

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

**КУРСОВАЯ РАБОТА**  
**по дисциплине «Программирование»**  
**ТЕМА: РАБОТА С ИЗОБРАЖЕНИЯМИ**

Студент гр. 3342

Корниенко А.Е.

Преподаватель

Глазунов С.А.

Санкт-Петербург

2024

## ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Корниенко А.Е.

Группа 3342

Тема работы: Работа с изображениями

Исходные данные:

Вариант 4.2

### ***Задание***

Программа должна иметь следующую функции по обработке изображений:

- Заменяет все пиксели одного заданного цвета на другой цвет. Флаг для выполнения данной операции: `--color_replace`. Функционал определяется:
  - Цвет, который требуется заменить. Флаг `--old_color` (цвет задаётся строкой `rrr.ggg.bbb`, где `rrr/ggg/bbb` – числа, задающие цветовую компоненту. пример `--old_color 255.0.0` задаёт красный цвет)
  - Цвет на который требуется заменить. Флаг `--new_color` (работает аналогично флагу `--old_color`)
- Фильтр rgb-компонент. Флаг для выполнения данной операции: `--rgbfilter`. Этот инструмент должен позволять для всего изображения либо установить в диапазоне от 0 до 255 значение заданной компоненты. Функционал определяется
  - Какую компоненту требуется изменить. Флаг `--component_name`. Возможные значения `red`, `green` и `blue`
  - В какой значение ее требуется изменить. Флаг `--`

`component_value`'. Принимает значение в виде числа от 0 до 255

- Разделяет изображение на  $N \times M$  частей. Флаг для выполнения данной операции: `--split`. Реализация: провести линии заданной толщины. Функционал определяется:
  - Количество частей по “оси” Y. Флаг `--number_x`. На вход принимает число больше 1
  - Количество частей по “оси” X. Флаг `--number_y`. На вход принимает число больше 1
  - Толщина линии. Флаг `--thickness`. На вход принимает число больше 0
  - Цвет линии. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где `rrr/ggg/bbb` – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет)

Дата выдачи задания: 18.03.2024

Дата сдачи реферата: 27.05.2024

Дата защиты реферата: 29.05.2024

Студент \_\_\_\_\_ Корниенко А.Е.

Преподаватель \_\_\_\_\_ Глазунов С.А.

## **АННОТАЦИЯ**

Курсовая работа представляет собой программу, реализующую CLI и обрабатывающую изображение формата BMP в соответствии с переданными пользователем в неё опциями (замена пикселей цвета, замена компоненты цвета, разделение изображения на части). Для выполнения этой задачи программа использует функции различных библиотек языка Си, в том числе “getopt.h” для реализации CLI. Выполнив необходимые преобразования создаётся файл с изменённым изображением.

## **SUMMARY**

The course work is a program that implements the CLI and processes an image in BMP format in accordance with the options passed to it by the user (replacing color pixels, replacing color components, dividing the image into parts). To accomplish this task, the program uses functions from various C language libraries, including “getopt.h” to implement the CLI. After performing the necessary transformations, a file with a modified image is created.

## СОДЕРЖАНИЕ

Введение.....	7
1. Структуры .....	8
1.1 Структура BitmapFileHeader .....	8
1.2 Структура BitmapInfoHeader.....	8
1.3 Структура Rgb .....	9
2. Функции .....	10
2.1 Функции первого задания .....	11
2.2 Функции второго задания .....	10
2.3 Функции третьего задания .....	10
2.4 Функции работы с файлами .....	10
2.5 Другие функции .....	11
2.6 Функция main .....	11
Заключение .....	12
Список используемой литературы .....	13
Приложение А .....	14
Приложение Б .....	17

## **ВВЕДЕНИЕ**

Целью данной работы является написание программы, осуществляющую обработку изображения в соответствии с опциями, введёнными пользователем. Для этого требуется:

- Создать функции, реализующие CLI.
- Создать функции, обрабатывающие изображение в соответствии с выбором пользователя.

# 1. СТРУКТУРЫ

## 1.1. Структура **BitmapFileHeader**

Структура **BitmapFileHeader** состоит из таких полей как:

- unsigned short signature - поле заголовка, используемое для идентификации файла BMP и DIB, имеет шестнадцатеричное значение, равное BM в ASCII.
- unsigned int filesize - размер файла BMP в байтах.
- unsigned short reserved1 - зарезервировано; фактическое значение зависит от приложения, создающего изображение.
- unsigned short reserved2 - зарезервировано; фактическое значение зависит от приложения, создающего изображение.
- unsigned int pixelArrOffset - смещение, т. е. начальный адрес байта, в котором находятся данные изображения (массив пикселей).

## 1.2. Структура **BitmapInfoHeader**

Структура **BitmapInfoHeader** состоит из таких полей как:

- unsigned int headerSize – размер этого заголовка в байтах.
- unsigned int width – ширина изображения в пикселях.
- unsigned int height – длина изображения в пикселях.
- unsigned short planes – количество цветовых плоскостей.
- unsigned short bitsPerPixel – глубина цвета изображения.
- unsigned int compression – используемый метод сжатия.
- unsigned int imageSize – размер изображения.
- unsigned int xPixelsPerMeter – горизонтальное разрешение изображения.
- unsigned int yPixelsPerMeter – вертикальное разрешение изображения.



- `unsigned int colorsInColorTable` – количество цветов в цветовой палитре.
- `unsigned int importantColorCount` – количество используемых важных цветов.

### 1.3. Структура **Rgb**

Структура `Rgb` состоит из таких полей как:

- `unsigned char b` – синяя компонента цвета.
- `unsigned char g` – зелёная компонента цвета.
- `unsigned char r` – красная компонента цвета.

## **2. ФУНКЦИИ**

### **2.1. Функции первого задания**

Функция `replace_color(Rgb** arr, unsigned int H, unsigned int W, unsigned char r_old, unsigned char g_old, unsigned char b_old, unsigned char r_new, unsigned char g_new, unsigned char b_new)` принимает в качестве аргументов массив пикселей, высоту и ширину изображения, значения пикселей, которые нужно заменить и значения, на которые нужно заменить. Для преобразования строки типа `rrr.ggg.bbb` в массив используется функция `parsing_args(char* str)`, который возвращает массив типа `int*`.

### **2.2. Функции второго задания**

Функция `rgbfilter(Rgb** arr, unsigned int H, unsigned int W, char* component_name, int component_value)` принимает в качестве аргументов массив пикселей, высоту и ширину изображения, название компоненты, которую нужно изменить, значение, которое нужно изменить.

### **2.3. Функции третьего задания**

Функция `split(Rgb** arr, unsigned int H, unsigned int W, int countY, int countX, int thicknessm, char* color)` принимает в качестве аргументов массив пикселей, высоту и ширину изображения, количество частей по оси Y, количество частей по оси X, толщину линий, цвет линий. Для рисования линий используется функция `draw_line`.

### **2.4. Функции работы с файлами**

Функция `read_bmp(char file_name[], BitmapFileHeader * bmfh, BitmapInfoHeader * bmih)` считывает переданный файл в структуры `BitmapFileHeader` и `BitmapInfoHeader`, все пиксели в массив типа `Rgb**` и возвращает его.

Функция `isBMP (BitmapFileHeader * bmfh, BitmapInfoHeader * bmih)` проверяет соответствует ли файл формату `bmp`.

Функция `write_bmp (char filename[], Rgb ** arr, unsigned int H, unsigned int W, BitmapFileHeader bmfh, BitmapInfoHeader bmih)` записывает в новый файл все данные, полученные после обработки изображения.

## 2.5. Другие функции

Функция `printFileHeader (BitmapFileHeader header)` печатает всю информацию о заголовке файла.

Функция `printInfoHeader (BitmapInfoHeader header)` печатает всю информацию о заголовке изображения.

Функция `help` выводит информацию о доступных опциях.

Для каждого задания написаны функции, которые проверяют правильность переданных аргументов:  
`check_data_for_function_<name_function>`.

## 2.6. Функция `main`

Функция `int main()` осуществляет запись значений через CLI, используя функцию `getopt_long`.

Примеры работы программы см. в приложении А.

Разработанный программный код см. в приложении В.

## **ЗАКЛЮЧЕНИЕ**

Курсовая работа включала в себя с изображениях и способах их обработки, изучение и при помощи функций стандартной библиотеки языка С.

Для работы с bmp-файлом были использованы структуры. Были изучены особенности языка, связанные с обработкой изображений и реализацией CLI. Реализованы основные функции чтения и обработки BMP изображений.

В итоге написана программа, которая выполняет поставленные задачи.

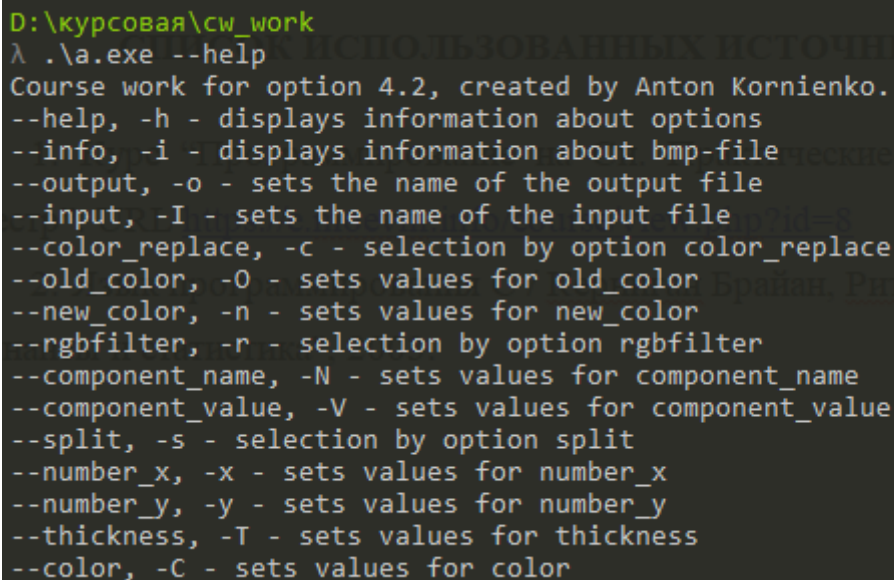
## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Курс “Программирование на Си. Практические задания. Второй семестр”. URL <https://e.moevm.info/course/view.php?id=8>
2. Язык программирования С / Керниган Брайан, Ритчи Деннис. СПб.: "Финансы и статистика", 2003.

## ПРИЛОЖЕНИЕ А

### ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

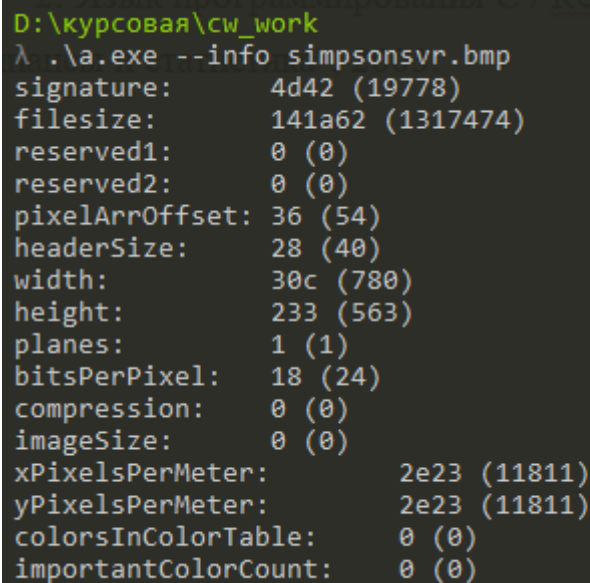
ПРИМЕР 1 – Тестирование вывода справки.



```
D:\курсовая\cw_work
λ .\a.exe --help
Course work for option 4.2, created by Anton Kornienko.
--help, -h - displays information about options
--info, -i - displays information about bmp-file
--output, -o - sets the name of the output file
--input, -I - sets the name of the input file
--color_replace, -c - selection by option color_replace
--old_color, -O - sets values for old_color
--new_color, -n - sets values for new_color
--rgbfilter, -r - selection by option rgbfilter
--component_name, -N - sets values for component_name
--component_value, -V - sets values for component_value
--split, -s - selection by option split
--number_x, -x - sets values for number_x
--number_y, -y - sets values for number_y
--thickness, -T - sets values for thickness
--color, -C - sets values for color
```

Рисунок 1 – вывод справки в консоль

ПРИМЕР 2 – Тестирование вывода информации о файле.



```
D:\курсовая\cw_work
λ .\a.exe --info simpsonsvr.bmp
signature:      4d42 (19778)
filesize:      141a62 (1317474)
reserved1:      0 (0)
reserved2:      0 (0)
pixelArrOffset: 36 (54)
headerSize:     28 (40)
width:          30c (780)
height:         233 (563)
planes:         1 (1)
bitsPerPixel:   18 (24)
compression:    0 (0)
imageSize:      0 (0)
xPixelsPerMeter: 2e23 (11811)
yPixelsPerMeter: 2e23 (11811)
colorsInColorTable: 0 (0)
importantColorCount: 0 (0)
```

Рисунок 2 – вывод информации о файле в консоль



Рисунок 3 – исходное изображение

ПРИМЕР 3 – Тестирование функции color\_replace.

Параметры запуска: `.\cw --color_replace --old_color 255.255.255 --new_color 0.0.0 simpsonsvr.bmp`



Рисунок 4 – результат работы функции color\_replace

ПРИМЕР 4 – Тестирование функции rgbfilter

Параметры запуска: `.\cw --rgbfilter --component_name red --component_value 255 simpsonsvr.bmp`



Рисунок 5 – результат работы функции `rgbfilter`

#### ПРИМЕР 5 – Тестирование функции `split`

Параметры запуска: `.\cw --split --number_x 2 --number_y 3 --thickness 5 --color 255.0.0 simpsonsvr.bmp`

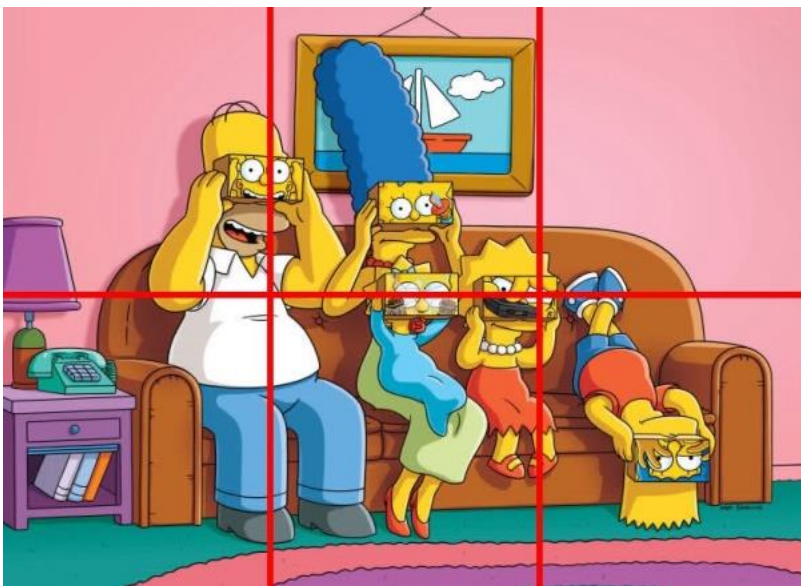


Рисунок 6 – результат работы функции `split`



## ПРИЛОЖЕНИЕ Б

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define ERROR_UNCORRECT_ARGUMENTS 40

#define ERROR_FILE_READ_ERROR 41

#define ERROR_MEMORY_ALLOCATION_FAILURE 42

#define ERROR_WITH_OPTION 43

int* parsing_args(char* str){

    int idx = 2;
    int* array = malloc(3 * sizeof(int));
    if(array == NULL){
        printf("Memory error\n");
        exit(ERROR_MEMORY_ALLOCATION_FAILURE);
    }

    int temp = 0;
    int place_number = 1;

    for(int i = strlen(str) - 1; i >= 0; i--){
        if(str[i] == '.'){
            array[idx] = temp;
            idx--;
            temp = 0;
            place_number = 1;
        } else {
            temp += (str[i] - '0') * place_number;
            place_number *= 10;
        }
    }
    array[idx] = temp;
    return array;
}

#pragma pack (push, 1)
typedef struct BitmapFileHeader
{
    unsigned short signature;    // определение типа файла
    unsigned int filesize;      // размер файла
    unsigned short reserved1;    // должен быть 0
    unsigned short reserved2;    // должен быть 0
}
```

```

        unsigned int pixelArrOffset; // начальный адрес байта, в котором
находятся данные изображения (массив пикселей)
    } BitmapFileHeader;

typedef struct BitmapInfoHeader
{
    unsigned int headerSize;           // размер этого заголовка в
байтах
    unsigned int width;                // ширина изображения в пикселях
    unsigned int height;               // высота изображения в пикселях
    unsigned short planes;              // кол-во цветовых плоскостей
(должно быть 1)
    unsigned short bitsPerPixel;        // глубина цвета изображения
    unsigned int compression;           // тип сжатия; если сжатия не
используется, то здесь должен быть 0
    unsigned int imageSize;             // размер изображения
    unsigned int xPixelsPerMeter;        // горизонтальное разрешение
(пиксель на метр)
    unsigned int yPixelsPerMeter;        // вертикальное разрешение
(пиксель на метр)
    unsigned int colorsInColorTable;     // кол-во цветов в цветовой
палитре
    unsigned int importantColorCount;    // кол-во важных цветов (или 0,
если каждый цвет важен)
} BitmapInfoHeader;

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

#pragma pack(pop)

void isBMP(BitmapFileHeader bmfh, BitmapInfoHeader bmif){
    if(!(bmfh.signature == 0x4d42 && bmif.headerSize == 40 &&
bmif.bitsPerPixel == 24 && bmif.compression == 0)){
        printf("file not bmp\n");
        exit(ERROR_FILE_READ_ERROR);
    }
}

void printFileHeader(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 printInfoHeader(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 swap_int(int *a, int *b){
    int t = *a;
    *a = *b;
    *b = t;
}

void draw_line(Rgb **arr, int H, int W, int x0, int y0, int x1, int
y1, int thickness, char* color)
{
    int* color_draw = parsing_args(color);
    if (x0 < 0 || y0 < 0 || x1 < 0 || y1 < 0 || thickness <= 0){
        return ;
    }
    // vertical
    if (x0 == x1){
        if (y0 > y1){
            swap_int(&y0, &y1);
        }
        for (int y = y0; y <= y1; y++){
            for (int j = 0; j <= thickness; j++){
                if (H - y >= 0 && x0 - j >= 0 && x0 - j < W && H
- y < H){
                    arr[H - y][x0 - j].r = color_draw[0];
                    arr[H - y][x0 - j].g = color_draw[1];
                    arr[H - y][x0 - j].b = color_draw[2];
                }
            }
        }
    } else if (y0 == y1){
        if (x0 > x1){
            swap_int(&x0, &x1);
        }
        for (int x = x0; x <= x1; x++){

```

```

        for (int j = 0; j <= thickness; j++){
            if (H - y0 + j >= 0 && x >= 0 && x < W && H - y0
+ j < H){
                arr[H - y0 + j][x].r = color_draw[0];
                arr[H - y0 + j][x].g = color_draw[1];
                arr[H - y0 + j][x].b = color_draw[2];
            }
        }
    }

    free(color_draw);
}
//swap(&arr[i][j].r, &arr[i][j].g);
void swap(char *a, char *b){
    char t = *a;
    *a = *b;
    *b = t;
}

void check_data_for_function_replace_color(Rgb** arr, unsigned int H,
unsigned int W, int r_old, int g_old, int b_old, int r_new, int g_new,
int b_new){
    if(!(arr != NULL && 0 <= r_old && r_old <= 255 && 0 <= g_old
&& g_old <= 255 && 0 <= b_old && b_old <= 255 && 0 <= r_new &&
r_new <= 255 && 0 <= g_new && r_new <= 255 && 0 <= b_new && b_new <=
255)){
        printf("uncorrect data\n");
        exit(ERROR_UNCORRECT_ARGUMENTS);
    }
}

void replace_color(Rgb** arr, unsigned int H, unsigned int W, unsigned
char r_old, unsigned char g_old, unsigned char b_old, unsigned char
r_new, unsigned char g_new, unsigned char b_new)
{
    for(int i = 0; i < H; i++){
        for(int j = 0; j < W; j++){
            if(arr[i][j].r == r_old && arr[i][j].g == g_old &&
arr[i][j].b == b_old){
                arr[i][j].r = r_new;
                arr[i][j].g = g_new;
                arr[i][j].b = b_new;
            }
            //printf("%d  %d  %d\n", arr[i][j].r, arr[i][j].g,
arr[i][j].b);
        }
    }
}

void check_data_for_function_rgbfilter(Rgb** arr, unsigned int H,
unsigned int W, char* component_name, int component_value)
{

```

```

        if(!(arr != NULL && (strcmp(component_name, "red") == 0 ||
        strcmp(component_name, "green") == 0 || strcmp(component_name, "blue")
        == 0) && 0 <= component_value && component_value <= 255)){
            printf("incorrect data\n");
            exit(ERROR_UNCORRECT_ARGUMENTS);
        }
    }

void rgbfilter(Rgb** arr, unsigned int H, unsigned int W, char*
component_name, int component_value)
{
    unsigned char v = component_name[0];

    for(int i = 0; i < H; i++){
        for(int j = 0; j < W; j++){
            if(v == 'r'){
                arr[i][j].r = component_value;
            }
            else if(v == 'g'){
                arr[i][j].g = component_value;
            }
            else if(v == 'b'){
                arr[i][j].b = component_value;
            }
        }
    }
}

void check_data_for_function_split(Rgb** arr, unsigned int H, unsigned
int W, int count_y, int count_x, int thicknessm, char* color){
    int* color_check = parsing_args(color);
    if(!(arr != NULL && count_y > 1 && count_x > 1 && thicknessm > 0
&& 0 <= color_check[0] && color_check[0] <= 255 && 0 <= color_check[1]
&& color_check[1] <= 255 && 0 <= color_check[2] && color_check[2] <=
255)){
        printf("incorrect data\n");
        free(color_check);
        exit(ERROR_UNCORRECT_ARGUMENTS);
    }
    free(color_check);
}

void split(Rgb** arr, unsigned int H, unsigned int W, int count_y,
int count_x, int thicknessm, char* color)
{
    int countlines_y = count_y - 1;
    int countlines_x = count_x - 1;
    // vertical
    int x0 = W / count_x;
    int y0 = H;
    int y1 = 0;
    int cnt_x = 0;
    while (cnt_x != countlines_x)
    {
        draw_line(arr, H, W, x0, y0, x0, y1, thicknessm, color);
        x0 += W / count_x;
        cnt_x ++;
    }
}

```

```

    x0 = 0;
    int x1 = W;
    y0 = H / count_y;
    int cnt_y = 0;
    while (cnt_y != countlines_y)
    {
        draw_line(arr, H, W, x0, y0, x1, y0, thicknessm, color);
        y0 += H / count_y;
        cnt_y ++;
    }
}

Rgb **read_bmp(char file_name[], BitmapFileHeader*bmfh,
BitmapInfoHeader* bmif){
    FILE *f = fopen(file_name, "rb");
    if(!f){
        printf("file reading error\n");
        exit(ERROR_FILE_READ_ERROR);
    }
    fread(bmfh, 1, sizeof(BitmapFileHeader), f);
    fread(bmif, 1, sizeof(BitmapInfoHeader), f);
    isBMP(*bmfh, *bmif);
    unsigned int H = bmif->height;
    unsigned int W = bmif->width;
    Rgb **arr = malloc(H * sizeof(Rgb*));
    if(arr == NULL){
        printf("Memory error\n");
        exit(ERROR_MEMORY_ALLOCATION_FAILURE);
    }
    for(int i = 0; i < H; i++){
        arr[i] = malloc(W * sizeof(Rgb) + (4 - (W * sizeof(Rgb)) %
4) % 4);
        if(arr[i] == NULL){
            printf("Memory error\n");
            exit(ERROR_MEMORY_ALLOCATION_FAILURE);
        }
        fread(arr[i], 1, W * sizeof(Rgb) + (4 - (W * sizeof(Rgb)) %
4) % 4, f);
    }
    fclose(f);
    return 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, sizeof(BitmapFileHeader), 1, ff);
    fwrite(&bmif, sizeof(BitmapInfoHeader), 1, ff);
    int padding = (4 - (W * 3) % 4) % 4;
    uint8_t paddingBytes[3] = { 0 };
    for(int i = 0; i < H; i++){
        fwrite(arr[i], sizeof(Rgb), W, ff);
        fwrite(paddingBytes, sizeof(uint8_t), padding, ff);
    }
}

```

```

        isBMP(bmfh, bmif);
        fclose(ff);
    }

void Help_output(){
    printf("Course work for option 4.2, created by Anton
Kornienko.\n");
    printf("--help, -h - displays information about options\n");
    printf("--info, -i - displays information about bmp-file\n");
    printf("--output, -o - sets the name of the output file\n");
    printf("--input, -I - sets the name of the input file\n");
    printf("--color_replace, -c - selection by option
color_replace\n");
    printf("--old_color, -O - sets values for old_color\n");
    printf("--new_color, -n - sets values for new_color\n");
    printf("--rgbfilter, -r - selection by option rgbfilter\n");
    printf("--component_name, -N - sets values for
component_name\n");
    printf("--component_value, -V - sets values for
component_value\n");
    printf("--split, -s - selection by option split\n");
    printf("--number_x, -x - sets values for number_x\n");
    printf("--number_y, -y - sets values for number_y\n");
    printf("--thickness, -T - sets values for thickness\n");
    printf("--color, -C - sets values for color\n");
}

void f(Rgb **arr, int H, int W, float value){
    for(int y = 0; y < H; y++){
        for(int x = 0; x < W; x++){
            float n1 = (float)arr[y][x].r / 255;
            float n2 = (float)arr[y][x].g / 255;
            float n3 = (float)arr[y][x].b / 255;

            float arr1 = powf(n1, value) * 255;
            float arr2 = powf(n2, value) * 255;
            float arr3 = powf(n3, value) * 255;

            arr[y][x].r = (int)arr1;
            arr[y][x].g = (int)arr2;
            arr[y][x].b = (int)arr3;
        }
    }
}

int main(int argc, char *argv[]){
    char* input_file = argv[argc - 1];
    char* output_file = "output.bmp";
    const char* short_options = "hio:I:ct:n:rN:V:sx:y:T:Q:gv:";

    const struct option long_options[] = {
        {"help", no_argument, 0, 'h'},
        {"info", no_argument, 0, 'i'},
        {"output", required_argument, 0, 'o'},

```

```

        {"input", required_argument, 0, 'I' },
        {"color_replace", no_argument, 0, 'c'},
        {"old_color", required_argument, 0, 'O'},
        {"new_color", required_argument, 0, 'n'},
        {"rgbfilter", no_argument, 0, 'r'},
        {"component_name", required_argument, 0, 'N'},
        {"component_value", required_argument, 0, 'V'},
        {"split", no_argument, 0, 's'},
        {"number_x", required_argument, 0, 'x'},
        {"number_y", required_argument, 0, 'y'},
        {"thickness", required_argument, 0, 'T'},
        {"color", required_argument, 0, 'C'},
        {"gamma", no_argument, 0, 'g'},
        {"value", required_argument, 0, 'v'},
        {0, 0, 0, 0}
};

int opt;
int option_index;
int make_info_about_file = 0;
int option = 0;

char* str;
float value;

char* old_color;
char* new_color;
char* component_name;
int component_value = -1;
int number_x = -1;
int number_y = -1;
int thickness = -1;
char* color;

while ((opt=getopt_long(argc,argv,short_options,
long_options,&option_index))!=-1){
    switch(opt){
        case 'h': {
            Help_output();
            exit(EXIT_SUCCESS);
            break;
        };
        case 'g':{
            option = 4;

            break;
        };
        case 'v':{
            str = optarg;
            char* end;
            value = strtod(str, &end);
            break;
        }
        case 'c': {
            option = 1;
            break;

```



```

};
case 'r': {
    option = 2;
    break;
};
case 's': {
    option = 3;
    break;
}
case 'o':{
    output_file = optarg;
    break;
};
case 'i':{
    make_info_about_file = 1;
    break;
};
case 'O':{
    old_color = optarg;
    break;
};
case 'n':{
    new_color = optarg;
    break;
};
case 'N':{
    component_name = optarg;
    break;
}
case 'V':{
    component_value = atoi(optarg);
    break;
};
case 'x':{
    number_x = atoi(optarg);
    break;
};
case 'y':{
    number_y = atoi(optarg);
    break;
};
case 'T':{
    thickness = atoi(optarg);
    break;
};
case 'C':{
    color = optarg;
    break;
};
case 'I':{
    input_file = optarg;
    break;
};
case '?': {
    printf("found unknown option\n");
    exit(ERROR_WITH_OPTION);
    break;
};

```

```

        };

    }

}

BitmapFileHeader bmfh;
BitmapInfoHeader bmif;
Rgb **arr = read_bmp(input_file, &bmfh, &bmif);

if(make_info_about_file == 1){
    printFileHeader(bmfh);
    printInfoHeader(bmif);
    exit(EXIT_SUCCESS);
}

unsigned int H = bmif.height;
unsigned int W = bmif.width;

switch(option){
    case 1:{
        int* old_data_color = parsing_args(old_color);
        int* new_data_color = parsing_args(new_color);
        check_data_for_function_replace_color(arr, H, W,
old_data_color[0], old_data_color[1], old_data_color[2],
new_data_color[0], new_data_color[1], new_data_color[2]);
        replace_color(arr, H, W, old_data_color[0],
old_data_color[1], old_data_color[2],
new_data_color[0], new_data_color[1], new_data_color[2]);
        free(old_data_color);
        free(new_data_color);
        break;
    };

    case 2:{
        check_data_for_function_rgbfilter(arr, H, W,
component_name, component_value);
        rgbfilter(arr, H, W, component_name, component_value);
        break;
    };

    case 3:{
        check_data_for_function_split(arr, H, W, number_x,
number_y, thickness, color);
        split(arr, H, W, number_x, number_y, thickness, color);
        break;
    };

    case 4:{

```

```

        f(arr, H, W, value);
        break;
    };

    default:{
        printf("didn't select the option\n");
        exit(ERROR_WITH_OPTION);
        break;
    }
}

write_bmp(output_file, arr, H, W, bmfh,bmif);

for(int i = 0; i < H; i++){
    free(arr[i]);
}

free(arr);

return 0;
}

```