## Кодирование информации

**Кодирование** – процесс преобразования знаков одной знаковой системы в другую знаковую систему, удобную для хранения, передачи или обработки информации.

Обратный процесс – **декодирование**.

**Код** – совокупность знаков (символов) и система правил, с помощью которых информация может быть закодирована.

Код = кодовый алфавит + алгоритм кодирования

Если в кодовом алфавите меньше символов, чем в исходном (на практике почти всегда), то один символ исходного алфавита кодируется последовательностью кодовых символов – **кодовым словом**.

**Кодирование в двоичный формат**

***Числа***. Машинная сетка. Машинное слово, полуслово, байт. Дополнительный код. Представление чисел с плавающей запятой.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | –5 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 5 |

5d = 101b

–5d = –101b

1.

2.

37 598 845 =3,760 \* 107 = 3,760E07

66,77d = 1000010,11000101…b = 1,00001011000101…\*26

0,001b = 1,0\*2–3

Мантисса

Экспонента

Double (8 байт) смещение 1 1111 1111b

–66,77d = –1000010,11000101…b

Single

1. M = –1,00001011000101 E = +110
2. 00000110 (E)

+

01111111 (смещение)

\_\_\_\_\_\_\_\_\_

10000101

|  |  |  |  |
| --- | --- | --- | --- |
| 1100 0010 | 1000 0101 | 1000 101. | …….. |

Double

1. M = –1,00001011000101 E = +110
2. 00 0000 0110

+ 01 1111 1111

\_\_\_\_\_\_\_\_\_\_\_

10 0000 0101

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1100 0000 | 1010 0001 | 0110 … | … | … | … | … | … |

***Текст***. ASCII, Unicode.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **.0** | **.1** | **.2** | **.3** | **.4** | **.5** | **.6** | **.7** | **.8** | **.9** | **.A** | **.B** | **.C** | **.D** | **.E** | **.F** |
| **0.** | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS | TAB | LF | VT | FF | CR | SO | SI |
| **1.** | DLE | DC1 | DC2 | DC3 | DC4 | NAK | SYN | ETB | CAN | EM | SUB | ESC | FS | GS | RS | US |
| **2.** |  | ! | " | # | $ | % | & | ' | ( | ) | \* | + | , | — | . | / |
| **3.** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| **4.** | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| **5.** | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | \_ |
| **6.** | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| **7.** | p | q | r | s | t | u | v | w | x | y | z | { | | | } | ~ | DEL |

***Графика***. Палитра цветов, RGB.

RGB(255,0,0) – 3 байт/пикс.

|  |  |
| --- | --- |
|  |  |
|  |  |

(255,255,255) (255,0,0) (255,255,0)(0,0,255)

***Аудио и видео***. Кодеки.

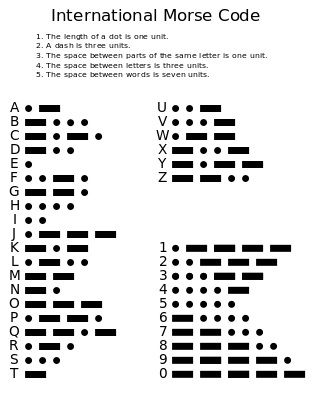
### Код постоянной длины

Все кодовые слова имеют постоянную длину – достаточную для кодирования. Чаще всего избыточный. Не учитывает частоту появления символов.

### Код переменной длины

Разные символы кодируются словами разной длины.

**Азбука Морзе**



Чем чаще встречается символ, тем меньше точек и тире используется. Для изучения и декодирования используется «дерево».

|  |  |  |  |
| --- | --- | --- | --- |
| T – | M – – | O – – – |  |
|  |
| G – – · | Q – – · – |
| Z – – · · |
| N – · | K – · – | Y – · – – |
| C – · – · |
| D – · · | X – · · – |
| B – · · · |
| E · | A · – | W · – – | J · – – – |
| P · – – · |
| R · – · |  |
| L · – · · |
| I · · | U · · – |  |
| F · · – · |
| S · · · | V · · · – |
| H · · · · |

· · · – – – · · ·

– –  – – –  · – ·  · · ·  ·

**Префиксный код** – построен так, что ни одно кодовое слово не является началом другого слова.

Пример: префиксный двухбитный код: 0, 10, 11.

10 0 11 10 11 0 10

**Первая теорема Шеннона** (в практической формулировке): при отсутствии помех передачи всегда возможен такой вариант передачи, при котором избыточность будет сколь угодно близкой к нулю.

При этом средняя длина двоичного кода сколь угодно близка к средней информации 1 символа исходного сообщения.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Буква** | **Частота** | **Буква** | **Частота** | **Буква** | **Частота** | **Буква** | **Частота** |
| «—» | 0,145 | Р | 0,041 | я | 0,019 | Х | 0,009 |
| о | 0,095 | В | 0,039 | ы | 0,016 | Ж | 0,008 |
| е | 0,074 | Л | 0,036 | з | 0,015 | Ю | 0,007 |
| а | 0,064 | К | 0,029 | ъ, ь | 0,015 | Ш | 0,006 |
| и | 0,064 | М | 0,026 | б | 0,015 | Ц | 0,004 |
| т | 0,056 | Д | 0,026 | г | 0,014 | Щ | 0,003 |
| н | 0,056 | П | 0,024 | ч | 0,013 | Э | 0,003 |
| с | 0,047 | У | 0,021 | й | 0,010 | Ф | 0,002 |

Сколько в среднем двоичных символов потребуется для кодирования русского алфавита? Сколько символов потребуется для кода постоянной длины?

**Алгоритм Шеннона-Фано**

1. Символы первичного алфавита выписывают по убыванию частот.
2. Символы полученного алфавита делят на две части, суммарные частоты символов которых максимально близки друг другу.
3. Для первой части алфавита присваивается двоичная цифра «0», второй части — «1».
4. Если получившаяся половина содержит более 1 символа, к ней снова применяется алгоритм с шага 1 (рекурсивно).

Пример

|  |  |  |
| --- | --- | --- |
| ai | pi | Код |
| ● | 0,4 | 0 |
| ─ | 0,2 | 100 |
| ■ | 0,15 | 101 |
| ► | 0,1 | 110 |
| ◄ | 0,1 | 1110 |
| ▲ | 0,05 | 1111 |

**101111100**100100011100101

■●◄●──●◄●■

■▲●●…

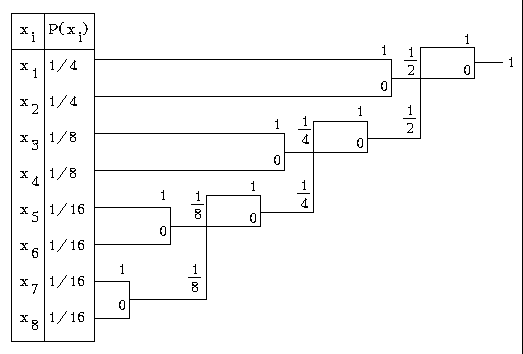
Недостаток – низкая устойчивость к ошибкам. Искажение или потеря одного бита «портит» все сообщение до конца.

**Алгоритм Хаффмана**

В ходе алгоритма строится кодировочное дерево. Сортировать символы необязательно, но так удобнее.

1. Каждый символ соответствует листьям дерева.
2. Выбираются два узла с наименьшими частотами.
3. Одному из них в код записывается 1, другому – 0.
4. Выбранные узлы объединяются в один узел, их частоты суммируются.
5. Если осталось больше 1 узла, алгоритм повторяется с шага 1.

|  |
| --- |
| 01 |
| 00 |
| 101 |
| 100 |
| 1101 |
| 1100 |
| 1111 |
| 1110 |



Сжатие данных по Хаффману применяется при сжатии фото- и видеоизображений (JPEG, MPEG), в архиваторах (PKZIP, LZH и др.)

**Сжатие** – кодирование с целью уменьшения объема информации. Варианты:

* без потерь;
* с потерями.

Рассмотренные алгоритмы не учитывают сочетания символов.

**Алгоритм сжатия RLE**

Простейший алгоритм сжатия без потерь. Последовательность одинаковых символов кодируется парой символ-количество повторов. используется в графических файлах с большими плоскостями одного цвета (BMP).

**Алгоритм сжатия LZW (Ле́мпеля — Зи́ва — Ве́лча)**

В процессе сжатия составляется **словарь** сочетаний символов (фраз).

1. Словарь заполняется односимвольными фразами. Текущая фраза W пуста.
2. Считать очередной символ K из кодируемого сообщения.
3. Если фраза WK присутствует в словаре, то W:=WK и вернуться к шагу 2.
4. Иначе (новая фраза) вписать WK словарь, вписать код W в результат, W:=K.
5. Если K = КОНЕЦ\_СООБЩЕНИЯ, то вписать в результат код K, иначе возврат к шагу 2.

В словаре односимвольные фразы кодируются кодом постоянной длины. В дальнейшем, при записи новых фраз код может удлиняться. При этом в результат тоже будет записываться больше символов.

**TOBEORNOTTOBEORTOBEORNOT#**

Мощность алфавита: 27 символов.

Длина сообщения: 25 символов.

Без сжатия: 5\*25 = 125 бит

Исходный словарь:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | **#** | 00000 | 9 | **I** | 01001 | 18 | **R** | 10010 |
| 1 | **A** | 00001 | 10 | **J** | 01010 | 19 | **S** | 10011 |
| 2 | **B** | 00010 | 11 | **K** | 01011 | 20 | **T** | 10100 |
| 3 | **C** | 00011 | 12 | **L** | 01100 | 21 | **U** | 10101 |
| 4 | **D** | 00100 | 13 | **M** | 01101 | 22 | **V** | 10110 |
| 5 | **E** | 00101 | 14 | **N** | 01110 | 23 | **W** | 10111 |
| 6 | **F** | 00110 | 15 | **O** | 01111 | 24 | **X** | 11000 |
| 7 | **G** | 00111 | 16 | **P** | 10000 | 25 | **Y** | 11001 |
| 8 | **H** | 01000 | 17 | **Q** | 10001 | 26 | **Z** | 11010 |

Сжатие:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Шаг | W | K | WK | В словарь | | | В результат |
|  |  | T | T |  |  |  |  |
|  | T | O | TO | 27 | **TO** | 11011 | 10100 |
|  | O | B | OB | 28 | **OB** | 11100 | 01111 |
|  | B | E | BE | 29 | **BE** | 11101 | 00010 |
|  | E | O | EO | 30 | **EO** | 11110 | 00101 |
|  | O | R | OR | 31 | **OR** | 11111 | 01111 |
|  | R | N | RN | 32 | **RN** | 100000 | 10010 |
|  | N | O | NO | 33 | **NO** | 100001 | 001110 |
|  | O | T | OT | 34 | **OT** | 100010 | 001111 |
|  | T | T | TT | 35 | **TT** | 100011 | 010100 |
|  | T | O | TO |  |  |  |  |
|  | TO | B | TOB | 36 | **TOB** | 100100 | 011011 |
|  | B | E | BE |  |  |  |  |
|  | BE | O | BEO | 37 | **BEO** | 100101 | 011101 |
|  | O | R | OR |  |  |  |  |
|  | OR | T | ORT | 38 | **ORT** | 100110 | 011111 |
|  | T | O | TO |  |  |  |  |
|  | TO | B | TOB |  |  |  |  |
|  | TOB | E | TOBE | 39 | **TOBE** | 100111 | 100100 |
|  | E | O | EO |  |  |  |  |
|  | EO | R | EOR | 40 | **EOR** | 101000 | 011110 |
|  | R | N | RN |  |  |  |  |
|  | RN | O | RNO | 41 | **RNO** | 101001 | 100000 |
|  | O | T | OT |  |  |  |  |
|  | OT | # | OT# | 42 | **OT#** | 101010 | 100010  000000 |

Результат:

101000111100010001010111110010001110001111010100011011011101011111100100011110100000100010000000

96 бит

**Шифрование** – кодирование с целью защиты информации. **Шифр** – код, известный ограниченному кругу лиц. Методы шифрования изучает **криптография**.