

## Лабораторная работа № 1

### Кодирование целых чисел

Ставится задача кодирования последовательности целых неотрицательных чисел двоичными разделимыми кодами. Максимальное значение целого числа не задается (хотя нам достаточно будет рассмотреть только числа, укладывающиеся в тип `int`).

Существует и можно еще придумать множество схем построения таких кодов. Заранее ни одной из них нельзя отдать предпочтение. В конкретной предметной области можно экспериментально (или с теоретическим обоснованием) выбрать наилучшую схему, дающую максимальное сжатие.

Рассмотрим следующий подход, порождающий цепочку кодов  $\varphi_0$ ,  $\varphi_1$ ,  $\varphi_2$  и т.д. Обозначим через  $\text{bin}(x)$  двоичную запись числа  $x$ , а через  $\text{bint}(x)$  – усеченную двоичную запись без ведущей единицы.

$x$	$\text{bin}(x)$	$\text{bint}(x)$
0	0	
1	1	
2	10	0
3	11	1
4	100	00
5	101	01
6	110	10
7	111	11
8	1000	000
...		

Очевидно, что ни  $\text{bin}(x)$ , ни  $\text{bint}(x)$  не могут кодировать целые числа в последовательности, т.к. код получается неразделимым.

Определим код  $\varphi_0(x)$  следующим образом: записываем  $x$  нулей, за которыми следует единица.

$x$	$\varphi_0(x)$
0	1
1	01
2	001
3	0001
4	00001

Данный код разделим (единица отделяет кодовые слова друг от друга). Длина кода для числа  $x$  равна  $x+1$ .

Определим теперь более экономичный код  $\varphi_1(x)$ . Идея состоит в том, чтобы использовать двоичную запись числа  $x$ , предварительно указав ее длину с помощью разделимого кода  $\varphi_0(x)$ . Так как двоичная запись (кроме записи для нуля) всегда начинается с единицы, то целесообразно использовать  $\text{bint}(x)$ , а ведущую единицу декодер может добавить автоматически. Правило построения кода можно формально записать следующим образом:  $\varphi_1(x) = \varphi_0(|\text{bin}(x)|) \circ \text{bint}(x)$ , где значком  $\circ$  показана конкатенация (сцепление) кодовых слов. Для нуля делается исключение: он кодируется одной единицей.

$x$	$\varphi_1(x)$
0	1
1	01
2	001 0
3	001 1
4	0001 00
5	0001 01
6	0001 10
7	0001 11
8	00001 000

В примере для наглядности части кода отделены пробелом, на самом деле никаких пробелов вставлять не нужно. Правило декодирования простое: считаем количество нулей до первой единицы (обозначим его через  $k$ ); если  $k = 0$ , то получаем  $x = 0$ ; если  $k > 0$ , то берем  $k$  бит, включая первую единицу, и они дают нам двоичную запись числа  $x$ .

Легко видеть, что для больших  $x$  длина кода  $|\varphi_1(x)| \approx 2 \log(x)$ , т.е. код  $\varphi_1$  значительно экономнее кода  $\varphi_0$ .

Можно построить еще более экономный код для больших чисел, если кодировать длину двоичного представления не кодом  $\varphi_0$ , а кодом  $\varphi_1$ . Таким образом получаем код  $\varphi_2(x) = \varphi_1(|\text{bin}(x)|) \circ \text{bint}(x)$ , делая опять исключение для нуля.

$x$	$\varphi_2(x)$
0	1
1	01
2	001 0 0
3	001 0 1
4	001 1 00
5	001 1 01
6	001 1 10
7	001 1 11
8	0001 00 000

Для больших чисел  $x$  справедливо приближенное равенство  $|\varphi_2(x)| \approx 2 \log \log(x) + \log(x)$ .

Представленную схему можно продолжать до бесконечности, строя все более экономные коды для больших чисел.

**Задание на ЛР.** Запрограммировать коды  $\varphi_0$ ,  $\varphi_1$ ,  $\varphi_2$  с соответствующими декодерами. Проверить корректность их работы на тестовых последовательностях целых чисел.

Аудиторное время на выполнение и защиту работы – 4 часа.

Дополнительную информацию по кодам целых чисел легко найти в Интернете, см. гамма и дельта коды Элайеса (Elias gamma (delta) coding), код Левенштейна, коды Голомба (Golomb coding).