Сжатие с учётом контекста. Словарные методы, где словарём является несжатый текст — семейство LZ77

Александра Игоревна Кононова

МИЭТ

7 октября 2022 г. — актуальную версию можно найти на `https://gitlab.com/illinc/otik`

Источник с памятью

Вероятность порождения элемента определяется предысторией источника $\rightarrow$ при сжатии необходимо учитывать контекст символа.

источник Маркова (N-го порядка) — состояние на i-м шаге зависит от состояний на N предыдущих шагах: $i-1, i-2, \ldots, i-N$ $\rightarrow$ сжимаются не отдельные символы, а устойчивые сочетания — слова: коды Зива—Лемпеля (LZ77, LZ78);

аналоговый сигнал — источник количественных данных Маркова 1-го порядка $\rightarrow$ кодирование длин повторений (Run Length Encoding, RLE).

При расчёте количества информации источника Маркова N-го порядка вероятности зависят от предыдущих N символов (непостоянны).

Размер байта – 4 бита

Для компактной иллюстрации ограничений алгоритмов примем, что для устройства «доска» байт (символ кодирования) составляет не 8 бит — октет (как для Intel x86/amd64), и не 6 бит (как для IBM 7030 Stretch), а 4 бита — тетраду, или одну 16-ричную цифру:

$0 = 0000 = 0$	$8 = 1000 = 8$
$1 = 0001 = 1$	$9 = 1001 = 9$
$2 = 0010 = 2$	$A = 1010 = 10$
$3 = 0011 = 3$	$B = 1011 = 11$
$4 = 0100 = 4$	$C = 1100 = 12$
$5 = 0101 = 5$	$D = 1101 = 13$
$6 = 0110 = 6$	$E = 1110 = 14$
$7 = 0111 = 7$	$F = 1111 = 15$

Символьная таблица (аналог ASCII) даётся отдельно в каждом примере, либо отсутствует.

Концепция RLE

Модель источника данных — Маркова первого порядка (аналоговый сигнал): вероятность символа зависит от предыдущего символа.

Run Length Encoding (RLE): вместо кодирования данных кодируются длины участков, на которых данные сохраняют неизменное значение.

$$\mathrm{AAAAAAAABCCCC} \to 8 \times \mathrm{A}, 1 \times \mathrm{B}, 4 \times \mathrm{C}$$

Повторение символа c подряд L раз $(L \times c)$ — цепочку длины L, $L \geqslant 1$ будем записывать как пару (L, c).

Если c и L одного размера — это имеет смысл (приводит к сжатию цепочки) только при $L > 2$ $(L \geqslant 3)$.

Основные варианты кодирования RLE

1. Наивный — любая цепочка $L \times c$ для любого L ($L \geqslant 1$) заменяется парой (L, c): $\forall L: \ L \times c \to (L, c)$
2. Сжатые и несжатые цепочки (флаг-бит) — парой (L, c) заменяются только длинные цепочки $L \geqslant 3$, короткие цепочки из одного и двух символов объединяются в более длинные несжатые цепочки, каждая из которых также предваряется длиной L;
   для различения сжатых и несжатых цепочек выделяется один бит поля L:
   $$\begin{cases} L \times c, L \geqslant 3 & \to & (1 + L, c), \\ c_1 \dots c_L, L \geqslant 1 & \to & (0 + L, c_1 \dots c_L). \end{cases}$$
3. Префикс сжатой цепочки в несжатом тексте — текст записывается как есть, длинные цепочки заменяются **тройками** (p, L, c), где $p \in A$; сжатие — только при $L \geqslant 4$:
   $$\begin{cases} L \times c, L \geqslant 4 & \to & (p, L, c), \\ c_1 \dots c_L, L \geqslant 1 & \to & c_1 \dots c_L. \\ L \times p, L \geqslant 1 & \to & (p, L, p). \end{cases}$$
   Префикс p выбирается как самый редкий символ текста.

Сравнение (без коррекции длины) I

Исходные сообщения:

- AAAAAAAAAAAAAAAABB (18) — хороший случай ($16 \times$ A и $2 \times$ B)
- AABCDDEF0012 (12) — нормальный случай
- 0123456789AAAAAAAABCDEF (23) — нормальный случай ($8 \times$ A)
- 0123456789ABCDEF (16) — наихудший случай

1. Наивный вариант:

   AAAAAAAAAAAAAAAABB $\rightarrow$ $\begin{bmatrix} \text{FA1A2B} & (6) \\ \text{AFA1B2} & (6) \end{bmatrix}$ для наивного RLE допустимы оба порядка полей — (L, c) или (c, L); далее используется (L, c) для единообразия

   AABCDDEF0012 $\rightarrow$ 2A1B1C2D1E1F2011112 (18)

   0123456789AAAAAAAABCDEF $\rightarrow$ 10111213141516171819 8A1B1C1D1E1F (32)

   0123456789ABCDEF $\rightarrow$ 101112131415161718191A1B1C1D1E1F (32)

   Увеличение объёма в наихудшем случае — в 2 раза.

   Длинные цепочки встречаются редко, поэтому поле L обычно равно по длине символу c (однобайтовое)

Сравнение (без коррекции длины) II

2. Флаг-бит: $1 \leqslant L \leqslant 7$ (3 бита), $1 + 7 \sim 1111 = \texttt{F}$, $1 + 2 \sim 1010 = \texttt{A}$

$\texttt{AAAAAAAAAAAAAAAABB} \to \begin{bmatrix} \texttt{AAAAAAA,}\ \texttt{AAAAAAA,}\ \texttt{AABB} \to \texttt{FAFA4AABB}\ (9) \\ \texttt{AAAAAAA,}\ \texttt{AAAAAAA,}\ \texttt{AA,}\ \texttt{BB} \to \texttt{FAFAAAAB}\ (8) \end{bmatrix}$

но $\texttt{AABCDDEF0012} \to \begin{bmatrix} \texttt{AABCDDE,}\ \texttt{F0012} \to \texttt{7AABCDDE5F0012}\ (14) \\ \texttt{AA,}\ \texttt{BC,}\ \texttt{DD,}\ \texttt{EF,}\ \texttt{00,}\ \texttt{12} \to \texttt{AA2BCAD2EFA0212}\ (15) \end{bmatrix}$

$\texttt{0123456789AAAAAAAABCDEF} \to \texttt{0123456, 789, AAAAAAA, ABCDEF} \to$
$\to \texttt{701234563789FA6ABCDEF}\ (21)$

$\texttt{0123456789ABCDEF} \to \texttt{0123456, 789ABCD, EF} \to \texttt{70123456789ABCD2EF}\ (19)$

здесь порядок полей (L, c) важен: иначе не декодируются несжатые цепочки

Увеличение объёма в наихудшем случае — для байта-тетрады на $\frac{1}{7}$ объёма файла (к каждым семи символам добавляется восьмой — флаг-бит + L), для байта-октета — на $\frac{1}{127}$ (менее процента).

Сравнение (без коррекции длины) III

3. Префикс в несжатом тексте, $p = \texttt{E}$: $\texttt{AAAAAAAAAAAAAAAABB} \to \texttt{EFAABB}$ (6)

Два варианта кодирования одиночного символа p: $1 \times p \to \begin{bmatrix}(p,1,p)\\(p,0)\end{bmatrix}$

$\texttt{0123456789AAAAAAAABCDEF} \to \begin{bmatrix}\texttt{0123456789E8ABCDE1EF}\ (20)\\\texttt{0123456789E8ABCDE0F}\ (19)\quad L \neq 0\end{bmatrix}$

но только один — для двойного p: $\ldots\texttt{DEEF} \to \ldots\texttt{DE2EF}$ (а не $\ldots\texttt{DE0E0F}$).
Если $\texttt{E1E}$, то допустимо и $(p, L, c) \sim \texttt{E1E/E8A}$, и $(p, c, L) \sim \texttt{EE1/EA8}$.
Но если мы хотим $\texttt{E0}$, то только (p, L, c).

$\texttt{0123456789ABCDEF} \to \begin{bmatrix}\texttt{0123456789ABCDE1EF}\ (18)\\\texttt{0123456789ABCDE0F}\ (17)\end{bmatrix}$

Увеличение объёма в наихудшем случае (все символы одиночные) при $(p, 0)$ — на $\nu(p)$ от объёма файла, где $\nu(p)$ — частота p: для байта-тетрады $\nu(p) \leqslant \frac{1}{16}$, для байта-октета $\nu(p) \leqslant \frac{1}{256} < 0{,}4\%$ (p — самый редкий символ).

Сравнение (с коррекцией длины)

Максимальная длина L определяется разрядностью поля (обычно байта), минимальная — алгоритмом $\Rightarrow$ вместо L записывается $\widetilde{L} = L - \min L$
Цепочки с коррекцией L длиннее $\Rightarrow$ их в файле меньше $\Rightarrow$ файл короче.

1. Наивный: $\forall L: \; L \times c \to (L-1, c)$
   AAAAAAAAAAAAAAAABB $\to 16 \times$ A, $2 \times$ B $\to$ FA1B (4)

2. Флаг-бит: $\begin{cases} L \times c, L \geqslant 3 & \to & \Big(1 + (L-3), c\Big), \\ c_1 \ldots c_L, L \geqslant 1 & \to & \Big(0 + (L-1), c_1 \ldots c_L\Big). \end{cases}$
   AAAAAAAAAAAAAAAABB $\to 10 \times$ A, $6 \times$ A, BB $\to$ FABA1BB (7)
   0123456789AAAAAAAABCDEF $\to$ 01234567, 89, AAAAAAAA, BCDEF $\to$
   $\to$701234567189DA4BCDEF (20)
   0123456789ABCDEF $\to$ 01234567, 89ABCDEF $\to$ 7012345677 89ABCDEF (18)

3. Префикс: $\begin{cases} L \times c, c \neq p, L \geqslant 4 & \to & (p, \; L-1, \; c) \\ L \times p, L \geqslant 2 & \to & (p, \; L-1, \; p), \quad L-1 \neq 0 \\ 1 \times p & \to & (p, 0), \end{cases}$
   AAAAAAAAAAAAAAAABB $\to 16 \times$ A, BB $\to$ EFABB (5)
   0123456789AAAAAAAABCDEF $\to$ 0123456789E7ABCDE0F (19)
   0123456789ABCDEF $\to$ 0123456789ABCDE0F (17)

Код Зива–Лемпеля, LZ77/LZ1

1977 г., Якоб Зив (Jacob Ziv) и Абрахам Лемпель (Abraham Lempel)
J. Ziv and A. Lempel, «A universal algorithm for sequential data compression», in IEEE Transactions on Information Theory, vol. 23, no. 3, pp. 337-343, May 1977, doi: 10.1109/TIT.1977.1055714.
— идея замены слова ссылкой (S, L) и концепт $(S, L, c)/(0, 0, c)$:

если цепочка символов (не обязательно одинаковых) встречается более одного раза, то каждое **следующее вхождение** цепочки (слова) **заменяется ссылкой на предыдущее;** ссылка состоит из относительного смещения $S \geqslant 1$ и длины L цепочки **в несжатом тексте:** (S, L)

Поиск предыдущего вхождения — основная сложность алгоритмов семейства LZ77 и неоднозначность кодирования.
Алгоритм LZSS (реализация LZ77) — дерево для ускорения поиска.

Скользящее окно: область перед текущей позицией кодирования, в которой можем искать и адресовать $(w \leqslant \max(S))$ ссылки
Если S записывается N битами, то максимальный размер окна $w = \max(S) = 2^N - 1$.

Алгоритмы семейства LZ77

Алгоритм = идея + отделение ссылок (S, L) от несжатых символов + + кодирование ссылок + алгоритм поиска оригинала.

1. Идея: замена повторного вхождения слова длины L на ссылку (S, L).
2. Отделение ссылок (S, L) от несжатых символов c:
   - строгое чередование — концепт Зива и Лемпеля;
   - однобитный префикс как для ссылки, так и для несжатого символа — флаг-бит, флаг-байт (группировка флаг-битов);
   - односимвольный (однобайтовый) префикс для ссылки + экранирование символа-префикса в несжатом тексте.
3. Кодирование ссылок (S, L):
   - размер полей S и L (может быть не целым байтом, если: а) (S, L) — целое число байт; или б) для чередования и флаг-бита, если входной поток — битовый, а не символьный;
   - порядок полей S и L;
   - коррекция (смещение) S и L: если из-за особенностей алгоритма $L \geqslant k$.

Alternative vexillum codicis inf. interpretatio (AVCII): `␣,абвдеиклмнорты`
`0123456789ABCDEF`

`01234567890123456789012345678901234567890123456789012345678901234567890123`
`там␣корабли␣лавировали,␣лавировали,␣да␣не␣вылавировали`
$\begin{cases} S_1 = 23 - 11 = 12 = \texttt{C}, \\ L_1 = 13 = \texttt{D} \end{cases}$ — пробел №23 принадлежит обеим цепочкам

`01234567890123456789012345678901234567890123456789012345678901234567890123`
`там␣корабли␣лавировали,␣лавировали,␣да␣не␣вылавировали`
$\begin{cases} S_2 = 44 - 24 = 20 > 15 = \texttt{F}, \\ L_2 = 10 = \texttt{A} \end{cases}$

Простейший вариант кодирования — S и L по одному байту, тогда $1 \leqslant S \leqslant 15$ и $1 \leqslant L \leqslant 15$; максимальное окно $w = \max S = 15$.

При программной реализации (поиск только в окне) будет найдено только одно совпадение: (S_1, L_1). Возможен случай, когда адресуемых повторов вообще нет, даже односимвольных — как увеличить $\max S$?

`там␣корабли␣лавировали,`(S_1, L_1)`да␣не␣вылавировали`
$(S_1, L_1) = \texttt{CD} = \texttt{op}$ — как отличить ссылку от текста?

Основные варианты кодирования LZ77

1. Концепт Зива— Лемпеля — ссылки чередуются с несжатыми символами: цепочка, задаваемая ссылкой (S, L), за которой следует c — тройка (S, L, c); несжатый символ c — тройка $(0, 0, c)$.
   При необходимости текст дополняется (обычно нулями)
2. LZSS (Сторер, Сжимански): флаг-бит + символ c/ссылка (S, L).
3. Флаг-байт: символы c и ссылки (S, L) группируются по 8 (по числу бит в байте-октете) штук, каждая группа предваряется байтом, где каждый бит показывает тип соответствующего объекта:
   $\begin{cases} 0, & \text{несжатый символ — байт } c \\ 1, & \text{ссылка — два байта } (S, L), 0 < S \leqslant w, L \geqslant 3 \end{cases}$
   для байта-тетрады группируется, соответственно, по 4 объекта
   идея использована в LZJB (флаг-байт, поле S 10 бит, L 6 бит; коррекция $L + 3$).
4. Префикс p как маркер ссылки: $\begin{cases} \text{ссылка } (p, S, L): \ 0 < S \leqslant w, L \geqslant 4, \\ \text{символ } p \to (p, 0) \end{cases}$

␣,абвдеиклмнорты
0123456789ABCDEF $|S| = |L| = |c|$, окно $w = 15$ символов (макс.)

0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF012345
012345678901234567890123456789012345678901234567890123
там␣кораблик␣лавировали,␣лавировали,␣да␣не␣вылавировали (54)

1. Концепт: 00т00а00м00␣00к00о00р61б00л00и81л61в51рС1вС1лВ1,
CDдС1␣00н00е61в00ыС1а41и00р00о81а81и (84)

   Порядок полей допустим любой — и (S, L), и (L, S) — здесь (S, L).

2. Флаг-байт: 0там␣0кора0бли␣0лави0рова8ли,CD0да␣н0е␣вы0лави
0рова0ли00 (55) Допустимы и (S, L), и (L, S) — в статье SS (S, L).

3. Префикс $p =$ F $=$ ы: там␣корабли␣лавировали,ыCDда␣не␣вы0лавировали
(45)

   Префикс p — в начале. Разный порядок полей S и L приводит к разным способам экранирования символа p. Здесь (p, S, L).

Увеличение разрядности S

1. Записывать S двумя байтами. Для байта-тетрады $S \leqslant 2^8 - 1 = 255$; для байта-октета $S \leqslant 2^{16} - 1 = 65\,535$, окно w обычно дополнительно ограничивается ($w << \max S$) для ускорения поиска.

   Префикс: ссылка занимает 4 символа, тогда $L \geqslant 5$:

   - $\begin{cases} \text{ссылка } (S, L) \to (p, \ \lo S, \ \hi S, \ L), & \text{возможно } \lo S = 0 \text{ при } S \neq 0 \\ \text{символ } p \to (p, 0, 0), & p \text{ экранируется } S = 0 \end{cases}$
   - $\begin{cases} \text{ссылка } (S, L) \to (p, \ L, \ \lo S, \ \hi S), & \text{невозможно } L = 0 \\ \text{символ } p \to (p, 0), & p \text{ экранируется } L = 0 \end{cases}$

2. Добавить к S бит за счёт поля L (пара (S, L) — ровно 2 символа).

   Префикс: ссылка занимает 3 символа, $L \geqslant 4$:

   - $\begin{cases} \text{ссылка } (S, L) \to (p, \ \lo S, \ \hi S \cup L), & \text{возможно } \lo S = 0 \text{ при } S \neq 0 \\ \text{символ } p \to (p, 0, 0), \end{cases}$
   - $\begin{cases} \text{ссылка } (S, L) \to \boldsymbol{(p, \ L \cup \hi S, \ \lo S)}, & \text{при } L \neq 0 \text{ всегда } L \cup \hi S \neq 0 \\ \text{символ } p \to (p, 0), \end{cases}$

Увеличиваем S за счёт L на 1 бит: $\begin{cases} |S| = |c| + 1 = 5 \text{ бит:} & 0 \leqslant S \leqslant 31 \\ |L| = |c| - 1 = 3 \text{ бита:} & 0 \leqslant L \leqslant 7 \end{cases}$ для октета: $\begin{cases} 0 \leqslant S \leqslant 511 \\ 0 \leqslant L \leqslant 127 \end{cases}$

$\begin{cases} \text{ссылка } k = 3 \text{ символа } (p, \quad L \cup \operatorname{hi} S, \quad \operatorname{lo} S), \\ p \to (p, 0), \quad \text{для экранирования префикса необходимо } L \cup \operatorname{hi} S \neq 0 \end{cases}$

слово имеет смысл записывать ссылкой только при $L \geqslant 4$ ($L \neq 0 \to L \cup \operatorname{hi} S \neq 0$);
в итоге $4 \leqslant L \leqslant 7$ (меньше невыгодно, больше не помещается в три бита).
для октета $4 \leqslant L \leqslant 127$.

префикс $p = \texttt{F} =$ ы, окно $w = 31$:

там␣корабли␣лавировали,␣лавировали,␣да␣не␣вылавировали

$$\begin{cases} S_1 = 12 & \sim & 0\ 1100 \\ L_1^1 = 7 & \sim & 111 \end{cases} \to 0111\ 1100$$

там␣корабли␣лавировали,␣лавировали,␣да␣не␣вылавировали

$$\begin{cases} S_1 = 12 & \sim & 0\ 1100 \\ L_1^2 = 6 & \sim & 110 \end{cases} \to 0110\ 1100$$

там␣корабли␣лавировали,␣лавировали,␣да␣не␣вылавировали

$$\begin{cases} S_2 = 20 & \sim & 1\ 0100 \\ L_2 = 7 & \sim & 111 \end{cases} \to 1111\ 0100$$

там␣корабли␣лавировали,ы7Сы6Си,␣да␣не␣вы0ыF4али (47)

Для октета можно увеличить S на два бита за счёт L ($|S| = 10$ бит, $|L| = 6$ бит): $\begin{cases} 0 \leqslant S \leqslant 1023 \\ 0 \leqslant L \leqslant 63 \end{cases}$

Коррекция S и L (k — длина ссылки)

$S \geqslant 1$, $L \geqslant k+1 = 4$. Коррекция: записываем вместо S значение $\widetilde{S} = S - 1$, вместо L значение $\widetilde{L} = L - 3$ (не $L - 4$: для экранирования p нужно $\widetilde{L} \neq 0$)

там␣корабли␣лавировали,␣лавировали,␣да␣не␣вылавировали

$$\begin{cases} S_1 = 12 & \to & \widetilde{S_1} = 11 & \sim & 0\ 1011 \\ L_1 = 10 & \to & \widetilde{L_1} = 7 & \sim & 111 \end{cases} \to 0111\ 1011$$

01234567890123456789012345678901234567890123456789012345678901234567890123
там␣корабли␣лавировали,␣лавировали,␣да␣не␣вылавировали

$$\begin{cases} S_2 = 44 - 12 = 32 & \to & \widetilde{S_2} = 31 & \sim & 1\ 1111 \\ L_2 = 10 & \to & \widetilde{L_2} = 7 & \sim & 111 \end{cases} \to 1111\ 1111$$

там␣корабли␣лавировали,ы7Ви,␣да␣не␣вы0ыFF (41)

Для концепта $(S, L, c)/(0, 0, c)$ коррекция ни S, ни L невозможна.

LZJB: байт-октет, $|S| = 10$ и $|L| = 6$ бит, флаг-байт ($k = 2$), коррекция $\widetilde{L} = L - 3$:

$$\begin{cases} 0 \leqslant S \leqslant 1023 \\ 0 \leqslant \widetilde{L} \leqslant 63 \end{cases} \Rightarrow \begin{cases} 0 \leqslant S \leqslant 1023 \\ 3 \leqslant L \leqslant 66 \end{cases}$$ S в LZJB не корректируется

Сравнение RLE и LZ77

Как RLE, так и LZ77 не требуют отдельного словаря.

1. Повторение одинаковых символов: Префикс $p = \texttt{E}$, без коррекции

   RLE: $AAAAAAAA \sim 8 \times A$

   0123456789AAAAAAAABCDEF → 0123456789E8ABCDE0F $23 \to 19$

   LZ77: $\overbrace{AAAAAAAA}^{} \sim A + (S = 1, L = 7)$ Порядок (p, L, S)

   0123456789AAAAAAAABCDEF → 0123456789AE71BCDE0F $23 \to 20$

2. Повторение подстрок общего вида:

   RLE: $28 \to 24$

   01234560123478 9AAAAAAAABCDEF → 01234560123478 9E8ABCDE0F

   LZ77: $28 \to 23$

   0123456012347 89AAAAAAAABCDEF → 0123456E57789AE71BCDE0F

Для большинства файлов LZ77 эффективнее RLE.

Задача №1

Закодируйте различными реализациями RLE/LZ77 сообщение

$C =$

FFF0 0000 0123 4567 89AB CDEF FFF0 FFF0 0000 0000 0000 0000
1122 3344 5566 7788 99AA BBCC DDEE FFF0

исходная длина C в символах $n = 5 \cdot 16 = 80$ символов

1. в байте 4 бита (тетрада);
2. символ кодирования — 4-битный байт (0 – F); сгруппированы по 4 и 16 штук для удобства.

Задача №2

1. Рассчитать количество информации в сообщении $C = \text{«}\underbrace{\text{ля-ля-ля-...-ля}}_{\text{80 раз}}\text{»}$ (кодировка koi8-r)
2. Закодировать сообщение C алгоритмом LZ77

Спасибо за внимание!

МИЭТ
www.miet.ru

Александра Игоревна Кононова
illinc@mail.ru
gitlab.com/illinc/raspisanie

Информация и модели источника

Заменяя вероятности символов на их оценки по модели X, получаем **оценку** количества информации:

1. Без памяти, $A_1 = \{\text{л}, \text{я}, \text{-}\}$:
   $I_1 = 2 \cdot 80 \cdot \left(-\log_2 \left(\frac{80}{239}\right)\right) + 79 \cdot \left(-\log_2 \left(\frac{79}{239}\right)\right) = 254{,}2 \text{ бит} = 31{,}8 \text{ байт}$
2. Без памяти, $A_1 = \{\text{ля-}, \text{ля}\}$:
   $I_2 = 79 \cdot \left(-\log_2 \left(\frac{79}{80}\right)\right) + 1 \cdot \left(-\log_2 \left(\frac{1}{80}\right)\right) = 7{,}6 \text{ бит} = 0{,}9 \text{ байт}$
3. Маркова, $A_1 = \text{koi8-r}, N = 3$ со следующими оценками вероятностей:
   $p(c_1 = \text{л}) = p(c_2 = \text{я}) = p(c_3 = \text{-}) = \frac{1}{256}$, $p(\text{-}|\text{-ля}) = \frac{79}{80}$,
   $p(\text{л}|\text{ля-}) = p(\text{я}|\text{я-л}) = 1$, $p(eof|\text{-ля}) = \frac{1}{80}$:

$$I_3 = 3 \cdot \log_2 256 + 79 \cdot \left(-\log_2 \left(\frac{79}{80}\right)\right) + 1 \cdot \left(-\log_2 \left(\frac{1}{80}\right)\right) = 31{,}6 \text{ бит} = 3{,}9 \text{ байт}$$