**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Обработка BMP файла

|  |  |  |
| --- | --- | --- |
| Студент гр. 3341 |  | Костромитин М.М. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Костромитин М.М. | | |
| Группа 3341  Вариант 2 | | |
| 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) 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`)  (2) 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  (3) 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)  Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции. | | |
| Содержание пояснительной записки:  разделы «Аннотация», «Содержание», «Введение», «Ход работы», «Пример работы программы», «Заключение», «Список использованных источников» | | |
| Предполагаемый объем пояснительной записки:  Не менее 15 страниц. | | |
| Дата выдачи задания: 18.03.2024 | | |
| Дата сдачи реферата: 27.05.2024 | | |
| Дата защиты реферата: 29.05.2024 | | |
| Студент |  | Костромитин М.М. |
| Преподаватель |  | Глазунов С.А. |

**Аннотация**

В данной курсовой работе была реализована программа, обрабатывающая PNG изображения, не имеющие сжатия. Программа проверяет тип изображения, его версию, при соответствии требованиям в дальнейшем обрабатывает его и подаёт на выход изменённую копию изображения. Взаимодействие с программой осуществляется с помощью CLI (интерфейс командной строки).

**Summary**

In this course has been created a program that processes uncompressed PNG images. The program checks the type of image, its version, if it meets the requirements, it further processes it and outputs a modified copy of the image. Interaction with the program is performed using CLI (command line interface).

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 7 |
| 1. | Ход выполнения работы | 8 |
| 1.1 | Ввод начальных данных | 8 |
| 1.2 | Функции курсовой работы | 9 |
|  | Заключение | 11 |
|  | Список использованных источников | 13 |
|  | Приложение А. Пример работы программы | 14 |
|  | Приложение B. Исходный код программы | 20 |

**введение**

Целью курсовой работы является изучение форматов файлов BMP и

PNG, а также реализация функций для работы с этими форматами файлов.

Для достижения поставленной цели требуется решить следующие задачи:

1) изучить BMP и PNG форматы изображений;

2) получить информацию об изображении: размеры, содержимое и др.;

3) обработать массив пикселей в соответствии с заданием;

4) обработать исключительные случаи;

5) сохранить итоговое изображение в новый файл.

**1. ход выполнения работы**

**1.1 Ввод начальных данных**

Обработка входных данных осуществляется в функции int main(int argc, char\* argv[]), в которой создается массив struct option longopts[] хранящий информацию о длинных флагах. Далее с помощью функции getopt\_long идет последовательная обработка входных данных: проверяется наличие и корректность введенных аргументов для флагов, а также после обработки всех флагов проверяется наличие зависимых флагов для необходимых функций, и если все необходимые флаги присутствуют, то происходит выполенние нужной функции (например для выполнения функции void rgb\_filter проверяется наличие флагов –component\_name и –component\_value). После обработки или во время обработки флагов (в случае наличия флага --input) в функции Rgb\*\* read\_bmp происходит проверка того, что файл является файлом формата BMP, а после идет считывание файла: считывается заголовок файла (в структуру BitmapFileHeader), считывается заголовок с информацией (в структуру BitmapInfoHeader), считывается и возвращается функцией массив пикселей (в Rgb\*\* arr).

**1.2 Функции курсовой работы**

Rgb\*\* read\_bmp (char file\_name[], BitmapFileHeader\* bmfh, BitmapInfoHeader\* bmif) – функция считывания BMP файла, возвращает массив пикселей переданной картинки.

int check\_if\_bmp (BitmapFileHeader\* bmfh) – по заголовку файла проверяет - является ли входной файл файлом формата BMP.

void write\_bmp(char file\_name[], Rgb \*\*arr, int H, int W, BitmapFileHeader bmfh, BitmapInfoHeader bmif) – записывает результат в файл с именем file\_name.

void color\_replace(Rgb\*\* arr, int H, int W, Rgb old\_color, Rgb new\_color) – заменяет цвет old\_color на цвет new\_color в изображении.

int color\_cmp(Rgb f, Rgb s) – возвращает 1, если структуры Rgb f и s совпадают, иначе возвращает 0.

void color\_change(Rgb\* dest, Rgb src) – записывает в структуру Rgb dest значения структуры Rgb src, то есть изменяет цвет dest на цвет src.

void rgb\_filter(Rgb\*\* arr, int H, int W, char component[], int value) – изменяет значение некоторой компоненты (red, green или blue) на значение value для всех пикселей картинки.

void draw\_line(Rgb\*\* arr, int H, int W, int x1, int x2, int y1, int y2, Rgb color) – рисует произвольную линию по алгоритму Брезенхема толщины 1.

void split(Rgb\*\* arr, int horizontal, int vertical, int H, int W, int thickness, Rgb color) – чертит горизонтальные линии толщиной thickness и цвета color в количестве horizontal, а также вертикальные линии толщиной thickness и цвета color в количестве vertical, так, чтобы картинки поделилась на равные части по вертикали и горизонтали.

int color\_validator(Rgb color) – возвращает 0, если значения всех компонент цвета удовлетворяют формату RGB (то есть больше или рано 0 и меньше либо равно 255), иначе возвращает 1.

int coord\_validator(int max, int coord0, int coord1) – возвращает 0, если координаты по x или по y произвольной линии находятся в пределах границ изображения, иначе возвращает 1.

void swap\_int(int\* a, int\* b) – помещает в переменную a значение переменной b, а в переменную b значение переменной a.

unsigned int padding(unsigned int w) – возвращает значения сдвига для изображения.

unsigned int row\_len(unsigned int w) – возвращает количество пикселей в горизонтальном ряду с учетом сдвига.

void help() – выводит на экран справку о программе.

void info(BitmapInfoHeader info\_header, BitmapFileHeader file\_header) – выводит на экран информацию о файле BMP.

int component\_validator(int component) – возвращает 1, если значения одной из компоненты RGB меньше либо равна 255 и больше либо равна 0, иначе возвращает 0.

Rgb parse\_color(char color\_string[]) – преобразует строчку вида “rrr.ggg.bbb” в структуру Rgb и возвращает ее.

void print\_file\_header(BitmapFileHeader header) – выводит на экран информацию из заголовка файла.

void print\_info\_header(BitmapInfoHeader header) – выводит на экран информацию из заголовка с информацией.

**заключение**

В ходе выполнения курсовой работы были изучены формата файлов изображений BMP, а также разработаны функции для работы с ними. Подробное изучение особенностей и структуры данных данных форматов позволило успешно реализовать функционал, включающий получение информации об изображениях, обработку массива пикселей в соответствии с поставленными задачами, учет и обработку исключительных случаев, а также сохранение результирующего изображения в новом файле. А также была разобрана задача обработки флагов CLI с помощью функции getopt\_long из библиотеки getopt.h.

**список использованных источников**

1.https://se.moevm.info/lib/exe/fetch.php/courses:programming:programming\_cw\_metoda\_2nd\_course\_last\_ver.pdf.pdf - методические материалы для написания курсовой работы

2.https://www.r5.org/files/books/computers/languages/c/kr/Brian\_Kernighan\_Dennis\_Ritchie-The\_C\_Programming\_Language-RU.pdf – язык программирования Си

3.https://www.gnu.org/software/libc/manual/html\_node/Getopt.html - принцип работы функции getopt\_long и примеры обработки флагов с помощью неё.

4. https://ru.wikibooks.org/wiki/Реализации\_алгоритмов/Алгоритм\_Брезенхэма - алгоритм Брезенхэма

**приложение А**

**Исходный код программы**

Название файла: main.c

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <getopt.h>

#include <unistd.h>

#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)

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

int check\_if\_bmp(BitmapFileHeader\* bmfh);

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

void color\_replace(Rgb\*\* arr, int H, int W, Rgb old\_color, Rgb new\_color);

int color\_cmp(Rgb f, Rgb s);

void color\_change(Rgb\* dest, Rgb src);

void rgb\_filter(Rgb\*\* arr, int H, int W, char component[], int value);

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

void split(Rgb\*\* arr, int horizontal, int vertical, int H, int W, int thickness, Rgb color);

int color\_validator(Rgb color);

int coord\_validator(int max, int coord0, int coord1);

void swap\_int(int\* a, int\* b);

unsigned int padding(unsigned int w);

unsigned int row\_len(unsigned int w);

Rgb parse\_color(char color\_string[]);

void help();

void info(BitmapInfoHeader info\_header, BitmapFileHeader file\_header);

void print\_file\_header(BitmapFileHeader header);

void print\_info\_header(BitmapInfoHeader header);

int

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

printf("Course work for option 4.2, created by Kostromitin Mikhail\n");

BitmapFileHeader file\_header;

BitmapInfoHeader file\_info;

int is\_replace = -1;

int is\_filter = -1;

int is\_split = -1;

struct option longopts[] = {

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

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

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

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

{"color\_replace", no\_argument, &is\_replace, 'r'},

{"old\_color", required\_argument, NULL, 'a'},

{"new\_color", required\_argument, NULL, 'w'},

{"rgbfilter", no\_argument, &is\_filter, 'f'},

{"component\_name", required\_argument, NULL, 'n'},

{"component\_value", required\_argument, NULL, 'v'},

{"split", no\_argument, &is\_split, 's'},

{"number\_x", required\_argument, NULL, 'x'},

{"number\_y", required\_argument, NULL, 'y'},

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

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

{0, 0, 0, 0}

};

int gpt;

char shortopts[] = "ho:";

int\* indexptr = (int\*)malloc(sizeof(int));

char file\_name\_new[256];

file\_name\_new[0] = '\0';

char file\_name\_old[100];

strncpy(file\_name\_old, argv[argc - 1], strlen(argv[argc - 1]));

char colors\_replace\_old[12];

char colors\_replace\_new[12];

Rgb replace\_old;

Rgb replace\_new;

char component\_name[6];

int component\_value;

int x;

int y;

int th;

char split\_color[12];

Rgb clr;

int is\_read = 0;

Rgb\*\* arr;

while ((gpt = getopt\_long(argc, argv, shortopts, longopts, indexptr)) != -1){

switch(gpt){

case 'h':

help();

exit(0);

case 'o':

strncpy(file\_name\_new, optarg, strlen(optarg));

file\_name\_new[strlen(optarg)] = '\0';

break;

case 'p':

strncpy(file\_name\_old, optarg, strlen(optarg));

file\_name\_old[strlen(optarg)] = '\0';

arr = read\_bmp(file\_name\_old, &file\_header, &file\_info);

is\_read = 1;

break;

case 'i':

info(file\_info, file\_header);

exit(0);

case 'a':

strncpy(colors\_replace\_old, optarg, strlen(optarg));

colors\_replace\_old[strlen(optarg)] = '\0';

replace\_old = parse\_color(colors\_replace\_old);

break;

case 'w':

strncpy(colors\_replace\_new, optarg, strlen(optarg));

colors\_replace\_new[strlen(optarg)] = '\0';

replace\_new = parse\_color(colors\_replace\_new);

break;

case 'n':

strncpy(component\_name, optarg, strlen(optarg));

component\_name[strlen(optarg)] = '\0';

break;

case 'v':

component\_value = atoi(optarg);

break;

case 'x':

x = atoi(optarg);

break;

case 'y':

y = atoi(optarg);

break;

case 't':

th = atoi(optarg);

break;

case 'c':

strncpy(split\_color, optarg, strlen(optarg));

split\_color[strlen(optarg)] = '\0';

clr = parse\_color(split\_color);

break;

default:

break;

}

}

if (!is\_read) {

arr = read\_bmp(file\_name\_old, &file\_header, &file\_info);

}

int H, W;

H = file\_info.height;

W = file\_info.width;

if (is\_replace != -1){

color\_replace(arr, H, W, replace\_old, replace\_new);

}

if (is\_filter != -1){

rgb\_filter(arr, H, W, component\_name, component\_value);

}

if (is\_split != -1){

split(arr, x, y, H, W, th, clr);

}

if (file\_name\_new[0] == '\0'){

write\_bmp(file\_name\_old, arr, H, W, file\_header, file\_info);

} else{

write\_bmp(file\_name\_new, arr, H, W, file\_header, file\_info);

}

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

free(arr[i]);

free(arr);

free(indexptr);

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);

if (!check\_if\_bmp(bmfh)){

printf("Error 49: invalid file format\n");

exit(49);

}

unsigned int H = bmif->height;

unsigned int W = bmif->width;

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

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

arr[i] = malloc(row\_len(W));

fread(arr[i], 1, row\_len(W), f);

}

fclose(f);

return arr;

}

int

check\_if\_bmp(BitmapFileHeader\* bmfh){

if (bmfh->signature != 19778 || bmfh->reserved1 != 0 || bmfh->reserved2 != 0){

return 0;

}

return 1;

}

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[i], 1, row\_len(W), ff);

}

fclose(ff);

}

void

color\_replace(Rgb\*\* arr, int H, int W, Rgb old\_color, Rgb new\_color){

if (!(color\_validator(old\_color)) || !(color\_validator(new\_color))){

printf("Error 41: invalid color parameters");

exit(41);

}

for (int y = 0; y < H; y++){

for (int x = 0; x < W; x++){

if (color\_cmp(arr[y][x], old\_color)){

color\_change(&arr[y][x], new\_color);

}

}

}

}

int

color\_cmp(Rgb f, Rgb s){

if (f.g == s.g && f.r == s.r && f.b == s.b){

return 1;

}

return 0;

}

void

color\_change(Rgb\* dest, Rgb src){

(\*dest).b = src.b;

(\*dest).g = src.g;

(\*dest).r = src.r;

}

void

rgb\_filter(Rgb\*\* arr, int H, int W, char component[], int value){

if (value < 0 || value > 255){

printf("Error 42: invalid component value");

exit(42);

}

if (!strcmp(component, "red")){

for (int y = 0; y < H; y++){

for (int x = 0; x < W; x++){

arr[y][x].r = value;

}

}

} else if (!strcmp(component, "green")){

for (int y = 0; y < H; y++){

for (int x = 0; x < W; x++){

arr[y][x].g = value;

}

}

} else if (!strcmp(component, "blue")){

for (int y = 0; y < H; y++){

for (int x = 0; x < W; x++){

arr[y][x].b = value;

}

}

} else {

printf("Error 42: invalid component");

exit(42);

}

}

void

draw\_line(Rgb\*\* arr, int H, int W, int x1, int x2, int y1, int y2, Rgb color) {

if (!coord\_validator(W, x1, x2) || !coord\_validator(H, y1, y2) || !(color\_validator(color))){

printf("Error 43: invalid parameters for a line");

exit(43);

}

const int deltaX = abs(x2 - x1);

const int deltaY = abs(y2 - y1);

const int signX = x1 < x2 ? 1 : -1;

const int signY = y1 < y2 ? 1 : -1;

int error = deltaX - deltaY;

arr[y2- 1][x2 - 1] = color;

while(x1 != x2 || y1 != y2){

arr[y1][x1] = color;

int error2 = error \* 2;

if(error2 > -deltaY){

error -= deltaY;

x1 += signX;

}

if(error2 < deltaX){

error += deltaX;

y1 += signY;

}

}

}

void

split(Rgb\*\* arr, int horizontal, int vertical, int H, int W, int thickness, Rgb color){

if (horizontal < 2 || vertical < 2 || thickness <= 0){

printf("Error 44: invalid split parameters");

exit(44);

}

int y = 0;

for (int i = 1; i < horizontal; i++){

y += H / horizontal;

int y\_for\_thick = y;

for (int thick = 0; thick < thickness; thick++){

if (thick % 2 == 0){

y\_for\_thick -= thick;

draw\_line(arr, H, W, 0, W, y\_for\_thick, y\_for\_thick, color);

} else {

y\_for\_thick += thick;

draw\_line(arr, H, W, 0, W, y\_for\_thick, y\_for\_thick, color);

}

}

}

int x = 0;

for (int i = 1; i < vertical; i++){

x += W / vertical;

int x\_for\_thick = x;

for (int thick = 0; thick < thickness; thick++){

if (thick % 2 == 0){

x\_for\_thick -= thick;

draw\_line(arr, H, W, x\_for\_thick, x\_for\_thick, 0, H, color);

} else {

x\_for\_thick += thick;

draw\_line(arr, H, W, x\_for\_thick, x\_for\_thick, 0, H, color);

}

}

}

}

int

color\_validator(Rgb color){

if (color.r < 0 || color.r > 255 || color.g < 0 || color.g > 255 || color.b < 0 || color.b > 255){

return 0;

}

return 1;

}

int

coord\_validator(int max, int coord0, int coord1){

if (coord0 < 0 || coord0 > max || coord1 < 0 || coord1 > max){

return 0;

}

return 1;

}

void

swap\_int(int\* a, int\* b){

int tmp = \*a;

\*a = \*b;

\*b = tmp;

}

unsigned int

padding(unsigned int w){

unsigned int padding = (w \* sizeof(Rgb)) % 4;

if (padding) padding = 4 - padding;

return padding;

}

unsigned int

row\_len(unsigned int w){

return w \* sizeof(Rgb) + padding(w);

}

void

help(){

printf("Программа для обработки файла формата bmp.\n");

printf("\t-p / --input - имя входного файла.\n");

printf("\t-i / --info - флаг для печати подробной информации о bmp-файле.\n");

printf("\t-o / --output - переопределение названия конечного bmp-файла.\n");

printf("Заменяет все пиксели одного заданного цвета на другой цвет. Флаг для выполнения данной операции: `--color\_replace`. Функционал определяется:\n");

printf("\t--old\_color - цвет, который необходимо заменить: строчка вида \"rrr.ggg.bbb\", где rrr/ggg/bbb – числа, задающие цветовую компоненту.\n");

printf("\t--new\_color - новый цвет: строчка вида \"rrr.ggg.bbb\", где rrr/ggg/bbb – числа, задающие цветовую компоненту.\n");

printf("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:\n");

printf("\t--component\_name - имя компоненты, которую нужно заменить. Возможные значения `red`, `green` и `blue`\n");

printf("\t--component\_value - значение, на которое требуется заменить значения компоненты. Принимает значение в виде числа от 0 до 255.\n");

printf("Разделяет изображение на N\*M частей. Флаг для выполнения данной операции: `--split`. Реализация: провести линии заданной толщины. Функционал определяется:\n");

printf("\t--number\_x - количество частей по “оси” Y. На вход принимает число больше 1.\n");

printf("\t--number\_y - количество частей по “оси” X. На вход принимает число больше 1.\n");

printf("\t--thickness - толщина линии. На вход принимает число больше 0.\n");

printf("\t--color - цвет линии, цвет задаётся строкой \"rrr.ggg.bbb\", где rrr/ggg/bbb – числа, задающие цветовую компоненту.\n");

printf("При последовательном вводе операция, они имеют следующий приоритет выполнения: color\_replace, rgbfilter, split.\n");

}

void

info(BitmapInfoHeader info\_header, BitmapFileHeader file\_header){

print\_file\_header(file\_header);

print\_info\_header(info\_header);

}

int

component\_validator(int component){

if (component > 255 || component < 0){

return 1;

}

return 0;

}

Rgb

parse\_color(char color\_string[]){

Rgb color;

char\* tmp;

int curr\_color;

tmp = strtok(color\_string, ".");

curr\_color = atoi(tmp);

if (component\_validator(curr\_color)){

printf("Error 45: Invalid component for color replacement");

exit(45);

}

color.r = curr\_color;

tmp = strtok(NULL, ".");

curr\_color = atoi(tmp);

if (component\_validator(curr\_color)){

printf("Error 45: Invalid component for color replacement");

exit(45);

}

color.g = curr\_color;

tmp = strtok(NULL, ".");

curr\_color = atoi(tmp);

if (component\_validator(curr\_color)){

printf("Error 45: Invalid component for color replacement");

exit(45);

}

color.b = curr\_color;

return color;

}

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);

}

**приложение б**

**Тестирование**

Фото для обработки – RGB фильтр

./cw --input My-boy-be-ballin.bmp --rgbfilter --component\_name blue --component\_value 0



Результат работы программы:



Фото для обработки – замена одного цвета на другой

./cw --input Plant.bmp --color\_replace --old\_color 000.000.000 --new\_color 255.255.255



Результат работы программы:

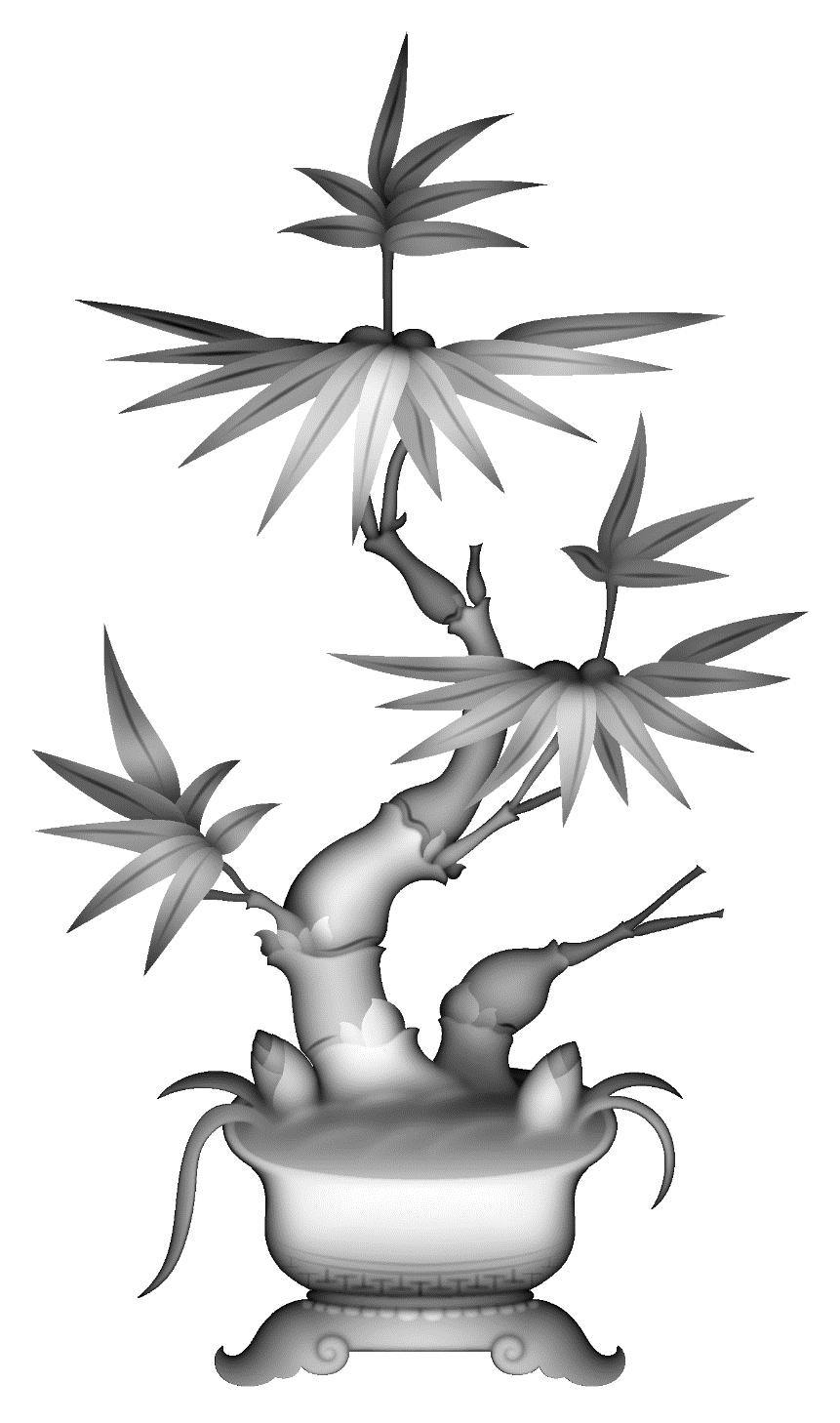


Фото для обработки – разделение картинки на равные части

./сw --input Obama.bmp --split --number\_x 10 --number\_y 10 --thickness 20 --color 0.0.0



Результат работы программы:

