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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: обработка изображений

Студентка гр. 4341 _____ Четвертная В.Л.

Преподаватель _____ Виноградова Е.В.

Санкт-Петербург

2025

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студентка Четвертная В.Л.

Группа 4341

Тема работы: Обработка изображений

Исходные данные: Разрабатывается CLI-утилита для обработки PNG-изображений с использованием библиотеки libpng. Утилита поддерживает: отражение области, рисование пентаграммы, рисование прямоугольника, рисование шестиугольника. Программа принимает аргументы через getopt, обрабатывает ошибки и выводит справку.

Содержание пояснительной записки: «Аннотация», «Содержание», «Введение», «Цель», «Задачи», «Описание программы», «Примеры работы программы», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:
Не менее 12 страниц.

Дата выдачи задания: 07.03.2025

Дата сдачи реферата: 18.05.2025

Дата защиты реферата: 23.05.2025

Студентка _____ Четвертная В.Л.

Преподаватель _____ Виноградова Е.В.

АННОТАЦИЯ

Курсовая работа посвящена разработке утилиты для обработки PNG-изображений в Linux-стиле. Используются методы парсинга аргументов (getopt), обработки изображений (libpng), валидации данных и исключений. Реализованы функции: отражение областей, рисование геометрических фигур (пентаграмма, прямоугольник, шестиугольник) с настройкой параметров. Программа корректно обрабатывает ошибки, поддерживает справку и сохранение в указанный файл.

SUMMARY

The coursework is devoted to the development of a utility for processing PNG images in the Linux style. The methods of argument parsing (getopt), image processing (libpng), data validation and exceptions are used. The following functions are implemented: reflection of areas, drawing geometric figures (pentagram, rectangle, hexagon) with parameter settings. The program correctly processes errors, supports help and saving to the specified file.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: создать CLI-утилиту для обработки PNG-изображений с поддержкой базовых графических операций

Задачи работы:

1. Реализовать обработку аргументов командной строки
2. Организовать базовую работу с PNG-изображением (чтение и вывод)
3. Реализовать функции рисования
4. Реализовать функции изменения изображения
5. Организовать обработку ошибок
6. Настроить сборку через Makefile
7. Протестировать программу и внести исправления

1. ОПИСАНИЕ ПРОГРАММЫ

1.1. Организация хранения данных

Для удобства работы созданы вспомогательные структуры point и colorRGB. Структура point предназначена для записи координат точки, содержит поля x и y типа int. Структура colorRGB предназначена для хранения информации о цвете в формате RGB, содержит поля r, g, b типа int для хранения значений каждой цветовой компоненты.

Для хранения данных о вводимых аргументах создана структура argStruct. В ней содержатся «поля-флаги» равные ABESNT = 0 в случае отсутствия данного аргумента во вводе и PRESENT = 1 при наличии, поля типа char* для хранения имён входного и выходного файлов, поля типа int для хранения информации о числовых данных (таких как толщина линии, радиус), поле типа char для информации об оси, поля типа colorRGB* для хранения информации о цветах (цвет линий, цвет заливки), поля типа point* для хранения точек (крайних точек, центра).

Для хранения данных изображения используется структура PNG. Содержит поля типов png_infop и png_structp для хранения информации об изображении (структур для чтения и записи изображения), поля типа int для записи размеров изображения и количества проходов, поля типа png_byte для хранения информации о пикселях и поле типа png_bytep* для хранение непосредственно информации каждого пикселя изображения.

1.2. Общая схема работы программы

Программа разбита на блоки, выполняющие мелкие задачи. Функции собраны в отдельные файлы, сборка программы происходит с помощью Makefile.

В основной функции осуществляется вызов функции считывания аргументов, считывания и вывода изображения и функций выполнения опций программы. Кроме того, в ней выделяется динамическая память для хранения входных данных.

Сначала происходит вызов функции получения аргументов командной строки, далее при необходимости вызывается функция чтения входного

изображения. В зависимости от полученных аргументов, происходит вызов необходимых функций. При корректной отработке опций происходит вызов функции записи изображения.

В следующих разделах подробно описана работа каждой составляющей программы.

1.3. Обработка ошибок

Для удобной обработки ошибок создано перечисление error_types, содержащее флаги успешного выполнения (ERR_NO = 0) и различных типов ошибок: ошибка памяти, работы с файлом, аргумента командной строки, работы с изображением, ошибка структуры изображения, а также флаг некорректного значения, флаги ошибок лежат в диапазоне *[40; 46]*.

В функциях создаётся переменная-флаг корректности работы. В критических моментах (выделение динамической памяти, получение аргументов командной строки, открытие файла, чтение структуры входного изображение, создание структуры выходного изображения и тп) проводятся проверки корректности. В случае отлова ошибки в стандартный поток ошибок выводится текстовое сообщение об ошибке и значение переменной-флага обновляется в зависимости от полученных данных, функция завершается, возвращая флаг полученной ошибки.

В основной функции происходит проверка корректности выполнения каждой отдельной части работы. Программа завершается, возвращая полученный флаг ошибки, либо флаг успешного завершения.

1.4. Обработка аргументов командной строки

Обработка аргументов проводится с помощью getopt. Для чтения создана структура опций. Для каждого аргумента задано значение наличие флага, полное и сокращённое названия. Функция обработки аргументов принимает на вход указатели на количество аргументов командной строки, массив аргументов и структуру хранения входных данных. Возвращает значение error_types. Флаги операций по умолчанию устанавливаются в 0, выделяется память под поля цветов и точек.

Чтение аргументов происходит через getopt_long, так как он может считать как длинные, так и короткие флаги. При получении неизвестного флага функция завершается с ошибкой аргумента. При получении корректной опции проверяется корректность необходимого аргумента (или отсутствие, если опция не принимает аргументов) и обновляются соответствующие данной опции поля в структуре входных данных. Для опций, принимающих на вход аргументы, проверяется также корректность следующего аргумента командной строки (это должен быть либо следующий флаг, либо последний аргумент командной строки, который по заданию может обозначать входной файл).
Ожидаемые на вход флаги соответствуют опциям из задания:

1. `--input`, `-i`: Задаёт имя входного изображения. Если флаг отсутствует, то предполагается, что имя входного изображения передаётся последним аргументом

2. `--output`, `-o`: Задаёт имя выходного изображения. Если флаг отсутствует, имя выходного файла задаётся «out.png»

3. `--info`: Печатает информацию об изображении

4. `--help`, `-h`: Выводит справку

5. `--mirror`: Отражение заданной области. Этот функционал определяется:

   - Выбором оси относительно которой отражать (горизонтальная или вертикальная). Флаг --axis, возможные значения «x» и «y»
   - 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

6. `--pentagram`: Рисование пентаграммы в круге. Пентаграмма определяется:

   - Координатами ее центра и радиусом. Флаги `--center` и `--radius`. Значение флаг `--center` задаётся в формате «x.y», где x – координата по оси

x, y – координата по оси y. Флаг `--radius` на вход принимает число больше 0

- Толщиной линий и окружности. Флаг `--thickness`. На вход принимает число больше 0
- Цветом линий и окружности. Флаг `--color` (цвет задаётся строкой «rrr.ggg.bbb», где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример «`--color 255.0.0`» задаёт красный цвет)

7. `--rect`: Рисование прямоугольника. Он определяется:
   - Координатами левого верхнего угла. Флаг `--left_up`
   - Координатами правого нижнего угла. Флаг `--right_down`
   - Толщиной линий. Флаг `--thickness`
   - Цветом линий. Флаг --`color`
   - 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`)
8. `--hexagon`: Рисование правильного шестиугольника. Он определяется:
   - Координатами его центра и радиусом в который он вписан. Флаги `--center` и `--radius`
   - Толщиной линий. Флаг `--thickness`
   - Цветом линий. Флаг `--color`
   - Шестиугольник может быть залит или нет. Флаг `--fill`
   - Цветом которым залит шестиугольник, если пользователем выбран залитый. Флаг `--fill_color`

После чтения аргументов заполняются поля входного и выходного файлов (если не были поданы соответствующие флаги), выполняется проверка корректности введённых данных: наличие всех необходимых аргументов для требуемой опции и отсутствие лишнего. При необходимости выполняется

корректировка координат левой верхней и правой нижней точек для дальнейшей работы.

1.5. Считывание и вывод изображения

Работа с изображением происходит через библиотеку libpng. Функции считывания и вывод изображения принимают на вход указатель на структуру PNG для хранения изображения и путь до входного или выходного файла соответственно. Возвращают значение error_types.

Считывание изображения. Из переданного файла, открытого в бинарном режиме чтения, читается PNG_HEADER, выполняется проверка сигнатуры файла на соответствие формату png, инициализируются структуры для чтения изображения и хранения информации об изображении. Выполняется проверка корректности создания структуры. Читается заголовок изображения, извлекаются основные параметры (ширина и высота, глубина цвета, тип цвета, метод фильтрации, тип сжатия, количество проходов). Далее создаётся массив указателей на строки изображения, записывается информация о каждой строке изображения.

Запись изображения в файл. Инициализируются структуры для записи изображения и хранения информации об изображении. Устанавливаются основные параметры изображения. Заголовок и данные изображения записываются в переданный файл, открытый в бинарном режиме записи. Очищаются структуры изображения.

В случае ошибки на каком-либо этапе работы функций, освобождается ранее использованная память и возвращается код ошибки.

1.6. Функции рисования

Для выполнения необходимых по заданию опций рисования созданы вспомогательные функции изменения цвета пикселя, заливки многоугольника, проверки принадлежности линии, функции рисования линии и круга, а также функция поиска координат вершин правильного многоугольника.

Функция изменения цвета пикселя. Void-функция, принимает на вход указатель на изображение, координаты пикселя и цвет в формате colorRGB*.

Выполняется проверка корректности переданных координат, обновляется информация соответствующих цветовых компонент нужного пикселя изображения.

Функция заливки многоугольника. Void-функция, принимает на вход указатель на изображение, указатель на список вершин (point**), количество вершин многоугольника, толщину линий и цвет заливки (colorRGB*). Выполняется поиск крайних точек многоугольника по осям x, y. Для каждого пикселя из прямоугольника, образованного крайними точками выполняется проверка на принадлежность линии (вызов функции проверки принадлежности линии) и с помощью алгоритма подсчёта числа пересечений сторон проверяется лежит ли пиксель внутри фигуры. Если пиксель лежит внутри области и не на линии, вызывается функции изменения цвета данного пикселя. Алгоритм подсчёта числа пересечений сторон. Изначально предполагаем, что пиксель лежит вне фигуры. Для каждой стороны (заданной двумя соседними вершинами) проверяются условия: 1) y должен лежать между нижней и верхней вершинами; 2) x должен лежат слева от данной стороны. Если оба условия выполняются, меняем значение принадлежности на противоположное. После проверки всех сторон, значение принадлежности будет зависеть от количества пересечений — при нечётном истина, при чётном — ложь.

Функция проверки принадлежности пикселя линии. Принимает на вход координаты проверяемого пикселя, координаты крайних точек отрезка в формате point* и толщину линии. Возвращает 1 если точка принадлежит линии и 0 в противном случае. С помощь вычисления проекции пикселя на линию (через скалярное произведение векторов (p2-p1) и (точка-p1)) проверяем, попадает ли пиксель на нужный отрезок, полученное значение должно лежать в промежутке от 0 до квадрата длины отрезка. Затем вычисляем расстояние от пикселя до прямой, оно должно быть не больше половины толщины линии. При выполнении обоих условий пиксель принадлежит заданной стороне.

Функция рисования линии. Void-функция, принимает на вход указатель на изображение, координаты крайних точек линии в формате point*, толщину линии

и цвет в формате colorRGB*. Для каждого пикселя в прямоугольнике от минимальной координаты y из вершин минус половина толщины до максимальной с прибавлением половины толщины и аналогичными краями по x проверяется принадлежность линии, принадлежащие линии пиксели закрашиваются в нужный цвет (вызов функции покраски пикселя).

Функция рисования круга. Void-функция, принимает на вход указатель на изображение, радиус, толщину линии, координаты центра в формате point* и цвет в формате colorRGB*. Проверяются пиксели в квадратной области вокруг центра со стороной равной радиусу, увеличенному на половину толщины. Для каждого пикселя высчитывается расстояние до центра, если расстояние отличается от радиуса не более, чем на половину толщины, пиксель закрашивается.

Функция поиска вершин правильного многоугольника. Void-функция, принимает на вход указатель на массив вершин типа point**, координаты центра в формате point*, радиус и количество вершин. Выделятся память под массив вершин. Далее непосредственно вычисляются координаты вершин. Вычисляется начальная фаза по формуле $p=(-\pi/2)*(n\%2)$ (где n - число вершин), необходимая для того, чтобы верхняя грань многоугольников с чётным числом вершин была горизонтальной, а у многоугольников с нечётным числом вершин одна вершина находилась ровно над центром. Координаты каждой вершины вычисляются в цикле по формулам $x[i]=x_c+r*\cos(2i\pi/n+p)$, $y[i]=y_c+r*\sin(2i\pi/n+p)$, где x_c, y_c - координаты центра, r - радиус

Для выполнения опций рисования по заданию созданы функции рисования пентаграммы в круге, прямоугольника и правильного шестиугольника. Функции принимают на вход указатель на изображение (PNG*) и указатель на структуру с входными данными. Возвращают флаг ошибки или корректного выполнения (error_types).

Функция рисования пентаграммы в круге. Вызывается функция рисования круга с необходимыми данными и функция поиска координат вершин. Затем вызывается функция рисования линий, соединяющих найденные вершины через 1 для получения пентаграммы.

Функция рисования прямоугольника. Создаётся и заполняется массив вершин, две вершины передаются изначально (верхняя левая и правая нижняя точки), две оставшиеся вершины получаются соединением координат по разным осям из верхней левой и правой нижней вершин. В цикле вызываются функции рисования линии для каждой пары соседних вершин и функция рисования круга с радиусом равным четверти толщины и толщиной линии в половину толщины с центром в каждой вершине для рисования углов. Если задана заливка, вызывается функция заливки многоугольника.

Функция рисования правильного шестиугольника. Вызывается функция поиска координат вершин, затем в цикле вызываются функции рисования линий для каждой пары соседних вершин и функция рисования круга с радиусом равным четверти толщины линии и толщиной линии в половину толщины с центром в каждой вершине для рисования углов. Если задана заливка, вызывается функция заливки многоугольника.

1.7. Функции изменения изображения

Для изменения изображения создана вспомогательная функция перестановки двух пикселей. Void-функция, принимает на вход указатель на изображение (PNG*) и координаты двух пикселей. Выполняется проверка корректности переданных координат. Через прямой доступ к данным пикселей переставляются местами все цветовые компоненты.

Функция отражения заданной области. Принимает на вход указатель на изображение (PNG*) и указатель на структуру с входными данными. Возвращает флаг ошибки или корректного выполнения (error_types). Если задана ось отражения y, Запускается цикл по y от меньшей координаты до середины между координатами по оси y, и по x от меньшей до большей из заданных координат включительно. Для каждого пикселя вызывается функция перестановки с пикселем имеющим ту же координату x и координату по y равную $y_{max} + y_{min} - y$, где y_{max} и y_{min} — границы отражаемой области по y, y - координата текущего пикселя. Для отражения вдоль оси x используется аналогичный алгоритм.

1.8. Вывод справочной информации

Вывод информации об изображении. Для вывода информации об изображении используется void-функция, принимающая на вход указатель на изображение (PNG*). Функция печатает в консоль краткую основную информацию о входном изображении.

Вывод справки. Для вывода справки используется void-функция, не принимающая аргументов на вход. В консоль печатается краткая информация о программе (доступные опции и необходимые флаги)

2. ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

2.1. Примеры корректных запросов

1. Проверка опции вывода справки

Таблица 1.1. Длинный флаг

Ввод	`./cw --help`
Вывод	Course work for option 5.15, created by Varvara Chetvertnaya --Functions information-- --help, -h: function information. Optional flag --info: image information. Optional flag --input, -i: input file name. Optional flag, defaults to the last argument entered as the input file name --output, -o: output file name. Optional flag, defaults the result is saved to the out.png file --mirror: reflection of a given area. Flags: --left_up: coordinates of the upper left corner. The value is specified 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 specified in the format 'right.down', where 'right' is the x-coordinate, 'down' is the y-coordinate --pentagram: drawing a pentagram in a circle. Flags: --center: center coordinates. The value is specified in the format 'x.y', where 'x' is the x-coordinate, 'y' is the y-coordinate --radius: radius. Accepts a number greater than 0 as input --thickness: line and circle thickness. Accepts a number greater than 0 as input --color: line and circle color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component --rect: drawing a rectangle. Flags: --left_up: coordinates of the upper left corner. The value is specified 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 specified in the format 'right.down', where 'right' is the x-coordinate, 'down' is the y-coordinate --thickness: line thickness. Accepts a number greater than 0 as input --color: line color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component --fill: rectangle fill. Works as a binary value: no flag - false, flag present - true --fill_color: rectangle fill color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component --hexagon: drawing a regular hexagon. Flags: --center: center coordinates. The value is specified in the format 'x.y', where 'x' is the x-coordinate, 'y' is the y-coordinate --radius: radius. Accepts a number greater than 0 as input --thickness: line thickness. Accepts a number greater than 0 as input --color: line color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component --fill: hexagon fill. Works as a binary value: no flag - false, flag present - true --fill_color: hexagon fill color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component

Таблица 1.2. Короткий флаг

Ввод	`./cw -h`
Вывод	Course work for option 5.15, created by Varvara Chetvertnaya --Functions information-- --help, -h: function information. Optional flag --info: image information. Optional flag --input, -i: input file name. Optional flag, defaults to the last argument entered as the input file name --output, -o: output file name. Optional flag, defaults the result is saved to the out.png file --mirror: reflection of a given area. Flags: --left_up: coordinates of the upper left corner. The value is specified 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 specified in the format 'right.down', where 'right' is the x-coordinate, 'down' is the y-coordinate --pentagram: drawing a pentagram in a circle. Flags: --center: center coordinates. The value is specified in the format 'x.y', where 'x' is the x-coordinate, 'y' is the y-coordinate --radius: radius. Accepts a number greater than 0 as input --thickness: line and circle thickness. Accepts a number greater than 0 as input --color: line and circle color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component --rect: drawing a rectangle. Flags: --left_up: coordinates of the upper left corner. The value is specified 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 specified in the format 'right.down', where 'right' is the x-coordinate, 'down' is the y-coordinate --thickness: line thickness. Accepts a number greater than 0 as input --color: line color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component --fill: rectangle fill. Works as a binary value: no flag - false, flag present - true --fill_color: rectangle fill color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component --hexagon: drawing a regular hexagon. Flags: --center: center coordinates. The value is specified in the format 'x.y', where 'x' is the x-coordinate, 'y' is the y-coordinate --radius: radius. Accepts a number greater than 0 as input --thickness: line thickness. Accepts a number greater than 0 as input --color: line color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component --fill: hexagon fill. Works as a binary value: no flag - false, flag present - true --fill_color: hexagon fill color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component

2. Проверка опции рисования пентаграммы в круге

Таблица 2.1. С полными флагами входного и выходного файлов

Ввод	`./cw --pentagram --output ./output.png --radius 392 --color 50.87.140 --center 76.110 --input ./input/bird.png --thickness 45`

Вывод	`Course work for option 5.15, created by Varvara Chetvertnaya` (входное изображение см. Приложение Б. Рисунок 2.1.1, выходное изображение см. Приложение Б. Рисунок 2.1.2)

Таблица 2.2. Без флагов входного и выходного файлов

Ввод	`./cw --radius 100 --pentagram --center 250.150 --thickness 15 --color 200.104.154 ./input/car.nST`
Вывод	`Course work for option 5.15, created by Varvara Chetvertnaya` (входное изображение см. Приложение Б. Рисунок 2.2.1, выходное изображение см. Приложение Б. Рисунок 2.2.2)

3. Проверка опции рисования прямоугольника

Таблица 3.1. Без заливки

Ввод	`./cw -o rect_out.Gfe --rect --right_down 200.150 --left_up 20.80 --color 40.89.21 --thickness 10 ./input/milk.XyA`
Вывод	`Course work for option 5.15, created by Varvara Chetvertnaya` (входное изображение см. Приложение Б. Рисунок 3.1.1, выходное изображение см. Приложение Б. Рисунок 3.1.2)

Таблица 3.2. С заливкой

Ввод	`./cw --rect --right_down 300.300 --fill_color 45.120.45 --left_up -20.10 --fill --color 120.45.120 --thickness 10 ./input/eleph.png`
Вывод	`Course work for option 5.15, created by Varvara Chetvertnaya` (входное изображение см. Приложение Б. Рисунок 3.2.1, выходное изображение см. Приложение Б. Рисунок 3.2.2)

4. Проверка опции рисования шестиугольника

Таблица 4.1. Фигура полностью на изображении

Ввод	`./cw --fill_color 123.104.238 --color 139.0.139 --fill --hexagon -i ./input/radio.vQh --center 150.150 --radius 120 --thickness 24`
Вывод	`Course work for option 5.15, created by Varvara Chetvertnaya` (входное изображение см. Приложение Б. Рисунок 4.1.1, выходное

	изображение см. Приложение Б. Рисунок 4.1.2)

Таблица 4.2. С выходом за границы изображения

Ввод	`./cw --color 255.69.0 --fill --fill_color 173.255.47 --hexagon --center 500.400 --radius 200 --thickness 12 ./input/radio.vQh`
Вывод	`Course work for option 5.15, created by Varvara Chetvertnaya` (входное изображение см. Приложение Б. Рисунок 4.2.1, выходное изображение см. Приложение Б. Рисунок 4.2.2)

5. Проверка опции отражения участка

Таблица 5.1. Отражение вдоль оси x

Ввод	`./cw --mirror --axis x --left_up -100.20 --right_down 300.300 ./input/eleph.png`
Вывод	`Course work for option 5.15, created by Varvara Chetvertnaya` (входное изображение см. Приложение Б. Рисунок 5.1.1, выходное изображение см. Приложение Б. Рисунок 5.1.2)

Таблица 5.2. Отражение вдоль оси y

Ввод	`./cw --axis y --mirror --left_up 100.0 --right_down 450.370 ./input/bird.png`
Вывод	`Course work for option 5.15, created by Varvara Chetvertnaya` (входное изображение см. Приложение Б. Рисунок 5.2.1, выходное изображение см. Приложение Б. Рисунок 5.2.2)

2.2. Примеры обработки ошибочных случаев

Таблица 6.1. Некорректный ввод

Ввод	`./cw --effbnewrb weffvqerrg -qergwre`
Вывод	`./cw: unrecognized option '--effbnewrb'` `Incorrect argument`

Таблица 6.2. Отсутствие обязательного флага

Ввод	`./cw --axis y --mirror --right_down 450.370 ./input/bird.png`
Вывод	`Missing required flags`

Таблица 6.3. Ошибочный входной файл

Ввод	`./cw --rect --info --right_down 300.300 --fill_color 45.120.45 --left_up -20.10 --fill --color 120.45.120 --thickness 10 ./input/help_me_please.odt`
Вывод	`Incorrect input file`

Таблица 6.4. Введённый лишний параметр

Ввод	`./cw --fill --radius 100 --pentagram --center 250.150 --thickness 15 --color 200.104.154 ./input/car.nST`
Вывод	`Unnecessary parameters`

Таблица 6.5. Подача неверного значения аргумента

Ввод	`./cw --rect --info --right_down 300.300 --fill_color 45.120.45 --left_up -20.10 --fill --color 400.45.120 --thickness 10 ./input/fruit.png`
Вывод	`Incorrect color`

Таблица 6.6. Подача аргумента к флагу, не принимающему аргументы

Ввод	`./cw --info цукпуцк ./input/bird.png`
Вывод	`This option does not take arguments`

Таблица 6.7. Подача лишнего аргумента

Ввод	`./cw --axis y --mirror --left_up 100.0 how_are_you? --right_down 450.370 ./input/bird.png`

Вывод	`Excess arguments`

ЗАКЛЮЧЕНИЕ

В рамках курсовой работы создана CLI-утилита для обработки PNG-изображений. Реализована обработка аргументов командной строки, организована базовая работа с PNG-изображениями, разработаны функции рисования пентаграммы в круге, прямоугольника и шестиугольника, отражения заданной области. Организована обработка ошибок, все функции собраны в отдельные файлы, написана сборка через Makefile. Программа протестирована, обнаруженные ошибки устранены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Базовые сведения к выполнению курсовой работы по дисциплине «Программирование». Второй семестр / сост.: М. М. Заславский, А. А. Лисс, А. В. Гаврилов и др.. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2024. 36 с.

2. A description on how to use and modify libpng. URL: http://www.libpng.org/pub/png/libpng-1.2.5-manual.html#section-2 (дата обращения: 02.05.2025)

3. «Язык программирования Си» Издание 3-е, исправленное/Б. Керниган, Д. Ритчи. Спб.: "Невский Диалект", 2001. - 352 с

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

Файл Makefile

```
all: main.o ./src/arguments.o ./src/picture.o ./src/drawing.o ./src/process.o ./src/help.o
	gcc main.o ./src/arguments.o ./src/picture.o ./src/drawing.o ./src/process.o ./src/help.o -lpng -lm -o cw

main.o: main.c ./src/arguments.h ./src/errors.h ./src/picture.h ./src/drawing.h ./src/process.h ./src/help.h
	gcc -c main.c

arguments.o: ./src/arguments.c ./src/arguments.h ./src/errors.h
	gcc -c ./src/arguments.c

picture.o: ./src/picture.c ./src/picture.h ./src/errors.h
	gcc -c ./src/picture.c

drawing.o: ./src/drawing.c ./src/drawing.h ./src/picture.h ./src/arguments.h ./src/errors.h
	gcc -c ./src/drawing.c

process.o: ./src/process.c ./src/process.h ./src/picture.h ./src/arguments.h ./src/errors.h
	gcc -c ./src/process.c

help.o: ./src/help.c ./src/help.h
	gcc -c ./src/help.c

clean:
	rm -rf *.o
```

Файл main.c

```
#include "./src/arguments.h"
#include "./src/picture.h"
#include "./src/drawing.h"
#include "./src/process.h"
#include "./src/help.h"

int main(int argc, char** argv){
    enum error_types err=ERR_NO;
```

```
    argStruct* data=(argStruct*)calloc(1, sizeof(argStruct));
      /*освобождение происходит в main(...), внутри if (!err) после выполнения основной программы;
        ошибка на данном этапе может быть поймана только на выделении памяти под data*/
    if (!data){
        fprintf(stderr, "Memory error\n");
        err = ERR_MEMORY;
    }

    if (!err){
        err = read_arguments(&argc, &argv, data);

        if (!err){
            PNG* img=(PNG*)calloc(1, sizeof(PNG));
            /*освобождение происходит в main(...), после выполнения всех действий с изображением;
                    проверка корректности выделения находится на том же уровне вложенности, что и само выделение памяти*/
            if (!img){
                fprintf(stderr, "Memory error\n");
                err = ERR_MEMORY;
            }

            if (data->in && !err){
                err = read_png(img, data->input);
            }

            if (!err){
                    printf("Course work for option 5.15, created by Varvara Chetvertnaya\n");

                if (data->help)
                    help_print();
                if (data->in){
                    if (data->info)
                        info_print(img);
                    else if (data->mirror)
                        err = mirror_draw(img, data);
                    else if (data->pentagram)
                        err = pentagram_draw(img, data);
```

```
                    else if (data->rect)
                        err = rect_draw(img, data);
                    else if (data->hexagon)
                        err = hexagon_draw(img, data);
                    else if (data->compress)
                        err = compress_img(img, data);

                         if (!err && data->mirror+data->pentagram+data->rect+data-
>hexagon+data->compress)
                        err = write_png(img, data->output);
                }
                  else if (data->info+data->mirror+data->pentagram+data->rect+data-
>hexagon+data->compress > 0){
                    fprintf(stderr, "Input file missing\n");
                    err = ERR_FILE;
                }
            }

            if (img){
                if (img->row_pointers){
                    for (int i=0; i < img->height; i++){
                        if (img->row_pointers[i])
                            free(img->row_pointers[i]);
                    }
                    free(img->row_pointers);
                }
                free(img);
            }
        }

        if (data->left_up)
            free(data->left_up);
        if (data->right_down)
            free(data->right_down);
        if (data->center)
            free(data->center);
        if (data->color)
            free(data->color);
        if (data->output)
            free(data->output);
        if (data->fill_color)
            free(data->fill_color);
```

```
        if (data->input)
            free(data->input);

        free(data);
    }

    return err;
}
```

Файл process.c

```
#include "process.h"

void swap_pixel(PNG* img, int x1, int y1, int x2, int y2){
    if (x1 >= 0 && x1 < img->width && y1 >= 0 && y1 < img->height
        && x2 >= 0 && x2 < img->width && y2 >= 0 && y2 < img->height){
        png_bytep row1=img->row_pointers[y1], row2=img->row_pointers[y2];
        png_bytep p1=&(row1[x1*RGB_LEN]), p2=&(row2[x2*RGB_LEN]);

        for (int i=0; i < RGB_LEN; i++){
            png_byte tmp=p1[i];
            p1[i] = p2[i];
            p2[i] = tmp;
        }
    }
}

int mirror_draw(PNG* img, argStruct* data){
    if (data->axis == 'y'){
                                    for    (int    y=data->left_up->y;    y    <=
(data->right_down->y+data->left_up->y)/2; y++){
            for (int x=data->left_up->x; x <= data->right_down->x; x++)
                swap_pixel(img, x, y, x, data->left_up->y+data->right_down->y-y);
        }
    }

    if (data->axis == 'x'){
        for (int y=data->left_up->y; y <= data->right_down->y; y++){
             for (int x=data->left_up->x; x <= (data->right_down->x+data->left_up-
>x)/2; x++)
                swap_pixel(img, x, y, data->right_down->x+data->left_up->x-x, y);
        }
    }
    return ERR_NO;
```

```
}

int compress_img(PNG* img, argStruct* data){
    enum error_types err=ERR_NO;

    int new_width = img->width/data->num;
    int new_height = img->height/data->num;

    png_bytep* new_img = (png_bytep*)calloc(new_height, sizeof(png_bytep));
    if (!new_img){
        fprintf(stderr, "Memory error\n");
        err = ERR_MEMORY;
    }
    for (int y=0; !err && y < new_height; y++){
        new_img[y] = (png_byte*)calloc(1, new_width*RGB_LEN);
        if (!new_img[y]){
            fprintf(stderr, "Memory row error\n");
            err = ERR_MEMORY;
        }
    }

    for (int y=0; !err && y < new_height; y++){
        for (int x=0; x < new_width; x++){
            int r=0, g=0, b=0;

            for (int i=y*data->num; i < (y+1)*data->num; i++){
                for (int j=x*data->num; j < (x+1)*data->num; j++){
                    png_bytep row=img->row_pointers[i];
                    png_bytep p=&(row[j*RGB_LEN]);
                    r += p[0]; g += p[1]; b += p[2];
                }
            }
              r /= (data->num*data->num); g /= (data->num*data->num); b /= (data-
>num*data->num);

            png_bytep new_row=new_img[y];
            png_bytep new_p=&(new_row[x*RGB_LEN]);
            new_p[0] = r; new_p[1] = g; new_p[2] = b;

        }
    }
```

```
    if (!err){
        img->width = new_width;
        img->height = new_height;

                img->row_pointers  =  (png_bytep*)realloc(img->row_pointers,  img-
>height*sizeof(png_bytep));
        if (!img->row_pointers){
            fprintf(stderr, "Memory error\n");
            err = ERR_MEMORY;
        }
        for (int y=0; !err && y < img->height; y++){
              img->row_pointers[y] = (png_byte*)realloc(img->row_pointers[y], img-
>width*RGB_LEN);
            if (!img->row_pointers[y]){
                fprintf(stderr, "Memory row error\n");
                err = ERR_MEMORY;
            }
        }

        for (int y=0; !err && y < img->height; y++){
            for (int x=0; x < img->width; x++){
                png_bytep row1=img->row_pointers[y], row2=new_img[y];
                png_bytep p1=&(row1[x*RGB_LEN]), p2=&(row2[x*RGB_LEN]);
                p1[0] = p2[0]; p1[1] = p2[1]; p1[2] = p2[2];
            }
        }
    }

    if (new_img){
        for (int y=0; y < new_height; y++){
            if (new_img[y])
                free(new_img[y]);
        }
        free(new_img);
    }

    return err;
}
```

Файл process.h

```
#ifndef PROCESS
#define PROCESS
```

```
#include "picture.h"
#include "arguments.h"

void swap_pixel(PNG* img, int x1, int y1, int x2, int y2);
int mirror_draw(PNG* img, argStruct* data);
int compress_img(PNG* img, argStruct* data);

#endif
```

Файл drawing.c

```
#include "drawing.h"

void repaint_pixel(PNG* img, int x, int y, colorRGB* color){
    if (x >= 0 && x < img->width && y >= 0 && y < img->height){
        png_bytep row=img->row_pointers[y];
        png_bytep ptr=&(row[x*RGB_LEN]);

        ptr[0] = color->r; ptr[1] = color->g; ptr[2] = color->b;
    }
}

void fill_area(PNG* img, point** vert, int cnt_vert, int thick, colorRGB* color){
    int x_min=vert[0]->x, x_max=vert[0]->x;
    int y_min=vert[0]->y, y_max=vert[0]->y;
    for (int i=1; i < cnt_vert; i++){
        x_min = (vert[i]->x < x_min ? vert[i]->x : x_min);
        x_max = (vert[i]->x > x_max ? vert[i]->x : x_max);

        y_min = (vert[i]->y < y_min ? vert[i]->y : y_min);
        y_max = (vert[i]->y > y_max ? vert[i]->y : y_max);
    }

    for (int y=y_min; y <= y_max; y++){
        for (int x=x_min; x <= x_max; x++){
            int board=0;
            for (int i=0; i < cnt_vert && !board; i++)
                board = check_line(x, y, vert[i], vert[(i+1)%cnt_vert], thick);

            int inside=0;
            for (int i=0; i < cnt_vert; i++){
                int xi=vert[i%cnt_vert]->x, yi=vert[i%cnt_vert]->y;
```

```
                int xj=vert[(i+1)%cnt_vert]->x, yj=vert[(i+1)%cnt_vert]->y;

                if ((yi > y) != (yj > y) && (x < (xj-xi)*(y-yi)/(yj-yi)+xi))
                    inside = !inside;
            }

            if (!board && inside)
                repaint_pixel(img, x, y, color);
        }
    }
}

int check_line(int x, int y, point* p1, point* p2, int thick){
    int is_line=0;
    int seg=(p2->x-p1->x)*(x-p1->x)+(p2->y-p1->y)*(y-p1->y);
    int sq_len=(p2->x-p1->x)*(p2->x-p1->x)+(p2->y-p1->y)*(p2->y-p1->y);
    int dist=abs((p2->y-p1->y)*(x-p1->x)-(p2->x-p1->x)*(y-p1->y))/sqrt(sq_len);

    if (0 <= seg && seg <= sq_len && dist <= thick/2)
        is_line = 1;
    return is_line;
}

void draw_line(PNG* img, point* p1, point* p2, int thick, colorRGB* color){
    int x_min=(p1->x < p2->x ? p1->x : p2->x), x_max=(p1->x > p2->x ? p1->x : p2-
>x);
    int y_min=(p1->y < p2->y ? p1->y : p2->y), y_max=(p1->y > p2->y ? p1->y : p2-
>y);

    for (int y=y_min-thick/2; y <= y_max+thick/2; y++){
        for (int x=x_min-thick/2; x <= x_max+thick/2; x++){
            if (check_line(x, y, p1, p2, thick))
                repaint_pixel(img, x, y, color);
        }
    }
}

void draw_circle(PNG* img, int radius, int thick, point* center, colorRGB* color){
    for (int y=-radius-thick/2; y <= radius+thick/2; y++){
        for (int x=-radius-thick/2; x <= radius+thick/2; x++){
            int d=sqrt(x*x+y*y);
            if (d >= radius-thick/2 && d <= radius+thick/2)
```

```
                repaint_pixel(img, center->x+x, center->y+y, color);
        }
    }
}

void find_vertex(point*** vert, point* center, int radius, int cnt_vert){
    *vert = (point**)calloc(cnt_vert, sizeof(point*));
        /*освобождение происходит непосредственно в функциях рисования, откуда вызывается данная функция;
             сразу после вызова find_vertex(...), после выполнения необходимых дейтсвий, но независимо от них, выполняется проверка корректности выделения vert, проходит очистка*/
    if (!(*vert)){
        fprintf(stderr, "Memory error\n");
    }
    else{
        int err=0;
        double phase=(-M_PI/2)*(cnt_vert%2);
        for (int k=0; !err && k < cnt_vert; k++){
            (*vert)[k] = (point*)calloc(1, sizeof(point));
             /*освобождение происходит непосредственно в функциях рисования, откуда вызывается данная функция, перед очисткой vert;
                 организовано аналогично с очисткой vert, проверяется существование конкретного vert[k] и его очистка*/
            if (!(*vert)[k]){
                fprintf(stderr, "Memory error\n");
                err = ERR_MEMORY;
            }
            if (!err){
                double ang=2*M_PI*k/cnt_vert+phase;
                (*vert)[k]->x = center->x+radius*cos(ang);
                (*vert)[k]->y = center->y+radius*sin(ang);
            }
        }
    }
}

int pentagram_draw(PNG* img, argStruct* data){
    enum error_types err=ERR_NO;

    draw_circle(img, data->radius, data->thick, data->center, data->color);
```

```
    point** vert;
    find_vertex(&vert, data->center, data->radius, PENT_VERT);

    if (!vert){
        err = ERR_MEMORY;
    }
    else{
        for (int i=0; !err && i <= PENT_VERT/2; i++){
            if (!vert[i] || !vert[i+2] || !vert[i+3])
                err = ERR_MEMORY;

            if (!err){
                draw_line(img, vert[i], vert[i+2], data->thick, data->color);
                if (i+3 < PENT_VERT)
                    draw_line(img, vert[i], vert[i+3], data->thick, data->color);
            }
        }

        for (int i=0; i < PENT_VERT; i++){
            if (vert[i])
                free(vert[i]);
        }
        free(vert);
    }

    return err;
}

int rect_draw(PNG* img, argStruct* data){
    enum error_types err=ERR_NO;

    point** vert=(point**)calloc(RECT_VERT, sizeof(point*));
    /*освобождение происходит в rect_draw(...), после выполнения основных действий;
          проверка корректности выделения находится на том же уровне вложенности,
что и само выделение памяти*/
    if (!vert){
        fprintf(stderr, "Memory error\n");
        err = ERR_MEMORY;
    }

    for (int i=0; !err && i < RECT_VERT; i++){
        vert[i] = (point*)calloc(1, sizeof(point));
```

```
            /*освобождение происходит в rect_draw(...), после выполнения основных действий, перед очисткой vert;
                    проверка корректности выделения находится на том же уровне вложенности, что и само выделение памяти*/
        if (!vert[i]){
            fprintf(stderr, "Memory error\n");
            err = ERR_MEMORY;
        }
    }

    if (!err){
        vert[0]->x = data->left_up->x; vert[0]->y = data->left_up->y;
        vert[1]->x = data->right_down->x; vert[1]->y = data->left_up->y;
        vert[2]->x = data->right_down->x; vert[2]->y = data->right_down->y;
        vert[3]->x = data->left_up->x; vert[3]->y = data->right_down->y;

        for (int i=0; i < RECT_VERT; i++){
                draw_line(img, vert[i], vert[(i+1)%RECT_VERT], data->thick, data->color);
            draw_circle(img, data->thick/4, data->thick/2, vert[i], data->color);
        }

        if (data->fill){
            fill_area(img, vert, RECT_VERT, data->thick, data->fill_color);
        }
    }

    if (vert){
        for (int i=0; i < RECT_VERT; i++){
            if (vert[i])
                free(vert[i]);
        }
        free(vert);
    }

    return err;
}

int hexagon_draw(PNG* img, argStruct* data){
    enum error_types err=ERR_NO;

    point** vert;
```

```
    find_vertex(&vert, data->center, data->radius, HEX_VERT);

    if (!vert){
        fprintf(stderr, "Memory error\n");
        err = ERR_MEMORY;
    }
    else{
        for (int i=0; !err && i < HEX_VERT; i++){
            if (!vert[i] || !vert[(i+1)%HEX_VERT])
                err = ERR_MEMORY;
            if (!err){
                  draw_line(img, vert[i], vert[(i+1)%HEX_VERT], data->thick, data-
>color);
                    draw_circle(img, data->thick/4, data->thick/2, vert[i], data-
>color);
            }
        }

        if (!err && data->fill){
            fill_area(img, vert, HEX_VERT, data->thick, data->fill_color);
        }

        for (int i=0; i < HEX_VERT; i++){
            if (vert[i])
                free(vert[i]);
        }
        free(vert);
    }

    return err;
}
```

Файл drawing.h

```
#ifndef DRAW
#define DRAW

#include <math.h>

#include "picture.h"
#include "arguments.h"

#define PENT_VERT 5
```

```
#define HEX_VERT 6
#define RECT_VERT 4

void repaint_pixel(PNG* img, int x, int y, colorRGB* color);
void fill_area(PNG* img, point** vert, int cnt_vert, int thick, colorRGB* color);
int check_line(int x, int y, point* p1, point* p2, int thick);
void draw_line(PNG* img, point* p1, point* p2, int thick, colorRGB* color);
void draw_circle(PNG* img, int radius, int thick, point* center, colorRGB* color);
void find_vertex(point*** vert, point* center, int radius, int cnt_vert);

int pentagram_draw(PNG* img, argStruct* data);
int rect_draw(PNG* img, argStruct* data);
int hexagon_draw(PNG* img, argStruct* data);

#endif
```

Файл picture.c

```
#include "picture.h"

int read_png(PNG* img, char* path){
    enum error_types err=ERR_NO;

    FILE* fp=fopen(path, "rb");
    if (!fp){
        fprintf(stderr, "Incorrect input file\n");
        err = ERR_FILE;
    }
    else{
        char header[PNG_HEADER];
        fread(header, 1, PNG_HEADER, fp);
        int is_png = !png_sig_cmp(header, 0, PNG_HEADER);
        if (!is_png) {
            fprintf(stderr, "Not PNG\n");
            err = ERR_IMAGE;
        }

        if (!err){
                img->png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL,
NULL, NULL);
            if (!img->png_ptr){
```

```
                fprintf(stderr, "PNG_read_struct error\n");
                err = ERR_IMG_STRUCT;
            }

            if (!err){
                img->info_ptr = png_create_info_struct(img->png_ptr);
                if (!img->info_ptr){
                    png_destroy_read_struct(&img->png_ptr, NULL, NULL);
                    fprintf(stderr, "PNG_info_struct error\n");
                    err = ERR_IMG_STRUCT;
                }

                if (!err){
                    if (setjmp(png_jmpbuf(img->png_ptr))){
                            png_destroy_read_struct(&img->png_ptr, &img->info_ptr,
NULL);
                        fprintf(stderr, "Input image error\n");
                        err = ERR_IMAGE;
                    }

                    if (!err){
                        png_init_io(img->png_ptr, fp);
                        png_set_sig_bytes(img->png_ptr, PNG_HEADER);

                        png_read_info(img->png_ptr, img->info_ptr);

                              img->width = png_get_image_width(img->png_ptr, img-
>info_ptr);
                             img->height = png_get_image_height(img->png_ptr, img-
>info_ptr);
                             img->bit_depth = png_get_bit_depth(img->png_ptr, img-
>info_ptr);
                           img->color_type = png_get_color_type(img->png_ptr, img-
>info_ptr);
                            img->filter_method = png_get_filter_type(img->png_ptr,
img->info_ptr);
                             img->compression_type = png_get_compression_type(img-
>png_ptr, img->info_ptr);
                         img->interlace_type = png_get_interlace_type(img->png_ptr,
img->info_ptr);
                           img->number_of_passes = png_set_interlace_handling(img-
>png_ptr);
```

```
                        png_read_update_info(img->png_ptr, img->info_ptr);

                            img->row_pointers = (png_bytep*)calloc(img->height,
sizeof(png_bytep));
                        /*освобождение происходит в main(...), после выполнения
действий с изображением;
                            проверка корректности выделения происходит всегда,
если успешно выделилась память под изображение (т.е. могло быть вызвано данное
выделение),
                                        если произошёл сбой выделения памяти для
изображения, то даннай функция не могла быть вызвана*/
                        if (!img->row_pointers){
                            fprintf(stderr, "Memory error\n");
                            err = ERR_MEMORY;
                        }
                        for (int y=0; !err && y < img->height; y++){
                                    img->row_pointers[y] = (png_byte*)calloc(1,
png_get_rowbytes(img->png_ptr, img->info_ptr));
                            /*освобождение происходит в main(...), после выполнения
действий с изображением, перед очисткой img->row_pointers;
                                        организовано аналогично с очисткой img-
>row_pointers, проверяется существование конкретного img->row_pointers[i] и его
очистка*/
                            if (!img->row_pointers[y]){
                                fprintf(stderr, "Memory row error\n");
                                err = ERR_MEMORY;
                            }
                        }
                        if (!err)
                            png_read_image(img->png_ptr, img->row_pointers);
                    }
                }
            }
        }
        fclose(fp);
    }
    return err;
}

int write_png(PNG* img, char* path){
    enum error_types err=ERR_NO;
```

```
    FILE* fp = fopen(path, "wb");
    if (!fp) {
        fprintf(stderr, "Incorrect output file\n");
        err = ERR_FILE;
    }
    else{
        png_structp png_ptr = png_create_write_struct(PNG_LIBPNG_VER_STRING, NULL,
NULL, NULL);
        if (!png_ptr) {
            fprintf(stderr, "PNG_write_struct error\n");
            err = ERR_IMG_STRUCT;
        }

        if (!err){
            png_infop info_ptr = png_create_info_struct(png_ptr);
            if (!info_ptr) {
                fprintf(stderr, "PNG_info_struct error\n");
                err = ERR_IMG_STRUCT;
            }

            if (!err){
                if (setjmp(png_jmpbuf(png_ptr))) {
                    fprintf(stderr, "Output image error\n");
                    err = ERR_IMAGE;
                }

                if (!err){
                    png_init_io(png_ptr, fp);

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

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

                    png_destroy_write_struct(&png_ptr, &info_ptr);
                }
            }
```

```
        }
        fclose(fp);
    }
    return err;
}

void info_print(PNG* img){
    printf("--Image Information--\n");
    printf("    Dimensions: %d x %d pixels\n", img->width, img->height);
    printf("    Bit depth: %d\n", img->bit_depth);
    printf("    Compression method: %d\n", img->compression_type);
    printf("    Filter method: %d\n", img->filter_method);
    printf("    Number of passes: %d\n", img->number_of_passes);
}
```

Файл picture.h

```
#ifndef PICTURE
#define PICTURE

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <png.h>

#include "errors.h"

#define PNG_HEADER 8
#define RGB_LEN 3

typedef struct {
    png_infop info_ptr;
    png_structp png_ptr;

    int width, height;
    png_byte bit_depth;
    png_byte color_type;
    png_byte filter_method;
    png_byte interlace_type;
    png_byte compression_type;

    int number_of_passes;
```

```
    png_bytep* row_pointers;
} PNG;

int read_png(PNG* img, char* path);
int write_png(PNG* img, char* path);
void info_print(PNG* img);

#endif
```

Файл arguments.c

```
#include "arguments.h"

int read_arguments(int* arg_c, char*** arg_v, argStruct* data){
    enum error_types err=ERR_NO;

    int argc=*arg_c;
    char** argv=*arg_v;

    data->help = 0; data->info = 0; data->in = 0; data->out = 0;

    data->mirror = 0; data->pentagram = 0; data->hexagon = 0; data->rect = 0;
    data->compress = 0;

    data->axis = -1; data->radius = -1; data->thick = -1;
    data->col = 0; data->fill = 0; data->fc = 0;
    data->lu = 0; data->rd = 0; data->cent = 0;
    data->num = 0;

    data->left_up = (point*)calloc(1, sizeof(point));
    data->right_down = (point*)calloc(1, sizeof(point));
    data->center = (point*)calloc(1, sizeof(point));
    data->color = (colorRGB*)calloc(1, sizeof(colorRGB));
    data->fill_color = (colorRGB*)calloc(1, sizeof(colorRGB));
    /*освобождение происходит в main(...), после выполнения основных действий;
         если корректно выделилась память под структуру data, проверяется отдельно
корректности выделения памяти под каждое из данных полей и его очистка,
            в случае ошибки выделения data, данная функция не будет вызвана*/
     if (!data->right_down || !data->left_up || !data->center || !data->color || !
data->fill_color){
        fprintf(stderr, "Memory error\n");
        err = ERR_MEMORY;
    }
```

```
    int opt, arg_err=0, last=0;
                  while    (!err    &&    (opt    =    getopt_long(argc,    argv,
"hIi:o:MPRHAn:a:u:d:c:r:t:C:fF:?", options, NULL)) != -1){
        switch (opt){
            case 'h':
                data->help = 1;
                           if (optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                    fprintf(stderr, "This option does not take arguments\n");
                    err = ERR_ARGUMENT;
                }
                else if (optind == argc)
                    last = 1;
                break;

            case 'I':
                data->info = 1;
                           if (optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                    fprintf(stderr, "This option does not take arguments\n");
                    err = ERR_ARGUMENT;
                }
                break;

            case 'i':
                data->in = 1;
                data->input = (char*)calloc(strlen(optarg), sizeof(char));
                  /*освобождение происходит в main(...), после выполнения основных
действий;
                           если корректно выделилась память под структуру data,
проверяется корректность выделения памяти под data->input и его очистка,
                        в случае ошибки выделения data, данная функция не будет
вызвана*/
                if (!data->input){
                    fprintf(stderr, "Memory error\n");
                    err = ERR_MEMORY;
                }
                if (!err)
                    strncpy(data->input, optarg, strlen(optarg));

                           if (optind < argc-1 && (argv[optind][0] != '-' ||
```

```
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                    fprintf(stderr, "Excess arguments\n");
                    err = ERR_ARGUMENT;
                }
                break;

            case 'o':
                data->out = 1;
                data->output = (char*)calloc(strlen(optarg), sizeof(char));
                  /*освобождение происходит в main(...), после выполнения основных
действий;
                           если корректно выделилась память под структуру data,
проверяется корректность выделения памяти под data->output и его очистка,
                         в случае ошибки выделения data, данная функция не будет
вызвана*/
                if (!data->output){
                    fprintf(stderr, "Memory error\n");
                    err = ERR_MEMORY;
                }
                if (!err)
                    strncpy(data->output, optarg, strlen(optarg));

                           if (optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                    fprintf(stderr, "Excess arguments\n");
                    err = ERR_ARGUMENT;
                }
                break;

            case 'M':
                data->mirror = 1;
                           if (optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                    fprintf(stderr, "This option does not take arguments\n");
                    err = ERR_ARGUMENT;
                }
                break;

            case 'P':
                data->pentagram = 1;
                           if (optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
```

```
                fprintf(stderr, "This option does not take arguments\n");
                err = ERR_ARGUMENT;
            }
            break;

        case 'R':
            data->rect = 1;
                        if (optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                fprintf(stderr, "This option does not take arguments\n");
                err = ERR_ARGUMENT;
            }
            break;

        case 'H':
            data->hexagon = 1;
                        if (optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                fprintf(stderr, "This option does not take arguments\n");
                err = ERR_ARGUMENT;
            }
            break;

        case 'A':
            data->compress = 1;
                        if (optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                fprintf(stderr, "This option does not take arguments\n");
                err = ERR_ARGUMENT;
            }
            break;

        case 'n':
            arg_err = sscanf(optarg, "%d", &(data->num));
            if (arg_err == -1 || data->num <= 1){
                fprintf(stderr, "Incorrect num\n");
                err = ERR_ARGUMENT;
            }
            break;

        case 'a':
            data->axis = optarg[0];
```

```
                    if (strlen(optarg) != 1 || (data->axis != 'x' && data->axis !=
'y')){
                        fprintf(stderr, "Incorrect axis\n");
                        err = ERR_ARGUMENT;
                    }

                          if (!err && optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                        fprintf(stderr, "Excess arguments\n");
                        err = ERR_ARGUMENT;
                    }
                    break;


                case 'u':
                       arg_err = sscanf(optarg, "%d.%d", &(data->left_up->x), &(data-
>left_up->y));
                    if (arg_err == -1){
                        fprintf(stderr, "Incorrect left_up\n");
                        err = ERR_ARGUMENT;
                    }
                    if (!err)
                        data->lu = 1;

                          if (!err && optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                        fprintf(stderr, "Excess arguments\n");
                        err = ERR_ARGUMENT;
                    }
                    break;

                case 'd':
                     arg_err = sscanf(optarg, "%d.%d", &(data->right_down->x), &(data-
>right_down->y));
                    if (arg_err == -1){
                        fprintf(stderr, "Incorrect right_down\n");
                        err = ERR_ARGUMENT;
                    }
                    if (!err)
                        data->rd = 1;

                          if (!err && optind < argc-1 && (argv[optind][0] != '-' ||
```

```
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                    fprintf(stderr, "Excess arguments\n");
                    err = ERR_ARGUMENT;
                }
                break;

            case 'c':
                    arg_err = sscanf(optarg, "%d.%d", &(data->center->x), &(data-
>center->y));
                if (arg_err == -1){
                    fprintf(stderr, "Incorrect center\n");
                    err = ERR_ARGUMENT;
                }
                if (!err)
                    data->cent = 1;

                      if (!err && optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                    fprintf(stderr, "Excess arguments\n");
                    err = ERR_ARGUMENT;
                }
                break;

            case 'r':
                data->radius = atoi(optarg);
                if (data->radius <= 0){
                    fprintf(stderr, "Incorrect radius\n");
                    err = ERR_ARGUMENT;
                }

                      if (!err && optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                    fprintf(stderr, "Excess arguments\n");
                    err = ERR_ARGUMENT;
                }
                break;

            case 't':
                data->thick = atoi(optarg);
                if (data->thick <= 0){
                    fprintf(stderr, "Incorrect thickness\n");
                    err = ERR_ARGUMENT;
```

```
                }

                     if (!err && optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                    fprintf(stderr, "Excess arguments\n");
                    err = ERR_ARGUMENT;
                }
                break;

            case 'C':
                   arg_err = sscanf(optarg, "%d.%d.%d", &(data->color->r), &(data-
>color->g), &(data->color->b));
                if (arg_err == -1 || data->color->r < 0 || data->color->r > 255 ||
data->color->g < 0 || data->color->g > 255 || data->color->b < 0 || data->color->b
> 255){
                    fprintf(stderr, "Incorrect color\n");
                    err = ERR_ARGUMENT;
                }
                if (!err)
                    data->col = 1;

                     if (!err && optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                    fprintf(stderr, "Excess arguments\n");
                    err = ERR_ARGUMENT;
                }
                break;

            case 'f':
                data->fill = 1;
                            if (optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                    fprintf(stderr, "This option does not take arguments\n");
                    err = ERR_ARGUMENT;
                }
                break;

            case 'F':
                     arg_err = sscanf(optarg, "%d.%d.%d", &(data->fill_color->r),
&(data->fill_color->g), &(data->fill_color->b));
                 if (arg_err == -1 || data->fill_color->r < 0 || data->fill_color-
>r > 255 || data->fill_color->b < 0 || data->fill_color->b > 255 || data-
```

```
>fill_color->g < 0 || data->fill_color->g > 255){
                    fprintf(stderr, "Incorrect fill_color\n");
                    err = ERR_ARGUMENT;
                }
                if (!err)
                    data->fc = 1;

                    if (!err && optind < argc-1 && (argv[optind][0] != '-' ||
(strlen(argv[optind]) == 1 && argv[optind][0] == '-'))){
                    fprintf(stderr, "Excess arguments\n");
                    err = ERR_ARGUMENT;
                }
                break;

            case '?':
                fprintf(stderr, "Incorrect argument\n");
                err = ERR_ARGUMENT;
        }
    }

    if (!err){
        if (!data->in){
               if (argc < 2 && (data->mirror || data->rect || data->pentagram ||
data->hexagon)){
                fprintf(stderr, "Absent input file\n");
                err = ERR_FILE;
            }
            else if (argc >= 2 && !last){
                data->in = 1;
                data->input = (char*)calloc(strlen(argv[argc-1]), sizeof(char));
                   /*второй случай выеления памяти под data->input, в один запуск
программы выполниться оба выделение не могут, т.к. вызываются при наличии либо
отсутствии ввода конкретного флага;
                    освобождение происходит в main(...), после выполнения основных
действий;
                            если корректно выделилась память под структуру data,
проверяется корректность выделения памяти под data->input и его очистка,
                                в случае ошибки выделения data, данная функция не
будет вызвана*/
                if (!data->input){
                    fprintf(stderr, "Memory error\n");
                    err = ERR_MEMORY;
```

```
                }
                if (!err)
                    strncpy(data->input, argv[argc-1], strlen(argv[argc-1]));
            }
        }

        if (!data->out){
            data->output = (char*)calloc(8, sizeof(char));
                /*второй случай выеления памяти под data->output, в один запуск программы выполниться оба выделение не могут, т.к. вызываются при наличии либо отсутствии ввода конкретного флага;
                    освобождение происходит в main(...), после выполнения основных действий;
                        если корректно выделилась память под структуру data, проверяется корректность выделения памяти под data->output и его очистка,
                        в случае ошибки выделения data, данная функция не будет вызвана*/
            if (!data->output){
                fprintf(stderr, "Memory error\n");
                err = ERR_MEMORY;
            }
            if (!err)
                strncpy(data->output, "out.png\0", 8);
        }

        if (!err && data->in && data->out && !strcmp(data->input, data->output)){
            fprintf(stderr, "Input and output files are the same\n");
            err = ERR_FILE;
        }

          if (!err && data->mirror+data->pentagram+data->rect+data->hexagon+data->help+data->info+data->compress > 1){
            fprintf(stderr, "Too many options\n");
            err = ERR_ARGUMENT;
        }

         if (!err && ((data->mirror && (data->axis == -1 || !data->lu || !data->rd))
         || (data->pentagram && (!data->cent || data->radius == -1 || data->thick == -1 || !data->col))
         || (data->rect && (!data->lu || !data->rd || data->thick == -1 || !data->col || (data->fill && !data->fc) || (!data->fill && data->fc)))
```

```
        || (data->hexagon && (!data->cent || data->radius == -1 || data->thick ==
-1 || !data->col || (data->fill && !data->fc) || (!data->fill && data->fc)))
        || data->compress && !data->num)){
            fprintf(stderr, "Missing required flags\n");
            err = ERR_ARGUMENT;
        }

        if (!err && ((data->mirror && (data->num || data->cent || data->radius !=
-1 || data->thick != -1 || data->col || data->fill || data->fc))
        || (data->pentagram && (data->num || data->axis != -1 || data->lu || data-
>rd || data->fill || data->fc))
         || (data->rect && (data->num || data->axis != -1 || data->cent || data-
>radius != -1))
         || (data->hexagon && (data->num || data->axis != -1 || data->lu || data-
>rd))
        || (data->num && (data->cent || data->radius != -1 || data->thick != -1 ||
data->col || data->fill || data->fc || data->axis != -1 || data->lu || data-
>rd)))){
            fprintf(stderr, "Unnecessary parameters\n");
            err = ERR_ARGUMENT;
        }

        if (!err && data->rd && data->lu){
            if (data->left_up->x > data->right_down->x){
                int tmp=data->left_up->x;
                data->left_up->x = data->right_down->x;
                data->right_down->x = tmp;
            }
            if (data->left_up->y > data->right_down->y){
                int tmp=data->left_up->y;
                data->left_up->y = data->right_down->y;
                data->right_down->y = tmp;
            }
        }
    }

    return err;
}
```

Файл arguments.h

```
#ifndef ARGUMENTS
#define ARGUMENTS
```

```
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include <string.h>

#include "errors.h"

enum flagValue{
    ABSENT = 0,
    PRESENT = 1
};

typedef struct{
    int x, y;
} point;

typedef struct{
    int r, g, b;
} colorRGB;

typedef struct{
    enum flagValue help, info, in, out, lu, rd, cent, col, fc;
    int mirror, pentagram, rect, hexagon, compress;

    char axis;
    char *input, *output;
    int radius, thick, fill, num;
    colorRGB *color, *fill_color;
    point *center, *left_up, *right_down;
} argStruct;

static struct option options[] = {
    {"help", no_argument, 0, 'h'},
    {"info", no_argument, 0, 'I'},
    {"input", required_argument, 0, 'i'},
    {"output", required_argument, 0, 'o'},
    {"mirror", no_argument, 0, 'M'},
    {"pentagram", no_argument, 0, 'P'},
    {"rect", no_argument, 0, 'R'},
    {"hexagon", no_argument, 0, 'H'},
    {"axis", required_argument, 0, 'a'},
```

```
    {"left_up", required_argument, 0, 'u'},
    {"right_down", required_argument, 0, 'd'},
    {"center", required_argument, 0, 'c'},
    {"radius", required_argument, 0, 'r'},
    {"thickness", required_argument, 0, 't'},
    {"color", required_argument, 0, 'C'},
    {"fill", no_argument, 0, 'f'},
    {"fill_color", required_argument, 0, 'F'},
    {"compress", no_argument, 0, 'A'},
    {"num", required_argument, 0, 'n'},
    {0, 0, 0, 0}
};

int read_arguments(int* arg_c, char*** arg_v, argStruct* data);

#endif
```

Файл errors.h

```
#ifndef ERRORS
#define ERRORS

enum error_types{
    ERR_NO = 0,

    ERR_MEMORY = 40,
    ERR_FILE,
    ERR_ARGUMENT,
    ERR_IMAGE,
    ERR_IMG_STRUCT,
    ERR_VALUE,
    FALSE_VALUE = 46
};

#endif
```

Файл help.c

```
#include "help.h"

void help_print(){
    printf("--Functions information--\n");
    printf("\n    --help, -h: function information. Optional flag\n");
```

```
    printf("    --info: image information. Optional flag\n");
    printf("    --input, -i: input file name. Optional flag, defaults to the last argument entered as the input file name\n");
    printf("    --output, -o: output file name. Optional flag, defaults the result is saved to the out.png file\n");

    printf("\n    --mirror: reflection of a given area. Flags:\n");
    printf("        --left_up: coordinates of the upper left corner. The value is specified 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 specified in the format 'right.down', where 'right' is the x-coordinate, 'down' is the y-coordinate\n");

    printf("\n    --pentagram: drawing a pentagram in a circle. Flags:\n");
    printf("         --center: center coordinates. The value is specified in the format 'x.y', where 'x' is the x-coordinate, 'y' is the y-coordinate\n");
    printf("            --radius: radius. Accepts a number greater than 0 as input\n");
    printf("          --thickness: line and circle thickness. Accepts a number greater than 0 as input\n");
    printf("         --color: line and circle color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component\n");

    printf("\n    --rect: drawing a rectangle. Flags:\n");
    printf("        --left_up: coordinates of the upper left corner. The value is specified 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 specified in the format 'right.down', where 'right' is the x-coordinate, 'down' is the y-coordinate\n");
    printf("        --thickness: line thickness. Accepts a number greater than 0 as input\n");
    printf("        --color: line color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component\n");
    printf("         --fill: rectangle fill. Works as a binary value: no flag - false, flag present - true\n");
    printf("        --fill_color: rectangle fill color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component\n");

    printf("\n    --hexagon: drawing a regular hexagon. Flags:\n");
    printf("         --center: center coordinates. The value is specified in the
```

```
format 'x.y', where 'x' is the x-coordinate, 'y' is the y-coordinate\n");
      printf("            --radius: radius. Accepts a number greater than 0 as input\n");
     printf("         --thickness: line thickness. Accepts a number greater than 0 as input\n");
    printf("        --color: line color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component\n");
     printf("          --fill: hexagon fill. Works as a binary value: no flag - false, flag present - true\n");
     printf("         --fill_color: hexagon fill color. Is specified by the string `rrr.ggg.bbb`, where rrr/ggg/bbb are numbers specifying the color component\n");
}
```

Файл help.h

```
#ifndef HELP
#define HELP

#include <stdio.h>

void help_print();

#endif
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

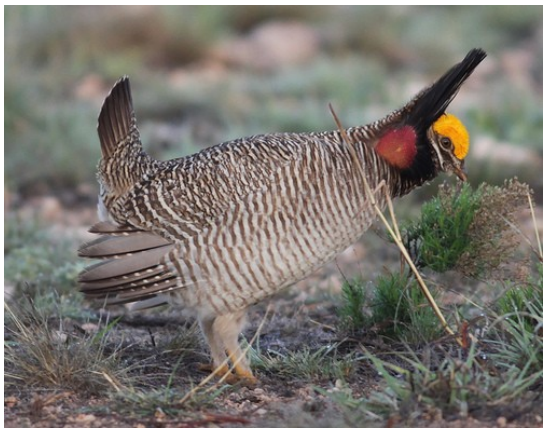

Рисунок 2.1.1. ./input/bird.png

Рисунок 2.1.2. ./output.png

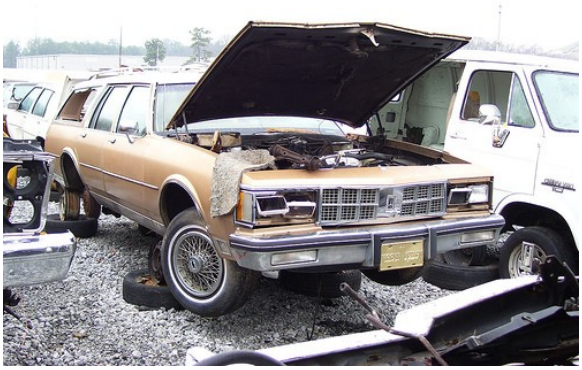

Рисунок 2.2.1. ./input/car.nST

Рисунок 2.2.1. out.png

Рисунок 3.1.1. ./input/milk.XyA

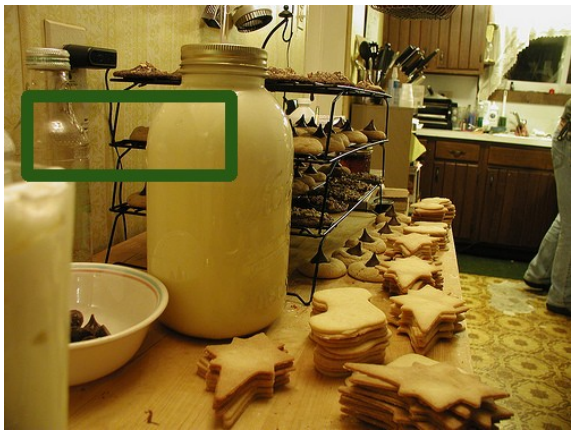

Рисунок 3.1.2. rect_out.Gfe

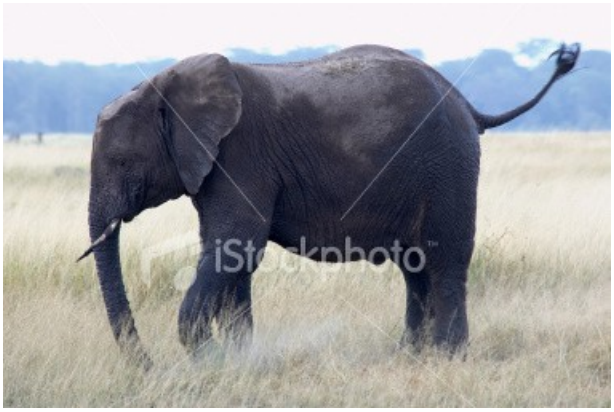

Рисунок 3.2.1. ./input/eleph.png

Рисунок 3.2.2. out.png

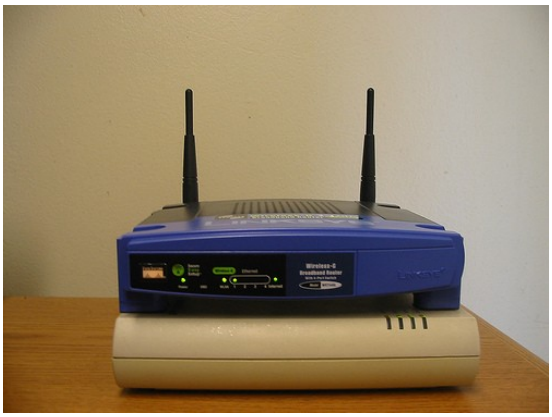

Рисунок 4.1.1. /input/radio.vQh

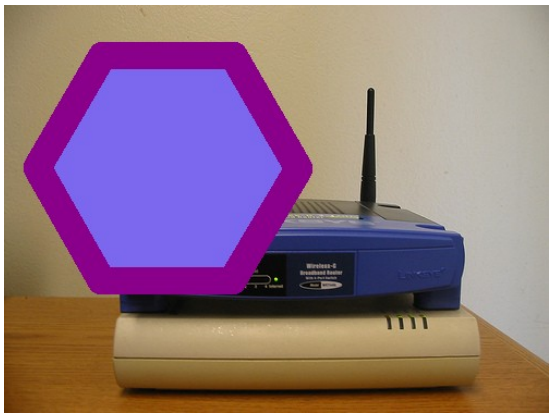

Рисунок 4.1.2. out.png

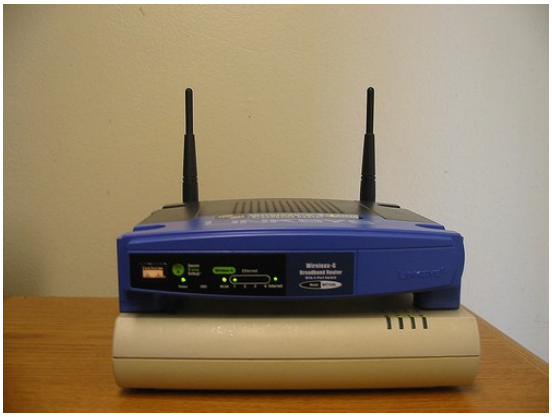

Рисунок 4.2.1. ./input/radio.vQh

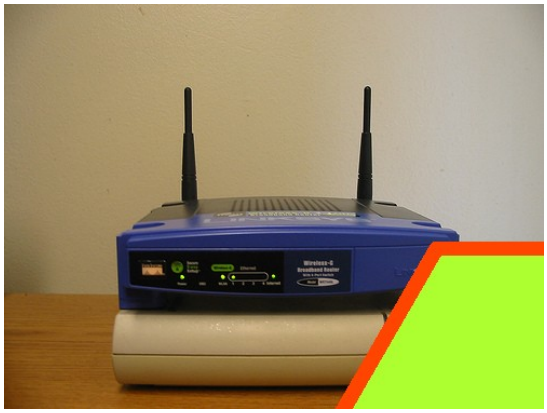

Рисунок 4.2.2. out.png

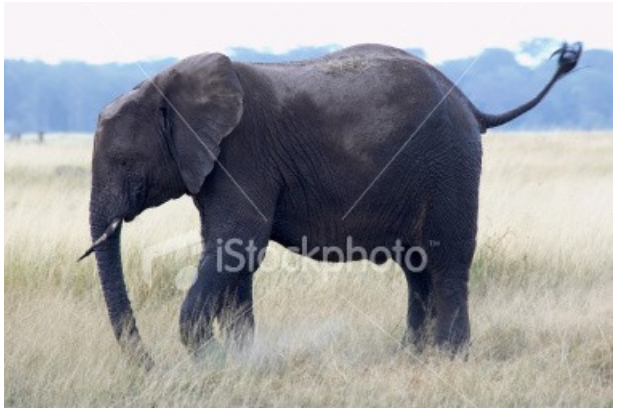

Рисунок 5.1.1. ./input/eleph.png

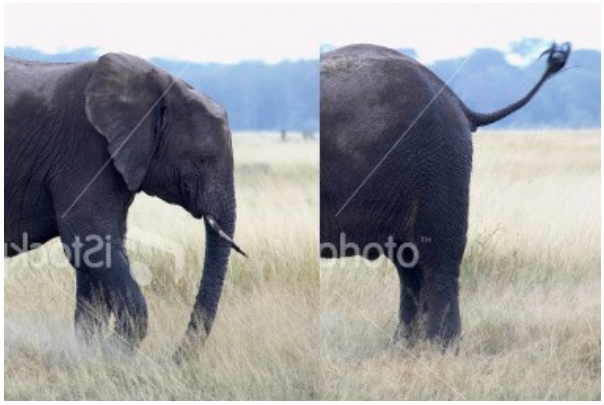

Рисунок 5.1.2. out.png

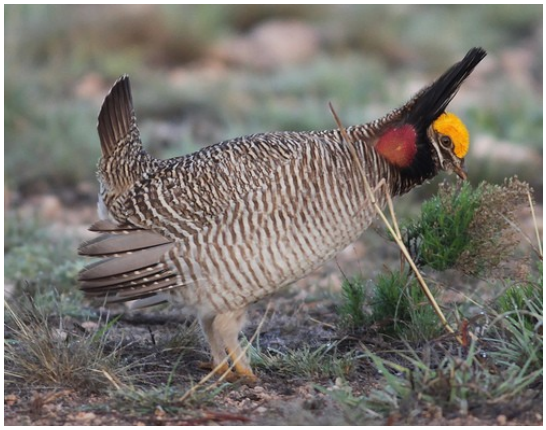

Рисунок 5.2.1. ./input/bird.png

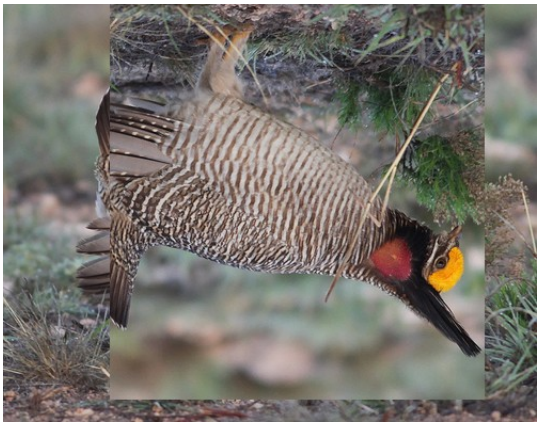

Рисунок 5.2.2. out.png