**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Программа по обработке bmp-файла

|  |  |  |
| --- | --- | --- |
| Студент гр. 3344 |  | Ханнанов А.Ф. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Ханнанов А.Ф. | | |
| Группа 3344 | | |
| Тема работы: “Программа по обработке bmp-файла” | | |
| Исходные данные:   * 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). | | |
| Содержание пояснительной записки:   1. Содержание 2. Введение 3. Задание варианта 4. Функции программы 5. Полученные результаты 6. Заключение 7. Список использованных источников 8. Инструкция по запуску 9. Пример работы программы 10. Код программы | | |
| Предполагаемый объем пояснительной записки:  Не менее 30 страниц. | | |
| Дата выдачи задания: 18.03.2024 | | |
| Дата сдачи реферата: 15.05.2024 | | |
| Дата защиты реферата: 15.05.2024 | | |
| Студент |  | Ханнанов А.Ф. |
| Преподаватель |  | Глазунов С.А. |

**Аннотация**

Курсовая работа подразумевает создание программы по редактированию bmp-файла с реализацией CLI интерфейса. Программа пишется на языке C++. Она реализует методы копирование части изображения, отображение относительно заданной оси, изменение компонент цветовых пикселей. Результатом работы программы является обработанное изображение, сохранённое в заданный файл. Помимо этого, программа обрабатывает ошибки, которые могут возникнуть при неправильных входных данных.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Аннотация | 4 |
|  | Содержание | 5 |
|  | Введение | 6 |
| 1. | Задание варианта | 7 |
| 2. | Функции программы | 9 |
| 2.1. | main.cpp | 9 |
| 2.2. | inputFunctions.cpp | 9 |
| 2.3. | secondary\_functions.cpp | 9 |
| 2.4. | check\_functions.cpp | 10 |
| 2.5. | functions.cpp | 10 |
| 3. | Полученные результаты | 11 |
| 4. | Заключение | 12 |
| 5. | Список использованных источников | 13 |
| 6. | Инструкция по запуску | 14 |
| 7. | Пример работы программы | 15 |
|  | Приложение А. Код программы | 17 |

**введение**

Целью работы является создание программы для обработки bmp-изображения через CLI интерфейс.

Для достижения цели нужно решить следующие задачи:

1. Изучение bmp формата
2. Изучение работы с данными, вводящимися через консоль (библиотека getopt)
3. Реализация функций работы с изображением
4. Обработка некорректных входных данных
5. **Задание варианта**

Программа должна иметь следующую функции по обработке изображений:

1. Отражение заданной области. Флаг для выполнения данной операции: `--mirror`. Этот функционал определяется:
   * выбором оси относительно которой отражать (горизонтальная или вертикальная). Флаг `--axis`, возможные значения `x` и `y`
   * Координатами левого верхнего угла области. Флаг `--left\_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y
   * Координатами правого нижнего угла области.  Флаг `--right\_down`, значение задаётся в формате `right.down`, где right – координата по x, down – координата по y
2. Копирование заданной области. Флаг для выполнения данной операции: `--copy`. Функционал определяется:
   * Координатами левого верхнего угла области-источника. Флаг `--left\_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y
   * Координатами правого нижнего угла области-источника. Флаг `--right\_down`, значение задаётся в формате `right.down`, где right – координата по x, down – координата по y
   * Координатами левого верхнего угла области-назначения. Флаг `--dest\_left\_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y
3. 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`)
4. 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

* Каждую подзадачу следует вынести в отдельную функцию, функции сгруппировать в несколько файлов (например, функции обработки текста в один, функции ввода/вывода в другой). Сборка должна осуществляться при помощи make и Makefile или другой системы сборки

1. **Функции программы**
   1. **main.cpp**
      * main – основная функция, из которой идёт вызов всех остальных а также возвращение результата
   2. **inputFunctions.cpp**
      * inputFunction – с помощью getopt считывает данные, переданные в консоли
   3. **secondary\_functions.cpp**
      * print\_file\_header – выводит данные заголовка
      * print\_info\_header – выводит информацию о файле
      * readBMP – читает bmp файл
      * writeBMP – записывает bmp файл
      * stringDecomposeFunc – разбивает строку вида “ЧИСЛО.ЧИСЛО” на два числа и возвращает число в зависимости от корректности считывания
      * colorDecomposeFunc – разбивает строку вида “ЧИСЛО.ЧИСЛО. ЧИСЛО” на 3 компоненты цвета и возвращает число в зависимости от корректности считывания
      * checkCoords – меняет значения координат, если они не входят в область изображения
      * findInputFile – ищет среди переданных функции флагов флаг задания входного файла
      * findOutputFile – ищет среди переданных функции флагов флаг задания выходного файла
      * checkFileName – проверяет корректность названия файла
   4. **check\_functions.cpp**
      * checkMirrorValues – проверяет значения, переданные при вызове опции –mirror. В случае из корректности вызывает функцию mirrorFunc
      * checkCopyValues – проверяет значения, переданные при вызове опции –copy. В случае из корректности вызывает функцию copyFunc
      * checkReplaceValues – проверяет значения, переданные при вызове опции –color\_replace. В случае из корректности вызывает функцию replaceFunc
      * checkFilterValues – проверяет значения, переданные при вызове опции –rgbfilter. В случае из корректности вызывает функцию filterFunc
   5. **functions.cpp**
      * mirrorFunc – выполняет опцию –mirror
      * copyFunc – выполняет опцию --copy
      * replaceFunc – выполняет опцию –color\_replace
      * filterFunc – выполняет опцию –rgbfilter
2. **полученные результаты**

Созданная программа выполняет поставленные задачи. Она позволяет пользователю заниматься редактированием bmp изображений через консоль и имеет следующие возможности:

* Открытие и считывание изображений bmp формата
* Редактирование этих изображений с помощью четырёх различных опций
* Запись bmp файла
* Анализ входных данных на корректность

1. **заключение**

Созданная программа успешно выполняет поставленные задачи. Она способна считывать, обрабатывать и записывать изображения bmp формата.

1. **список использованных источников**

1. Базовые сведения к выполнению курсовой работы по дисциплине «программирование». Второй семестр / сост.: М. М. Заславский, А. А. Лисс, А. В. Гаврилов и др.. СПб.: Изд-во СПбГЭТУ “ЛЭТИ”, 2024. 36 с. (Дата обращения: 20.04.2024)

2. Сайт с примерами выполнения прикладных задач по программированию // GeeksforGeeks. URL: <https://www.geeksforgeeks.org/> (Дата обращения: 23.04.2024)

3. Сайт с информацией по программированию // Metanit. URL: <https://metanit.com/> (Дата обращения: 23.04.2024)

1. **Инструкция по запуску**

Для запуска программы нужно провести сбору с помощью команды “make”. После сборки для запуска программы надо ввести “./cw” и начать ввод данных.

1. **Примеры работы программы**
   1. Функция –mirror:

*Входные данные:*

./cw —mirror —left\_up 250.20 —right\_down 350.175 —axis y -i test\_image.bmp

*Выходные данные:*



* 1. Функция –copy:

*Входные данные:*

./cw —copy —left\_up 50.50 —right\_down 150.150 —dest\_left\_up 250.100 -i test\_image.bmp

*Выходные данные:*



* 1. Функция –rgbfilter:

*Входные данные:*

./cw —rgbfilter —component\_value 200 —component\_name green -i test\_image.bmp

*Выходные данные:*



* 1. Функция –color\_replace:

*Входные данные:*

./cw —color\_replace —old\_color 252.255.255 —new\_color 0.0.255 -i test\_image.bmp

*Выходные данные:*



**приложение А**

**Код программы**

**Makefile**

CC = g++

all: cw

cw: main.o functions.o check\_functions.o secondary\_functions.o inputFunctions.o

$(CC) -o cw main.o functions.o secondary\_functions.o check\_functions.o inputFunctions.o

main.o: main.cpp BMP.hpp functions.hpp inputFunctions.hpp

$(CC) -c main.cpp

functions.o: functions.cpp BMP.hpp

$(CC) -c functions.cpp

check\_functions.o: check\_functions.cpp functions.hpp secondary\_functions.hpp BMP.hpp

$(CC) -c check\_functions.cpp

secondary\_functions.o: secondary\_functions.cpp BMP.hpp

$(CC) -c secondary\_functions.cpp

inputFunctions.o: inputFunctions.cpp

$(CC) -c inputFunctions.cpp

clean:

rm -f \*.o cw

**BMP.hpp**

#ifndef \_BMP\_H\_

#define \_BMP\_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)

#endif

**check\_functions.cpp**

#include <string>

#include <vector>

#include <utility>

#include <cstdlib>

#include <sstream>

#include <iostream>

#include "BMP.hpp"

#include "functions.hpp"

#include "secondary\_functions.hpp"

#include "check\_functions.hpp"

using namespace std;

int

checkMirrorValues (vector <pair<string, string>> dict, Rgb\*\* arr, int H, int W) {

unsigned short int countValues = 0;

int left, up, right, down, isCorrect;

char axis;

for (size\_t i = 1; i < dict.size(); i++) {

if (dict[i].second == "NULL") return 0;

if (dict[i].first == "A") {

if (dict[i].second.length() != 1 || (dict[i].second != "x" && dict[i].second != "y")) {

cerr << "Введено некорректное значение для флага --axis\n";

return 0;

}

axis = dict[i].second[0];

} else if (dict[i].first == "L") {

isCorrect = stringDecomposeFunc(&left, &up, dict[i].second);

if (isCorrect != 1) {

cerr << "Введено некорректное значение для флага --left\_up\n";

return 0;

}

} else if (dict[i].first == "R") {

isCorrect = stringDecomposeFunc(&right, &down, dict[i].second);

if (isCorrect != 1) {

cerr << "Введено некорректное значение для флага --right\_down\n";

return 0;

}

} else if (dict[i].first == "i" || dict[i].first == "o") {

continue;

} else {

cerr << "Введён некорректный флаг опции\n";

return 0;

}

countValues++;

}

if (countValues == 3) {

checkCoords(&left, &up, &right, &down, H, W);

mirrorFunc(arr, axis, left, up, right, down, H, W);

} else {

cerr << "Введены не все флаги, необходимые для опции\n";

return 0;

}

return 1;

}

int

checkCopyValues (vector <pair<string, string>> dict, Rgb\*\* arr, int H, int W) {

unsigned short int countValues = 0;

int left, up, right, down, newLeft, newUp, isCorrect;

for (size\_t i = 1; i < dict.size(); i++) {

if (dict[i].second == "NULL") return 0;

if (dict[i].first == "L") {

isCorrect = stringDecomposeFunc(&left, &up, dict[i].second);

if (isCorrect != 1) {

cerr << "Введено некорректное значение для флага --left\_up\n";

return 0;

}

} else if (dict[i].first == "R") {

isCorrect = stringDecomposeFunc(&right, &down, dict[i].second);

if (isCorrect != 1) {

cerr << "Введено некорректное значение для флага --right\_down\n";

return 0;

}

} else if (dict[i].first == "D") {

stringDecomposeFunc(&newLeft, &newUp, dict[i].second);

if (isCorrect != 1) {

cerr << "Введено некорректное значение для флага --dest\_left\_up\n";

return 0;

}

} else if (dict[i].first == "i" || dict[i].first == "o") {

continue;

} else {

cerr << "Введён некорректный флаг опции\n";

return 0;

}

countValues++;

}

if (countValues == 3) {

if (left < 0 || right >= W || right <= left || up < 0 || down >= H || down <= up || newLeft + (right - left) >= W || newUp + (down - up) >= H || newLeft < 0 || newUp < 0)

return 1;

copyFunc(arr, left, H - up - 1, right, H - down - 1, newLeft, H - newUp - 1, H, W);

} else {

cerr << "Введены не все флаги, необходимые для опции" << endl;

return 0;

}

return 1;

}

int

checkReplaceValues (vector <pair<string, string>> dict, Rgb\*\* arr, int H, int W) {

unsigned short int countValues = 0;

int r, g, b, newR, newG, newB, isCorrect;

for (size\_t i = 1; i < dict.size(); i++) {

if (dict[i].second == "NULL") return 0;

if (dict[i].first == "O") {

isCorrect = colorDecomposeFunc(&r, &g, &b, dict[i].second);

if (isCorrect == 1) {

cerr << "Введено некорректное значение для флага --old\_color\n";

return 0;

}

if (isCorrect == 2) {

cerr << "Введено некорректное знаение компоненты цвета\n";

return 0;

}

} else if (dict[i].first == "N") {

isCorrect = colorDecomposeFunc(&newR, &newG, &newB, dict[i].second);

if (isCorrect == 1) {

cerr << "Введено некорректное значение для флага --old\_color\n";

return 0;

}

if (isCorrect == 2) {

cerr << "Введено некорректное знаение компоненты цвета\n";

return 0;

}

} else if (dict[i].first == "i" || dict[i].first == "o") {

continue;

} else {

cerr << "Введён некорректный флаг опции\n";

return 0;

}

countValues++;

}

if (countValues == 2) {

replaceFunc(arr, r, g, b, newR, newG, newB, H, W);

} else {

cerr << "Введены не все флаги, необходимые для опции" << endl;

return 0;

}

return 1;

}

int

checkFilterValues(vector <pair<string, string>> dict, Rgb\*\* arr, int H, int W) {

unsigned short int countValues = 0;

string component;

size\_t compVal;

for (size\_t i = 1; i < dict.size(); i++) {

if (dict[i].second == "NULL") return 0;

if (dict[i].first == "C") {

component = dict[i].second;

if (component != "red" && component != "blue" && component != "green") {

cerr << "Введено некорректное значение для флага --component\_name\n";

return 0;

}

} else if (dict[i].first == "V") {

stringstream s;

s << dict[i].second;

s >> compVal;

if (s.str() != to\_string(compVal) || compVal < 0 || compVal > 255) {

cerr << "Введено некорректное значение для флага --component\_value\n";

return 0;

}

}else if (dict[i].first == "i" || dict[i].first == "o") {

continue;

} else {

cerr << "Введён некорректный флаг опции\n";

return 0;

}

countValues++;

}

if (countValues == 2) {

filterFunc(arr, component, compVal, H, W);

} else {

cerr << "Введены не все флаги, необходимые для опции\n";

return 0;

}

return 1;

}

**functions.cpp**

#include <cstdlib>

#include <string>

#include <iostream>

#include "BMP.hpp"

#include "functions.hpp"

void

mirrorFunc (Rgb\*\* arr, char axis, int left, int up, int right, int down, int H, int W) {

unsigned char r, g, b;

Rgb a;

if (axis == 'y') {

for (size\_t y = H - 1 - down; y < (H - 1) - (up + down) / 2; y++) {

for (size\_t x = left; x < right + 1; x++) {

a = arr[y][x];

arr[y][x] = arr[2 \* (H - 1) - up - down - y][x];

arr[2 \* (H - 1) - up - down - y][x] = a;

}

}

} else if (axis == 'x') {

for (size\_t y = H - 1 - down; y < H - up; y++) {

for (size\_t x = left; x < (left + right) / 2 + 1; x++) {

a = arr[y][x];

arr[y][x] = arr[y][right + left - x];

arr[y][right + left - x] = a;

}

}

}

}

void

copyFunc(Rgb\*\* arr, int left, int up, int right, int down, int newLeft, int newUp, int H, int W) {

if (left >= right || up <= down) return;

Rgb\*\* temporaryArr = new Rgb\* [up - down];

for (size\_t i = 0; i < up - down; i++) {

temporaryArr[i] = new Rgb [right - left];

for (size\_t j = 0; j < right - left; j++) {

temporaryArr[i][j] = arr[up - i][left + j];

}

}

for (size\_t i = 0; i < up - down; i++) {

for (size\_t j = 0; j < right - left; j++) {

if (newUp - i >= H || newUp - i < 0 || newLeft + j >= W || newLeft + j < 0) continue;

arr[newUp - i][newLeft + j] = temporaryArr[i][j];

}

delete [] temporaryArr[i];

}

delete [] temporaryArr;

}

void

replaceFunc(Rgb\*\* arr, int r, int g, int b, int newR, int newG, int newB, int H, int W) {

for (size\_t y = 0; y < H; y++) {

for (size\_t x = 0; x < W; x++) {

if (arr[y][x].r == r && arr[y][x].b == b && arr[y][x].g == g) {

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

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

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

}

}

}

}

void

filterFunc(Rgb\*\* arr, std::string component, int compVal, int H, int W) {

for (size\_t y = 0; y < H; y++) {

for (size\_t x = 0; x < W; x++) {

if (component == "red") {

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

} else if (component == "blue") {

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

} else if (component == "green") {

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

}

}

}

}

void

printHelp () {

std::cout << "Course work for option 5.10, created by Artem Khannanov\n\n";

std::cout << "Описание функций:\n\n\

--mirror - отражение заданной области;\n\

--axis - ось, по которой будет отражение\n\

--left\_up - верхняя левая координата области\n\

--right\_down - правая нижняя координата области\n\

--copy - копирование заданной области и вставка её в впо другим координатам;\n\

--left\_up - верхняя левая координата области\n\

--right\_down - правая нижняя координата области\n\

--dest\_left\_up - верхняя левая координата новой области\n\

--color\_replace - замена одного цвета на другой по всему изображению;\n\

--old\_color - старый цвет\n\

--new\_color - новый цвет\n\

--rgbfilter - замена компонента цвета на заданное значение по всему изображению;\n\

--component\_name - название компоненты\n\

--component\_value - новое значение компоненты\n";

}

**inputFunctions.cpp**

#include <vector>

#include <string>

#include <getopt.h>

#include "inputFunctions.hpp"

using namespace std;

int

inputFunction (vector<pair<string, string>>\* dict, int argc, char\* argv[]) {

int opt, optionIndex, valuesCount;

char\* charValue = nullptr;

valuesCount = 0;

static struct option longOptions[] =

{

{"mirror", no\_argument, 0, 'm'}, {"axis", required\_argument, 0, 'A'},

{"left\_up", required\_argument, 0, 'L'}, {"right\_down", required\_argument, 0, 'R'},

{"copy", no\_argument, 0, 'c'}, {"dest\_left\_up", required\_argument, 0, 'D'},

{"color\_replace", no\_argument, 0, 'r'}, {"old\_color", required\_argument, 0, 'O'},

{"new\_color", required\_argument, 0, 'N'},

{"rgbfilter", no\_argument, 0, 'f'}, {"component\_name", required\_argument, 0, 'C'},

{"component\_value", required\_argument, 0, 'V'},

{"input", required\_argument, 0, 'i'}, {"output", required\_argument, 0, 'o'},

{"info", no\_argument, 0, 'I'}, {"help", no\_argument, 0, 'h'},

{0, 0, 0, 0}

};

opt = getopt\_long(argc, argv, "mA:L:R:cD:rO:N:fC:V:i:o:h", longOptions, &optionIndex);

while (opt != -1) {

charValue = optarg;

valuesCount++;

string value;

char a[1];

a[0] = opt;

string name(a);

if (charValue != NULL) {

valuesCount++;

value += charValue;

} else {

value += "NULL";

}

(\*dict).emplace\_back(name, value);

opt = getopt\_long(argc, argv, "mA:L:R:cD:rO:N:fC:V:i:o:h", longOptions, &optionIndex);

}

return valuesCount;

}

**main.cpp**

#include <unistd.h>

#include <getopt.h>

#include <stdlib.h>

#include <stdio.h>

#include <string>

#include <vector>

#include <iostream>

#include <sstream>

#include "BMP.hpp"

#include "functions.hpp"

#include "check\_functions.hpp"

#include "secondary\_functions.hpp"

#include "inputFunctions.hpp"

using namespace std;

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

int isCorrectFile, valuesCount, isCheckCorrect;

string InputFileName, OutputFileName;

vector <pair<string, string>> optDict;

BitmapFileHeader bmfh;

BitmapInfoHeader bmif;

// Считывание данных, первичная проверка ввода

valuesCount = inputFunction(&optDict, argc, argv);

if (valuesCount == 0) {

cerr << "Введите данные для работы программы\n";

return 40;

}

if (optDict[0].first == "h") {

printHelp();

return 0;

}

// ----------------------

// Обработка входного файла

if (argc - 1 > valuesCount) {

InputFileName = argv[argc - 1];

} else {

if (findInputFile(optDict, &InputFileName) == 0) {

cerr << "Введите название входного файла\n";

return 40;

}

}

isCorrectFile = checkFileName(InputFileName);

if (isCorrectFile == 0) {

cerr << "Введено некорректное имя входного файла\n";

return 40;

}

// -----------------------

// Обработка выходного файла

OutputFileName = "out.bmp";

findOutputFile(optDict, &OutputFileName);

isCorrectFile = checkFileName(OutputFileName);

if (isCorrectFile == 0) {

cerr << "Введено некорректное имя выходного файла\n";

return 40;

}

// -----------------------

// Проверка входного и выходного файлов на равенство

if (OutputFileName == InputFileName) {

cerr << "Входной и выходной файлы должны иметь разные названия\n";

return 41;

}

// -----------------------

// Создание массива пикселей

Rgb\*\* arr = readBMP(InputFileName, &bmfh, &bmif);

if (arr == NULL) return 42;

// ----------------------

if (optDict[0].first == "I") {

print\_info\_header(bmif);

cout << "\n";

print\_file\_header(bmfh);

return 0;

}

isCheckCorrect = 1;

if (optDict[0].first == "m") {

isCheckCorrect = checkMirrorValues(optDict, arr, bmif.height, bmif.width);

} else if (optDict[0].first == "c") {

isCheckCorrect = checkCopyValues(optDict, arr, bmif.height, bmif.width);

} else if (optDict[0].first == "r") {

isCheckCorrect = checkReplaceValues(optDict, arr, bmif.height, bmif.width);

} else if (optDict[0].first == "f") {

isCheckCorrect = checkFilterValues(optDict, arr, bmif.height, bmif.width);

} else {

cerr << "Введены неверные данные\n";

return 49;

}

if (isCheckCorrect == 0) return 43;

writeBMP(OutputFileName, arr, bmif.height, bmif.width, bmfh, bmif);

return 0;

}

**secondary\_functions.cpp**

#include <cstdio>

#include <string>

#include <sstream>

#include <iostream>

#include <vector>

#include <fstream>

#include "BMP.hpp"

#include "secondary\_functions.hpp"

#include "secondary\_functions.hpp"

using namespace std;

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);

}

Rgb\*\*

readBMP (string& file\_name, BitmapFileHeader\* bmfh, BitmapInfoHeader\* bmif) {

// Открытие файла

ifstream f(file\_name, ios::binary);

if (!f.is\_open()) {

cerr << "Не удалось считать файл\n";

return NULL;

}

// -----------------

// Считывание заголовков

f.read(reinterpret\_cast <char\*> (bmfh), sizeof(BitmapFileHeader));

f.read(reinterpret\_cast <char\*> (bmif), sizeof(BitmapInfoHeader));

// -----------------

// Проверка на корректность

if (bmfh->signature != 0x4D42) {

cerr << "Файл не является bmp\n";

return NULL;

}

if (bmif->compression != 0 || bmif->bitsPerPixel != 24 || bmif->headerSize != 40) {

cerr << "Данная версия bmp не поддерживается\n";

return NULL;

}

// ----------------------

// Считывание массива пикселей

unsigned int H = bmif->height;

unsigned int W = bmif->width;

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

Rgb\*\* arr;

arr = new Rgb\* [H];

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

for (size\_t i = 0; i < H; i++) {

arr[i] = new Rgb [W \* sizeof(Rgb) + padding];

f.read(reinterpret\_cast <char\*> (arr[i]), W \* sizeof(Rgb) + padding);

}

// ----------------------

f.close();

return arr;

}

void

writeBMP (string& fileName, Rgb\*\* arr, int H, int W, BitmapFileHeader bmfh, BitmapInfoHeader bmif) {

ofstream f(fileName, ios::binary);

f.write(reinterpret\_cast <char\*> (&bmfh), sizeof(BitmapFileHeader));

f.write(reinterpret\_cast <char\*> (&bmif), sizeof(BitmapInfoHeader));

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

for (size\_t i = 0; i < H; i++) {

f.write(reinterpret\_cast <char\*> (arr[i]), W \* sizeof(Rgb) + padding);

}

f.close();

}

// 1 - True; 0 - False

int

stringDecomposeFunc(int\* first, int\* second, string value) {

char dot;

istringstream sd(value);

sd >> \*first >> dot >> \*second;

sd.clear();

if (sd.fail() || dot != '.') return 0;

return 1;

}

// 0 - True; 1 - Wrong type; 2 - Wrong color

int

colorDecomposeFunc(int\* r, int\* g, int\* b, string value) {

char dot1, dot2;

istringstream sd(value);

sd >> \*r >> dot1 >> \*g >> dot2 >> \*b;

sd.clear();

if (sd.fail() || dot1 != '.' || dot2 != '.') return 1;

if (\*r > 255 || \*r < 0 || \*b > 255 || \*b < 0 || \*g > 255 || \*g < 0) return 2;

return 0;

}

void

checkCoords (int\* left, int\* up, int\* right, int\* down, int H, int W) {

if (\*right >= W) \*right = W - 1;

if (\*left < 0) \*left = 0;

if (\*down >= H) \*down = H - 1;

if (\*up < 0) \*up = 0;

}

int

findInputFile (vector <pair<string, string>> dict, string\* InputFileName) {

for (size\_t i = 0; i < dict.size(); i++) {

if (dict[i].first == "i" && dict[i].second != "NULL") {

(\*InputFileName) = dict[i].second;

return 1;

}

}

return 0;

}

void

findOutputFile (vector <pair<string, string>> dict, string\* OutputFileName) {

for (size\_t i = 0; i < dict.size(); i++) {

if (dict[i].first == "o" && dict[i].second != "NULL") {

(\*OutputFileName) = dict[i].second;

return;

}

}

}

// 0 - wrong file; 1 - normal file

int

checkFileName (string filename) {

if (filename.size() < 5) return 0;

return 1;

}