

ОСНОВНЫЕ МЕТОДЫ КОДИРОВАНИЯ ДАННЫХ

НЕОБХОДИМЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Теория кодирования и теория информации возникли в начале XX века.

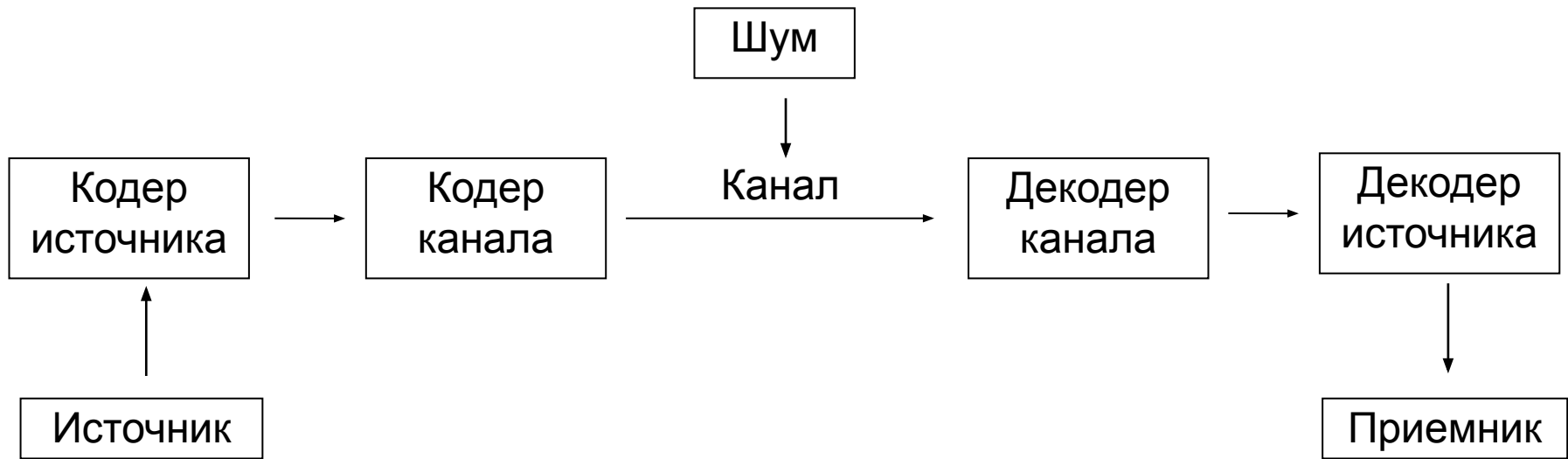
Начало развитию этих теорий как научных дисциплин положило появление в **1948** г. статей *Клода Шеннона*, которые заложили фундамент для дальнейших исследований в этой области.

Кодирование – способ представления информации в виде, удобном для хранения и передачи.

В связи с развитием информационных технологий кодирование является центральным вопросом при решении самых разных **задач программирования**, таких как:

- **представление данных** произвольной структуры (числа, текст, графика) в памяти компьютера;
- **обеспечение помехоустойчивости** при передаче данных по каналам связи;
- **сжатие информации** в базах данных.

Основной моделью, которую изучает теория информации, является ***модель системы передачи сигналов:***



Начальным звеном в приведенной выше модели является ***источник информации.***

Рассмотрим ***дискретные источники без памяти,*** в которых ***выходом*** является **последовательность символов некоторого фиксированного алфавита.**

Определение. Множество всех различных символов, порождаемых некоторым источником, называется ***алфавитом источника***, количество символов в этом множестве – ***размер алфавита источника***.

Например, ***текст на русском языке*** порождается источником с алфавитом из 33 русских букв, пробела и знаков препинания.

Определение.

Кодирование дискретного источника заключается в сопоставлении символов алфавита ***A*** источника символам некоторого другого алфавита ***B***.

Определение. Обычно символу исходного алфавита *A* ставится в соответствие не один, а группа символов алфавита *B*, которая называется *кодовым словом*.

Определение. *Кодовый алфавит* – множество различных символов, используемых для записи кодовых слов.

Определение. *Кодом* называется совокупность всех кодовых слов, применяемых для представления порождаемых источником символов.

Пример. *Азбука Морзе* является известным кодом из символов телеграфного алфавита, в котором буквам русского языка соответствуют

Далее будем рассматривать **двоичное кодирование**, т.е. размер кодового алфавита равен 2.

Определение. Конечную последовательность битов (нулей или единиц) назовем **кодовым словом**, а

количество битов в этой последовательности – **длиной кодового слова**.

Пример. Код ASCII (**американский стандартный код для обмена информацией**) каждому символу ставит в однозначное соответствие **кодовое слово длиной 8 бит**.

Дадим строгое определение кодирования.

Пусть даны алфавит источника $A = \{a_1, a_2, \dots, a_n\}$,
кодировый алфавит $B = \{b_1, b_2, \dots, b_n\}$.

Обозначим через A^* множество всевозможных
последовательностей в алфавите A .

Множество всех сообщений в алфавите A
обозначим S .

Тогда отображение $F: S \rightarrow B^*$, которое
преобразует множество сообщений S в кодовые
слова в алфавите B , называется
кодированием.

Обратное отображение F^{-1} (если оно
существует) называется ***декодированием***.

Задача кодирования сообщения ставится следующим образом:

требуется при заданных алфавитах A и B и множестве сообщений S найти такое кодирование E , которое обладает определенными свойствами и оптимально в некотором смысле.

Свойства, которые требуются от кодирования, могут быть различными. Приведем некоторые из них:

- существование декодирования;
- помехоустойчивость или исправление ошибок при кодировании;
- обладает заданной трудоемкостью (время, объем памяти).

Известны **два класса методов кодирования** дискретного источника информации: **равномерное и неравномерное кодирование**.

Под **равномерным кодированием** понимается использование кодов со словами постоянной длины.

Для того чтобы декодирование равномерного кода было возможным, разным символам алфавита источника должны соответствовать разные кодовые слова.

При этом **длина кодового слова** должна быть не меньше $\lceil \log_n m \rceil$ символов, где m – размер исходного алфавита, n – размер кодового алфавита.

Пример. Для кодирования источника, порождающего **26 букв латинского алфавита**, равномерным двоичным кодом требуется построить **кодовые слова** длиной не меньше $\lceil \log_2 m \rceil = 5$ бит.

При **неравномерном кодировании источника** используются кодовые слова разной длины.

Причем кодовые слова обычно строятся так, что **часто встречающиеся символы** кодируются более короткими кодовыми словами,

а **редко встречающиеся символы** – более длинными кодовыми словами.

За счет этого и достигается **«сжатие» данных**.

Под **сжатием данных** понимается компактное представление данных, достигаемое за счет **избыточности информации**, содержащейся в сообщениях.

Большое значение для практического использования имеет **неискажающее сжатие**, позволяющее полностью восстановить исходное сообщение.

При **неискажающем сжатии** происходит кодирование сообщения перед началом передачи или хранения, а после окончания процесса сообщение однозначно декодируется.

Это соответствует **модели канала без шума** (помех).

Методы сжатия данных можно разделить на две группы: **статические методы** и **адаптивные методы**.

Статические методы сжатия данных предназначены для кодирования конкретных источников информации с известной статистической структурой, порождающих определенное множество сообщений.

Эти методы базируются на знании статистической структуры исходных данных.

К наиболее известным статическим методам сжатия относятся **коды Хаффмана, Шеннона, Фано, Гилберта-Мура, арифметический код** и другие методы, которые используют известные сведения о вероятностях порождения источником

Если статистика источника информации неизвестна или изменяется с течением времени, то для кодирования сообщений такого источника применяются ***адаптивные методы сжатия***.

В ***адаптивных методах*** при кодировании очередного символа текста используются сведения о ранее закодированной части сообщения для оценки вероятности появления очередного символа.

В процессе кодирования адаптивные методы **«настраиваются» на статистическую структуру кодируемых сообщений**, т.е. коды символов меняются в зависимости от накопленной статистики данных.

Это позволяет адаптивным методам эффективно и быстро кодировать сообщение за один просмотр.

КОДИРОВАНИЕ ЦЕЛЫХ ЧИСЕЛ (CODING OF INTEGERS)

Рассмотрим *семейство методов кодирования*, не учитывающих вероятности появления символов источника.

Поскольку все символы алфавита источника можно пронумеровать, то в будем считать, что алфавит источника состоит из целых чисел.

Каждому целому числу из определенного диапазона ставится в соответствие **свое кодовое слово**.

Поэтому эту группу методов также называют **представлением целых чисел** (*representation of integers*).

Основная идея кодирования целых чисел:

отдельно кодировать **порядок значения** элемента («экспоненту») и отдельно – **значащие цифры** значения («мантиссу»_{*i*}).

Значащие цифры мантиссы начинаются со старшей ненулевой цифры,

порядок числа определяется позицией старшей ненулевой цифры в двоичной записи числа.

Как и при десятичной записи,

порядок равен числу цифр в записи числа без предшествующих незначащих нулей.

Пример. Двоичное число **000001101**

Порядок равен **4**, мантисса – **1101**

Рассмотрим две группы методов кодирования целых чисел.

Условно их можно обозначить так:

- ***Fixed + Variable***

(фиксированная длина порядка + переменная длина мантиссы)

- ***Variable + Variable***

(переменная длина порядка + переменная длина мантиссы)

Коды класса Fixed + Variable

В кодах класса Fixed + Variable

под запись значения **порядка числа** отводится *фиксированное количество бит,*

а **значение порядка числа** определяет, сколько бит потребуется *под запись мантиссы.*

Для кодирования целого числа необходимо произвести с числом **две операции:**

- определение порядка числа
- выделение бит мантиссы

Можно также хранить в памяти **заранее построенную таблицу кодовых слов** и по ней получать код числа.

Пример. Код класса *Fixed + Variable*.

Пусть $R = 15$ – количество бит исходного числа.

Отведем $E = 4$ бита под **порядок**, т.к. $R \leq 2^4$.

При записи мантииссы можно **сэкономить 1 бит**:
не писать первую единицу, т.к. это всегда будет
только единица.

Таким образом, количество бит мантииссы **меньше**
на один бит, чем **значение порядка** числа.

Число	двоичное представление	кодовое слово	длина кодового слова
0	0000000000000000	0000	4
1	0000000000000001	0001	4
2	0000000000000010	0010 0	5
3	0000000000000011	0010 1	5
4	0000000000000100	0011 00	6
5	0000000000000101	0011 01	6
6	0000000000000110	0011 10	6
7	0000000000000111	0011 11	6
8	0000000000001000	0100 000	7
9	0000000000001001	0100 001	7
10	0000000000001010	0100 010	7
...
15	0000000000001111	0100 111	7
16	0000000000010000	0101 0000	8
17	0000000000010001	0101 0001	8
...

Коды класса Variable + Variable

В качестве кода числа берется двоичная

последовательность, построенная следующим образом:

несколько нулей (количество нулей равно значению порядка числа), затем **мантисса переменной длины**.

число	двоичное представление	кодовое слово	длина кодового слова
0	0000000000	1	1
1	0000000001	0 1	2
2	0000000010	00 10	4
3	0000000011	00 11	4
4	0000000100	000 100	6
5	0000000101	000 101	6
6	0000000110	000 110	6
7	0000000111	000 111	6
8	0000001000	0000 1000	8
9	0000001001	0000 1001	8
10	0000001010	0000 1010	8

Если в рассмотренном выше коде **исключить кодовое слово для нуля**, то можно **уменьшить длины кодовых слов на 1 бит, убрав первый ноль** во всех кодовых словах.

Таким образом строится **гамма-код Элиаса (γ-код Элиаса)**.

число	кодовое слово	длина кодового слова
1	1	1
2	0 10	3
3	0 11	3
4	00 100	5
5	00 101	5
6	00 110	5
7	00 111	5
8	000 1000	7
9	000 1001	7
10	000 1010	7
...	...	

Другим примером кода класса Variable + Variable является **омега-код Элиаса (ω -код Элиаса)**.

В нем первое значение (кодированное слово для единицы) задается отдельно.

Другие кодированные слова состоят из последовательности групп длиной $L_1 L_2 \dots L_m$, начинающихся с единицы.

Конец всей последовательности задается нулевым битом.

Длина первой группы составляет 2 бита, длина каждой следующей группы равна двоичному значению битов предыдущей группы плюс 1.

Значение битов последней группы является итоговым значением всей последовательности групп, т.е. первые $m-1$ групп служат лишь для указания длины последней группы, которая

ЧИСЛО	КОДОВОЕ СЛОВО	ДЛИНА КОДОВОГО СЛОВА
1	0	1
2	10 0	3
3	11 0	3
4	10 100 0	6
5	10 101 0	6
6	10 110 0	6
7	10 111 0	6
8	11 1000 0	7
9	11 1001 0	7
..
15	11 1111 0	7
16	10 100 10000 0	11
17	10 100 10001 0	11
..
31	10 100 11111 0	11
32	10 101 100000 0	12

В омега-коде Элиаса (P. Elias):

При кодировании формируется сначала последняя группа, затем предпоследняя и т.д., пока процесс не будет завершен.

При декодировании, наоборот, сначала считывается первая группа, по значению ее битов определяется длина следующей группы, или итоговое значение числа, если следующая группа – 0.

Рассмотренные типы кодов могут быть **эффективны в следующих случаях:**

- **Вероятности чисел убывают** с ростом значений элементов и их распределение близко к такому:

$$P(x) \geq P(x+1), \text{ при любом } x,$$

т.е. маленькие числа встречаются чаще, чем большие.

- **Диапазон значений входных элементов не ограничен или неизвестен.**

Например, при кодировании 32-битовых чисел реально большинство чисел маленькие, но могут быть и большие.

- При использовании **в составе других схем кодирования**, например, *кодирования длин*

Кодирование длин серий ***Run Length Encoding (RLE)***

Метод кодирования длин серий (RLE), предложенный ***П. Элиасом (P.Elias)***, при построении использует **коды целых чисел**.

Входной поток для кодирования рассматривается как последовательность из нулей и единиц.

Идея кодирования заключается в том, чтобы кодировать **последовательности одинаковых элементов** (например, нулей) как **целые числа**, указывающие **количество элементов** в этой последовательности.

Последовательность одинаковых элементов называется ***серией***, количество элементов в ней –

Пример. Входную последовательность (31 бит) можно разбить на серии, а затем закодировать их длины.

000000 1 00000 1 0000000 1 1 00000000 1

Длины серий нулей: 6 5 7 0 8

Используем, например, *γ-код Элиаса*.

Поскольку в *γ-коде Элиаса* нет кодового слова для нуля, то будем кодировать длину серии +1:

6 5 7 0 8 \Rightarrow 7 6 8 1 9

7 6 8 1 9 \Rightarrow 00111 00110 0001000 1 0001001

Длина полученной кодовой последовательности 25 бит.

Метод длин серий актуален для кодирования данных, в которых есть длинные

последовательности одиночных бит. В нашем