# ДОМАШНЕЕ ЗАДАНИЕ №1

## Глубокий анализ кодирования изображений: от битов до чанков

**Симаранов Александр Андреевич БИВТ-25-17**

Доп. материал в ходе выполнения работы можно найти тут: https://github.com/s1rne/s1rne\_computing\_machines\_homeworks

## Описание

Данная работа посвящена детальному изучению кодирования изображений на низкоуровневом уровне с использованием hex-редактора 010 Editor. Проведен анализ внутренней структуры файлов изображений различных форматов, изучены механизмы сжатия данных и принципы работы алгоритмов. Особое внимание уделено формату PNG с его системой чанков, а также разбору структуры JPEG. Получены практические навыки работы с бинарными данными и понимания принципов кодирования изображений.

## 1. ВВЕДЕНИЕ

Понимание принципов кодирования графической информации на низкоуровневом уровне позволяет лучше понять цифровые изображения.

### Методы исследования:

* Hex-анализ с использованием 010 Editor
* Сравнительный анализ различных форматов изображений
* Изучение алгоритмов сжатия и кодирования
* Практическое исследование структуры файлов

## 2. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ

### 2.1 Принципы кодирования изображений

Цифровое изображение представляет собой двумерную матрицу пикселей, где каждый пиксель содержит информацию о цвете. Процесс кодирования включает три основных этапа:

**1. Дискретизация** - преобразование непрерывного изображения в дискретные элементы (пиксели) **2. Квантование** - установление количества возможных значений цвета (например, 256 уровней для 8-битного изображения) **3. Кодирование** - представление информации в цифровом виде с использованием различных алгоритмов сжатия

### 2.2 Контрольные суммы CRC

**CRC (Cyclic Redundancy Check)** - это алгоритм обнаружения ошибок в данны. Для PNG файлов CRC используется для проверки целостности чанков.

#### Как работает CRC:

1. **Инициализация:** CRC начинается с определенного начального значения
2. **Обработка данных:** Каждый байт данных обрабатывается через специальную функцию
3. **Финальное значение:** На выходе получается 32-битная контрольная сумма
4. **Проверка:** При чтении файла CRC пересчитывается и сравнивается с сохраненным значением

#### Пример расчета CRC:

Данные: "IHDR" + параметры изображения  
CRC полином: x^32 + x^26 + x^23 + x^22 + x^16 + x^12 + x^11 + x^10 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1  
Результат: 4-байтовая контрольная сумма в формате big-endian

### 2.3 Форматы изображений

**PNG (Portable Network Graphics):** - Безпотерьное сжатие с использованием алгоритма DEFLATE - Поддержка прозрачности и различных цветовых моделей - Модульная структура на основе чанков - Оптимален для графики с четкими границами

**JPEG (Joint Photographic Experts Group):** - Потерянное сжатие с использованием дискретного косинусного преобразования (DCT) - Оптимизирован для фотографий и изображений с плавными переходами - Настраиваемое качество сжатия - Структура на основе маркеров

**BMP (Bitmap):** - Простая структура без сжатия или с RLE-сжатием - Прямое хранение пикселей - Большие размеры файлов - Простота реализации

## 3. АНАЛИЗ СТРУКТУРЫ PNG ФАЙЛОВ

### 3.1 Общая структура PNG

PNG файл состоит из следующих частей: 1. **Сигнатура** (8 байт) - идентификация формата 2. **Чанки** - модульные блоки данных 3. **Конечный маркер** - завершение файла

### 3.2 Сигнатура PNG

Каждый PNG файл начинается с одинаковой 8-байтовой сигнатуры:

89 50 4E 47 0D 0A 1A 0A

**Разбор сигнатуры:** 89 (137) - не ASCII символ, проверка на текстовые файлы. 50 4E 47 - “PNG” в ASCII кодировке. 0D 0A - CRLF, проверка на конвертацию окончаний строк. 1A - EOF символ, проверка на вывод в консоль. 0A - LF символ

### 3.3 Система чанков PNG

Каждый чанк имеет следующую структуру:

[Длина: 4 байта] [Тип: 4 байта] [Данные: N байт] [CRC: 4 байта]

**Поля чанка:** **Длина** - количество байтов в поле данных (big-endian) . **Тип** - 4-символьный ASCII код типа чанка. **Данные** - содержимое чанка (может быть пустым). **CRC**

### 3.4 Обязательные чанки PNG

#### IHDR (Image Header)

**Назначение:** Содержит основную информацию об изображении

**Структура данных (13 байт):**

Ширина: 4 байта; Высота: 4 байта  
Глубина цвета: 1 байт (1, 2, 4, 8, 16, …)  
Тип цвета: 1 байт (0=Grayscale, 2=RGB, 3=Palette, 4=Grayscale+Alpha, 6=RGB+Alpha)  
Метод сжатия: 1 байт (0=DEFLATE)  
Метод фильтрации: 1 байт (0=адаптивная)  
Метод чередования: 1 байт (0=нет, 1=Adam7)

#### IDAT (Image Data)

**Назначение:** Содержит сжатые данные изображения

**Особенности:** Может быть несколько IDAT чанков. Данные сжаты алгоритмом DEFLATE. Перед сжатием применяются фильтры для улучшения сжатия. Каждая строка изображения предваряется типом фильтра

#### IEND (Image End)

**Назначение:** Маркер конца файла

**Структура:** Длина = 0, тип = “IEND”, данные отсутствуют

### 3.5 Практический анализ минимального PNG

Рассмотрим структуру PNG файла размером 1×1 пиксель (черный цвет):

#### Hex-дамп файла (57 байт):

00000000: 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  
00000010: 00 00 00 01 00 00 00 01 08 02 00 00 00 90 77 53  
00000020: DE 00 00 00 0C 49 44 41 54 08 99 01 01 00 00 FF  
00000030: FF 00 00 00 02 00 01 E2 21 BC 33 00 00 00 00 49  
00000040: 45 4E 44 AE 42 60 82

#### Детальный разбор:

**1. Сигнатура PNG (байты 0-7):**

89 50 4E 47 0D 0A 1A 0A

Стандартная сигнатура PNG файла.

**2. Чанк IHDR (байты 8-32):**

00 00 00 0D 49 48 44 52 00 00 00 01 00 00 00 01 08 02 00 00 00 90 77 53 DE

Разбор IHDR: Длина: 00 00 00 0D = 13 байт - Тип: 49 48 44 52 = “IHDR”. Ширина: 00 00 00 01 = 1 пиксель. Высота: 00 00 00 01 = 1 пиксель. Глубина: 08 = 8 бит на канал. Тип цвета: 02 = RGB (без альфа-канала). Сжатие: 00 = DEFLATE. Фильтрация: 00 = адаптивная. Чередование: 00 = нет чередования. CRC: 90 77 53 DE

**3. Чанк IDAT (байты 33-44):**

00 00 00 0C 49 44 41 54 08 99 01 01 00 00 FF FF 00 00 00 02 00 01 E2 21 BC 33

Разбор IDAT: Длина: 00 00 00 0C = 12 байт. Тип: 49 44 41 54 = “IDAT”. Сжатые данные: 08 99 01 01 00 00 FF FF 00 00 00 02 00 01. CRC: E2 21 BC 33

**Анализ сжатых данных:** При распаковке DEFLATE получаем 4 байта: 00 00 00 FF. 00 - тип фильтра (None). 00 00 00 - RGB значения черного пикселя (0,0,0)

**4. Чанк IEND (байты 45-56):**

00 00 00 00 49 45 4E 44 AE 42 60 82

* Длина: 00 00 00 00 = 0 байт
* Тип: 49 45 4E 44 = “IEND”
* CRC: AE 42 60 82

## 4. АНАЛИЗ СТРУКТУРЫ JPEG ФАЙЛОВ

### 4.1 Общая структура JPEG

JPEG файл состоит из последовательности сегментов, каждый из которых начинается с маркера:

[Маркер SOI] [Сегменты данных] [Маркер EOI]

### 4.2 Маркеры JPEG

JPEG использует двухбайтовые маркеры, начинающиеся с FF:

**Основные маркеры:** FF D8 - SOI (Start of Image) - начало изображения. FF D9 - EOI (End of Image) - конец изображения. FF DA - SOS (Start of Scan) - начало сканирования данных. FF C0 - SOF0 (Start of Frame) - параметры изображения. FF DB - DQT (Define Quantization Table) - таблицы квантования. FF C4 - DHT (Define Huffman Table) - таблицы Хаффмана

### 4.3 Структура сегментов JPEG

Каждый сегмент (кроме SOI и EOI) имеет формат:

[Маркер: 2 байта] [Длина: 2 байта] [Данные: N байт]

### 4.4 Практический анализ JPEG файла

Рассмотрим структуру JPEG файла размером 10×10 пикселей:

#### Hex-дамп начала файла:

00000000: FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 48  
00000010: 00 48 00 00 FF DB 00 43 00 08 06 06 07 06 05 08  
00000020: 07 07 07 09 09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12  
00000030: 13 0F 14 1D 1A 1F 1E 1D 1A 1C 1C 20 24 2E 27 20  
00000040: 22 2C 23 1C 1C 28 37 29 2C 30 31 34 34 34 1F 27

#### Детальный разбор:

**1. Маркер SOI:**

FF D8

Начало JPEG файла.

**2. Сегмент APP0 (JFIF):**

FF E0 00 10 4A 46 49 46 00 01 01 01 00 48 00 48 00 00

Разбор APP0: - Маркер: FF E0 = APP0 - Длина: 00 10 = 16 байт - Идентификатор: 4A 46 49 46 = “JFIF” - Версия: 00 01 = 1.1 - Единицы плотности: 01 = DPI - X плотность: 00 48 = 72 DPI - Y плотность: 00 48 = 72 DPI - Размер превью: 00 00 = нет превью

**3. Сегмент DQT (таблица квантования):**

FF DB 00 43 00 08 06 06 07 06 05 08 07 07 07 09 09 08 0A...

Разбор DQT: Маркер: FF DB = DQT. Длина: 00 43 = 67 байт. Информация таблицы: 00 = таблица 0, 8 бит точность. Данные таблицы: 64 байта квантованных значений

### 4.5 Алгоритм сжатия JPEG

**Этапы сжатия JPEG:**

1. **Преобразование цветового пространства:** RGB → YCbCr
2. **Дискретно-косинусное преобразование (DCT):** 8×8 блоки
3. **Квантование:** деление на таблицы квантования
4. **Зигзаг-сканирование:** преобразование в одномерный массив
5. **Кодирование Хаффмана:** сжатие данных
6. **Упаковка в битовый поток**

**Особенности JPEG:** - Потерянное сжатие (информация теряется на этапе квантования). Оптимизирован для фотографий с плавными переходами. Настраиваемое качество через таблицы квантования. Неэффективен для изображений с четкими границами

#### DEFLATE (PNG):

* **Принцип:** LZ77 + Хаффман кодирование
* **Тип:** Без потерь
* **Эффективность:** Высокая для простых изображений
* **Особенности:** Применение фильтров перед сжатием

#### DCT (JPEG):

* **Принцип:** Дискретное косинусное преобразование
* **Тип:** С потерями
* **Эффективность:** Высокая для фотографий
* **Особенности:** Настраиваемое качество

#### RLE (BMP):

* **Принцип:** Кодирование повторяющихся значений
* **Тип:** Без потерь
* **Эффективность:** Низкая для сложных изображений

## 5. РАБОТА С HEX-РЕДАКТОРОМ 010 EDITOR

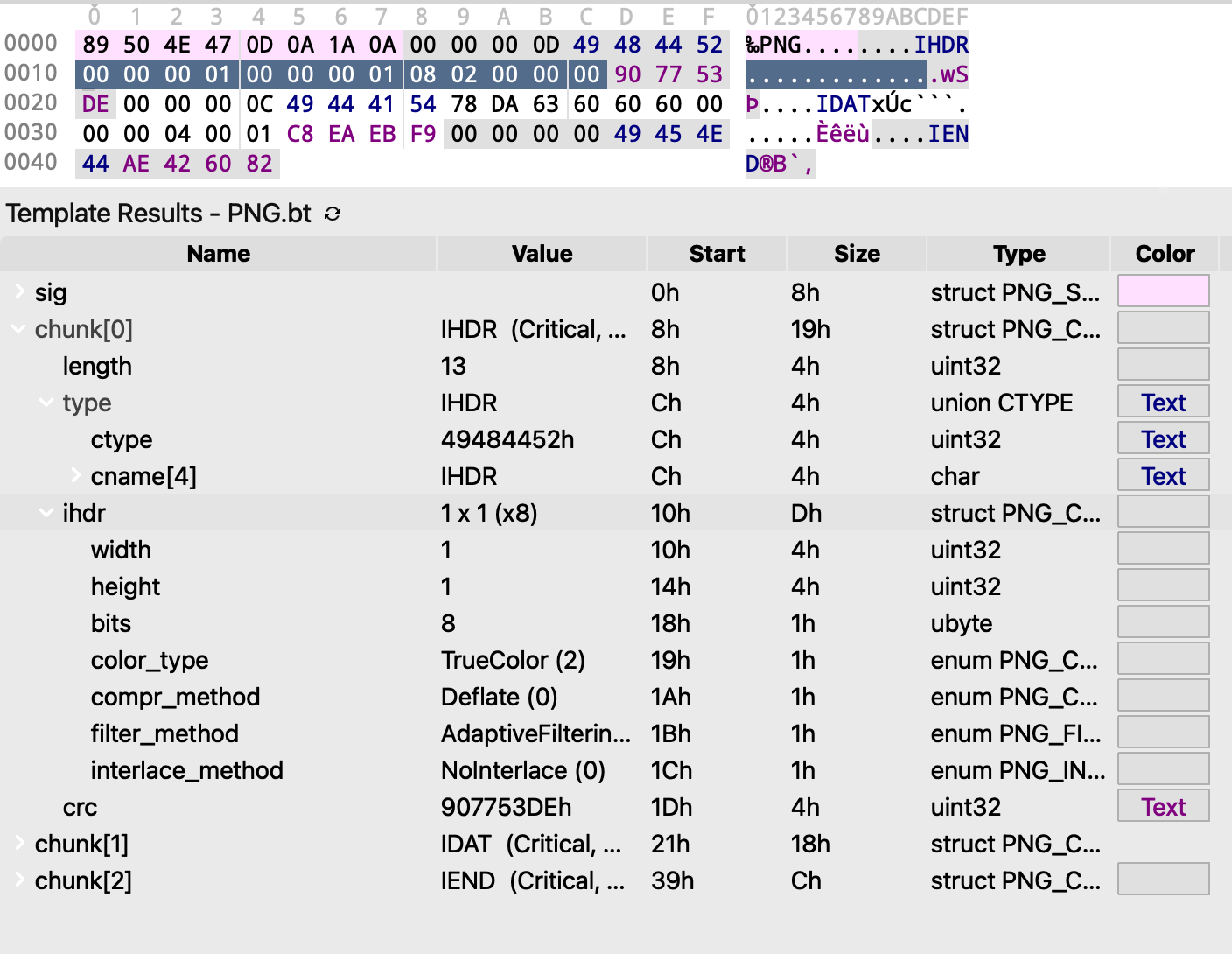
### 5.1 Преимущества 010 Editor

**010 Editor** - Мощный hex-редактор с поддержкой шаблонов для анализа различных форматов файлов.

**Основные возможности:** Визуализация бинарных данных в hex и ASCII. Поддержка шаблонов для различных форматов. Вычисление контрольных сумм и хешей. Поиск и замена данных. Сравнение файлов

### 5.2 Анализ PNG с помощью шаблонов

**PNG в 010 Editor для визуализации структуры файла:**



Полная структура файла. Детальная информация о каждом чанке. Проверка целостности данных. Возможность модификации с сохранением валидности. Расшифровка всех ключевых данных.

### 5.2 Анализ JPEG с помощью шаблонов

**JPEG в 010 Editor для визуализации структуры файла:** Изображение выглядит как текст, снимок экрана, Параллельный, Красочность

Контент, сгенерированный ИИ, может содержать ошибки.

### 6. СРАВНИТЕЛЬНЫЙ АНАЛИЗ ФОРМАТОВ

### Практические результаты преобразований форматов изображений:

При помощи python скрипта (demo.py), создал различного типа и размера изображения и посчитал их размеры.

Файл Размер (байт) Формат

-------------------------------------------------------

test\_minimal\_black.png 69 PNG

test\_gradient\_10x10.png 106 PNG

test\_gradient\_10x10.jpg 708 JPG

test\_gradient\_10x10.bmp 374 BMP

test\_colorful\_50x50.png 121 PNG

test\_colorful\_50x50.jpg 1098 JPG

test\_colorful\_50x50.bmp 7654 BMP

test\_geometric\_100x100.png 682 PNG

test\_geometric\_100x100.jpg 2555 JPG

test\_geometric\_100x100.bmp 30054 BMP

**Коэффициенты сжатия относительно BMP:** PNG: 1.09 - 8.41. JPEG: 0.49 - 13.23

## 7. ЗАКЛЮЧЕНИЕ

Проведенное исследование показало важность понимания низкоуровневых аспектов кодирования изображений. Использование hex-редактора 010 Editor позволило получить глубокое понимание структуры графических файлов.

### Основные выводы:

* **PNG оптимален для графики** - модульная структура и безпотерьное сжатие
* **JPEG эффективен для больших фотографий** - высокая степень сжатия с настраиваемым качеством
* **CRC критичен для целостности** - обеспечивает надежность хранения данных
* **Hex-анализ необходим для понимания** - позволяет глубоко изучить структуру файлов

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. <https://habr.com/ru/articles/454944/>
2. <https://habr.com/ru/companies/ruvds/articles/787302/>
3. 010 Editor Documentation
4. DEFLATE Compressed Data Format Specification