Требования к курсовым (весенний семестр)

CLI (Command Line Interface), обязательно

Программу требуется реализовать в виде утилиты, подобной стандартным linux-утилитам с которыми Вы уже имели дело: управление осуществляется посредством аргументов командной строки. Конкретный список команд зависит от решаемой задачи, но в общем случае обязательно:

- Наличие справки, которая распечатывается при вызове утилиты без аргументов или стандартными ключами (-h, –help)
- Обязательная обработка всех возможных исключительных ситуаций (даже бьющийся в агонии по клавиатуре пользователь не должен “уронить” вашу программу неверными аргументами)
- Для всех(для которых это имеет смысл) ключей должны быть как полные так и сокращенные версии(-h, –help)
- По-умолчанию последним аргументом утилите должно передаваться имя входного bmp/png файла. Имя выходного файла по умолчанию должно быть `out.bmp` (`out.png` для png), однако это должно быть возможным переопределить соответствующими ключом `-o`, `–output`
- Утилита должна иметь функцию печати подробной информации о bmp\png-файле, ключ `–info`
- В случае, если программой будут поддерживаться не все версии bmp/png-файлов, программа должна выводить об этом внятное сообщение, а не крашиться.
- Программа за один запуск выполняет только одно действие (если программа может рисовать прямоугольник и круг, то за один запуск рисуется либо прямоугольник, либо круг)
- Для каждого инструмента должен быть соответствующий ключ и ключи для его конфигурирования. Например, рисование прямоугольника может выглядеть как-то так:

```
mypaint --rectangle --start 0 0 --end 100 50 --color red picture.bmp
```

Реализация интерфейса должна быть с использованием **getopt или argp**

Ссылки на ресурсы

- Кратко о getopt: https://www.gnu.org/software/libc/manual/html_node/Getopt.html
- Подробно о getopt: https://www.gnu.org/software/libc/manual/pdf/libc.pdf В основном раздел 25.1.2 Parsing Program Arguments, но ctrl+f тоже принесет плоды

Необязательные дополнения к курсовой работе

Описанные в данной секции дополнения/требования являются **необязательными**, получить желаемую оценку можно без них.

GUI (Graphical User Interface), опционально в дополнении к CLI

Рядовые пользователи очень плохо умеют пользоваться терминалом, поэтому для их удобства зачастую реализуют графический интерфейс. В программе по обработке изображений такое напрашивается само собой (иначе убийца photoshop'а будет выглядеть несолидно).

Программа реализовывается в виде оконного графического приложения с использованием C++ фреймворка Qt (надо быть готовым в изучению C++ самостоятельно). Конкретный функционал приложения зависит от решаемой задачи, но в общем случае должно быть:

- Окно отображения изображения
- Некоторая палитра инструментов и возможность их настройки (цвет, например)(см. Paint, GIMP, Krita и т.д.)
- Пользователь должен иметь возможность, выбрав инструмент, на рисунке выделить область его применения
- Некоторое меню, позволяющее открыть/сохранить изображение, посмотреть подробную информацию об открытом bmp-файле, посмотреть информацию об авторе и краткую справку по приложению.
- Никакие действия пользователя не должны приводить к “падению” программы.
- В случае, если программой будут поддерживаться не все версии bmp-файлов, программа должна выводить об этом внятное сообщение, а не крашиться.

Ссылки на ресурсы

- Официальный сайт Qt: https://www.qt.io
- Откуда скачать Qt Creator: https://www.qt.io/qt-features-libraries-apis-tools-and-ide/
- “Первые шаги” для начинающий на Qt wiki: https://wiki.qt.io/Qt_for_Beginners
- Видеолекции Кринкина Кирилла Владимировича по Qt: https://www.youtube.com/playlist?list=PL754BBE9A89BA9D3B

Нюансы реализации

Qt в том числе имеет средства для работы с изображениями. Этими средствами (за исключением средств визуального отображения) пользоваться запрещено. Что это значит:

- Должен быть реализован класс, хранящий в себе представление изображения.
- Должны быть реализованы методы для загрузки изображения из файла и выгрузки его в файл.
- Заголовки изображения должны быть представлены в виде отдельных структур для каждого заголовка, если их несколько.
- Все операции из задания должны быть реализованы в виде отдельных public методов и реализованы должны быть самостоятельно (нельзя использовать стандартные средства, который сами все нарисуют)
- В заданиях с форматом PNG разрешается использовать стороннюю библиотеку для сжатия изображения, например, libpng. Подробнее: https://github.com/moevm/pr1-examples/tree/master/libpng

Документация

В любом хорошем проекте должна быть документация. Документация означает, что для каждой функции, структуры (и классов для C++) есть адекватное описание. Фразы в духе «Данная структура содержит данные для изображения» или «Функция обработки изображения» являются **плохим описанием**.

- Для функций дополнительно расписаны аргументы (описание, назначение, возможные значения)
- Для структур (и классов в C++) дополнительно расписаны все поля

Документацию можно написать сразу в коде, обратите вниммание на Doxygen

Создание .so библиотеки

Зачастую код пишется не в формате отдельной программы, а в формате библиотеки, чтобы вашими наработками могли воспользоваться в другом проекте другие разработчики. Поэтому можно разделить курсовую на две части:

1. Библиотека формата `.so` , которая содержит только функции для работы с изображениями: открытие и сохранение изображения, обработка изображений, ...
2. Главная часть, которая содержит в себе только main с CLI (и опционально GUI) обработкой и вызовом функций из разработанной библиотеки

ВАЖНО: Не нужно по умолчанию выполнять установку библиотеки в систему, сделайте для этого отдельную цель, например `make install`