**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка bmp изображений на языке C

|  |  |  |
| --- | --- | --- |
| Студент гр. 3388 |  | Пьянков М.Ф. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

**2024**

# ЗАДАНИЕ

# НА КУРСОВУЮ РАБОТУ

Студент Пьянков М.Ф.

Группа 3388

Тема работы: Обработка bmp изображений на языке C

Исходные данные:

Вариант 6

Программа обязательно должна иметь CLI (опционально дополнительное использование GUI). Более подробно тут: <http://se.moevm.info/doku.php/courses:programming:rules_extra_kurs> Программа должна реализовывать весь следующий функционал по обработке bmp-файла

Общие сведения

24 бита на цвет, без сжатия, файл может не соответствовать формату BMP, т.е. необходимо проверка на

BMP формат (дополнительно стоит помнить, что версий у формата несколько).

Если файл не соответствует формату BMP или его версии, то программа должна завершиться с соответствующей ошибкой. Обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями. обратите внимание на порядок записи пикселей. Все поля стандартных BMP заголовков в выходном файле должны иметь те же значения что и во входном (разумеется кроме тех, которые должны быть изменены).

Программа должна иметь следующую функции по обработке изображений:

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

Отрезок определяется: координатами начала. Флаг `--start`, значение задаётся в формате `x.y`, где x – координата по x, y – координата по y координатами конца. Флаг `--end` (аналогично флагу `--start`) цветом. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет) ○ толщиной. Флаг `--thickness`. На вход принимает число больше 0

(2) Инвертировать цвета в заданной окружности. Флаг для выполнения данной операции: `--inverse\_circle`. Окружность определяется координатами ее центра и радиусом. Флаги `--center` и `--radius`. Значение флаг `--center` задаётся в формате `x.y`, где x – координата по оси x, y – координата по оси y. Флаг `--radius` На вход принимает число больше 0

(3) Обрезка изображения. Флаг для выполнения данной операции: `--trim`. Требуется обрезать изображение по заданной области. Область определяется: Координатами левого верхнего угла. Флаг `--left\_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y. Координатами правого нижнего угла. Флаг `--right\_down`, значение задаётся в формате `right.down`, где right – координата по x, down – координата по y

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

Дата выдачи задания: 6.04.2024

Дата сдачи реферата:20.05.2024

Дата защиты реферата: 22.05.2024

|  |  |  |
| --- | --- | --- |
| Студент гр. 3388 |  | Пьянков М. Ф. |
| Преподаватель |  | Заславский М. М. |

# АННОТАЦИЯ

Курсовая работа посвящена разработке программного обеспечения для обработки изображений на языке программирования Си. Программа предоставляет пользователю возможность рисовать прямые линии заданной толщины, инвертировать цвета в пределах заданной окружности и обрезать изображение до любой прямоугольной формы. Для удобства использования программа оснащена графическим интерфейсом командной строки (CLI), который позволяет пользователю задавать необходимые параметры для выполнения операций. В процессе разработки были использованы библиотеки getopt.h, стандартная библиотека Си и регулярные выражения для обработки команд.

# SUMMARY

The course work is devoted to the development of software for image processing in the C programming language. The program provides the user with the ability to draw straight lines of a given thickness, invert colors within a given circle, and crop the image to any rectangular shape. For ease of use, the program is equipped with a graphical command line interface (CLI), which allows the user to set the necessary parameters to perform operations. During the development process, the getopt.h libraries, the standard C library and regular expressions were used to process commands.

СОДЕРЖАНИЕ

[Задание на курсовую работу 2](#__RefHeading___Toc135_2330072175)-3

Аннотация5

Введение [8](#__RefHeading___Toc3559_2330072175)

Ход выполнение работы9-12

Считывание изображения9

Функция Rgb\*\* read\_bmp(char\* file\_name, BitmapFileHeader\* bmfh, BitmapInfoHeader\* bmif)9

CLI9-11

Функция int main(int argc, char\*\* argv) [1](#__RefHeading___Toc3559_2330072175)0

Структура longopts 10

Функция getopt\_long() 10

Функция switch(), обработка флагов 10

Функция int extend\_command\_message(char\* command) [11](#__RefHeading___Toc3559_2330072175)

Функция int check\_args(char\* optarg, const char\* exp, const char\* ext\_exp) [11](#__RefHeading___Toc3559_2330072175)

Обработка изображения11-12

Функция set\_pixel(Rgb\*\* arr, int x, int y, int\* color, int H, int W) [1](#__RefHeading___Toc3559_2330072175)1

Функция void Circle(Rgb\*\* arr, int xc, int yc, int\* color, int rad, int H, int W, int x0, int y0, int x1, int y1) [1](#__RefHeading___Toc3559_2330072175)1

Функция void draw\_line(Rgb\*\* arr, int H, int W, int x0, int y0, int x1, int y1, int\* color, double thickness)11

Функция void invert\_circle(Rgb\*\* arr, int H, int W, int x0, int y0, int r) [11](#__RefHeading___Toc3559_2330072175)

Функция Rgb\*\* trim(Rgb\*\* arr, BitmapInfoHeader\* bmif, BitmapFileHeader\* bmfh, int x0, int y0, int x1, int y1) [1](#__RefHeading___Toc3559_2330072175)2

Функция void write\_bmp(char\* file\_name, Rgb\*\* arr, int H, int W, BitmapFileHeader bmfh, BitmapInfoHeader bmif) [1](#__RefHeading___Toc3559_2330072175)2

Заключение13

Список использованной литературы14

Приложение А с примерами работы программы15-17

Приложение Б. Исходный код программы18-30

# ВВЕДЕНИЕ

Цель работы:

Написать программу, обрабатывающую изображение формата bmp[3], согласно командам пользователя.

Подробная характеристика запросов пользователя изложена в разделе ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ (стр. 2)

Основные теоретические положения:

Создана структура Rgb представляющая цвет в формате RGB, поля структуры: r, g, b.

Для хранения данных об изображении создаётся двумерный динамический массив структур Rgb типа Rgb\*\*.

Библиотеки: stdio.h, stdlib.h, string.h, regex.h, getopt.h.

Функции: Обработка изображения: malloc(), fwrite(), fread().

Вывод сообщений, ошибок: exit(), puts(), strcat(), strcpy(), malloc(). Обработка команд: switch(), getopt\_long(), sscanf(), regcomp(), regexec(). Подробнее см. в ПРИЛОЖЕНИЕ Б

# ХОД ВЫПОЛНЕНИЯ РАБОТЫ

## Считывание изображения

Чтение изображения начинается со считывания заголовочного файла и информационного заголовочного файла (BitmapFileHeader, BitmapInfoHeader). Для хранения данных об этом блоке были созданы соответствующие структуры с необходимыми полями.

Далее производиться считывание таблицы цветов непосредственно в двумерный массив структур Rgb. Следует обратить внимание на такое понятие как выравнивание: объём памяти занимаемый каждой строкой таблицы цветов в bmp файле кратен 4, поэтому если необходимо, то записываются мусорные данные: arr[H - i - 1] = malloc(W \* sizeof(Rgb) + (4 - ((W \* sizeof(Rgb)) %4) ) % 4). Для подробностей см. ПРИЛОЖЕНИЕ Б.

### Функция Rgb\*\* read\_bmp(char\* file\_name, BitmapFileHeader\* bmfh, BitmapInfoHeader\* bmif)

Функция начинается с создания файлового указателя FILE \*f. Далее считываются заголовочные блоки функцией fread() и таблица цветов записывается в двумерный динамический массив. Запись происходит таким образом, чтобы было удобнее работать с командами в рамках курсовой работы (начало координат в командах - левый верхний угол, условное начало координат в стандартном считывании — левый нижний угол) поэтому строка считывается с таким индексом: H - i — 1. Работа функции разобрана подробнее в методичке к курсовой работе[1].

## CLI

Обработка команд производится в функции main(). Реализована она через функцию getopt() и switch(), использовалась бибилиотека getopt.h[2]. Команды, принимающие аргументы проверяются соответствующими функциями, переопределяются значения флагов для каждой команды. После функции switch() происходит обработка флагов и запуск необходимых функций обработки изображения. см. ПРИЛОЖЕНИЕ Б.

### Структура static struct option longopts[]

В этой структуре записаны команды в формате, необходимом для работы функции getopt\_long().

### Функция int getopt\_long(argc, argv, opts, longopts, &longidx)

Функция обрабатывает массив char\*\* argv, подаваемые на вход программы через терминал.

### Функция switch (opt)

Функция проверяет отдельную команду (opt).

Команда -h (--help): выводится сообщение определённое как HELP\_MESSAGE, программа успешно завершается с кодом выхода 0. См. Рисунок 2.

Команда -f (--info): переопределяется флаг info\_flag = 1. См. ПРИЛОЖЕНИЕ А: Рисунок 3.

Команда -i (--input): переопределяется флаг file\_name\_flag = 1, задаётся имя входного файла.

Команда -o(--output): переопределяется имя выходного файла (изначально оно определено как NEW\_FILE\_NAME.

Команды -l(--line), -I(--inverse\_circle), -T(--trim): переопределяются соответствующие флаги line\_flag, inverse\_circle\_flag, trim\_flag на значение 1. См. ПРИЛОЖЕНИЕ А: Рисунок 4, Рисунок 5, Рисунок 6 и Рисунок 10, Рисунок 11, Рисунок 12 соответственно.

Команды -s(--start), -e(--end), -L(--left\_up), -R(--right\_down): проверка аргументов команды функцией check\_coords\_arg(optarg), функция sscanf() считывает значения в соответствующие переменные.

Команда -c(--color): проверка аргументов функцией check\_color\_arg(optarg) и считывание функцией sscanf().

Команды -t(--thickness), -r(--radius): проверка аргумента функцией check\_one\_arg(optarg) и считывание функцией sscanf().

Для каждой командой предусматривающей аргументы идёт их обработка, и вывод соответствующей информации об ошибке (результат проверки функцией check\_args() - 2) или завершении программы с кодом ошибки ERROR\_CODE (результат проверки функцией check\_args() - 0).

### Функция void extend\_command\_message(char\* command)

Функция создаёт сообщение об ошибки, когда пользователь ошибся и ввёл слишком много аргументов для команды.

### Функция int check\_args(char\* optarg, const char\* exp, const char\* ext\_exp)

Функция проверяет аргументы команды с помощью регулярных выражений: функции regcomp(), regexec(). Если аргумент полностью правильный, то функция возвращает 1, если аргументов слишком много — то 2, в остальных случаях считается, что пользователь ввёл неверную команду, возвращается 0. Обработка ошибок пользователя: см. ПРИЛОЖЕНИЕ А: Рисунок 7, Рисунок 8, Рисунок 9.

## Обработка изображения

### Функция void set\_pixel(Rgb\*\* arr, int x, int y, int\* color, int H, int W)

Проверяет вхождение ячейки в таблицу цветов и меняет её цвет на заданный.

### Функция void Circle(Rgb\*\* arr, int xc, int yc, int\* color, int rad, int H, int W, int x0, int y0, int x1, int y1)

Функция использует алгоритм Брезенхема для рисования окружностей.

### Функция void draw\_line(Rgb\*\* arr, int H, int W, int x0, int y0, int x1, int y1, int\* color, double thickness)

Функция использует алгоритм Брезенхема для рисования линии. Каждый элемент линии представлен окружностей, рисуемой функцией Circle().

### Функция void invert\_circle(Rgb\*\* arr, int H, int W, int x0, int y0, int r)

Функция используя двойной цикл и уравнение круга меняет цвет подходящих пикселей (инвертирует) с помощью функции set\_pixel().

### Функция Rgb\*\* trim(Rgb\*\* arr, BitmapInfoHeader\* bmif, BitmapFileHeader\* bmfh, int x0, int y0, int x1, int y1)

Функция используя двойной цикл и уравение прямоугольника (int rectangle\_func(int x, int y, int x0, int y0, int x1, int y1), заполняет новый динамический массив Rgb new\_arr, тем самым выполняя обрезку изображения. Функция также переопределяет поля заголовочных блоков в соответствии с заданными параметрами.

### void write\_bmp(char\* file\_name, Rgb\*\* arr, int H, int W, BitmapFileHeader bmfh, BitmapInfoHeader bmif)

Функция аналогично функции read\_bmp(), записывает данные в файл, также создавая файловый указатель и применяя выравнивание. Разбор функции представлен в методичке к курсовой[1].

Для подробностей см. ПРИЛОЖЕНИЕ Б.

# Заключение

В результате написания курсовой работы было создано консольное приложение обрабатывающее bmp изображения согласно инструкциям пользователя. Интерфейс представлен интерфейсом командной строки (CLI) Реализован следующий функционал: рисование линии заданной толщиной, инвертирования цветов в заданной окружности и обрезка изображения до любой прямоугольной формы.

Все задачи были успешно выполнены. Предусмотрены случаи ошибочного ввода пользователем и вывод сообщения об ошибке или завершение программы с соответствующим кодом.

# Список используемой литературы

[БАЗОВЫЕ СВЕДЕНИЯ К ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ». ВТОРОЙ СЕМЕСТР](https://se.moevm.info/lib/exe/fetch.php/courses:programming:programming_cw_metoda_2nd_course_last_ver.pdf.pdf) (Методичка)[1]

[getopt](https://www.gnu.org/software/libc/manual/html_node/Getopt.html)[2]

[BMP формат](https://docs.fileformat.com/ru/image/bmp/)[3]

# ПРИЛОЖЕНИЕ А. с примерами работы программы

Пользователь не ввёл команд

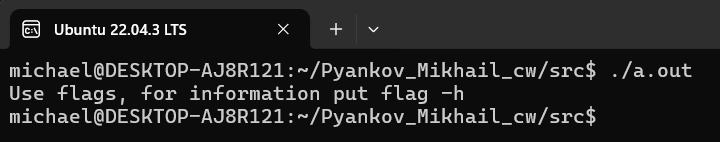


Рисунок 1 — Запуск без команд

Флаг -h

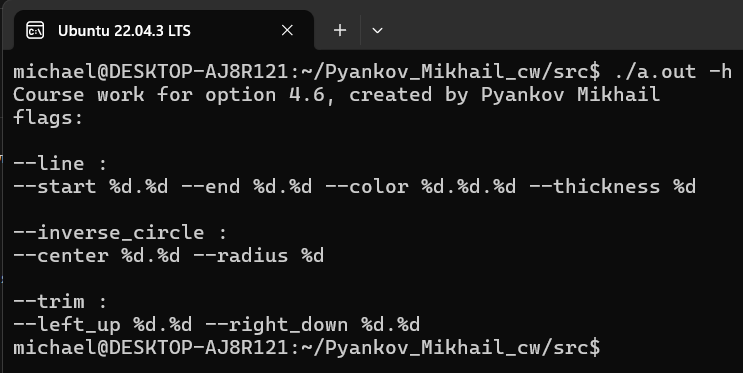


Рисунок 2 — Запуск с командой -h

Флаг —info

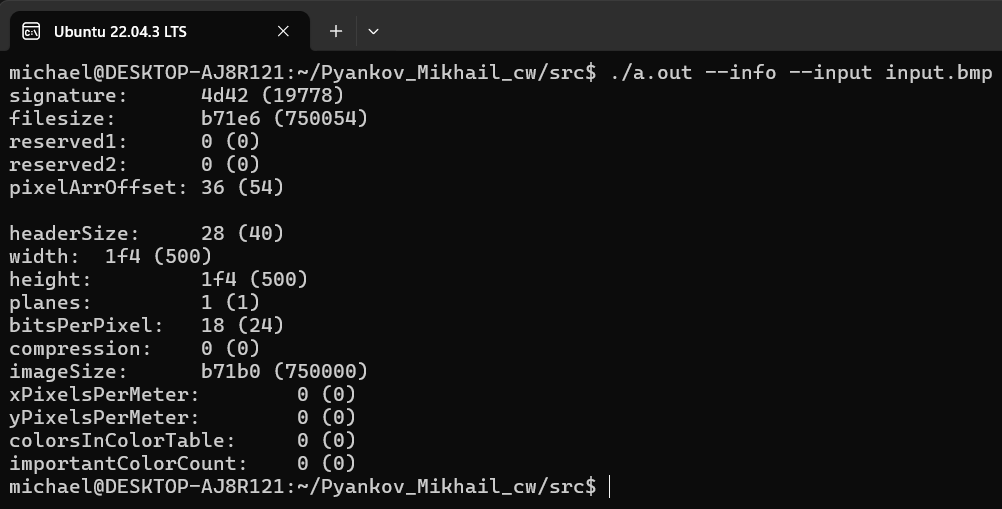


Рисунок 3 — Запуск с командой --info

Флаг —line

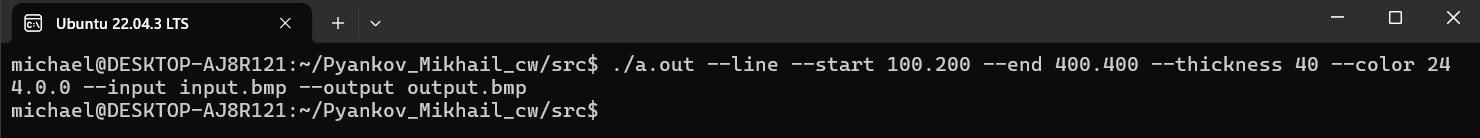
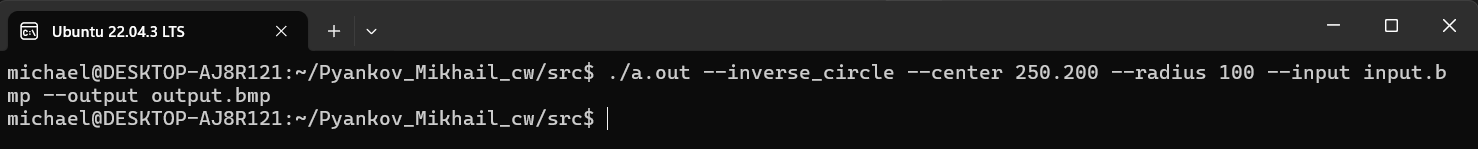
Рисунок 10 — Команда рисования линии



Рисунок 4 — Запуск с командой --line

Флаг —inverse\_circle

Рисунок 11 — Команда инвертирования цветов в окружности

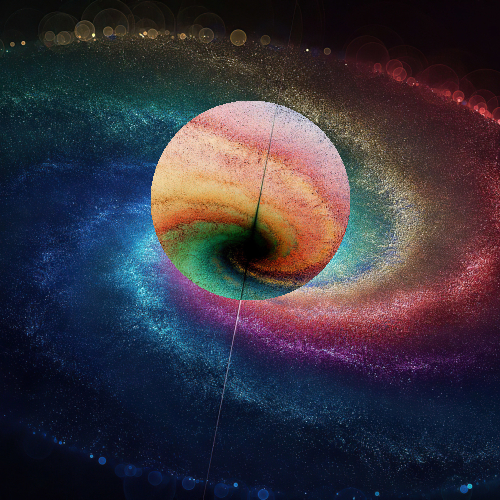


Рисунок 5 — Запуск с командой --inverse\_circle

Флаг --trim

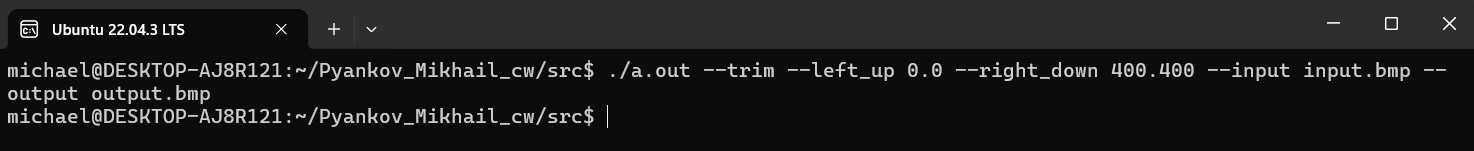
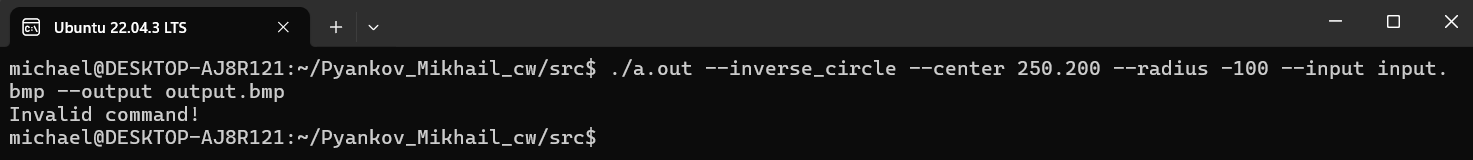
Рисунок 12 — Команда обрезки изображения

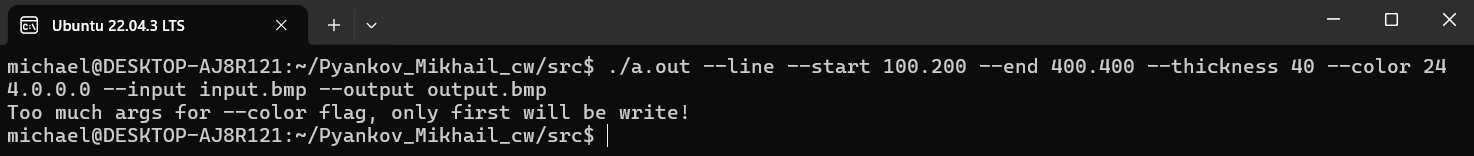


Рисунок 6 — Запуск с командой –trim

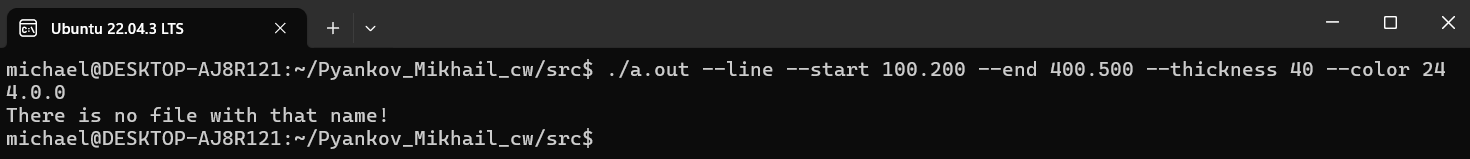
Ошибка в команде

Рисунок 7 — Ошибка в команде

Слишком много аргументов

Рисунок 8 — Запуск с флагом с чрезмерным количеством параметров

Не указан входной файл

Рисунок 9 — Запуск программы без входного файла

# ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main\_cw.c

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<getopt.h>

#include<regex.h>

#include<math.h>

#define no\_argument 0

#define required\_argument 1

#define SIZE 10

#define COMMAND\_SIZE 50

#define GROUPS 1

#define NEW\_FILE\_NAME "./out.bmp"

#define HELP\_MESSAGE "Course work for option 4.6, created by Pyankov Mikhail\nflags:\n\n--line :\n--start %d.%d --end %d.%d --color %d.%d.%d --thickness %d\n\n--inverse\_circle :\n--center %d.%d --radius %d\n\n--trim :\n--left\_up %d.%d --right\_down %d.%d"

#define DEFAULT\_MESSAGE "Use flags, for information put flag -h"

#define REG\_COORDS "^-?[0-9]+\\.-?[0-9]+$"

#define REG\_COORDS\_EXTENDED "^-?[0-9]+\\.-?[0-9]+(\\.-?[0-9]+)+$"

#define REG\_LINE\_COLOR "^[0-9]+\\.[0-9]+\\.[0-9]+$"

#define REG\_LINE\_COLOR\_EXTENDED "^[0-9]+\\.[0-9]+\\.[0-9]+(\\.-?[0-9]+)+$"

#define REG\_ONE\_ARG "^[0-9]+$"

#define REG\_ONE\_ARG\_EXTENDED "^[0-9]+(\\.-?[0-9]+)+$"

#define FILE\_NAME "[0-9A-z~@#\\$\\^-\_\\(\\)\\{\\}'`]+\\.[0-9A-z~@#\\$\\^-\_\\(\\)\\{\\}'`]+$"

#define INVALID\_COMMAND "Invalid command!"

#define EXTENDED\_COMMAND\_1 "Too much args for "

#define EXTENDED\_COMMAND\_2 " flag, only first will be write!"

#define LINE\_COMMAND\_ERROR "Missed args for --line flag!"

#define INVERSE\_CIRCLE\_COMMAND\_ERROR "Missed args for --inverse\_circle flag!"

#define TRIM\_COMMAND\_ERROR "Missed args for --trim flag!"

#define TRIM\_COORDS\_ERROR "--trim coords error!"

#define FILE\_NAME\_FORGOTTEN "Put file\_name!"

#define FILE\_NAME\_ERROR "There is no file with that name!"

#define FILE\_TYPE\_ERROR "File corrupted!"

#define OUTPUT\_FILE\_ERROR "Output file name error, default name will be out.BMP!"

#define SAME\_INP\_OUT\_FILES "Input and output files are same!"

#define ERROR\_CODE 40

#pragma pack (push, 1)

typedef struct {

unsigned short signature;

unsigned int filesize;

unsigned short reserved1;

unsigned short reserved2;

unsigned int pixelArrOffset;

} BitmapFileHeader;

typedef struct {

unsigned int headerSize;

unsigned int width;

unsigned int height;

unsigned short planes;

unsigned short bitsPerPixel;

unsigned int compression;

unsigned int imageSize;

unsigned int xPixelsPerMeter;

unsigned int yPixelsPerMeter;

unsigned int colorsInColorTable;

unsigned int importantColorCount;

} BitmapInfoHeader;

typedef struct {

unsigned char b;

unsigned char g;

unsigned char r;

} Rgb;

#pragma pack(pop)

void print\_file\_header(BitmapFileHeader header)

{

printf("signature:\t%x (%hu)\n", header.signature, header.signature);

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

printf("reserved1:\t%x (%hu)\n", header.reserved1, header.reserved1);

printf("reserved2:\t%x (%hu)\n", header.reserved2, header.reserved2);

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

}

void print\_info\_header(BitmapInfoHeader header)

{

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

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

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

printf("planes: \t%x (%hu)\n", header.planes, header.planes);

printf("bitsPerPixel:\t%x (%hu)\n", header.bitsPerPixel, header.bitsPerPixel);

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

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

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

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

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

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

}

void extend\_command\_message(char\* command)

{

char\* message = malloc(COMMAND\_SIZE);

strcpy(message, EXTENDED\_COMMAND\_1);

strcat(message, command);

strcat(message, EXTENDED\_COMMAND\_2);

puts(message);

}

int check\_args(char\* optarg, const char\* exp, const char\* ext\_exp)

{

regex\_t regex\_compiled;

regmatch\_t group\_array[GROUPS];

regcomp(&regex\_compiled, exp, REG\_EXTENDED);

if(regexec(&regex\_compiled, optarg, GROUPS, group\_array, 0) == 0)

return 1;

regcomp(&regex\_compiled, ext\_exp, REG\_EXTENDED);

if(regexec(&regex\_compiled, optarg, GROUPS, group\_array, 0) == 0)

return 2;

return 0;

}

int check\_bmp(char\* file\_name)

{

FILE \*f = fopen(file\_name, "rb");

BitmapFileHeader bmfh;

BitmapInfoHeader bmif;

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

fread(&bmif, 1, sizeof(BitmapInfoHeader), f);

if((bmfh.signature == 19778 || bmfh.signature == 16706 || bmfh.signature == 674115 || bmfh.signature == 675907 || bmfh.signature == 672585 || bmfh.signature == 676944)

&& bmif.bitsPerPixel == 24 && bmif.compression == 0);

return 1;

return 0;

}

int check\_file\_name(char\* file\_name)

{

FILE \*f = fopen(file\_name, "rb");

if(f == NULL)

return 0;

return 1;

}

int check\_file\_str(char\* file\_name)

{

regex\_t regex\_compiled;

regmatch\_t group\_array[GROUPS];

regcomp(&regex\_compiled, FILE\_NAME, REG\_EXTENDED);

if(regexec(&regex\_compiled, file\_name, GROUPS, group\_array, 0) == 0)

return 1;

return 0;

}

Rgb\*\* read\_bmp(char\* file\_name, BitmapFileHeader\* bmfh, BitmapInfoHeader\* bmif)

{

FILE \*f = fopen(file\_name, "rb");

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

fread(bmif, 1, sizeof(BitmapInfoHeader), f);

unsigned int H = bmif->height;

unsigned int W = bmif->width;

Rgb\*\* arr = malloc(H \* sizeof(Rgb\*));

for(int i = 0; i < H; i++)

{

arr[H - i - 1] = malloc(W \* sizeof(Rgb) + (4 - ((W \* sizeof(Rgb)) %4) ) % 4);

fread(arr[H - i - 1], 1, W \* sizeof(Rgb) + (4 - ((W \* sizeof(Rgb)) %4) ) % 4,f);

}

fclose(f);

return arr;

}

int circle\_func(int x, int y, int x0, int y0, int r)

{

return ((x - x0) \* (x - x0) + (y - y0) \* (y - y0)) < (r\*r);

}

void set\_pixel(Rgb\*\* arr, int x, int y, int\* color, int H, int W)

{

if(y >= H || y < 0 || x >= W || x < 0)

return;

arr[y][x].r = \*(color);

arr[y][x].g = \*(color + 1);

arr[y][x].b = \*(color + 2);

}

void Circle(Rgb\*\* arr, int X1, int Y1, int\* color, int R, int H, int W)

{

int x = 0;

int y = R;

int delta = 1 - 2 \* R;

int error = 0;

while (y >= x)

{

set\_pixel(arr, X1 + x, Y1 + y, color, H, W);

set\_pixel(arr, X1 + x, Y1 - y, color, H, W);

set\_pixel(arr, X1 - x, Y1 + y, color, H, W);

set\_pixel(arr, X1 - x, Y1 - y, color, H, W);

set\_pixel(arr, X1 + y, Y1 + x, color, H, W);

set\_pixel(arr, X1 + y, Y1 - x, color, H, W);

set\_pixel(arr, X1 - y, Y1 + x, color, H, W);

set\_pixel(arr, X1 - y, Y1 - x, color, H, W);

error = 2 \* (delta + y) - 1;

if((delta < 0) && (error <= 0))

{

delta += 2 \* ++x + 1;

continue;

}

if((delta > 0) && (error > 0))

{

delta -= 2 \* --y + 1;

continue;

}

delta += 2 \* (++x - --y);

}

}

void draw\_line(Rgb\*\* arr, int H, int W, int x0, int y0, int x1, int y1, int\* color, int thickness)

{

int dx = (x1 - x0 >= 0 ? 1 : -1);

int dy = (y1 - y0 >= 0 ? 1 : -1);

int lengthX = abs(x1 - x0);

int lengthY = abs(y1 - y0);

int length = (lengthX - lengthY > 0 ? lengthX : lengthY);

if (length == 0)

Circle(arr, x0, y0, color, thickness, H, W);

int x = x0;

int y = y0;

if(lengthY <= lengthX)

{

int d = -lengthX;

length++;

while (length--)

{

if(x >= 0 && y >= 0 && x < W && y < H);

Circle(arr, x, y, color, thickness, H, W);

x += dx;

d += 2 \* lengthY;

if(d > 0)

{

d -= 2 \* lengthX;

y += dy;

}

}

}

else

{

int d = -lengthY;

length++;

while (length--)

{

if(x >= 0 && y >= 0 && x < W && y < H);

Circle(arr, x, y, color, thickness, H, W);

y += dy;

d += 2 \* lengthX;

if(d > 0)

{

d -= 2 \* lengthY;

x += dx;

}

}

}

}

void invert\_circle(Rgb\*\* arr, int H, int W, int x0, int y0, int r)

{

for(int i = 0; i < H; i++)

{

for(int j = 0; j < W; j++)

{

if(circle\_func(j, i, x0, y0, r))

{

int color[3] = {255 - arr[i][j].r, 255 - arr[i][j].g, 255 - arr[i][j].b};

set\_pixel(arr, j, i, color, H, W);

}

}

}

}

int rectangle\_func(int x, int y, int x0, int y0, int x1, int y1)

{

return (x >= x0 && y >= y0) && (x <= x1 && y <= y1);

}

Rgb\*\* trim(Rgb\*\* arr, BitmapInfoHeader\* bmif, BitmapFileHeader\* bmfh, int x0, int y0, int x1, int y1)

{

int H = bmif->height;

int W = bmif->width;

if(x0 >= x1 || y0 >= y1 || x1 < 0 || y1 < 0 || y0 > H || x0 > W)

{

puts(TRIM\_COORDS\_ERROR);

exit(ERROR\_CODE);

}

x0 = x0 >= 0 ? x0 : 0;

y0 = y0 >= 0 ? y0 : 0;

y1 = y1 < H ? y1 : H;

x1 = x1 < W ? x1 : W;

int new\_W = x1 - x0;

int new\_H = y1 - y0;

Rgb\*\* new\_arr = malloc(new\_H \* sizeof(Rgb \*));

for (int i = 0; i < new\_H; i++)

{

new\_arr[i] = malloc(new\_W \* sizeof(Rgb) + (4 - ((new\_W \* sizeof(Rgb)) %4) ) % 4);

}

for(int i = 0; i < H; i++)

{

for(int j = 0; j < W; j++)

{

if(rectangle\_func(j, i, x0, y0, x1, y1))

{

int color[3] = {arr[i][j].r, arr[i][j].g, arr[i][j].b};

set\_pixel(new\_arr, j - x0, i - y0, color, new\_H, new\_W);

}

}

}

bmif->height = new\_H;

bmif->width = new\_W;

bmfh->filesize = (new\_H)\*((new\_W)\*sizeof(Rgb) + (4 - ((new\_W \* sizeof(Rgb)) %4) ) % 4);

bmif->imageSize = new\_H\*new\_W;

return new\_arr;

}

void write\_bmp(char\* file\_name, Rgb\*\* arr, int H, int W, BitmapFileHeader bmfh, BitmapInfoHeader bmif)

{

FILE \*ff = fopen(file\_name, "wb");

fwrite(&bmfh, 1, sizeof(BitmapFileHeader), ff);

fwrite(&bmif, 1, sizeof(BitmapInfoHeader), ff);

for(int i = 0; i < H; i++)

{

fwrite(arr[H - i - 1], 1, W \* sizeof(Rgb) + (4 - ((W \* sizeof(Rgb)) %4) ) % 4, ff);

}

fclose(ff);

}

void romb\_draw\_line(Rgb\*\* arr, int H, int W, int x0, int y0, int x1, int y1, int\* color, int thickness)

{

draw\_line(arr, H, W, x0, y0, x1, y1, color, (thickness-1)/2);

if(thickness % 2 == 0)

{

if(fabs((float)(x0 - x1)) <= fabs((float)(y0 - y1)))

draw\_line(arr, H, W, x0+1, y0, x1+1, y1, color, (thickness-1)/2);

else

draw\_line(arr, H, W, x0, y0+1, x1, y1+1, color, (thickness-1)/2);

}

}

int fill\_romb(Rgb\*\* arr, int H, int W, int xv, int yv, int\* color, int\*\* used)

{

if(xv < H && xv >= 0 && yv < W && yv >=0 && !used[yv][xv] && !(arr[yv][xv].r == color[0] && arr[yv][xv].g == color[1] && arr[yv][xv].b == color[2]))

{

set\_pixel(arr, xv, yv, color, H, W);

used[yv][xv] = 1;

return fill\_romb(arr, H, W, xv+1, yv, color, used) || fill\_romb(arr, H, W, xv-1, yv, color, used) ||

fill\_romb(arr, H, W, xv, yv+1, color, used) || fill\_romb(arr, H, W, xv, yv-1, color, used);

}

else

return 0;

}

void romb(Rgb\*\* arr, int H, int W, int xv, int yv, int size, int\* color)

{

int thickness = 1;

double sqrt2 = 1.41421356237;

double x0 = xv;

double y0 = yv;

double x1 = xv-size/sqrt2;

double y1 = yv+size/sqrt2;

double x2 = xv;

double y2 = yv+2\*size/sqrt2;

double x3 = xv+size/sqrt2;

double y3 = yv+size/sqrt2;

romb\_draw\_line(arr, H, W, x0, y0, x1, y1, color, (thickness-1)/2);

romb\_draw\_line(arr, H, W, x1, y1, x2, y2, color, (thickness-1)/2);

romb\_draw\_line(arr, H, W, x2, y2, x3, y3, color, (thickness-1)/2);

romb\_draw\_line(arr, H, W, x3, y3, x0, y0, color, (thickness-1)/2);

int\*\* used = malloc(sizeof(int\*) \* H);

for(int i = 0; i < H; i++)

{

used[i] = malloc(sizeof(int) \* W);

}

for(int i = 0; i < H; i++)

{

for(int j = 0; j < W; j ++)

used[i][j] = 0;

}

fill\_romb(arr, H, W, xv, yv+2, color, used);

fill\_romb(arr, H, W, xv, yv+size/sqrt2, color, used);

fill\_romb(arr, H, W, xv, yv+10, color, used);

}

int main(int argc, char\*\* argv)

{

char\* file\_name;

int file\_name\_flag = 0;

char\* new\_file\_name = NEW\_FILE\_NAME;

BitmapFileHeader bmfh;

BitmapInfoHeader bmif;

Rgb\*\* array;

char \*opts = "hls:e:c:t:IC:r:TL:R:o:i:f";

int longidx;

static struct option longopts[] =

{

{"help", no\_argument, NULL, 'h'},

{"line", no\_argument, NULL, 'l'},

{"start", required\_argument, NULL, 's'},

{"end", required\_argument, NULL, 'e'},

{"color", required\_argument, NULL, 'c'},

{"thickness", required\_argument, NULL, 't'},

{"inverse\_circle", no\_argument, NULL, 'I'},

{"center", required\_argument, NULL, 'C'},

{"radius", required\_argument, NULL, 'r'},

{"trim", no\_argument, NULL, 'T'},

{"left\_up", required\_argument, NULL, 'L'},

{"right\_down", required\_argument, NULL, 'R'},

{"info", no\_argument, NULL, 'f'},

{"output", required\_argument, NULL, 'o'},

{"input", required\_argument, NULL, 'i'},

{"square\_rhombus", no\_argument, NULL, 'S'},

{"upper\_vertex", required\_argument, NULL, 'u'},

{"size", required\_argument, NULL, 'z'},

{"fill\_color", required\_argument, NULL, 'F'},

{NULL, no\_argument, NULL, 0 }

};

int opt = getopt\_long(argc, argv, opts, longopts, &longidx);

if(argc == 1)

{

puts(DEFAULT\_MESSAGE);

exit(0);

}

int cnt\_commands = 0;

int info\_flag = 0;

int x0,y0,x1,y1,thickness;

int color[3];

int line\_flag = 0; int start\_flag = 0; int end\_flag = 0; int thickness\_flag = 0; int color\_flag = 0;

int xc,yc,rad;

int center\_flag = 0; int rad\_flag = 0; int inverse\_circle\_flag = 0;

int xl,yl,xr,yr;

int left\_flag = 0; int right\_flag = 0; int trim\_flag = 0;

int rombus\_flag = 0;

int vert\_flag = 0;

int size\_flag = 0;

int fill\_flag = 0;

int fill\_color[3];

int rb\_size;

int xv, yv;

while(opt != -1)

{

switch (opt)

{

case('h'):

{

puts(HELP\_MESSAGE);

exit(0);

}

case('f'):

{

info\_flag = 1;

cnt\_commands++;

break;

}

case('i'):

{

file\_name\_flag = 1;

file\_name = optarg;

cnt\_commands+=2;

break;

}

case('o'):

{

cnt\_commands+=2;

if(check\_file\_str(optarg))

{

new\_file\_name = optarg;

}

else

puts(OUTPUT\_FILE\_ERROR);

break;

}

case('l'):

{

line\_flag = 1;

cnt\_commands++;

break;

}

case('s'):

{

if(check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED))

sscanf(optarg, "%d.%d", &x0, &y0);

if(check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED) == 2)

extend\_command\_message("--start");

if(!check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED))

{

puts(INVALID\_COMMAND);

exit(ERROR\_CODE);

}

start\_flag = 1;

cnt\_commands+=2;

break;

}

case('e'):

{

if(check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED))

sscanf(optarg, "%d.%d", &x1, &y1);

if(check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED) == 2)

extend\_command\_message("--end");

if(!check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED))

{

puts(INVALID\_COMMAND);

exit(ERROR\_CODE);

}

end\_flag = 1;

cnt\_commands+=2;

break;

}

case('c'):

{

if(check\_args(optarg, REG\_LINE\_COLOR, REG\_LINE\_COLOR\_EXTENDED))

sscanf(optarg, "%d.%d.%d", &color[0], &color[1], &color[2]);

if(check\_args(optarg, REG\_LINE\_COLOR, REG\_LINE\_COLOR\_EXTENDED) == 2)

extend\_command\_message("--color");

if(!check\_args(optarg, REG\_LINE\_COLOR, REG\_LINE\_COLOR\_EXTENDED) || (color[0] > 255 || color[1] > 255 || color[2] > 255))

{

puts(INVALID\_COMMAND);

exit(ERROR\_CODE);

}

color\_flag = 1;

cnt\_commands+=2;

break;

}

case('t'):

{

if(check\_args(optarg, REG\_ONE\_ARG, REG\_ONE\_ARG\_EXTENDED))

sscanf(optarg, "%d", &thickness);

if(check\_args(optarg, REG\_ONE\_ARG, REG\_ONE\_ARG\_EXTENDED) == 2)

extend\_command\_message("--thickness");

if(!check\_args(optarg, REG\_ONE\_ARG, REG\_ONE\_ARG\_EXTENDED))

{

puts(INVALID\_COMMAND);

exit(ERROR\_CODE);

}

thickness\_flag = 1;

cnt\_commands+=2;

break;

}

case('I'):

{

inverse\_circle\_flag = 1;

cnt\_commands++;

break;

}

case('C'):

{

if(check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED))

sscanf(optarg, "%d.%d", &xc, &yc);

if(check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED) == 2)

extend\_command\_message("--center");

if(!check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED))

{

puts(INVALID\_COMMAND);

exit(ERROR\_CODE);

}

center\_flag = 1;

cnt\_commands+=2;

break;

}

case('r'):

{

if(check\_args(optarg, REG\_ONE\_ARG, REG\_ONE\_ARG\_EXTENDED))

sscanf(optarg, "%d", &rad);

if(check\_args(optarg, REG\_ONE\_ARG, REG\_ONE\_ARG\_EXTENDED) == 2)

extend\_command\_message("--radius");

if(!check\_args(optarg, REG\_ONE\_ARG, REG\_ONE\_ARG\_EXTENDED))

{

puts(INVALID\_COMMAND);

exit(ERROR\_CODE);

}

rad\_flag = 1;

cnt\_commands+=2;

break;

}

case('T'):

{

trim\_flag = 1;

cnt\_commands++;

break;

}

case('L'):

{

if(check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED))

sscanf(optarg, "%d.%d", &xl, &yl);

if(check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED) == 2)

extend\_command\_message("--left\_up");

if(!check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED))

{

puts(INVALID\_COMMAND);

exit(ERROR\_CODE);

}

left\_flag = 1;

cnt\_commands+=2;

break;

}

case('R'):

{

if(check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED))

sscanf(optarg, "%d.%d", &xr, &yr);

if(check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED) == 2)

extend\_command\_message("--right\_down");

if(!check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED))

{

puts(INVALID\_COMMAND);

exit(ERROR\_CODE);

}

right\_flag = 1;

cnt\_commands+=2;

break;

}

case('S'):

{

rombus\_flag = 1;

break;

cnt\_commands++;

}

case('u'):

{

if(check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED))

sscanf(optarg, "%d.%d", &xv, &yv);

if(check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED) == 2)

extend\_command\_message("--upper\_vertex");

if(!check\_args(optarg, REG\_COORDS, REG\_COORDS\_EXTENDED))

{

puts(INVALID\_COMMAND);

exit(ERROR\_CODE);

}

cnt\_commands+=2;

vert\_flag = 1;

break;

}

case('z'):

{

if(check\_args(optarg, REG\_ONE\_ARG, REG\_ONE\_ARG\_EXTENDED))

sscanf(optarg, "%d", &rb\_size);

if(check\_args(optarg, REG\_ONE\_ARG, REG\_ONE\_ARG\_EXTENDED) == 2)

extend\_command\_message("--size");

if(!check\_args(optarg, REG\_ONE\_ARG, REG\_ONE\_ARG\_EXTENDED))

{

puts(INVALID\_COMMAND);

exit(ERROR\_CODE);

}

size\_flag = 1;

cnt\_commands+=2;

break;

}

case('F'):

{

if(check\_args(optarg, REG\_LINE\_COLOR, REG\_LINE\_COLOR\_EXTENDED))

sscanf(optarg, "%d.%d.%d", &fill\_color[0], &fill\_color[1], &fill\_color[2]);

if(check\_args(optarg, REG\_LINE\_COLOR, REG\_LINE\_COLOR\_EXTENDED) == 2)

extend\_command\_message("--fill\_color");

if(!check\_args(optarg, REG\_LINE\_COLOR, REG\_LINE\_COLOR\_EXTENDED) || (fill\_color[0] > 255 || fill\_color[1] > 255 || fill\_color[2] > 255))

{

puts(INVALID\_COMMAND);

exit(ERROR\_CODE);

}

fill\_flag = 1;

cnt\_commands+=2;

break;

}

default:

{

puts(INVALID\_COMMAND);

exit(ERROR\_CODE);

}

}

opt = getopt\_long(argc, argv, opts, longopts, &longidx);

}

if(!file\_name\_flag)

{

if(check\_file\_str(argv[argc-1]) && cnt\_commands == argc - 2)

file\_name = argv[argc-1];

else

{

puts(FILE\_NAME\_FORGOTTEN);

exit(ERROR\_CODE);

}

}

if(!strcmp(file\_name, new\_file\_name))

{

puts(SAME\_INP\_OUT\_FILES);

exit(ERROR\_CODE);

}

if(!check\_file\_name(file\_name))

{

puts(FILE\_NAME\_ERROR);

exit(ERROR\_CODE);

}

if(!check\_bmp(file\_name))

{

puts(FILE\_TYPE\_ERROR);

exit(ERROR\_CODE);

}

array = read\_bmp(file\_name, &bmfh, &bmif);

if(info\_flag)

{

print\_file\_header(bmfh);

printf("\n");

print\_info\_header(bmif);

}

if(line\_flag)

{

if(start\_flag && end\_flag && color\_flag && thickness\_flag)

{

draw\_line(array, bmif.height, bmif.width, x0, y0, x1, y1, color, (thickness-1)/2);

if(thickness % 2 == 0)

{

if(fabs((float)(x0 - x1)) <= fabs((float)(y0 - y1)))

draw\_line(array, bmif.height, bmif.width, x0+1, y0, x1+1, y1, color, (thickness-1)/2);

else

draw\_line(array, bmif.height, bmif.width, x0, y0+1, x1, y1+1, color, (thickness-1)/2);

}

}

else

{

puts(LINE\_COMMAND\_ERROR);

exit(ERROR\_CODE);

}

}

if(inverse\_circle\_flag)

{

if(center\_flag && rad\_flag)

invert\_circle(array, bmif.height, bmif.width, xc, yc, rad);

else

{

puts(INVERSE\_CIRCLE\_COMMAND\_ERROR);

exit(ERROR\_CODE);

}

}

if(trim\_flag)

{

if(left\_flag && right\_flag)

array = trim(array, &bmif, &bmfh, xl, yl, xr, yr);

else

{

puts(INVERSE\_CIRCLE\_COMMAND\_ERROR);

exit(ERROR\_CODE);

}

}

if(rombus\_flag)

{

if(size\_flag && vert\_flag && fill\_flag)

{

romb(array, bmif.height, bmif.width, xv, yv, rb\_size, fill\_color);

}

else

{

puts("SQUARE\_ERROR!");

exit(ERROR\_CODE);

}

}

if(!info\_flag)

write\_bmp(new\_file\_name, array, bmif.height, bmif.width, bmfh, bmif);

exit(0);

}