Лабораторная работа №4

Тема: Простейшие методы кодирования информации

1. Цель работы

Целью лабораторной работы является изучение простейших методов кодирования информации, анализ их эффективности и надёжности, и почему простые, интуитивные способы кодирования оказываются неэффективными.

2. Преемственность и программное продолжение

Данная лабораторная работа является программным продолжением лабораторных работ №1–3. Студенту разрешается и рекомендуется дополнять и расширять ранее разработанную программу. Создание новой программы с нуля не является обязательным.

Используются:

- алфавит источника информации;
- вероятности символов;
- энтропия источника;
- ранее реализованные функции анализа и кодирования.

3 Теория

На предыдущих этапах изучения курса «Основы кодирования» мы рассматривали, что такое информация, как измеряется её количество, что такое энтропия источника и почему она является фундаментальной характеристикой сообщений.

Однако до этого момента кодирование рассматривалось лишь косвенно — через длины сообщений и теоретические пределы

3.1 Равномерный двоичный код

Равномерным называется кодирование, при котором всем символам алфавита назначаются кодовые слова одинаковой длины. В равномерном коде длина кодового слова определяется формулой и одинакова для всех символов. Это самый простой и наглядный способ кодирования.

Если мощность алфавита равна |A|, то минимальная длина кодового слова l должна удовлетворять условию:

$$2^l \geq |A|.$$

Отсюда:

$$l = \lceil \log_2 |A| \rceil.$$

Таким образом, длина кодового слова определяется исключительно мощностью алфавита, а не длиной сообщения или вероятностями символов.

Часто ошибочно пытаются кодировать «сообщение целиком», не выделяя алфавит. Однако код строится НЕ для сообщения, а для алфавита источника.

Пример:

Сообщение: ABACA

Алфавит — это множество **различных** символов, встречающихся в сообщении:

Алфавит: {A, B, C}

Мощность алфавита:

|A| = 3

Кодирование строится для A, B и C, а не для всей строки ABACA.

A = {A, B, C}.

Определение минимальной длины кодового слова

Так как мы используем **двоичное кодирование**, количество возможных кодовых слов длины l равно 2^l.

Нужно, чтобы:

$$2^l \geq |A|.$$

Подставляем:

$2^1 = 2 < 3$
$2^2 = 4 \geq 3$

Следовательно,
l = 2 { бита}

👉 **Каждому символу будет назначен двоичный код длины 2 бита.**

Назначение равномерных кодов

Назначим коды **произвольно**, но с соблюдением условий:

- коды разной длины запрещены;
- одинаковые коды запрещены.

Например:

Символ	Код
A	00
B	01
C	10

Все кодовые слова:

- двоичные,
- одинаковой длины (2 бита),
- различимы.

Это и есть **равномерный код**.

Кодирование сообщения

Исходное сообщение:

ABAC
Заменяем каждый символ его кодом:

- A → 00
- B → 01
- A → 00
- C → 10

Получаем закодированное сообщение:
00010010
Длина закодированного сообщения

- длина сообщения: N = 4 символа;
- длина кода: l = 2 бита.

Общая длина:

$$B = N \cdot l = 4 \cdot 2 = 8 \text{ бит}$$

Этот пример показывает:

- код строится **для алфавита**, а не для сообщения;
- длина кода зависит от |A|, а не от длины сообщения;
- равномерное кодирование:
  - простое,
  - наглядное,
  - не учитывает вероятности символов;
- такой код **не обязательно эффективен**, но всегда корректен.

👉 Именно поэтому равномерное кодирование используется как **отправная точка**, а затем заменяется статистическими методами (Шеннон–Фано, Хаффман).

3.2. Табличное кодирование

При табличном кодировании формируется явная таблица соответствия
между символами алфавита и их кодовыми словами.

Каждому символу назначается уникальный двоичный код фиксированной длины.
Основное требование — различимость кодовых слов,
то есть отсутствие одинаковых кодов у разных символов.
Табличный код — это код, где
соответствие «символ → код» задаётся явно
и **не обязано следовать формуле**.

Назначение кодов (вручную)

Символ	Код
A	0

Символ	Код
B	10
C	11

Обратите внимание:

- коды **разной длины**;
- формула для длины **не использовалась**;
- код задан **таблицей**.

Кодирование сообщения

A B A C

0 10 0 11

Закодированное сообщение:

010011

Ключевая мысль

В табличном коде
длины кодовых слов могут отличаться,
если это не запрещено условиями.

---

3 Сравнение (самое важное)

Критерий	Равномерный код	Табличный код
Как задаётся	По формуле	Вручную
Длины кодов	Одинаковые	Могут быть разными
Обязателен ли	Да (в ЛР4)	Нет
Использует вероятности	❌ Нет	❌ Нет
Может быть неравномерным	❌ Нет	☑ Да
Основа для Шеннона/Хаффмана	Косвенно	**Да**

Главный методический вывод (обязательно проговаривать)

- ◇ **Равномерный код — это частный случай табличного кода**
- ◇ **Не каждый табличный код является равномерным**

3.3 Позиционное кодирование

Позиционный код — это код, где:

- каждому символу назначается **номер в алфавите**;
- код символа — это **двоичная запись этого номера**;
- длина кода фиксируется так же, как и в равномерном коде.

Зададим порядок:

A → 0

B → 1

C → 2

D → 3

Длина кодового слова

$$l = \lceil \log_2 4 \rceil = 2$$

(та же самая, что и у равномерного кода)

Позиционный код

Символ	Номер	Код
A	0	00
B	1	01
C	2	10
D	3	11

Обратите внимание:
коды **не выбирались вручную**, они **получены автоматически**.

Кодирование сообщения

B A C A D
01 00 10 00 11

Закодированное сообщение:
0100100011

◇ **Результат**

– в данном примере **одинаковый битовый поток (как и в равномерном кодирования представленный выше)**

◇ **Смысл**

– **принцип построения кода разный**

Главное различие равномерный и позиционный коды

Критерий	Равномерный код	Позиционный код
Что задаётся	Длина кода	Порядок символов
Как назначаются коды	Вручную	По формуле
Есть свобода выбора	Да	Нет
Алгоритмичность	✗	☑
Частный случай	—	**Равномерного**

◇ Равномерный код отвечает на вопрос
«какой длины должны быть коды?»

◇ Позиционный код отвечает на вопрос
«как автоматически эти коды получить?»

Именно поэтому:

- в данной лабораторной работе сначала вводится равномерное кодирование;

- затем позиционное — как его алгоритмическая форма;
- и только потом — статистические методы (Шеннон–Фано, Хаффман).

3.4 Простейшие коды с контролем ошибок

До этого момента (равномерный, табличный, позиционный коды):
- мы **кодировали символы**;
- но **не защищались от искажений** при передаче.

В реальных каналах связи:
- биты могут искажаться;
- возможны одиночные ошибки (0 → 1 или 1 → 0).

👉 **Контроль ошибок** позволяет:
- обнаружить ошибку,
- а иногда — и исправить её.

Самый простой способ контроля: избыточный бит

К информационным битам добавляется **контрольный бит**, который:
- не несёт новой информации;
- служит для проверки корректности передачи.

Это **простейший код с избыточностью**.

Пример: код с битом чётности (even parity)

Шаг 1. Исходное двоичное слово

Пусть передаётся 4-битное слово:
1011
Количество единиц:
1 + 0 + 1 + 1 = 3 (нечётное)

Шаг 2. Формирование контрольного бита

Используем **бит чётности** (even parity):
- если число единиц нечётное → добавляем 1;
- если чётное → добавляем 0.

В нашем случае:
p = 1

Шаг 3. Передаваемое кодовое слово

Добавляем контрольный бит в конец:

1011 | 1
Итого передаётся:
10111

4. Проверка на приёме

Ситуация 1: ошибок нет

Принято:
10111
Число единиц = 4 (чётное)
✓ Ошибок **не обнаружено**

Ситуация 2: одиночная ошибка

Пусть один бит исказился:
10011
Число единиц = 3 (нечётное)
❌ **Обнаружена ошибка**

Что этот код МОЖЕТ и НЕ МОЖЕТ

✓ **Может:**

- обнаружить **любую одиночную ошибку**;
- прост в реализации;
- почти не требует вычислений.

❌ **Не может:**

- указать, **где именно ошибка**;
- исправить ошибку;
- обнаружить **чётное число ошибок**.

Почему это всё ещё «простейший код»

Этот код:

- добавляет **избыточность**;
- не использует сложную математику;
- является первым шагом к:
  - кодам Хэмминга,
  - линейным блоковым кодам,
  - помехоустойчивому кодированию.

◇ Кодирование бывает:

- **для представления** информации;
- **для надёжности** передачи.

◇ Контроль ошибок всегда требует **избыточности**.

Пример: неравномерное кодирование без Хаффмана, где мы вручную назначаем более короткие коды более вероятным символам и сравниваем со равномерным.

Дано (алфавит и вероятности. Пусть источник выдаёт 4 символа:

Символ	**p_i**
A	0.50
B	0.25
C	0.125
D	0.125

2) Неравномерное кодирование (без Хаффмана)

Принцип

Назначаем **короче** тем, кто встречается **чаще**.

Чтобы код был корректным и однозначно декодируемым, удобно сделать его **префиксным** (даже если мы не используем алгоритм Хаффмана).

Назначим коды «вручную» так:

Символ	**p_i**	**Код (неравномерный)**	**l_i**
A	0.50	0	1
B	0.25	10	2
C	0.125	110	3
D	0.125	111	3

Проверка префиксности (коротко)

- «0» не является началом «10», «110», «111»
- «10» не является началом «110», «111»
- «110» и «111» не являются префиксами друг друга

☑ код префиксный → декодирование однозначно

Средняя длина неравномерного кода и сравнение с равномерным

Средняя длина неравномерного кода вычисляется по формуле:

$\bar{L}\_\text{неравн} = \Sigma\, p_i \cdot l_i$

Рассчитаем вклад каждого символа:
• $0.50 \cdot 1 = 0.50$
• $0.25 \cdot 2 = 0.50$
• $0.125 \cdot 3 = 0.375$

• $0.125 \cdot 3 = 0.375$

Суммируя полученные значения, получаем:

$\bar{L}\_неравн = 0.50 + 0.50 + 0.375 + 0.375 = 1.75$ бит/символ

Таким образом, средняя длина неравномерного кода составляет 1.75 бит на символ.

Сравнение с равномерным кодированием

Для равномерного кодирования при мощности алфавита $|A| = 4$ средняя длина кода равна:

$\bar{L}\_равн = 2.00$ бит/символ

Для неравномерного (ручного) кодирования:

$\bar{L}\_неравн = 1.75$ бит/символ

Выигрыш по средней длине кода:

$\Delta\bar{L} = 2.00 - 1.75 = 0.25$ бит/символ

Относительное уменьшение средней длины:

$0.25 / 2.00 = 12.5\%$

Следовательно, за счёт разумного назначения более коротких кодовых слов более вероятным символам удалось уменьшить среднюю длину кода на 12.5% без применения алгоритма Хаффмана.

☑ **Уменьшили среднюю длину на 12.5%** без Хаффмана — просто разумным назначением кодов по вероятностям.

Почему нельзя сравнивать равномерный код с энтропией напрямую

Энтропия — это теоретический предел,
а равномерный код не использует вероятности символов.

Пример:
Если один символ встречается в 90% случаев,
а остальные редко,
равномерный код всё равно выделяет им одинаковое число бит,

что приводит к избыточности.

Почему выигрыш Шеннона может быть небольшим

Выигрыш в битах кажется «маленьким».
Однако даже 1–2% выигрыша при больших объёмах данных
имеют принципиальное значение.
Пример:
Экономия 7 бит на 86 символов
даёт тысячебитную экономию на больших потоках данных.

Таким образом:

1. Равномерный код прост, но не использует статистику → часто избыточен.
2. Если частые символы кодировать короче, средняя длина уменьшается.
3. Чтобы код был однозначно декодируемым, удобно требовать префиксность.
4. Хаффман — это уже **алгоритм**, который делает такое назначение **оптимально**.

4. Выбранные методы кодирования

В лабораторной работе рассматриваются следующие простейшие методы кодирования:
1) равномерный двоичный код;
2) позиционный код фиксированной длины;
3) табличное символьное кодирование;
4) простейшие коды с контролем ошибок;
5) неравномерное кодирование на основе вероятностей.

5. Задания

Все задания выполняются последовательно и в рамках одной программы.

5.1 Позиционный код фиксированной длины

Реализовать позиционный код с основанием, отличным от двух (например, троичный). Сравнить длину закодированного сообщения с двоичным кодом.

5.2 Табличное символьное кодирование

Сформировать таблицу соответствия символов алфавита и кодовых слов. Закодировать сообщение с использованием таблицы.

5.3 Простейшие коды с контролем ошибок

Добавить к кодовым словам бит чётности или реализовать удвоение битов. Оценить увеличение длины сообщения и повышение надёжности.

5.4 Неравномерное кодирование

Используя вероятности символов, назначить более короткие кодовые слова более вероятным символам без применения алгоритма Хаффмана. Сравнить среднюю длину кода с равномерным.

5.5 Требования к программе на Python

Программа должна быть продолжением предыдущих лабораторных работ и включать:

- реализацию всех выбранных методов кодирования;
- использование алфавита и вероятностей символов;
- расчёт длины закодированного сообщения и средней длины кода;
- вывод сравнительной таблицы результатов.

Использование готовых библиотек кодирования запрещено!

6. Требования к отчёту

Отчёт должен содержать описание реализованных методов кодирования, результаты расчётов, сравнительный анализ и выводы. Исходный код программы приводится в приложении.

7. Контрольные вопросы

1. Что называют простейшими методами кодирования?
2. В чём недостаток равномерного кодирования?
3. Как основание системы счисления влияет на длину кода?
4. Зачем используются коды с контролем ошибок?
5. Почему неравномерные коды могут быть эффективнее равномерных?

8. Выводы

В выводах необходимо сравнить эффективность различных методов кодирования и обосновать необходимость оптимальных кодов.