**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«Сибирский государственный университет науки и технологий   
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедра информационно-управляющих систем

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Теория информации

|  |
| --- |
| Определение избыточности сообщений. Оптимальное неравномерное кодирование. |

Руководитель А.Н. Бочаров

подпись, дата инициалы, фамилия

Обучающийся БПИ22-02, 221217039 Н.Ю. Сюткин

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2023 г.

# ЦЕЛЬ РАБОТЫ

Закрепление знаний по методам кодирования информации.

# порядок выполнения работы

1. Ознакомиться с общей постановкой задачи.
2. Ознакомится с вариантом задания – соответствует вашему номеру в списке группы.
3. Выполнить задания согласно варианту.
4. Подготовить отчет по лабораторной работе. Отчет должен включать в себя:   
   •  титульный лист;   
   •  цель лабораторной работы;   
   •  постановку задачи;   
   •  ход работы;   
   •  краткие ответы на контрольные вопросы;   
   •  выводы по лабораторной работе.
5. Защитить лабораторную работу перед преподавателем.

# постановка задачи

1. Используя текстовый файл из лабораторных работ 1–2, определить:   
 – избыточность заданного текста, вызванную неравновероятностью появления символов в сообщении Dp;   
 – избыточность, вызванную статистической связью между соседними символами Ds;   
 – полную избыточность D.   
 2. Используя этот же текстовый файл, построить оптимальный неравномерный код, применяя:   
 – метод Шеннона–Фано;   
 – метод Хаффмана.   
 3. Определить для каждого метода среднюю длину символа исходного алфавита lср, коэффициент статистического сжатия Kcc, коэффициент относительной эффективности Kоэ.   
 4. Написать программу кодирования и декодирования исходного текста методом Шеннона–Фано и методом Хаффмана. В отчете привести отрывок текста объемом не менее 4-х абзацев, закодированного и декодированного обоими методами.

# ХОД РАБОТЫ

Для выполнения данной лабораторной работы был закодирован алфавит в Excel методом Шеннона-Фано и Хаффмана с помощью дерева.

Исходные данные, которые потребовались для кодирования, представлены в Таблице 1 ниже.

Таблица 1 – Вероятности символов

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Буква | Вероятность | Буква | Вероятность | Буква | Вероятность | Буква | Вероятность |
| пробел | 0,149 | р | 0,047 | я | 0,022 | х | 0,007 |
| о | 0,085 | в | 0,043 | ы | 0,017 | ж | 0,004 |
| е | 0,074 | л | 0,032 | з | 0,014 | ю | 0,009 |
| а | 0,071 | к | 0,031 | ъ | 0,015 | ш | 0,008 |
| и | 0,057 | м | 0,03 | б | 0,016 | ц | 0,001 |
| н | 0,052 | д | 0,022 | г | 0,015 | щ | 0,004 |
| т | 0,05 | п | 0,025 | ч | 0,014 | э | 0,002 |
| с | 0,042 | у | 0,026 | й | 0,01 | ф | 0,002 |

Кодирование выполнялось с помощью таблицы Excel. Результаты кодирования методом Шеннона-Фано можно видеть в Таблице 2.

Таблица 2 – Результат кодирования методом Шеннона-Фано

|  |  |
| --- | --- |
| Буквы | Код |
| 1 | 2 |
| Пробел | 111 |
| о | 110 |
| а | 1011 |
| е | 1010 |
| н | 1001 |
| и | 1000 |
| т | 0111 |
| с | 01101 |
| р | 01100 |
| л | 0101 |
| в | 01001 |
| к | 01000 |
| м | 00111 |
| п | 00110 |
| у | 00101 |
| д | 001001 |
| я | 001000 |
| ь | 000111 |
| ы | 000110 |

Окончание Таблицы 2

|  |  |
| --- | --- |
| 1 | 2 |
| б | 000101 |
| з | 000100 |
| г | 000011 |
| ч | 0000101 |
| й | 0000100 |
| ш | 0000011 |
| х | 0000010 |
| ю | 00000011 |
| ж | 00000010 |
| щ | 000000011 |
| э | 000000010 |
| ц | 000000001 |
| ф | 000000000 |

Результаты кодирования методом Хаффмана можно видеть в Таблице 3.

Таблица 3 – Результат кодирования методом Хаффмана

|  |  |
| --- | --- |
| Буквы | Код |
| 1 | 2 |
| Пробел | 111 |
| о | 1101 |
| а | 1011 |
| е | 1001 |
| н | 0111 |
| и | 0101 |
| т | 0011 |
| с | 0001 |
| р | 11001 |
| л | 11000 |
| в | 10101 |
| к | 10001 |
| м | 01101 |
| п | 01001 |
| у | 01000 |
| д | 00101 |
| я | 00001 |
| ь | 00000 |
| ы | 101001 |
| б | 100001 |
| з | 100000 |
| г | 011001 |

Окончание Таблицы 3

|  |  |
| --- | --- |
| 1 | 2 |
| ч | 001001 |
| й | 001000 |
| ш | 1010001 |
| х | 0110001 |
| ю | 10100001 |
| ж | 01100001 |
| щ | 101000001 |
| э | 101000000 |
| ц | 011000001 |
| ф | 011000000 |

Теперь посчитаем все нужные по заданию значения.

Частная избыточность взаимосвязанных элементов:

Ds=1- = 1 - = 0,21055

Частная избыточность, вызванная неравновероятными появлениями символов:

Dp=1- = 1- = 1 - = 0,125281

Полная избыточность:

D = Dp + Ds - Dp\*Ds = 0,125281 + 0,21055 -0,125281 \* 0,21055 = 0,309453

Средняя длина кода по Шеннону-Фано:

Lср = 4,4309

Средняя длина кода по Хаффману:

Lср = 4,4163

Коэффициент статического сжатия по Хаффману:

Ксс = = = 1,13217

Коэффициент относительной эффективности по Хаффману:

Коэ = = = 0,99033

Коэффициент статического сжатия по Шеннону-Фано:

Ксс = = = 1,12844

Коэффициент относительной эффективности по Шеннону-Фано:

Коэ = = = 0,98707

Далее была разработана программа на языке C#, которая кодирует/декодирует заданный отрывок текста. Исходный отрывок кодируется/декодируется слева направо, сравнивая каждый элемент с соответствующим кодом/символом.

Метод CheckCode предназначен для проверки кодов букв на свопадение с указанием на конкретные совпавшие символы.

public class CheckCodes

{

public static void CheckCode(Dictionary<string, string> ToCheck)

{

List<string> symbols = new List<string>(ToCheck.Keys);

List<string> codes = new List<string>(ToCheck.Values);

Console.WriteLine("Кол-во букв: " + ToCheck.Count);

for(int i = 0; i < symbols.Count; i++)

{

for(int j = 0; j < symbols.Count; j++)

{

if (codes[i] == codes[j])

{

if(i == j)

{

continue;

}

else

Console.WriteLine("Совпадение!" + " Буква: " + symbols[i] + " Коды:" + codes[i] + " , " + codes[j]);

}

}

}

}

}

Далее был реализован метод DeletePunc для удаления всевозможной пунктуации, чтобы исключить ошибки в процессе кодирования.

public class DeletePunctuation

{

public static string DeletePunc(string text)

{

string punctuation = "!\"#$%&'()\*+,-./:;<=>?@[\\]^\_`{|}~";

foreach (char c in punctuation)

{

text = text.Replace(c.ToString(), "");

}

return text;

}

}

Также, программно был спроектирован класс Coding, включащий в себя методы Encoding (для кодирования заданного текста), Decoding (для декодирования заданного текста).

public class Coding

{

public static string Encoding(string text, Dictionary<string, string> dict)

{

string result = string.Empty;

List<string> symbols = new List<string>(dict.Keys);

List<string> codes = new List<string>(dict.Values);

for(int i = 0; i < text.Length; i++)

{

for(int j = 0; j < codes.Count; j++)

{

if (text[i].ToString() == symbols[j])

{

result += codes[j];

}

}

}

return result;

}

public static string Decoding(string text, Dictionary<string, string> dict)

{

string result = string.Empty;

List<string> symbols = new List<string>(dict.Keys);

List<string> codes = new List<string>(dict.Values);

string temp = string.Empty;

for (int i = 0; i < text.Length; i++)

{

temp += text[i];

for (int j = 0; j < codes.Count; j++)

{

if (temp == codes[j])

{

result += symbols[j];

temp = string.Empty;

}

}

}

return result;

}

}

Итоговым методом является метод Main, использующий все приведенные выше методы, считывающий текст с файла, создающий два словаря, соответствующие двум методам кодирования символов. Формат составления словаря: символ – код.

static void Main()

{

Dictionary<string, string> ShennonCode = new Dictionary<string, string>()

{

{" ", "111" },

{"о", "110" },

{"а", "1011" },

{"е", "1010" },

{"н", "1001" },

{"и", "1000" },

{"т", "0111" },

{"с", "01101" },

{"р", "01100" },

{"л", "0101" },

{"в", "01001" },

{"к", "01000" },

{"м", "00111" },

{"п", "00110" },

{"у", "00101" },

{"д", "001001" },

{"я", "001000" },

{"ь", "000111" },

{"ы", "000110" },

{"б", "000101" },

{"з", "000100" },

{"г", "000011" },

{"ч", "0000101" },

{"й", "0000100" },

{"ш", "0000011" },

{"х", "0000010" },

{"ю", "00000011" },

{"ж", "00000010" },

{"щ", "000000011" },

{"э", "000000010" },

{"ц", "000000001" },

{"ф", "000000000" }

};

Dictionary<string, string> HaffmanCode = new Dictionary<string, string>()

{

{" ", "111" },

{"о", "1101" },

{"а", "1011" },

{"е", "1001" },

{"н", "0111" },

{"и", "0101" },

{"т", "0011" },

{"с", "0001" },

{"р", "11001" },

{"л", "11000" },

{"в", "10101" },

{"к", "10001" },

{"м", "01101" },

{"п", "01001" },

{"у", "01000" },

{"д", "00101" },

{"я", "00001" },

{"ь", "00000" },

{"ы", "101001" },

{"б", "100001" },

{"з", "100000" },

{"г", "011001" },

{"ч", "001001" },

{"й", "001000" },

{"ш", "1010001" },

{"х", "0110001" },

{"ю", "10100001" },

{"ж", "01100001" },

{"щ", "101000001" },

{"э", "101000000" },

{"ц", "011000001" },

{"ф", "011000000" }

};

string filePath = "input.txt";

string text = File.ReadAllText(filePath).ToLower();

;

text = DeletePunctuation.DeletePunc(text);

//Метод Шеннона-Фано

Console.WriteLine("Метод Шеннона-Фано");

Console.WriteLine("----------------------------------------------------------------" +

"-----------------------------------------" +

"---------------");

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Исходный текст: \n");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine(text);

Console.WriteLine("----------------------------------------------------------------" +

"-----------------------------------------" +

"---------------");

text = Coding.Encoding(text, ShennonCode);

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine("Закодированный текст: \n");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine(text);

Console.WriteLine("----------------------------------------------------------------" +

"-----------------------------------------" +

"---------------");

text = Coding.Decoding(text, ShennonCode);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Декодированный текст: \n");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine(text);

Console.WriteLine("----------------------------------------------------------------" +

"-----------------------------------------" +

"---------------");

//Метод Хаффмана

text = File.ReadAllText(filePath).ToLower();

text = DeletePunctuation.DeletePunc(text);

Console.WriteLine("Метод Хаффмана");

Console.WriteLine("----------------------------------------------------------------" +

"-----------------------------------------" +

"---------------");

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("Исходный текст: \n");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine(text);

Console.WriteLine("----------------------------------------------------------------" +

"-----------------------------------------" +

"---------------");

text = Coding.Encoding(text, HaffmanCode);

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine("Закодированный текст: \n");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine(text);

Console.WriteLine("----------------------------------------------------------------" +

"-----------------------------------------" +

"---------------");

text = Coding.Decoding(text, HaffmanCode);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Декодированный текст: \n");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine(text);

Console.WriteLine("----------------------------------------------------------------" +

"-----------------------------------------" +

"---------------");

}

}

На рисунке 1 представлен исходный текст.

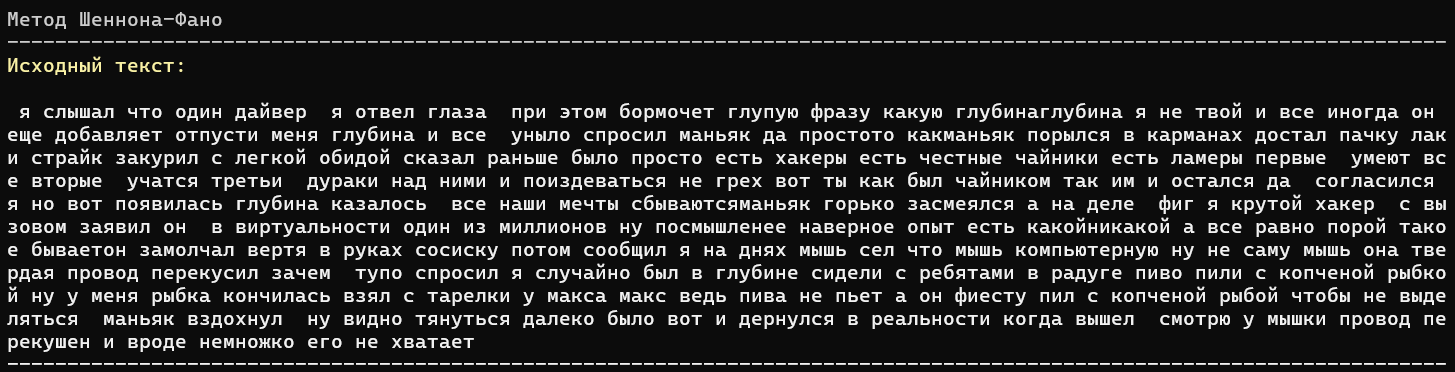


Рисунок 1 – Исходный текст

Рисунки 2-3 и рисунки 4-5 представляют закодированные и декодированные варианты исходного текста методами Шеннона-Фано и Хаффмана соответственно.

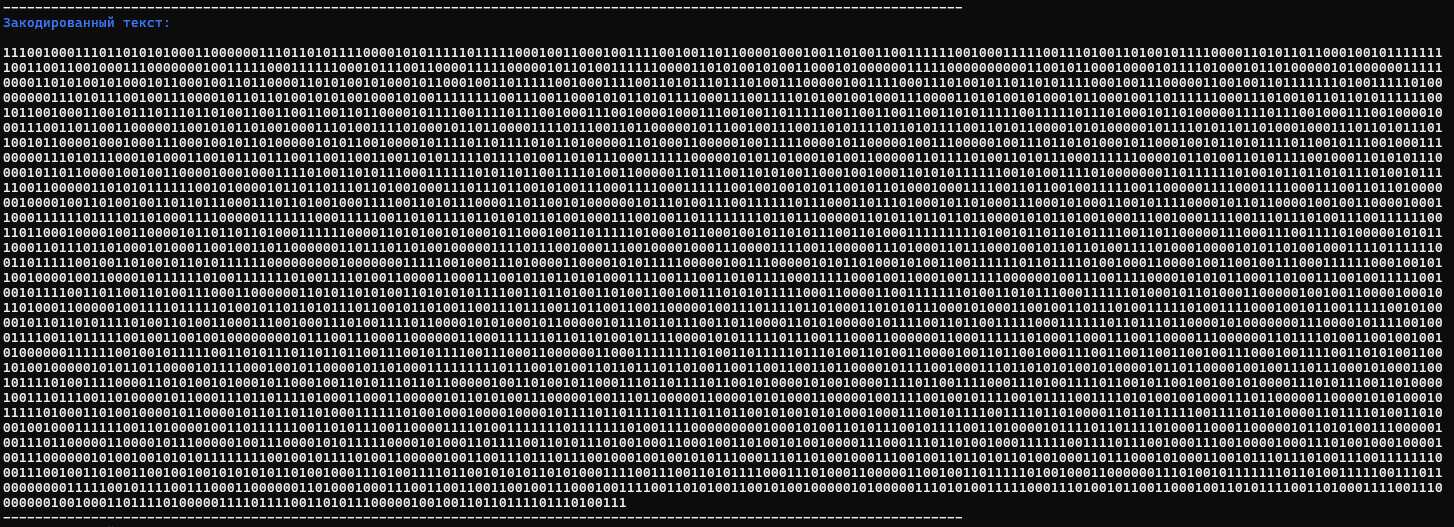


Рисунок 2 ­– Кодирование методом Шеннона-Фано

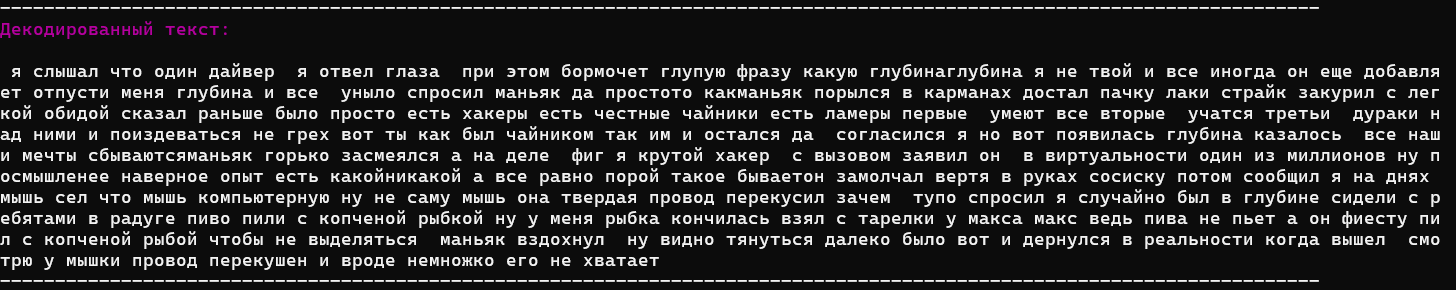


Рисунок 3 – Декодирование текста

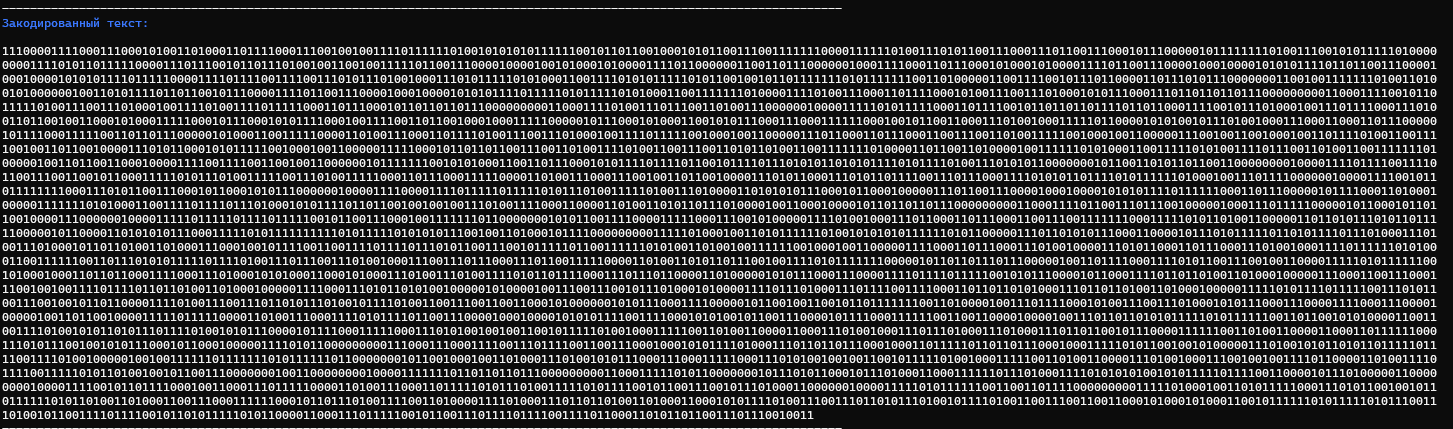


Рисунок 4 – Кодирование методом Хаффмана

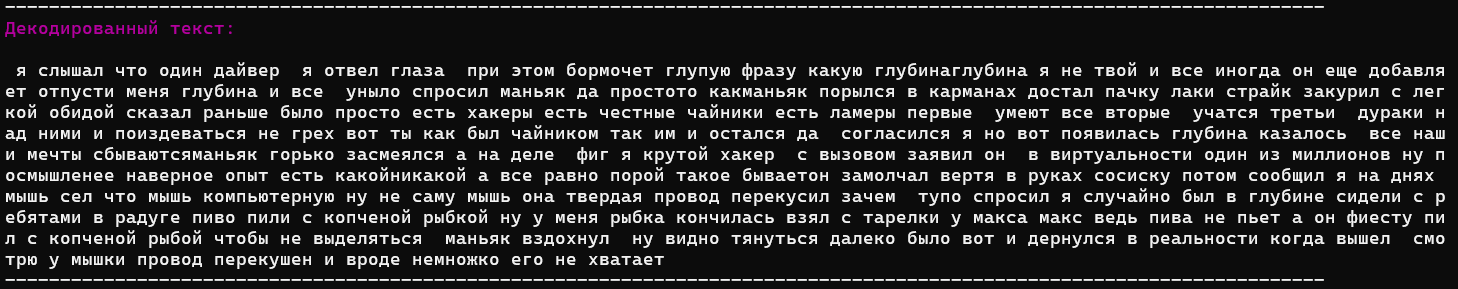


Рисунок 5 – Декодирование текста

# ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что лежит в основе определения избыточности?   
 Определение избыточности информации основывается на понятии энтропии, которая представляет собой меру неопределенности источника информации.

2. Какие частные избыточности вы знаете?

Частные избыточности включают систематическую избыточность (когда код содержит дополнительные символы для обеспечения надежности передачи), временную избыточность (когда информация передается с избыточной скоростью) и пространственную избыточность (когда используется дополнительное пространство для кодирования).

3. В чем заключается оптимальное неравномерное кодирование?

Оптимальное неравномерное кодирование заключается в том, что более вероятные символы кодируются более короткими кодами, что позволяет снизить среднюю длину кода.

4. Как проводится кодирование информации методом Шеннона– Фано?

Кодирование методом Шеннона-Фано осуществляется путем разделения символов на группы по вероятностям и последующего присвоения кодовых слов.

5. Как кодируется информация методом Хаффмана?

Метод Хаффмана использует жадный алгоритм для построения оптимального префиксного кода, где часто встречающиеся символы кодируются более короткими кодами.

6. Как определяется средняя длина кода?

Средняя длина кода определяется как сумма произведений вероятности каждого символа на его длину кода.

7. Как вычисляются коэффициенты статистического сжатия и коэффициент относительной эффективности?

Коэффициент статистического сжатия вычисляется как отношение исходного количества информации к количеству бит после сжатия. Коэффициент относительной эффективности показывает, насколько полученное сжатие близко к оптимальному.

8. Как формируется информационная модель канала связи?

Информационная модель канала связи формируется на основе передатчика, канала и приемника, учитывая возможные помехи и шум.

9. Как определяется пропускная способности канала связи?

Пропускная способность канала связи определяется максимальной скоростью передачи данных через канал.

10. В чем заключаются недостатки оптимального кодирования?

Недостатки оптимального кодирования могут включать сложность вычислений для построения оптимальных кодов, а также необходимость передачи дополнительной информации для декодирования.

11. В чем суть основной теоремы Шеннона о кодировании для дискретного канала без помех?

Основная теорема Шеннона о кодировании для дискретного канала без помех утверждает, что существует способ построения кода с произвольно малой вероятностью ошибки декодирования при условии, что скорость передачи информации меньше энтропии канала.

12. В чем суть теоремы Шеннона для побуквенного кодирования?

Теорема Шеннона для побуквенного кодирования утверждает, что при передаче информации через канал с шумом можно достичь произвольно малой вероятности ошибки декодирования при скорости передачи, не превышающей ёмкость канала.

# ВЫВОДЫ

В ходе лабораторной работы были изучены и применены методы Хаффмана, Шеннона-Фано, что позволило расширить знания о кодировании.

Коэффициент статического сжатия по Хаффману:

Ксс = = = 1,13217> 1

Коэффициент относительной эффективности по Хаффману:

Коэ = = = 0,99033 <1

Коэффициент статического сжатия по Шеннону-Фано:

Ксс = = = 1,12844> 1

Коэффициент относительной эффективности по Шеннону-Фано:

Коэ = = = 0,98707 <1

Ксс(Хаффман) 1,13217 > Ксс(Шеннон-Фано) 1,12844;

Коэ(Хаффман) 0,99033 > Коэ(Шеннон-Фано) 0,98707

Таким образом, можно сделать вывод, что кодирование выполнено правильно. Также из коэффициентов можно сделать вывод, что метод Хаффмана эффективнее метода Шеннона-Фано.