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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка изображений формата BMP

Студент гр. 3383 _____ Боривец С. Ю.

Преподаватель _____ Гаврилов А.В.

Санкт-Петербург

2024

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Боривец С. Ю.

Группа 3383

Тема работы: Обработка изображений формата BMP

Вариант 3

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 bmp-файла

Общие сведения

- 24 бита на цвет
- без сжатия
- файл может не соответствовать формату BMP, т.е. необходимо проверка на BMP формат (дополнительно стоит помнить, что версий у формата несколько). Если файл не соответствует формату BMP или его версии, то программа должна завершиться с соответствующей ошибкой.
- обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями.
- обратите внимание на порядок записи пикселей
- все поля стандартных BMP 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:

1. Фильтр rgb-компонент. Флаг для выполнения данной операции: `--rgbfilter`. Этот инструмент должен позволять для всего изображения либо установить в диапазоне от 0 до 255 значение заданной компоненты. Функционал определяется
    - Какую компоненту требуется изменить. Флаг `--component_name`. Возможные значения `red`, `green` и `blue`.
    - В какой значение ее требуется изменить. Флаг `--component_value`. Принимает значение в виде числа от 0 до 255
2. Рисование квадрата. Флаг для выполнения данной операции: `--square`. Квадрат определяется:
    - Координатами левого верхнего угла. Флаг `--left_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y
    - Размером стороны. Флаг `--side_size`. На вход принимает число больше 0
    - 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)
    - 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`)
3. Поменять местами 4 куска области. Флаг для выполнения данной операции: `--exchange`. Выбранная пользователем прямоугольная область делится на 4 части и эти части меняются местами. Функционал определяется:
    - Координатами левого верхнего угла области. Флаг `--left_up`,

значение задаётся в формате `left.up`, где left – координата по x, up – координата по y

- Координатами правого нижнего угла области. Флаг `--right_down`, значение задаётся в формате `right.down`, где right – координата по x, down – координата по y
- Способом обмена частей: “по кругу”, по диагонали. Флаг `--exchange_type`, возможные значения: `clockwise`, `counterclockwise`, `diagonals`

4. Находит самый часто встречаемый цвет и заменяет его на другой заданный цвет. Флаг для выполнения данной операции: `--freq_color`. Функционал определяется:
    - Цветом, в который надо перекрасить самый часто встречаемый цвет. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет)

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.

Содержание пояснительной записки:
«Аннотация», «Содержание», «Введение», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 18.03.2024

Дата сдачи реферата: 13.05.2024

Дата защиты реферата: 15.05.2024

Студент гр. 3383	Боривец С. Ю.
Преподаватель	Гаврилов А.В.

АННОТАЦИЯ

Курсовая работа заключается в написании программы на языке C++, которая должна обработать полученное от пользователя изображение формата BMP, по команде, которую он укажет.

Первой строкой программа выводит вариант курсовой работы, имя и фамилию автора. Предусмотрена работа с командной строкой, управление осуществляется посредством её аргументов. Существуют множество флагов, информацию о которых можно посмотреть, используя флаг -h/--help. Исходя из флагов и аргументов программа делает нужную обработку BMP-изображения. Перед обработкой изображения по флагам, программа проверяет входное изображение на количество бит на цвет, сжатие, BMP формат, выравнивание. Предусмотрена обработка ошибок.

Обработанное изображение сохраняется в файле BMP с названием введенным пользователем. Если название не введено, то файл будет называться “out.bmp”.

SUMMARY

The course work consists in writing a program in C++, which should process the BMP image received from the user, according to the command that he specifies.

The first line of the program displays the version of the term paper, the first and last name of the author. It is provided to work with the command line, control is carried out through its arguments. There are many flags, information about which can be viewed using the -h/--help flag. Based on the flags and arguments, the program does the necessary processing of the BMP image. Before processing the image by flags, the program checks the input image for the number of bits for color, compression, BMP format, alignment. Error handling is provided.

The processed image is saved in a BMP file with the name entered by the user. If the name is not entered, the file will be called “out.bmp".

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: создать программу, которая будет обрабатывать изображение, по флагам введённых в командой строке.

Основные задачи:

- Реализовать класс Bitmap для считывания, вывода, обработки изображений.
- Реализовать CLI (Command Line Interface) – интерфейс, в котором управление осуществляется посредством аргументов командной строки.
- Реализовать функции по обработке изображений, требуемые заданием.
- Написать Makefile для сборки программы.
- Сделать обработку всех возможных исключительных ситуаций.

1. Заголовочный файл my_lib.h. Структуры BitmapFileHeader, BitmapInfoHeader и Rgb.

1.1 Заголовочный файл my_lib

Здесь находится описанипе структур, все нужные стандартные библиотеки, макросы номеров ошибок, указывающие на причину ошибки. Для правильных записи и вывода структур было необходимо задать выравнивание полей структуры с помощью директивы #pragma, которая позволяет давать некоторые инструкции компилятору. Инструкция pack после ключевого слова pragma управляет выравниванием байтов. Для корректного считывания и записи было выставлено значение 1, то есть выравнивания совсем не будет, так как самое короткое возможное машинное слово 1 байт.

После описания всех структур следует восстановление выравнивание с помощью #pragma(pop).

Библиотеки, с которыми ведется работа:

- <iostream> - ввод и вывод через потоки(cout, cin).
- <cstdio> - дополнительные функции ввода и вывода из C(fread, fwrite).
- <cstring> - функции для работы со строками из C(strchr).
- <getopt.h> - библиотека для реализации CLI(getopt_long).
- <vector> - библиотека, позволяющая использовать динамический массив, обеспечивающий быстрое добавление новых элементов в конец и меняющий свой размер при необходимости.
- <map> - библиотека, позволяющая использовать словари.
- <clocale> - используется для подключения локали, ради корректного отображения кириллицы

1.2 Структура BitmapFileHeader

Структура, в которой содержится информация о файле: код формата, общий размер файла в байтах, два зарезервированных поля и адрес битового массива в данном файле.

1.3 Структура BitmapInfoHeader

Структура, в которой содержится информация об изображении: размер заголовка, ширина и длина растра в пикселях, количество цветовых плоскостей(должно быть равно 1), количество бит на пиксель, метод сжатия, горизонтальное и вертикальное разрешение, чисто цветов изображения, число основных цветов.

1.4 Структура Rgb

Структура, в которой содержатся данные о цвете пикселя: значение потоков синего, зеленого и красного.

2. Основной файл main.cpp

Здесь происходит основная работа программы – ввод изображений, взаимодействие с флагами и аргументами командной строки, вызов функций обработки, вывод изображений. Подключены все требующиеся заголовочные файлы.

Сначала выводится строка с информацией о варианте курсовой работе и об авторе программы. Затем устанавливается русская локаль. После этого идет структура с длинными опциями, в которой описаны флаги: их названия, наличие аргумента и короткий флаг. Далее идет двумерный словарь commander значений для функций обработки, в котором сохраняется значение аргументов по ключам в виде флагов для того, чтобы иметь возможность записывать их в любом порядке. Чтобы получить значение, нужно сначала обратиться по заглавному первому символу функции(короткий флаг функции), а затем обратиться к нужному флагу. После объявляются строки для названий входного и выходного файла, а также mode, где сохраняется короткий флаг функции.

Проверяется количество аргументов: если их нет, то выводится справка, программа завершает работу, иначе программа переходит к сбору информации с флагов. Функция getopt_long получает информацию по флагам. Она принимает на вход количество аргументов, аргументы, строку с короткими флагами(двоеточие в ней означает, что должен поступить аргумент) и структуру с длинными опциями. Далее идет оператор switch, в котором каждый case сохраняет значение флага в commander, кроме ‘h’, который выводит справку и завершает работу, игнорируя остальные введённые аргументы. После объявляется класс Bitmap(переменная BM). Проверяется, было введено название входного файла, если нет, то выводится ошибка. Затем считывается файл с изображением. Объявляется еще один оператор switch, который по переменной mode вызывает нужные функции, проверяя введённые данные перед передачей функциями из collectoprover.cpp в функции обработки. После обработки изображение записывается в выходной файл. Если название не было введено, то файл будет называться “out.bmp”.

3 Работа с BMP-файлом

3.1 Заголовочный файл bmp_class.h

В файле bmp_class.h сначала проверяется нет ли повторных подключений с помощью директив ifndef, endif. Подключаются другие заголовочные файлы, Объявляется класс Bitmap, в котором в публичном модификаторе доступа записываются прототипы функций для работы с BMP файлом, а в приватном – поля bmfh, bmif – заголовки, H и W – высота и ширина, pixels двумерный вектор для хранения пикселей.

3.2 Функции для работы с классом в bmp_class.cpp

В этом файле написаны функции для считывания, вывода и обработки BMP.

read_bmp – функция для считывания файла с изображением. В ней сначала открывается файл, присутствует обработка ошибок – предполагается, что пользователь мог забыть дописать расширение, поэтому если файл не открылся, то программа попытается открыть файл по названию добавленным расширением “.bmp”. Если файл всё равно не откроется, программа выведет ошибку и завершит работу. Далее идет чтение заголовка файла с помощью функции fread(будет произведено корректное чтение файла, так как было настроено нужное выравнивание), также считывается заголовок изображения. После идет проверка данных заголовка, как было сказано по условию – нужно проверять формат(4D42/424D), количество бит на пиксель(24), отсутствие сжатия. Сохраняются значения высоты и длины в поля класса. Объявляется двумерный вектор нужных размеров, куда будут сохранены значения пикселей. После проходит считывание в вектор, также учитываются лишние байты на конце строки, если они есть, иначе некоторые изображения будут некорректными. Закрывается файл, получаемый от пользователя.

write_bmp – функция для записи изображения. Изначально в ней записывается информация из заголовков, а уже после происходит запись вектора пикселей. Созданный файл с изображением закрывается.

print_file_header и print_info_header – функции, вызываемые для получения информации из заголовков. Выводится каждое поле структуры заголовков, вместе с его названием, которое поможет пользователю быстрее узнать нужную информацию.

get_height и get_width – функции, дающие доступ к полям длины и ширины. Возвращают их значения.

rgbfilter – функция первого задания – изменить компоненту каждого пикселя на передаваемое значение. Данные, которые принимает функция заранее проверены. С помощью оператора switch и названия компоненты component_name определяется, какой цвет нужно изменить. После в каждом case для красного, зеленого и синего изменяется соответствующая компонента каждого пикселя изображения на переданное значение.

pix_puter – вспомогательная функция, изменяющая значения пикселя. На вход принимает координаты(x, y) и цвет, и если значения x и y от нуля до ширины/длины, то цвет пикселя меняется на переданный.

fill_circle – вспомогательная функция, рисующая круг. Объявляются переменные сохраняющие минимальные и максимальные значения x и y, затем по формуле Пифагора определяется, принадлежит ли точка площади круга – в таком случае меняется цвет пикселя.

line - вспомогательная функция, рисующая линию. Здесь используется модифицированная версия алгоритма Брезенхема. В начале он вычисляет разницу между начальными и конечными точками линии по осям X и Y, а также определяет знаки направлений изменения координат X и Y. В цикле происходит перебор точек на линии, начиная с начальной точки и заканчивая конечной точкой. В точке, где должен ставиться пиксель, используется функция fill_circle, которая ставит круг(thickness / 2, так как функция запрашивает радиус), чтобы придать линии нужную толщину. Алгоритм вычисляет ошибку (error) и, в зависимости от ее значения, изменяет текущее положение точки по X и/или Y. Таким образом, выполнение цикла приводит к пошаговому изменению текущей точки в направлении конечной точки линии, с учетом толщины линии. Когда алгоритм достигает конечной точки, линия полностью нарисована.

square – функция второго задания – нарисовать квадрат по координатам, длине стороны, толщине, цвету границ, цвету заливки(по желанию). Все получаемые данные проверены. Если присутствовал флаг заливки, то сначала рисуется квадрат без границ(заливка), определяются координаты границ, а рисуются они с помощью функции line.

part_taker – вспомогательная функция, позволяющая сохранить прямоугольную область, используя координаты. Сначала определяется длина и ширина, затем создается двумерный вектор part, в котором будут сохранены пиксели области. Далее идет запись пикселей двумя циклами for ограниченными координатами. Функция возвращает вектор.

part_taker – вспомогательная функция, позволяющая разместить сохраненную в векторе область. Функция размещает область по координатам левого верхнего угла, заменяя пиксели предыдущей области на пиксели текущей.

exchange – функция третьего задания – сделать определенную смену фрагментов области, ограниченной координатами. Передаваемые параметры проверяются до их получения в функцию. Сначала проверяется, есть ли ошибки в координатах(выходит за границы изображения). Проверяется каждое значение, если есть выводится ошибки, и программа завершает работу. Также просматривается возможность беспорядка в координатах, когда левая больше правой или нижняя больше верхней. Выводится предупреждение, выполняется смена перепутанных координат. Далее под каждый фрагмент объявляется двумерный вектор. Определяется длина и ширина фрагмента. Объявляется словарь, в котором сохраняются координаты фрагментов. Чтобы получить значение нужной координаты, нужно сначала обратиться к части(сокращаются по первым буквам направления, например lu – left up, rd – right down), а затем уже к направлению первым символом направления. Далее с помощью функции part_taker берется каждая часть из изображения, а уже после с помощью оператора switch и переданного значения exchange_type определяется способ смены фрагментов области(Возможные варианты: по часовой, против часовой, диагональный). Фрагменты вставляются с помощью функции part_placer. Например, диагональный: левая верхняя часть и правая нижняя меняются местами, как и левая нижняя с правой верхней.

freq_color – функция четвертого задания – заменить самый часто встречаемый цвет на введенный пользователем. Объявляется словарь, в котором ключами будет значения цвета пикселя через точку, а значениями – их количество. Функция двумя циклами проходит по всему изображению, проверяя, есть ли строка с значениями пикселя среди ключей. Если нет, то появляется новая пара в словаре - цвет пикселя и 1, иначе к количеству пикселей добавляется 1. Далее совершается проход по словарю с целью

определить самый часто встречаемый пиксель. Если найдется пиксель, которого в изображении больше, то в переменную mx_str_color сохранится его значения. После того, как значение нашлось, строка со значениями цветов пикселей превращается в структуру Rgb, с помощью функции color_prove(о которой будет сказано дальше), а затем функция снова проходит по всему изображению, сравнивая все пиксели с найденным часто встречаемым, и если это он, то пиксель будет заменён на новый, переданный пользователем.

4 Функции для проверки значений и справка

4.1 Заголовочный файл collectoprover.h

В файле collectoprover.h сначала проверяется нет ли повторных подключений с помощью директив ifndef, endif. Здесь записаны прототипы функций, проверяющих корректность значений, передаваемых в функции для работы с BMP. Здесь также присутствует прототип функции, которая выводит справку.

4.2 Функции для проверки значений в collectoprover.cpp

int_prove – функция для проверки целочисленного неотрицательного значения. Сюда передаются параметры значения, которое нужно проверить, а также место, откуда была вызвана функция, чтобы можно было разобраться где/в чем ошибка. Сначала функция проверяет, пустая строка или нет. Если пустая – будет выведена ошибка и её причина, программа завершит работу. Далее циклом for проходится вся строка, и если в ней находится лишние символы, то они будут исключены и сохранены в строку для дальнейшего вывода предупреждения. Символы являющие числами будут сохранены в новую строку. Если хоть один символ был исключен, то будет выведено предупреждение, которое скажет в каком значении и какие символы были

исключены. Значение превращается в целочисленное с помощью функции stoi и возвращается.

color_value_prove – функция для проверки значений цвета. Эта функция использует int_prove, чтобы проверить значение на наличие лишних символов, а после проверяет, принадлежит ли оно диапазону от 0 до 255(включая). Если нет – программа завершает работу с выводом ошибки, иначе значение цвета возвращается.

color_prove – функция, которая проверяет значения цвета, которое подается в формате “red.green.blue”, и сохраняет их в структуру Rgb. Сначала проверяется, было ли введено значение(вывод ошибки, если не введено). С помощью цикла for, функция проходит, проверяя каждый символ, если она находит число – сохраняет его в строке color_val, в которой будут сохраняться значения цвета. Если находит точку, значение цвета проверяется функцией color_value_prove и записывается по порядку каждое значение пикселя в структуру. Запись определяется с помощью оператора switch и переменной mode(увеличивается на 1 при нахождении точки). В первый раз – значение красного, второй раз – значение зеленого, третий раз – значение синего и возвращение структуры Rgb. Если строка закончилась, но функция ничего не вернула – произойдет вывод ошибки: недостаточно аргументов цвета пикселей.

component_name_prove – функция проверяет название цвета – “red”, “green” или “blue”. Сначала понижает в строке comp регистр всех символов, а затем проверяет, соответствует ли строка comp возможным вариантам цвета. Если соответствует – то возвращает значение, иначе – вывод ошибки.

coord_prove – проверяет введенные координаты, исключает лишние символы и возвращает вектор с целочисленными координатами. Функция проходит циклом по каждому символу, проверяя чем он является. Если цифра – то сохраняется в строк coordinate, если точка, значит значение сохраняется в вектор. В какую ячейку вектора записать определяет cnt – количество координат. Если две координаты записаны(cnt = 2), то координата по y(индекс 1) переворачивается, так как в bmp ось y растет вверх, а по условию задания ось

у растет вниз. Если строка закончилась, а значение не было возвращено, программа завершает работу с ошибкой – нет или не хватает аргументов.

exchange_type_prove – проверяет корректность названия способа смены частей. Работает также, как и component_name_prove. Проверяет, пустая строка или нет(вывод ошибки предусмотрен), понижает регистр каждого символа, сравнивает с наименованиями возможных способов смены фрагментов частей(“clockwise”, “counterclockwise”, “diagonals”), если способ введенный пользователем существует, то возвращается первый по индексу символ способа обмена(так как нулевой по индексу у “clockwise” и “counterclockwise” одинаковый). Иначе – выводится ошибка.

4.3 Справка

Сначала определяются макросы для выделения текста цветом:

RESET – возвращение к стандартному выводу.

MAIN_TEXT – главный заголовок – повышение яркости(1), подчеркивание(4), циановый цвет(36).

SECOND_TEXT – обычный заголовок - повышение яркости(1), синий цвет(34).

FLAG_COLOR – цвет для флагов – повышение яркости(1), зеленый цвет(32).

DESCRIPTION – цвет для описания флагов – повышение яркости(1), цвет маджента(35)

Далее идет вывод флагов с их описанием. Перед текстом который нужно раскрасить идет макрос, затем текст, а после макрос RESET, чтобы вернуться к стандартному выводу.

5 Сборка программы, Makefile.

Программа разделена на цели:

- all – выполнение цели main и clean.

- main – сборка основной программы, получение исполняемого файла, результат целей main.o, collectoprover.o, bmp_class.o подключен здесь
- main.o – получение объектного файла main.c, заголовки my_lib.h, collectoprover.h, bmp_class.h
- bmp_class.o – получение объектного файла bmp_class.c, заголовки bmp_class.h, my_lib.h, collectoprover.h
- collectoprover.o – получение объектного файла collectoprover.c, заголовок my_lib.h подключен.

clean – удаление всех объектных файлов

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы реализованы функции для считывания, сбора информации, вывода и обработки изображения. Программа выполняет действия над изображениями, опираясь на указания пользователя в командной строке, все это реализовано с использованием библиотеки getopt.h. Доступны такие функции обработки, как – фильтр Rgb-компоненты, рисование квадрата, смены фрагментов части изображения и замены самого часто встречаемого цвета на новый. Написан Makefile, компилирующий и компонующий файлы исходного кода. Можно сделать вывод о соответствии полученного результата поставленной цели.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Брайан Керниган, Деннис Ритчи — Язык программирования Си.

2. Курс "Программирование на Си. Практические задания. Второй семестр" // moevm. URL: https://e.moevm.info/course/view.php?id=18. (дата обращения 29.04.2024)

3. Описание функций языка C/C++ // Всё о Hi-Tech. URL: http://all-ht.ru/inf/prog/c/func/index.html. (дата обращения 30.04.2024)

4. Описание функций языка C/C++ // MANPAGES.ORG. URL: https://ru.manpages.org/wcschr/3. (дата обращения 30.04.2024)

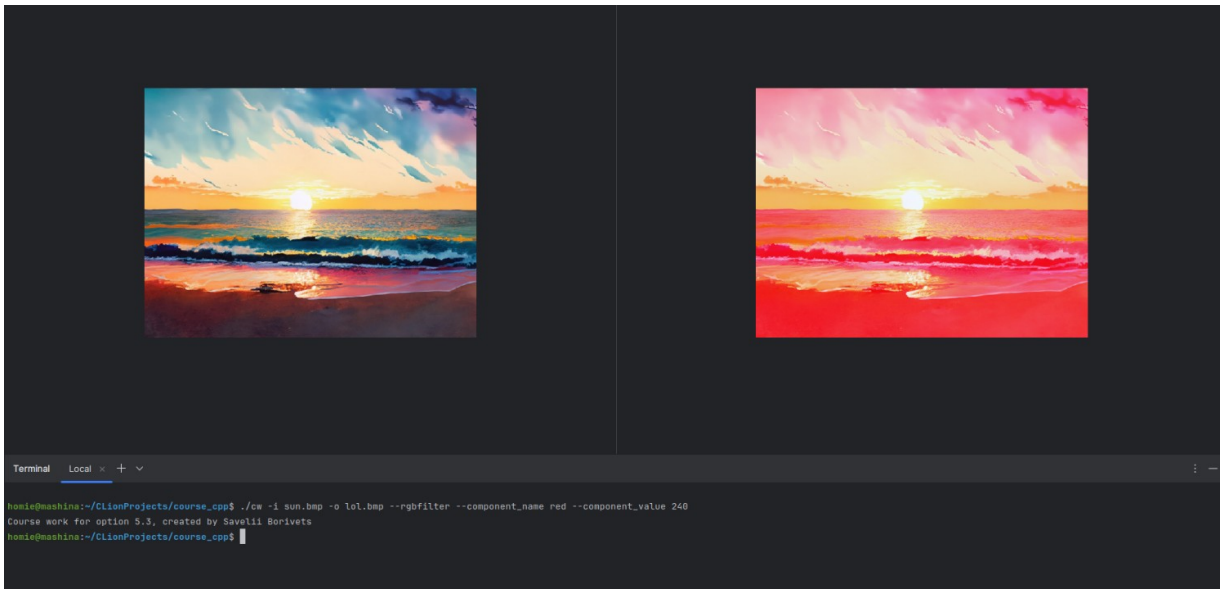
5. БАЗОВЫЕ СВЕДЕНИЯ К ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ». ВТОРОЙ СЕМЕСТР. // moevm. URL: https://se.moevm.info/lib/exe/fetch.php/courses:programming:programming_cw_metoda_2nd_course_last_ver.pdf.pdf. (дата обращения 30.04.2024)

ПРИЛОЖЕНИЕ А
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

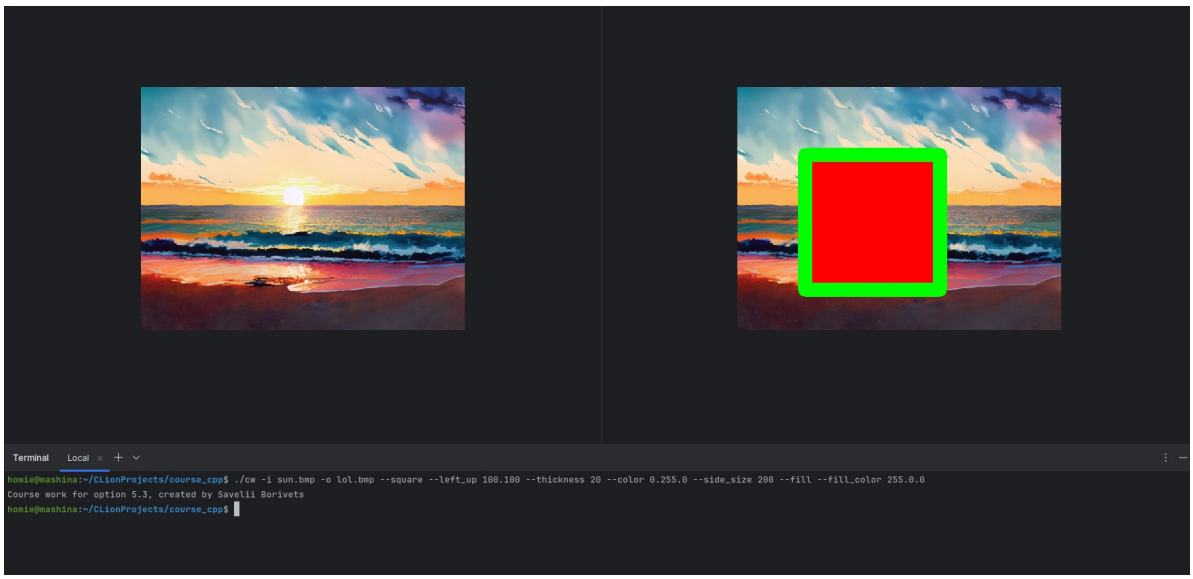
Пример 1: Вывод справки с помощью флага –help.

```
homie@mashina:~/CLionProjects/course_cpp$ ./cw --help
Course work for option 5.3, created by Savelii Borivets
ВАС ПРИВЕТСТВУЕТ СПРАВОЧНИК, КОТОРЫЙ РАССКАЖЕТ О РАБОТЕ ПРОГРАММЫ

ФЛАГИ СПРАВКИ, ВВОДА, ВЫВОДА, ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ

--help/-h - вызов справочника. Все остальные введённые аргументы будут проигнорированы.
--input/-i - название входного файла формата BMP. Если файла с таким названием не существует - будет выведена ошибка.
--output/-o - название выходного файла формата BMP. Если флаг отсутствует, файл будет назван out.bmp.
--info/-I - вывод подробной информации о введённом BMP файле.

ФЛАГИ ДЛЯ ИЗМЕНЕНИЯ ИЗОБРАЖЕНИЯ

--rgbfilter/-R - фильтр rgb-компонент. Устанавливает значение заданной компоненты в диапазоне от 0 до 255 для всего изображения.
Обязательные флаги:
--component_name/-n - компонента, которую необходимо изменить. Возможные значения: 'red', 'green', 'blue'.
--component_value/-v - значение компоненты в диапазоне от 0 до 255.

--square/-S - рисование квадрата по заданным координатам левого верхнего угла, размером стороны, толщине и цвету линий. Также можно выполнить заливку внутренней области квадрата.
Обязательные флаги:
--left_up/-l - координаты левого верхнего угла квадрата в формате 'left.up', где left - координата по x, up - координата по y.
--side_size/-s - размер стороны квадрата. Принимается значение больше 0. Отрицательные значения становятся положительными.
--thickness/-t - толщина линий квадрата. Принимается значение больше 0. Отрицательные значения становятся положительными.
--color/-C - цвет границ квадрата. Цвет задается строкой 'red.green.blue' - значения от 0 до 255 через точку.
Дополнительные флаги:
--fill/-f - флаг заливки. При его присутствии выполняется заливка.
--fill_color/-c - цвет заливки. вет задается строкой 'red.green.blue' - значения от 0 до 255 через точку. Если флаг --fill/-f отсутствует - заливка не выполняется.

--exchange/-E - смена 4 фрагментов области.
Обязательные флаги:
--left_up/-l - координаты левого верхнего угла области в формате 'left.up', где left - координата по x, up - координата по y.
--right_down/-r - координаты правого нижнего угла области в формате 'right.down', где right - координата по x, down - координата по y.
--exchange_type/-e - способы обмена частей. Возможные варианты: clockwise(по часовой), counterclockwise(против часовой), diagonals(по диагонали)

--freq_color/-F - находит самый часто встречаемый цвет и заменяет его на другой заданный цвет.
Обязательные флаги:
--color/-C - новый цвет. Цвет задается строкой 'red.green.blue' - значения от 0 до 255 через точку.
```

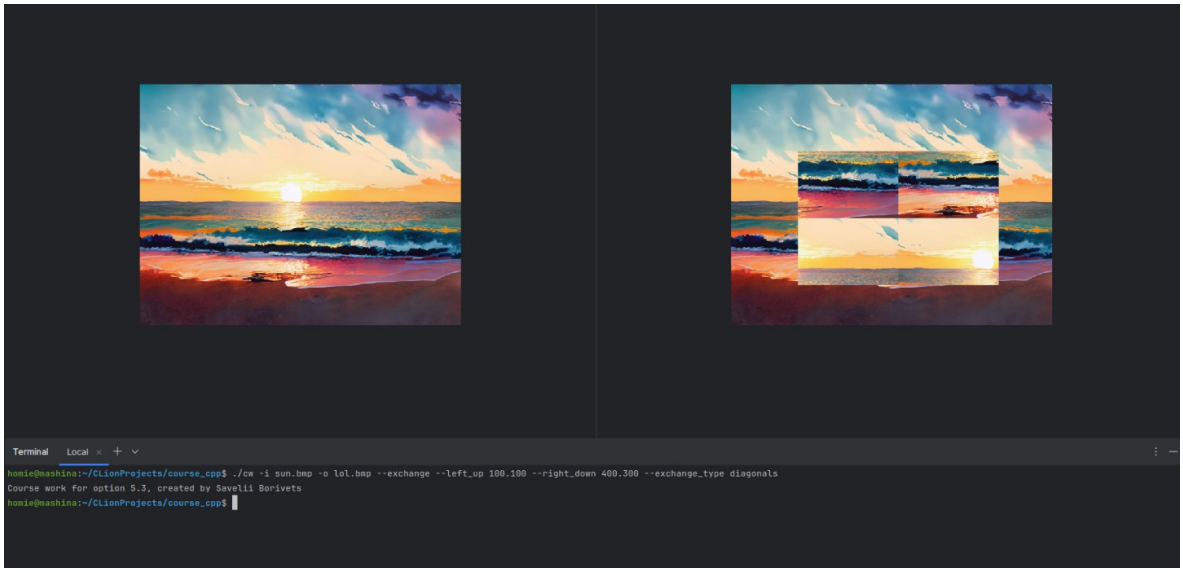
Пример 2: Работа функции rgbfilter. С помощью CLI передаются название цвета и его значения.



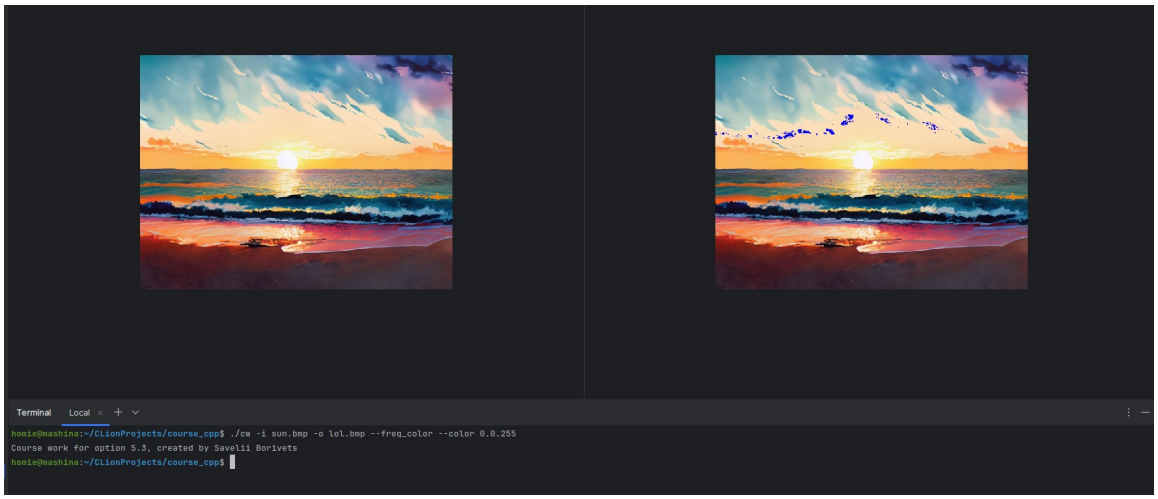
Пример 3: Работа функции square. Программе передаются значения размера толщины, длины стороны, координаты верхнего левого угла и выполняется заливка, так как указан флаг fill и цвет заливки.



Пример 4: Работа функции exchange. Проведено диагональное смещение фрагментов части изображения по нужным координатам.



Пример 5: Работа функции freq_color. Проведена замена самого частого цвета изображения на цвет, который подается пользователем.



Обработка ошибок:

Пример 1: Подача некорректного значения цвета пикселя. Программа завершает работу.

```
homie@mashina:~/CLionProjects/course_cpp$ ./cw -i sun.bmp -o lol.bmp --freq_color --color 0.0.400
Course work for option 5.3, created by Savelii Borivets
Ошибка: Значение цвета пикселя должно быть в диапазоне от 0 до 255. Было введено значение - 400
homie@mashina:~/CLionProjects/course_cpp$
```

Пример 2: Правая и нижняя координаты выходят за границы изображения. Программа завершает работу.

```
homie@mashina:~/CLionProjects/course_cpp$ ./cw -i sun.bmp -o lol.bmp --exchange --left_up 100.100 --right_down 481.361 --exchange_type diagonals
Course work for option 5.3, created by Savelii Borivets
Ошибка: Некорректная правая координата(Функция Exchange)
Ошибка: Некорректная нижняя координата(Функция Exchange)
homie@mashina:~/CLionProjects/course_cpp$
```

Пример 3: Координаты левого верхнего и правого нижнего угла – перепутаны. Программа продолжает работу, выполнив смену координат.

```
homie@mashina:~/CLionProjects/course_cpp$ ./cw -i sun.bmp -o lol.bmp --exchange --left_up 100.100 --right_down 0.0 --exchange_type diagonals
Course work for option 5.3, created by Savelii Borivets
Предупреждение: Смена координат - при вводе левая координата больше правой
Предупреждение: Смена координат - при вводе нижняя координата больше левой
homie@mashina:~/CLionProjects/course_cpp$
```

Пример 4: Некорректный способ смены фрагментов части изображения. Доступны варианты: diagonals, clockwise, counterclockwise. Программа завершает работу.

```
homie@mashina:~/CLionProjects/course_cpp$ ./cw -i sun.bmp -o lol.bmp --exchange --left_up 100.100 --right_down 200.200 --exchange_type idontknow
Course work for option 5.3, created by Savelii Borivets
Ошибка: Некорректный способ обмена частей - idontknow
homie@mashina:~/CLionProjects/course_cpp$
```

Пример 5: Ввод несуществующего флага. Программа завершает работу.

```
homie@mashina:~/CLionProjects/course_cpp$ ./cw -i sun.bmp -o lol.bmp --madesmth
Course work for option 5.3, created by Savelii Borivets
./cw: unrecognized option '--madesmth'
Ошибка: Проблема с флагами или аргументами. Подробнее написано выше
homie@mashina:~/CLionProjects/course_cpp$
```

Пример 6: Отсутствует имя входного файла.

```
homie@mashina:~/CLionProjects/course_cpp$ ./cw -o lol.bmp --exchange --left_up 100.100 --right_down 200.200 --exchange_type clockwise
Course work for option 5.3, created by Savelii Borivets
Ошибка: Имя файла не было введено.
homie@mashina:~/CLionProjects/course_cpp$
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: my_lib.h

```
#ifndef COURSE_MY_LIB_H
#define COURSE_MY_LIB_H

#include <iostream>
#include <cstdio>
#include <cstring>
#include <getopt.h>
#include <stdexcept>
#include <vector>
#include <map>
#include <clocale>

using namespace std;

#define ARG_ERR 40
#define NO_ARG_ERR 41
#define INCOR_TYPE_ERR 42

#pragma pack(push, 1)
typedef struct {
    unsigned short signature;
    unsigned int filesize;
    unsigned short reserved1;
    unsigned short reserved2;
    unsigned int pixelArrOffset;
} BitmapFileHeader;

typedef struct {
    unsigned int headerSize;
    unsigned int width;
    unsigned int height;
    unsigned short planes;
    unsigned short bitsPerPixel;
    unsigned int compression;
    unsigned int imageSize;
    unsigned int xPixelsPerMeter;
    unsigned int yPixelsPerMeter;
    unsigned int colorsInColorTable;
    unsigned int importantColorCount;
} BitmapInfoHeader;

typedef struct {
    unsigned char b;
    unsigned char g;
    unsigned char r;
} Rgb;
#pragma pack(pop)

#endif //COURSE_MY_LIB_H
```

Название файла: main.c

```
#include "my_lib.h"
```

```
#include "collectoprover.h"
#include "bmp_class.h"

int main(int argc, char **argv) {
    cout << "Course work for option 5.3, created by Savelii Borivets" <<
endl;
    setlocale(LC_ALL, "ru-RU");
    int opt;
    static struct option long_options[] = {
            {"rgbfilter",       no_argument,       0, 'R'},
            {"component_name",  required_argument, 0, 'n'},
            {"component_value", required_argument, 0, 'v'},
            {"square",          no_argument,       0, 'S'},
            {"left_up",         required_argument, 0, 'l'},
            {"side_size",       required_argument, 0, 's'},
            {"thickness",       required_argument, 0, 't'},
            {"color",           required_argument, 0, 'C'},
            {"fill",            no_argument,       0, 'f'},
            {"fill_color",      required_argument, 0, 'c'},
            {"exchange",        no_argument,       0, 'E'},
            {"right_down",      required_argument, 0, 'r'},
            {"exchange_type",   required_argument, 0, 'e'},
            {"freq_color",      no_argument,       0, 'F'},
            {"help",            no_argument,       0, 'h'},
            {"info",            no_argument,       0, 'I'},
            {"input",           required_argument, 0, 'i'},
            {"output",          required_argument, 0, 'o'},
            {0, 0,                                 0, 0}
    };
    map<char, map<string, string>> commander {
            {'R', {
                {"component_name", ""},
                {"component_value", ""}}},
            {'S', {
                {"left_up", ""},
                {"side_size", ""},
                {"thinkness", ""},
                {"color", ""},
                {"fill", ""},
                {"fill_color", ""}}},
            {'E', {
                {"left_up", ""},
                {"right_down", ""},
                {"exchange_type", ""}}},
            {'F', {
                {"color", ""}}}
    };
    string file_name_in;
    string file_name_out;
    char mode;
    //Собиратель данных
    if (argc <= 1) {
        help();
        return 0;
    }
                    while     ((opt     =     getopt_long(argc,     argv,
"Rn:v:Sl:s:t:C:fc:Er:e:FhIi:o:", long_options, nullptr)) != -1) {
        switch (opt) {
```

```
case 'R':
    mode = 'R';
    break;
case 'n':
    commander['R']["component_name"] = optarg;
    break;
case 'v':
    commander['R']["component_value"] = optarg;
    break;
case 'S':
    mode = 'S';
    break;
case 'l':
    commander['S']["left_up"] = optarg;
    commander['E']["left_up"] = optarg;
    break;
case 's':
    commander['S']["side_size"] = optarg;
    break;
case 't':
    commander['S']["thickness"] = optarg;
    break;
case 'C':
    commander['S']["color"] = optarg;
    commander['F']["color"] = optarg;
    break;
case 'f':
    commander['S']["fill"] = "true";
    break;
case 'c':
    commander['S']["fill_color"] = optarg;
    break;
case 'E':
    mode = 'E';
    break;
case 'r':
    commander['E']["right_down"] = optarg;
    break;
case 'e':
    commander['E']["exchange_type"] = optarg;
    break;
case 'F':
    mode = 'F';
    break;
case 'h':
    help();
    return 0;
case 'I':
    mode = 'I';
    break;
case 'i':
    file_name_in = optarg;
    break;
case 'o':
    file_name_out = optarg;
    break;
default:
```

```
                    cout << "Ошибка: Проблема с флагами или аргументами.
Подробнее написано выше" << endl;
                exit(ARG_ERR);
        }
    }
    //Открытие и чтение
    Bitmap BM;
    if (file_name_in.empty()) {
        cout << "Ошибка: Имя файла не было введено." << endl;
        exit(ARG_ERR);
    }
    BM.read_bmp(file_name_in);
    //Работа с файлом
    switch (mode) {
        case 'R':{
             string component_name = component_name_prove(commander[mode]
["component_name"]);
                                      unsigned   char   component_value   =
color_value_prove(commander[mode]["component_value"]);
            BM.rgbfilter(component_name, component_value);
            break;}
        case 'S':{
                 vector<int> coordinates = coord_prove(commander[mode]
["left_up"], BM.get_height());
            int left = coordinates[0], up = coordinates[1];
                 int side_size = int_prove(commander[mode]["side_size"],
"Сторона квадрата");
                 int thickness = int_prove(commander[mode]["thickness"],
"Толщина стороны квадрата");
            Rgb color = color_prove(commander[mode]["color"]);
            bool fill = false;
            Rgb fill_color;
            if (commander['S']["fill"] == "true") {
                fill = true;
                fill_color = color_prove(commander[mode]["fill_color"]);
            }
                 BM.square(left, up, side_size, thickness, color, fill,
fill_color);
            break;}
        case 'E':{
                 vector<int> coordinates1 = coord_prove(commander[mode]
["left_up"], BM.get_height());
                 vector<int> coordinates2 = coord_prove(commander[mode]
["right_down"], BM.get_height());
              int left = coordinates1[0], up = coordinates1[1], right =
coordinates2[0], down = coordinates2[1];
                                   BM.exchange(left,   up,   right,   down,
exchange_type_prove(commander[mode]["exchange_type"]));
            break;}
        case 'F':{
            Rgb color = color_prove(commander[mode]["color"]);
            BM.freq_color(color);
            break;}
        case 'I':
            BM.print_file_header();
            BM.print_info_header();
            return 0;
    }
```

```
    //Запись измененного файла
    if (file_name_out.empty()) {
        file_name_out = "out.bmp";
    }
    BM.write_bmp(file_name_out);
    return 0;
}
```

Название файла: bmp_class.cpp

```
#include "bmp_class.h"

//Чтение Bitmap
void Bitmap::read_bmp(string filename) {
    FILE* f = fopen(filename.c_str(), "rb");
    if (!f) {
        string filename_add_bmp = filename + ".bmp";
        f = fopen(filename_add_bmp.c_str(), "rb");
        if (!f) {
                    cout << "Ошибка: Файл " << filename << " или " <<
filename_add_bmp << " не найден. Проверьте название файла." << endl;
            exit(ARG_ERR);
        }
    }
    //Чтение заголовка файла
    fread(&bmfh, 1, sizeof(BitmapFileHeader), f);
    if (bmfh.signature != 19778 && bmfh.signature != 16973) {
         cout << "Ошибка: Версия BMP " << hex << bmfh.signature << " не
поддерживается. Поддерживаемые форматы 4D42/424D" << endl;
        exit(INCOR_TYPE_ERR);
    }
    //Чтение заголовка изображения
    fread(&bmif, 1, sizeof(BitmapInfoHeader), f);
    //Проверка параметров
    if (bmif.bitsPerPixel != 24) {
        cout << "Ошибка: Количество бит на пиксель должно быть равно 24.
Текущее значение: " << bmif.bitsPerPixel << endl;
        exit(INCOR_TYPE_ERR);
    } else if (bmif.compression != 0) {
         cout << "Ошибка: Изображение без сжатия(0). Текущее значение: "
<< bmif.compression << endl;
        exit(INCOR_TYPE_ERR);
    }
    H = bmif.height;
    W = bmif.width;
    pixels.resize(H, vector<Rgb>(W));
    for (size_t y = 0; y < H; y++) {
        fread(pixels[y].data(), sizeof(Rgb), W, f);
        // Считываем лишние байты на конце строки, если они есть
        int padding = (4 - W * 3 % 4) % 4;
        fseek(f, padding, SEEK_CUR);
    }
    fclose(f);
}

//Запись Bitmap
void Bitmap::write_bmp(string filename){
    FILE *ff = fopen(filename.c_str(), "wb");
```

```
    fwrite(&bmfh, 1, sizeof(BitmapFileHeader), ff);
    fwrite(&bmif, 1, sizeof(BitmapInfoHeader), ff);
    H = bmif.height;
    W = bmif.width;
    for(size_t y = 0; y < H; y++){
        fwrite(pixels[y].data(), 1, W * sizeof(Rgb) + (4 - W * 3 % 4) %
4, ff);
    }
    fclose(ff);
}

//Вывод заголовка файла
void Bitmap::print_file_header(){
    BitmapFileHeader header = bmfh;
                                printf("--------------------===Заголовок
файла===--------------------\n");
        printf("Сигнатура файла BMP:\t%x (%hu)\n", header.signature,
header.signature);
        printf("Размер файла в байтах:\t%x (%u)\n", header.filesize,
header.filesize);
      printf("Зарезервированное поле 1:\t%x (%hu)\n", header.reserved1,
header.reserved1);
      printf("Зарезервированное поле 2:\t%x (%hu)\n", header.reserved2,
header.reserved2);
       printf("Местонахождение данных растрового массива:\t%x (%u)\n",
header.pixelArrOffset, header.pixelArrOffset);
}

//Вывод заголовка изображения
void Bitmap::print_info_header(){
    BitmapInfoHeader header = bmif;
                                    printf("-----------------===Заголовок
изображения===-----------------\n");
          printf("Размер данной структуры в байтах:\t%x (%u)\n",
header.headerSize, header.headerSize);
    printf("Ширина: \t%x (%u)\n", header.width, header.width);
    printf("Высота: \t%x (%u)\n", header.height, header.height);
      printf("Число цветовых плоскостей: \t%x (%hu)\n", header.planes,
header.planes);
         printf("Бит на пиксель:\t%x (%hu)\n", header.bitsPerPixel,
header.bitsPerPixel);
            printf("Метод сжатия:\t%x (%u)\n", header.compression,
header.compression);
          printf("Размер изображения:\t%x (%u)\n", header.imageSize,
header.imageSize);
                  printf("Горизонтальное разрешение:\t%x (%u)\n",
header.xPixelsPerMeter, header.xPixelsPerMeter);
    printf("Вертикальное разрешение:\t%x (%u)\n", header.yPixelsPerMeter,
header.yPixelsPerMeter);
                printf("Число цветов изображения:\t%x (%u)\n",
header.colorsInColorTable, header.colorsInColorTable);
                  printf("Число основных цветов:\t%x (%u)\n",
header.importantColorCount, header.importantColorCount);
}

unsigned int Bitmap::get_height() {
    return H;
}
```

```
unsigned int Bitmap::get_width() {
    return W;
}

//Функция rgbfilter
void    Bitmap::rgbfilter(string    component_name,    unsigned    char
component_value) {
    switch (component_name[0]) {
        case 'r':
            for (size_t y = 0; y < H; y++) {
                for (size_t x = 0; x < W; x++) {
                    pixels[y][x].r = component_value;
                }
            }
            break;
        case 'g':
            for (size_t y = 0; y < H; y++) {
                for (size_t x = 0; x < W; x++) {
                    pixels[y][x].g = component_value;
                }
            }
            break;
        case 'b':
            for (size_t y = 0; y < H; y++) {
                for (size_t x = 0; x < W; x++) {
                    pixels[y][x].b = component_value;
                }
            }
            break;
    }
}

void Bitmap::pix_puter(int x, int y, Rgb color) {
    if (0 <= x && x < W && 0 <= y && y < H) {
        pixels[y][x] = color;
    }
}

//Функция square
void Bitmap::square(int left, int up, int side_size, int thickness, Rgb
color, bool fill, Rgb fill_color) {
    if (side_size > 0 && fill) {
        for (int y = up; up - y < side_size; y--) {
            for(int x = left; x - left < side_size; x++) {
                pix_puter(x, y, fill_color);
            }
        }
    }
    int right = left + side_size, down = up - side_size;
    line(left, up, right, up, thickness, color);
    line(left, up, left, down, thickness, color);
    line(right, up, right, down, thickness, color);
    line(left, down, right, down, thickness, color);
}

//Функция для сохранения части картинки по координатам
```

```
vector<vector<Rgb>> Bitmap::part_taker(int left, int up, int right, int
down) {
    int height = up - down + 1;
    int width = right - left + 1;
    vector<vector<Rgb>> part(height, vector<Rgb>(width));
    for (int i = 0; down + i <= up; i++) {
        for(int j = 0; left + j <= right; j++) {
            part[i][j] = pixels[down + i][left + j];
        }
    }
    return part;
}

void Bitmap::part_placer(int left, int up, vector<vector<Rgb>> part) {
    for (int i = 0; i < part.size(); i++) {
        for (int j = 0; j < part[i].size(); j++) {
            pix_puter(left + j, up - i, part[part.size() - 1 - i][j]);
        }
    }
}

void Bitmap::exchange(int left, int up, int right, int down, char
exchange_type) {
    bool err = false;
    if (left < 0 || left >= W) {
        cout << "Ошибка: Некорректная левая координата(Функция Exchange)"
<< endl;
        err = true;
    }
    if (up < 0 || up >= H) {
            cout << "Ошибка: Некорректная верхняя координата(Функция
Exchange)" << endl;
        err = true;
    }
    if (right < 0 || right >= W) {
            cout << "Ошибка: Некорректная правая координата(Функция
Exchange)" << endl;
        err = true;
    }
    if (down < 0 || down >= H) {
            cout << "Ошибка: Некорректная нижняя координата(Функция
Exchange)" << endl;
        err = true;
    }
    if (err) {
        exit(ARG_ERR);
    }
    int buf;
    if (left > right) {
            cout << "Предупреждение: Смена координат - при вводе левая
координата больше правой" << endl;
        buf = left;
        left = right;
        right = buf;
    }
    if (down > up) {
           cout << "Предупреждение: Смена координат - при вводе нижняя
координата больше левой" << endl;
```

```
        buf = down;
        down = up;
        up = buf;
    }
    vector<vector<Rgb>> lu_part, ld_part, ru_part, rd_part;
    int part_height = (up - down) / 2 - 1, part_width = (right - left) /
2 - 1;
    map<string, map<char, int>> part_coord {
            {"lu", {
                    {'l', left},
                    {'u', up},
                    {'r', left + part_width},
                    {'d', up - part_height}
            }},
            {"ld", {
                    {'l', left},
                    {'u', up - part_height - 1},
                    {'r', left + part_width},
                    {'d', up - 2 * part_height - 1}
            }},
            {"ru", {
                    {'l', left + part_width + 1},
                    {'u', up},
                    {'r', left + 2 * part_width + 1},
                    {'d', up - part_height}
            }},
            {"rd", {
                    {'l', left + part_width + 1},
                    {'u', up - part_height - 1},
                    {'r', left + 2 * part_width + 1},
                    {'d', up - 2 * part_height - 1}
            }}
    };
      lu_part = part_taker(part_coord["lu"]['l'], part_coord["lu"]['u'],
part_coord["lu"]['r'], part_coord["lu"]['d']);//left up
      ld_part = part_taker(part_coord["ld"]['l'], part_coord["ld"]['u'],
part_coord["ld"]['r'], part_coord["ld"]['d']);//left down
      ru_part = part_taker(part_coord["ru"]['l'], part_coord["ru"]['u'],
part_coord["ru"]['r'], part_coord["ru"]['d']);//right up
      rd_part = part_taker(part_coord["rd"]['l'], part_coord["rd"]['u'],
part_coord["rd"]['r'], part_coord["rd"]['d']);//right down
    switch (exchange_type) {
        case 'l': //clockwise = 'l'
                part_placer(part_coord["lu"]['l'], part_coord["lu"]['u'],
ld_part);//new left up part
                part_placer(part_coord["ld"]['l'], part_coord["ld"]['u'],
rd_part);//new left down part
                part_placer(part_coord["ru"]['l'], part_coord["ru"]['u'],
lu_part);//new right up part
                part_placer(part_coord["rd"]['l'], part_coord["rd"]['u'],
ru_part);//new right down part
            break;
        case 'o'://counterclockwise - 'o'
                part_placer(part_coord["lu"]['l'], part_coord["lu"]['u'],
ru_part);//new left up part
                part_placer(part_coord["ld"]['l'], part_coord["ld"]['u'],
lu_part);//new left down part
```

```
                part_placer(part_coord["ru"]['l'], part_coord["ru"]['u'],
rd_part);//new right up part
                part_placer(part_coord["rd"]['l'], part_coord["rd"]['u'],
ld_part);//new right down part
            break;
        case 'i'://diagonals - 'i'
                part_placer(part_coord["lu"]['l'], part_coord["lu"]['u'],
rd_part);//new left up part
                part_placer(part_coord["ld"]['l'], part_coord["ld"]['u'],
ru_part);//new left down part
                part_placer(part_coord["ru"]['l'], part_coord["ru"]['u'],
ld_part);//new right up part
                part_placer(part_coord["rd"]['l'], part_coord["rd"]['u'],
lu_part);//new right down part
            break;
    }
}

void Bitmap::freq_color(Rgb new_color) {
    map<string, unsigned int> pix_counter;
    for (int i = 0; i < H; i++) {
        for (int j = 0; j < W; j++) {
            Rgb cur_pix = pixels[i][j];
                    string key_color = to_string(cur_pix.r) + '.' +
to_string(cur_pix.g) + '.' + to_string(cur_pix.b);
            if (pix_counter.count(key_color)) {
                pix_counter[key_color] += 1;
            } else {
                pix_counter[key_color] = 1;
            }
        }
    }
    string mx_str_color;
    unsigned int mx_cnt_color = 0;
    for (const auto& [key_color, cnt_color] : pix_counter) {
        if (cnt_color >= mx_cnt_color) {
            mx_str_color = key_color;
            mx_cnt_color = cnt_color;
        }
    }
    Rgb mx_rgb_color = color_prove(mx_str_color);
    for (int i = 0; i < H; i++) {
        for (int j = 0; j < W; j++) {
              if (pixels[i][j].r == mx_rgb_color.r && pixels[i][j].g ==
mx_rgb_color.g && pixels[i][j].b == mx_rgb_color.b) {
                pix_puter(j, i, new_color);
            }
        }
    }
}

void Bitmap::fill_circle(int Xmid, int Ymid, int r, Rgb color){
    int xmin = Xmid - r;
    int ymin = Ymid - r;
    int xmax = Xmid + r;
    int ymax = Ymid + r;
    for (int y = ymin; y <= ymax; y++){
        for (int x = xmin; x <= xmax; x++){
```

```
            if ((x - Xmid) * (x - Xmid) + (y - Ymid) * (y - Ymid) <= r *
r){
                pix_puter(x, y, color);
            }
        }
    }
}

void Bitmap::line(int x0, int y0, int x1, int y1, int thickness,  Rgb
color) {
    const int deltaX = abs(x1 - x0);
    const int deltaY = abs(y1 - y0);
    const int signX = x0 < x1 ? 1 : -1;
    const int signY = y0 < y1 ? 1 : -1;
    int error = deltaX - deltaY;
    fill_circle(x1, y1, (thickness) / 2, color);
    while (x0 != x1 || y0 != y1)
    {
        fill_circle(x0, y0, (thickness) / 2, color);
        const int error2 = error * 2;
        if (error2 > -deltaY) {
            error -= deltaY;
            x0 += signX;
        }
        if (error2 < deltaX) {
            error += deltaX;
            y0 += signY;
        }
    }
}
```

Название файла: bmp_class.h

```
#ifndef COURSE_BMP_CLASS_H
#define COURSE_BMP_CLASS_H

#include "my_lib.h"
#include "collectoprover.h"

class Bitmap {
public:
    void read_bmp(string filename);
    void write_bmp(string filename);
    void print_file_header();
    void print_info_header();
    unsigned int get_height();
    unsigned int get_width();
    void rgbfilter(string component_name, unsigned char component_value);
    void pix_puter(int x, int y, Rgb color);
      void square(int left, int up, int side_size, int thickness, Rgb
color, bool fill, Rgb fill_color);
       vector<vector<Rgb>> part_taker(int left, int up, int right, int
down);
    void part_placer(int left, int up, vector<vector<Rgb>> part);
        void exchange(int left, int up, int right, int down, char
exchange_type);
    void freq_color(Rgb color);
    void fill_circle(int Xmid, int Ymid, int r, Rgb color);
```

```
    void line(int x0, int y0, int x1, int y1, int thickness, Rgb color);
private:
    BitmapFileHeader bmfh;
    BitmapInfoHeader bmif;
    unsigned int H;
    unsigned int W;
    vector<vector<Rgb>> pixels;
};


#endif //COURSE_BMP_CLASS_H
```

Название файла: collectoprover.cpp

```
#include "my_lib.h"

int int_prove(string value, string place) {
    if (value.size() == 0) {
        cout << "Ошибка: Не было введено - " << place << endl;
        exit(NO_ARG_ERR);
    }
    bool flag = false;
    string err_sym = "";
    string new_value = "";
    for (int i = 0; i < value.size(); i++) {
        if (!isdigit(value[i])) {
               if (strchr(err_sym.c_str(), value[i]) == NULL) err_sym +=
value[i];
            flag = true;
        } else {
            new_value += value[i];
        }
    }
    if (flag) {
        cout << "Возможная ошибка устранена: Лишние символы [" << err_sym
<< "] в значении - " << place << endl;
    }
    return stoi(new_value);
}

unsigned char color_value_prove(string value) {
    int pix_val = int_prove(value, "Значение цвета пикселя");
    if (0 <= pix_val && pix_val <= 255) return pix_val;
    cout << "Ошибка: Значение цвета пикселя должно быть в диапазоне от 0
до 255. Было введено значение - " << pix_val << endl;
    exit(ARG_ERR);
}

Rgb color_prove(string color) {
    if (color.size() == 0) {
        cout << "Ошибка: Значение цвета не было введено" << endl;
        exit(NO_ARG_ERR);
    }
    Rgb new_color;
    color += '.';
    string color_val = "";
    int mode = 0; //0 - red, 1 - green, 2 - blue
    for (int i = 0; i < color.size(); i++) {
```

```
        if (isdigit(color[i])) {
            color_val += color[i];
        } else if (color[i] == '.') {
            switch (mode) {
                case 0:
                    new_color.r = color_value_prove(color_val);
                    break;
                case 1:
                    new_color.g = color_value_prove(color_val);
                    break;
                case 2:
                    new_color.b = color_value_prove(color_val);
                    return new_color;
            }
            color_val = "";
            mode++;
        }
    }
    cout << "Ошибка: Недостаточно аргументов цвета пикселей" << endl;
    exit(NO_ARG_ERR);
}

string component_name_prove(string comp) {
    for (int i = 0; i < comp.size(); i++) {
        comp[i] = tolower(comp[i]);
    }
    if (comp == "red" || comp == "green" || comp == "blue") {
        return comp;
    }
    cout << "Ошибка: Некорректный цвет - " << comp << endl;
    exit(ARG_ERR);
}

vector<int> coord_prove(string coord, int H) {
    coord += '.';
    vector<int> v_coord(2);
    int cnt = 0;
    string coordinate;
    for (int i = 0; i < coord.size(); i++) {
        if (isdigit(coord[i])) {
            coordinate += coord[i];
        } else if (coord[i] == '.') {
            v_coord[cnt] = int_prove(coordinate, "Значение координат.");
            cnt++;
            coordinate = "";
        }
        if (cnt == 2) {
            //Переворот второй координаты
            v_coord[1] = H - 1 - v_coord[1];
            return v_coord;
        }
    }
    cout << "Ошибка: Нет или не хватает значений координат.";
    exit(NO_ARG_ERR);
}

char exchange_type_prove(string exchange_type) {
    if (!exchange_type.size()) {
```

```
        cout << "Ошибка: Не введен способ обмена частей" << endl;
        exit(NO_ARG_ERR);
    }
    for (int i = 0; i < exchange_type.size(); i++) {
        exchange_type[i] = tolower(exchange_type[i]);
    }
        if (exchange_type == "clockwise" || exchange_type == "counterclockwise" || exchange_type == "diagonals") {
        return exchange_type[1];
    }
        cout << "Ошибка: Некорректный способ обмена частей - " << exchange_type << endl;
    exit(ARG_ERR);
}

#define RESET "\x1B[0m" //Обычный вывод в консоль
#define MAIN_TEXT "\x1B[1;4;36m" //Главный заголовок
#define SECOND_TEXT "\x1B[1;34m" //Заголовки
#define FLAG_COLOR "\x1B[1;32m" //Флаги
#define DESCRIPTION "\x1B[1;35m" //Описание функций

void help() {
    cout << MAIN_TEXT << "ВАС ПРИВЕТСТВУЕТ СПРАВОЧНИК, КОТОРЫЙ РАССКАЖЕТ О РАБОТЕ ПРОГРАММЫ" << RESET << endl << endl;

     cout << MAIN_TEXT << "ФЛАГИ СПРАВКИ, ВВОДА, ВЫВОДА, ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ" << RESET <<  endl << endl;

    cout << FLAG_COLOR << "--help/-h" << RESET;
    cout << DESCRIPTION << " - вызов справочника. Все остальные введённые аргументы будут проигнорированы." << RESET << endl;
    cout << FLAG_COLOR << "--input/-i" << RESET;
    cout << DESCRIPTION << " - название входного файла формата BMP. Если файла с таким названием не существует - будет выведена ошибка." << RESET << endl;
    cout << FLAG_COLOR << "--output/-o" << RESET;
    cout << DESCRIPTION << " - название выходного файла формата BMP. Если флаг отсутствует, файл будет назван out.bmp." << RESET << endl;
    cout << FLAG_COLOR << "--info/-I" << RESET;
    cout << DESCRIPTION << " - вывод подробной информации о введённом BMP файле." << RESET << endl << endl;

     cout << MAIN_TEXT << "ФЛАГИ ДЛЯ ИЗМЕНЕНИЯ ИЗОБРАЖЕНИЯ" << RESET << endl << endl;

    cout << FLAG_COLOR << "--rgbfilter/-R" << RESET;
      cout << DESCRIPTION << " - фильтр rgb-компонент. Устанавливает значение заданной компоненты в диапазоне от 0 до 255 для всего изображения." << RESET << endl;
    cout << SECOND_TEXT << "Обязательные флаги:" << RESET << endl;
    cout << FLAG_COLOR << "--component_name/-n" << RESET;
     cout << DESCRIPTION << " - компонента, которую необходимо изменить. Возможные значения: 'red', 'green', 'blue'." << RESET << endl;
    cout << FLAG_COLOR << "--component_value/-v" << RESET;
     cout << DESCRIPTION << " - значение компоненты в диапазоне от 0 до 255." << RESET << endl << endl;

    cout << FLAG_COLOR << "--square/-S" << RESET;
```

```
    cout << DESCRIPTION << " - рисование квадрата по заданным координатам левого верхнего угла, размером стороны, толщине и цвету линий. Также можно выполнить заливку внутренней области квадрата." << RESET << endl;
    cout << SECOND_TEXT << "Обязательные флаги:" << RESET << endl;
    cout << FLAG_COLOR << "--left_up/-l" << RESET;
    cout << DESCRIPTION << " - координаты левого верхнего угла квадрата в формате 'left.up', где left - координата по x, up - координата по y." << RESET << endl;
    cout << FLAG_COLOR << "--side_size/-s" << RESET;
      cout << DESCRIPTION << " - размер стороны квадрата. Принимается значение больше 0. Отрицательные значения становятся положительными." << RESET << endl;
    cout << FLAG_COLOR << "--thickness/-t" << RESET;
      cout << DESCRIPTION << " - толщина линий квадрата. Принимается значение больше 0. Отрицательные значения становятся положительными." << RESET << endl;
    cout << FLAG_COLOR << "--color/-C" << RESET;
      cout << DESCRIPTION << " - цвет границ квадрата. Цвет задается строкой 'red.green.blue' - значения от 0 до 255 через точку." << RESET << endl;
    cout << SECOND_TEXT << "Дополнительные флаги:" << RESET << endl;
    cout << FLAG_COLOR << "--fill/-f" << RESET;
       cout << DESCRIPTION << " - флаг заливки. При его присутствии выполняется заливка." << RESET << endl;
    cout << FLAG_COLOR << "--fill_color/-c" << RESET;
      cout << DESCRIPTION << " - цвет заливки. вет задается строкой 'red.green.blue' - значения от 0 до 255 через точку. Если флаг --fill/-f отсутствует - заливка не выполняется." << RESET << endl << endl;

    cout << FLAG_COLOR << "--exchange/-E" << RESET;
     cout << DESCRIPTION << " - смена 4 фрагментов области." << RESET << endl;
    cout << SECOND_TEXT << "Обязательные флаги:" << RESET << endl;
    cout << FLAG_COLOR << "--left_up/-l" << RESET;
     cout << DESCRIPTION << " - координаты левого верхнего угла области в формате 'left.up', где left - координата по x, up - координата по y." << RESET << endl;
    cout << FLAG_COLOR << "--right_down/-r" << RESET;
     cout << DESCRIPTION << " - координаты правого нижнего угла области в формате 'right.down', где right - координата по x, down - координата по y." << RESET << endl;
    cout << FLAG_COLOR << "--exchange_type/-e" << RESET;
    cout << DESCRIPTION << " - способы обмена частей. Возможные варианты: clockwise(по часовой), counterclockwise(против часовой), diagonals(по диагонали)" << RESET << endl << endl;

    cout << FLAG_COLOR << "--freq_color/-F" << RESET;
     cout << DESCRIPTION << " - находит самый часто встречаемый цвет и заменяет его на другой заданный цвет." << RESET << endl;
    cout << SECOND_TEXT << "Обязательные флаги:" << RESET << endl;
    cout << FLAG_COLOR << "--color/-C" << RESET;
       cout << DESCRIPTION << " - новый цвет. Цвет задается строкой 'red.green.blue' - значения от 0 до 255 через точку." << RESET << endl << endl;
}
```

Название файла: collectoprover.h

```
#ifndef COURSE_COLLECTOPROVER_H
#define COURSE_COLLECTOPROVER_H

int int_prove(string value, string place);
Rgb color_prove(string color);
string component_name_prove(string comp);
unsigned char color_value_prove(string value);
vector<int> coord_prove(string coord, int H);
char exchange_type_prove(string exchange_type);
void help();

#endif //COURSE_COLLECTOPROVER_H
```

Название файла: Makefile

```
all: main clean

main: main.o collectoprover.o bmp_class.o
	g++ main.o collectoprover.o bmp_class.o -o cw

main.o: main.cpp my_lib.h collectoprover.h bmp_class.h
	g++ -c -g main.cpp

bmp_class.o: bmp_class.cpp bmp_class.h my_lib.h collectoprover.h
	g++ -c -g bmp_class.cpp

collectoprover.o: collectoprover.cpp my_lib.h
	g++ -c -g collectoprover.cpp

clean:
	rm -rf *.o
```