**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

КУРСОВАЯ РАБОТА

**по дисциплине «Программирование»**

Тема: Обработка изображений

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3341 |  | Шуменков А. П. |
| Преподаватель |  | Глазунов С. А. |

Санкт-Петербург

2024

|  |  |  |
| --- | --- | --- |
| **ЗАДАНИЕ**  **НА КУРСОВУЮ РАБОТУ**  Студент Шуменков Александр Павлович  Группа 3341  Тема работы: Обработка изображений  Исходные данные:  **Вариант 3.2**  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:   * Инверсия цвета на всём изображении. Флаг для выполнения данной операции: `--inverse` * Установить компоненту цвета, как сумму двух других. Флаг для выполнения данной операции: `--component\_sum`. Функционал определяется   Какую компоненту требуется изменить. Флаг `--component\_name`. Возможные значения `red`, `green` и `blue`. | | |
| Содержание пояснительной записки:  «Аннотация», «Содержание», «Введение», «Ход выполнения работы», «Заключение», «Список использованных источников», «Пример работы программы», «Исходный код программы». | | |
| Предполагаемый объем пояснительной записки:  Не менее 15 страниц. | | |
| Дата выдачи задания: 18.03.2024 | | |
| Дата сдачи реферата: 26.05.2024 | | |
| Дата защиты реферата: 2.05.2024 | | |
| Студент |  | Шуменков А. П. |
| Преподаватель |  | Глазунов С. А. |

**Аннотация**

Курсовой проект по варианту 3.2 включает в себя разработку программы с CLI (и опционально GUI), 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. Важно обеспечить корректное выравнивание данных в файле, заполняя мусорные данные нулями. Все поля стандартных BMP-заголовков в выходном файле должны соответствовать значениям входного файла, за исключением тех, которые подлежат изменению.

Функционал программы включает: Инверсия цвета на всём изображении. Флаг для выполнения данной операции: `--inverse`. Установить компоненту цвета, как сумму двух других. Флаг для выполнения данной операции: `--component\_sum`. Функционал определяется: Какую компоненту требуется изменить. Флаг `--component\_name`. Возможные значения `red`, `green` и `blue`.

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

Исходный код программы приведён в приложении A.

Демонстрация работы приведена в приложении B.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 8 |
| 1. | Ход выполнения работы | 10 |
|  | Заключение | 12 |
|  | Список использованных источников | 13 |
|  | Приложение А. Исходный код программы | 14 |
|  | Приложение B. Демонстрация работы программы | 19 |

**введение**

Курсовой проект по варианту 3.2 включает в себя разработку программы с CLI (и опционально GUI), 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. Важно обеспечить корректное выравнивание данных в файле, заполняя мусорные данные нулями. Все поля стандартных BMP-заголовков в выходном файле должны соответствовать значениям входного файла, за исключением тех, которые подлежат изменению.

Функционал программы включает: Инверсия цвета на всём изображении. Флаг для выполнения данной операции: `--inverse`. Установить компоненту цвета, как сумму двух других. Флаг для выполнения данной операции: `--component\_sum`. Функционал определяется: Какую компоненту требуется изменить. Флаг `--component\_name`. Возможные значения `red`, `green` и `blue`.

Программа завершает работу после выполнения одного из действий, выбранных пользователем. Исходный код программы: Приложение А. Тестирование и демонстрация работы программы: Приложение Б.

ВВЕДЕНИЕ Целью данной работы является создание программы для обработки BMP-изображений с использованием командной строки (CLI) и, опционально, графического пользовательского интерфейса (GUI). Программа будет обеспечивать проверку соответствия файлов формату BMP, их обработку согласно заданным параметрам.

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: Инверсия цвета на всём изображении, Установить компоненту цвета, как сумму двух других. Обеспечение проверки BMP-формата и корректной обработки ошибок. Реализация выравнивания данных в файле и сохранение стандартных значений BMP-заголовков.

**Ход выполнения работы**

**BitmapFileHeader:** Эта структура представляет заголовок файла BMP. Она содержит информацию о типе файла, размере файла, зарезервированных полях и смещении массива пикселей.

**BitmapInfoHeader:** Эта структура представляет информационный заголовок BMP. Она содержит детали об изображении, такие как размер заголовка, ширина и высота изображения, количество плоскостей, количество бит на пиксель, тип сжатия, размер изображения, разрешение по горизонтали и вертикали, количество цветов в таблице цветов и количество важных цветов.

**Rgb:** Эта структура представляет цвет пикселя в формате RGB.

**Rgb\_int:** Эта структура представляет цвет пикселя в формате RGB, где каждый компонент цвета представлен целым числом.

**printFileHeader(BitmapFileHeader header):** Эта функция выводит информацию о заголовке файла BMP.

**printInfoHeader(BitmapInfoHeader header):** Эта функция выводит информацию о информационном заголовке BMP.

**print\_help():** Эта функция выводит справочную информацию о доступных опциях программы.

**read\_bmp(char file\_name[], BitmapFileHeader \*bmfh, BitmapInfoHeader \*bmif):** Эта функция считывает BMP-файл и возвращает двумерный массив структур Rgb, представляющих пиксели изображения.

**write\_bmp(char file\_name[], Rgb \*\*arr, int H, int W, BitmapFileHeader bmfh, BitmapInfoHeader bmif):** Эта функция записывает изображение в BMP-файл.

**read\_opts(int argc, char \*argv[], char \*\*filename\_input, char \*\*filename\_output, char \*\*component\_name):** Эта функция обрабатывает аргументы командной строки и возвращает команду, которую следует выполнить.

**run\_command(enum Commands command, Rgb \*\*arr, BitmapFileHeader bmfh, BitmapInfoHeader bmif, char \*\*component\_name):** Эта функция выполняет указанную команду над изображением. Команда может включать вывод информации о файле, замену цвета или установку всех компонентов пикселя как максимальной из них.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы была разработана программа для обработки BMP-изображений с использованием командной строки (CLI) и, опционально, графического пользовательского интерфейса (GUI). Программа успешно обеспечивает проверку соответствия файлов формату BMP и их обработку согласно заданным параметрам. Были изучены основы формата BMP. Разработан CLI для взаимодействия с пользователем и обработки команд. Реализованы функции для обработки изображений, включая инверсию цвета на всём изображении, установка компоненты цвета, как сумму двух других. Программа успешно обеспечивает проверку BMP-формата и корректную обработку ошибок. Реализовано выравнивание данных в файле и сохранение стандартных значений BMP-заголовков. Тестирование программы на различных входных данных показало ее эффективность и корректность работы. Программа удобна в использовании, с четко определенными функциями и параметрами для обработки изображений. В результате, цель курсовой работы была успешно достигнута. Все поставленные задачи были выполнены, что позволяет считать работу завершенной. Возможны дальнейшие исследования для расширения функционала программы и улучшения ее производительности.

**список использованных источников**

1. Кринкин К. В., Берленко Т. А., Заславский М. М., Чайка К. В., Допира В. Е., Гаврилов А. В. Методические указания по выполнению курсовой и лабораторных работ по дисциплине программирование. Второй семестр, 2022.

**приложение А**

**ИСХОДНЫЙ КОД программы**

Название файла: main.c

#include <getopt.h>

#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, g, r;

} Rgb;

#pragma pack(pop)

typedef struct {

int r, g, b;

} Rgb\_int;

enum RGB{

R,

G,

B,

};

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 print\_help() {

printf("Next options available:\n");

printf("-i, --input: input image filename.\n");

printf("-o, --output: output image file name.\n");

printf("-h, --help: this message.\n");

printf("--info: information about BMP file.\n");

printf("--inverse: color inversion across the entire image.\n");

printf("--component\_sum: set a color component as the sum of two others.\n");

printf("--component\_name: which component needs to be changed `red`, `green` and `blue`.\n");

exit(0);

}

enum Commands {

NONE,

INFO,

INVERSE,

SUM,

};

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(W \* sizeof(Rgb) + (4 - (W \* 3) % 4) % 4);

fread(arr[i], 1, W \* sizeof(Rgb) + (4 - (W \* 3) % 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, 1, sizeof(BitmapFileHeader), ff);

fwrite(&bmif, 1, sizeof(BitmapInfoHeader), ff);

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

fwrite(arr[i], 1, W \* sizeof(Rgb) + (4 - (W \* 3) % 4) % 4, ff);

}

fclose(ff);

}

enum Commands read\_opts(int argc, char \*argv[],

char \*\*filename\_input,

char \*\*filename\_output,

char \*\*component\_name) {

enum Commands command = NONE;

const struct option long\_options[] = {

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

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

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

{"info", no\_argument, (int \*)&command, INFO},

{"inverse", no\_argument, (int \*)&command, INVERSE},

{"component\_sum", no\_argument, (int \*)&command, SUM},

{"component\_name", required\_argument, NULL, 0},

{NULL, 0, NULL, 0}};

int c;

int option\_index;

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

switch (c) {

case 'h':

print\_help();

exit(0);

break;

case 'o':

\*filename\_output = optarg;

break;

case 'i':

\*filename\_input = optarg;

break;

case 0:

if (strcmp(long\_options[option\_index].name, "component\_name") == 0)

\*component\_name = optarg;

break;

case '?':

default:

exit(40);

break;

};

};

if (command == NONE)

print\_help();

if (!\*filename\_output)

\*filename\_output = "out.bmp";

if (!\*filename\_input && argc > optind && (optind + 1 == argc)) {

\*filename\_input = argv[optind];

} else if (!\*filename\_input) {

printf("Missing input file\n");

exit(40);

}

if (strcmp(\*filename\_output, \*filename\_input) == 0) {

printf("output should not be same as input");

exit(40);

}

return command;

}

void run\_command(enum Commands command, Rgb \*\*arr, BitmapFileHeader bmfh, BitmapInfoHeader bmif, char \*component\_name) {

switch (command) {

case INFO:

printFileHeader(bmfh);

printInfoHeader(bmif);

exit(0);

break;

case INVERSE:

for(int i=0; i<bmif.height; i++){

for(int j=0; j<bmif.width; j++){

arr[i][j].r = 255 - arr[i][j].r;

arr[i][j].g = 255 - arr[i][j].g;

arr[i][j].b = 255 - arr[i][j].b;

}

}

break;

case SUM:{

enum RGB component;

if (strcmp(component\_name, "red") == 0){

component = R;

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

component = G;

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

component = B;

}else{

printf("Wrong component name\n");

exit(40);

}

if(component == R){

for(int i = 0; i < bmif.height; i++){

for(int j = 0; j < bmif.width; j++){

int summ = arr[i][j].g + arr[i][j].b;

if(summ > 255) summ = 255;

arr[i][j].r = summ;

}

}

}

if(component == G){

for(int i = 0; i < bmif.height; i++){

for(int j = 0; j < bmif.width; j++){

int summ = arr[i][j].r + arr[i][j].b;

if(summ > 255) summ = 255;

arr[i][j].g = summ;

}

}

}

if(component == B){

for(int i = 0; i < bmif.height; i++){

for(int j = 0; j < bmif.width; j++){

int summ = arr[i][j].r + arr[i][j].g;

if(summ > 255) summ = 255;

arr[i][j].b = summ;

}

}

}

}

default:

break;

}

}

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

char \*filename\_input = NULL;

char \*filename\_output = NULL;

char \*component\_name = NULL;

printf("Course work for option 3.2, created by Shumenkov Aleksandr\n");

enum Commands command = read\_opts(argc, argv, &filename\_input, &filename\_output, &component\_name);

BitmapFileHeader bmfh;

BitmapInfoHeader bmif;

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

if (bmfh.signature != 0x4D42 || bmif.bitsPerPixel != 24 || bmif.headerSize != 40) {

printf("not a valid BMP\n");

exit(40);

}

run\_command(command, arr, bmfh, bmif, component\_name);

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

return 0;

}}

**ПРИЛОЖЕНИЕ B**

**ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ**