# МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

# «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

**Кафедра МО ЭВМ**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Программирование»**

**Тема: Обработка изображений на языке С**

|  |  |
| --- | --- |
| Студент гр. 1384 | Усачева Д.В. |
| Преподаватель | Жангиров Т.Р. |

Санкт-Петербург 2022

# ЗАДАНИЕ

# НА КУРСОВУЮ РАБОТУ

Студентка Усачева Д.В.

Группа 1384

Тема работы: Обработка изображений на языке С

Исходные данные:

Программе на вход подается одно или несколько (в зависимости от задания) изображений в формате BMP, 24 бита на цвет, без сжатия. Нужно реализовать функционал по обработке этого (этих) файлов.

Содержание пояснительной записки:

“Содержание”, “Введение”, “Описание структур”, “Ввод”, “Удаление повторно встречающихся предложений”, “Функции”, “Основная программа”, “Примеры работы программы”, “Заключение”

Предполагаемый объем пояснительной записки:

Не менее 25 страниц.

Дата выдачи задания: 22.03.2022

Дата сдачи реферата: 7.05.2022

Дата защиты реферата: 9.05.2022

|  |  |
| --- | --- |
| Студент гр. 1384 | Усачева Д.В. |
| Преподаватель | Жангиров Т.Р. |

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ](#_TOC_250021) 4

1. ЗАДАНИЕ 1 6
2. ЗАДАНИЕ 2 6
3. ЗАДАНИЕ 3 6
4. ДРУГИЕ НЕОБХОДИМЫЕ ФУНКЦИИ 7
5. [ОСНОВНАЯ ПРОГРАММА](#_TOC_250016) 8
6. [ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ 9](#_TOC_250005)

[ЗАКЛЮЧЕНИЕ 14](#_TOC_250000)

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ. 15

ПРИЛОЖЕНИЕ A. ИСХОДНЫЙ КОД ПРОГРАММЫ 16

# ВВЕДЕНИЕ

Программа должна реализовывать весь следующий функционал по обработке bmp-файла

# Общие сведения

* 24 бита на цвет
* без сжатия
* файл всегда соответствует формату BMP (но стоит помнить, что версий у формата несколько)
* обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями.
* обратите внимание на порядок записи пикселей
* все поля стандартных BMP заголовков в выходном файле должны иметь те же значения что и во входном (разумеется кроме тех, которые должны быть изменены).

Программа должна реализовывать следующий функционал по обработке bmp- файла:

* (1) Заменяет все пиксели одного заданного цвета на другой цвет. Функционал определяется:
  + Цвет, который требуется заменить
  + Цвет на который требуется заменить
* (2) Фильтр rgb-компонент. Этот инструмент должен позволять для всего изображения либо установить в 0 либо установить в 255 значение заданной компоненты. Функционал определяется
  + Какую компоненту требуется изменить
  + В какой значение ее требуется изменить
* (3) Разделяет изображение на N\*M частей. Реализация: либо провести линии заданной толщины, тем самым разделив изображение либо сохранение каждой части в отдельный файл. -- по желанию студента (можно и оба варианта). Функционал определяется:
  + Количество частей по “оси” Y
  + Количество частей по “оси” X
  + Толщина линии
  + Цвет линии
  + Либо путь куда сохранить кусочки

1. **ЗАДАНИЕ 1**

Функция color\_replacement осуществляет замену старого цвета на новый.

Функционал определяется двумя цветами, которые пользователь вводит через соответственные аргументы.

Программа ищет все пиксели изображения, удовлетворяющие старому цвету, затем заменяет и на новый введенный пользователем цвет.

Если пользователь не ввёл какой-либо цвет, то выводится подсказка.

# ЗАДАНИЕ 2

Функция filter\_rgb\_component изменяет один из введенных пользователем через аргумент rgb компоненты, диапазон изменений – (0 to 255).

Программа во всех пикселях изображения меняет соответствующую компоненту на новую и записывает обработанное изображение в новый файл.

В случае некорректного ввода пользователем программа выведет сообщение об ошибке.

# ЗАДАНИЕ 3

Функция cut\_on\_lines делит изображения на N x M частей, рисуя линии. Пользователю нужно ввести количество частей по Ох, по Оу, толщину

линии и цвет через соответствующие аргументы, которые указаны в описании программы(help).

Функция correct позволяет определить корректность толщины прямой, если толщина-четное число, то для симметричного разделения изображения к толщине прибавляется 1.

При некорректном вводе ключей и значений пользователем программа выведет сообщения об ошибках и подсказки.

1. **ДРУГИЕ НЕОБХОДИМЫЕ ФУНКЦИИ**

# Функция read\_image

Для считывания изображения реализована функция. Первым делом осуществляется попытка открыть изображение на чтение в байтовом режиме. Если же не получается открыть изображение, то выводится соответствующее сообщение. Затем происходит считывание заголовка файла. Дальше отдельно считываем headerSize и сохраняем это значение в соответствующую переменную. После чего происходит дальнейшее считывание headerSize байт. Такая реализация позволяет добавить поддержку нескольких версий BMP файлов. Функция возвращает объект структуры BMP и выводит сообщение об успешном считывании изображения.

# Функция write\_image

Функция записывает обработанное изображение в новый файл, имя которого пользователь должен указать в конце ряда аргументов при их вводе. В случае некорректного ввода программа выведет сообщения об ошибке.

# Функция checking\_values

Данная функция проверяет на корректность введенных данных пользователем. В случае ошибки программа выведет подсказки.

# 4.4Функция print\_help

Данная функция содержит справку к программе, описание задач и аргументов, необходимых для их вызова ключей.

**4.5Функции printInfoHeader и printFileHeader**

Данные функции позволяют распечатать на экран подробную информации о bmp-файле.

# ОСНОВНАЯ ПРОГРАММА

В функции main.c реализовано считывания аргументов и также проверка на ошибки.

Программа определяет, какую операцию и с какими параметрами нужно выполнить, с помощью утилиты getopt с подключенным соответствующим заголовком getopt.h. Для этого была реализована структура, хранящая аргументы команды в удобной форме, так как задание предполагает ввод и длинных опций, то для этих целей создается массив longOpts для хранения информации о длинных опциях, поддерживаемых программой. После добавления в конец структуры соответствующие короткие опции, можно обрабатывать длинные без добавочного программного кода. Строка с опциями (optstring) состоит из одного символа для каждого аргумента командной строки. После опций, которые имеют параметры, ставится двоеточие. Для избежания неправильного функционирования программы была использована переменная opt, установленная в 0. Как только getopt\_long() вернет -1, что означает, что остальные аргументы команды ключами не являются, происходит вызов основных функций.

1. ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

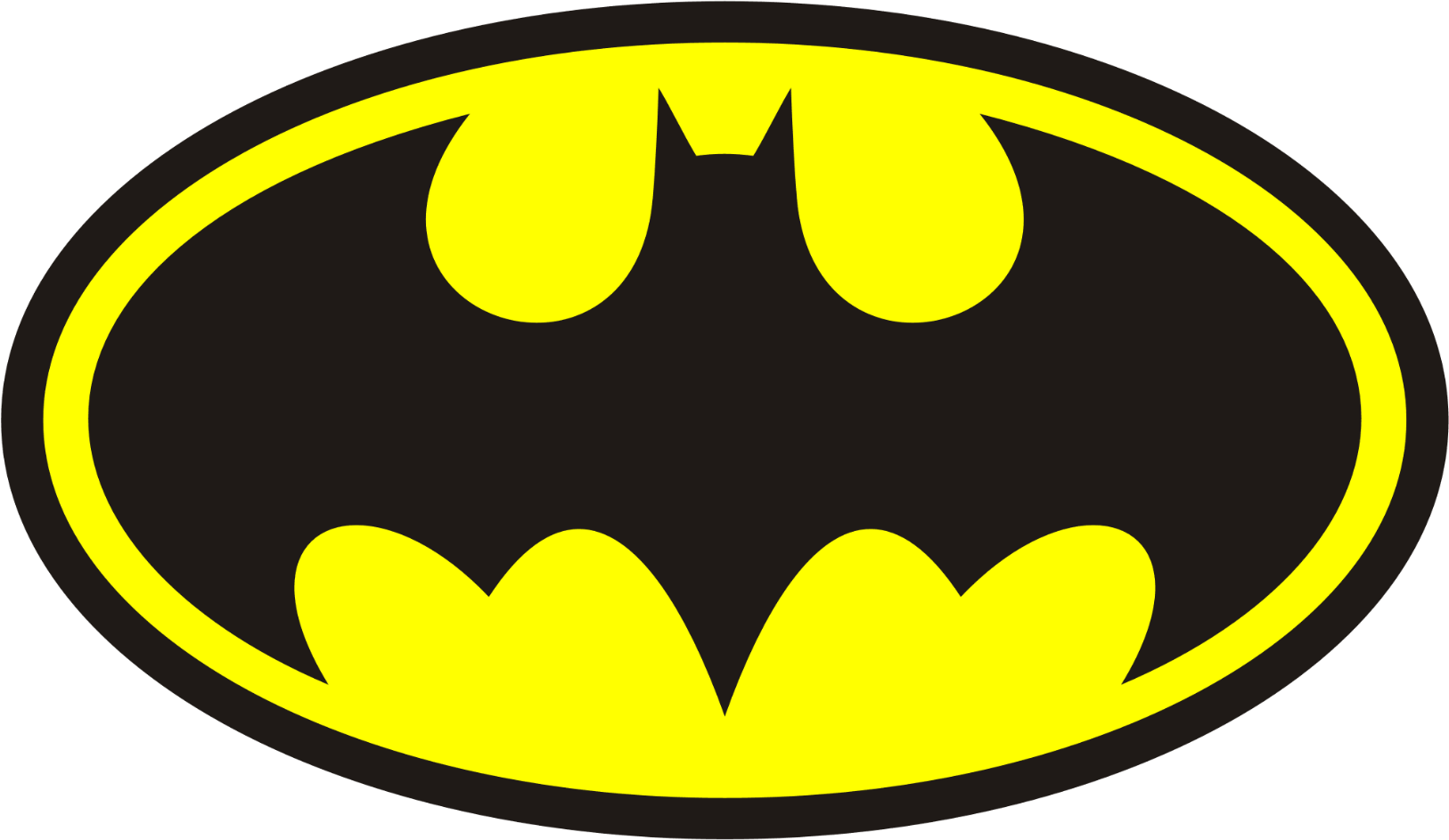
**Пример 1**: 

Рисунок 1.1 – Пример входного изображения для запроса 1

* Команда запуска: image.bmp -o 0,0,0 -n 255,0,2с out.bmp

Результат: Введите аргумент, где ВСЕ ТРИ компонента rgb корректны (0 - 255 без пробелов между значениями)

Используйте ключ --help для вызова справки

* Команда запуска: image.bmp -o 0,0,0 -n 255,0,2 out.bmp

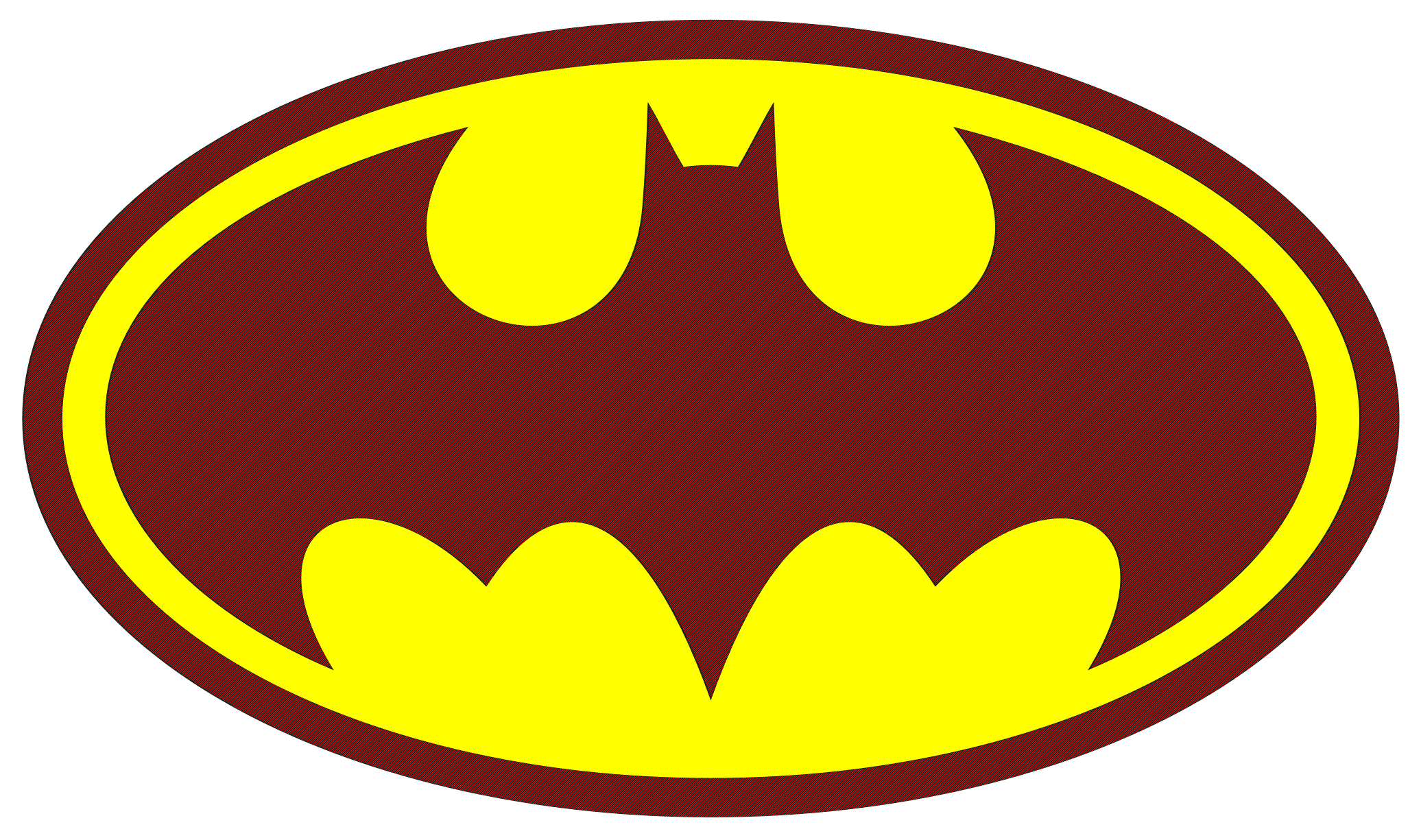


Рисунок 1.2 – Результат выполнения первого запроса

**Пример 2:**

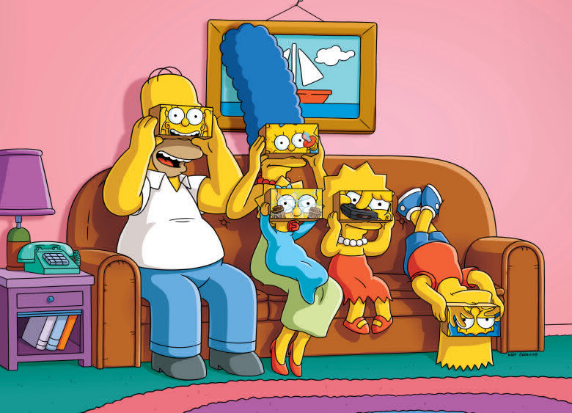


Рисунок 2.1 – Пример входного изображения для запроса 2

* Команда запуска: simpsonsvr.bmp -g 255a out.bmp

Результат: Неверный ввод, аргумент '-g/--green' должен иметь числовое значение. Попробуйте еще раз

Ипользуйте ключ --help для вызова справки

* Команда запуска: simpsonsvr.bmp -g 255 out.bmp



Рисунок 2.2 – Результат выполнения второго запроса

**Пример 3:**

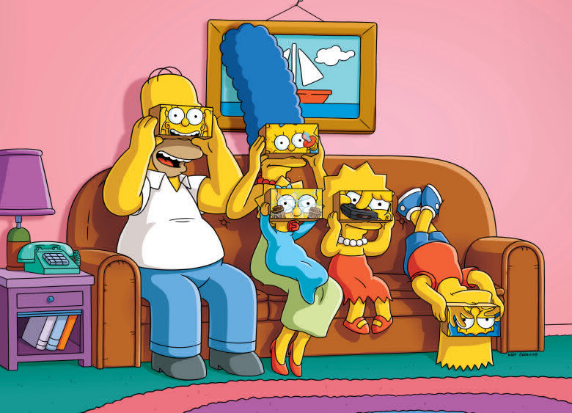


Рисунок 3.1 – Пример входного изображения для запроса 3

* Команда запуска: simpsonsvr.bmp -x 3 -y 55 -t 2 -c 255,0b,0 out.bmp

Результат: Введите коректные данные для ключа -C(--colors)

Используйте ключ --help для вызова справки

* Команда запуска: simpsonsvr.bmp -x 35 -y 55 -t 2 -c 255,0,0 out.bmp

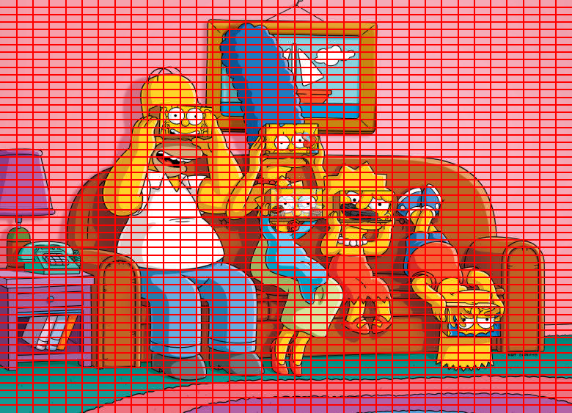


Рисунок 2.2 – Результат выполнения третьего запроса

**Пример 4:**

* Команда запуска: image.bmp –i

Результат:

signature: 4d42 (19778)

filesize: 26bf0e (2539278)

reserved1: 0 (0)

reserved2: 0 (0)

pixelArrOffset: 436 (1078)

headerSize: 28 (40)

width: 818 (2072)

height: 4c9 (1225)

planes: 1 (1)

bitsPerPixel: 8 (8)

compression: 0 (0)

imageSize: 0 (0)

xPixelsPerMeter: 1274 (4724)

yPixelsPerMeter: 1274 (4724)

colorsInColorTable: 100 (256)

importantColorCount: 100 (256)

**Пример 5:**

* Команда запуска: –h

Результат:

[DESCRIPTIONS]

Программа поддерживает интерфейс командной строки и работает только с файлами BMP версии 3

Bmp файлы с таблицей цветов не поддерживаются

Программа поддерживает только файлы с глубиной 24 пикселя на бит

Файл не должен быть сжат

[FUNCTIONS]

(1) - заменить цвет

Заменяет все пиксели одного заданного цвета на другой цвет

Обязательные аргументы:

-o/--oldcolors

-n/--newcolors

(2) - Фильтр rgb-компонент

Этот инструмент должен позволять для всего изображения либо установить в 0 либо установить в 255 значение заданной компоненты.

Обязательные аргументы:

-r/--red

-g/--green

-b/--blue

(3) - Разделяет изображение на N\*M частей

Проводит линии заданной толщины, тем самым разделив изображение

Обязательные аргументы:

-y/--ylines

-x/--xlines

-t/--thickness

-c/--linecolor

[KEYS]

-o/--oldcolors/-n/--newcolors <red>,<green>,<blue> цвет в формате RGB

-r/--red/-g/--green/-b/--blue <value> устанавливает RGB компонент <от 0 до 255>

-y/--ylines <value> Количество частей по оси 'Y'+1

-x/--xlines <value> Количество частей по оси 'X'+1

-t/--thickness <value> толщина линии

-c/--linecolor <red>,<green>,<blue> цвет линии

-h/-?/--help справка для пользователя

-i/--info информация об изображении

ЗАКЛЮЧЕНИЕ

В результате выполнение курсовой работы было создано приложение подобное linux-утилитам, для обработки изображений в формате BMP. Программа была протестирована, результат работы программы совпал с заданными условиями и рекомендациями.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ.

1. Керниган Б. и Ритчи Д. Язык программирования Си. М.: Вильямс, 1978 288c.
2. Программирование: учеб.-метод. пособие. СПб.: Изд-во СПбГЭТУ

«ЛЭТИ», 2017.

# ПРИЛОЖЕНИЕ A.

# ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

#include <stdio.h>

#include <dirent.h>

#include <string.h>

#include <sys/types.h>

#include <stdlib.h>

#include <unistd.h>

#include <getopt.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)

typedef struct{

BitmapFileHeader bfh;

BitmapInfoHeader bih;

Rgb \*\*arr;

int error\_signal;

int file\_error;

} BMP;

typedef struct{

char\*\* tval\_new;

char\*\* tval\_old;

int error\_flag;

} tokens;

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 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 DESCRIPTIONS(){

printf("\n[DESCRIPTIONS]\n");

printf("\n Программа поддерживает интерфейс командной строки и работает только с файлами BMP версии 3");

printf("\n Bmp файлы с таблицей цветов не поддерживаются");

printf("\n Программа поддерживает только файлы с глубиной 24 пикселя на бит");

printf("\n Файл не должен быть сжат\n");

}

void printHelp(){

printf("\n[FUNCTIONS]\n");

printf(" (1) - заменить цвет\n");

printf(" Заменяет все пиксели одного заданного цвета на другой цвет \n");

printf(" Обязательные аргументы:\n");

printf(" -o/--oldcolors\n");

printf(" -n/--newcolors\n");

printf("\n (2) - Фильтр rgb-компонент \n");

printf(" Этот инструмент должен позволять для всего изображения либо установить в 0 либо установить в 255 значение заданной компоненты.\n");

printf(" Обязательные аргументы:\n");

printf(" -r/--red\n");

printf(" -g/--green\n");

printf(" -b/--blue\n");

printf("\n (3) - Разделяет изображение на N\*M частей\n");

printf(" Проводит линии заданной толщины, тем самым разделив изображение\n");

printf(" Обязательные аргументы:\n");

printf(" -y/--ylines\n");

printf(" -x/--xlines\n");

printf(" -t/--thickness\n");

printf(" -c/--linecolor\n\n");

printf("[KEYS]\n");

printf(" -o/--oldcolors/-n/--newcolors <red>,<green>,<blue> цвет в формате RGB\n");

printf(" -r/--red/-g/--green/-b/--blue <value> устанавливает RGB компонент <от 0 до 255>\n");

printf(" -y/--ylines <value> Количество частей по оси 'Y'+1\n");

printf(" -x/--xlines <value> Количество частей по оси 'X'+1\n");

printf(" -t/--thickness <value> толщина линии\n");

printf(" -c/--linecolor <red>,<green>,<blue> цвет линии\n");

printf(" -h/-?/--help справка для пользователя\n");

printf(" -i/--info информация об изображении\n");

}

BMP read\_image(char\* path){

BMP image;

BitmapFileHeader bfh;

BitmapInfoHeader bih;

FILE\* img = fopen(path, "rb");

if( (!img) || (!strstr(path, ".bmp")) ){

image.error\_signal = 1;

return image;

}

fread(&bfh, 1, sizeof(BitmapFileHeader), img);

fread(&bih, 1, sizeof(BitmapInfoHeader), img);

if(bih.headerSize != 40){

printf("FILE ERROR\n");

image.file\_error = 1;

return image;

}

unsigned int H = bih.height;

unsigned int W = bih.width;

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

int offset = (W \* sizeof(Rgb))%4;

offset = offset ? 4-offset : 0;

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

arr[i] = malloc(W \* sizeof(Rgb) + offset);

fread(arr[i], 1, W \* sizeof(Rgb) + offset, img);

}

fclose(img);

image.bfh = bfh;

image.bih = bih;

image.arr = arr;

image.error\_signal = 0;

return image;

}

void write\_image(char \*path, BMP data){

BitmapFileHeader bfh = data.bfh;

BitmapInfoHeader bih = data.bih;

Rgb \*\*arr = data.arr;

unsigned int H = bih.height;

unsigned int W = bih.width;

FILE \*f\_out = fopen(path, "wb");

fwrite(&bfh, 1, sizeof(BitmapFileHeader), f\_out);

fwrite(&bih, 1, sizeof(BitmapInfoHeader), f\_out);

int offset = (W \* sizeof(Rgb))%4;

offset = offset ? 4-offset : 0;

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

fwrite(arr[i], 1, W \* sizeof(Rgb) + offset, f\_out);

}

fclose(f\_out);

}

tokens checking\_values(char\*\* argv, int argc){

tokens tmp;

int val\_old;

int val\_new;

int val\_c;

char\*\* tval\_new = calloc(3, sizeof(char\*));

char\*\* tval\_old = calloc(3, sizeof(char\*));

int n = 0;

int o = 0;

int ok\_N = 1;

int ok\_O = 1;

char flag []="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!?/.,|@#$^&\*()\_+-=;:~`][{}";

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

if(ok\_O == 1 && (!strcmp(argv[i],"-o") || !strcmp(argv[i],"--oldcolors"))){

ok\_O = 0;

val\_old = i+1;

char\* token = strtok(argv[val\_old], ",");

while (token){

tval\_old[o] = token;

if (!(strpbrk (flag,tval\_old[o])==NULL)){

printf("\nВведите корректные данные для ключа-o(--oldcolors)\n");

printf("Используйте ключ --help для вызова справки\n\n");

tmp.error\_flag = 1;

return tmp;

}

o++;

token = strtok(NULL, ",");

}

if(o!=3){

tmp.error\_flag = 1;

printf("\nВведите аргумент, где все ТРИ компонента rgb корректны (0 - 255 без пробелов между значениями)\n");

printf("Используйте ключ --help для вызова справки\n\n");

return tmp;

}

}

if(ok\_N == 1 && (!strcmp(argv[i],"-n") || !strcmp(argv[i],"--newcolors"))){

ok\_N = 0;

val\_new = i+1;

char\* token = strtok(argv[val\_new], ",");

while (token){

tval\_new[n] = token;

if (!(strpbrk (flag,tval\_new[n])==NULL)){

printf("\nВведите корректные данные для ключа -n(--newcolors)\n");

printf("Используйте ключ --help для вызова справки\n\n");

tmp.error\_flag = 1;

return tmp;

}

n++;

token = strtok(NULL, ",");

}

if(o!=3 || n!=3){

tmp.error\_flag = 1;

printf("\nВведите аргумент, где все ТРИ компонента rgb корректны (0 - 255 без пробелов между значениями)\n");

printf("Используйте ключ --help для вызова справки\n\n");

return tmp;

}

}

}

tmp.tval\_new = tval\_new;

tmp.tval\_old = tval\_old;

tmp.error\_flag = 0;

return tmp;

}

int color\_replacement(tokens tmp, BMP data){

char\*\* tval\_new = tmp.tval\_new;

char\*\* tval\_old = tmp.tval\_old;

int count = 0;

if(atoi(tval\_old[0]) >= 255 || atoi(tval\_old[1]) >= 255 || atoi(tval\_old[2]) >= 255 || atoi(tval\_new[0]) >= 255 || atoi(tval\_new[1]) >= 255 || atoi(tval\_new[2]) >= 255){

if (atoi(tval\_old[0]) <= 0 || atoi(tval\_old[1]) <= 0 || atoi(tval\_old[2]) <= 0 || atoi(tval\_new[0]) <= 0 || atoi(tval\_new[1]) <= 0 || atoi(tval\_new[2]) <= 0){

printf("\nВведите аргумент, где ВСЕ ТРИ компонента rgb корректны (0 - 255 без пробелов между значениями)\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

}

Rgb \*\*arr = data.arr;

unsigned int H = data.bih.height;

unsigned int W = data.bih.width;

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

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

if( (arr[i][j].r == atoi(tval\_old[0])) && (arr[i][j].g == atoi(tval\_old[1])) && (arr[i][j].b == atoi(tval\_old[2])) ){

arr[i][j].r = atoi(tval\_new[0]);

arr[i][j].g = atoi(tval\_new[1]);

arr[i][j].b = atoi(tval\_new[2]);

}

}

}

return 1;

}

int filter\_rgb\_component(char\*\* rgb\_value, BMP data){

char\* component = rgb\_value[0];

char\* value = rgb\_value[1];

Rgb \*\*arr = data.arr;

unsigned int H = data.bih.height;

unsigned int W = data.bih.width;

if(strcmp(component, "-r") == 0){

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

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

arr[i][j].r = atoi(value);

}

}

return 1;

}

if(strcmp(component, "-g") == 0){

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

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

arr[i][j].g = atoi(value);

}

}

return 1;

}

if(strcmp(component, "-b") == 0){

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

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

arr[i][j].b = atoi(value);

}

}

return 1;

}

printf("\nВведите данные корректно ( -//-.bmp -r 255 -//-.bmp)\n");

printf("Используйте ключ --help для вызова справки\n");

return 0;

}

int correct(int x){

if(x%2 == 0) return 1;

return 0;

}

int cut\_on\_lines(char \*\*array, BMP data){

char flag []="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!?/.,|@#$^&\*()\_+-=;:~`][{}";

char \*\*rgb = calloc(3, sizeof(char\*));

char\* token = strtok(array[3], ",");

int j = 0;

while (token){

rgb[j] = token;

j++;

token = strtok(NULL, ",");

}

if(j!= 3){

printf("\nВведите аргумент, где ВСЕ ТРИ компонента rgb корректны (0 - 255 без пробелов между значениями). Например:: -C/--colors value1,value2,value3\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

for(int i = 0; i < 3; i++){

if (!(strpbrk (flag,rgb[i])==NULL)){

printf("\nВведите коректные данные для ключа -C(--colors)\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

}

Rgb \*\*arr = data.arr;

unsigned int H = data.bih.height;

unsigned int W = data.bih.width;

int N = atoi(array[0]);

int M = atoi(array[1]);

int T = atoi(array[2]);

for(int n = 1; n < N; n++){

for(int n1 = ((n\*H)/N) - (T/2-correct(T)); n1 <= ((n\*H)/N)+T/2; n1++ ){

for(int w1 = 0; w1 <= W; w1++){

arr[n1][w1].r = atoi(rgb[0]);

arr[n1][w1].g = atoi(rgb[1]);

arr[n1][w1].b = atoi(rgb[2]);

}

}

}

for(int m = 1; m < M; m++){

for(int m1 = ((m\*W)/M) - (T/2-correct(T)); m1 <= ((m\*W)/M)+T/2; m1++ ){

for(int h1 = 0; h1 < H; h1++){

arr[h1][m1].r = atoi(rgb[0]);

arr[h1][m1].g = atoi(rgb[1]);

arr[h1][m1].b = atoi(rgb[2]);

}

}

}

}

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

char\* opts ="o:n:r:g:b:y:x:t:c:hi";

struct option longOpts[]={

{"oldcolors", 1, NULL, 'o'},

{"newcolors", 1, NULL, 'n'},

{"red", 1, NULL, 'r'},

{"green", 1, NULL, 'g'},

{"blue", 1, NULL, 'b'},

{"xlines", 1, NULL, 'x'},

{"ylines", 1, NULL, 'y'},

{"thickness", 1, NULL, 't'},

{"linecolor", 1, NULL, 'c'},

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

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

{ NULL, 0, NULL, 0}

};

int opt;

int longIndex;

opt=getopt\_long(argc,argv,opts,longOpts,&longIndex);

char\*\* rgb\_value = calloc(2, sizeof(char\*));

char\*\* lines = calloc(4, sizeof(char\*));

char flag []="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!?/.,|@#$^&\*()\_+-=;:~`][{}";

int z11 = 0;

int z12 = 0;

int z2 = 0;

int z30 = 0;

int z31 = 0;

int z32 = 0;

int z33 = 0;

int inf = 0;

while(opt != -1){

switch(opt){

case 'o':

z11 = 1;

break;

case 'n':

z12 = 1;

break;

case 'r':

z2 += 1;

rgb\_value[0] = "-r";

rgb\_value[1] = optarg;

if (!(strpbrk (flag,rgb\_value[1])==NULL)){

printf("\nНеверный ввод, аргумент '-r/--red' должен иметь числовое значение. Попробуйте еще раз\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

break;

case 'g':

z2 += 1;

rgb\_value[0] = "-g";

rgb\_value[1] = optarg;

if (!(strpbrk (flag,rgb\_value[1])==NULL)){

printf("\nНеверный ввод, аргумент '-g/--green' должен иметь числовое значение. Попробуйте еще раз\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

break;

case 'b':

z2 += 1;

rgb\_value[0] = "-g";

rgb\_value[1] = optarg;

if (!(strpbrk (flag,rgb\_value[1])==NULL)){

printf("\nНеверный ввод, аргумент '-b/--blue' должен иметь числовое значение. Попробуйте еще раз\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

break;

case 'y':

z30 += 1;

lines[0] = optarg;

if (!(strpbrk (flag,lines[0])==NULL)){

printf("\nНеверный ввод, аргумент '-y/--ylines' должен иметь числовое значение. Попробуйте еще раз\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

break;

case 'x':

z31 += 1;

lines[1] = optarg;

if (!(strpbrk (flag,lines[1])==NULL)){

printf("\nНеверный ввод, аргумент '-x/--xlines' должен иметь числовое значение. Попробуйте еще раз\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

break;

case 't':

z32 += 1;

lines[2] = optarg;

if (!(strpbrk (flag,lines[2])==NULL)){

printf("\nНеверный ввод, аргумент '-t/--thicness' должен иметь числовое значение. Попробуйте еще раз\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

break;

case 'c':

z33 += 1;

lines[3] = optarg;

break;

case 'h':

DESCRIPTIONS();

printHelp();

return 0;

break;

case 'i':

inf=1;

break;

}

opt = getopt\_long(argc, argv, opts, longOpts, &longIndex);

}

tokens ch = checking\_values(argv, argc);

argc -= optind;

argv += optind;

char\* path\_in = argv[0];

char\* path\_out = argv[argc-1];

BMP img = read\_image(path\_in);

if(inf){

printFileHeader(img.bfh);

printInfoHeader(img.bih);

return 0;

}

if(img.file\_error == 1){

return 0;

}

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

if(!strstr(argv[i], ".bmp")){

printf("\nВведите данные корректно, без лишних и неизвестных для программы ключей.\nИпользуйте ключ --help для вызова справки\n\n");

return 0;

}

}

if( (!path\_out) || (!path\_in) || (img.error\_signal == 1) || (!strstr(path\_out, ".bmp"))){

if((img.error\_signal == 1) && strstr(path\_in, ".bmp")){

printf("\n%s - ошибка чтения файла.Такой файл отсутсвует в данной папке, проверьте правильность написания", path\_in);

printf("\nПо умолчанию первым аргументом должно быть передано имя читаемого bmp-файла\n");

printf("Последним передается имя bmp-файла, в который будут записаны изменения\n");

printf("Попробуйте снова (В данном формате: -//-.bmp)\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

printf("\nПо умолчанию первым аргументом должно быть передано имя читаемого bmp-файла\n");

printf("Последним передается имя bmp-файла, в который будут записаны изменения\n");

printf("Попробуйте снова (В данном формате: -//-.bmp)\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

if( (z11 == 1) || (z12 == 1) ){

if((z2 == 1) || (z30 == 1) || (z31 == 1) || (z32 == 1) || (z33 == 1)){

printf("\nВведите только те аргументы, которые указаны для данной функции\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

if((z11 == 1) && (z12 == 0)){

printf("\nНедостаточно аргументов. Попробуйте снова\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

if((z11 == 0) && (z12 == 1)){

printf("\nНедостаточно аргументов. Попробуйте снова\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

if(ch.error\_flag == 0){

if(color\_replacement(ch, img)){

write\_image(path\_out, img);

}

else{

return 0;

}

}

else{

return 0;

}

}

if(z2 > 0){

if((z11 == 1) || (z12 == 1) || (z30 == 1) || (z31 == 1) || (z32 == 1) || (z33 == 1)){

printf("\nВведите только те аргументы, которые указаны для данной функции\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

if(z2 > 1){

printf("\nВы можете ввести только один компонент RGB\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

int filter = filter\_rgb\_component(rgb\_value, img);

if(filter == 0){

return 0;

}

write\_image(path\_out, img);

}

if(z30 > 0 || z31 > 0 || z32 > 0 || z33 > 0){

if((z11 == 1) || (z12 == 1) || (z2 == 1)){

printf("\nВведите только те аргументы, которые указаны для данной функции\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

if(z30 == 0 || z31 == 0 || z32 == 0 || z33 == 0){

printf("\nОшибка при вводе аргументов, каждый аргумент используется один раз.Попробуйте снова\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

if(z30 + z31 + z32 + z33 < 4){

printf("\nНедостаточно аргументов. Попробуйте снова\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

if(z30 + z31 + z32 + z33 > 4){

printf("\nОшибка при вводе аргументов, каждый аргумент используется один раз.Попробуйте снова\n");

printf("Используйте ключ --help для вызова справки\n\n");

return 0;

}

int cur = cut\_on\_lines(lines, img);

if(cur)

write\_image(path\_out,img);

else

return 0;

}

if((z11 == 0) && (z12 == 0) && (z2 == 0) && (z30 == 0) && (z31 == 0) && (z32 == 0) && (z33 == 0)){

write\_image(path\_out, img);

}

printf("\nРезультат успешно записан в файл - %s.\n\n", path\_out);

free(rgb\_value);

free(lines);

free(img.arr);

free(ch.tval\_new);

free(ch.tval\_old);

return 0;

}