110110101101011001011011110100010110101111010001010110111101101000001011110100010110101111010001011000111101  
101011011011110100010101101111010001101011101011011001011011110110100011001111011010110101100101101111010001011000111101  
101000100011110110100001101111010001010010111101000101101110101101100101101111011010001100111101101011010110010110111101  
000110110011110110100010101111011010000110111101101000000011110110101101101111011010110010111101101000010110101101100101  
101111011010001000111101101000001011110100011001101111011010110110111101101011111011110110101111001111011010000111100101  
101111011010000101100101101111011010000000111101000101011011110110100000101111010001011100111101101000001011110110101100  
1011110110100001101111010001011111101000

Результат работы программы по 4 символа:

110100011000101110110101110111110101000000101101011101111101010000011000100110110100000110010101110111011010111000110100  
000110010100000010110101110111110101011101111001110110001001101101011101110110101110001101011100110110101011100110111101  
011110110101110111100111011000100110110101110111101101000001101101000000110100101000000110010101000000101101011101111101  
010111011110011100110110101110000010011011010111011100101000000100101011101111111010111011111010101110111100111001101101  
011101001010111100010011011010000011011010111101101011100110110101011101001010000111010101110111110111001101101011101111  
001010111101101011101111111010000111011100011010001001101101011100101101011100111001001101101011100110111101011110110101  
110111100101011100111110101111001010111010010100001110101011101000100110110100000010110101110011111010111010010101110111  
110101011101001010000110110101110111101101011100011010111010001001101101011101110010101110011111010111011110010101110011  
111010000011001010111011110110101110011111010111010001011000100110110100000010110101110011100100110110101110111001010000  
011011010000001001010111100101000011101010111010001001101101011110010101111011010000011001010111001111101011101110110100  
001110101011101000100110110100000110110101111011010000011001010111011110110101111011010000001011010111001111101011101000  
100110110100000110110101111011010000011001010111001110101011101111101010111011101101011110110101110111100111001101101011  
100111110101110100111001101101000001100101011101110110101110011111010000001101000100110110101111001010111001111101011101  
111101010111011101101000000100101000000110100010011011010111001110101011101111101010000011001010111011111010101110100101  
011101111011010000001001110001101000100110110101110011001110101110011111010111010010101111011010000011011010111000001001  
101101000000110000010100000010010100000010110100001100010011011010111001101101010111101101011101011010111011111010100000  
010110101110100101011100111001001101101011101011010111000110101110111001010111101101011101111001010111000001001101101000  
001100101011101111011010111011101101011100111110101110011011010100001100010011011010111001111101011101110010100000110110  
101110011111010111010110101110100101000011101010111011111011100110110101111001010111011111010101110111100101000011101010  
111011111011100110110100000011001010101110111100101011100111110101110011101010000110110100000011010011100110110100000110  
010101110100101011101111101010111011100101011110001011000100110110101110011111010111100101011100011010111011110010100001  
101101011101001010111001110010011011010000011001010000111010101110011101010111101101011101110110100000011000111010111000  
110101110111110101000001100101000000110100111001101101011101001010111100010011011010000001011010000011011010111001111101  
000000101101000000100101011110110100000110110100001110111011000100110110101110100101011110001001101101000000100101011101  
111001010111000110100000011000011010000001001110110001001101101011101001010111100010011011010000110010101110111101101011  
100011010111001110101011110110101110011011110101110000010110001001101101011101111001010111001111101000000110001110101111  
011010111011111110101110111110101011100001110011011010111000001001101101011100111010100000110110101110011111010000001100  
1010101110011111010111001101111010111000110100000011001011100011010

Результат работы программы по 8 символов:

100010010101011011111110011011011111000101111000011110101010010111000011111100110110111110000110000001111001010100101110  
110001111110001100100100000110000001111000001000011000100011111000011111100110110111110000110101100101101111001010111110  
011111101110110111110000110101100011111010001111000011001001011000111100111111101001111101111010110000011101001011011101  
101000010110011110001001011100110011110110001100100100000111011010010011100111111101001110011011110110011001101000111111  
101111100111110010101000001001011100011011110010101110110100000111011010000111100001001101000110010011011110110011001100  
111100101011100011011001111010011111010011100110111100100111010010000111101101001010101101001110011011110000110010010000  
111100001001001011001100110100011011110000110010010000111101101111101111101010100100100000110101100110100011110101100001  
111010101001101000100001111001001110110101101111101010101100111100100001111001101111101111100011111101111100111110000101  
100100101100111100010010010011010001111101001000011001011011110001011101111100111111001100010100111101100011110100100011  
101101111111001100011111011001111000111001011100101011101001000001110101101111000011110000100101010011010110001111001010  
011110011010010101110001100110100011100011111110100111110111101011001001111101111100001111101001111101111010110110000111  
100001110110100110111001010100010001101011000011110011111110111110101011101000010011011110011010010011101011100000111001  
010100011111011001111000011111101001110110111010010010101000101111100101110110111110001111001000110101100011111010001111  
011001100001100110101100110110011110100100001101101001100000011110001111101000111101100111100001110101110001011101011001  
110000001111000111110100011110001010111000010010111001101110100101100011001011001001101111000001110110100010111110010010  
1101110110111110101111010110010110111100110111000110011010110000111101101011000110010111011000010010110

Метод Шеннона-Фано

Результат работы программы по 2 символа:

0010000000100101000001001000000010010010000001000000011010100100100000001001001000000100000001101010101001010100100000001101010110000000100100100000100100000001001001101001000000011010101100000001101010010010000000100100100000010000000100100110010000010010000100101010010000000100100100000100100000001001001101001000000010010010001100000001001001000100100000001001001000010010000000100100110010000010010000100101010010000000100100110010010000000110101010010000000110100111000000011010100010000000110101001001000000010010010000001000000010010011001000001010100100000001001001101010010101001000000010010010000011000000011010100110000000100100100000000100000001001001000000100000001001001100100000101010010000000100100110001000000010010010000101001010100100000001101010100100000001001001000010010000000100100100011000000010010011000100000001101001001100000001001001000000000001010100100000001001001100110000000100100100001001000000010010010000000010000000110100100100000100110100101010010000000100100101001001000000010010011101001010100100000001001001000100100000001001001000010010000000100100110011000000010010011100100000001001001000011000000010010011000100000001101001001100000001001001100001001010100100000001101010010010000000100100111001000000010010011000100000001001001000000100000001001001100010000000110100100000100000001001001100100100000001001001101001000000010010011000010010101001000000010010010000011000000010010011100100000001001001100110000000100100111001000000011010101100000001001001100100100000001001001110010000000100100110000100100100001001010100100000001101010010010000000100100111010010101001000000010010010000011000000011010101001000000011010100110000000100100100001100000001101001001100000001001001100001001010100100000001001001000011000000010010010000100100000001101010110000000100100111001000000010010010000010010000000110100100110000000100100110000100101010010000000110101010010000000100100100001001000000011010101100000001001001100100100000001001001000010010000000110101001001000000010010011100100000001001001100001001010100100000001101010100100000001001001000010010000000110101011000000010010011110000000100100100000010000000100100100000100100000001001001000010010000000100100110010000010101001000000010010011100100000001001001100000000101010010000000110101011000000010010010000010010000000100100111001000000011010011010010101001000000010010010000110000000100100111001000000010010010000001000000010010010000010010000000110101001100000001101001101001010100100000001001001111000000010010010000001000000011010101100000001001001000000100000001001001100010000000100100110010010000000110101001000001001101001010100100000001001001010000010000000100100111001000000010010011000100000001001001000010010000000110101010010000000100100110101001010100100000001101011100000001101010011000000011010100100100000001101001000010010101001000000010010010001100000001001001000010010000000100100110000010000000100100100000010000000110101001001000000010010011000100000001001001110100101010010000000100100110000010000000100100110100100000001001001000001100000001001001000010010000000100100110011000000010010011010100101010010000000110101011000000010010011001001000000010010010000010010000000100100111001000000010010010001100000001101001000010010101001000000010010011100100000001001001000001100000001101010100100000001001001110010000000100100110000010000000100100110001000000011010010011000000010010010000000000010101001000000010010010000110000000100100100000010000000100100110011000000011010010011000000010010010000000000010101001000000011010100010000000100100110011000000010010011100100000001001001111000000011010010000010000000110100110000010101001000000011010101100000001001001100010000000100100100000010000000100100100000110000000100100100001010010010000100101010010000000100100111001000000010010010000110000000100100110100100000001001001100110000000110100100000100000001001001100010000000100100111010010101001000000011010101100000001101001001100000001001001111000000010010010000100100000001001001000001001000000011010010100100000001001001101001000000010010010000001000000011010101100000001101001100000101010010000000100100110001000000010010010000101001010100100000001101010010010000000110101010010000000100100111001000000011010100100100000001101010011000000010010010000100100000001101010100100000001101001001000001001000010010101001000000010010011000100000001001001000010100101010010000000110101001100000001001001100110000000100100110100100000001101011001000000011010100100000100100001001010100100000001001001100010000000100100100001010010101001000000011010010001000000010010011001001000000010010011010010000000100100111100000001001001000010010000000100100100010010000000100100110101001001000010010101001000000010010011001100000001001001110010000000110100101001000000010010010000100100000001001001000000001000000010010010000001000000010010011010000010101001000000010010011010100101010010000000100100111100000001101010100100000001001001110010000000110101000100000001001001110010000000100100100010010000000100100110100100000001101010000000010011

Результат работы программы по 4 символа:

010001111101110010100000111100100000010011100001000001111001000000100110111000101010001001101100000000111001010000011001  
010001001101100000010011100001000001111001000000001111011100101110001010100000111001010000011001010000011111001000000001  
111100010000010000100000111101110010111000101010000011110101000100110101000100111110110000001001111110010000001001110000  
100000111100100000000111101110011010100000110011100010101000001110011000000100111000100000000111100010000011110010000000  
011110111001101010000011000100000000100011000101010001001101010000010000100000111110010000000011000100000010010011000000  
001111001100110101000001111011000000001000010000011110001000100100111001111101110001010100000111000010000011101110001010  
100000111110001000001000010000011110110000000011101010000010001000000001100010000001001001100000000110001100010101000100  
111000010000011101010000011000100000000111100100000000110001000000100101010000011110101000001100101000001100011000101010  
000011100110000000011101010000011110110000000011101010001001101100000000111101010000011101010000011000110000111000101010  
001001110000100000111011100010101000001110011000000100110101000100111000100000000100010000001001001100000000110001100010  
101000001000100000000100001000100110110000000011101010000011100101000100100110000000011000110001010100010011010100000100  
001000100110110000000011110101000001000010001001110000100000111010100000110001100010101000100110101000001000010001001101  
100000000111011000000001111001000000001110010100000100001000001111011100110101000001110101000001100011001101010001001101  
100000000111001010000011101010001001111101110001010100000100010000000011101010000011110010000000011100101000100111000100  
000010011111011100010101000001110110000000011110010000001001101100000000111100100000000110001000000001111010100010011100  
011001111101110001010100000111111000100000111010100000110001000000001000010001001101010000011001110001010100010011111111  
000000100111000100000010011100001000100101110001010100000111110010000000010000100000110000100000111100100000010011100001  
000001100010000000011101110001010100000110000100000110010100000111001100000000100001000001111011000000001100111000101010  
001001101100000000111101010000011100101000001110101000001111100100000010010111000101010000011101010000011100110000001001  
101010000011101010000011000010000011000100000010010011000000001111001100110101000001000100000000111100100000000111101100  
000010010011000000001111001100110101000100111111001000000001111011000000001110101000001110110000001001010100010011111011  
100110101000100110110000000011000100000000111100100000000111001100000000100011000011100010101000001110101000001000100000  
000110010100000111101100000010010101000001100010000000011101110001010100010011011000000100100110000000011101100000000100  
001000001110010100010011111101010000011001010000011110010000001001101100000010011111011100110101000001100010000000010001  
100010101000100111000010001001101010000011101010001001110000100010011100010000000010000100010011010100010010011100101110  
001010100000110001000000001000110001010100010011100010000000011110110000000011001010001001111111010001001110001100101110  
001010100000110001000000001000110001010100010010110000000011110101000001100101000001110110000000010000100000111110001000  
001100111000011100010101000001111011000000001110101000100111111010100000100001000001111000100000111100100000000110011100  
110101000001100111000101010000011101100000010011010100000111010100010011111100100000000111010100000111110001000001100101  
0001001111110011001111101

Результат работы программы по 8 символов:

000111111100010010011011000100100110001010000100010011100010001100000001000100110110001001001101011101010000110001000110  
000000110110011101100010000000010101110101000010100001000100011100010011110010011011000100100110101100100011000001000011  
000100110110011001000100100110101100100010000100110000001000010001000100000000110110011000010011101000110011100100010100  
011000000001001000110101111000010000111111000001101000011011000100000000101000101100010000000110000100111010001100001010  
000101100001100010000100110010011000010011110000000101000011000000010000101000011000100101100010100010110001000100110100  
000110001000011111001010000101100001101000011000100110001000101100110000000111010001100001010000100000001100000000100010  
010110001100010001101000110000101000010001000100000000100010011010000100000000101100001100010000101000010001000100000000  
100010010110011001001110001000100000000101011001000011000100001110010001000100111000100001100011100010100001000000010110  
001001001110001000101100111100010100000110011001001100010011001001100001001101000010111111000010000111110100001100010000  
100110000000010001000110000010000111110000110110011011000111000001000011011001100000000100001000100100110110001000010010  
110100001000010001100000001100010011000000001010001110000010000100010011010000110001000011000110110011110000010000011000  
110001001100010000110001000010001000010011101000110011100100010000000110010011010001110100011001110010001110010011010001  
011000011001111000000011100011100100010001001100001001100100111000100110000011110010100000110001000000011100000001010001  
111000000010000100101101000010001001110100010110011000000001100010001111010001100000001001001100010011110001110010001000  
010011000000100001011000100010000110001011000101100110000000010001000110101110101000010000100110000001000010110011010001  
110000000111110000010111110101000010000100110000001000011100000011010000110000000011001100000000110110001100001111001010  
000101000101100011110100010000000010010001001001110000110010001100000100000110011000100001100011100100101100011011000110  
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> &table);  
  
#endif //INFORMATION\_THEORY\_TASK5\_H

task5.cpp

#include "task5.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;  
}

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\_theory\\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 С++ 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::vector<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::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;  
}

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\_theory\\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.