

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

КУРСОВАЯ РАБОТА
по дисциплине «Программирование»
Тема: Обработка бинарных данных

Студент гр. 1304

Шаврин А.П.

Преподаватель

Чайка К.В.

Санкт-Петербург

2022

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Шаврин А.П

Группа 1304

Тема работы: Обработка бинарных данных

Исходные данные:

Реализовать программу на языке программирования Си, с интерфейсом командной строки, предоставляющую функционал по обработке изображения в формате bmp.

Содержание пояснительной записки:

«Содержание», «Введение», «Ход работы», «Документация по использованию программы», «Тестирование и примеры работы программы», «Заключение», «Список использованных источников», «Приложение А. Исходный код программы»

Предполагаемый объем пояснительной записки:

Не менее 111 страниц.

Дата выдачи задания: 22.03.2022

Дата сдачи реферата: 02.06.2022

Дата защиты реферата: 04.06.2022

Студент

Шаврин А.П.

Преподаватель

Чайка К.В.

АННОТАЦИЯ

В ходе выполнения работы была разработана программа на языке Си, которая выполняет обработку изображений в формате bmp. Для взаимодействия пользователя с программой был реализован интерфейс командной строки. При финальном тестировании программы ошибок выявлено не было.

SUMMARY

In the course of the work, a C program was developed that performs image processing in bmp format. A command line interface was implemented for user interaction with the program. No errors were detected during the final testing of the program.

СОДЕРЖАНИЕ

Введение	5
1. Ход работы	6-15
1.1. Техническое задание	6-7
1.2. Разбиение на подзадачи	7
1.3. Используемые структуры	8
1.4. Считывание команд пользователя	8-12
1.5. Считывание входного файла и создание выходного файла	12
1.6. Реализация основных задач	12-14
1.7. Объединение всех задач	15
2. Документация по использованию программы	16-20
2.1. Общая информация	16
2.2. Считывание входного файла	16-17
2.3. Считывание выходного файла	17
2.4. Рисование шестиугольника	17-18
2.5. Копирование области	18-19
2.6. Замена цвета	19
2.7. Рисование рамки	19-20
3. Тестирование и примеры работы с программой	21-34
3.1. Тестирование открытия файла	21-22
3.2. Тестирование сохранения файла	22-23
3.3. Тестирование передачи флагов	23-24
3.4. Тестирование передачи аргументов	25-27
3.5. Тестовые запуски всех задач	27-35
Заключение	36
Список использованных источников	37
Приложение А. Исходный код программы	38-
	111

ВВЕДЕНИЕ

Цели:

Основной целью работы является улучшение навыков программирования на языке Си и применение на практике материала, изученного в течение семестра.

Программа должна вызываться из командной строки, анализировать поданные ей на вход ключи и обрабатывать изображение методом, выбранным пользователем.

Задачи:

Для достижения поставленных целей требуется решить следующие задачи:

- Изучить работу с getopt.
- Выполнение задания курсовой работы по написанию программы, выполняющей обработку изображений в формате bmp.
- Тестирование и отладка разработанного функционала.

1. ХОД РАБОТЫ

1.1. Техническое задание

1.1.1 Вариант 7

Программа должна иметь CLI или GUI. Более подробно тут:
http://se.moevm.info/doku.php/courses:programming:rules_extra_kurs

Общие сведения

- 24 бита на цвет
- без сжатия
- файл всегда соответствует формату BMP (но стоит помнить, что версий у формата несколько)
- 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 реализовывать весь следующий функционал по обработке bmp-файла

1. Рисование правильного шестиугольника. Шестиугольник определяется:
 - либо координатами левого верхнего и правого нижнего угла квадрата, в который он вписан, либо координатами его центра и радиусом в который он вписан
 - толщиной линий
 - цветом линий
 - шестиугольник может быть залит или нет
 - цветом которым залит шестиугольник, если пользователем выбран залитый
2. Копирование заданной области. Функционал определяется:
 - Координатами левого верхнего угла области-источника
 - Координатами правого нижнего угла области-источника
 - Координатами левого верхнего угла области-назначения

3. Заменяет все пиксели одного заданного цвета на другой цвет.

Функционал определяется:

- Цвет, который требуется заменить
- Цвет на который требуется заменить

4. Сделать рамку в виде узора. Рамка определяется:

- Узором (должно быть несколько на выбор. Красивый узор можно получить используя фракталы)
- Цветом
- Шириной

1.2. Разбиение на подзадачи

Таким образом, код можно разбить на следующие подзадачи:

- Реализовать структуры для удобной работы с командами пользователя.
- Реализовать считывание команд и обработку возможных ошибок.
- Реализовать функцию печати подсказки для пользователя по работе с программой.
- Реализовать считывание входного bmp файла.
- Реализовать создание выходного bmp файла.
- Реализовать рисование правильного шестиугольника.
- Реализовать копирование заданной области.
- Реализовать замену пикселей заданного цвета.
- Реализовать рисование рамки с использованием фракталов.
- Реализовать функции очистки динамически выделяемой памяти в программе.

1.3. Используемые структуры

- ***Point***

Структура для хранения координат точек. Имеет поля *x* и *y* типа *int*.

- ***Rgb***

Структура хранит в себе информацию о *RGB* компонентах каждого пикселя и имеет поля *r*, *g*, *b* типа *unsigned char*.

- ***Configs***

Структура для хранения задач которые пользователь хочет реализовать. Имеет поля *draw_hexagon*, *copy_area*, *switch_color*, *draw_frame*, *bmp_info* типа *int*, а также *input_file* и *output_file* типа *char**.

- ***Hexagon, Area, Switch_color, Frame***

Структуры для работы с каждой командой пользователя и имеющие все необходимые поля для корректной работы программы.

- ***BitmapFileHeader, BitmapInfoHeader***

Структуры для хранения заголовков файлов.

- ***File***

Для хранения в памяти идущей на вход программе картинки была реализована структура *File*.

1.4. Считывание команд пользователя

Для работы с командами пользователя была использована встроенная библиотека *getopt.h*. Все ключи, которые принимает программа на вход описаны в файле *main.c*:

- Короткие: *"h?i:o:xf:cse"*
- Длинные

В этом же файле реализован алгоритм считывания всех ключей при помощи *switch*.

Обработка каждой задачи вынесена в отдельный файл, в каждом из которых реализованы вспомогательные функции по считыванию и обработке ошибок

при вводе необходимых аргументов каждого дополнительного ключа для описания задачи:

▪ **getopt_files**

Функции:

- Проверка файла на тип:

*int is_bmp(char *arg)*

- Считывание названия входного файла:

*int case_input_file(Configs *config, char *optarg)*

- Считывание названия выходного файла:

*int case_output_file(Configs *config, char *optarg)*

- Поиск входного файла без ключа:

*int search_input_file(int argc, char *argv[], int *optind, Configs *config)*

- Проверка был ли передан входной файл:

void no_input_file()

▪ **getopt_hexagon**

Функции:

- Считывание левого верхнего угла квадрата, в который вписан:

*int case_hexstart(int argc, char *argv[], int *optind, Configs *config, Hexagon *hexagon)*

- Считывание правого нижнего угла квадрата, в который вписан:

*int case_hexend(int argc, char *argv[], int *optind, Configs *config, Hexagon *hexagon)*

- Считывание координат центра:

*int case_hexcenter(int argc, char *argv[], int *optind, Configs *config, Hexagon *hexagon)*

- Считывание радиуса:

*int case_hexradius(int argc, char *argv[], char *optarg, Configs *config, Hexagon *hexagon)*

- Считывание цвета линии:

*int case_hexline_color(int argc, char *argv[], int *optind, Configs *config, Hexagon *hexagon)*

- Считывание цвета заливки:

*int case_hexfilling(int argc, char *argv[], int *optind, Configs *config, Hexagon *hexagon)*

▪ **getopt_copy**

Функции:

- Считывание левого верхнего угла области копирования:

*int case_start_copy(int argc, char *argv[], int *optind, Configs *config, Area *area)*

- Считывание правого нижнего угла области копирования:

*int case_end_copy(int argc, char *argv[], int *optind, Configs *config, Area *area)*

- Считывание левого верхнего угла области вставки:

*int case_insert(int argc, char *argv[], int *optind, Configs *config, Area *area)*

▪ **getopt_switch_color**

Функции:

- Считывание цвета, который заменяем:

*int case_old_color(int argc, char *argv[], int *optind, Configs *config, SwitchColor *switch_color)*

- Считывание цвета, на который заменяем:

*int case_new_color(int argc, char *argv[], int *optind, Configs *config, SwitchColor *switch_color)*

▪ **getopt_frame**

Функции:

- Считывание типа рамки:

*int case_edge_type(int argc, char *argv[], char *optarg, Configs *config, Frame *frame)*

- Считывание ширины рамки:

*int case_edge_width(int argc, char *argv[], char *optarg, Configs *config, Frame *frame)*

- Считывание цвета узора рамки:

*int case_edge_color(int argc, char *argv[], int *optind, Configs *config, Frame *frame)*

- Считывание цвета заливки рамки:

*int case_bg_color(int argc, char *argv[], int *optind, Configs *config, Frame *frame)*

В файле *getopt* реализована печать подсказки пользователя и общие функции, используемые в каждом из файлов типа *getopt_”задача”*.

▪ **Getopt**

Функции:

- Вывод инструкции:

void PrintHelp()

- Проверка на число:

*int is_number(char *arg)*

- Проверка на rgb компоненты:

int is_rgb_numbers(int args[])

- Считывание нескольких аргументов после ключа:

*int read_args(int argc, char *argv[], int *optind, int count, int args[])*

1.5. Считывание входного файла и создание выходного файла

Для считывания и создания файла был реализован файл *file_processing*, в котором были прописаны функции:

- считывания файла:

*int read_input_file(Configs *config, File *file, BitmapFileHeader *bmfh, BitmapInfoHeader *bmih)*

- создания выходного файла:

*int create_output_file(Configs *config, File *file)*

1.6. Реализация основных задач

Был прописан файл *draw_common_func*, в котором были реализованы функции, связанные с рисованием на картинке и используемые в большинстве файлов по решению задач пользователя.

Функции:

- Закрашивание пикселя:

*void set_pixel(Rgb *pixel, Rgb *new)*

- Проверка на совпадение цвета:

```
int is_check_color(Rgb *current, Rgb *new)
```

- Рисование прямоугольника:

```
void draw_rectangle(int x0, int y0, int x1, int y1, Rgb **arr, Rgb *color)
```

- Рисование линии алгоритмом Брезенхема:

```
void draw_line(int x0, int y0, int x1, int y1, Rgb **arr, Rgb *color)
```

Для реализации каждой задачи были созданы файлы, в каждом из которых реализованы все необходимые функции для решения поставленных задач, а также обработка всех возможных ошибок, которые могут повлиять на корректную работу программы:

- **draw_hexagon**

Функции:

- Заливка шестиугольника:

```
void hexagon_flood_fill(int x, int y, Rgb **arr, Rgb *new_color, Rgb *boarder_color)
```

- Получение вершин:

```
void get_hexpoints(int x, int y, int r, Rgb **arr, Point *points_arr);
```

- Преобразование координат квадрата в который вписан шестиугольник к радиусу и центру:

```
int transformation(Hexagon *hexagon);
```

- Рисование шестиугольника:

```
int DrawHexagon(Hexagon *hexagon, Rgb **arr, File *file);
```

- **copy_area**

Функции:

- Проверка корректности координат:

*int check_correct_copy_coordinates(Area *area, File *file);*

- Создание копии:

*Rgb **create_cpy_area(Rgb **arr, Area *area);*

- Удаление копии:

*void free_copy_area(Rgb **arr, Area *area);*

- Копирование:

*int CopyAreaFunc(Rgb **arr, Area *area, File *file);*

▪ *switch_color*

Функции:

- Смена цвета:

*int SwitchColorFunc(Rgb **arr, SwitchColor *switch_color, File *file);*

▪ *draw_frame.*

Функции:

- Заливка рамки:

*void draw_bg_frame(Frame *frame, Rgb **arr, File *file);*

- Рисование итоговой рамки:

*int DrawFrame(Frame *frame, Rgb **arr, File *file);*

Для файла *draw_frame* было создано дополнительно 3 файла с названиями *draw_frame_serp*, *draw_frame_koch*, *draw_frame_minc*, в которых были реализованы 3 типа рамок с использованием фракталов Серпинского, Коха и Минковского.

1.7. Объединение всех задач

Когда все необходимые задачи и подзадачи были реализованы, в файле *main.c* были созданы все необходимые структуры, прописан код, выполняющий считывание входного файла, выполнение всех введенных пользователем задач и конечную запись измененной картинки в результирующий файл.

2. ДОКУМЕНТАЦИЯ ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ

2.1. Общая информация

Для отображения краткой версии данной справки запустите программу с флагом -h (-?, --help).

Для корректной работы программы обязательно передать название входного файла в любом месте цепочки флагов, либо по флагу -i (--inputfile). При отсутствии указания флага выходного файла -o (--outputfile), обработанное изображение будет сохранено в директорию, в которой находится исполняемый файл, с именем «out.bmp» (сохранение по умолчанию).

После указания флага исполняемой опции (все возможные опции будут перечислены далее) в качестве аргумента передаются все необходимые параметры, ввод параметров производится через символ « » (пробел).

При вызове нескольких опций все они выполняются в следующем порядке:

- Копирование области
- Замена цвета
- Рисование шестиугольника
- Рисование рамки

Если в качестве параметра опция принимает цвет, то его можно указать как последовательное перечисление значений красной, зеленой и синей компонент через пробел. Значения компонент должны быть в диапазоне от 0 до 255.

Началом координат считается левый нижний угол.

2.2. Считывание входного файла

Для открытия файла используется флаг -i (--inputfile), в качестве аргумента передается путь к открываемому файлу. Если данный флаг не указан, то программа будет искать название входного файла среди всех лишних значений, переданных без флага. Если же программа и так не найдет входной

файл, то будет выведено сообщение об отсутствии входного файла, справка по работе с программой и программа завершится.

2.3. Считывание выходного файла

Для сохранения выходного файла используется флаг -o (--outputfile), в качестве аргумента указывается путь к файлу в формате bmp, в который необходимо записать результат работы программы (обработанную картинку). Если при обработке аргумента произойдет ошибка, то будет выведено сообщение об ошибке и программа закончит свою работу. Если данный флаг не будет указан, то файл будет сохранен в директорию, в которой находится исполняемый файл с именем «out.bmp».

2.4. Рисование шестиугольника

Производит рисование шестиугольника с центром в заданной точке с определенным радиусом, а также толщиной и цветом линии обводки.

(По желанию пользователя, он может указать не центр и радиус, а координаты левого верхнего и правого нижнего угла квадрата, в который он вписан. Однако при указании обоих вариантов, программа отдаст предпочтение квадрату.)

Шестиугольник может быть залит.

Флаги для данной задачи:

➤ -x (--hexagon)

Флаг задачи, не принимает аргументов, но обязан быть передан до следующих флагов

➤ --center

Принимает целочисленные координаты центра (x и y соответственно)

➤ --radius

Принимает целочисленное значение радиуса окружности, в которую вписан шестиугольник

➤ --hexstart

Принимает целочисленные координаты (x и y соответственно) левого верхнего угла квадрата, в который вписан шестиугольник

➤ --hexend

Принимает целочисленные координаты (x и y соответственно) правого нижнего угла квадрата, в который вписан шестиугольник

➤ --linewidth

Принимает целочисленное значение ширины линии обводки в пикселях (по умолчанию 1 пиксель и данный флаг является не обязательным)

➤ --linergb

Принимает 3 rgb компоненты через пробел для линии обводки (по умолчанию цвет черный «0 0 0» и данный флаг является не обязательным)

➤ -f (--fill)

Принимает 3 rgb компоненты через пробел для цвета заливки (по умолчанию шестиугольник не закрашен и данный флаг является не обязательным)

2.5. Копирование области

Производит копирование области заданной левым верхним и правым нижним углом области копирования и вставляет в область заданную левым верхним углом.

Флаги для данной задачи:

➤ -c (--copy)

Флаг задачи, не принимает аргументов, но обязан быть передан до следующих флагов

➤ --start

Принимает целочисленные координаты (x и y соответственно) левого верхнего угла области копирования

➤ --end

Принимает целочисленные координаты (x и y соответственно) правого нижнего угла области копирования

➤ --insert

Принимает целочисленные координаты (x и y соответственно) левого верхнего угла области вставки

2.6. Замена цвета

Производит замену всех пикселей с заданной компонентой rgb на новые значения этих компонент

Флаги для этой задачи:

➤ -s (--switchcolor)

Флаг задачи, не принимает аргументов, но обязан быть передан до следующих флагов

➤ --old

Принимает 3 rgb компоненты через пробел для цвета, который будет изменен

➤ --new

Принимает 3 rgb компоненты через пробел для цвета который будет записан вместо старого

2.7. Рисование рамки

Производит рисование рамки заданного типа, заданной толщины, цвета узора и цвета заливки.

Флаги для этой задачи:

➤ -e (--edge)

Флаг задачи, не принимает аргументов, но обязан быть передан до следующих флагов

➤ --type

Принимает значение типа рамки от 0 до 2 включая 0 и 1.

0 – Рамка с использованием ковра Серпинского (минимальная ширина рамки 1 пиксель)

1 – Рамка с использованием кривой Коха (минимальная ширина рамки 10 пикселей)

2 – Рамка с использованием кривой Минковского (минимальная ширина рамки 17 пикселей)

(По умолчанию тип 0 и данный флаг является не обязательным)

➤ --width

Принимает целочисленное значение ширины рамки в пикселях

(По умолчанию 20 пикселей и данный флаг является не обязательным)

➤ --edgergb

Принимает 3 rgb компоненты через пробел для цвета, которым будет отрисован узор

(По умолчанию белый «255 255 255» и данный флаг является не обязательным)

➤ --bgrgb

Принимает 3 rgb компоненты через пробел для цвета, которым будет залит фон рамки

(По умолчанию черный «0 0 0» и данный флаг является не обязательным)

3. ТЕСТИРОВАНИЕ И ПРИМЕРЫ РАБОТЫ С ПРОГРАММОЙ

В пункте 3.5 “Тестовые запуски всех задач” будут приведены не общие примеры вывода ошибок, а особенные для каждой задачи

3.1. Тестирование открытия файла

- 1) Запуск без передачи файла

Вводимая команда: ./main

Ожидаемый результат: Вывод подсказки

Результат программы:

```
GETOPT HINT

GENERAL
General
h -f --help <-- help
--helpversion <-- supported bmp file versions
--helpinfo <-- bmp file information
i -filename.bmp --inputfile -filename.bmp <-- input BMP file (file to be changed)
o -filename.bmp --outputfile -filename.bmp <-- output BMP file (file where changes are saved)

HEXAGON
h --hexagon <-- draw regular hexagon flag
--center <cx> <cy> <-- coordinates (type int) of the center of the hexagon (the origin is in the lower left corner)
--radius <radius> <-- hexagon radius (type int)

Instead of the coordinates of the center and radius, you can set the coordinates of the left upper and right down corner of the square in which it is inscribed
--hexstart <sx> <sy> <-- coordinates (type int) of the upper left corner of the copy area (x, y)(the origin is in the lower left corner)
--hexend <ex> <ey> <-- coordinates (type int) of the lower right corner of the copy area (x, y)(the origin is in the lower left corner)

--linewidth <width> <-- boarder width (default: 1px)
--linest <red> <green> <blue> <-- boarder color (default: {0, 0, 0}) (each color component ranging from 0 to 255)
-f <red> <green> <blue> -f fill <red> <green> <blue> <-- hexagon fill color (each color component ranging from 0 to 255)

COPY AREA
h --copy <-- copy area flag
--start <sx> <sy> <-- coordinates (type int) of the upper left corner of the copy area (x, y)(the origin is in the lower left corner)
--end <ex> <ey> <-- coordinates (type int) of the lower right corner of the copy area (x, y)(the origin is in the lower left corner)
--insert <ix> <iy> <-- coordinates (type int) of the upper left corner of the paste area (x, y)

SWITCH COLOR
h --switchcolor <-- switch color flag
--old <red> <green> <blue> <-- color to be change (each color component ranging from 0 to 255)
--new <red> <green> <blue> <-- color to be applied (each color component ranging from 0 to 255)

FRAME
h --edge <-- frame flag
--type <pattern> <-- frame pattern {0, 1, 2} (default: 0)
--width <width> <-- frame width (default: 28px)
--edgecolor <red> <green> <blue> <-- frame color (default: {255, 255, 255}) (each color component ranging from 0 to 255)
--border <red> <green> <blue> <-- background frame color (default: {0, 0, 0}) (each color component ranging from 0 to 255)
```

- 2) Попытка открытия несуществующего файла.

Вводимая команда: ./main -i adfg.bmp

Ожидаемый результат: Программа выведет ошибку открытия файла.

Результат программы:

```
Unable to open input file
```

- 3) Попытка открытия файла неподдерживаемого типа.

Вводимая команда: `./main -i T-34-85.jpg`

Ожидаемый результат: Программа выведет ошибку открытия файла.

Результат программы:

```
T-34-85.jpg - Input file is not BMP!
```

- 4) Попытка открытия правильного файла.

Вводимая команда:

Ожидаемый результат: Программа не выведет ошибок.

Результат программы:

```
$ ./main -i simpsonsvr.bmp  
$
```

- 5) Передача правильного файла без специального флага с флагом информации о файле.

Вводимая команда: `./main simpsonsvr.bmp --bmpinfo`

Ожидаемый результат: Программа выведет информацию о файле.

Результат программы

```
FileHeader  
signature: 4d42 (19778)  
filesize: 141a62 (1317474)  
reserved1: 0 (0)  
reserved2: 0 (0)  
pixelArrOffset: 36 (54)  
InfoHeader  
headerSize: 28 (40)  
width: 30c (780)  
height: 233 (563)  
planes: 1 (1)  
bitsPerPixel: 18 (24)  
compression: 0 (0)
```

3.2. Тестирование сохранения файла

- 1) Сохранение без передачи выходного файла

Вводимая команда: `./main simpsonsvr.bmp`

Ожидаемый результат: Программа сохранит изображение в текущей директории с именем out.bmp

Результат программы:

```
$ ./main -i simpsonsvr.bmp  
$
```



- 2) Сохранение с передачей неправильного типа выходного файла

Вводимая команда: `./main simpsonsvr.bmp -o out.jpg`

Ожидаемый результат: Вывод ошибки о несоответствии типа

Результат программы:

```
out.jpg - Output file is not BMP!
```

- 3) Сохранение с передачей пути в несуществующую директорию

Вводимая команда: `./main -i simpsonsvr.bmp -o ./pictures/out.bmp`

Ожидаемый результат: Вывод ошибки о несоответствии типа

Результат программы:

```
alex@alex-VirtualBox:~/PR/2/Course_Work/end$ ./main -i simpsonsvr.bmp -o ./pictures/out.bmp  
Unable to create output file
```

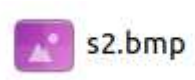
- 4) Правильная передача выходного файла

Вводимая команда: `./main simpsonsvr.bmp -o s2.bmp`

Ожидаемый результат: Программа сохранит изображение в текущей директории с именем s2.bmp

Результат программы:

```
$ ./main -i simpsonsvr.bmp -o s2.bmp  
$
```



3.3. Тестирование передачи флагов

В данном разделе тестирование происходит на задаче отрисовки шестиугольника, такие же результаты будут получены на других задачах.

- 1) Не передача флага задачи, но передача остальных флагов

Вводимая команда: `./main -i simpsonsvr.bmp --center 3 4 --radius 20`

Ожидаемый результат: Программа выведет ошибку

Результат программы:

```
Hexagon flag {-x}/{--hexagon} not specified!
```

- 2) Не передача всех необходимых флагов

Вводимая команда: `./main -i simpsonsvr.bmp --center 3 4`

Ожидаемый результат: Программа выведет ошибку

Результат программы:

```
Not all necessary flags for draw hexagon were passed
```

- 3) Передача несуществующего флага

Вводимая команда: `./main -i simpsonsvr.bmp --center 3 4 --radius 2 --angle`

Ожидаемый результат: Программа выведет ошибку и подсказку

Результат программы:

```
./main: unrecognized option '--angle'

SETUP HINT

GENERAL
./main --help <-- help
./main --bmv <-- supported bmp file versions
./main --bmpi <-- bmp file information
./main -i <filename.bmp> --o <outfile> <filename.bmp> <-- input BMP file (file to be changed)
./main -i <filename.bmp> --o <outfile> <filename.bmp> <-- output BMP file (file were changes are saved)

HEXAGON
./main --hexagon <-- draw regular hexagon flag
--center <cx> <cy> <-- coordinates (type int) of the center of the hexagon (the origin is in the lower left corner)
--radius <radius> <-- hexagon radius (type int)

Instead of the coordinates of the center and radius, you can set the coordinates of the left upper and right down corner of the square in which it is inscribed
--hexstart <cx> <cy> <-- coordinates (type int) of the upper left corner of the copy area (x, y)(the origin is in the lower left corner)
--hexend <cx> <cy> <-- coordinates (type int) of the lower right corner of the copy area (x, y)(the origin is in the lower left corner)
--linewidth <width> <-- boarder width (default: 1px)
--linetype <red> <green> <blue> <-- boarder color (default: (0, 0, 0)) (each color component ranging from 0 to 255)
--f <red> <green> <blue> --fill <red> <green> <blue> <-- hexagon fill color (each color component ranging from 0 to 255)

COPY AREA
./main --copy <-- copy area flag
--start <cx> <cy> <-- coordinates (type int) of the upper left corner of the copy area (x, y)(the origin is in the lower left corner)
--end <cx> <cy> <-- coordinates (type int) of the lower right corner of the copy area (x, y)(the origin is in the lower left corner)
--insert <cx> <cy> <-- coordinates (type int) of the upper left corner of the paste area (x, y)

SWITCH_COLOR
./main --switchcolor <-- switch color flag
--old <red> <green> <blue> <-- color to be change (each color component ranging from 0 to 255)
--new <red> <green> <blue> <-- color to be applied (each color component ranging from 0 to 255)

FRAME
./main --edge <-- frame flag
--type <pattern> <-- frame pattern (0, 1, 2) (default: 0)
--width <width> <-- frame width (default: 20px)
--edgecolor <red> <green> <blue> <-- frame color (default: (255, 255, 255)) (each color component ranging from 0 to 255)
--bgcolor <red> <green> <blue> <-- background frame color (default: (0, 0, 0)) (each color component ranging from 0 to 255)
```

- 4) Верная передача флагов

Вводимая команда: `./main -i simpsonsvr.bmp --center 3 4 --radius 2`

Ожидаемый результат: Программа не выведет ошибки

Результат программы:

```
$ ./main -i simpsonsvr.bmp -x --center 3 4 --radius 2
$
```


3.4. Тестирование передачи аргументов

➤ 1) Аргументы не того типа

Вводимая команда: `./main -i simpsonsvr.bmp --center a 4 -radius 2`

`./main -i simpsonsvr.bmp -x --center 20 11 --radius 10 --linergb 255 0 g`

Ожидаемый результат: Программа выведет ошибку типа аргумента

Результат программы:

```
Flag [--center] takes arguments of type int  
Flag [--linergb] takes arguments of type int
```

➤ 2) Отрицательные аргументы

Вводимая команда: `./main -i simpsonsvr.bmp --center -4 4 -radius 2`

Ожидаемый результат: Программа выведет ошибку передачи не всех аргументов по флагу, поскольку знак “-” будет воспринят как начало нового флага.

Результат программы:

```
Flag [--center] taken 2 argument, but was given less argument
```

➤ 3) Аргументы не соответствующие значениям RGB компонент

Вводимая команда: `./main -i simpsonsvr.bmp --center 20 11 -radius 10`

Ожидаемый результат: Программа выведет ошибку некорректных данных компонент

Результат программы:

```
Argumentes <red> <green> <blue> must be in the range from 0 to 255
```

➤ 4) Не передача аргументов

Вводимая команда: `./main -i simpsonsvr.bmp --center 20 11 -radius 10 -f`

Ожидаемый результат: Программа выведет ошибку не передачи аргументов и подсказку

Результат программы:

```

./main: option requires an argument -- '?'
./main: main

General
  -h, --help          -- help
  --version          -- supported bmp file versions
  --info             -- bmp file information
  --filename bmp     -- filename of the input bmp file (file to be changed)
  --filename bmp     -- filename of the output bmp file (file name changes are saved)

Hexagon
  -x, --hexagon       -- draw regular hexagon flag
  --center x y       -- coordinates (type int) of the center of the hexagon (the origin is in the lower left corner)
  --radius radius     -- hexagon radius (type int)

Setting of the coordinates of the center and radius, you can set the coordinates of the left upper and right down corner of the hexagon to which it is attached
  --corner x1 y1     -- coordinates (type int) of the upper left corner of the copy area (x, y) (the origin is in the lower left corner)
  --corner x2 y2     -- coordinates (type int) of the lower right corner of the copy area (x, y) (the origin is in the lower left corner)
  --border width     -- border width (default: 1px)
  --border color r g b -- border color (default: (0, 0, 0)) (each color component ranging from 0 to 255)
  --fill r g b       -- fill color (default: (0, 0, 0)) (each color component ranging from 0 to 255)

Copy area
  -c, --copy         -- copy area flag
  --x1 x1 y1         -- coordinates (type int) of the upper left corner of the copy area (x, y) (the origin is in the lower left corner)
  --x2 x2 y2         -- coordinates (type int) of the lower right corner of the copy area (x, y) (the origin is in the lower left corner)
  --x3 x3 y3         -- coordinates (type int) of the upper left corner of the paste area (x, y)

Switch color
  -s, --switch-color  -- switch color flag
  --old r g b         -- color to be changed (each color component ranging from 0 to 255)
  --new r g b         -- color to be applied (each color component ranging from 0 to 255)

Frame
  -f, --frame         -- frame flag
  --type pattern     -- frame pattern (0, 1, 2) (default: 0)
  --width width     -- frame width (default: 10px)
  --border r g b     -- border color (default: (255, 255, 255)) (each color component ranging from 0 to 255)
  --bg r g b         -- background frame color (default: (0, 0, 0)) (each color component ranging from 0 to 255)

```

➤ 5) Передача лишних аргументов

Вводимая команда: `./main -i simpsonsvr.bmp --center 50 50 --radius 40 23 s f`

Ожидаемый результат: Программа проигнорирует последующие данные после первого аргумента.

Результат программы:



➤ 6) Передача аргумента из нужного типа и букв

Вводимая команда: `./main -i simpsonsvr.bmp --center 50 50 --radius 10dfgh`

Ожидаемый результат: Программа проигнорирует последующие данные после последнего символа типа `int`.

Результат программы:



(При передачи слишком большого значения перед беквами, программа сработает так же, как если бы букв не было)

➤ 7) Верная передача аргументов

Вводимая команда: `./main -i simpsonsvr.bmp --center 20 11 --radius 10 -f 255 255 255`

Ожидаемый результат: Программа не выведет ошибок

Результат программы:

```
$ ./main -i simpsonsvr.bmp -x --center 100 150 --radius 50 --linergb 255 0 3 --fill 255 255 255
$
```

3.5. Тестовые запуски всех задач

➤ 1) Рисование шестиугольника

- **Особенные ошибки:**

1. Маленький радиус

```
alex@alex-VirtualBox:~/PR/2/Course_Work/end$ ./main simpsonsvr.bmp -x --center 3 3 --radius 1
Radius too small.
alex@alex-VirtualBox:~/PR/2/Course_Work/end$ ./main simpsonsvr.bmp -x --hexstart 0 5 --hexend 5 0
Radius too small.
```

2. Выход за пределы картинки

```
alex@alex-VirtualBox:~/PR/2/Course_Work/end$ ./main simpsonsvr.bmp -x --center 300 500 --radius 500
Hexagon is not fit in width.
alex@alex-VirtualBox:~/PR/2/Course_Work/end$ ./main simpsonsvr.bmp -x --center 300 600 --radius 5000
Center is not in a picture.
alex@alex-VirtualBox:~/PR/2/Course_Work/end$ ./main simpsonsvr.bmp -x --center 400 500 --radius 200
Hexagon is not fit in height.
```

3. Переданы не координаты квадрата

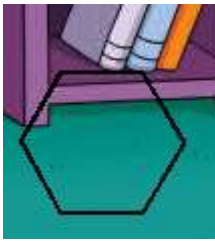
```
alex@alex-VirtualBox:~/PR/2/Course_Work/end$ ./main -i honeycomb.bmp --hexagon --hexstart 100 500 --hexend 300 200
These are not square coordinates
```

- **Center + radius**

Вводимая команда: `./main simpsonsvr.bmp -x --center 50 50 --radius 40`

Ожидаемый результат: Рисование шестиугольника с толщиной линии и цветом по умолчанию

Результат программы:



- **Hexstart + Hexend**

Вводимая команда: `./main simpsonsvr.bmp -x --hexstart 0 50 --hexend 50 0`

Ожидаемый результат: Рисование шестиугольника с толщиной линии и цветом по умолчанию

Результат программы:



- **Center + radius + fill**

Вводимая команда: `./main simpsonsvr.bmp -x --center 50 50 --radius 40 --fill 255 255 255`

Ожидаемый результат: Рисование шестиугольника с толщиной линии и цветом по умолчанию и заливка желтым

Результат программы:

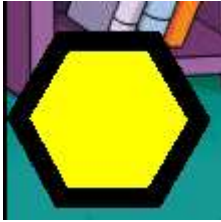


- **Center + radius + fill + linewidth**

Вводимая команда: `./main simpsonsvr.bmp -x --center 50 50 --radius 40 --fill 255 255 0 --linewidth 10`

Ожидаемый результат: Рисование шестиугольника с толстыми границами цвета по умолчанию и заливка желтым

Результат программы:



- **Center + radius + fill + linewidth + linergb**

Вводимая команда: `./main simpsonsvr.bmp -x --center 50 50 --radius 40 --fill 255 255 0 --linewidth 10 --linergb 0 0 255`

Ожидаемый результат: Рисование шестиугольника с толстыми гранями синего цвета и желтой заливкой

Результат программы:



➤ 2) Рисование рамки

- **Особенные ошибки:**

1. Несуществующий тип рамки

```
alex@alex-VirtualBox:~/PR/2/Course_Work/end$ ./main simpsonsvr.bmp -e --type 3
Introduced non-existent frame type
```

2. Невозможный размер рамки

```
alex@alex-VirtualBox:~/PR/2/Course_Work/end$ ./main simpsonsvr.bmp -e --width 0
Unacceptably frame width value
alex@alex-VirtualBox:~/PR/2/Course_Work/end$ ./main simpsonsvr.bmp -e --width 300
Unacceptably frame width value
```

3. Слишком маленькая рамка для конкретного типа

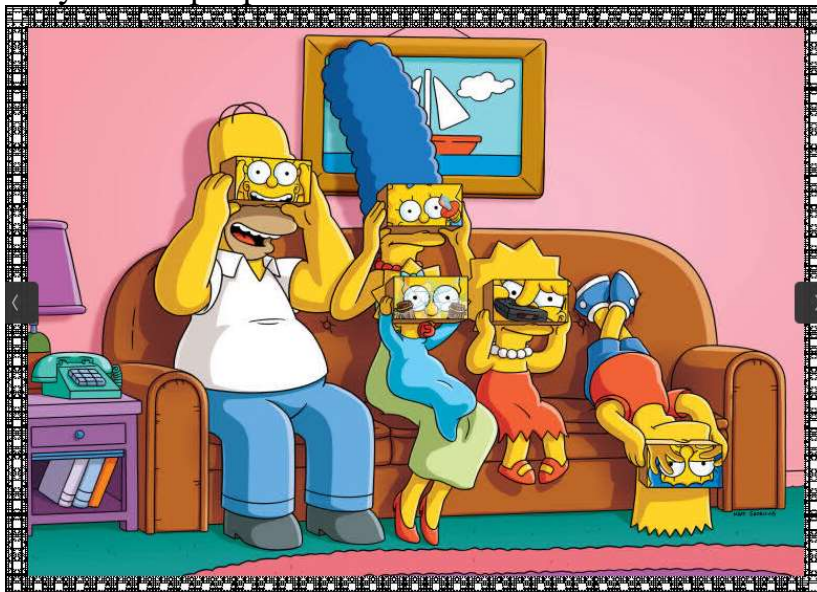
```
alex@alex-VirtualBox:~/PR/2/Course_Work/end$ ./main simpsonsvr.bmp -e --type 2 --width 1
Pattern number 2 requires a minimum border width of 17px
alex@alex-VirtualBox:~/PR/2/Course_Work/end$ ./main simpsonsvr.bmp -e --type 1 --width 1
Pattern number 1 requires a minimum border width of 10px
alex@alex-VirtualBox:~/PR/2/Course_Work/end$ ./main simpsonsvr.bmp -e --type 0 --width 1
alex@alex-VirtualBox:~/PR/2/Course_Work/end$
```

- **Edge**

Вводимая команда: `./main simpsonsvr.bmp -e`

Ожидаемый результат: Рисование рамки с значениями по умолчанию

Результат программы:



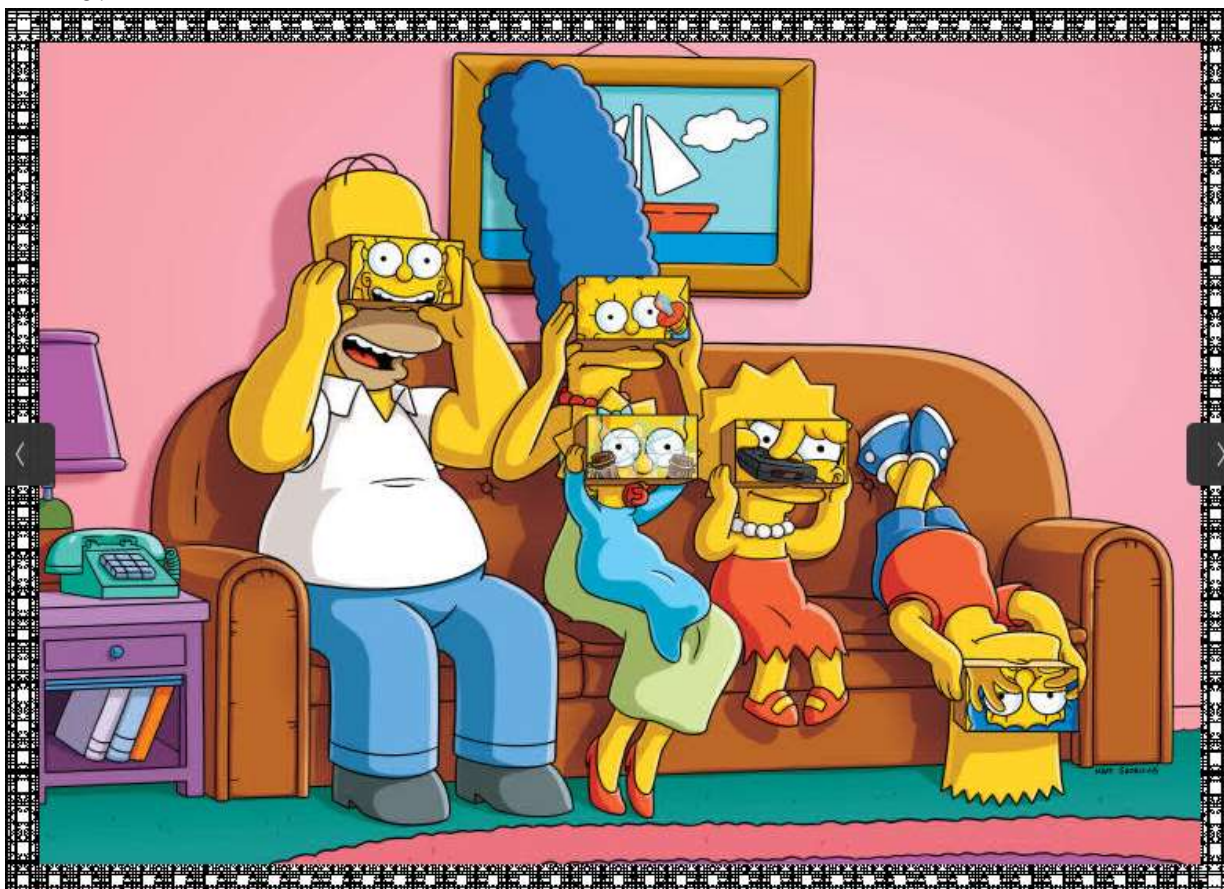
- **Edge + type**

Вводимая команда: `./main simpsonsvr.bmp -e -type 0/1/2`

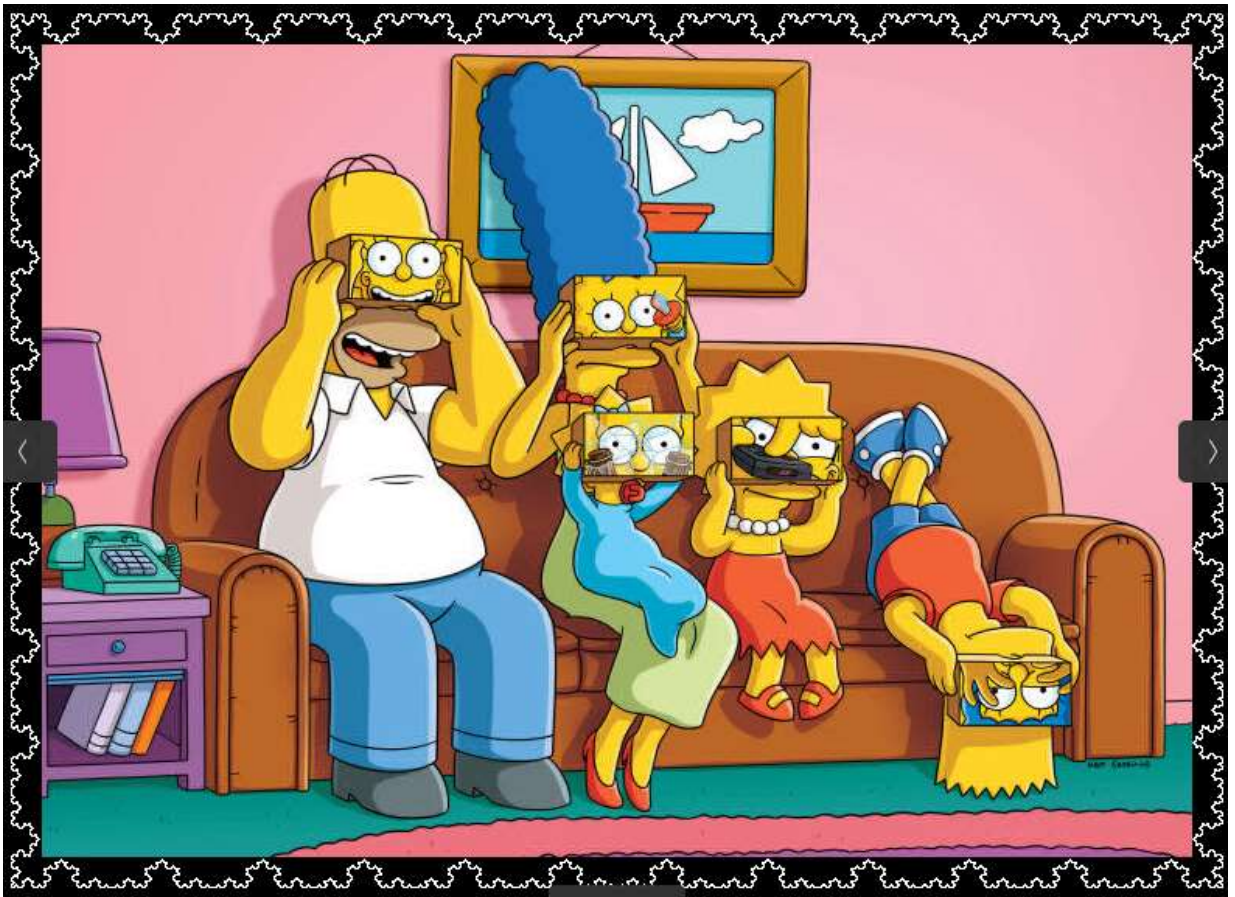
Ожидаемый результат: Рисование рамки заданного типа

Результат программы:

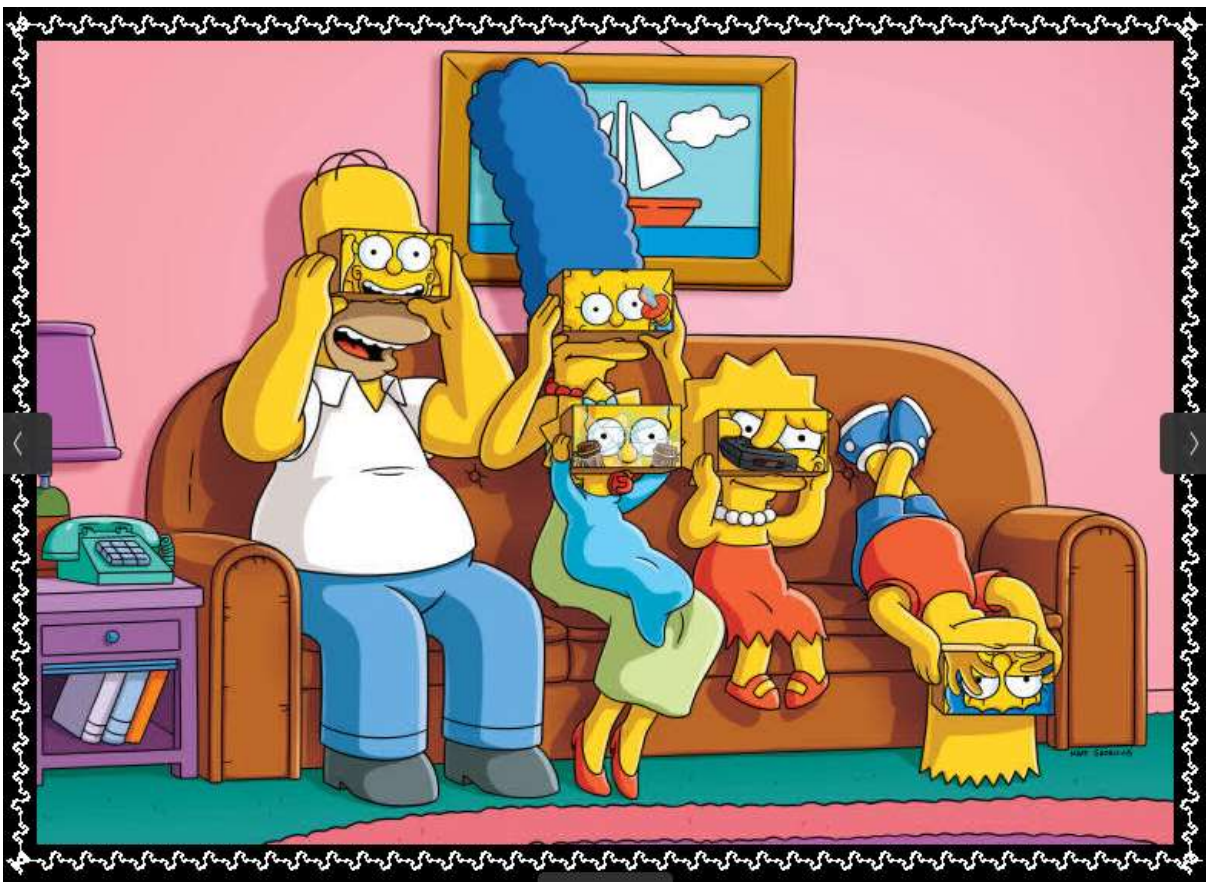
Тип 0:



Тип 1:



Тип 2:



- Edge + type + edgergb

Вводимая команда: `./main simpsonsvr.bmp -e -type 0 -edgergb 0 255 0`

Ожидаемый результат: Рисование рамки заданного типа и цвета узора

Результат программы:

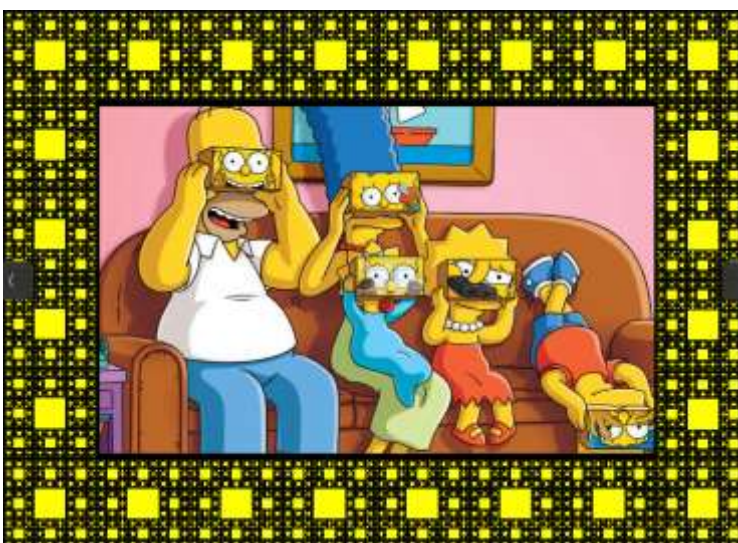


- **Edge + type + edgergb + width**

Вводимая команда: `./main simpsonsvr.bmp -e -type 0 -edgergb 255 255 0 -width 100`

Ожидаемый результат: Рисование рамки заданного типа и цвета узора и ширины

Результат программы:

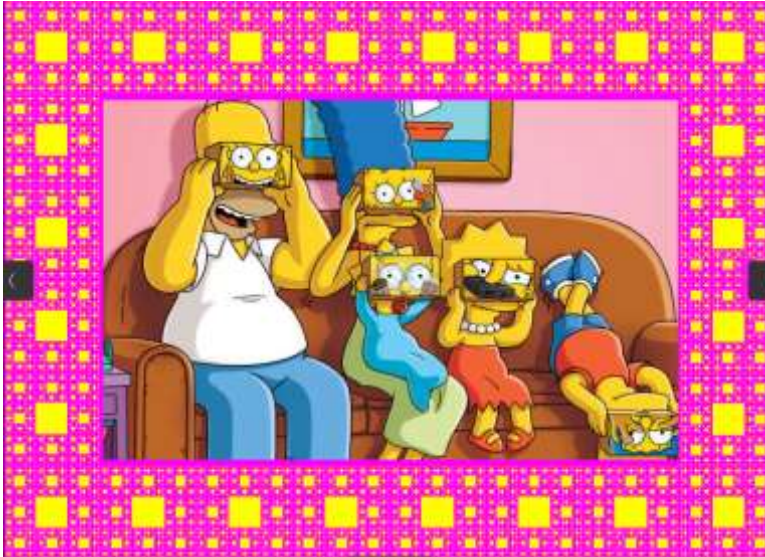


- **Edge + type + edgergb + width + bgrgb**

Вводимая команда: `./main simpsonsvr.bmp -e -type 0 -edgergb 255 255 0 -width 100 -bgrgb 255 0 255`

Ожидаемый результат: Рисование рамки заданного типа и цвета узора и ширины

Результат программы:



➤ 3) Копирование области

- **Особенные ошибки:**

1. **Область не принадлежит картинке**

```
alexgalex-VirtualBox: ~/PR/2/Course_Work/enc$ ./main simpsonsvr.bmp -c --start 500 400 --end 800 200 --insert 100 500
The specified area goes beyond the boundaries of the picture (min x = 0, min y = 0, max x = 779, max y = 567)
```

2. **Область невозможно вставить целиком**

```
alexgalex-VirtualBox: ~/PR/2/Course_Work/enc$ ./main simpsonsvr.bmp -c --start 400 400 --end 779 200 --insert 700 500
Unable to insert a copied area
```

- **Start + End + Insert**

Вводимая команда: `./main simpsonsvr.bmp -s --old 247 159 171 --new 255 255 255`

Ожидаемый результат: Копирование и вставка заданной области

Результат программы:



➤ 4) Смена цвета

- **Особенные ошибки:**

- 1. Замена несуществующего цвета**

Программа не выведет ошибки, в результате будет исходное изображение.

- **Old + New**

Вводимая команда: `./main simpsonsvr.bmp -s --old 247 159 171 --new 255 255 255`

Ожидаемый результат: Замена пикселей заданного цвета на белый

Результат программы:



ЗАКЛЮЧЕНИЕ

В результате проделанной работы была разработана программа на языке программирования Си, выполняющая обработку изображений в формате bmp. Для взаимодействия данной программы с пользователем был разработан интерфейс командной строки. В ходе выполнения работы было изучено, как файлы в формате bmp представлены в памяти компьютера, а также как взаимодействовать с файлами, используя язык программирования Си.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Керниган Б.В., Ричи Д.М. Язык С. М.: Вильямс, 2017.
2. Основная информация о языке программирования C++ // cplusplus.com:
<http://www.cplusplus.com/>
3. Библиотека среды выполнения С // Документация Microsoft Docs:
<https://docs.microsoft.com/ru-ru/cpp/c-runtime-library/c-run-time-library-reference?view=msvc-170>
4. Сайт Википедия: https://ru.wikipedia.org/wiki/Алгоритм_Брезенхэма
5. Сайт Википедия: https://ru.wikipedia.org/wiki/Треугольник_Серпинского
6. Сайт Википедия: https://ru.wikipedia.org/wiki/Кривая_Коха
7. Сайт Википедия: https://ru.wikipedia.org/wiki/Кривая_Минковского

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: *structs.h*

```
#pragma once
```

```
typedef struct Point{  
    int x;  
    int y;  
} Point;
```

```
typedef struct Rgb{  
    unsigned char b;  
    unsigned char g;  
    unsigned char r;  
} Rgb;
```

```
typedef struct Configs{  
    //hexagon  
    int draw_hexagon; // 0 - false  
    //copy  
    int copy_area; // 0 - false  
    //switch  
    int switch_color; // 0 - false  
    //frame  
    int draw_frame; // 0 - false  
    //files  
    char *input_file;  
    char *output_file; // default out.bmp
```

```

        int bmp_info;           // 0 - do not print
    } Configs;

typedef struct Hexagon{
    int flags[5];           //{center, radius, fill, start, end}

    //can be entered instead of center and radius
    Point start;
    Point end;

    Point center;
    int radius;
    int line_width;         //default 1px
    Rgb line_color;         //default black
    Rgb filling_color;
} Hexagon;

```

```

typedef struct Area{
    int flags[3];           //{start, end, insert}
    Point start;
    Point end;
    int h;
    int w;
    Point insert;
} Area;

```

```

typedef struct SwitchColor{

```

```
int flags[2];      //{old, new}  
    Rgb old_color;  
    Rgb new_color;  
} SwitchColor;
```

```
typedef struct Frame{  
    int type;      //default 0;  
    int width;     //default 10px;  
    Rgb edge_color;    //default white;  
    Rgb bg_color;     //default black;  
} Frame;
```

```
#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;
#pragma pack(pop)
```

```
typedef struct File{
    BitmapFileHeader *bmfh;
    BitmapInfoHeader *bmih;
    unsigned int H;
    unsigned int W;
    Rgb **arr;
} File;
```

Название файла: getopt.h

```
#pragma once
```

```
void PrintHelp();
int is_number(char *arg);
int is_rgb_numbers(int args[]);
int read_args(int argc, char *argv[], int *optind, int count, int args[]);
```

Название файла: getopt.c

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
```

```

void PrintHelp(){
    //hint
    printf("\n\t\x1b[32mGETOPT HINT\x1b[0m\n\n");

    //general

    printf("\x1b[33m_____
_____GENERAL_____
_____ \x1b[0m\n");

    printf("\t\x1b[33mGeneral\x1b[0m\n");
    printf("\x1b[35m-h -? --help\x1b[0m <-- help\n");
    printf("\x1b[35m--bmpversion\x1b[0m <-- supported bmp file versions\n");
    printf("\x1b[35m--bmpinfo\x1b[0m <-- bmp file information\n");

    printf("\x1b[35m-i\x1b[36m <filename.bmp> \x1b[35m--inputfile\x1b[36m
<filename.bmp>\x1b[0m <-- input BMP file (file to be changed)\n");
    printf("\x1b[35m-o\x1b[36m <filename.bmp> \x1b[35m--outputfile\x1b[36m
<filename.bmp>\x1b[0m <-- output BMP file (file were changes are saved)\n\n");

    //hexagon

    printf("\x1b[33m_____
_____HEXAGON_____
_____ \x1b[0m\n");

    printf("\t\x1b[33mHexagon\x1b[0m\n");
    printf("\x1b[35m-x --hexagon\x1b[0m <-- draw regular hexagon flag\n");
    printf("\t\x1b[35m--center\x1b[36m <x> <y>\x1b[0m <-- coordinates (type int) of
the center of the hexagon (the origin is in the lower left corner)\n");
    printf("\t\x1b[35m--radius\x1b[36m <radius>\x1b[0m <-- hexagon radius (type
int)\n");

```

printf("\n\x1b[32mInstead of the coordinates of the center and radius, you can set the coordinates of the left upper and right down corner of the square in which it is inscribed\x1b[0m\n");

printf("\t\x1b[35m--hexstart\x1b[36m <x> <y>\x1b[0m <-- coordinates (type int) of the upper left corner of the copy area (x, y)(the origin is in the lower left corner)\n");

printf("\t\x1b[35m--hexend\x1b[36m <x> <y>\x1b[0m <-- coordinates (type int) of the lower right corner of the copy area (x, y)(the origin is in the lower left corner)\n\n");

printf("\t\x1b[35m--linewidth\x1b[36m <width>\x1b[0m <-- boarder width (default: 1px)\n");

printf("\t\x1b[35m--linergb\x1b[36m <red> <green> <blue>\x1b[0m <-- boarder color (default: (0, 0, 0)) (each color component ranging from 0 to 255)\n");

printf("\t\x1b[35m-f\x1b[36m <red> <green> <blue> \x1b[35m--fill\x1b[36m <red> <green> <blue>\x1b[0m <-- hexagon fill color (each color component ranging from 0 to 255)\n\n");

//copy area

printf("\x1b[33m_____
_____COPY_AREA_____
_____ \x1b[0m\n");

printf("\t\x1b[33mCopy area\x1b[0m\n");

printf("\x1b[35m-c --copy\x1b[0m <-- copy area flag\n");

printf("\t\x1b[35m--start\x1b[36m <x> <y>\x1b[0m <-- coordinates (type int) of the upper left corner of the copy area (x, y)(the origin is in the lower left corner)\n");

printf("\t\x1b[35m--end\x1b[36m <x> <y>\x1b[0m <-- coordinates (type int) of the lower right corner of the copy area (x, y)(the origin is in the lower left corner)\n");

printf("\t\x1b[35m--insert\x1b[36m <x> <y>\x1b[0m <-- coordinates (type int) of the upper left corner of the paste area (x, y)\n\n");

//switch color

printf("\x1b[33m_____
_____SWITCH_COLOR_____
_____ \x1b[0m\n");

printf("\t\x1b[33mSwitch color\x1b[0m\n");

printf("\x1b[35m-s --switchcolor\x1b[0m <-- switch color flag\n");

printf("\t\x1b[35m--old\x1b[36m <red> <green> <blue>\x1b[0m <-- color to be change (each color component ranging from 0 to 255)\n");

printf("\t\x1b[35m--new\x1b[36m <red> <green> <blue>\x1b[0m <-- color to be applied (each color component ranging from 0 to 255)\n\n");

//frame

printf("\x1b[33m_____
_____FRAME_____
_____ \x1b[0m\n");

printf("\t\x1b[33mFrame\x1b[0m\n");

printf("\x1b[35m-e --edge\x1b[0m <-- frame flag\n");

printf("\t\x1b[35m--type\x1b[36m <pattern>\x1b[0m> <-- frame pattern {0, 1, 2} (default: 0)\n");

printf("\t\x1b[35m--width\x1b[36m <width>\x1b[0m> <-- frame width (default: 20px)\n");

```

    printf("\t\x1b[35m--edgergb\x1b[36m <red> <green> <blue>\x1b[0m <-- frame
color (default: (255, 255, 255)) (each color component ranging from 0 to 255)\n");
    printf("\t\x1b[35m--bgrgb\x1b[36m <red> <green> <blue>\x1b[0m <--
background frame color (default: (0, 0, 0)) (each color component ranging from 0 to
255)\n\n");
}

```

```

int is_number(char *arg){
    if (atoi(arg) || (!atoi(arg) && isdigit(arg[0]))){
        return 1;
    }
    return 0;
}

```

```

int is_rgb_numbers(int args[]){
    //iterate over all rgb arguments
    for (int i = 0; i < 3; i++){

        //conformity check
        if (args[i]<0 || args[i]>255){
            printf("\x1b[31mArgumentes <red> <green> <blue> must be in the range
from 0 to 255\x1b[0m\n");
            return 0;
        }
    }
    return 1;
}

```

```

int read_args(int argc, char *argv[], int *optind, int count, int args[]){
    (*optind)--;

```

```

    int i = 0;

    //iterate over all arguments passed after the flag
    for (;*optind < argc && *argv[*optind]!='-' && i < count; (*optind)++){

        //type matching check
        if (is_number(argv[*optind])){
            args[i++] = atoi(argv[*optind]);
        } else{
            return -1;
        }
    }

    return i;
}

```

Название файла: getopt_copy.h

```
#pragma once
```

```

int case_start_copy(int argc, char *argv[], int *optind, Configs *config, Area *area);
int case_end_copy(int argc, char *argv[], int *optind, Configs *config, Area *area);
int case_insert(int argc, char *argv[], int *optind, Configs *config, Area *area);

```

Название файла: getopt_copy.c

```

#include <stdio.h>
#include <getopt.h>
#include "getopt.h"
#include "structs.h"

```

```

int case_start_copy(int argc, char *argv[], int *optind, Configs *config, Area
*area){
    //flag passing check
    if (config->copy_area){
        int args[2]; //temporary array of arguments
        int count = read_args(argc, argv, optind, 2, args); //number of
arguments read

        //error checking
        if (count == -1){
            printf("\x1b[31mFlag {--start} takes arguments of type
int\x1b[0m\n");
            return -1;
        }

        //checking if all arguments are passed
        if (count == 2){
            area->start.x = args[0];
            area->start.y = args[1];
            return 0;
        } else{
            printf("\x1b[31mFlag {--start} taken 2 argument, but was given less
argument\x1b[0m\n");
            return -1;
        }
    } else{
        printf("\x1b[31mCopy flag {-c}/{--copy} not specified!\x1b[0m\n");
        return -1;
    }
}
}

```

```

int case_end_copy(int argc, char *argv[], int *optind, Configs *config, Area *area){
    //flag passing check
    if (config->copy_area){
        int args[2]; //temporary array of arguments
        int count = read_args(argc, argv, optind, 2, args);    //number of
arguments read

        //error checking
        if (count == -1){
            printf("\x1b[31mFlag {--end} takes arguments of type
int\x1b[0m\n");
            return -1;
        }

        //checking if all arguments are passed
        if (count == 2){
            area->end.x = args[0];
            area->end.y = args[1];
            return 0;
        } else{
            printf("\x1b[31mFlag {--end} taken 2 argument, but was given
less argument\x1b[0m\n");
            return -1;
        }
    } else{
        printf("\x1b[31mCopy flag {-c}/{--copy} not specified!\x1b[0m\n");
        return -1;
    }
}

```


}

```
int case_insert(int argc, char *argv[], int *optind, Configs *config, Area *area){
    //flag passing check
    if (config->copy_area){
        int args[2]; //temporary array of arguments
        int count = read_args(argc, argv, optind, 2, args);    //number of
arguments read

        //error checking
        if (count == -1){
            printf("\x1b[31mFlag {--insert} takes arguments of type
int\x1b[0m\n");
            return -1;
        }

        //checking if all arguments are passed
        if (count == 2){
            area->insert.x = args[0];
            area->insert.y = args[1];
            return 0;
        } else{
            printf("\x1b[31mFlag {--insert} taken 2 argument, but was given
less argument\x1b[0m\n");
            return -1;
        }
    } else{
        printf("\x1b[31mCopy flag {-c}/{--copy} not specified!\x1b[0m\n");
        return -1;
    }
}
```

```

    }
}

```

Название файла: *getopt_switch_color.h*

#pragma once

```

int case_old_color(int argc, char *argv[], int *optind, Configs *config, SwitchColor
*switch_color);
int case_new_color(int argc, char *argv[], int *optind, Configs *config, SwitchColor
*switch_color);

```

Название файла: *getopt_switch_color.c*

#include <stdio.h>

#include "getopt.h"

#include "structs.h"

```

int case_old_color(int argc, char *argv[], int *optind, Configs *config, SwitchColor
*switch_color){
    //flag passing check
    if (config->switch_color){
        int args[3]; //temporary array of arguments
        int count = read_args(argc, argv, optind, 3, args);    //number of
arguments read

        //error checking
        if (count == -1){
            printf("\x1b[31mFlag {--old} takes arguments of type
int\x1b[0m\n");
            return -1;

```

```

    }

    //checking if all arguments are passed
    if(count == 3){

        if(!is_rgb_numbers(args)){
            return -1;
        }

        switch_color->old_color = (Rgb){args[2], args[1], args[0]};
        return 0;
    } else{
        printf("\x1b[31mFlag {--old} taken 3 argument, but was given less
argument\x1b[0m\n");
        return -1;
    }
} else{
    printf("\x1b[31mSwitch color flag {-s}/{--switchcolor} not
specified!\x1b[0m\n");
    return -1;
}
}

```

```

int case_new_color(int argc, char *argv[], int *optind, Configs *config, SwitchColor
*switch_color){
    //flag passing check
    if (config->switch_color){
        int args[3]; //temporary array of arguments
    }
}

```

```

        int count = read_args(argc, argv, optind, 3, args);    //number of
arguments read

        //error checking
        if(count == -1){
            printf("\x1b[31mFlag {--new} takes arguments of type
int\x1b[0m\n");
            return -1;
        }

        //checking if all arguments are passed
        if(count == 3){

            if(!is_rgb_numbers(args)){
                return -1;
            }

            switch_color->new_color = (Rgb){args[2], args[1], args[0]};
            return 0;
        } else{
            printf("\x1b[31mFlag {--new} taken 3 argument, but was given
less argument\x1b[0m\n");
            return -1;
        }
    } else{
        printf("\x1b[31mSwitch color flag {-s}/{--switchcolor} not
specified!\x1b[0m\n");
        return -1;
    }
}

```

Название файла: *getopt_hexagon.h*

#pragma once

```
int case_hexstart(int argc, char *argv[], int *optind, Configs *config, Hexagon
*hexagon);
int case_hexend(int argc, char *argv[], int *optind, Configs *config, Hexagon
*hexagon);
int case_hexcenter(int argc, char *argv[], int *optind, Configs *config, Hexagon
*hexagon);
int case_hexradius(int argc, char *argv[], char *optarg, Configs *config, Hexagon
*hexagon);
int case_hexline_width(int argc, char *argv[], char *optarg, Configs *config,
Hexagon *hexagon);
int case_hexline_color(int argc, char *argv[], int *optind, Configs *config, Hexagon
*hexagon);
int case_hexfilling(int argc, char *argv[], int *optind, Configs *config, Hexagon
*hexagon);
```

Название файла: *getopt_hexagon.c*

#include <stdio.h>

#include <stdlib.h>

#include "getopt.h"

#include "structs.h"

```
int case_hexstart(int argc, char *argv[], int *optind, Configs *config, Hexagon
*hexagon){
    //flag passing check
    if (config->draw_hexagon){
        int args[2]; //temporary array of arguments
```

```

        int count = read_args(argc, argv, optind, 2, args);    //number of
arguments read

        //error checking
        if (count == -1){
            printf("\x1b[31mFlag {--hexstart} takes arguments of type
int\x1b[0m\n");
            return -1;
        }

        //checking if all arguments are passed
        if (count == 2){
            hexagon->start.x = args[0];
            hexagon->start.y = args[1];
            return 0;
        } else{
            printf("\x1b[31mFlag {--hexstart} taken 2 argument, but was given
less argument\x1b[0m\n");
            return -1;
        }
    } else{
        printf("\x1b[31mCopy flag {-x}/{--hexagon} not specified!\x1b[0m\n");
        return -1;
    }
}

```

```

int case_hexend(int argc, char *argv[], int *optind, Configs *config, Hexagon
*hexagon){
    //flag passing check

```

```

if (config->draw_hexagon){
    int args[2]; //temporary array of arguments
    int count = read_args(argc, argv, optind, 2, args); //number of
arguments read

    //error checking
    if (count == -1){
        printf("\x1b[31mFlag {--hexend} takes arguments of type
int\x1b[0m\n");
        return -1;
    }

    //checking if all arguments are passed
    if (count == 2){
        hexagon->end.x = args[0];
        hexagon->end.y = args[1];
        return 0;
    } else{
        printf("\x1b[31mFlag {--hexend} taken 2 argument, but was given less
argument\x1b[0m\n");
        return -1;
    }
} else{
    printf("\x1b[31mCopy flag {-x}/{--hexagon} not specified!\x1b[0m\n");
    return -1;
}
}

```

```

int case_hexcenter(int argc, char *argv[], int *optind, Configs *config, Hexagon
*hexagon){
    //flag passing check
    if (config->draw_hexagon){
        int args[2]; //temporary array of arguments
        int count = read_args(argc, argv, optind, 2, args);    //number of
arguments read

        //error checking
        if (count == -1){
            printf("\x1b[31mFlag {--center} takes arguments of type
int\x1b[0m\n");
            return -1;
        }

        //checking if all arguments are passed
        if (count == 2){
            hexagon->center.x = args[0];
            hexagon->center.y = args[1];
            return 0;
        } else{
            printf("\x1b[31mFlag {--center} taken 2 argument, but was given
less argument\x1b[0m\n");
            return -1;
        }
    } else{
        printf("\x1b[31mHexagon flag {-x}/{--hexagon} not
specified!\x1b[0m\n");
        return -1;
    }
}

```


}

```
int case_hexradius(int argc, char *argv[], char *optarg, Configs *config, Hexagon
*hexagon){
    //flag passing check
    if (config->draw_hexagon){

        //error checking
        if (!is_number(optarg)){
            printf("\x1b[31mFlag {--radius} takes argument of type
int\x1b[0m\n");
            return -1;
        }

        hexagon->radius = atoi(optarg);
        return 0;
    } else{
        printf("\x1b[31mHexagon flag {-x}/{--hexagon} not
specified!\x1b[0m\n");
        return -1;
    }
}
```

```
int case_hexline_width(int argc, char *argv[], char *optarg, Configs *config,
Hexagon *hexagon){
    //flag passing check
    if (config->draw_hexagon){
```

```

        //error checking
        if (!is_number(optarg)){
            printf("\x1b[31mFlag {--linewidth} takes argument of type
int\x1b[0m\n");
            return -1;
        }

        hexagon->line_width = atoi(optarg);
        return 0;
    } else{
        printf("\x1b[31mHexagon flag {-x}/{--hexagon} not specified!\x1b[0m\n");
        return -1;
    }
}

```

```

int case_hexline_color(int argc, char *argv[], int *optind, Configs *config, Hexagon
*hexagon){
    //flag passing check
    if (config->draw_hexagon){
        int args[3]; //temporary array of arguments
        int count = read_args(argc, argv, optind, 3, args); //number of
arguments read

        //error checking
        if (count == -1){
            printf("\x1b[31mFlag {--linergb} takes arguments of type
int\x1b[0m\n");
            return -1;
        }
    }
}

```

```

    if (!is_rgb_numbers(args)){
        return -1;
    }

    //checking if all arguments are passed
    if (count == 3){
        hexagon->line_color = (Rgb){args[2], args[1], args[0]};
        return 0;
    } else{
        printf("\x1b[31mFlag {--linergb} taken 3 argument, but was given less
argument\x1b[0m\n");
        return -1;
    }
} else{
    printf("\x1b[31mHexagon flag {-x}/{--hexagon} not specified!\x1b[0m\n");
    return -1;
}
}

```

```

int case_hexfilling(int argc, char *argv[], int *optind, Configs *config, Hexagon
*hexagon){
    //flag passing check
    if (config->draw_hexagon){
        int args[3]; //temporary array of arguments
        int count = read_args(argc, argv, optind, 3, args);    //number of
arguments read

        //error checking

```

```

        if (count == -1){
            printf("\x1b[31mFlag {-f}/{--fill} takes arguments of type
int\x1b[0m\n");
            return -1;
        }

        if (!is_rgb_numbers(args)){
            return -1;
        }

        //checking if all arguments are passed
        if (count == 3){
            hexagon->filling_color = (Rgb){args[2], args[1], args[0]};
            return 0;
        } else{
            printf("\x1b[31mFlag {-f}/{--fill} taken 3 argument, but was given
less argument\x1b[0m\n");
            return -1;
        }
    } else{
        printf("\x1b[31mHexagon flag {-x}/{--hexagon} not
specified!\x1b[0m\n");
        return -1;
    }
}

```

Название файла: getopt_frame.h

#pragma once

```

int case_edge_type(int argc, char *argv[], char *optarg, Configs *config, Frame
*frame);
int case_edge_width(int argc, char *argv[], char *optarg, Configs *config, Frame
*frame);
int case_edge_color(int argc, char *argv[], int *optind, Configs *config, Frame
*frame);
int case_bg_color(int argc, char *argv[], int *optind, Configs *config, Frame
*frame);

```

Название файла: getopt_frame.c

```

#include <stdio.h>
#include <stdlib.h>
#include "getopt.h"
#include "structs.h"

```

```

int case_edge_type(int argc, char *argv[], char *optarg, Configs *config, Frame
*frame){
    //flag passing check
    if (config->draw_frame){

        //error checking
        if (!is_number(optarg)){
            printf("\x1b[31mFlag {--type} takes arguments of type
int\x1b[0m\n");
            return -1;
        }

        frame->type = atoi(optarg);
        return 0;
    }
}

```

```

    } else{
        printf("\x1b[31mDraw frame flag {-e}/{--edge} not
specified!\x1b[0m\n");
        return -1;
    }
}

```

```

int case_edge_width(int argc, char *argv[], char *optarg, Configs *config, Frame
*frame){
    //flag passing check
    if (config->draw_frame){

        //error checking
        if (!is_number(optarg)){
            printf("\x1b[31mFlag {--breadth} takes arguments of type
int\x1b[0m\n");
            return -1;
        }

        frame->width = atoi(optarg);
        return 0;
    } else{
        printf("\x1b[31mDraw frame flag {-e}/{--edge} not
specified!\x1b[0m\n");
        return -1;
    }
}

```

```

int case_edge_color(int argc, char *argv[], int *optind, Configs *config, Frame
*frame){
    //flag passing check
    if (config->draw_frame){
        int args[3]; //temporary array of arguments
        int count = read_args(argc, argv, optind, 3, args);    //number of
arguments read

        //error checking
        if (count == -1){
            printf("\x1b[31mFlag {--edgergb} takes arguments of type
int\x1b[0m\n");
            return -1;
        }

        //checking if all arguments are passed
        if (count == 3){
            if (!is_rgb_numbers(args)){
                return -1;
            }

            frame->edge_color = (Rgb){args[2], args[1], args[0]};
            return 0;
        } else{
            printf("\x1b[31mFlag {--edgergb} taken 3 argument, but was
given less argument\x1b[0m\n");
            return -1;
        }
    } else{

```

```

        printf("\x1b[31mDraw frame flag {-e}/{--edge} not
specified!\x1b[0m\n");
        return -1;
    }
}

```

```

int case_bg_color(int argc, char *argv[], int *optind, Configs *config, Frame
*frame){
    //flag passing check
    if (config->draw_frame){
        int args[3]; //temporary array of arguments
        int count = read_args(argc, argv, optind, 3, args);    //number of
arguments read

        //error checking
        if (count == -1){
            printf("\x1b[31mFlag {--bgrgb} takes arguments of type
int\x1b[0m\n");
            return -1;
        }

        //checking if all arguments are passed
        if (count == 3){
            if (!is_rgb_numbers(args)){
                return -1;
            }

            frame->bg_color = (Rgb){args[2], args[1], args[0]};
            return 0;
        }
    }
}

```



```

        } else{
            printf("\x1b[31mFlag {--bgrgb} taken 3 argument, but was given
less argument\x1b[0m\n");
            return -1;
        }
    } else{
        printf("\x1b[31mDraw frame flag {-e}/{--edge} not
specified!\x1b[0m\n");
        return -1;
    }
}

```

Название файла:*getopt_files.h*

#pragma once

```

int is_bmp(char *arg);
int case_input_file(Configs *config, char *optarg);
int case_output_file(Configs *config, char *optarg);
int search_input_file(int argc, char *argv[], int *optind, Configs *config);
void no_input_file();

```

Название файла:*getopt_files.c*

```

#include <stdio.h>
#include <string.h>
#include "getopt.h"
#include "structs.h"

```

```

int is_bmp(char *arg){
    if (strstr(arg, ".bmp") != NULL){

```

```

        return 1;
    }

    return 0;
}

int case_input_file(Configs *config, char *optarg){
    if (is_bmp(optarg)){
        strcpy(config->input_file, optarg);
        return 0;
    } else{
        printf("%s - \x1b[31mInput file is not BMP!\x1b[0m\n", optarg);
        return -1;
    }
}

int case_output_file(Configs *config, char *optarg){
    if (is_bmp(optarg)){
        strcpy(config->output_file, optarg);
        return 0;
    } else{
        printf("%s - \x1b[31mOutput file is not BMP!\x1b[0m\n", optarg);
        return -1;
    }
}

int search_input_file(int argc, char *argv[], int *optind, Configs *config){

```

```

    argc -= (*optind);
    argv += (*optind);

    //checking for the presence of the name of the multable file
    if (!strstr(config->input_file, ".bmp")){
        for(int i=0; i<argc; i++){
            if (is_bmp(argv[i])){
                strcpy(config->input_file, argv[i]);
                return 1;
            }
        }
    } else{
        return 1;
    }

    return 0;
}

```

```

void no_input_file(){
    printf("\x1b[31mInput file name wasn't given!\x1b[0m\n");
}

```

Название файла: *file_processing.h*

#pragma once

```

int read_input_file(Configs *comnfig, File *file, BitmapFileHeader *bmfh,
BitmapInfoHeader *bmih);
int create_output_file(Configs *config, File *file);

```

Название файла: *file_processing.c*

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include "structs.h"
```

```
#include "bmp_info.h"
```

```
int read_input_file(Configs *config, File *file, BitmapFileHeader *bmfh,  
BitmapInfoHeader *bmih){
```

```
    FILE *f = fopen(config->input_file, "rb");
```

```
    //error checking
```

```
    if (f == NULL){
```

```
        printf("\x1b[31mUnable to open input file\x1b[0m\n");
```

```
        return -1;
```

```
    }
```

```
    //reading headlines
```

```
    fread(bmfh,1,sizeof(BitmapFileHeader),f);
```

```
    fread(bmih,1,sizeof(BitmapInfoHeader),f);
```

```
    //getting height and width
```

```
    file->H = bmih->height;
```

```
    file->W = bmih->width;
```

```
    //pictures reading
```

```
    Rgb **arr = malloc(file->H*sizeof(Rgb*));
```

```
    //error checking
```

```

    if (arr == NULL){
        printf("\x1b[31mH Unable to allocate memmory for image\x1b[0m\n");
        return -1;
    }

    for(int i=0; i<file->H; i++){
        arr[i] = malloc(file->W * sizeof(Rgb) + (file->W*3)%4);

        //error checking
        if (arr[i] == NULL){
            printf("\x1b[31mW Unable to allocate memmory for
image\x1b[0m\n");
            return -1;
        }
        fread(arr[i],1,file->W * sizeof(Rgb) + (file->W*3)%4,f);
    }
    file->arr = arr;
    fclose(f);
    return 0;
}

```

```

int create_output_file(Configs *config, File *file){
    FILE *f = fopen(config->output_file, "wb");

    //error checking
    if (f == NULL){
        printf("\x1b[31mUnable to create output file\x1b[0m\n");
        return -1;
    }
}

```

```

//writing headlines
fwrite(file->bmfh, 1, sizeof(BitmapFileHeader),f);
fwrite(file->bmih, 1, sizeof(BitmapInfoHeader),f);
unsigned int w = file->W * sizeof(Rgb) + (file->W*3)%4;

for(int i=0; i<file->H; i++){
    fwrite(file->arr[i],1,w,f);
}

fclose(f);
return 0;
}

```

Название файла: bmp_info.h

```
#pragma once
```

```

void printFileHeader(BitmapFileHeader header);
void printInfoHeader(BitmapInfoHeader header);

```

Название файла: bmp_info.c

```
#include <stdio.h>
```

```
#include "structs.h"
```

```

void printFileHeader(BitmapFileHeader header){
    printf("\tFileHeader\n");
    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 printInfoHeader(BitmapInfoHeader header){
    printf("\tInfoHeader\n");
    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);
}

```

Название файла: copy_area.h

```
#pragma once
```

```

int check_correct_copy_coordinates(Area *area, File *file);
Rgb **create_cpy_area(Rgb **arr, Area *area);
void free_copy_area(Rgb **arr, Area *area);
int CopyAreaFunc(Rgb **arr, Area *area, File *file);

```

Название файла: copy_area.c

```

#include <stdio.h>
#include <stdlib.h>
#include "structs.h"
#include "getopt.h"

```

```

int check_correct_copy_coordinates(Area *area, File *file){
    //checking that all coordinates of the copy area are in pictures
    if ((area->start.x < 0) || (area->start.x > file->W-1) || (area->start.y < 0) ||
        (area->start.y > file->H-1) || (area->end.x < 0) || (area->end.x > file->W-1) ||
        (area->end.y < 0) || (area->end.y > file->H-1)){
        printf("\x1b[31mThe specified area goes beyond the boundaries of the
picture (min x = 0, min y = 0, max x = %d, max y = %d)\x1b[0m\n", file->W-1, file-
>H-1);

        return -1;
    }

    //checking the correctness of the entered coordinates of the left upper corner
and right lower corner
    if ((area->start.x > area->end.x) || (area->start.y < area->end.y)){
        printf("\x1b[31mIncorrect coordinates of the left upper corner and the
right lower corner (the origin is in the lower left corner)\x1b[0m\n");

        return -1;
    }

    //checking the possibility of inserting a selected area
    if ((area->insert.x < 0) || (area->insert.x > file->W-1) || (area->insert.y < 0)
        || (area->insert.y > file->H-1) || (area->insert.x + area->w > file->W-1) || (area-
        >insert.y - area->h < 0)){
        printf("\x1b[31mUnable to insert a copied area\x1b[0m\n");

        return -1;
    }

    return 0;
}

```



```

Rgb **create_cpy_area(Rgb **arr, Area *area){
    //seting initial coordinates for copying
    int y_start = area->end.y;
    int x_start = area->start.x;

    //copy area
    Rgb **cpy_arr = malloc(area->h * sizeof(Rgb*));
    if (cpy_arr == NULL){
        printf("\x1b[31mh Unable to allocate memmory for
cpy_arr\x1b[0m\n");
        return NULL;
    }

    for(int i = 0; i < area->h; i++){
        cpy_arr[i] = malloc(area->w * sizeof(Rgb));
        if (cpy_arr[i] == NULL){
            printf("\x1b[31mw Unable to allocate memmory for
cpy_arr\x1b[0m\n");
            return NULL;
        }

        for (int j = 0; j < area->w; j++){
            cpy_arr[i][j] = arr[y_start+i][x_start+j];
        }
    }

    return cpy_arr;
}

```

```

void free_copy_area(Rgb **arr, Area *area){
    for (int i = 0; i < area->h; i++){
        free(arr[i]);
    }
    free(arr);
}

```

```

int CopyAreaFunc(Rgb **arr, Area *area, File *file){
    //checking if all flags are passed
    if (!area->flags[0] || !area->flags[1] || !area->flags[2]){
        printf("\x1b[31mNot all necessary flags for coping were
passed\x1b[0m\n");
        return -1;
    }

```

```

    //getting the height and width of the copied area
    area->w = abs(area->end.x - area->start.x) + 1;
    area->h = abs(area->start.y - area->end.y) + 1;

```

```

    if (check_correct_copy_coordinates(area, file) == -1){
        return -1;
    }

```

```

    Rgb **cpy_arr = create_cpy_area(arr, area);

```

```

    //checking if the area was copied to an array
    if (cpy_arr == NULL){

```

```

        return -1;
    }

    //setting initial coordinates for pasting
    int y_insert = area->insert.y - area->h + 1;
    int x_insert = area->insert.x;

    //insert area
    for (int i = 0; i < area->h; i++){
        for (int j = 0; j < area->w; j++){
            arr[y_insert+i][x_insert+j].b = cpy_arr[i][j].b;
            arr[y_insert+i][x_insert+j].g = cpy_arr[i][j].g;
            arr[y_insert+i][x_insert+j].r = cpy_arr[i][j].r;
        }
    }

    //free copy array
    free_copy_area(cpy_arr, area);

    return 0;
}

```

Название файла: draw_common_func.h

#pragma once

```

void set_pixel(Rgb *pixel, Rgb *new);
int is_check_color(Rgb *current, Rgb *new);
void draw_rectangle(int x0, int y0, int x1, int y1, Rgb **arr, Rgb *color);
void draw_line(int x0, int y0, int x1, int y1, Rgb **arr, Rgb *color);

```

Название файла: draw_common_func.c

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include "structs.h"
```

```
void set_pixel(Rgb *pixel, Rgb *new){
```

```
    pixel->b = new->b;
```

```
    pixel->g = new->g;
```

```
    pixel->r = new->r;
```

```
}
```

```
int is_check_color(Rgb *current, Rgb *new){
```

```
    if (current->b == new->b && current->g == new->g && current->r ==  
new->r){
```

```
        return 1;
```

```
    }
```

```
    return 0;
```

```
}
```

```
void draw_rectangle(int x0, int y0, int x1, int y1, Rgb **arr, Rgb *color){
```

```
    for (int i = y0; i < y1; i++){
```

```
        for (int j = x0; j < x1; j++){
```

```
            set_pixel(&arr[i][j], color);
```

```
        }
```

```
    }
```

```
}
```

```

void draw_line(int x0, int y0, int x1, int y1, Rgb **arr, Rgb *color){
    int delta_x = x1 - x0;
    int delta_y = y1 - y0;
    int slope;

    //checking which coordinate to change premanently
    if (abs(delta_y) > abs(delta_x)){
        slope = 1;
    } else{
        slope = -1;
    }

    int dir_y, dir_x;

    //choice to direction of change y
    if (delta_y > 0){
        dir_y = 1;
    } else{
        dir_y = -1;
    }

    //choice to direction of change x
    if (delta_x > 0){
        dir_x = 1;
    } else{
        dir_x = -1;
    }

    int error = 0;

```

```

int x = x0;
int y = y0;
set_pixel(&arr[y][x], color);

if (slope == -1){
    do{
        error += delta_y * dir_y;
        set_pixel(&arr[y][x+dir_x], color);
        if (error > 0){
            error -= delta_x * dir_x;
            y += dir_y;
        }
        x += dir_x;
        set_pixel(&arr[y][x], color);
    } while (x != x1 || y != y1);
} else{
    do{
        error += delta_x * dir_x;
        set_pixel(&arr[y+dir_y][x], color);
        if (error > 0){
            error -= delta_y * dir_y;
            x += dir_x;
        }
        y += dir_y;
        set_pixel(&arr[y][x], color);
    } while (x != x1 || y != y1);
}
}

```

Название файла: draw_frame.h

#pragma once

*void draw_bg_frame(Frame *frame, Rgb **arr, File *file);*

*int DrawFrame(Frame *frame, Rgb **arr, File *file);*

Название файла: draw_frame.c

#include <stdio.h>

#include <stdlib.h>

#include "getopt.h"

#include "structs.h"

#include "draw_frame.h"

#include "draw_common_func.h"

#include "draw_frame_serp.h"

#include "draw_frame_koch.h"

#include "draw_frame_minc.h"

*void draw_bg_frame(Frame *frame, Rgb **arr, File *file){*

//defining borders of the frame

int up = file->H-1 - frame->width;

int down = 0 + frame->width;

int left = 0 + frame->width;

int right = file->W-1 - frame->width;

//draw background frame

for (int i = 0; i < file->H; i++){

for (int j = 0; j < file->W; j++){

if (i < down || i > up){

set_pixel(&arr[i][j], &frame->bg_color);

}

```

        if (j < left || j > right){
            set_pixel(&arr[i][j], &frame->bg_color);
        }
    }
}

```

```

int DrawFrame(Frame *frame, Rgb **arr, File *file){
    //validation of border width value
    if (frame->width > file->W/2 || frame->width > file->H/2 || frame->width <=
0){
        printf("\x1b[31mUnacceptably frame width value\x1b[0m\n");
        return -1;
    }

    //draw background frame
    draw_bg_frame(frame, arr, file);

    //draw pattern
    switch (frame->type){
        case 0:
            draw_frame_serp(frame, arr, file);
            break;

        case 1:
            //checking minimum border
            if (frame->width < 10){
                printf("\x1b[31mPattern number 1 requires a minimum
border width of 10px\x1b[0m\n");

```



```

        return -1;
    }
    draw_frame_koch(frame, arr, file);
    break;

case 2:
    //checking minimum border
    if (frame->width < 17){
        printf("\x1b[31mPattern number 2 requires a minimum
border width of 17px\x1b[0m\n");
        return -1;
    }
    draw_frame_minc(frame, arr, file);
    break;

default:
    printf("\x1b[31mIntroduced non-existen frame type\x1b[0m\n");
    return -1;
}

return 0;
}

```

Название файла: draw_frame_koch.h

#pragma once

```

void draw_koch_curve(double x0, double y0, double x1, double y1, int n, double
angle, Rgb **arr, Rgb *color);
void draw_frame_koch(Frame *frame, Rgb **arr, File *file);

```

Название файла: draw_frame_koch.c

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <math.h>
```

```
#include "structs.h"
```

```
#include "draw_common_func.h"
```

```
void draw_koch_curve(double x0, double y0, double x1, double y1, int n, double  
angle, Rgb **arr, Rgb *color){
```

```
    if (n == 0){
```

```
        //draw line
```

```
        draw_line((int)x0, (int)y0, (int)x1, (int)y1, arr, color);
```

```
        return;
```

```
    }
```

```
    //delta
```

```
    double delta_x = (x1 - x0)/3;
```

```
    double delta_y = (y1 - y0)/3;
```

```
    //new coordinates
```

```
    double new_x0 = x0 + delta_x;
```

```
    double new_y0 = y0 + delta_y;
```

```
    double new_x1 = x1 - delta_x;
```

```
    double new_y1 = y1 - delta_y;
```

```
    //middle
```

```
    double mid_x = (delta_x/2) - (delta_y*sin(angle)) + new_x0;
```

```
    double mid_y = (delta_y/2) + (delta_x*sin(angle)) + new_y0;
```

```
    //recursion
```

```

    draw_koch_curve(x0, y0, new_x0, new_y0, n-1, angle, arr, color);
    draw_koch_curve(new_x0, new_y0, mid_x, mid_y, n-1, angle, arr, color);
    draw_koch_curve(mid_x, mid_y, new_x1, new_y1, n-1, angle, arr, color);
    draw_koch_curve(new_x1, new_y1, x1, y1, n-1, angle, arr, color);

}

```

```

void draw_frame_koch(Frame *frame, Rgb **arr, File *file){
    int n = 3;

    //getting the best parameters
    double step_x = 3*(2*frame->width/sqrt(3));
    double step_y = 3*(2*frame->width/sqrt(3));
    int count_x = ceil(file->W/step_x);
    int count_y = ceil(file->H/step_y);

    while (file->W-1 < count_x*step_x){
        step_x -= 0.0001;
    }
    while (file->H-1 < count_y*step_y){
        step_y -= 0.0001;
    }

    double angle = M_PI/3;

    //draw frame Koch
    for (int i = 0; i < count_x; i++){
        //bottom border

```

```

        draw_koch_curve(i*step_x, 0, (1+i)*step_x, 0, n, angle, arr, &frame-
>edge_color);

        //upper border
        draw_koch_curve(i*step_x, file->H-1, (1+i)*step_x, file->H-1, n, -
angle, arr, &frame->edge_color);
    }
    for (int j = 0; j < count_y; j++){
        //left border
        draw_koch_curve(0, j*step_y, 0, (1+j)*step_y, n, -angle, arr, &frame-
>edge_color);
        //right border
        draw_koch_curve(file->W-1, j*step_y, file->W-1, (1+j)*step_y, n,
angle, arr, &frame->edge_color);
    }
}

```

Название файла: draw_frame_minc.h

```
#pragma once
```

```
void draw_minkovsky_curve(int x0, int y0, int x1, int y1, int n, Rgb **arr, Rgb
*color);
```

```
void draw_frame_minc(Frame *frame, Rgb **arr, File *file);
```

Название файла: draw_frame_minc.c

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include "structs.h"
```

```
#include "draw_common_func.h"
```

```
#include <math.h>
```

```

void draw_minkovsky_curve(int x0, int y0, int x1, int y1, int n, Rgb **arr, Rgb
*color){
    if (n == 0){
        //draw line
        draw_line(x0, y0, x1, y1, arr, color);
        return;
    }

    //array of line points
    int x[9];
    int y[9];

    x[0] = x0;
    y[0] = y0;
    x[8] = x1;
    y[8] = y1;

    int delta;

    //slope detection
    if (y0 == y1){
        delta = (x1 - x0)/4;

        //getting new points
        x[1] = x[2] = x[0] + delta;
        x[3] = x[4] = x[5] = x[2] + delta;
        x[6] = x[7] = x[3] + delta;

        y[1] = y[4] = y[7] = y[0];
    }
}

```

```

        y[2] = y[3] = y[0] - delta;
        y[5] = y[6] = y[0] + delta;

    } else{
        delta = (y1 - y0)/4;

        //getting new points
        y[1] = y[2] = y[0] + delta;
        y[3] = y[4] = y[5] = y[2] + delta;
        y[6] = y[7] = y[3] + delta;

        x[1] = x[4] = x[7] = x[0];
        x[2] = x[3] = x[0] - delta;
        x[5] = x[6] = x[0] + delta;
    }

    //recursive calls of drawing a curve
    for (int i = 0; i < 8; i++) {
        draw_minkovsky_curve(x[i], y[i], x[i+1], y[i+1], n-1, arr, color);
    }
}

void draw_frame_minc(Frame *frame, Rgb **arr, File *file){
    int n = 2;

    //getting the best parameters
    double step_x = frame->width;
    double step_y = frame->width;
    int count_x = ceil(file->W/step_x);

```

```

int count_y = ceil(file->H/step_y);

while (file->W-1 < count_x*step_x){
    step_x -= 0.000001;
}
while (file->H-1 < count_y*step_y){
    step_y -= 0.000001;
}

//draw frame
for (int i = 0; i < count_x; i++){
    //bottom border
    draw_minkovsky_curve((0+i)*step_x, step_y/2, (1+i)*step_x, step_y/2,
n, arr, &frame->edge_color);
    //upper border
    draw_minkovsky_curve((0+i)*step_x, file->H - step_y/2, (1+i)*step_x,
file->H - step_y/2, n, arr, &frame->edge_color);
}

for (int i = 0; i < count_y; i++){
    //right border
    draw_minkovsky_curve(file->W - step_x/2, (0+i)*step_y, file->W -
step_x/2, (1+i)*step_y, n, arr, &frame->edge_color);
    //left border
    draw_minkovsky_curve(step_x/2, (0+i)*step_y, step_x/2, (1+i)*step_y,
n, arr, &frame->edge_color);
}
}

```

Название файла: draw_frame_serp.h

#pragma once

*void draw_serp(double x0, double y0, double x1, double y1, int n, Rgb **arr, Rgb *color);*

*void draw_frame_serp(Frame *frame, Rgb **arr, File *file);*

Название файла: draw_frame_serp.c

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "structs.h"

#include "draw_common_func.h"

*void draw_serp(double x0, double y0, double x1, double y1, int n, Rgb **arr, Rgb *color){*

if (n > 0){

//new coordinates

*double new_x0 = 2*x0/3 + x1/3;*

*double new_y0 = 2*y0/3 + y1/3;*

*double new_x1 = x0/3 + 2*x1/3;*

*double new_y1 = y0/3 + 2*y1/3;*

//draw rectangle

draw_rectangle((int)new_x0, (int)new_y0, (int)new_x1, (int)new_y1, arr, color);

//recursion

draw_serp(x0, y0, new_x0, new_y0, n-1, arr, color);

draw_serp(new_x0, y0, new_x1, new_y0, n-1, arr, color);


```

        draw_serp(new_x1, y0, x1, new_y0, n-1, arr, color);
        draw_serp(x0, new_y0, new_x0, new_y1, n-1, arr, color);
        draw_serp(new_x1, new_y0, x1, new_y1, n-1, arr, color);
        draw_serp(x0, new_y1, new_x0, y1, n-1, arr, color);
        draw_serp(new_x0, new_y1, new_x1, y1, n-1, arr, color);
        draw_serp(new_x1, new_y1, x1, y1, n-1, arr, color);
    }
}

```

```

void draw_frame_serp(Frame *frame, Rgb **arr, File *file){
    int n = 4;

    //getting the best parameters
    double step_x = frame->width;
    double step_y = frame->height;
    int count_x = ceil(file->W/step_x);
    int count_y = ceil(file->H/step_y);

    while (file->W-1 < count_x*step_x){
        step_x -= 0.0001;
    }
    while (file->H-1 < count_y*step_y){
        step_y -= 0.0001;
    }

    //draw frame
    for (int i = 0; i < file->W/frame->width; i++){
        //bottom border

```

```

        draw_serp(0+i*step_x, step_y, (i+1)*step_x, 0, n, arr, &frame-
>edge_color);

        //upper border
        draw_serp(file->W - (i+1)*step_x, file->H, file->W - i*step_x, file->H -
step_y, n, arr, &frame->edge_color);
    }
    for (int i = 0; i < file->H/frame->width; i++){
        //left border
        draw_serp(0, file->H - i*step_y, step_x, file->H - (i+1)*step_y, n, arr,
&frame->edge_color);
        //right border
        draw_serp(file->W - step_x, (i+1)*step_y, file->W, i*step_y, n, arr,
&frame->edge_color);
    }
}

```

Название файла: draw_hexagon.h

```
#pragma once
```

```

void hexagon_flood_fill(int x, int y, Rgb **arr, Rgb *new_color, Rgb
*boarder_color);
void get_hexpoints(int x, int y, int r, Rgb **arr, Point *points_arr);
int transformation(Hexagon *hexagon);
int DrawHexagon(Hexagon *hexagon, Rgb **arr, File *file);

```

Название файла: draw_hexagon.c

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "getopt.h"

```

```
#include "structs.h"
```

```
#include "draw_common_func.h"
```

```
void hexagon_flood_fill(int x, int y, Rgb **arr, Rgb *new_color, Rgb  
*boarder_color){  
    set_pixel(&arr[y][x], new_color);  
  
    //checking for boarder color or already filled  
    if (!is_check_color(&arr[y][x+1], boarder_color) &&  
!is_check_color(&arr[y][x+1], new_color)){  
        hexagon_flood_fill(x+1, y, arr, new_color, boarder_color);  
    }  
    if (!is_check_color(&arr[y][x-1], boarder_color) &&  
!is_check_color(&arr[y][x-1], new_color)){  
        hexagon_flood_fill(x-1, y, arr, new_color, boarder_color);  
    }  
    if (!is_check_color(&arr[y+1][x], boarder_color) &&  
!is_check_color(&arr[y+1][x], new_color)){  
        hexagon_flood_fill(x, y+1, arr, new_color, boarder_color);  
    }  
    if (!is_check_color(&arr[y-1][x], boarder_color) && !is_check_color(&arr[y-  
1][x], new_color)){  
        hexagon_flood_fill(x, y-1, arr, new_color, boarder_color);  
    }  
}
```

```
void get_hexpoints(int x, int y, int r, Rgb **arr, Point *points_arr){  
    double angle = 0; //start angle
```

```

    unsigned int n = 6;          //count pick
    for (int i = 0; i < n+1; i++){
        points_arr[i].x = x + r * cos(angle*M_PI/180);
        points_arr[i].y = y + r * sin(angle*M_PI/180);
        angle += 360.0/n;
    }
}

int is_correct_hex_coordinates(Hexagon *hexagon, File *file){
    int H = (hexagon->radius + hexagon->line_width) * sqrt(3) / 2;
    if (hexagon->center.x < 0 // hexagon->center.x > file->W-1 // hexagon-
>center.y < 0 // hexagon->center.y > file->H-1){
        printf("\x1b[31mCenter is not in a picture.\x1b[0m\n");
        return 0;
    }

    if (hexagon->center.x + hexagon->radius + hexagon->line_width > file->W-1
// hexagon->center.x - hexagon->radius - hexagon->line_width < 0){
        printf("\x1b[31mHexagon is not fit in width.\x1b[0m\n");
        return 0;
    }

    if (hexagon->center.y + H > file->H-1 // hexagon->center.y - H < 0){
        printf("\x1b[31mHexagon is not fit in height.\x1b[0m\n");
        return 0;
    }

    return 1;
}

```

```

int transformation(Hexagon *hexagon){
    //getting the sides of a rectangle
    int a = (hexagon->end.x - hexagon->start.x);
    int b = (hexagon->start.y - hexagon->end.y);

    //check if it's a square
    if (a != b){
        printf("\x1b[31mThese are not square coordinates\x1b[0m\n");
        return -1;
    }

    //conversion to base data
    hexagon->center.x = hexagon->end.x - a/2;
    hexagon->center.y = hexagon->start.y - b/2;
    hexagon->radius = (int)(hexagon->end.x - hexagon->center.x)-hexagon-
>line_width;
}

```

```

int DrawHexagon(Hexagon *hexagon, Rgb **arr, File *file){
    //checking if another flags are passed
    if (hexagon->flags[3] && hexagon->flags[4]){
        if (transformation(hexagon) == -1){
            return -1;
        }
        hexagon->flags[0] = 1;
        hexagon->flags[1] = 1;
        hexagon->flags[3] = 1;
    }
}

```

```

        hexagon->flags[4] = 1;
    }

    //checking if all flags are passed
    if (!hexagon->flags[0] || !hexagon->flags[1]){
        printf("\x1b[31mNot all necessary flags for draw hexagon were
passed\x1b[0m\n");
        return -1;
    }

    //checking correct coordinates
    if (!is_correct_hex_coordinates(hexagon, file)){
        return -1;
    }

    //checking correct radius
    if (hexagon->radius < 2){
        printf("\x1b[31mRadius too small.\x1b[0m\n");
        return -1;
    }

    //create an array of vertices
    unsigned int count_vertices = 6;
    Point *points_arr = malloc((count_vertices+1)*sizeof(Point));
    if (points_arr == NULL){
        printf("\x1b[31mIt is impossible to allocate memory for an array of
vertices.\x1b[0m\n");
        return -1;
    }

```

```

        //draw hexagon boarder
        for (int j = hexagon->radius; j <= hexagon->radius + hexagon->line_width;
j++){

            get_hexpoints(hexagon->center.x, hexagon->center.y, j, arr,
points_arr);

            for (int i = 0; i < count_vertices; i++){
                draw_line(points_arr[i].x, points_arr[i].y, points_arr[i+1].x,
points_arr[i+1].y, arr, &hexagon->line_color);
            }
        }

        //free array of vertices
        free(points_arr);

        //fill hexagon
        if (hexagon->flags[2]){
            hexagon_flood_fill(hexagon->center.x, hexagon->center.y, arr,
&hexagon->filling_color, &hexagon->line_color);
            hexagon_flood_fill(hexagon->center.x + (hexagon->radius-2), hexagon-
>center.y, arr, &hexagon->filling_color, &hexagon->line_color);
            hexagon_flood_fill(hexagon->center.x - (hexagon->radius-2), hexagon-
>center.y, arr, &hexagon->filling_color, &hexagon->line_color);
            hexagon_flood_fill(hexagon->center.x, hexagon->center.y + (hexagon-
>radius/2), arr, &hexagon->filling_color, &hexagon->line_color);
            hexagon_flood_fill(hexagon->center.x, hexagon->center.y - (hexagon-
>radius/2), arr, &hexagon->filling_color, &hexagon->line_color);
        }

        return 0;
    }

```

Название файла: switch_color.h

#pragma once

*int SwitchColorFunc(Rgb **arr, SwitchColor *switch_color, File *file);*

Название файла: switch_color.c

#include <stdio.h>

#include "structs.h"

#include "draw_common_func.h"

#include "getopt.h"

*int SwitchColorFunc(Rgb **arr, SwitchColor *switch_color, File *file){*

int y = 200;

int x = 50;

printf("r = %d g = %d b = %d\n", arr[y][x].r, arr[y][x].g, arr[y][x].b);

printf("r = %d g = %d b = %d\n", arr[y+300][x].r, arr[y+300][x].g,

arr[y+300][x].b);

//checking if all flags are passed

if (!switch_color->flags[0] || !switch_color->flags[1]){

*printf("\x1b[31mNot all necessary flags for switch color were
passed\x1b[0m\n");*

return -1;

}

for(int i=0; i<file->H; i++){

for(int j=0; j<file->W; j++){

if (is_check_color(&arr[i][j], &switch_color->old_color)){


```

        set_pixel(&arr[i][j], &switch_color->new_color);
    }
}

return 0;
}

```

Название файла: main.c

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include "structs.h"
#include "getopt.h"
#include "getopt_files.h"
#include "bmp_info.h"
#include "getopt_copy.h"
#include "getopt_hexagon.h"
#include "getopt_switch_color.h"
#include "getopt_frame.h"
#include "copy_area.h"
#include "switch_color.h"
#include "draw_hexagon.h"
#include "draw_frame.h"
#include "file_processing.h"
#define OUTPUT_FILE_NAME "out.bmp"

```

```

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

    //start init structs

    Configs config = {0, 0, 0, 0, calloc(50, sizeof(char)), calloc(50, sizeof(char))};
    strcpy(config.output_file, OUTPUT_FILE_NAME);
    Hexagon hexagon = {{0, 0, 0, 0, 0}, {0, 0}, {0, 0}, {0, 0}, 0, 1, {0, 0, 0}};
    SwitchColor switch_color = {{0, 0}, {0, 0, 0}, {0, 0, 0}};
    Area area = {{0, 0, 0}, {0, 0}, {0, 0}, 0, 0, {0, 0}};
    Frame frame = {0, 20, {255, 255, 255}, {0, 0, 0}};

    //_____GETOPT_____

    char *opts = "h?i:o:xf:cse";
    struct option LongOpts[]={
        {"help", no_argument, NULL, 'h'},
            {"bmpversion", no_argument, NULL, 0},
        {"bmpinfo", no_argument, NULL, 0},
        {"inputfile", required_argument, NULL, 'i'},
        {"outfile", required_argument, NULL, 'o'},
            {"hexagon", no_argument, NULL, 'x'},
            {"hexstart", required_argument, NULL, 14},
            {"hexend", required_argument, NULL, 15},
            {"center", required_argument, NULL, 1},
            {"radius", required_argument, NULL, 2},
            {"linewidth", required_argument, NULL, 3},
            {"linergb", required_argument, NULL, 4},
        {"fill", required_argument, NULL, 'f'},
        {"copy", no_argument, NULL, 'c'},
            {"start", required_argument, NULL, 5},
    }
}

```

```

        {"end", required_argument, NULL, 6},
        {"insert", required_argument, NULL, 7},
        {"switchcolor", no_argument, NULL, 's'},
        {"old", required_argument, NULL, 8},
        {"new", required_argument, NULL, 9},
        {"edge", no_argument, NULL, 'e'},
        {"type", required_argument, NULL, 10},
        {"width", required_argument, NULL, 11},
        {"edgergb", required_argument, NULL, 12},
        {"bgrgb", required_argument, NULL, 13},
        { NULL, no_argument, NULL, 0}
    };

    int opt;
    int longIndex;
    opt = getopt_long(argc, argv, opts, LongOpts, &longIndex);

    //checking the launch of the program without flags
    if (opt == -1){
        PrintHelp();
        return 0;
    }

    while(opt != -1){
        switch(opt){
            //_____INPUT_FI
            LENAME_____
                case 'i':
                    if (case_input_file(&config, optarg) == -1){
                        return -1;

```

```
    }  
    break;
```

```
//_____OUTPUT_  
FILENAME_____
```

```
    case 'o':  
        if (case_output_file(&config, optarg) == -1){  
            return -1;  
        }  
        break;
```

```
//_____HEXAGON
```

```
    case 'x':  
        //hexagon  
        config.draw_hexagon = 1;  
        break;
```

```
    case 14:  
        //hexstart  
        hexagon.flags[3] = 1;  
        if (case_hexstart(argc, argv, &optind, &config, &hexagon)  
== -1){  
            return -1;  
        }  
        break;
```

```
    case 15:  
        //hexend  
        hexagon.flags[4] = 1;
```

```

        if (case_hexend(argc, argv, &optind, &config, &hexagon)
== -1){

            return -1;

        }

        break;

    case 1:

        //hexcenter
        hexagon.flags[0] = 1;
        if (case_hexcenter(argc, argv, &optind, &config,
&hexagon) == -1){

            return -1;

        }

        break;

    case 2:

        //hexradius
        hexagon.flags[1] = 1;
        if (case_hexradius(argc, argv, optarg, &config, &hexagon)
== -1){

            return -1;

        }

        break;

    case 3:

        //hexline_width
        if (case_hexline_width(argc, argv, optarg, &config,
&hexagon) == -1){

            return -1;

        }

```

```

        break;

    case 4:
        //hexline_color
        if (case_hexline_color(argc, argv, &optind, &config,
&hexagon) == -1){
            return -1;
        }
        break;

    case 'f':
        //hexfilling
        hexagon.flags[2] = 1;
        if (case_hexfilling(argc, argv, &optind, &config,
&hexagon) == -1){
            return -1;
        }
        break;

//_____COPY_AR
EA_____

    case 'c':
        //copy
        config.copy_area = 1;
        break;

    case 5:
        //start copy
        area.flags[0] = 1;

```

```

        if (case_start_copy(argc, argv, &optind, &config, &area)
== -1){

            return -1;

        }

        break;

case 6:

    //end copy
    area.flags[1] = 1;
    if (case_end_copy(argc, argv, &optind, &config, &area)
== -1){

        return -1;

    }

    break;

case 7:

    //insert copy
    area.flags[2] = 1;
    if (case_insert(argc, argv, &optind, &config, &area) == -
1){

        return -1;

    }

    break;

//_____SWITCH_C
OLOR_____

case 's':

    //switch color
    config.switch_color = 1;
    break;

```

```

    case 8:
        //old color
        switch_color.flags[0] = 1;
        if (case_old_color(argc, argv, &optind, &config,
&switch_color) == -1){
            return -1;
        }
        break;

    case 9:
        //new color
        switch_color.flags[1] = 1;
        if (case_new_color(argc, argv, &optind, &config,
&switch_color) == -1){
            return -1;
        }
        break;

//_____FRAME_____

```

```

    case 'e':
        //frame
        config.draw_frame = 1;
        break;

    case 10:
        //edge type
        if (case_edge_type(argc, argv, optarg, &config, &frame)
== -1){

```



```

        return -1;
    }
    break;

case 11:
    //edge width
    if (case_edge_width(argc, argv, optarg, &config, &frame)
== -1){
        return -1;
    }
    break;

case 12:
    //edge color
    if (case_edge_color(argc, argv, &optind, &config, &frame)
== -1){
        return -1;
    }
    break;

case 13:
    //background color
    if (case_bg_color(argc, argv, &optind, &config, &frame)
== -1){
        return -1;
    }
    break;

// _____INFO_____
_____

```

```

        case 0:
            if (!strcmp(LongOpts[longIndex].name, "bmpversion")){
                printf("This program supports only 24 bits per pixel
version of bmp file!\n");
            } else{
                config.bmp_info = 1;
            }
            break;

//_____HELP_____

        case 'h':
        case '?':
            default:
                PrintHelp();
                return 0;
        }

        opt = getopt_long(argc, argv, opts, LongOpts, &longIndex);
    }

    //checking multable file
    if (!search_input_file(argc, argv, &optind, &config)){
        no_input_file();
        return -1;
    }

```

```
// _____FILE_PRO  
CESSING_____
```

```
    //init file structs
```

```
    File file = {malloc(sizeof(BitmapFileHeader)),  
malloc(sizeof(BitmapInfoHeader)), 0, 0, NULL};
```

```
    BitmapFileHeader bmfh;
```

```
    BitmapInfoHeader bmih;
```

```
    //reading file
```

```
    if (read_input_file(&config, &file, &bmfh, &bmih) == -1){  
        return -1;  
    }
```

```
    //headlines recording
```

```
    file.bmfh = &bmfh;
```

```
    file.bmih = &bmih;
```

```
    //checking bmpinfo flag
```

```
    if (config.bmp_info){  
        printFileHeader(*file.bmfh);  
        printInfoHeader(*file.bmih);  
        printf("\n");  
        //return 0;  
    }
```

```
    //checking bmp version
```

```
    if (file.bmih->bitsPerPixel != 24){  
        printf("\x1b[31mThis file %hu bits per pixel\x1b[0m\n", file.bmih-  
>bitsPerPixel);
```

```

        printf("\x1b[31mBut this program supports only 24 bits per pixel version
of bmp file!\x1b[31m\n");
        return -1;
    }

```

```

//_____TASK_CO
MPETITION_____

```

```

    //tasks
    if (config.copy_area){
        //copy area
        if (CopyAreaFunc(file.arr, &area, &file) == -1){
            return -1;
        }
    }

    if (config.switch_color){
        //switch color
        if (SwitchColorFunc(file.arr, &switch_color, &file) == -1){
            return -1;
        }
    }

    if (config.draw_hexagon){
        //draw hexagon
        if (DrawHexagon(&hexagon, file.arr, &file) == -1){
            return -1;
        }
    }
}

```

```

        if (config.draw_frame){
            //draw frame
            if (DrawFrame(&frame, file.arr, &file) == -1){
                return -1;
            }
        }

//_____TOTAL_____

        //writing the resulting file
        if (create_output_file(&config, &file) == -1){
            return -1;
        }

        return 0;
}

```

Название файла: Makefile

CC = gcc

```

all: main.o getopt.o getopt_files.o bmp_info.o getopt_copy.o getopt_hexagon.o
getopt_frame.o getopt_switch_color.o copy_area.o switch_color.o file_processing.o
draw_hexagon.o draw_common_func.o draw_frame_serp.o draw_frame_koch.o
draw_frame_minc.o draw_frame.o

$(CC) -g -Wall main.o getopt.o getopt_files.o bmp_info.o getopt_copy.o
getopt_hexagon.o getopt_frame.o getopt_switch_color.o copy_area.o switch_color.o
file_processing.o draw_hexagon.o draw_common_func.o draw_frame_serp.o
draw_frame_koch.o draw_frame_minc.o draw_frame.o -o main -lm

```

*main.o: main.c structs.h getopt.h getopt_files.h bmp_info.h getopt_copy.h
getopt_hexagon.h getopt_frame.h getopt_switch_color.h copy_area.h switch_color.h
\$(CC) -c main.c*

*getopt_hexagon.o: getopt_hexagon.c getopt.h structs.h
\$(CC) -c getopt_hexagon.c*

*getopt_switch_color.o: getopt_switch_color.c getopt.h structs.h
\$(CC) -c getopt_switch_color.c*

*geopt_copy.o: getopt_copy.c getopt.h structs.h
\$(CC) -c getopt_copy.c*

*getopt_frame.o: getopt_frame.c getopt.h structs.h
\$(CC) -c getopt_frame.c*

*getopt_files.o: getopt_files.c getopt.h structs.h
\$(CC) -c getopt_files.c*

*bmp_info.o: bmp_info.c structs.h
\$(CC) -c bmp_info.c*

*file_processing.o: file_processing.c structs.h
\$(CC) -c file_processing.c*

*copy_area.o: copy_area.c structs.h getopt.h
\$(CC) -c copy_area.c*

*switch_color.o: switch_color.c draw_common_func.h structs.h
\$(CC) -c switch_color.c*

draw_hexagon.o: draw_hexagon.c draw_common_func.h structs.h getopt.h
\$(CC) -c draw_hexagon.c

draw_frame.o: draw_frame.c draw_common_func.h draw_frame_serp.h structs.h
getopt.h
\$(CC) -c draw_frame.c

draw_frame_serp.o: draw_frame_serp.c draw_common_func.h structs.h
\$(CC) -c draw_frame_serp.c

draw_frame_koch.o: draw_frame_koch.c draw_common_func.h structs.h
\$(CC) -c draw_frame_koch.c

draw_frame_minc.o: draw_frame_minc.c structs.h
\$(CC) -c draw_frame_minc.c

draw_common_func.o: draw_common_func.c structs.h
\$(CC) -c draw_common_func.c

getopt.o: getopt.c structs.h
\$(CC) -c getopt.c

clean:
*rm -r *.o*