**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МО ЭВМ**

# КУРСОВАЯ РАБОТА

**по дисциплине «Программирование»**

**ТЕМА: РАБОТА С ИЗОБРАЖЕНИЯМИ**

Студент гр. 3342 _____ Пушко К.Д.

Преподаватель _____ Глазунов С. А.

Санкт-Петербург

2024

**ЗАДАНИЕ**
**НА КУРСОВУЮ РАБОТУ**

Студент: Пушко К.Д.

Группа: 3342

Тема работы: Работа с изображениями

Вариант 5.14

Программа обязательно должна иметь CLI (опционально дополнительное использование GUI). Более подробно тут: http://se.moevm.info/doku.php/courses:programming:rules_extra_kurs

Программа должна реализовывать весь следующий функционал по обработке png-файла

Общие сведения

Формат картинки PNG (рекомендуем использовать библиотеку libpng)

без сжатия

файл может не соответствовать формату PNG, т.е. необходимо проверка на PNG формат. Если файл не соответствует формату PNG, то программа должна завершиться с соответствующей ошибкой.

обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями.

все поля стандартных PNG заголовков в выходном файле должны иметь те же значения что и во входном (разумеется кроме тех, которые должны быть изменены).

Программа должна иметь следующую функции по обработке изображений:

Рисование треугольника. Флаг для выполнения данной операции: `--triangle`. Треугольник определяется

Координатами его вершин. Флаг `--points`, значение задаётся в формате `x1.y1.x2.y2.x3.y3` (точки будут (x1; y1), (x2; y2) и (x3; y3)), где x1/x2/x3 – координаты по x, y1/y2/y3 – координаты по y

Толщиной линий. Флаг `--thickness`. На вход принимает число больше 0

Цветом линий. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет)

Треугольник может быть залит или нет. Флаг `--fill`. Работает как бинарное значение: флага нет – false , флаг есть – true.

цветом которым он залит, если пользователем выбран залитый. Флаг `--fill_color` (работает аналогично флагу `--color`)

Находит самый большой прямоугольник заданного цвета и перекрашивает его в другой цвет. Флаг для выполнения данной операции: `--biggest_rect`. Функционал определяется:

Цветом, прямоугольник которого надо найти. Флаг `--old_color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--old_color 255.0.0` задаёт красный цвет)

Цветом, в который надо его перекрасить. Флаг `--new_color` (работает аналогично флагу `--old_color`)

Создать коллаж размера N*M из одного изображения. Флаг для выполнения данной операции: `--collage`. Коллаж представляет собой это же самое изображение повторяющееся N*M раз.

Количество изображений по “оси” Y.  Флаг `--number_y`. На вход принимает число больше 0

Количество изображений по “оси” X.  Флаг `--number_x`. На вход принимает число больше 0

Рисование отрезка. Флаг для выполнения данной операции: `--line`.

Отрезок определяется:

координатами начала. Флаг `--start`, значение задаётся в формате `x.y`, где x – координата по x, y – координата по y

координатами конца. Флаг `--end` (аналогично флагу `--start`)

цветом. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет)

толщиной. Флаг `--thickness`. На вход принимает число больше 0

Каждую подзадачу следует вынести в отдельную функцию, функции сгруппировать в несколько файлов (например, функции обработки текста в один, функции ввода/вывода в другой). Сборка должна осуществляться при помощи make и Makefile или другой системы сборки

Разделы пояснительный записки: «Содержание», «Введение», «Структуры», «Функции», «Заключение», «Список использованных источников», «Приложение А. Примеры работы программы», «Приложение Б. Исходный код программы».

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 18.03.2024

Дата сдачи реферата: 17.05.2024

Дата защиты реферата: 22.05.2024

Студент _____ Пушко К. Д.

Преподаватель _____ Глазунов С. А.

# АННОТАЦИЯ

Курсовая работа заключается в создании программы для обработки изображений в формате PNG на языке C++. Она представляет собой набор компонентов для обработки файлов формата PNG, включающих функции рисования треугольника и линий, создания коллажа и перекрашивания наибольшего прямоугольника заданного цвета. Структура проекта включает каталог src для файлов исходного кода, заголовочных файлов, Makefile для сборки проекта и файл help со справкой к программе. Так же в самом каталоге проекта находится отчет. Для сборки проекта используется команда make. После сборки проекта можно выполнять различные операции с файлами PNG, указывая соответствующие параметры командной строки. Программа зависит от библиотеки libpng, поэтому перед сборкой необходимо убедиться, что она установлены в системе.

Пример работы программы приведен в приложении А.

Исходный код программы приведен в приложении Б.

## ANNOTATION

The course work consists in creating a program for processing PNG images in C++. It is a set of components for processing PNG files, including the functions of drawing a triangle and lines, creating a collage and repainting the largest rectangle of a given color. The project structure includes an src directory for source code files, header files, a Makefile for building the project, and a help file with help for the program. There is also a report in the project directory itself. The make command is used to build the project. After building the project, you can perform various operations with PNG files by specifying the appropriate command line options. The program depends on the libpng library, so before building it, you need to make sure that it is installed on the system.

An example of how the program works is given in Appendix A.

The source code of the program is given in Appendix B.

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

Цель работы: написать программу на языке C++, которая считывает изображение и обрабатывает его требуемым пользователем образом. Для этого требуется реализовать:

- Загрузку, хранение изображений из файла и запись изображения в файл;
- Считывание аргументов из командной строки и их валидация;
- Функции для рисования на загруженном изображении.
- Функция для создания нового изображения

# 1. КЛАССЫ

Чтобы облегчить написание и чтение кода, было принято решение реализовать и использовать 2 класса: Png и Arguments.

## 1.1 Класс Png

Класс Png представляет изображение в формате PNG. Он содержит следующие поля:

- width: Ширина изображения в пикселях.
- height: Высота изображения в пикселях.
- color_type: Тип цвета изображения (например, RGB, оттенки серого).
- bit_depth: Глубина цвета изображения.
- channels: Количество цветовых каналов.
- png_ptr: Указатель на структуру libpng для чтения/записи данных PNG.
- info_ptr: Указатель на структуру libpng для хранения информации о PNG.
- row_pointers: Указатель на массив указателей, каждый из которых указывает на строку данных изображения.

Так же в данном классе реализованы публичные методы для обработки изображения

- void print_png_info(): выводит в консоль информацию о Png файле.
- Png* create_collage(int size_x, int size_y): Возвращает указатель на созданный коллаж.
- void recolor_biggest_rect(int old_color[],int new_color[]): Перекрашивает наибольший прямоугольник заданного цвета.
- void draw_triangle(int x0, int y0, int x1,int y1,int x2,int y2,int color[], float thickness, bool is_fill = false, int fill_color[] = NULL): Рисует треугольник с заданными параметрами.

- void draw_line(int x0, int y0, int x1,int y1,int color[], float thickness): Рисует линию с заданными параметрами.

Для разделения логики и инкапсуляции в классе реализованы приватные методы:

- void set_pixel(int x, int y, int color[]): Устанавливает цвет пикселя.
- bool is_point_in_image(int x, int y): Проверка на нахождение пикселя в рамках изображения.
- bool check_pixel_color(int x0,int y0,int color[]): Проверка на цвет пикселя.
- void fill_circle(int x0,int y0,float thickness, int color[]): Рисует заполненный круг.
- bool is_point_in_triangle(int x, int y,int x0, int y0, int x1,int y1,int x2,int y2): Проверка на нахождение пикселя внутри треугольника.
- void fill_triangle(int x0, int y0, int x1,int y1,int x2,int y2,int color[]): Рисует заполненный треугольник.
- void insert_image(Png* image,int x0,int y0): Вставляет изображение по заданным координатам.
- void fill_rect(int x0,int y0,int x1,int y1,int color[]): Рисует заполненный прямоугольник.
- bool is_rect_filled(int x0,int y0,int x1,int y1,int color[]): Проверка на заполненность прямоугольника одним цветом.

Так же класс имеет конструктор, который инициализирует поля по умолчанию, и деструктор, который очищает память, занимаемую полями класса.

Этот класс необходим для представления информации о PNG-изображении в программе и обработке изображения.

### 1.2 Класс Argumenst

Класс Arguments необходим для хранения в себе значений по умолчанию и введённых пользователем значений. Включает в себя следующие поля:

- x0: координата по оси x.

- x1: координата по оси x.
- x2: координата по оси x.
- y0: координата по оси y.
- y1: координата по оси y.
- y2: координата по оси y.
- opt_number: номер выбранной опции.
- thickness: толщина линии.
- number_x: количество повторения изображения по оси x в коллаже.
- number_y: количество повторения изображения по оси x в коллаже.
- old_color[4]: цвет искомого прямоугольника.
- new_color[4]: цвет, в который необходимо перекрасить искомый прямоугольник.
- fill_color[4]: цвет заливки.
- color[4]: цвет линии.
- is_fill: флаг, отвечающий за необходимость заливки.
- is_help: флаг, отвечающий за необходимость вывода справки о работе программы.
- is_info: флаг, отвечающий за необходимость вывода информации об изображении.
- input_img_path: путь, по которому хранится изображение, которое необходимо обработать.
- output_img_path: путь, по которому необходимо сохранить обработанное изображение.

# 2. ФУНКЦИИ

## 2.1 Функция main

Функция main является главной функцией программы, которая обрабатывает аргументы командной строки и выполняет задачи над изображением. Она принимает аргументы argc (количество аргументов командной строки) и argv (массив строк, содержащий аргументы командной строки).

Внутри функции происходят следующие шаги:

Инициализация класса Argiments, в который с помощью функции read_arguments записываются считаные данные.

После этого программа выводит справку, если была выбрана данная опция.

Далее идет инициализация изображения и его считывание из файла.

После этого программа выводит информацию об изображении, если была выбрана данная опция.

В итоге с помощью оператора switch – case, программа обрабатывает и сохраняет изображение, согласно введённым аргументам.

При успехе выполнения программа выводит сообщение “Done” и завершает работу с кодом 0.

## 2.2 Функции по работе с файловой системой и считывании аргуметнтов.

Функция read_png_file возвращает указатель на экземпляр класса Png, считав изображение с указанного пути. Принимает на вход путь, по которому хранится изображение. В данной функции реализованы проверки на: успешность открытия файла с изображением, сигнатуру PNG файла, успешность создания структур png_ptr и info_ptr.

Функция save_png_file сохраняет обработанное изображение по указанному пути. Принимает на вход путь, по которому необходимо сохранить изображение, а так же указатель на само изображение. В данной функции так же

присутствуют проверки на: успешность сохранения файла, успешность создания png_ptr и info_ptr.

Функция read_arguments используется для чтения и обработки аргументов, переданных в программу при ее запуске. Принимает на вход количество аргументов и массив строк, содержащие эти аргументы. Изначально создается экземпляр класса Arguments, в который будет записана вся информация с ввода. Далее в функции определяются две строки: short_opts и long_opts. Эти строки содержат информацию о допустимых аргументах, которые может принимать программа. Аргументы могут быть как короткими (один символ, например, -f), так и длинными (начинаются с двух тире, например, --info). В строке short_opts определены все допустимые короткие аргументы, а в массиве long_opts - длинные. Далее в функции присутствует цикл while, который будет выполняться до тех пор, пока все аргументы не будут обработаны. Внутри цикла вызывается функция getopt_long, которая принимает на вход количество аргументов argc, массив строк argv, строку short_opts, массив long_opts и пустой указатель NULL. Эта функция возвращает следующий аргумент, который должен быть обработан, или значение -1, если все аргументы уже обработаны. Так же при считывании происходит первичная валидация данных. Если значение переменной opt не совпадает ни с одним из допустимых значений, то выводится сообщение об ошибке "Unknown flag" и программа завершается с кодом 43. В конце функции возвращается указатель на экземпляр класса Arguments.

Внутри цикла while присутствует оператор switch, который используется для обработки текущего аргумента, сохраненного в переменной opt. В зависимости от значения переменной opt выполняется соответствующий блок кода.

### 2.3 Функции выполнения заданий

Функция print_help выводит справочное сообщение, объясняющее использование программы и ее опций. Выводит информацию о курсовой работе и ее создателе, синтаксис использования, а также описания доступных опций,

включая краткие и длинные формы и их соответствующие объяснения. Сама справочная информация хранится в файле, который данная функция и считывает.

Метод print_png_info выводит информацию о PNG-изображении. Выводит различные детали о PNG-изображении, включая его ширину, высоту, тип цвета, глубину цвета и количество цветовых каналов. Тип цвета выводится как строковое представление, глубина цвета указывает на количество битов на выборку или на канал в изображении.

Метод draw_line рисует линию по заданным аргументам. Принимает на вход координаты двух точек, цвет и толщину линии. С помощью алгоритма Брезенхема, программа рассчитывает точки, которые лежат в центре линии и с помощью метода fill_circle рисует окружности с центрами в этих точках. Необходимость в рисовании окружности обуславливается тем, что с помощью окружностей, можно достоверно и легко создать необходимую толщину линии на рисунке.

Метод draw_triangle рисует треугольник по заданным аргументам. Принимает на вход координаты трех вершин, цвет и толщину линии, а также логическую переменную, отвечающую за необходимость заливки треугольника и цвет заливки. Сначала программа проверяет необходимость заливки, и если данная опция выбрана, то заливает треугольную область. Далее три раза идет вызов метода draw_line, тем самым проводя линии от вершины к вершине.

Метод recolor_biggest_rect находит наибольший прямоугольник заданного цвета и перекрашивает его. Принимает на вход искомый цвет и цвет, в который необходимо перекрасить прямоугольник. Метод проходит по всему изображению, начиная с левого верхнего края, и при соответствии пикселю искомому цвету начинает искать его ширину и высоту. Найдя эти величины, метод проверяет заполнена ли вся область искомым цветом, и при положительном исходе сравнивает площадь текущего прямоугольника с максимальной найденной площадью. Если найденная площадь наибольшая, то

метод записывает координаты вершин. Пройдясь по всему изображению, метод перекрашивает область с заданными координатами вершин в требуемый цвет.

Метод create_collage возвращает на указатель на экземпляр класса Png, в котором и реализован коллаж. Принимает на вход количество повторений изображения по горизонтали и вертикали. Сначала в методе создается новый экземпляр класса Png, после этого его полям присваиваются необходимые значения и создаются png_ptr и info_ptr. Далее идет выделение памяти на само изображение. В итоге, с помощью метода insert_image, идет создание коллажа. В результате программа возвращает указатель на созданный коллаж.

### 3.4 Вспомогательные функции для работы с изображением

Метод fill_rect закрашивает прямоугольник заданного цвета на изображении. Прямоугольник определяется диагональными координатами (x0, y0) и (x1, y1). Цвет задается массивом из трех целых чисел, каждое из которых представляет собой интенсивность одного из основных цветов (красный, зеленый, синий) в диапазоне от 0 до 255.

Метод is_rect_filled проверяет, закрашен ли каждый пиксель внутри прямоугольника заданным цветом. Прямоугольник определяется диагональными координатами (x0, y0) и (x1, y1). Цвет задается массивом из трех целых чисел, каждое из которых представляет собой интенсивность одного из основных цветов (красный, зеленый, синий) в диапазоне от 0 до 255. Метод возвращает true, если все пиксели внутри прямоугольника закрашены заданным цветом, и false в противном случае.

Метод insert_image вставляет одно изображение в другое в заданной позиции. Первым аргументом метод принимает указатель на изображение, которое необходимо вставить. Вторым и третьим аргументами передаются координаты (x0, y0), определяющие верхний левый угол области, в которую будет вставлено изображение.

Метод fill_triangle закрашивает треугольник заданного цвета на изображении. Треугольник определяется координатами трех вершин (x0, y0),

(x1, y1) и (x2, y2). Цвет задается массивом из трех целых чисел, каждое из которых представляет собой интенсивность одного из основных цветов (красный, зеленый, синий) в диапазоне от 0 до 255.

Метод is_point_in_triangle проверяет, принадлежит ли точка с координатами (x, y) треугольнику, заданному тремя вершинами (x0, y0), (x1, y1) и (x2, y2). Метод возвращает true, если точка принадлежит треугольнику, и false в противном случае.

Метод fill_circle закрашивает круг заданного цвета и толщины на изображении. Круг определяется координатами центра (x0, y0) и толщиной, заданной числом с плавающей точкой thickness. Цвет задается массивом из трех целых чисел, каждое из которых представляет собой интенсивность одного из основных цветов (красный, зеленый, синий) в диапазоне от 0 до 255.

Метод check_pixel_color проверяет, совпадает ли цвет пикселя с координатами (x0, y0) на изображении с заданным цветом. Цвет задается массивом из трех целых чисел, каждое из которых представляет собой интенсивность одного из основных цветов (красный, зеленый, синий) в диапазоне от 0 до 255. Метод возвращает true, если цвета пикселя и заданного цвета совпадают, и false в противном случае.

Метод is_point_in_image проверяет, принадлежит ли точка с координатами (x, y) изображению. Метод возвращает true, если точка принадлежит изображению, и false в противном случае.

Метод set_pixel устанавливает цвет пикселя с координатами (x, y) на изображении. Цвет задается массивом из трех целых чисел, каждое из которых представляет собой интенсивность одного из основных цветов (красный, зеленый, синий) в диапазоне от 0 до 255.

Разработанный программный код см. в приложении Б.

# ЗАКЛЮЧЕНИЕ

Была написана программу на языке C++, которая считывает изображение и обрабатывает его требуемым пользователем образом. Для этого были реализованы:

- Загрузка, хранение изображений из файла и запись изображения в файл;
- Ввод аргументов из коммандной строки;
- Функции для рисования на загруженном изображении.
- Функция для создания нового изображения

Ввод аргументов из командной строки был осуществлен с помощью функции getopt_long из стандартной библиотеки Си. Программа при любом желании пользователя не упадёт с ошибкой, а корректно завершит работу.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Язык программирования C / Керниган Брайан, Ритчи Деннис. СПб.: "Финансы и статистика", 2003.
2. Освой самостоятельно C++/Джесс Либерти. Вильямс, 2006.
3. Мануал по использованию libpng // libpng manual. URL: http://www.libpng.org/pub/png/libpng-1.2.5-manual.html.
4. Статья по функционалу C++. URL: https://metanit.com/cpp/tutorial/2.14.php.
5. Мануал по использованию getopt. URL: https://www.gnu.org/software/libc/manual/html_node/Getopt.html.

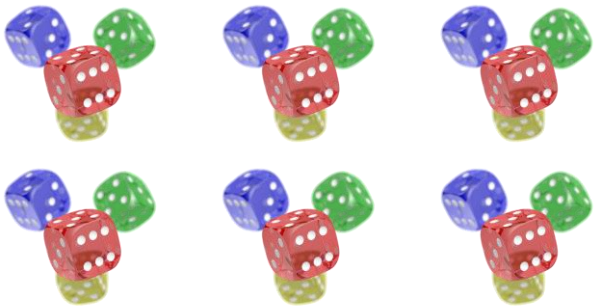
## ПРИЛОЖЕНИЕ А

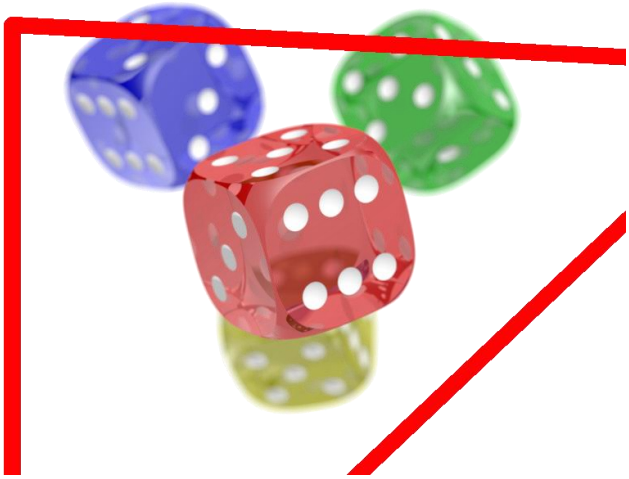
## ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

Таблица 1 – исходные изображения

| Изображение test1.png | Изображение input.png |
|---|---|
|  |  |

Таблица 2 – примеры работы программы с заданными аргументами

| Входные данные | Выходные данные |
|---|---|
| ./cw --collage --number_x 3 --number_y 2 test1.png |  |
| ./cw --line --start 20.20 --end 800.400 --thickness 25 --color 244.54.3 test2.png |  |

| | |
|---|---|
| ./cw --triangle --points 50.50.1000.100.50.1000 --thickness 20 --color 255.2.3 test1.pngcolor_replace --old_color 39.32.24 --new_color 1.1.1 input.png |  |
| ./cw --triangle --points 50.50.500.100.50.600 --thickness 20 --color 34.2.200 --fill --fill_color 255.255.255 test2.png |  |

# ПРИЛОЖЕНИЕ Б
# ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.cpp:

```
#include <iostream>
#include <getopt.h>
#include "png_class.h"
#include "file_interactions.h"
#include "input_system.h"
using namespace std;




void print_help() {
    FILE *file = fopen("help", "rb");
    if (!file) {
        printf("Help_info_file is not found.\n");
        exit(9);
    }
    char ch = (char) fgetc(file);
    while (ch != EOF) {
        printf("%c", ch);
        ch = (char) fgetc(file);
    }
    fclose(file);

}


int main(int argc, char** argv) {
    Arguments* args = readArguments(argc, argv);

    if (args == NULL)
            {
                cout << "Memory error\n";
                exit(45);
            }

    if(args->is_help)
```

```
    {
        print_help();
        exit(0);
    }
    Png* image = read_png_file(args->input_img_path);

    if (image == NULL)
            {
                cout << "Memory error\n";
                exit(45);
            }

    if(args->is_info)
    {
        image->print_png_info();
        exit(0);
    }


    switch (args->opt_number)
    {
        case 1:
        {
            if ( args->thickness <= 0)
            {
                cout << "Invalid thickness value\n";
                exit(42);
            }
            image->draw_triangle(args->x0,args->y0,args->x1,args->y1,args-
>x2,args->y2,args->color,args->thickness,args->is_fill,args->fill_color);
            save_png_file(image,args->output_img_path);
            break;
        }
        case 2:
        {
            image->recolor_biggest_rect(args->old_color,args->new_color);
            save_png_file(image,args->output_img_path);
            break;
        }
        case 3:
        {
            if (args->number_x <= 0 || args->number_y <= 0)
```

```
            {
                cout << "Invalid number_x or number_y value\n";
                exit(41);
            }
            Png* collage = image->create_collage(args->number_x, args-
>number_y);
            if (collage == NULL)
            {
                cout << "Memory error\n";
                exit(45);
            }

            save_png_file(collage,args->output_img_path);
            delete collage;
            break;
        }
        case 4:
        {
            if (args->thickness <= 0)
            {
                cout << "Invalid thickness value\n";
                exit(42);
            }
            image->draw_line(args->x0,args->y0,args->x1,args->y1,args-
>color,args->thickness);
            save_png_file(image,args->output_img_path);
            break;
        }
        case 5:
        {

            image->rhombus(args->color);
            save_png_file(image,args->output_img_path);
            break;
        }
        default:
        {
            cout << "Choose option!\n";
            exit(43);
        }
    }
```

```
    delete image;
    delete args;
    cout << "Done!\n";
    return 0;
}
```

Файл Makefile:

```
CXX = g++
CXXFLAGS = -std=c++11 -Wall -Werror
LDLIBS = -lpng

SRCS = main.cpp png_class.cpp file_interactions.cpp input_system.cpp
OBJS = $(SRCS:.cpp=.o)
HEADERS = png_class.h file_interactions.h input_system.h

all: $(OBJS)
	$(CXX) $(CXXFLAGS) -o cw $(OBJS) $(LDLIBS)

main.o: main.cpp $(HEADERS)
	$(CXX) $(CXXFLAGS) -c main.cpp -o main.o

png_class.o: png_class.cpp $(HEADERS)
	$(CXX) $(CXXFLAGS) -c png_class.cpp -o png_class.o

file_interactions.o: file_interactions.cpp $(HEADERS)
	$(CXX) $(CXXFLAGS) -c file_interactions.cpp -o file_interactions.o

input_system.o: input_system.cpp $(HEADERS)
	$(CXX) $(CXXFLAGS) -c input_system.cpp -o input_system.o

clean:
	rm -f $(OBJS)
```

Файл file_interactions.cpp:

```
#include "file_interactions.h"


Png* read_png_file(std::string file_name) {
    Png* image = new Png();


    if (image == NULL)
            {
                std :: cout << "Memory error\n";
                exit(45);
            }



    unsigned char header[8];
    FILE *fp = fopen(file_name.c_str(), "rb");
```

```
    if (!fp) {
        std::cout << "File could not be opened or not found.\n";
        exit(1);
    }
    fread(header, 1, 8, fp);
    if (png_sig_cmp(header, 0, 8)) {
        std::cout << "File is not recognized as a PNG.\n";
        exit(44);
    }

    image->png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL,
NULL, NULL);
    if (!image->png_ptr) {
        std::cout << "png_create_read_struct failed.\n";
        exit(45);
    }
    image->info_ptr = png_create_info_struct(image->png_ptr);
    if (!image->info_ptr) {
        std::cout << "png_create_info_struct failed.\n";
        exit(46);
    }
    png_init_io(image->png_ptr, fp);
    png_set_sig_bytes(image->png_ptr, 8);
    png_read_info(image->png_ptr, image->info_ptr);
    image->width = png_get_image_width(image->png_ptr, image->info_ptr);
    image->height = png_get_image_height(image->png_ptr,
image->info_ptr);
    image->bit_depth = png_get_bit_depth(image->png_ptr,
image->info_ptr);
    image->color_type = png_get_color_type(image->png_ptr,
image->info_ptr);
    image->channels = png_get_channels(image->png_ptr, image->info_ptr);
    png_read_update_info(image->png_ptr, image->info_ptr);

    image->row_pointers = (png_byte**) new png_byte*[image->height];
    if (image->row_pointers == NULL)
            {
                std::cout << "Memory error\n";
```

```
                exit(45);
            }

    for (int i = 0; i < image->height; i++) {
        image->row_pointers[i] = new
png_byte[png_get_rowbytes(image->png_ptr, image->info_ptr)];

        if (image->row_pointers[i] == NULL)
            {
                std::cout << "Memory error\n";
                exit(45);
            }
    }

    png_read_image(image->png_ptr, image->row_pointers);
    fclose(fp);
    return image;
}

void save_png_file(Png* image, std::string save_path) {
    FILE *fp = fopen(save_path.c_str(), "wb");
    if (!fp) {
        std::cout << "File save failed\n";
        exit(44);
    }

    image->png_ptr = png_create_write_struct(PNG_LIBPNG_VER_STRING, NULL,
NULL, NULL);
    if (!image->png_ptr) {
        std::cout << "png_ptr failed\n";
        exit(45);
    }

    image->info_ptr = png_create_info_struct(image->png_ptr);
    if (!image->info_ptr) {
        std::cout << "info_ptr failed\n";
        exit(46);
    }
```

```
    png_init_io(image->png_ptr, fp);

    png_set_IHDR(image->png_ptr, image->info_ptr, image->width,
image->height, image->bit_depth, image->color_type,
                 PNG_INTERLACE_NONE, PNG_COMPRESSION_TYPE_BASE,
PNG_FILTER_TYPE_BASE);

    png_write_info(image->png_ptr, image->info_ptr);
    png_write_image(image->png_ptr, image->row_pointers);
    png_write_end(image->png_ptr, NULL);

    fclose(fp);
}
```

Файл file_interactions.h:

```
#ifndef FILES_INTERACTIONS_H
#define FILES_INTERACTIONS_H

#include "png_class.h"
#include <string>

Png* read_png_file(std::string file_name);
void save_png_file(Png* image, std::string save_path);


#endif
```

Файл help:

```
Course work for option 5.14, created by Pushko Konstantin.
Usage: ./cw [OPTIONS] [input_file]

Options:
1)Рисование треугольника.
2)Находит самый большой прямоугольник заданного цвета и перекрашивает его
в другой цвет.
3)Создать коллаж размера N*M из одного изображения.
4)Рисование отрезка.

Флаги:
--help (-h): Выведение справки по работе с программой.

--info (-f): Выведение информации о PNG-файле.

--input (-i) {filename}: Ввод имени исходного PNG-файла.

--output (-o) {filename}: Ввод имени обработанного PNG-файла.
```

--triangle (-t) {--points, --color, --thickness, --is_fill, --fill_color}: Рисует треугольник с заданными координатами вершин, цветом и толщиной линии. Наличие флага fill отвечает за заливку треугольника, цвет заливки так же необходимо задать.

--biggest_rect (-r) {--old_color, --new_color}: Перекрашивает наибольший прямоугольник заданого цвета в другой цвет.

--collage (-c) {--number_y, --number_x}: Создает коллаж изображения размером N*M.

--line (-l) {--start, --end, --color, --thickness}: Рисует линию с заданными координатами начала и конца, цветом и толщиной линии.

--points (-p): Формат - x1.y1.x2.y2.x3.y3

--color (-c) Формат - rrr.ggg.bbb

--fill_color (-I) Формат - rrr.ggg.bbb

--old_color (-O) Формат - rrr.ggg.bbb

--new_color (-N) Формат - rrr.ggg.bbb

--thickness (-t): Формат - число

--number_y (-y): Формат - число

--number_x (-x): Формат - число

--start (-s): Формат - x.y

--end (-e): Формат - x.y

Файл input_system.cpp:

```
#include "input_system.h"
#include <getopt.h>
#include <iostream>



using namespace std;


Arguments::Arguments()
{
    x0 = 0;
    x1 = 0;
    x2 = 0;
    y0 = 0;
    y1 = 0;
    y2 = 0;
```

```
opt_number = 0;
thickness = -1;
number_x = 0;
number_y = 0;

old_color = new int[4];

if (old_color == NULL)
        {
            cout << "Memory error\n";
            exit(45);
        }

old_color[0] = 0;
old_color[1] = 0;
old_color[2] = 0;
old_color[3] = 255;

new_color = new int[4];

if (new_color == NULL)
        {
            cout << "Memory error\n";
            exit(45);
        }

new_color[0] = 0;
new_color[1] = 0;
new_color[2] = 0;
new_color[3] = 255;

fill_color = new int[4];

if (fill_color == NULL)
        {
            cout << "Memory error\n";
            exit(45);
```

```
            }

    fill_color[0] = 0;
    fill_color[1] = 0;
    fill_color[2] = 0;
    fill_color[3] = 255;

    color = new int[4];

    if (color == NULL)
            {
                cout << "Memory error\n";
                exit(45);
            }

    color[0] = 0;
    color[1] = 0;
    color[2] = 0;
    color[3] = 255;

    is_fill = false;
    is_help = false;
    is_info = false;

    input_img_path = "";
    output_img_path = "output.png";
}

Arguments::~Arguments()
{
    delete[] old_color;
    delete[] new_color;
    delete[] fill_color;
    delete[] color;
}

Arguments* readArguments(int argc, char** argv)
{
```

```
    Arguments* args = new Arguments();

    if (args == NULL)
            {
                cout << "Memory error\n";
                exit(45);
            }

    args->input_img_path = argv[argc - 1];

    const char* short_opts = "vhf:TRCLp:t:c:FI:o:O:N:y:x:s:e:";

    int opt;
    const struct option long_opts[] ={
            {"rhombus",no_argument,NULL,'v'},
            {"help", no_argument, NULL, 'h'},
            {"info", required_argument,NULL,'f'},
            {"triangle", no_argument,NULL,'T'},
            {"biggest_rect", no_argument,NULL,'R'},
            {"collage", no_argument,NULL,'C'},
            {"line", no_argument,NULL, 'L'},
            {"points", required_argument, NULL, 'p' },
            {"thickness", required_argument, NULL, 't' },
            {"color", required_argument, NULL, 'c' },
            {"fill", no_argument, NULL, 'F' },
            {"fill_color", required_argument, NULL, 'I' },
            {"output", required_argument, NULL, 'o' },
            {"input", required_argument, NULL, 'i' },
            {"old_color", required_argument, NULL, 'O' },
            {"new_color", required_argument, NULL, 'N' },
            {"number_y", required_argument, NULL, 'y' },
            {"number_x", required_argument, NULL, 'x' },
            {"start", required_argument, NULL, 's' },
            {"end", required_argument, NULL, 'e' }
    };

    while ((opt = getopt_long(argc, argv, short_opts, long_opts,
NULL)) != -1)
```

```
{
    switch (opt)
    {

        case 'h':
        {
            args->is_help = true;
            break;
        }
        case 'f':
        {
            args->is_info = true;
            break;
        }
        case 'T':
        {
            args->opt_number = 1;
            break;
        }
        case 'v':
        {
            args->opt_number = 5;

            break;
        }
        case 'R':
        {
            args->opt_number = 2;
            break;
        }
        case 'C':
        {
            args->opt_number = 3;
            break;
        }
        case 'L':
        {
            args->opt_number = 4;
```

```
                break;
            }
            case 'p':
            {
                int n = sscanf(optarg, "%d.%d.%d.%d.%d.%d", &args->x0,
&args->y0, &args->x1, &args->y1,&args->x2, &args->y2);
                if (n != 6)
                {
                    cout << "Invalid points format\n";
                    exit(41);
                }
                break;
            }
            case 't':
            {
                int n = sscanf(optarg, "%f", &args->thickness);
                if(n==1)
                {
                    if(args->thickness <= 0)
                    {
                        cout << "Invalid thickness value\n";
                        exit(42);
                    }
                }
                else
                {
                    cout << "Invalid thickness format\n";
                    exit(41);
                }
                break;
            }
            case 'c':
            {
                int n = sscanf(optarg, "%d.%d.%d", &args->color[0],
&args->color[1], &args->color[2]);
                if (n == 3)
                {
```

```
                    if(!(args->color[0] >= 0 && args->color[0] <= 255 &&
args->color[1] >= 0 && args->color[1] <= 255 && args->color[2] >= 0 &&
args->color[2] <= 255))
                    {
                        cout << "Invalid color values";
                        exit(42);
                    }
                }
                else
                {
                    cout << "Invalid color format\n";
                    exit(41);
                }
                break;
            }
            case 'F':
            {
                args->is_fill = true;
                break;
            }
            case 'I':
            {
                int n = sscanf(optarg, "%d.%d.%d", &args->fill_color[0],
&args->fill_color[1], &args->fill_color[2]);
                if (n == 3)
                {
                    if(!(args->fill_color[0] >= 0 && args->fill_color[0]
<= 255 && args->fill_color[1] >= 0 && args->fill_color[1] <= 255 &&
args->fill_color[2] >= 0 && args->fill_color[2] <= 255))
                    {
                        cout << "Invalid color values";
                        exit(42);
                    }

                }
                else
                {
                    cout << "Invalid color format\n";
```

```
                exit(41);
            }
            break;
        }
        case 'o':
        {
            args->output_img_path = optarg;
            break;
        }
        case 'i':
        {
            args->input_img_path = optarg;
            break;
        }
        case 'O':
        {
            int n = sscanf(optarg, "%d.%d.%d", &args->old_color[0],
&args->old_color[1], &args->old_color[2]);
            if (n == 3)
            {
                if(!(args->old_color[0] >= 0 && args->old_color[0] <=
255 && args->old_color[1] >= 0 && args->old_color[1] <= 255 &&
args->old_color[2] >= 0 && args->old_color[2] <= 255))
                {
                    cout << "Invalid color values";
                    exit(42);
                }
            }
            else
            {
                cout << "Invalid color format\n";
                exit(41);
            }
        }
        case 'N':
        {
            int n = sscanf(optarg, "%d.%d.%d", &args->new_color[0],
&args->new_color[1], &args->new_color[2]);
```

```
                if (n == 3)
                {
                    if(!(args->new_color[0] >= 0 && args->new_color[0] <=
255 && args->new_color[1] >= 0 && args->new_color[1] <= 255 &&
args->new_color[2] >= 0 && args->new_color[2] <= 255))
                    {
                        cout << "Invalid color values";
                        exit(42);
                    }
                }
                else
                {
                    cout << "Invalid color format\n";
                    exit(41);
                }
                break;
            }
            case 'y':
            {
                int n = sscanf(optarg,"%d",&args->number_y);
                if(n!=1)
                {
                    cout << "Invalid int value";
                    exit(41);
                }
                if(args->number_y<1)
                {
                    cout << "Invalid number_y value";
                    exit(42);
                }

                break;
            }
            case 'x':
            {

                int n = sscanf(optarg,"%d",&args->number_x);
                if(n!=1)
```

```
            {
                cout << "Invalid int value";
                exit(41);
            }
            if(args->number_x<1)
            {
                cout << "Invalid number_x value";
                exit(42);
            }

            break;
        }case 's':
        {
            int n = sscanf(optarg, "%d.%d",&args->x1, &args->y1);
            if (n != 2)
            {
                cout << "Invalid points format\n";
                exit(41);
            }
            break;
        }
        case 'e':
        {
            int n = sscanf(optarg, "%d.%d", &args->x0, &args->y0);
            if (n != 2)
            {
                cout << "Invalid points format\n";
                exit(41);
            }
            break;

        }
        default:
        {
            cout << "Unknown flag\n";
            exit(43);
        }
    }
```

```
    }


    return args;


}
```

Файл input_system.h:

```
#ifndef INPUTSYSTEM_H
#define INPUTSYSTEM_H

#include <iostream>
class Arguments
{
public:
    int x0;
    int x1;
    int x2;
    int y0;
    int y1;
    int y2;
    int opt_number;
    float thickness;
    int number_x;
    int number_y;
    int old_color[4];
    int new_color[4];
    int fill_color[4];
    int color[4];
    bool is_fill;
    bool is_help;
    bool is_info;
    std::string input_img_path;
    std::string output_img_path;


    Arguments();

};

Arguments* readArguments(int argc, char** argv);
#endif
```

Файл png_class.cpp:

```
#include "png_class.h"


Png::Png() {
    width = 0;
    height = 0;
    color_type = 0;
    bit_depth = 0;
```

```
    channels = 0;
    png_ptr = nullptr;
    info_ptr = nullptr;
    row_pointers = nullptr;
}

Png::~Png() {
    for (int i = 0; i < height; i++) {
        free(row_pointers[i]);
    }
    free(row_pointers);
    if (png_ptr != nullptr) {
        png_destroy_read_struct(&png_ptr, &info_ptr, nullptr);
    }
}

void Png::print_png_info() {
    std::cout << "Image Width: " << width << '\n';
    std::cout << "Image Height: " << height << '\n';
    std::cout << "Image Bit Depth: " << bit_depth << '\n';
    std::cout << "Image Channels: " << channels << '\n';
    if (color_type == PNG_COLOR_TYPE_RGB) {
        std::cout << "Image Colour Type: RGB\n";
    } else {
        std::cout << "Image Colour Type: RGB_A\n";
    }
}

Png* Png::create_collage(int size_x, int size_y) {
    Png* new_image = new Png();
    new_image->width = size_x * this->width;
    new_image->height = size_y * this->height;
    new_image->color_type = this->color_type;
    new_image->bit_depth = this->bit_depth;
    new_image->channels = this->channels;

    new_image->png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING,
nullptr, nullptr, nullptr);
```

```
    new_image->info_ptr = png_create_info_struct(new_image->png_ptr);

    png_read_update_info(new_image->png_ptr, new_image->info_ptr);

    new_image->row_pointers = (png_byte **)malloc(sizeof(png_byte *) *
new_image->height);
    for (int y = 0; y < new_image->height; y++) {
        new_image->row_pointers[y] = (png_byte
*)malloc((png_get_rowbytes(this->png_ptr, this->info_ptr) / this->width)
* new_image->width);
    }

    for (int y = 0; y < this->height*size_y; y+=this->height) {
        for (int x = 0; x < this->width*size_x; x+=this->width) {
            this->insert_image(new_image, x, y);
        }
    }

    return new_image;
}

void Png::recolor_biggest_rect(int old_color[], int new_color[]) {
    int max_area = 0;
    int max_rect_cords[4];

    for (int y = 0; y < height; ++y) {
        for (int x = 0; x < width; ++x) {
            int current_wight = 0;
            while (check_pixel_color(x + current_wight, y, old_color)) {
                current_wight++;
            }

            int current_height = 0;
            while (check_pixel_color(x, y + current_height, old_color)) {
                current_height++;
            }

            if (max_area < current_wight * current_height) {
```

```
                if (is_rect_filled(x, y, x + current_wight, y +
current_height, old_color)) {
                    max_area = current_wight * current_height;
                    max_rect_cords[0] = x;
                    max_rect_cords[1] = y;
                    max_rect_cords[2] = x + current_wight;
                    max_rect_cords[3] = y + current_height;
                }
            }
        }
    }

    if (max_area > 1) {
        fill_rect(max_rect_cords[0], max_rect_cords[1],
max_rect_cords[2], max_rect_cords[3], new_color);
    } else
    {
        std::cout << "Rect not found\n";
    }
}

void Png::draw_triangle(int x0, int y0, int x1, int y1, int x2, int y2,
int color[], float thickness, bool is_fill, int fill_color[]) {

    if (fill_color == nullptr) {
        int def_arr[] = { 0, 0, 0, 255 };
        fill_color = def_arr;
    }

    if (is_fill) {
        fill_triangle(x0, y0, x1, y1, x2, y2, fill_color);
    }

    draw_line(x0, y0, x1, y1, color, thickness);
    draw_line(x1, y1, x2, y2, color, thickness);
    draw_line(x2, y2, x0, y0, color, thickness);
}
```

```
void Png::draw_line(int x0, int y0, int x1, int y1, int color[], float
thickness) {

    int dx = abs(x1 - x0);
    int dy = abs(y1 - y0);
    int signX = (x0 < x1) ? 1 : -1;
    int signY = (y0 < y1) ? 1 : -1;
    int err = dx - dy;

    while (x0 != x1 || y0 != y1) {


            fill_circle(x0, y0, thickness, color);


        int err2 = err * 2;

        if (err2 > -dy) {
            err -= dy;
            x0 += signX;
        }
        if (err2 < dx) {
            err += dx;
            y0 += signY;
        }
    }
}

void Png::set_pixel(int x, int y, int color[]) {
    row_pointers[y][x * channels + 0] = color[0];
    row_pointers[y][x * channels + 1] = color[1];
    row_pointers[y][x * channels + 2] = color[2];
    if (color_type == PNG_COLOR_TYPE_RGBA) {
        row_pointers[y][x * channels + 3] = color[3];
    }
}

bool Png::is_point_in_image(int x, int y) {
```

```
    return ((x >= 0 && x < width) && (y >= 0 && y < height));
}

bool Png::check_pixel_color(int x0, int y0, int color[]) {
    if (!is_point_in_image(x0, y0)) {
        return false;
    }
    if (color_type == PNG_COLOR_TYPE_RGB) {
        if (row_pointers[y0][x0 * channels + 0] == color[0] &&
row_pointers[y0][x0 * channels + 1] == color[1] &&
            row_pointers[y0][x0 * channels + 2] == color[2]) {
            return true;
        }
    }
    else if (color_type == PNG_COLOR_TYPE_RGBA) {
        if (row_pointers[y0][x0 * channels + 0] == color[0] &&
row_pointers[y0][x0 * channels + 1] == color[1] &&
            row_pointers[y0][x0 * channels + 2] == color[2] &&
row_pointers[y0][x0 * channels + 3] == color[3]) {
            return true;
        }
    }
    return false;
}

void Png::fill_circle(int x0, int y0, float thickness, int color[]) {
    thickness++;
    for (int x = -floor(thickness / 2); x < round(thickness / 2); x++)
    {
        if (x0 + x > this->width || x0+x < 0)
        {
            continue;
        }
        int height = sqrt(floor(thickness / 2) * round(thickness / 2) - x
* x);
        for (int y = -height; y < height; y++)
        {
```

```
                if (!is_point_in_image(x+x0, y+y0)) {
                    continue;
                }
                if (check_pixel_color(x+x0, y+y0, color)) {
                    continue;
                }
                set_pixel(x+x0, y+y0, color);

        }
    }
}

bool Png::is_point_in_triangle(int x, int y, int x0, int y0, int x1, int
y1, int x2, int y2) {
    int a = (x0 - x) * (y1 - y0) - (x1 - x0) * (y0 - y);
    int b = (x1 - x) * (y2 - y1) - (x2 - x1) * (y1 - y);
    int c = (x2 - x) * (y0 - y2) - (x0 - x2) * (y2 - y);
    return ((a >= 0 && b >= 0 && c >= 0) || (a <= 0 && b <= 0 && c <=
0));
}

void Png::fill_triangle(int x0, int y0, int x1, int y1, int x2, int y2,
int color[]) {
    int max_x = std::max(x0, std::max(x1, x2));
    int min_x = std::min(x0, std::min(x1, x2));
    int max_y = std::max(y0, std::max(y1, y2));
    int min_y = std::min(y0, std::min(y1, y2));

    for (int i = min_x; i < max_x; ++i) {
        for (int j = min_y; j < max_y; ++j) {
            if (is_point_in_triangle(i, j, x0, y0, x1, y1, x2, y2)) {
                set_pixel(i, j, color);
            }
        }
    }
}

void Png::insert_image(Png* image, int x0, int y0) {
```

```
    for (int x = 0; x < width; ++x) {
        for (int y = 0; y < height; ++y) {
            int color_arr[4];

            color_arr[0] = row_pointers[y][x * channels + 0];
            color_arr[1] = row_pointers[y][x * channels + 1];
            color_arr[2] = row_pointers[y][x * channels + 2];
            if (color_type == PNG_COLOR_TYPE_RGBA) {
                color_arr[3] = row_pointers[y][x * channels + 3];
            }
            image->set_pixel(x + x0, y + y0, color_arr);
        }
    }
}

void Png::fill_rect(int x0, int y0, int x1, int y1, int color[]) {
    for (int y = y0; y < y1; ++y) {
        for (int x = x0; x < x1; ++x) {
            set_pixel(x, y, color);
        }
    }
}

bool Png::is_rect_filled(int x0, int y0, int x1, int y1, int color[]) {
    for (int y = y0; y < y1; ++y) {
        for (int x = x0; x < x1; ++x) {
            if (!check_pixel_color(x, y, color)) {
                return false;
            }
        }
    }
    return true;
}
```

Файл png_class.h:

```
#ifndef PNG_CLASS_H
#define PNG_CLASS_H
#include <png.h>
#include <iostream>
```

```
#include <cmath>

class Png {
public:
    int width, height;
    png_byte color_type;
    png_byte bit_depth;
    png_byte channels;
    png_structp png_ptr;
    png_infop info_ptr;
    png_byte **row_pointers;

    Png();
    ~Png();

    void print_png_info();
    Png* create_collage(int size_x, int size_y);
    void recolor_biggest_rect(int old_color[],int new_color[]);
    void draw_triangle(int x0, int y0, int x1,int y1,int x2,int y2,int
color[], float thickness, bool is_fill = false, int fill_color[] = NULL);
    void draw_line(int x0, int y0, int x1,int y1,int color[], float
thickness);

private:
    void set_pixel(int x, int y, int color[]);
    bool is_point_in_image(int x, int y);
    bool check_pixel_color(int x0,int y0,int color[]);
    void fill_circle(int x0,int y0,float thickness, int color[]);
    bool is_point_in_triangle(int x, int y,int x0, int y0, int x1,int
y1,int x2,int y2);
    void fill_triangle(int x0, int y0, int x1,int y1,int x2,int y2,int
color[]);
    void insert_image(Png* image,int x0,int y0);
    void fill_rect(int x0,int y0,int x1,int y1,int color[]);
    bool is_rect_filled(int x0,int y0,int x1,int y1,int color[]);
}
#endif
```