# Лабораторная работа № 3

# Проектирование формата файлов. Работа с двоичным представлением информации на ЯВУ

## Задание на лабораторную работу

1. Выбрать сигнатуру (постоянную для команды и уникальную в пределах группы) для маркировки формата, разработанного данной командой.

|  |  |  |  |
| --- | --- | --- | --- |
| **Расширение файла** | **Размер** | **UTF-8 Расшифровка** | **HEX сигнатура** |
| vyatskiysas | 4 байта | VSAS | 56 53 41 53 |

2. Разработать формат файла-архива, хранящего

* Заголовок
* данные, необходимые для декодирования
* закодированные данные.

Заголовок должен содержать сигнатуру, а также иные данные, необходимые для различения формата и декодирования файла-архива.

Выбранные в данной лабораторной работе сигнатуру и формат файла необходимо использовать во всех позднейших программах кодирования/декодирования. Соответственно, заголовок должен включать как постоянную часть, необходимую для различения конкретного подтипа формата, так и элементы, зависящие от подтипа/алгоритма кодирования.

Необходимо предусмотреть в разработанном формате возможность следующих видов кодирования:

— энтропийное сжатие (с учётом частотности);

— сжатие с учётом контекста;

— защита от помех;

причём необходимо учесть, что

— к одному файлу может последовательно применяться несколько видов кодирования;

— один вид кодирования включает несколько методов (так, энтропийное сжатие включает и метод Хаффмана, и арифметическое сжатие); а один и тот же метод кодирования может быть реализован различными способами.

Описать структуру файла в отчёте.

**Структура архива VSAS**

1. **[Заголовок архива] – 16 байт**

|  |  |  |
| --- | --- | --- |
| **Смещение** | **Размер** | **Назначение** |
| 0 | 4 | Сигнатура (VSAS / 56 53 41 53) |
| 4 | 1 | Номер алгоритма энтропийного сжатия (0 – не использовать) |
| 5 | 1 | Номер алгоритма сжатия с учётом контекста (0 – не использовать) |
| 6 | 1 | Номер алгоритма защиты от помех (0 – не использовать) |
| 7 | 1 | Номер алгоритма кодирования данных (0 – не исп, 1 - дефолтный) |
| 8 | 1 | Количество файлов в архиве |
| 9 | 7 | Запасные байты |

1. **[Заголовок содержимого файла i] – 128 байт**

|  |  |  |
| --- | --- | --- |
| **Смещение** | **Размер** | **Назначение** |
| 0 | 4 | Сигнатура файла |
| 4 | 4 | Размер после сжатия – если сжатие не использовалось то 0 |
| 8 | 4 | Размер без сжатия |
| 12 | 1 | Длина имени файла |
| 13 | 4 | Смещение заголовка расшифровки |
| 17 | 4 | Смещение данных файла |
| 21 | 104 | Имя файла |
| 125 | 3 | Запасные байты |

1. **[Заголовок расшифровки i] – содержит длины блоков разделенные символом ‘q’**
2. **[Данные содержимого файла i (закодированные)]**

4. Написать программу кодирования/декодирования, использующую разработанный формат для хранения файла/файлов без сжатия, защиты от помех, шифрования либо иных изменений.

В простейшем случае файл-архив разработанного формата должен хранить один файл (не только текстовый, но и любой бинарный) и содержать

— заголовок;

— оригинальное имя файла;

— несжатые данные файла.

Имя архива и расширение при кодировании могут задаваться произвольно (для единообразия можно использовать, например, расширение .otik); при декодировании тип файла должен определяться по заголовку, а не по расширению.

Функционал программы:

— на вход программы поступает либо файл, содержащий исходный текст, либо файл-архив, содержащий заголовок (разработанная сигнатура файла и, при необходимости, дополнительная информация) и данные. Режим работы (кодирование, декодирование, тестирование[[1]](#footnote-1) архива, просмотр содержимого архива[[2]](#footnote-2)) задаётся с помощью параметров командной строки, с помощью GUI, либо выбирается исходя из анализа заголовка входного файла;

— в режиме кодирования программа в результате обработки исходного файла формирует файл-архив, содержащий заголовок и данные. Если на входе уже файл-архив с соответствующей сигнатурой, выдавать сообщение об ошибке и не выполнять кодирование (при желании можно предусмотреть опцию, включающее такое кодирование, но она должна быть отключаемой);

— в режиме декодирования программа анализирует файл-архив и восстанавливает исходный файл (если формат архива соответствует разработанному ранее, а конкретный вариант формата — заданию). В случае некорректного файла на входе должно выдаваться сообщение об ошибке.

Если сигнатура и формат архива соответствует разработанным данной командой, но архив создан программой из другой лабораторной работы, в режимах распаковки и тестирования необходимо вывести сообщение об этом.

5. Написать программу кодирования/декодирования, использующую разработанный формат для кодирования/декодирования файла в соответствии с вариантом.

При этом исходный файл имеет произвольную бинарную структуру, может содержать произвольные данные и иметь любую длину. Восстановленный из архива файл должен быть полностью идентичен исходному.

6 (необязательное задание, +2 балла). Предусмотреть в программе возможность собрать в один архив несколько файлов и восстановить их с прежними именами, причём эта возможность должна быть учтена при разработке заголовка архива. При этом нет необходимости сжимать папки.

7 (необязательное задание, +2 балла). Предусмотреть в программе возможность сжать не только несколько файлов, но и иерархическую структуру папок.

## Варианты заданий

|  |  |
| --- | --- |
| 1 | Разделение данных на блоки случайной (переменной от 1 до 16 байт) длины, разделяемые символом (байтом) 'q' и порядковым номером блока с 1 (после 255 следует номер 1) |
| 2 | Разделение данных на блоки случайной (переменной от 1 до 16 байт) длины, размер каждого блока указывается в его заголовке |
| 3 | Добавление после каждого байта данных двух нулевых битов (то есть 4 байта исходного файла кодируются 5 байтами) |

## Порядок выполнения работы

1. В таблице заданий выберите свой вариант, определив его по формуле

(№ команды — 1) % (Количество вариантов) + 1 (1)

где % — оператор получения остатка от деления. В частности, если варианта 3, то у студента с номером компьютера 5 будет 2-й вариант, т. к. (5-1)%3+1=2.

2. В произвольной программной оболочке создайте программу/программы в соответствии с заданием.

3. Продемонстрируйте кодирование/декодирование на текстовых и бинарных файлах:

— корректно ли происходит кодирование?

— как выглядит заголовок архива в hexdump, где расположены сигнатура и основные поля?

— выдаётся ли сообщение о попытке повторного кодирования файла-архива?

— возможно ли без дополнительных указаний пользователя корректное восстановление файла, хранимого в архиве без изменений (задание 4) и файла, изменённого по варианту (задание 5)?

— совпадает ли восстановленный файл с исходным?

— выдаётся ли сообщение об ошибке при попытке декодирования файла, не являющегося архивом разработанного формата?

1. В простейшем случае предполагается проверка целостности заголовка файла. [↑](#footnote-ref-1)
2. Достаточно вывести названия файлов, хранящихся в архиве, и их формат. [↑](#footnote-ref-2)