**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\*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**

Введение7

1. Структуры8

1.1 Структура BitmapFileHeader8

1.2 Структура BitmapInfoHeader8

1.3 Структура Rgb9

2. Функции10

2.1 Функции первого задания11

* 1. Функции второго задания10

2.3 Функции третьего задания10

2.4 Функции работы с файлами10

2.5 Другие функции11

2.6 Функция main11

Заключение12

Список используемой литературы13

Приложение А14

Приложение Б17

**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. **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. **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. **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\*.

* 1. **. 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.

* 1. **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.

* 1. **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.

* 1. **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>.

* 1. **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 B.

**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 C / Керниган Брайан, Ритчи Деннис. СПб.: "Финансы и статистика", 2003.

**приложение А**

**ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ**

Пример 1 – Тестирование вывода справки.

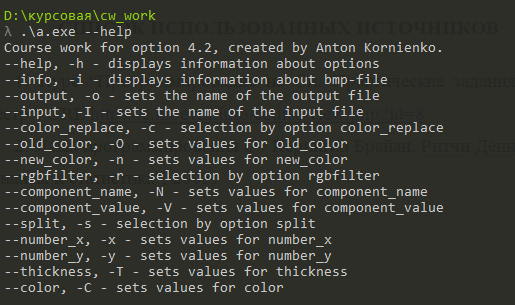


Рисунок 1 – вывод справки в консоль

ПРИМЕР 2 – Тестирование вывода информации о файле.

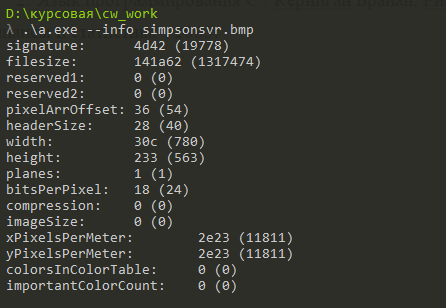


Рисунок 2 – вывод информации о файле в консоль



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.bm

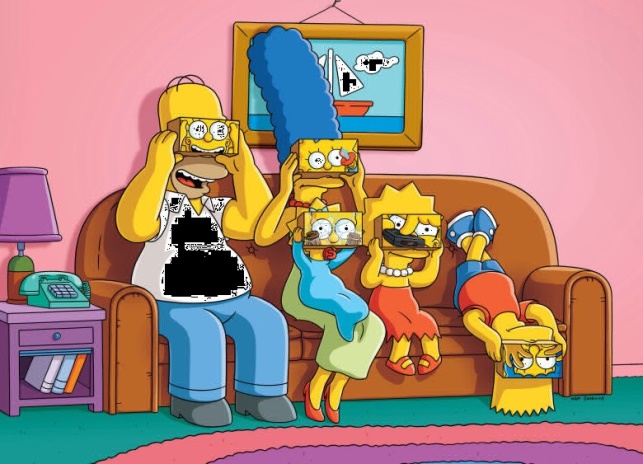


Рисунок 4 – результат работы функции color\_replace

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

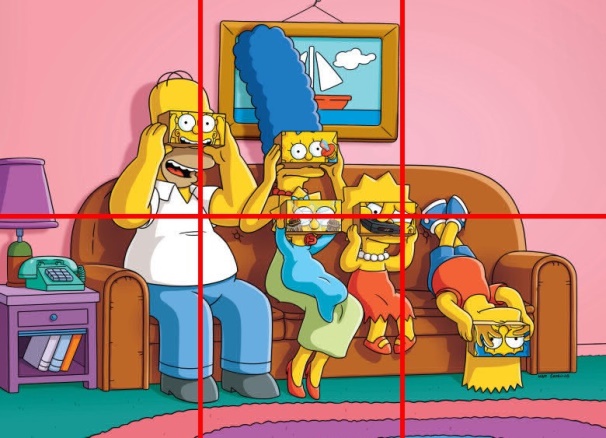


Рисунок 6 – результат работы функции split

**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("uncorrect 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("uncorrect 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 = strtof(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;

}