**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Создание программ в Си

|  |  |  |
| --- | --- | --- |
| Студент гр. 3341 |  | Анисимов Д.А. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

**ЗАДАНИЕ на курсовую работу**

Студент Анисимов Д.А.

Группа 3341

Тема работы: Обработка изображений

|  |  |  |
| --- | --- | --- |
| Дата выдачи задания: 18.03.2024 | | |
| Дата сдачи реферата: 21.05.2024 | | |
| Дата защиты реферата: 29.05.2024 | | |
| Студент |  | Анисимов Д.А. |
| Преподаватель |  | Глазунов С.А. |

## Исходные данные

Вариант 3.1

Программа **обязательно должна иметь CLI** (опционально дополнительное использование GUI). Более подробно тут: [**http://se.moevm.info/doku.php/courses:programming:rules\_extra\_kurs**](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 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 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`)
* Установить все компоненты пикселя как максимальную из них. Флаг для выполнения данной операции: `--component\_max`. Т.е. если пиксель имеет цвет (100, 150, 200), то после применения операции цвет будет (200, 200, 200)

# Аннотация

Курсовой проект по варианту 3.1 включает в себя разработку программы с CLI, способной обрабатывать BMP-изображения. Программа должна поддерживать работу с несжатыми BMP-файлами, проверять соответствие файла формату BMP и завершать работу с ошибкой в случае несоответствия. Все поля стандартных BMP-заголовков в выходном файле должны соответствовать значениям входного файла, за исключением тех, которые подлежат изменению.

Функционал программы включает:

Заменяет все пиксели одного заданного цвета на другой цвет

(--color\_replace) (--old\_color) (--new\_color).

Установить все компоненты пикселя как минимальную из них. Флаг для выполнения данной операции: (--component\_min).

Программа завершает работу после выполнения одного из действий, выбранных пользователем.

Исходный код программы: Приложение А.

Тестирование и демонстрация работы программы: Приложение Б.

# СОДЕРЖАНИЕ

|  |  |  |
| --- | --- | --- |
|  | Введение |  |
| 1. | Ход выполнения работы |  |
| 1.1. | Структуры данных и функции |  |
|  | Заключение |  |
|  | Приложение А. Исходный код программы |  |
|  | Приложение Б. Демонстрация работы программы |  |

# Введение

Целью данной работы является создание программы для обработки PNG-изображений с использованием командной строки (CLI). Программа должна будет обеспечивать проверку соответствия файлов формату BMP.

Для достижения цели необходимо выполнить следующие задачи:

Изучение формата BMP.

Разработка CLI для взаимодействия с пользователем и обработки команд.

Реализация функций для обработки изображений, включая:

Замена параметров RGB у пикселя.

Обеспечение проверки BMP-формата и корректной обработки ошибок.

Реализация выравнивания данных в файле и сохранение стандартных значений BMP-заголовков.

Тестирование программы на различных входных данных.

Программа должна быть удобной в использовании, с четко определенными функциями и параметрами для обработки изображений. Все операции должны быть реализованы в виде отдельных функций, что облегчит тестирование и дальнейшее расширение функционала программы.

# Ход выполнения работы

## 1.1. Структуры данных и функции

**Структуры:**

*struct BitmapFileHeader*: структура для хранения данных из Header файла BMP.

*struct BitmapInfoHeader*: стркутура для хранения информации о BMP файле.

*struct Rgb:* структура для хранения информации о цвете пискля.

*struct Rgb\_i:* структура для хранения информации о цветах которые подаются на вход в программу.

**Функции:**

*Rgb\*\* read\_bmp(char file\_name[], BitmapFileHeader \*bmfh, BitmapInfoHeader \*bmif):* Функция для считывания картинки и сохранения ее в массив пикселей.

*void write\_bmp(char file\_name[], Rgb \*\*arr, int H, int W, BitmapFileHeader bmfh, BitmapInfoHeader bmif)*: Функция для создания и записи обработанного BMP файла.

*void color\_replace(unsigned char \*a, unsigned char \*b, unsigned char \*c, unsigned char d, unsigned char f, unsigned char s, unsigned char i, unsigned char g, unsigned char p):* Функция для замены одного выбранного цвета на другой выбранный цвет.*void help():* Вывод справки;

*void max\_color(unsigned char \*a, unsigned char \*b, unsigned char \*c)*: Функция для установки всех компонентов пикселя как максимального из них.

*void printFileHeader(BitmapFileHeader header) и void printInfoHeader(BitmapInfoHeader header)*: Функции для вывода информации о bmp файле.

*int main:* В функции main, используются все выше описанные функции и идет работа с CLI через библиотеку “getopt.h”.

# ЗАКЛЮЧЕНИЕ

Разработана программа на языке программирования Си, обрабатывающая BMP изображения и имеющая CLI. В ходе выполнения работы было изучено устройство BMP файлов; изучены методы считывание и записи файлов; получены навыки обработки изображений;

# приложение А Исходный код программы

Название файла: main.c

#include <getopt.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.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;

typedef struct

{

int b;

int g;

int r;

} Rgb\_i;

#pragma pack(pop)

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);

}

unsigned int padding(unsigned int w) { return (4 - (w \* sizeof(Rgb)) % 4) % 4; }

unsigned int row\_len(unsigned int w) { return w \* sizeof(Rgb) + padding(w); }

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

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

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

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

unsigned int H = bmif->height;

unsigned int W = bmif->width;

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

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

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

fread(arr[i], 1, row\_len(W), 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, 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(unsigned char \*a, unsigned char \*b, unsigned char \*c, unsigned char d, unsigned char f, unsigned char s, unsigned char i, unsigned char g, unsigned char p) {

if (\*a == d && \*b == f && \*c == s) {

\*a = i;

\*b = g;

\*c = p;

}

}

void max\_color(unsigned char \*a, unsigned char \*b, unsigned char \*c) {

if (\*a >= \*b && \*a >= \*c) {

char t = \*a;

\*b = t;

\*c = t;

} else if (\*b >= \*a && \*b >= \*c) {

char t = \*b;

\*c = t;

\*a = t;

} else {

char t = \*c;

\*a = t;

\*b = t;

}

}

void help() {

printf("-h, -helh Выводит справку.\n--info Печатает информацию об изображении и завершает работу.\n-i, --input Задаёт имя входного изображения.\n-o, --output Задаёт имя выходного изображения.\n");

printf("--color\_replace Заменяет все пиксели одного заданного цвета на другой цвет.");

printf("Функционал определяется: \nЦвет, который требуется заменить. Флаг `--old\_color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту.\n");

printf("Цвет на который требуется заменить. Флаг `--new\_color` (работает аналогично флагу `--old\_color`)");

printf("\n--component\_max Установить все компоненты пикселя как максимальную из них. Т.е. если пиксель имеет цвет (100, 150, 200), то после применения операции цвет будет (200, 200, 200).");

exit(0);

}

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

char \*output = NULL;

char \*input = NULL;

char comand = 0;

char \*old\_col = NULL;

char \*new\_col = NULL;

printf("Course work for option 3.1, created by Dmitrii Anisimov.\n");

const struct option long\_options[] = {

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

{"info", no\_argument, NULL, 0},

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

{"color\_replace", no\_argument, NULL, 0},

{"old\_color", required\_argument, NULL, 0},

{"new\_color", required\_argument, NULL, 0},

{"component\_max", no\_argument, NULL, 0},

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

{NULL, 0, NULL, 0}};

int code;

int option\_index;

while ((code = getopt\_long(argc, argv, "ho:i:", long\_options, &option\_index)) != -1) {

switch (code) {

case 'h':

help();

exit(0);

break;

case 'o': {

output = optarg;

break;

};

case 'i': {

input = optarg;

break;

};

case 0: {

const char \*opt\_name = long\_options[option\_index].name;

if (strcmp(opt\_name, "info") == 0) {

if (comand == 0)

comand = 'i';

else

exit(46);

} else if (strcmp(opt\_name, "color\_replace") == 0) {

if (comand == 0)

comand = 'r';

else

exit(46);

} else if (strcmp(opt\_name, "component\_max") == 0) {

if (comand == 0)

comand = 'm';

else

exit(46);

} else if (strcmp(opt\_name, "old\_color") == 0) {

old\_col = optarg;

} else if (strcmp(opt\_name, "new\_color") == 0) {

new\_col = optarg;

}

break;

}

default: {

exit(41);

break;

};

};

};

if (output == NULL)

output = "out.bmp";

if (input == NULL) {

if (argc > optind && (optind + 1 == argc))

input = argv[optind];

else if (argc > optind) {

printf("no file name entered\n");

exit(41);

}

}

if (strcmp(output, input) == 0) {

exit(43);

}

Rgb\_i old, new;

if (old\_col == NULL && comand == 'r') {

printf("missing color\n");

exit(47);

} else if (comand == 'r') {

if (sscanf(old\_col, "%d.%d.%d", &old.r, &old.g, &old.b) != 3) {

printf("color error1\n");

exit(47);

}

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

printf("color error2\n");

exit(47);

}

}

if (new\_col == NULL && comand == 'r') {

printf("missing color\n");

exit(47);

} else if (comand == 'r') {

if (sscanf(new\_col, "%d.%d.%d", &new.r, &new.g, &new.b) != 3) {

printf("color error3\n");

exit(47);

}

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

printf("color error4\n");

exit(47);

}

}

if (comand == 0)

help();

BitmapFileHeader bmfh;

BitmapInfoHeader bmif;

Rgb \*\*arr = read\_bmp(input, &bmfh, &bmif);

if (bmfh.signature != 0x4d42) {

printf("xnjn-yt nfr ");

exit(48);

}

if (bmif.headerSize != 40) {

printf("unsuported bmp version");

exit(49);

}

if (bmif.bitsPerPixel != 24) {

exit(49);

}

if (comand == 'i') {

printFileHeader(bmfh);

printInfoHeader(bmif);

exit(0);

}

unsigned int H = bmif.height;

unsigned int W = bmif.width;

if (comand == 'r') {

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

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

color\_replace(&arr[i][j].r, &arr[i][j].g, &arr[i][j].b, old.r, old.g, old.b, new.r, new.g, new.b);

}

} else if (comand == 'm') {

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

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

max\_color(&arr[i][j].r, &arr[i][j].g, &arr[i][j].b);

}

}

}

write\_bmp(output, arr, bmif.height, bmif.width, bmfh, bmif);

printf("\n");

return 0;

}

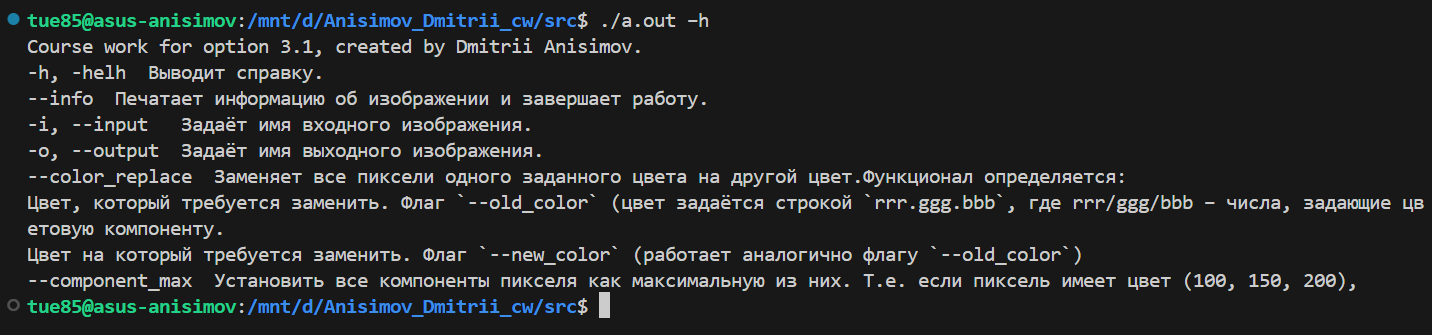
# приложение б Демонстрация работы программы

./a.out --color\_replace --old\_color 0.0.0 --new\_color 0.200.0 files.bmp





./a.out –h



./a.out –component\_max file.bmp





./a.out –info file.bmp

