МИНОБРНАУКИ РОССИИ
Санкт-Петербургский государственный
электротехнический университет
«ЛЭТИ» им. В.И. Ульянова (Ленина)
Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка bmp файлов на языке Си

Студентка гр. 3382 _____ Копасова К. А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студентка Копасова К. А.

Группа 3382

Тема работы: Обработка bmp файлов на языке Си

Исходные данные:

Вариант 4.3

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. Рисование прямоугольника. Флаг для выполнения данной операции: `--rect`. Он определяется:

- Координатами левого верхнего угла. Флаг `--left_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y

- Координатами правого нижнего угла. Флаг `--right_down`, значение задаётся в формате `right.down`, где right – координата по x, down – координата 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)
- Прямоугольник 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`)

2. Сделать рамку в виде узора. Флаг для выполнения данной операции: `--ornament`. Рамка определяется:

- Узором. Флаг `--pattern`. Обязательные значения: rectangle и circle, semicircles. Также можно добавить свои узоры (красивый узор можно получить используя фракталы). Подробнее здесь: https://se.moevm.info/doku.php/courses:programming:cw_spring_ornament
- 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)
- Шириной. Флаг `--thickness`. На вход принимает число больше 0
- Количеством. Флаг `--count`. На вход принимает число больше 0

При необходимости можно добавить дополнительные флаги для необозначенных узоров

3. Поворот изображения (части) на 90/180/270 градусов. Флаг для выполнения данной операции: `--rotate`. 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
- Углом поворота. Флаг `--angle`, возможные значения: `90`, `180`, `270`

Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции.

Предполагаемый объем пояснительной записки:

Не менее 25 страниц.

Дата выдачи задания: 18.03.2024

Дата сдачи реферата: 16.05.2024

Дата защиты реферата: 22.05.2024

Студентка _____ Копасова К. А.

Преподаватель _____ Глазунов С.А.

АННОТАЦИЯ

Данная программа обрабатывает BMP-изображения. Она позволяет выполнять следующие операции:

- Рисовать прямоугольники с закругленными углами.
- Рисовать орнаменты (рамки) из прямоугольников, кругов или полукругов.
- Поворачивать часть изображения на 90, 180 или 270 градусов.
- Выводить информацию о BMP-файле.
- Выводить справку о программе и её функциях.

Все команды и параметры задаются через аргументы командной строки. Если ввод команд или параметров некорректный, то программа заканчивает работу с соответствующей ошибкой.

SUMMARY

This program processes BMP images. It allows you to perform the following operations:

- Draw rectangles with rounded corners.
- Draw ornaments (frames) from rectangles, circles or semicircles.
- Rotate part of the image by 90, 180 or 270 degrees.
- Output information about the BMP file.
- Display information about the program and its functions.

All commands and parameters are set via command line arguments. If the input of commands or parameters is incorrect, the program ends with a corresponding error.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является обобщение полученных знаний за 2 семестр по дисциплине «программирование» о языке программирования Си, а также их применение на практике.

Задачи:

1. Разобраться с заголовочными файлами;
2. Разобраться со структурой bmp – файлов;
3. Разобраться с подачей флагов для программы;
4. Реализовать функции чтения и записи bmp – файлов;
5. Реализовать функцию setPixels;
6. Реализовать функцию рисовании линии;
7. Реализовать функцию рисовании круга;
8. Реализовать функции 1, 2 и 3.

ОСНОВНАЯ ЧАСТЬ

1. Объявление библиотек и постоянных

Подключим заголовочные файлы стандартных библиотек Си:

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <getopt.h>

Далее определяем структуры и макросы, необходимые для работы с BMP-изображениями.

Макросы:

#define MAX_FILENAME 500

#define MAX_CONST 100

#define INFO_ABOUT_STUD "Course work for option 4.3, created by Ksenia Kopasova.\n"

1. MAX_FILENAME: Задает максимальную длину имени файла (500 символов). Используется для буферов, где будут храниться имена файлов.

2. MAX_CONST: Задает некоторое максимальное значение (100). Возможно, это значение используется в других частях программы для различных целей.

3. INFO_ABOUT_STUD: Строковая константа с информацией о студенте и курсовой работе. Вероятно, выводится в консоль или записывается в лог.

Выравнивание стуктур:

#pragma pack(push, 1)

Строка выше указывает компилятору выравнивать структуры с размером упаковки 1 байт. Это важно для правильного считывания и записи бинарных данных из BMP-файлов, которые имеют строгую структуру.

Далее определяем структуры:

1. Структура BitmapFileHeader содержит информацию о заголовке файла изображения bmp;

2. Структура BitmapInfoHeader содержит информацию о характеристиках изображения bmp;

3. Структура Rgb содержит представление цвета в формате RGB.

4. Структура option long_options для обработки аргументов командной строки с использованием функции getopt_long() из библиотеки getopt.h. Она определяет длинные опции (длинные флаги), которые пользователь может передавать при запуске программы.

2. Функции

2. 1. Функции ввода и сохранения изображения

Функция read_bmp: функция read_bmp предназначена для чтения BMP-изображения из файла и возвращения двумерного массива пикселей. Функция также считывает и проверяет заголовки файла и информации о изображении.

Функция начинается с открытия файла для чтения в бинарном режиме. Если файл не удается открыть, функция выводит сообщение об ошибке и завершает программу с кодом 41.

После успешного открытия файла, функция считывает заголовок файла BMP (BitmapFileHeader) и заголовок информации о изображении (BitmapInfoHeader). Заголовки читаются последовательно из файла, используя fread.

Затем функция извлекает высоту и ширину изображения из заголовка информации о изображении. Эти значения используются для выделения памяти под двумерный массив структур Rgb, которые представляют пиксели изображения.

Для каждого из H строк изображения выделяется память с учетом выравнивания каждой строки по границе в 4 байта. Это выравнивание достигается добавлением необходимого количества байтов (от 0 до 3) к каждой строке, чтобы ее размер был кратен 4. Выравнивание необходимо из-за специфики формата BMP, который требует, чтобы каждая строка имела размер кратный 4 байтам.

После выделения памяти для каждой строки, функция считывает пиксели изображения из файла, включая байты выравнивания. Считывание выполняется в цикле для каждой строки изображения.

После завершения чтения всех строк, функция закрывает файл.

Затем функция проверяет корректность заголовков. Она проверяет, что размер заголовка информации о изображении равен 40 байтам, что глубина цвета изображения равна 24 битам на пиксель, что сигнатура файла равна 0x4D42 (что соответствует символам 'BM' в ASCII) и что изображение не сжато. Если хотя бы одно из этих условий не выполняется, функция завершает программу с кодом 41.

Если все проверки проходят успешно, функция возвращает указатель на двумерный массив пикселей Rgb. Этот массив может быть затем использован для дальнейшей обработки изображения в других частях программы.

Функция write_bmp: Функция write_bmp предназначена для записи двумерного массива пикселей в файл в формате BMP. Она принимает название выходного файла, массив пикселей, высоту и ширину изображения, а также указатели на заголовки файла и информации о изображении.

Сначала функция открывает файл для записи в бинарном режиме. Если файл не удается открыть, функция выводит сообщение об ошибке и завершает программу с кодом 42.

После успешного открытия файла, функция записывает заголовок файла BMP и заголовок информации о изображении в файл, используя fwrite. Эти заголовки уже должны быть корректно заполнены и переданы в функцию.

Затем функция вычисляет количество байтов, которое необходимо записывать для каждой строки изображения с учетом выравнивания. Ширина строки в байтах вычисляется как произведение ширины изображения и размера структуры Rgb, плюс дополнительные байты выравнивания, чтобы общая длина строки была кратна 4 байтам. Это делается для соблюдения спецификаций формата BMP.

Далее функция в цикле проходит по каждой строке изображения и записывает её в файл, используя fwrite. Она записывает всю строку, включая байты выравнивания.

После записи всех строк изображений, функция закрывает файл, вызывая fclose.

Таким образом, функция завершает свою работу, создавая или перезаписывая файл с BMP-изображением, основанным на переданном массиве пикселей и соответствующих заголовках.

2.2. Дополнительные функции

Функция setPixels: функция setPixels предназначена для установки цвета определенного пикселя в двумерном массиве пикселей arr. Она принимает массив пикселей, координаты пикселя (x, y) и массив color, содержащий значения цветов красного, зеленого и синего каналов. Сначала функция проверяет, что координаты пикселя неотрицательны. Если координаты (x, y) удовлетворяют этому условию, функция присваивает значения цветов из массива color соответствующим полям структуры Rgb в массиве arr по указанным координатам. Цвета красного, зеленого и синего каналов пикселя устанавливаются соответственно значениями color[0], color[1] и color[2]. Таким образом, функция изменяет цвет пикселя по координатам (x, y) в массиве arr на указанный цвет, если координаты находятся в допустимых пределах.

Функция max: функция находит среди двух элементов наибольший.

Функция min: функция находит среди двух элементов минимальный.

Функция swap_int: функция выполняет обмен значениями двух чисел.

Функция DrawSemic_OY: функция для рисования вертикальных компонент рамки из полукругов. На вход подается двумерный массив структур Rgb arr, представляющий изображение, высота изображения H, длина изображения W, координаты центра полукруга xc и yc, радиус полукруга rad, толщина линии полукруга thickness, цвет круга color и количество полукругов count, которое нужно нарисовать. Она выполняет нужное количество итераций в цикле while, чтобы нарисовать заданное количество полукруглых орнаментальных элементов, размещенных вертикально на изображении arr с указанными параметрами. С каждой итерацией координата yc уменьшается на константное значение 2*rad – thickness, чир обеспечивает вертикальное смещение для следующего полукруглого элемента.

Функция DrawSemic_OX: функция для рисования горизонтальных компонент рамки из полукругов. На вход подается двумерный массив структур Rgb arr, представляющий изображение, высота изображения H, длина изображения W, координаты центра полукруга xc и yc, радиус полукруга rad, толщина линии полукруга thickness, цвет круга color и количество полукругов count, которое нужно нарисовать. Функция выполняет нужное количество итераций в цикле while, чтобы нарисовать заданное количество полукруглых орнаментальных элементов, размещенных горизонтально на изображении arr с указанными параметрами. С каждой итерацией координата xc увеличивается на константное значение 2*rad – thickness, чир обеспечивает горизонтальное смещение для следующего полукруглого элемента.

Функция print_file_header: выводит информацию структуры BitmapFileHeader.

Функция print_info_header: выводит информацию структуры BitmapInfoHeader.

Функция printHelp: выводит справку о работе программы.

2.3. Функции построения линий и кругов

1. *Функция draw_line:* функция используется для рисования горизонтальных или вертикальных линий на изображении, представленном двумерным массивом arr типа Rgb. В зависимости от переданных параметров x0, y0, x1, y1 и thickness, функция рисует линию определенной толщины и цвета. На вход подаются следующие параметры: двумерный массив структур Rgb arr, представляющий изображение, высота изображения H, ширина изображения W, начальные координаты линии x0 и y0, конечные координаты линии x1 и y1, толщина линии thickness, color: массив из трех целых чисел [r, g, b], определяющих цвет линии.

Функция начинает с проверки предусловий: если начальные или конечные координаты находятся за пределами изображения или толщина линии некорректна (меньше или равна нулю), то программа завершается с кодом ошибки 41.

Затем функция определяет тип линии: если x0 равно x1, это означает вертикальную линию. Функция проходит по каждой координате y от y0 до y1 и устанавливает пиксели с заданным смещением по толщине thickness от вертикальной координаты x0 (в зависимости от ориентации оси Y на изображении). Если y0 равно y1, это означает горизонтальную линию. Функция проходит по каждой координате x от x0 до x1 и устанавливает пиксели с заданным смещением по толщине thickness от горизонтальной координаты y0.

Внутренний цикл для каждой координаты линии итерируется через толщину thickness, устанавливая пиксели на заданной координате с заданным цветом. Функция setPixels используется для установки цвета пикселя по заданным координатам.

Функция draw_line обеспечивает возможность рисования горизонтальных и вертикальных линий с заданной толщиной и цветом на

изображении arr, учитывая предварительные проверки координат и толщины линии для корректного отображения на изображении.

2. *Функция drawCircleOrn:* функция используется для рисования окружностей или круговых участков на изображении, представленном двумерным массивом arr типа Rgb. Она принимает следующие параметры:

На вход подаются следующие параметры: двумерный массив структур Rgb arr, представляющий изображение, высота изображения H, ширина изображения W, начальные центра окружности x0 и y0, радиус окружности radius, толщина линии thickness, color: массив из трех целых чисел [r, g, b], определяющих цвет линии, флаг, указывающий тип рисования fl_cir. Если fl_cir = 1 - рисование окружности, исключая ее контур, fl_cir = 0 - рисование контура окружности, fl_cir = 2 - рисование всей окружности.

Инициализируем дополнительную переменную t нулем для управления толщиной окружности. Определяем cur_rad как radius - t, где radius - радиус окружности, а t - текущая толщина.

Используем цикл while для отрисовки толщины: пока t меньше thickness, выполняются следующие действия:

1) Вычисляется текущий радиус cur_rad.

2) Двойной цикл for проходит по координатам x и y в квадратной области, описанной около центра (x0, y0) с радиусом cur_rad.

3) Для каждой точки (x, y) внутри этой области выполняется следующее:

4) Проверяется, что x и y находятся в пределах размеров изображения (W, H).

5) Вычисляются смещения dx и dy от центра (x0, y0) до текущей точки (x, y).

6) Вычисляется сумма квадратов sum_sq для определения расстояния от центра до текущей точки.

В зависимости от значения fl_cir:

1) Если fl_cir = 1 и текущая точка находится за пределами окружности (sum_sq >= cur_rad * cur_rad), пиксель устанавливается в заданный цвет с помощью функции setPixels.

2) Если fl_cir = 0 и текущая точка находится на контуре окружности (sum_sq >= (cur_rad - 1) * (cur_rad - 1) и < cur_rad * cur_rad), пиксель устанавливается в заданный цвет.

3) Если fl_cir = 2 и текущая точка находится внутри окружности (sum_sq < cur_rad * cur_rad), пиксель устанавливается в заданный цвет.

После каждой итерации цикла while, переменная t увеличивается на единицу, что увеличивает текущую толщину cur_rad окружности.

Функция drawCircleOrn позволяет рисовать окружности или их части на изображении, учитывая заданную толщину и тип рисования (fl_cir). Все точки, соответствующие условиям внутри циклов, закрашиваются в указанный цвет с использованием функции setPixels.

2.4. Основные функции

1. Функция DrawRectOrn:

Функция предназначена для рисования прямоугольников с возможностью скругления углов или отрисовки рамок на изображении, представленном двумерным массивом file_bmp типа Rgb. Она принимает следующие параметры:

- file_bmp - двумерный массив структур Rgb, представляющий изображение.
- H - высота изображения.
- W - ширина изображения.
- x0, y0 - координаты левого верхнего угла прямоугольника.
- x1, y1 - координаты правого нижнего угла прямоугольника.
- thickness - толщина линий.
- color - массив из трех целых чисел [r, g, b], определяющих цвет линий.
- fill_flag - флаг указывает, нужно ли заливать прямоугольник.
- fill_color - массив из трех целых чисел [r, g, b], определяющих цвет заливки.
- FLAG - флаг, определяющий тип рисования:
    - FLAG = 0 - рисование рамки прямоугольника.
    - FLAG = 1 - рисование скругленного прямоугольника с возможной заливкой.

Сначала происходит проверка флага fill_flag: если fill_flag установлен (fill_flag != 0), значит нужно рисовать и заливать прямоугольник.

Далее вычисляется thickness_fill как абсолютное значение разницы между x1 и x0 (ширина прямоугольника) плюс половина thickness (для заливки). В зависимости от направления рисования (слева направо или справа налево, сверху вниз или снизу вверх) вызывается функция draw_line для отрисовки и заливки прямоугольника.

Рисование простого прямоугольника или рамки: если fill_flag не установлен (fill_flag == 0), то рисуется простой прямоугольник или рамка в зависимости от значения FLAG.

- Если FLAG = 0 (рамка):
    - Вызываются последовательно четыре вызова draw_line для рисования каждой стороны прямоугольника.
- Если FLAG = 1 (скругленный прямоугольник):
    - Вычисляются границы прямоугольника с учетом толщины (thickness).
    - Вызываются четыре вызова draw_line для отрисовки прямоугольника с закругленными углами.

Функция DrawRectOrn предоставляет возможность создания прямоугольников с различными видами линий и формами (скругленные или прямоугольные) на изображении, задавая необходимые параметры через аргументы функции.

2. Функция drawCircleOrn:

Функция предназначена для рисования окружностей с заданными параметрами (центром, радиусом, толщиной линии и цветом) на изображении, представленном двумерным массивом arr типа Rgb. Она принимает следующие параметры:

- arr - двумерный массив структур Rgb, представляющий изображение.
- H - высота изображения.
- W - ширина изображения.
- x0, y0 - координаты центра окружности.
- radius – радиус окружности.
- thickness - толщина линий.
- color - массив из трех целых чисел [r, g, b], определяющих цвет линий.

- fl_cir – флаг, определяющий, какую окружность рисовать:
    - fl_cir = 0 – отрисовка пикселей, находящихся вне окружности (рисуется кольцо с внешним радиусом cur_rad).
    - fl_cir = 1 – отрисовка пикселей, находящихся на окружности (рисуется сама окружность с радиусом cur_rad).
    - fl_cir = 2 – отрисовка пикселей, находящихся внутри окружности (рисуется круг с радиусом cur_rad).

Инициализируем переменную t для отслеживания текущей толщины линии (начальное значение - 0). Определим радиус окружности cur_rad (cur_rad вычисляется как radius - t, где radius - заданный радиус окружности), который уменьшается с увеличением толщины t.

Цикл while выполняется до достижения максимальной толщины thickness. На каждой итерации цикла происходит отрисовка кольца с увеличивающейся толщиной.

Двойной вложенный цикл for итерируется по координатам x и y в пределах квадрата, описывающего окружность с текущим радиусом cur_rad и центром (x0, y0). Проверяется, что координаты x и y находятся в пределах изображения (меньше ширины W и высоты H). Далее вычисляются расстояния dx и dy от текущей точки (x, y) до центра окружности (x0, y0). Вычисляется квадрат расстояния до центра sum_sq = dx * dx + dy * dy.

В зависимости от значения fl_cir (определяющего тип отрисовки окружности), производятся различные действия:

- fl_cir == 1: Отрисовка пикселей, находящихся вне окружности (рисуется кольцо с внешним радиусом cur_rad).
- fl_cir == 0: Отрисовка пикселей, находящихся на окружности (рисуется сама окружность с радиусом cur_rad).
- fl_cir == 2: Отрисовка пикселей, находящихся внутри окружности (рисуется круг с радиусом cur_rad).

В соответствии с условиями проверки вызывается функция setPixels, которая устанавливает заданный цвет color для пикселя с координатами (x, y) на изображении.

После отрисовки каждого кольца инкрементируется переменная t, чтобы уменьшить текущий радиус cur_rad и увеличить толщину следующего кольца.

Функция drawCircleOrn позволяет гибко рисовать окружности с изменяемой толщиной линии и задавать различные типы отрисовки (внешняя граница, сама окружность, внутренняя область) в зависимости от значения fl_cir.

3. Функция RotatePixct:

Функция предназначена для поворота определенной прямоугольной области с заданными параметрами (координаты левого верхнего угла и правого нижнего угла) на изображении, представленном двумерным массивом arr типа Rgb. Она принимает следующие параметры:

- arr - двумерный массив структур Rgb, представляющий изображение.
- H - высота изображения.
- W - ширина изображения.
- x0, y0 - координаты левого верхнего угла поворачиваемой области.
- x1, y1 - координаты правого нижнего угла поворачиваемой области.
- angle – угол поворота поворачиваемой области (возможное значение – 90, 180, 270).

Вычислим размеры и центр прямоугольной области: height и width вычисляются как абсолютные разности между y1 и y0, а также x1 и x0 соответственно, чтобы определить высоту и ширину области. xc и yc вычисляются как центральные координаты прямоугольной области.

Вычислим начальные координаты для поворота: start_x и start_y определяют начальные координаты, откуда начнется поворот. Они вычисляются таким образом, чтобы центр прямоугольника оказался в центре повернутой области, а также учитывается расположение прямоугольника относительно координатной сетки изображения.

Создается временный массив copy_color, в который копируются цвета пикселей из области, которая будет повернута. Это делается с помощью двойного цикла for, который проходит через каждый пиксель в прямоугольной области и копирует его цвет в copy_color.

Далее, в зависимости от угла поворота (angle), пиксели в copy_color поворачиваются и устанавливаются в итоговом массиве arr. Это происходит также с помощью двойного цикла for, в котором пиксели из временного массива переносятся в итоговый массив с учетом корректировки координат и порядка пикселей для заданного угла поворота.

В конце функция освобождает выделенную для временного массива память.

Важно отметить, что данная реализация не обрабатывает все возможные углы поворота и может работать некорректно для углов, отличных от 90, 180 и 270 градусов. Кроме того, необходимо убедиться, что функция setPixels правильно устанавливает цвет пикселя в итоговом массиве.

3. Функция main

Определим переменные, которые будут использоваться в программе для хранения различных параметров, передаваемых через аргументы командной строки, а также для внутренних вычислений и управления процессом работы программы. Рассмотрим каждую переменную подробнее:

1) filename[MAX_FILENAME] и output_filename[MAX_FILENAME] - это массивы символов, предназначенные для хранения имени входного и выходного файлов соответственно. MAX_FILENAME определяет максимальную длину имени файла.

2) thickness, fill_flag - эти переменные используются для хранения толщины линии и флага, указывающего наличие заливки при рисовании фигур.

3) x0, y0, x1, y1 - эти переменные хранят координаты прямоугольника, который может быть нарисован на изображении.

4) rrr, ggg, bbb, rrr1, ggg1, bbb1 - эти переменные представляют цветовые компоненты RGB для цвета линий и цвета заливки соответственно.

5) color[3], fill_color[3] - массивы целых чисел, используемые для хранения RGB-компонент цвета и цвета заливки.

6) pattern[MAX_CONST] - массив символов для хранения шаблона орнамента.

7) count – целочисленная переменная для хранения количества орнаментов.

8) flag_orn_pattern - переменная для хранения информации о выбранном орнаменте (прямоугольный, круглый, полукруглый).

9) angle - переменная для хранения угла поворота изображения.

10) flag_rect, flag_ornam, flag_rotate, flag_info - флаги, которые устанавливаются в единицу, если соответствующие действия были запрошены через аргументы командной строки.

11) opt, option_index- эти переменные используются для обработки аргументов командной строки с помощью функции getopt_long_only. opt хранит текущий обработанный аргумент, а option_index используется для отслеживания индекса аргумента в массиве аргументов.

Переменные играют ключевую роль в управлении поведением программы, обеспечивая хранение переданных параметров и управление процессом их обработки.

Далее обрабатываем аргументы командной строки, переданных программе при её запуске. Разберем каждый шаг подробнее:

opt = getopt_long_only(argc, argv, "i:o:hl:r:t:c:f:_:e:m:a:n:", long_options, &option_index);:

Функция getopt_long_only используется для разбора аргументов командной строки.

Она принимает параметры:

- argc: количество аргументов командной строки.
- argv: массив строк, содержащий сами аргументы.
- "i:o:hl:r:t:c:f:_:e:m:a:n:": строка, определяющая короткие аргументы. Например, "i:" означает, что после аргумента -i должен следовать еще один аргумент.
- long_options: массив структур option, определяющий длинные аргументы (аргументы, начинающиеся с --).
- &option_index: указатель на переменную, в которой будет сохранен индекс найденного аргумента в массиве long_options.

Результатом работы функции является очередной аргумент командной строки или -1, если аргументы закончились.

while (opt != -1) { ... }:

Цикл выполняется до тех пор, пока не будет обработан последний аргумент командной строки.

Внутри цикла аргументы обрабатываются с помощью оператора switch, который сопоставляет значение opt с различными случаями.

1. *case 'e'* (обработка аргумента --rect)- устанавливается флаг flag_rect, указывающий на необходимость рисования прямоугольника.
2. *case 10:* (обработка длинного аргумента –-ornament) - устанавливается флаг flag_ornam, указывающий на необходимость рисования орнамента.
3. *case 7:* (обработка аргумента --pattern) - читывается значение аргумента в переменную pattern. В зависимости от значения pattern устанавливается значение flag_orn_pattern для выбора типа орнамента.
4. *case 8:* (обработка аргумента --count) - считывается значение аргумента в переменную count.
5. *case 'r':* (обработка аргумента --rotate) - устанавливается флаг flag_rotate, указывающий на необходимость поворота изображения.
6. *case 9:* (обработка аргумента --angle) - считывается значение угла поворота в переменную angle.
7. *case 'n':* (обработка аргумента --info: устанавливается флаг flag_info, указывающий на запрос информации о файле.
8. *case 'i':* и *case 'o':* (обработка аргументов --input и --output): считывается имя входного и выходного файла соответственно в переменные filename и output_filename.
9. *case 1:, case 2:, case 3:, case 4:, case 5:, case 6:* (обработка остальных аргументов (координат, толщины линий, цветов)) - считываются и сохраняются соответствующие значения в переменные программы.

10. *case 'h':* (обработка аргумента --help) - выводится справка о программе с помощью функции printHelp().

opt = getopt_long_only(argc, argv, "i:o:hl:r:t:c:f:_:e:m:a:n:", long_options, &option_index); - повторный вызов getopt_long_only для получения следующего аргумента командной строки.

argc -= optind; argv += optind; - корректировка argc и argv, чтобы они указывали на оставшиеся аргументы командной строки после обработки текущих.

Далее программа обрабатывает параметры и выполненяет соответствующие действия в зависимости от установленных флагов и аргументов командной строки.

if (strcmp(filename, output_filename) == 0) { exit(42); } - этот блок проверяет, чтобы входное и выходное имена файлов не совпадали, иначе программа завершается с кодом ошибки 42.

Далее программа в соответствии с поданной функцией выполняет определенные действия:

1) if (flag_info) { ... } - если установлен флаг flag_info, программа выводит информацию о файле, считывая заголовки файла с помощью BitmapFileHeader и BitmapInfoHeader, а затем выводит их содержимое.

*2) if (flag_rect) { ... } - е*сли установлен флаг flag_rect, программа рисует прямоугольник на изображении.

- Считываются заголовки файла с помощью BitmapFileHeader и BitmapInfoHeader.
- Высота и ширина изображения сохраняются в переменные H и W.

- Проверяется корректность заданных координат прямоугольника и толщины линий.
- Изображение считывается из файла с помощью функции read_bmp и сохраняется в file_bmp.
- Вызывается функция DrawRectOrn, которая рисует прямоугольник на изображении.
- Затем вызываются функции drawCircleOrn для рисования скругленных углов прямоугольника.
- Измененное изображение записывается в выходной файл с помощью функции write_bmp.

3) if (flag_ornam) { ... } - проверяется наличие флага flag_ornam, указывающего на необходимость рисования рамочек.

Считываются заголовки файла с помощью BitmapFileHeader и BitmapInfoHeader, и само изображение считывается из файла с помощью функции read_bmp. Затем высота (H) и ширина (W) изображения сохраняются в соответствующих переменных.

3. 1) if (flag_orn_pattern == 0) { ... } - если установлен флаг flag_orn_pattern с значением 0, то рисуется прямоугольная рамочка.

Проверяется корректность толщины (thickness) и количества (count) прямоугольников. И далее в цикле рисуются прямоугольные рамочки с учетом заданных параметров с помощью построения линий функцией draw_line.

3. 2) if (flag_orn_pattern == 1) { ... } - *е*сли установлен флаг flag_orn_pattern с значением 1, то рисуется рамочка-кружочек.

Определяется центр изображения (xc, yc) и радиус рамочки. Далее рисуется круг с использованием функции drawCircleOrn. Затем рисуются две

прямые линии с помощью функции draw_line, которые дополняют круг, чтобы получилась рамочка-кружочек.

3. 3) if (flag_orn_pattern == 2) { ... } - если установлен флаг flag_orn_pattern с значением 2, то рисуется рамка с полукругами.

Проверяется корректность толщины (thickness) и количества (count) полукругов. Рассчитываются параметры полукругов и рисуются с использованием функций DrawSemic_OY и DrawSemic_OX.

В конце изображение с рамочками записывается в выходной файл с помощью функции write_bmp.

*4) `if (flag_rotate) { ... }* - проверяется наличие флага `flag_rotate`, указывающего на необходимость поворота изображения.

Объявляются переменные для хранения заголовков файла с изображением. Считываются заголовки файла и само изображение с помощью функции read_bmp. Заголовки сохраняются в соответствующие переменные. Определяются высота (`H`) и ширина (`W`) изображения. Далее вызывается функция `RotatePict`, которая выполняет поворот части изображения. После выполнения поворота, измененное изображение записывается в выходной файл с помощью функции `write_bmp`.

Заключение

В ходе курсовой работы я обобщила знания по дисциплине «программирование» за 2 семестр по языку программирования Си и закрепила их на практике.

Была написана программа, которая успешно отрабатывает bmp-файлы и выполняет все функции, которое требует техническое задание.

Список использованных источников

1. Онлайн-курс «Программирование на Си. Практические задания. Второй семестр» // URL: https://e.moevm.info/

2. Д. Ритчи, Б. Керниган Язык программирования Си: 3-е издание.

3. Базовые сведения к выполнению курсовой работы по дисциплине «программирование». Второй семестр.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: main.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <getopt.h>
#define MAX_FILENAME 500
#define MAX_CONST 100
#define INFO_ABOUT_STUD "Course work for option 4.3, created by
Ksenia Kopasova.\n"
#pragma pack(push, 1)
typedef struct BitmapFileHeader {
    unsigned short signature;
    unsigned int filesize;
    unsigned short reserved1;
    unsigned short reserved2;
    unsigned int pixelArrOffset;
} BitmapFileHeader;
typedef struct BitmapInfoHeader {
    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)
struct option long_options[] = {
    {"rect", no_argument, NULL, 'e'},
    {"rotate", no_argument, NULL, 'r'},
    {"input", required_argument, NULL, 'i'},
    {"output", required_argument, NULL, 'o'},
    {"info", no_argument, NULL, 'n'},
    {"help", no_argument, NULL, 'h'},
    {"left_up", required_argument, NULL, 1},
    {"right_down", required_argument, NULL, 2},
    {"thickness", required_argument, NULL, 3},
    {"color", required_argument, NULL, 4},
    {"fill", no_argument, NULL, 5},
    {"fill_color", required_argument, NULL, 6},
    {"ornament", no_argument, NULL, 10},
    {"pattern", required_argument, NULL, 7},
    {"count", required_argument, NULL, 8},
    {"angle", required_argument, NULL, 9},
    {NULL, 0, NULL, 0}
};
Rgb** read_bmp(const char filename[], BitmapFileHeader* bmfh,
BitmapInfoHeader* bmif);
void write_bmp(const char file_name[], Rgb **arr, int H, int W,
BitmapFileHeader* bmfh, BitmapInfoHeader* bmif);
```

```
void print_file_header(BitmapFileHeader header);

void print_info_header(BitmapInfoHeader header);

void printHelp();

void DrawRectOrn(Rgb** file_bmp, unsigned int H, unsigned int W,
int x0, int y0, int x1, int y1, int thickness, int* color, int
fill_flag, int* fill_color, int FLAG);

void swap_int(int* a, int* b);

void draw_line(Rgb **arr, int H, int W, int x0, int y0, int x1, int
y1, int thickness, int color[3]);

void RotatePict(Rgb** arr, int H, int W, int x0, int y0, int x1,
int y1, int angle);

void DrawSemic_OY(Rgb** arr, int H, int W, int xc, int yc, int rad,
int thickness, int* color, int count);

void DrawSemic_OX(Rgb** arr, int H, int W, int xc, int yc, int rad,
int thickness, int* color, int count);

void drawCircleOrn(Rgb** arr, int W, int H, int x0, int y0, int
radius, int thickness, int* color, int fl_cir);

void setPixels(Rgb** arr, int x, int y, int* color);

int main(int argc, char* argv[]){

    char filename[MAX_FILENAME];

    char output_filename[MAX_FILENAME];

    int thickness = 0;

    int fill_flag = 0;

    int x0 = -1, y0 = -1, x1 = -1, y1 = -1, rrr = 0, ggg = 0, bbb =
0, rrr1 = 0, ggg1 = 0, bbb1 = 0;

    int color[3];

    int fill_color[3];

    char pattern[MAX_CONST];

    int count = 0;

    int flag_orn_pattern = -1;

    int angle = 0;

    int flag_rect = 0;

    int flag_ornam = 0;

    int flag_rotate = 0;
```

```
    int flag_info = 0;
    int opt;
    int option_index = 0;
    opt = getopt_long_only(argc, argv, "i:o:hl:r:t:c:f:_:e:m:a:n:",
long_options, &option_index);
    while (opt != -1) {
        switch (opt) {
            case 'e':{//rect
                flag_rect = 1;
                break;
            }
            case 10:{//ornament
                flag_ornam = 1;
                break;
            }
            case 7:{//pattern
                strncpy(pattern, optarg, MAX_CONST - 1);
                if (strcmp(pattern, "rectangle") == 0){
                    flag_orn_pattern = 0;
                }
                if (strcmp(pattern, "circle") == 0){
                    flag_orn_pattern = 1;
                }
                if (strcmp(pattern, "semicircles") == 0){
                    flag_orn_pattern = 2;
                }
                if (flag_orn_pattern == -1){
                    exit(41);
                }
                break;
            }
```

```
case 8:{//count
    sscanf(optarg, "%d", &count);
    if (count <= 0){
        exit(41);
    }
    break;
}

case 'r':{//rotate
    flag_rotate = 1;
    break;
}

case 9:{//rotate
    sscanf(optarg, "%d", &angle);
    if (angle != 90 && angle != 180 && angle != 270){
        exit(41);
    }
    break;
}

case 'n':{//info
    flag_info = 1;
    break;
}

case 'i':{//задаем имя входного изображения
    strncpy(filename, optarg, MAX_FILENAME-1);
    if (filename == NULL){
        exit(41);
```

```
                }
                break;
            }
            case 'o':{//задаем имя выходного изображения
                strncpy(output_filename, optarg, MAX_FILENAME-1);
                if (output_filename == NULL){
                    exit(41);
                }
                break;
            }
            case 1:{
                sscanf(optarg, "%d.%d", &x0, &y0);
                if (x0<0 || y0<0){
                    exit(42);
                }
                break;
            }
            case 2:{//координаты правого нижнего угла. задаем
right.up - x.y
                sscanf(optarg, "%d.%d", &x1, &y1);
                if (x1<0 || y1<0){
                    exit(42);
                }
                break;
            }
            case 3:{//толщина линий. на вход число больше 0.
                thickness = atoi(optarg);
                if (thickness <= 0){
                    exit(42);
                }
                break;
```

```
            }
            case 4:{//color
                sscanf(optarg, "%d.%d.%d", &rrr, &ggg, &bbb);
                color[0] = rrr;
                color[1] = ggg;
                color[2] = bbb;
                if ((rrr<0 || rrr>255) || (ggg<0 || ggg>255) ||
(bbb<0 || bbb>255)){
                    exit(42);
                }
                break;
            }
            case 5:{//залит или нет
                fill_flag = 1;
                break;
            }
            case 6:{//цвет заливки если он есть
                sscanf(optarg, "%d.%d.%d", &rrr1, &ggg1, &bbb1);
                fill_color[0] = rrr1;
                fill_color[1] = ggg1;
                fill_color[2] = bbb1;
                if ((rrr1<0 || rrr1>255) || (ggg1<0 || ggg1>255) ||
(bbb1<0 || bbb1>255)){
                    exit(42);
                }
                break;
            }
            case 'h':{//справка
                printf(INFO_ABOUT_STUD);
                printHelp();
                break;
            }
```

```
        }
        opt = getopt_long_only(argc, argv,
"i:o:hl:r:t:c:f:_:e:m:a:n:", long_options, &option_index);
    }
    argc -= optind;
    argv += optind;
    if (strcmp(filename, output_filename) == 0){
        exit(42);
    }
    if (flag_info){//print info
        printf(INFO_ABOUT_STUD);
        BitmapFileHeader bmfh;
        BitmapInfoHeader bmif;
        //Rgb** file_bmp = read_bmp(filename, &bmfh, &bmif);
        print_file_header(bmfh);
        print_info_header(bmif);
        return 0;
    }
    if (flag_rect){//рисуем прямоугольнички
        printf(INFO_ABOUT_STUD);
        BitmapFileHeader bmfh;
        BitmapInfoHeader bmif;
        Rgb** file_bmp = read_bmp(filename, &bmfh, &bmif);
        unsigned int H = bmif.height;
        unsigned int W = bmif.width;
        if (x0>W || y0>H || x1>W || y1>H || thickness == 0 ){
            exit(41);
        }
        DrawRectOrn(file_bmp, H, W, x0, y0, x1, y1, thickness,
color, fill_flag, fill_color, 1);
```

```
        int rad = thickness/2;
        //сомнительно но окей, скругляем
        drawCircleOrn(file_bmp, H, W, x0, H-y0, rad, 1, color, 2);
        drawCircleOrn(file_bmp, H, W, x0, H-y1, rad, 1, color, 2);
        drawCircleOrn(file_bmp, H, W, x1, H-y1, rad, 1, color, 2);
        drawCircleOrn(file_bmp, H, W, x1, H-y0, rad, 1, color, 2);
        write_bmp(output_filename, file_bmp, H, W, &bmfh, &bmif);
    }
    if (flag_ornam){//рисуем рамочки
        printf(INFO_ABOUT_STUD);
        BitmapFileHeader bmfh;
        BitmapInfoHeader bmif;
        Rgb** file_bmp = read_bmp(filename, &bmfh, &bmif);
        unsigned int H = bmif.height;
        unsigned int W = bmif.width;
        if (flag_orn_pattern==0){//рисуем прямоугольную рамочку
            int start_x = 0;
            int start_y = 0;
            int y2 = H;
            int x2 = W;
            if (thickness == 0 || count == 0){
                exit(41);
            }
            while (count > 0){
                DrawRectOrn(file_bmp, H, W, start_x, start_y, x2,
y2, thickness, color, fill_flag, fill_color, 0);
                start_x += 2*thickness;
                start_y += 2*thickness;
                x2 -= 2*thickness;
                y2 -= 2*thickness;
                count--;
```

```
            }
        }
        if (flag_orn_pattern==1){//рисуем рамочку-кружочек
            int yc = H/2;
            int xc = W/2;
            int radius = 0;
            if (xc <= yc){
                radius = xc;
            }else{
                radius = yc;
            }
            drawCircleOrn(file_bmp, H, W, xc, yc, radius, 1, color,
1);
            if (radius == yc){
                draw_line(file_bmp, H, W, xc-radius, yc+radius+2,
xc-radius, yc-radius, xc-radius+1, color);
                draw_line(file_bmp, H, W, W, H+1, W, 0, xc-radius,
color);
            }else{
                draw_line(file_bmp, H, W, 0, yc-radius, xc+radius,
yc-radius, yc-radius, color);
                draw_line(file_bmp, H, W, W, H, 0, H, yc-radius,
color);
            }
        }

        if (flag_orn_pattern==2){//рамка с полукругами
            if (thickness == 0 || count == 0){
                exit(41);
            }
            int new_thick = thickness;
            int rad_OY, rad_OX, d_OY, d_OX;
            if (H%count){
```

```
            d_OY = H/count + 1;
        }else{
            d_OY = H/count;
        }
        if (d_OY%2){
            rad_OY = d_OY/2 + 1;
        }else{
            rad_OY = d_OY/2;
        }
        if (W%count){
            d_OX = W/count + 1;
        }else{
            d_OX = W/count;
        }
        if (d_OX%2){
            rad_OX = d_OX/2 + 1;
        }else{
            rad_OX = d_OX/2;
        }
        if (thickness%2){
            new_thick++;
        }
        DrawSemic_OY(file_bmp, H, W, 0, H - rad_OY , rad_OY +
thickness/2, thickness, color, count);
        DrawSemic_OY(file_bmp, H, W, W-1, H - rad_OY , rad_OY +
thickness/2, thickness, color, count);
        DrawSemic_OX(file_bmp, H, W, rad_OX - thickness/2 + 5,
0, rad_OX+thickness/2, thickness, color, count);
        DrawSemic_OX(file_bmp, H, W, rad_OX - thickness/2 + 5,
H - 1, rad_OX+thickness/2, thickness, color, count);
    }
```

```
        write_bmp(output_filename, file_bmp, H, W, &bmfh, &bmif);
    }
    if (flag_rotate){//поворачиваем часть изображения
        printf(INFO_ABOUT_STUD);
        BitmapFileHeader bmfh;
        BitmapInfoHeader bmif;
        Rgb** file_bmp = read_bmp(filename, &bmfh, &bmif);
        unsigned int H = bmif.height;
        unsigned int W = bmif.width;
        RotatePict(file_bmp, H, W, x0, y0, x1, y1, angle);
        write_bmp(output_filename, file_bmp, H, W, &bmfh, &bmif);
    }
    return 0;
}
void DrawSemic_OY(Rgb** arr, int H, int W, int xc, int yc, int rad,
int thickness, int* color, int count){
    int c = count;
    while (c>0){
        drawCircleOrn(arr, H, W, xc, yc, rad, thickness, color, 0);
        yc -= 2*rad - thickness ;
        c--;
    }
}
void DrawSemic_OX(Rgb** arr, int H, int W, int xc, int yc, int rad,
int thickness, int* color, int count){
    int c = count;
    while (c>0){
        drawCircleOrn(arr, H, W, xc, yc, rad, thickness, color, 0);
        xc += 2*rad - thickness;
        c--;
    }
}
```

```
size_t max(size_t a, size_t b){
    if(a>b){
        return a;
    }else{
        return b;
    }
}
size_t min(size_t a, size_t b){
    if(a<b){
        return a;
    }else{
        return b;
    }
}
void RotatePict(Rgb** arr, int H, int W, int x0, int y0, int x1,
int y1, int angle){
    int height = abs(y1 - y0);
    int width = abs(x1 - x0);
    int xc = (x0+x1)/2;
    int yc = (y0+y1)/2;
    int start_x = xc - height/2;
    int start_y = H - (yc - width/2) - 1;
    y0 = H - y0 - 1;
    y1 = H - y1 - 1;
    int max_x = max(x0, x1);
    int min_x = min(x0, x1);
    int max_y = max(y0, y1);
    int min_y = min(y0, y1);
    int pix_col[3];
```

```
    Rgb** copy_color = (Rgb**)malloc(height * sizeof(Rgb*));
    for (int i = 0; i < height; i++) {
        copy_color[i] = (Rgb*)malloc(width * sizeof(Rgb));
        for (int j=0; j<width; j++){
            copy_color[i][j] = arr[min_y + i + 1][min_x + j];
        }
    }
    if (angle == 90){
        for (int y = 0; y < width; y++){
            for (int x = 0; x < height; x++){
                pix_col[0] = copy_color[x][y].r;
                pix_col[1] = copy_color[x][y].g;
                pix_col[2] = copy_color[x][y].b;
                setPixels(arr, start_x + (height - x - 1), start_y
- (width - y - 1), pix_col);
            }
        }
    }else if (angle == 180){
        for (int y = max_y; y > min_y; y--){
            for (int x = min_x; x < max_x; x++){
                pix_col[0] = copy_color[(max_y - y)][width - (x -
min_x) - 1].r;
                pix_col[1] = copy_color[(max_y - y)][width - (x -
min_x) - 1].g;
                pix_col[2] = copy_color[(max_y - y)][width - (x -
min_x) - 1].b;
                setPixels(arr, x, y, pix_col);
            }
        }
    }else if (angle == 270){
        for (int y = 0; y < width; y++){
            for (int x = 0; x < height; x++){
```

```
                pix_col[0] = copy_color[x][y].r;
                pix_col[1] = copy_color[x][y].g;
                pix_col[2] = copy_color[x][y].b;

                setPixels(arr, start_x + x, start_y - y, pix_col);
            }
        }
    }
    for (int i = 0; i < height; i++){
        free(copy_color[i]);
    }
    free(copy_color);
}
void setPixels(Rgb** arr, int x, int y, int* color) {
    // Установка цвета пикселя в массиве arr по координатам (x, y)
    if (x >=0 && y>=0){
        arr[y][x].r = color[0];
        arr[y][x].g = color[1];
        arr[y][x].b = color[2];
    }
}
void drawCircleOrn(Rgb** arr, int H, int W, int x0, int y0, int
radius, int thickness, int* color, int fl_cir) {
    int t = 0;
    while (t < thickness) {
        int cur_rad = radius - t; // Текущий радиус окружности
(увеличивается с толщиной)
        for (int x = x0 - cur_rad; x <= x0 + cur_rad; x++) {
            for (int y = y0 - cur_rad; y <= y0 + cur_rad; y++) {
                if ( x < W  && y < H ) {
                    int dx = (x0 - x);
                    int dy = (y - y0);
```

```
                    int sum_sq = dx * dx + dy * dy;


                    if ((sum_sq >= cur_rad * cur_rad) && fl_cir ==
1) {
                        // Пиксель находится вне окружности
                        setPixels(arr, x, y, color);
                    }
                    if ((sum_sq >= (cur_rad - 1) * (cur_rad - 1))
&& (sum_sq < cur_rad * cur_rad) && fl_cir == 0) {
                        // Пиксель находится на окружности
                        setPixels(arr, x, y, color);
                    }
                    if ((sum_sq < cur_rad * cur_rad) && fl_cir == 2)
{
                        // Пиксель в окр
                        setPixels(arr, x, y, color);
                    }
                }
            }
        }
        t++;
    }
}
void DrawRectOrn(Rgb** file_bmp, unsigned int H, unsigned int W,
int x0, int y0, int x1, int y1, int thickness, int* color, int
fill_flag, int* fill_color, int FLAG){
    if (fill_flag){
            //рисуем и заливаем наш прямоугольник
            int thickness_fill = abs(x1-x0)+thickness/2;
            if (x1>x0){
                if (y0>y1){
                    draw_line(file_bmp, H, W, x1, y1, x1, y0,
thickness_fill, fill_color);
```

```
                }else{
                    draw_line(file_bmp, H, W, x1, y0, x1, y1,
thickness_fill, fill_color);
                }
            }else{
                if (y0>y1){
                    draw_line(file_bmp, H, W, x0, y1, x0, y0,
thickness_fill, fill_color);
                }else{
                    draw_line(file_bmp, H, W, x0, y0, x0, y1,
thickness_fill, fill_color);
                }
            }
            if (thickness%2==0){
                draw_line(file_bmp, H, W, x0+(thickness/2), y0,
x0+(thickness/2), y1, thickness, color);
                draw_line(file_bmp, H, W, x0, y1+(thickness/2)-1,
x1, y1+(thickness/2)-1, thickness, color);
                draw_line(file_bmp, H, W, x1+(thickness/2)-1, y1,
x1+(thickness/2)-1, y0, thickness, color);
                draw_line(file_bmp, H, W, x0, y0+(thickness/2), x1,
y0+(thickness/2), thickness, color);
            }else{
                draw_line(file_bmp, H, W, x0+(thickness/2), y0,
x0+(thickness/2), y1, thickness, color);
                draw_line(file_bmp, H, W, x0, y1+(thickness/2), x1,
y1+(thickness/2), thickness, color);
                draw_line(file_bmp, H, W, x1+(thickness/2), y1,
x1+(thickness/2), y0, thickness, color);
                draw_line(file_bmp, H, W, x0, y0+(thickness/2), x1,
y0+(thickness/2), thickness, color);
            }
        }else{
            //просто рисуем прямоугольник :)
            if (FLAG){//скругляемое
```

```
                    if (thickness%2==0){
                    draw_line(file_bmp, H, W, x0+(thickness/2), y0,
x0+(thickness/2), y1, thickness, color);
                    draw_line(file_bmp, H, W, x0, y1+(thickness/2)-
1, x1, y1+(thickness/2)-1, thickness, color);
                    draw_line(file_bmp, H, W, x1+(thickness/2)-1,
y1, x1+(thickness/2)-1, y0, thickness, color);
                    draw_line(file_bmp, H, W, x0, y0+(thickness/2),
x1, y0+(thickness/2), thickness, color);
                }else{
                    draw_line(file_bmp, H, W, x0+(thickness/2), y0,
x0+(thickness/2), y1, thickness, color);
                    draw_line(file_bmp, H, W, x0, y1+(thickness/2),
x1, y1+(thickness/2), thickness, color);
                    draw_line(file_bmp, H, W, x1+(thickness/2), y1,
x1+(thickness/2), y0, thickness, color);
                    draw_line(file_bmp, H, W, x0, y0+(thickness/2),
x1, y0+(thickness/2), thickness, color);
                }
            }else{//рамка
                draw_line(file_bmp, H, W, x0+thickness-1, y0+1,
x0+thickness-1, y1, thickness, color);
                draw_line(file_bmp, H, W, x0, y1, x1-1, y1,
thickness, color);
                draw_line(file_bmp, H, W, x1-1, y0+1, x1-1, y1,
thickness, color);
                draw_line(file_bmp, H, W, x0, y0+thickness, x1-1,
y0+thickness, thickness, color);
            }
        }
}
void swap_int(int* a, int* b){
    int t = *a;
    *a = *b;
    *b = t;
```

```
}
void draw_line(Rgb **arr, int H, int W, int x0, int y0, int x1, int
y1, int thickness, int color[3]) {
    if (x0 < 0 || y0 < 0 || x1 < 0 || y1 < 0 || thickness <= 0) {
        exit(41);
    }
    // вертикальная линия
    if (x0 == x1){
        if (y0 >= y1){
            swap_int(&y0, &y1);
        }
        for (int y = y0; y <= y1; y++){
            for (int j = 0; j < thickness; j++){
                if (H - y >= 0 && x0 - j >= 0 && x0 - j <W && H - y
< H){
                    setPixels(arr, x0-j, H-y, color);
                }
            }
            //горизонтальная линия
        }
    }else if (y0 == y1){
            if (x0 >= x1){
                swap_int(&x0, &x1);
            }
            for(int x=x0; x <= x1; x++){
                for(int j=0; j<thickness; j++){
                    if (H-y0+j >= 0 && x>=0 && x<W && H-y0+j<H){
                        setPixels(arr, x, H-y0+j, color);
                    }
                }
            }
        }
```

```
}
Rgb** read_bmp(const char filename[], BitmapFileHeader* bmfh,
BitmapInfoHeader* bmif){
    FILE *f = fopen(filename, "rb");
    if (f == NULL)
    {
        printf("Error opening input file\n");
        exit(41);
    }
    fread(bmfh, 1, sizeof(BitmapFileHeader), f);
    fread(bmif, 1, sizeof(BitmapInfoHeader), f);
    unsigned int H = bmif->height;
    unsigned int W = bmif->width;
    Rgb **arr = malloc(H * sizeof(Rgb *));
    for (int i = 0; i < H; i++)
    {
        arr[i] = calloc(1, W * sizeof(Rgb) + (4 - (W *
sizeof(Rgb)) % 4) % 4);
        fread(arr[i], 1,W * sizeof(Rgb) + (4 - (W * sizeof(Rgb)) %
4) % 4, f);
    }
    fclose(f);
    if (bmif->headerSize != 40 || bmif->bitsPerPixel != 24 || bmfh-
>signature != 0x4D42 || bmif->compression != 0){
        exit(41);
    }
    return arr;
}
void write_bmp(const char file_name[], Rgb **arr, int H, int W,
BitmapFileHeader* bmfh, BitmapInfoHeader* bmif){
    FILE *ff = fopen(file_name, "wb");
    if (ff == NULL)
    {
```

```
        printf("Error opening output file\n");
        exit(42);
    }
    fwrite(bmfh, 1, sizeof(BitmapFileHeader), ff);
    fwrite(bmif, 1, sizeof(BitmapInfoHeader), ff);
    unsigned int w  = W * sizeof(Rgb) + (4 - (W * sizeof(Rgb)) % 4) % 4;
    for (int i = 0; i < H; i++)
    {
        fwrite(arr[i], 1, w, ff);
    }
    fclose(ff);
}
void print_file_header(BitmapFileHeader header){
    printf("signature:\t%x (%hu)\n", header.signature, header.signature);
    printf("filesize:\t%x (%u)\n", header.filesize, header.filesize);
    printf("reserved1:\t%x (%hu)\n", header.reserved1, header.reserved1);
    printf("reserved2:\t%x (%hu)\n", header.reserved2, header.reserved2);
    printf("pixelArrOffset:\t%x (%u)\n", header.pixelArrOffset, header.pixelArrOffset);
}
void print_info_header(BitmapInfoHeader header){
    printf("headerSize:\t%x (%u)\n", header.headerSize, header.headerSize);
    printf("width: \t%x (%u)\n", header.width, header.width);
    printf("height: \t%x (%u)\n", header.height, header.height);
    printf("planes: \t%x (%hu)\n", header.planes, header.planes);
    printf("bitsPerPixel:\t%x (%hu)\n", header.bitsPerPixel, header.bitsPerPixel);
```

```
    printf("compression:\t%x (%u)\n", header.compression,
header.compression);

    printf("imageSize:\t%x (%u)\n", header.imageSize,
header.imageSize);

    printf("xPixelsPerMeter:\t%x (%u)\n", header.xPixelsPerMeter,
header.xPixelsPerMeter);

    printf("yPixelsPerMeter:\t%x (%u)\n", header.yPixelsPerMeter,
header.yPixelsPerMeter);

    printf("colorsInColorTable:\t%x (%u)\n",
header.colorsInColorTable, header.colorsInColorTable);

    printf("importantColorCount:\t%x (%u)\n",
header.importantColorCount, header.importantColorCount);

}

void printHelp(){

    printf("----Information about the use of the course work----
\n");

    printf("This program is designed to perform various functions
with images of the type.bmp. This program can:\n\n\n");

    printf("--rect -               Draw a rectangle\n");

    printf("These flags are used:\n\n");

    printf("--input, -i             Sets the name of the input
image;\n");

    printf("--output, -o            Sets the name of the output
image;\n");

    printf("--left_up               Coordinates of the upper-left
corner. The value is set in the format `left.up`, where left is the
x coordinate, up is the y coordinate;\n");

    printf("--right_down            Coordinates of the lower-right
corner. The value is set in the format `right.down`, where right is
the x coordinate, down is the y coordinate;\n");

    printf("--thickness             The thickness of the lines.
Accepts a number greater than 0 as input;\n");

    printf("--color                 The color of the lines. The
color is set by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are the
numbers specifying the color component;\n");

    printf("--fill                  The rectangle can be filled in
or not. It works as a binary value: there is no flag – false, there
is a flag – true;\n");
```

```
    printf("--fill_color            Works similarly to the `--
color` flag.\n\n\n");

    printf("--ornament -            Make a frame in the form of a
pattern: rectangle, circle or semicilcles.\n");

    printf("These flags are used:\n\n");

    printf("--pattern               Pattern. It can take the
following values: rectangle and circle, semicircles;\n");

    printf("--color                 The color of the lines. The
color is set by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are the
numbers specifying the color component;\n");

    printf("--thickness             The thickness of the lines.
Accepts a number greater than 0 as input;\n");

    printf("--count                 Amount. Accepts a number
greater than 0 as input.\n\n\n");

    printf("--rotate -              Rotate the image (part) by
90/180/270 degrees.\n");

    printf("These flags are used:\n\n");

    printf("--left_up               Coordinates of the upper-left
corner. The value is set in the format `left.up`, where left is the
x coordinate, up is the y coordinate;\n");

    printf("--right_down            Coordinates of the lower-right
corner. The value is set in the format `right.down`, where right is
the x coordinate, down is the y coordinate;\n");

    printf("--angle                 The angle of rotation. Possible
values: `90`, `180`, `270`.\n");

    printf("\n\nOther functions\n");

    printf("-info -                 Prints information about the
image.\n");

    printf("-help, -h -             Displays information about the
program and functions.\n");

}
```

ПРИЛОЖЕНИЕ Б

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

1. Вывод информации о курсовой работе:

```
typa@LAPTOP-RSJS12V1:/mnt/d/cw2024$ gcc main.c && ./a.out --input test2.bmp --info
Course work for option 4.3, created by Ksenia Kopasova.
```

Рисунок 1.

2. Проверка программы на работу флагов --info и –help, а так же проверка ошибок :

```
typa@LAPTOP-RSJS12V1:/mnt/d/cw2024$ gcc main.c && ./a.out --input test2.bmp --info
Course work for option 4.3, created by Ksenia Kopasova.
signature:      4d42 (19778)
filesize:       31fb36 (3275574)
reserved1:      0 (0)
reserved2:      0 (0)
pixelArrOffset: 36 (54)
headerSize:     28 (40)
width:  500 (1280)
height:         355 (853)
planes:         1 (1)
bitsPerPixel:   18 (24)
compression:    0 (0)
imageSize:      0 (0)
xPixelsPerMeter:        b12 (2834)
yPixelsPerMeter:        b12 (2834)
colorsInColorTable:     0 (0)
importantColorCount:    0 (0)
```

Рисунок 2.1.

```
typa@LAPTOP-RSJS12V1:/mnt/d/cw2024$ gcc main.c && ./a.out --input test2.bmp --help
Course work for option 4.3, created by Ksenia Kopasova.
----Information about the use of the course work----
This program is designed to perform various functions with images of the type.bmp. This program can:


--rect -              Draw a rectangle
These flags are used:

--input, -i           Sets the name of the input image;
--output, -o          Sets the name of the output image;
--left_up             Coordinates of the upper-left corner. The value is set in the format `left.up`, where left is the x coordinate, up is the y coordinate;
--right_down          Coordinates of the lower-right corner. The value is set in the format `right.down`, where right is the x coordinate, down is the y coordinate;
--thickness           The thickness of the lines. Accepts a number greater than 0 as input;
--color               The color of the lines. The color is set by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are the numbers specifying the color component;
--fill                The rectangle can be filled in or not. It works as a binary value: there is no flag - false, there is a flag - true;
--fill_color          Works similarly to the `--color` flag.


--ornament -          Make a frame in the form of a pattern: rectangle, circle or semicilcles.
These flags are used:

--pattern             Pattern. It can take the following values: rectangle and circle, semicircles;
--color               The color of the lines. The color is set by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are the numbers specifying the color component;
--thickness           The thickness of the lines. Accepts a number greater than 0 as input;
--count               Amount. Accepts a number greater than 0 as input.


--rotate -            Rotate the image (part) by 90/180/270 degrees.
These flags are used:

--left_up             Coordinates of the upper-left corner. The value is set in the format `left.up`, where left is the x coordinate, up is the y coordinate;
--right_down          Coordinates of the lower-right corner. The value is set in the format `right.down`, where right is the x coordinate, down is the y coordinate;--angle              The angle
ible values: `90`, `180`, `270`.


Other functions
-info -               Prints information about the image.
-help, -h -           Displays information about the program and functions.
```

Рисунок 2.2.

```
typa@LAPTOP-RSJS12V1:/mnt/d/cw2024$ gcc main.c && ./a.out --input test2.jnf --rect
Course work for option 4.3, created by Ksenia Kopasova.
Error opening input file
```

Рисунок 2.3.

3. Проверка корректной работы первой функции:

```
typa@LAPTOP-RSJS12V1:/mnt/d/cw2024$ gcc main.c && ./a.out --input test2.bmp --rect --thickness 25 --left_up 23.25 --right_down 344.456 --color 12.23.143 --o
utput res_test2.bmp
Course work for option 4.3, created by Ksenia Kopasova.
```

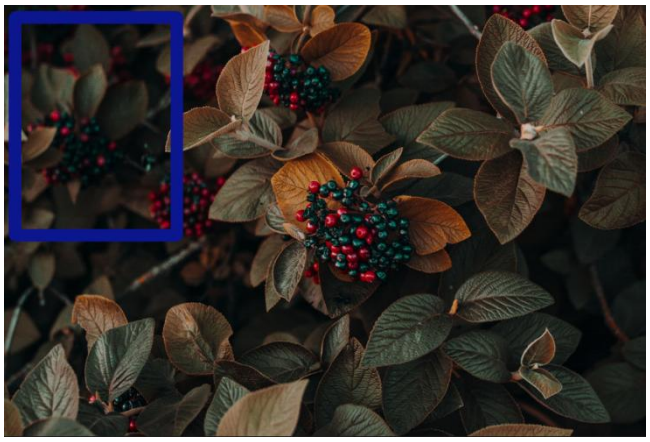

Рисунок 3.1.

```
typa@LAPTOP-RSJS12V1:/mnt/d/cw2024$ gcc main.c && ./a.out --input test2.bmp --rect --thickness 25 --left_up 23.25 --right_down 344.456 --color 12.23.143 --o
utput res_test2.bmp --fill
Course work for option 4.3, created by Ksenia Kopasova.
```

Рисунок 3.2.

4. Проверка корректной работы второй функции:

```
typa@LAPTOP-RSJS12V1:/mnt/d/cw2024$ gcc main.c && ./a.out --input test2.bmp --ornament --pattern rectangle --thickness 15 --count 5 --color 123.23.43 --outp
ut res_test2.bmp
Course work for option 4.3, created by Ksenia Kopasova.
```

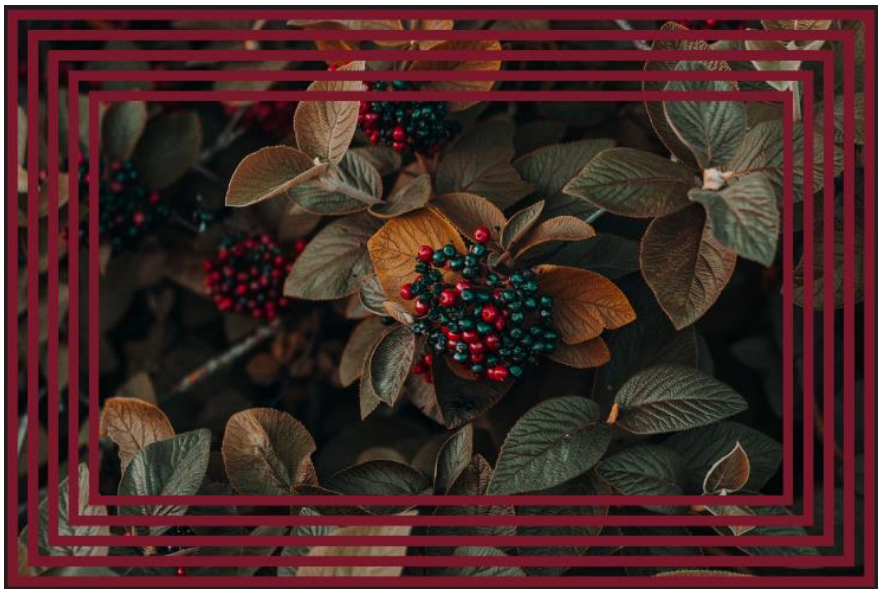

Рисунок 4.1.

```
typa@LAPTOP-RSJS12V1:/mnt/d/cw2024$ gcc main.c && ./a.out --input test2.bmp --ornament --pattern circle  --color 154.23.143 --output res_test2.bmp --count 5
 --thickness 24
Course work for option 4.3, created by Ksenia Kopasova.
```

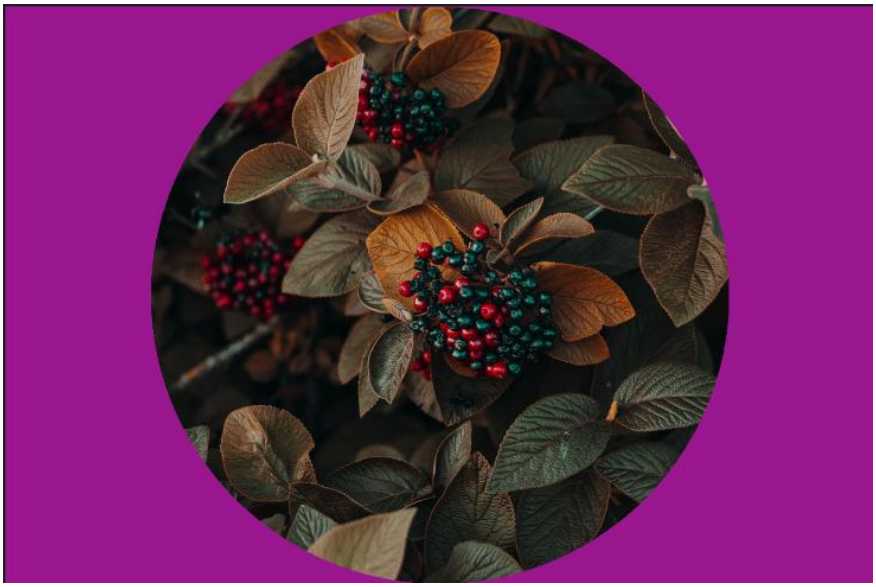

Рисунок 4.2.

```
typa@LAPTOP-RSJS12V1:/mnt/d/cw2024$ gcc main.c && ./a.out --input test2.bmp --ornament --pattern semicircles --output res_test2.bmp --count 5 --thickness 20
 --color 134.23.220
Course work for option 4.3, created by Ksenia Kopasova.
```

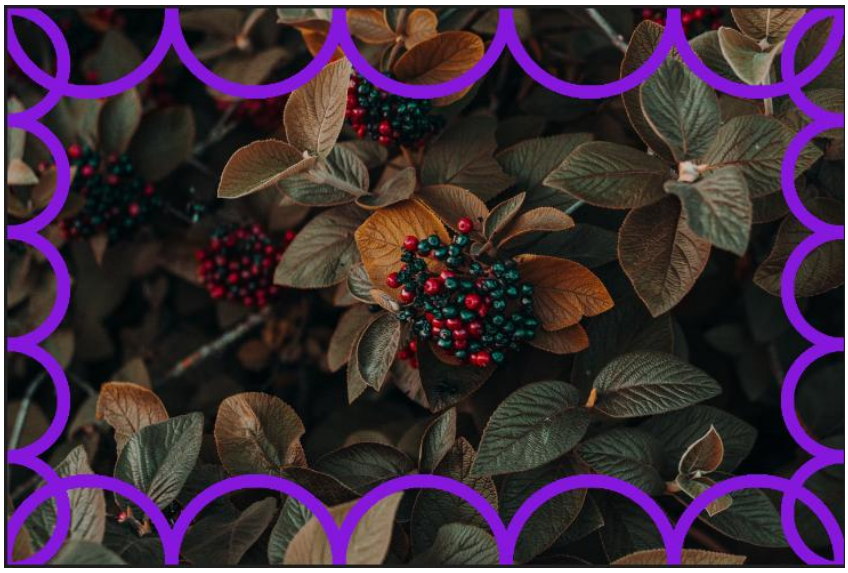

Рисунок 4.3.

5. Проверка корректной работы третьей функции:

```
typa@LAPTOP-RSJS12V1:/mnt/d/cw2024$ gcc main.c && ./a.out --input test2.bmp --rotate --angle 90 --left_up 23.25 --right_down 344.783 --output res_test2.bmp
Course work for option 4.3, created by Ksenia Kopasova.
```

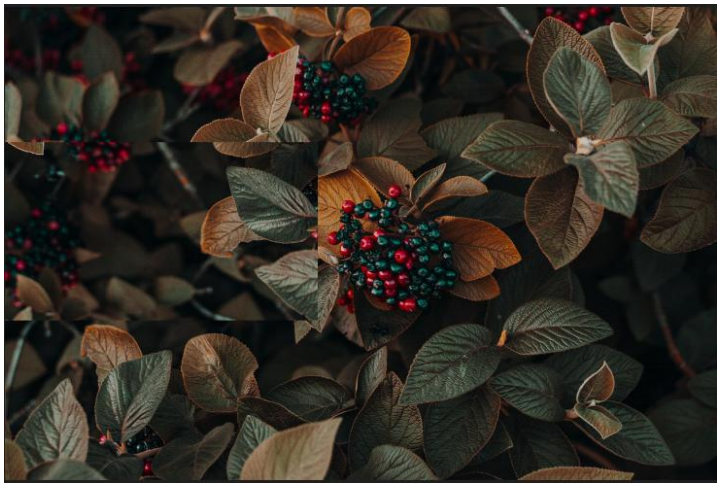

Рисунок 5.1.

```
typa@LAPTOP-RSJS12V1:/mnt/d/cw2024$ gcc main.c && ./a.out --input test2.bmp --rotate --angle 180 --left_up 23.25 --right_down 344.783 --output res_test2.bmp
Course work for option 4.3, created by Ksenia Kopasova.
```

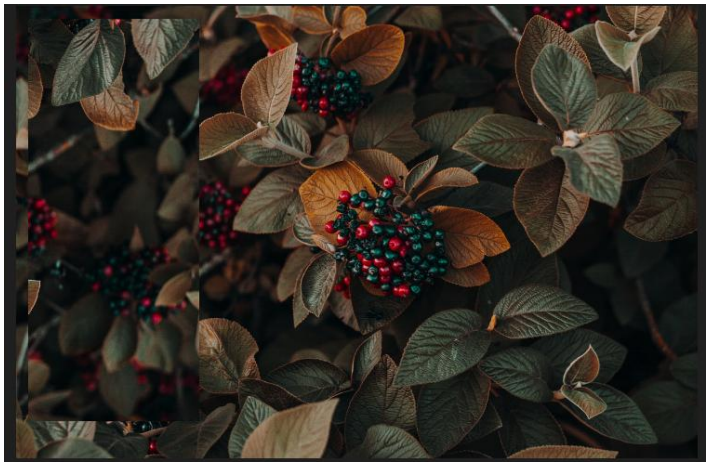

Рисунок 5.2.

```
typa@LAPTOP-RSJS12V1:/mnt/d/cw2024$ gcc main.c && ./a.out --input test2.bmp --rotate --angle 270 --left_up 383.234 --right_down 644.783 --output res_test2.b
mp
Course work for option 4.3, created by Ksenia Kopasova.
```

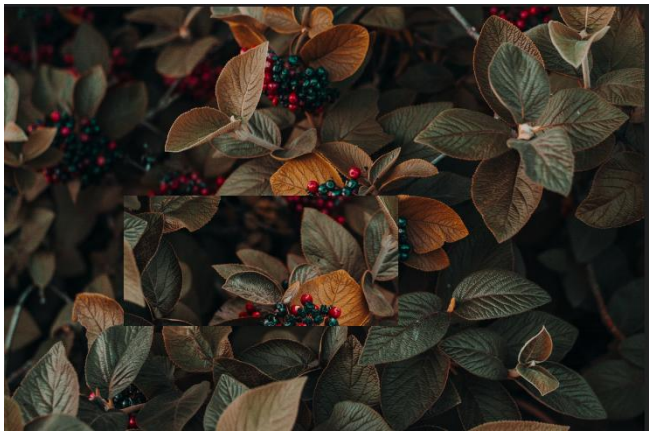

Рисунок 5.3.