

## Формат закодированного файла

Закодированный файл представляет собой бинарный файл, содержащий закодированный текст. Файл начинается с метаданных, включающих информацию о структуре дерева кодирования; количество различных символов в тексте, для которых создана кодовая строка; размер исходного файла в байтах.

### Метаданные

#### 1. Размер исходного файла в байтах:

- В начале сохраняется размер исходного файла.
- Как сохраняется: данное число переводится в его битовое представление. Полученная строка делится на подстроки длиной 8 бит. В файл сохраняем сначала количество подстрок, на которые разделилась данная строка; потом каждая подстрока переводится в его численное представление (например,  $110_2 = 6_{10}$ ) и это число записывается в файл (в файле это число предстанет в виде какого-то символа из таблицы кодировок), и перед последней подстрокой сохраняем количество нулей в начале этой последней подстроки до значащих цифр, чтобы можно было эти нули восстановить при декодировании (например, для строки 0001010 сохраним: 3 – количество незначащих нулей и  $1010_2 = 10_{10}$ ).

#### 2. Количество различных символов в тексте:

- Нужно для того чтобы считать все байты относящиеся к дереву Хаффмена и не задевать байты выделенные под закодированный текст.
- Сохранение аналогично сохранению количества всех элементов в тексте.

#### 3. Структура Дерева:

- Затем сохраняется структура дерева Хаффмана, включающая символы и их коды в представлении дерева.
- Как сохраняется: символ из текста переводится в тип `int` (получаем его кодировку из таблицы кодировок)  
Дальше сохраняем байты для кода Хаффмена этого символа:  
Эта строка из 0 и 1 делится на подстроки длиной 8 бит. В файл сохраняем сначала количество подстрок, на которые разделилась данная строка; потом каждая подстрока переводится в его численное представление (например,  $110_2 = 6_{10}$ ) и это число записывается в файл (в файле это число предстанет в виде какого-то символа из таблицы кодировок), и перед последней подстрокой сохраняем количество нулей в начале строки до значащих цифр, чтобы можно было эти нули восстановить (например, для строки 0001010 сохраним: 3 – количество незначащих нулей и  $1010_2 = 10_{10}$ ).  
Данное сохранение кода Хаффмена для символа позволяет сократить количество занимаемых байт файлом: например, если строку 1010 сохранять как строку, то она займёт 4 байта в 8-ми битной кодировке, в то время как описанный выше способ будет использовать всего 3 байта (байт на сохранение количества подстрок, ещё байт на сохранение незначащих нулей и ещё байт уже на само число  $1010_2 = 10_{10}$ )

### Закодированный текст

- После метаданных следует закодированный текст. Он содержит все символы исходного текста, но записанные с помощью особого кода для каждого символа. Соответствие кода символу выполняется на основании дерева Хаффмана.
  - Как сохраняется: закодированная строка делится на подстроки длиной 8 бит. В файл сохраняем сначала количество подстрок, на которые разделилась данная строка; потом каждая подстрока переводится в его численное представление (например,  $110_2 = 6_{10}$ ) и это число записывается в файл (в файле это число предстанет в виде какого-то символа из таблицы кодировок), и перед последней подстрокой сохраняем количество нулей в начале этой последней подстроки до значащих цифр, чтобы можно было эти нули восстановить при декодировании (например, для строки 0001010 сохраним: 3 – количество незначащих нулей и  $1010_2 = 10_{10}$ ).