Описание структуры файла формата .huf (кодирование по методу Хаффмана)

Формат .huf, реализованный в данном проекте, состоит из четырёх основных частей:

1. Заголовок файла
2. Размер исходных данных
3. Словарь частот (частотная таблица)
4. Закодированный поток битов

Ниже приводится подробное документированное описание структуры.

1. Заголовок

Заголовок служит для проверки корректности формата и его версии.

Смещение	**Размер**	**Описание**
0	3 байта	ASCII подпись "HFM"
3	1 байт	Версия формата (в данной реализации — 1)

Пример в hex:

48 46 4D 01 -> "HFM" + версия 1

2. Размер исходных данных

После заголовка следует одно 8-байтовое число (long, big-endian), которое содержит:

Количество байт во входном файле до кодирования.

Это необходимо, чтобы декодировщик точно знал, сколько байт нужно восстановить, даже если последний байт кодированного потока содержит "добивающие" нули.

Пример:

00 00 00 00 00 00 00 0A -> 10 байт

3. Словарь частот (частотная таблица)

Затем следует таблица частот - ключевая структура, определяющая построение дерева Хаффмана.

Структура:

Размер	**Описание**
4 байта (int)	количество различных символов m

Размер	Описание
m записей	каждая содержит: • 1 байт - символ (0–255) • 8 байт - частота появления (long)

Пример:

00 00 00 01 -> m = 1 символ

31 -> символ '1' (ASCII 0x31)

00 00 00 00 00 00 00 0A -> частота 10

4. Закодированный поток битов

После словаря следует основной блок данных - поток битов, полученных при кодировании файла по алгоритму Хаффмана.

Особенности записи:

- Биты упаковываются в байты **со старшего бита к младшему** (big-endian внутри байта).
- Последний байт может быть дополнен нулями.
- Количество значимых байт восстанавливается по полю «Размер исходных данных».

Пример упакованной последовательности:

00 00

(объяснение ниже)

Практический пример: файл test1.txt

Содержимое файла:

1111111111

То есть строка из **10 одинаковых символов** '1'.

Шаг 1 - Частотный анализ

Символ	ASCII	Частота
'1'	0x31	10

Шаг 2 - Построение дерева Хаффмана

Так как присутствует только один символ, он получает минимально возможный код:

'1' -> 0

Шаг 3 - Кодирование данных

Строка из 10 символов '1' превращается в 10 бит:

0000000000

Упаковка в байты:

- первые 8 бит -> 00000000 -> 0x00
- оставшиеся 2 бита -> 00 -> дополнены нулями -> 00000000 -> 0x00

Итого:

00 00

Полный hex-файл test1.huf, собранный по формату

48 46 4D 01 -> "HFM" + версия

00 00 00 00 00 00 00 0A -> длина исходных данных = 10

00 00 00 01 -> m = 1 запись

31 -> символ '1'

00 00 00 00 00 00 00 0A -> частота = 10

00 00 -> битовые данные