# Лабораторная работа 2 Декодирование PNG

## **Цель работы**

Изучить особенности работы с двоичными файлами, структурами, директивами препроцессора и сторонними библиотеками в C. Закрепить навык чтения спецификаций.

## **Стандарт языка**

С11 и новее.

## **Описание**

Необходимо реализовать программу, которая декодирует изображение PNG в PNM.

***должны поддерживаться:*** серые и цветные изображения (Color Type 0 и 2), 8 бит на канал.

***не нужно поддерживать:*** работу с прозрачностью, interlacing и цветовую коррекцию (гамму и пр.).

Необходимо использовать одну библиотеку из следующих: libdeflate, zlib или isa-l (на ваше усмотрение).

**Критерий “Все библиотеки”:** должно поддерживаться использование всех трёх сторонних библиотек: zlib, libdeflate и isa-l.

Библиотеки должны использоваться только для разжатия deflate-потока из изображения.

## **Использование сторонних библиотек**

Кроме zlib, libdeflate и isa-l запрещено использовать какие-либо другие сторонние библиотеки (стандартную библиотеку С использовать можно и нужно).

| **Библиотека** | **Версия** | **Подключение** |
| --- | --- | --- |
| zlib | 1.2.12 | #include <zlib.h> |
| libdeflate | 1.10 | #include <libdeflate.h> |
| isa-l | 2.30 | #include <include/igzip\_lib.h> |

Для вашего репозитория (на github) исходный файл, содержащий функцию main, должен лежать в корне репозитория. Пример:

| /c\_2-<github\_nickname>  ├── main.c /\* your src file\*/  ├── my\_header.h /\* [optional] your header file \*/  ├── .gitignore /\* [optional] \*/  ├── .clang-format  ├── … |
| --- |

**Внимание!** В репозитории не должно быть любых файлов zlib, libdeflate и isa-l. Вы можете их хранить рядом локально, но в этом случае они должны быть занесены .gitignore.

Выбор библиотеки должен определяться макросом, который указывается ключом компилятора, создающим define. Гарантируется, что при компиляции будет указан ровно один из трёх макросов.

| **Библиотека** | zlib | libdeflate | isa-l |
| --- | --- | --- | --- |
| **define** | ZLIB | LIBDEFLATE | ISAL |

В случае, если при компиляции был указан макрос, определяющий подключение библиотеки, которую ваша программа не поддерживает, то необходимо командой препроцессора #error() выводить сообщение о том, что данная библиотека не поддерживается. Подробнее: <https://en.cppreference.com/w/c/preprocessor/error>

Примеры:

| #if defined(ZLIB)  #error … | #ifdef ZLIB  #error … |
| --- | --- |

## **Формат аргументов командной строки**

Аргументы программе передаются через командную строку:

**c2 <имя\_входного\_PNG\_файла> <имя\_выходного\_PNM\_файла>**

## **Входной файл**

PNG-изображение. Не гарантируется корректность данных внутри файла (может быть любой файл, в том числе не PNG или файл с ошибками).

## **Выходной файл**

Формат выходных изображений: PNM (P5 или P6). Во всех PNM файлах (pgm, ppm) комментарии отсутствуют.

Если изображения во входном файле было в оттенках серого (Color Type 0), то выходное изображение должно быть в формате P5. Если входное было Color Type 2, то на выходе - P6.

Формат представлен ниже:

| P5 (PGM) | “P5\n<width> <height>\n255\n<Gray\_data>” |
| --- | --- |
| P6 (PPM) | “P6\n<width> <height>\n255\n<RGB\_data>” |

Windows:

* можно установить [WIC Codec](https://drive.google.com/drive/folders/1UnOW5W2UATMk4R9F2rqMX5j4ZwcfbIN3?usp=sharing) для просмотра PNM через стандартный просмотрщик изображений
* Gimp и Photoshop поддерживают PNM “из коробки”

Linux / Mac OS: поддерживают PNM “из коробки”

## **Требования к программе**

1. должна быть написана на C по заданному стандарту;
2. должна выполнять поставленную в ТЗ задачу;
3. не использовать внешние библиотеки, кроме указанных выше;
4. всегда корректно освобождать память и закрывать файлы;
5. обрабатывать ошибки:
   1. файл не открылся;
   2. не удалось выделить память;
   3. на вход передано неверное число аргументов командной строки;
   4. аргументы некорректны;
   5. формат файла не поддерживается;
   6. входной файл некорректен.

В этих случаях необходимо выдавать сообщение об ошибке и корректно завершаться с ненулевым кодом возврата (см. "return\_codes.h");

1. не писать в консоль ничего лишнего, кроме сообщений об ошибках и по желанию краткой справки по использованию (при запуске с неправильным числом аргументов).

**Сообщение об ошибке необходимо выводить в поток вывода ошибок**. Если вы будете выводить сообщение об ошибке в стандартный поток вывода, то это будет засчитываться за проваленный тест.

## **Ограничения**

1. Запрещено использование exit(...) в коде.
2. Запрещено создавать VLA-массивы (но можно VLA-указатели).
3. Ограничивается использование глобальных переменных (кроме констант) - необходимость их использования вы должны обосновать на защите. Ваш код должен быть максимально приспособлен к переносимости в другие проекты и/или использованию другими разработчиками.
4. Запрещается подключать системные библиотеки через #include "...".
5. Запрещается использовать setlocale(...). Учимся писать небольшие комментарии пользователю по-английски.
6. Запрещается использовать system(“pause”).

## **Пояснения по результатам первой проверки**

Кто получил плохую пометку W - смотрим сюда <https://t.me/c/1653237360/595> + на мой ответ в github

Кто прислал работы, которые совсем не связаны с текущим ТЗ (про png вообще ни строчки кода), получают только за codestyle, дедлайн зафиксирован.

У кого поддерживается 1 либа и более - проведена полная проверка.

Если же программа поддерживает только проверку сигнатуры входного файла или проверку обязательных блоков (IHDR и далее), то баллы были раскиданы по нескольким категориям, дедлайн зафиксирован. Почему не полная проверка? Потому что в первую очередь работа про конвертацию png в pnm, а не про валидацию png. Но, в целом, кто сделал больше, чем просто проверка сигнатуры png - вы достаточно близки к работающему варианту.

Важно (это касается большинства тех работ, которые я видела):

В пункте [Использование сторонних библиотек](#_kwjmnprttlq4) сказано про подключение заголовочных.

1. Они должны подключаться именно так и не иначе. Если вы подключите по-другому (будет ещё какая папка в заголовочном лишняя или же вы используете не те кавычки), то программа не соберётся и дальше работа проверяться не будет.
2. Подключать заголовочные надо только тогда, когда вы видите нужный макрос. Например, если вас собрали с макросом ZLIB, то только в этом случае вы подключаете его заголовочный и соответственно в коде вызываете функцию из него.
3. Если вы понимаете, что ни один из объявленный макросов вы не встретили, то смотрим про #error()

Если же вы не поддерживаете ни одну из библиотек, то вы вообще должны попадать в #error(), а не забить на эту конструкцию и работать. Во второй попытке это будет бить по работоспособности сильнее, чем при первой попытке.

И не обращайте внимания на то, как выглядит код на самом github (через web). Там немного едет форматирование визуально, но сами файлы при этом никак не изменяются github’ом.

Про .clang-format - этот файл будет меняться к лабе к лабе.

В течение дня в памятке появится её краткая версия с комментариями, где студенты часто бажат или делают не так: <https://docs.google.com/document/d/1wDz_yXuS7hlR7xosv89HjrKEqgcPDFoppNGZxqRgwKM/edit#heading=h.tlxly0ihghqa>

## **Пояснения по результатам второй проверки**

**[Отправка работы первый раз на проверку]** Сначала смотрим предыдущий раздел, а потом спускаемся сюда.

**[Хорошие работы]** У кого поддерживается как минимум zlib и есть валидация входного файла - вы молодцы и по большей части такое работы действительно было приятно смотреть. Работы такого уровня ожидались на этой проверке, но что-то пошло не так, о чём пойдет речь далее…

**[Подключение заголовочных]** За "zlib.h" и <return\_codes.h> вы получаете неполный балл. В ТЗ и комментариях в первой проверке было черным по белому написано, как подключаются библиотеки про deflate (и это касается не только include, но и сигнализации о том, что ваш код не поддерживает ту или иную библиотеку - страницы 2 и 3 ТЗ); как подключить return\_codes.h вам должно быть понятно ещё из лекций.

И также в вашем коде не должно быть #define ZLIB и прочее.

На защите вас будут спрашивать как вы подключали это библиотеки, что делали, как их собирали и в каком виде где они у вас хранились, поэтому вариант “ну вот у меня в CLion zlib подключился и всё” не прокатит.

**[clang-format]** Находится народ, который его не применяет. Тут я ничего не могу поделать, кроме как поставить 0 баллов по этому критерию...

**[Сообщения об ошибке]** Тем, кто делает printf(<сообщение об ошибке>), нужно срочно посмотреть предложение, выделенное жирным на странице 4 после нумерованного списка требований к программе.

**[VLA]** А для любителей создать массив по-простому - смотрим в Ограничения п. 2 (ТЗ, страница 4). В таблице это выглядит как по всем критериям ничего не стоит, кроме 0 за работу с памятью.

**[Работа с файлами, работа с память]** Ошибки, которые были в прошлой работе. В вашей программе должны корректно выделяться и освобождаться память, нет должно быть утечек памяти, должны правильно открываться и закрываться файлы, не должно быть незакрытых файлов при завершении работы программы, данные из файлов должны считываться правильно.

**[PNM]** Про выходной файл - находились работы, где народ недочитал/забивал на то, что PNM P5 и P6 - это двоичные файлы, за что терял на ровном месте кучу баллов по тестам.

**[Сигнатура PNG]** Желательно всем ещё раз проверить, что вы правильно проверяете сигнатуру png, потому что как оказалось, даже здесь можно набажить…

**[Не та работа]** Кто прислал работы, которые совсем не связаны с текущим ТЗ (отправка уровня "смотрите, я умею открывать и закрывать файл") - попытка не засчитывается, ибо у вас было 2 недели до этого сделать что-то более адекватное. В таблице это “-” в графе попытки.

**[Отправка ни о чём]** Появились студенты, которые отправляют повторно предыдущую (или почти не отличающуюся от предыдущей) попытку и ничего толком за неделю не сделали - вы пытались, но в этот раз я не засчитываю это за отправку работы. В таблице это “-” в графе попытки.