**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Обработка PNG файла

|  |  |  |
| --- | --- | --- |
| Студент гр. 3341 |  | Перевалов П.И. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Перевалов П.И. | | |
| Группа 3341  Вариант 16 | | |
| 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 png-файла  **Общие сведения**   * **Формат картинки PNG (рекомендуем использовать библиотеку libpng)** * без сжатия * файл может не соответствовать формату PNG, т.е. необходимо проверка на PNG формат. Если файл не соответствует формату PNG, то программа должна завершиться с соответствующей ошибкой. * обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями. * все поля стандартных PNG 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) Рисование квадрата. Флаг для выполнения данной операции: `--square`. Квадрат определяется:   + Координатами левого верхнего угла. Флаг `--left\_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y   + Размером стороны. Флаг `--side\_size`. На вход принимает число больше 0   + Толщиной линий. Флаг `--thickness`. На вход принимает число больше 0   + Цветом линий. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет)   + Может быть залит или нет. Флаг `--fill`. Работает как бинарное значение: флага нет – false , флаг есть – true.   + Цветом которым он залит, если пользователем выбран залитый. Флаг `--fill\_color` (работает аналогично флагу `--color`) * (2) Поменять местами 4 куска области. Флаг для выполнения данной операции: `--exchange`. Выбранная пользователем прямоугольная область делится на 4 части и эти части меняются местами. Функционал определяется:   + 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   + Способом обмена частей: “по кругу”, по диагонали. Флаг `--exchange\_type`, возможные значения: `clockwise`, `counterclockwise`, `diagonals` * (3) Находит самый часто встречаемый цвет и заменяет его на другой заданный цвет. Флаг для выполнения данной операции: `--freq\_color`. Функционал определяется:   + Цветом, в который надо перекрасить самый часто встречаемый цвет. 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: 19.05.2024 | | |
| Дата защиты реферата: 23.05.2024 | | |
| Студент |  | Перевалов П.И. |
| Преподаватель |  | Глазунов С.А. |

**Аннотация**

В данной курсовой работе была реализована программа, обрабатывающая PNG изображения. 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).

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 7 |
| 1. | Работа с файлами | 8 |
| 2. | Ввод аргументов | 9 |
| 3. | Обработка изображения | 10 |
|  | Заключение | 14 |
|  | Список использованных источников | 15 |
|  | Приложение А. Исходный код программы | 16 |
|  | Приложение Б. Тестирование | 35 |
|  |  |  |

**введение**

Цель работы заключается в создании программы для обработки bmp-файлов с использованием командной строки (CLI) и, при необходимости, графического интерфейса пользователя (GUI). Программа должна проверять соответствие файла формату PNG, а также реализовывать следующий функционал:

1. Рисование квадрата с возможностью указания координат точек, толщины линий и цвета.
2. Отражение частей изображения с возможностью указания оси отражения.
3. Нахождение самого часто встречаемого цвета с возможностью замены его на другой заданный цвет.

Программа должна также обеспечивать сохранение всех полей стандартных PNG заголовков с соответствующими значениями и обеспечивать выравнивание данных в файле. Каждая функция должна быть реализована отдельно, а ввод и вывод данных также должны быть организованы в виде отдельных функций.

**1. Работа с файлами**

Функция read\_png\_file отвечает за чтение данных из png-файла. Она принимает имя файла, а также ссылки на высоту, ширину, указатель где будет храниться информация о пикселях изображения, тип цвета и глубину цвета. Функция открывает файл в двоичном режиме, считывает заголовки и пиксели изображения, выделяя память для массива пикселей и осуществляя необходимое выравнивание данных. В случае неудачи или некорректного формата файла функция прекращает работу программы.

Функция write\_png\_file записывает данные в png-файл. Она принимает имя файла, высоту и ширину изображения, а также ссылку на двумерный массив содержащий цветовую информацию изображения. Функция открывает файл в двоичном режиме для записи, записывает заголовки и пиксели изображения с учетом выравнивания данных и закрывает файл.

**2. ВВОД АРГУМЕНТОВ**

CLI (Command Line Interface) в данной программе реализована с использованием опций командной строки. Для обработки аргументов командной строки используются структуры option, которые определяют различные действия, доступные в программе.

Для каждой основной команды (line, mirror, color) определены соответствующие наборы опций командной строки. Например, для команды line опции определены в структуре option\_line, а для команды color — в структуре option\_penta.

Функция processing\_flags осуществляет анализ аргументов командной строки и выбор нужного действия в зависимости от команды и вызывает функцию обработки этой функции.

Каждая функция обработки команды (processSquareCommand, processExchangeCommand, processFreqCommand) осуществляет парсинг опций командной строки и вызов соответствующей функции для выполнения задачи (например, line для команды line). В случае неверных данных или ошибочных аргументов функции выводят сообщение об ошибке и завершают работу программы с соответствующим кодом ошибки.

Таким образом, пользователь может использовать CLI для выполнения различных действий с png-файлами, таких как рисование линий, пентаграммы или отражение области изображения, передавая соответствующие аргументы командной строки.

**3. ОСНОВНЫЕ ФУНКЦИИ**

checkValidName: Проверяет валидность имени файла с помощью регулярного выражения.

findUnknownKey: Ищет неизвестные ключи среди аргументов командной строки.

outputHelp: Выводит справочную информацию о доступных опциях.

outputInfo: Выводит информацию о высоте и ширине изображения.

isPNG: Проверяет, является ли файл PNG-изображением.

convertCoords: Преобразует строку координат в целые числа для x и y.

read\_png\_file: Считывает PNG-файл и загружает его в структуру данных.

write\_png\_file: Записывает измененное изображение в новый PNG-файл.

set\_pixel\_color: Устанавливает цвет конкретного пикселя.

getRgb: Преобразует строку с цветом в формате RGB в массив целых чисел.

checkValidCoord: Проверяет, находятся ли координаты внутри границ изображения.

drawCircle: Рисует окружность на изображении.

drawLine: Рисует линию на изображении.

fillSquare: Заполняет квадрат заданным цветом.

drawSquare: Рисует квадрат на изображении.

processSquareCommand: Обрабатывает команду рисования квадрата.

processExchangeCommand: Обрабатывает команду обмена областями изображения.

findMostFrequentColor: Находит наиболее часто встречающийся цвет на изображении.

processFreqCommand: Обрабатывает команду замены наиболее часто встречающегося цвета.

**заключение**

В ходе выполнения данного проекта была разработана программа для обработки изображений в формате PNG. Программа имеет командную строку (CLI), что обеспечивает удобство взаимодействия с пользователем. Она реализует следующий функционал:

Рисование квадрата: Пользователь может указать координаты начала и конца линии, его толщину линий и цвет.

Отражение области: Пользователь задает координаты левого верхнего и правого нижнего угла, а так же ось отражения.

Замена цвета: Позволяет заменить самый часто встречающийся цвет. Пользователь указывает цвет.

Важным аспектом является обработка входных данных и валидация параметров пользовательского ввода. Программа проверяет соответствие входного изображения формату PNG и корректность всех переданных параметров.

**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

**приложение А**

**Исходный код программы**

Название файла: main.cpp

#include <iostream>

#include <fstream>

#include <getopt.h>

#include <png.h>

#include <regex>

#include <string.h>

#include <cmath>

#include <vector>

#define CW "Course work for option 4.16, created by Pasha Perevalov\n"

#define H "--square - рисование квадрата, --exchange - поменять 4 области местами, --freq\_color - замена самого часто встречаемого цвета\n"

#define signature\_size 8

typedef struct{

int R;

int G;

int B;

}Rgb;

static struct option keys[] = {

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

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

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

{"info", no\_argument, 0, 'd'},

{"square", no\_argument, 0, 's'},

{"exchange", no\_argument, 0, 'e'},

{"freq\_color", no\_argument, 0, 'y'},

{"left\_up", required\_argument, 0, 'u'},

{"right\_down", required\_argument, 0, 'p'},

{"side\_size", required\_argument, 0, 'q'},

{"thickness", required\_argument, 0, 'w'},

{"color", required\_argument, 0, 'z'},

{"fill", no\_argument, 0, 'f'},

{"fill\_color", required\_argument, 0, 'g'},

{"exchange\_type", required\_argument, 0, 't'},

{"radius", required\_argument, 0, 'l'},

{0, 0, 0, 0}

};

static struct option selectAction[] = {

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

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

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

{"info", no\_argument, 0, 'd'},

{"square", no\_argument, 0, 'r'},

{"exchange", no\_argument, 0, 'x'},

{"freq\_color", no\_argument, 0, 'c'},

{0, 0, 0, 0}

};

static struct option squareKeys[] = {

{"left\_up", required\_argument, 0, 'u'},

{"side\_size", required\_argument, 0, 'd'},

{"thickness", required\_argument, 0, 't'},

{"color", required\_argument, 0, 'c'},

{"fill", no\_argument, 0, 'f'},

{"fill\_color", required\_argument, 0, 'g'},

{0, 0, 0, 0}

};

static struct option exchangeKeys[] = {

{"exchange", no\_argument, 0, 'd'},

{"right\_down", required\_argument, 0, 'w'},

{"left\_up", required\_argument, 0, 'z'},

{"exchange\_type", required\_argument, 0, 'y'},

{0, 0, 0, 0}

};

static struct option freqKeys[] = {

{"color", required\_argument, 0, 'l'},

{0, 0, 0, 0}

};

bool checkValidName(std::string name){

return(std::regex\_match(name, std::regex("((./)||((\\w+/)+))?(\\w+).(\\w+)")));

}

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

int keyIndex;

int opt;

char\*\* argvCopy = new char\*[argc];

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

size\_t len = strlen(argv[i]) + 1;

argvCopy[i] = new char[len];

strncpy(argvCopy[i], argv[i], len);

}

while(true){

opt = getopt\_long(argc, argvCopy, "io:h", keys, &keyIndex);

if(opt == -1){

opt = 0;

break;

}

switch(opt){

case '?':

optind = 1;

return true;

break;

}

}

optind = 1;

return false;

}

void outputHelp(){

std::cout << H << std::endl;

}

void outputInfo(int &height, int &width){

std::cout << "Image height: " << height << ", image width: " << width << std::endl;

}

bool isPNG(const std::string& filename) {

const std::vector<unsigned char> pngSignature = {0x89, 0x50, 0x4E, 0x47, 0x0D, 0x0A, 0x1A, 0x0A};

std::ifstream file(filename, std::ios::binary);

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

std::cout << "Can't open file: " << filename << std::endl;

exit(41);

return false;

}

std::vector<unsigned char> fileSignature(8);

file.read(reinterpret\_cast<char\*>(fileSignature.data()), fileSignature.size());

if (file.gcount() != 8) {

std::cout << "File too small: " << filename << std::endl;

exit(41);

return false;

}

return fileSignature == pngSignature;

}

bool convertCoords(std::string stringCoords, int &x, int &y){

if(!std::regex\_match(stringCoords.c\_str(), std::regex("(-?[0-9]+).(-?[0-9]+)"))) return false;

x = atoi(stringCoords.c\_str());

int i = 0;

while(true){

if(stringCoords[i] == '.') break;

i++;

}

y = atoi(i + 1 + stringCoords.c\_str());

return true;

}

bool read\_png\_file(std::string filename, png\_bytep\* &new\_png, int &height, int &width, png\_byte &color\_type, png\_byte &bit\_depth){

if(isPNG(filename.c\_str()) == false) return false;

FILE \*fp = fopen(filename.c\_str(), "rb");

png\_structp png = png\_create\_read\_struct(PNG\_LIBPNG\_VER\_STRING, NULL, NULL, NULL);

if(!png) abort();

png\_infop info = png\_create\_info\_struct(png);

if(!info) abort();

if(setjmp(png\_jmpbuf(png))) abort();

png\_init\_io(png, fp);

png\_read\_info(png, info);

width = png\_get\_image\_width(png, info);

height = png\_get\_image\_height(png, info);

color\_type = png\_get\_color\_type(png, info);

bit\_depth = png\_get\_bit\_depth(png, info);

if(bit\_depth == 16)

png\_set\_strip\_16(png);

if(color\_type == PNG\_COLOR\_TYPE\_PALETTE)

png\_set\_palette\_to\_rgb(png);

if(color\_type == PNG\_COLOR\_TYPE\_GRAY && bit\_depth < 8)

png\_set\_expand\_gray\_1\_2\_4\_to\_8(png);

if(png\_get\_valid(png, info, PNG\_INFO\_tRNS))

png\_set\_tRNS\_to\_alpha(png);

if(color\_type == PNG\_COLOR\_TYPE\_RGB || color\_type == PNG\_COLOR\_TYPE\_GRAY || color\_type == PNG\_COLOR\_TYPE\_PALETTE)

png\_set\_filler(png, 0xFF, PNG\_FILLER\_AFTER);

if(color\_type == PNG\_COLOR\_TYPE\_GRAY || color\_type == PNG\_COLOR\_TYPE\_GRAY\_ALPHA)

png\_set\_gray\_to\_rgb(png);

png\_read\_update\_info(png, info);

if (new\_png) abort();

new\_png = (png\_bytep\*)malloc(sizeof(png\_bytep) \* height);

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

new\_png[y] = (png\_byte\*)malloc(png\_get\_rowbytes(png,info));

}

if (!new\_png) abort();

png\_read\_image(png, new\_png);

fclose(fp);

png\_destroy\_read\_struct(&png, &info, NULL);

return true;

}

void write\_png\_file(std::string filename, png\_bytep\* new\_png, int height, int width){

int y;

FILE \*fp = fopen(filename.c\_str(), "wb");

if(!fp) abort();

png\_structp png = png\_create\_write\_struct(PNG\_LIBPNG\_VER\_STRING, NULL, NULL, NULL);

if (!png) abort();

png\_infop info = png\_create\_info\_struct(png);

if (!info) abort();

if (setjmp(png\_jmpbuf(png))) abort();

png\_init\_io(png, fp);

png\_set\_IHDR(

png,

info,

width, height,

8,

PNG\_COLOR\_TYPE\_RGBA,

PNG\_INTERLACE\_NONE,

PNG\_COMPRESSION\_TYPE\_DEFAULT,

PNG\_FILTER\_TYPE\_DEFAULT

);

png\_write\_info(png, info);

if (!new\_png) abort();

png\_write\_image(png, new\_png);

png\_write\_end(png, NULL);

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

free(new\_png[y]);

}

free(new\_png);

fclose(fp);

png\_destroy\_write\_struct(&png, &info);

}

void set\_pixel\_color(int x, int y, int \*color, png\_bytep\* png){

png[y][(x \* 4)] = color[0];

png[y][(x \* 4)+1] = color[1];

png[y][(x \* 4)+2] = color[2];

}

void getRgb(std::string &color, int\* arrayRgb){

char colorRgb[color.size()+1];

char\* pointer;

strcpy(colorRgb, color.c\_str());

pointer = strtok(colorRgb, ".");

arrayRgb[0] = atoi(pointer);

pointer = strtok(NULL, ".");

arrayRgb[1] = atoi(pointer);

pointer = strtok(NULL, ".");

arrayRgb[2] = atoi(pointer);

}

bool checkValidCoord(int& x, int& y, int &height, int &width){

if(x >= width || x < 0) return false;

if(y >= height || y < 0) return false;

return true;

}

void drawCircle(int& x, int& y, int thickness, std::string& color,

png\_bytep\* &newPng, int &height, int &width){

int r = thickness / 2;

int arrayRgb[3];

getRgb(color, arrayRgb);

if(r < 1){

if(checkValidCoord(x, y, height, width)){

set\_pixel\_color(x, y, arrayRgb, newPng);

}

return;

}

int xc = r;

int yc = 0;

int P = 1 - r;

while (xc >= yc) {

int xMinusXC = x - xc;

int xPlusXC = x + xc;

int yPlusYC = y + yc;

int yMinusYC = y - yc;

int yPlusXC = y + xc;

int xPlusYC = x + yc;

int xMinusYC = x - yc;

int yMinusXC = y - xc;

for (int i = xMinusXC; (i <= xPlusXC)&&(i < width); i++) {

if ((yPlusYC >= 0)&&(i >= 0)&&(yPlusYC < height)){

set\_pixel\_color(i, yPlusYC, arrayRgb, newPng);

}

if ((yMinusYC >= 0)&&(i >= 0)&&(yMinusYC < height)) {

set\_pixel\_color(i, yMinusYC, arrayRgb, newPng);

}

}

for (int i = xMinusYC; (i <= xPlusYC)&&(i < width); i++) {

if ((yPlusXC >= 0)&&(i >= 0)&&(yPlusXC < height)) {

set\_pixel\_color(i, yPlusXC, arrayRgb, newPng);

}

if ((yMinusXC >= 0)&&(i >= 0)&&(yMinusXC < height)) {

set\_pixel\_color(i, yMinusXC, arrayRgb, newPng);

}

}

yc++;

if (P <= 0) {

P = P + 2 \* yc + 1;

} else {

xc--;

P = P + 2 \* (yc - xc) + 1;

}

}

}

void drawLine(int firstX, int firstY, int secondX, int secondY, int thickness,

std::string& color, png\_bytep\* &newPng, int &height, int &width){

int dx = abs(secondX - firstX);

int dy = abs(secondY - firstY);

int sx = firstX < secondX ? 1 : -1;

int sy = firstY < secondY ? 1 : -1;

int err = dx - dy;

int x = firstX;

int y = firstY;

while(x != secondX || y != secondY){

drawCircle(x, y, thickness, color, newPng, height, width);

int err2 = 2 \* err;

if(err2 > -dy){

err -= dy;

x += sx;

}

if(err2 < dx){

err += dx;

y += sy;

}

}

drawCircle(secondX, secondY, thickness, color, newPng, height, width);

}

void fillSquare(int leftX, int leftY, int rightX, int rightY, int thickness, png\_bytep\* &newPng, std::string &fillColor, int &height, int &width){

int arrayFillRgb[3];

getRgb(fillColor, arrayFillRgb);

for(int i = leftX; i <= rightX; i++){

for(int k = leftY; k <= rightY; k++){

if(checkValidCoord(i, k, height, width)){

set\_pixel\_color(i, k, arrayFillRgb, newPng);

}

}

}

}

void drawSquare(int leftX, int leftY, int side\_size, int thickness,

std::string& color, bool fill, std::string fillColor, png\_bytep\* &newPng, int &height, int &width){

int rightX = leftX + side\_size;

int rightY = leftY + side\_size;

if(fill == true) fillSquare(leftX, leftY, rightX, rightY, thickness, newPng, fillColor, height, width);

drawLine(leftX, leftY, rightX, leftY, thickness, color, newPng, height, width);

drawLine(rightX, leftY, rightX, rightY, thickness, color, newPng, height, width);

drawLine(leftX, rightY, rightX, rightY, thickness, color, newPng, height, width);

drawLine(leftX, leftY, leftX, rightY, thickness, color, newPng, height, width);

}

void processSquareCommand(int &height, int &width, png\_bytep\* &newPng, int argc, char\* argv[]){

int opt;

int keyIndex;

int leftX = -1;

int leftY = -1;

int side\_size = -1;

int thickness = -1;

std::string color = "";

std::string fillColor = "";

bool fill = false;

while(true){

opt = getopt\_long(argc, argv, "", squareKeys, &keyIndex);

if(opt == -1){

opt = 0;

break;

}

switch(opt){

case 'u':

convertCoords(optarg, leftX, leftY);

break;

case 'd':

side\_size = atoi(optarg);

break;

case 't':

if(std::regex\_match(optarg, std::regex("([0-9]+)"))) thickness = atoi(optarg);

break;

case 'c':

if(std::regex\_match(optarg, std::regex

("(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9]).(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9]).(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])")))

color = optarg;

break;

case 'f':

fill = true;

break;

case 'g':

if(std::regex\_match(optarg, std::regex

("(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9]).(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9]).(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])")))

fillColor = optarg;

break;

}

}

if(leftX == -1 || leftY == -1 || side\_size == -1 || thickness == -1 || color == ""){

std::cout << "Invalid data" << std::endl;

exit(41);

return;

}

if(fill == true && fillColor == ""){

std::cout << "Invalid data" << std::endl;

exit(41);

return;

}

drawSquare(leftX, leftY, side\_size, thickness, color, fill, fillColor, newPng, height, width);

}

void processExchangeCommand(int &height, int &width, png\_bytep\* & newPng, int argc, char\* argv[]){

int opt;

int keyIndex;

int leftX = -1;

int leftY = -1;

int rightX = -1;

int rightY = -1;

std::string exchangeType;

while(true){

opt = getopt\_long(argc, argv, "", exchangeKeys, &keyIndex);

if(opt == -1){

opt = 0;

break;

}

switch(opt){

case 'w':

convertCoords(optarg, rightX, rightY);

break;

case 'z':

convertCoords(optarg, leftX, leftY);

break;

case 'y':

exchangeType = optarg;

break;

}

}

if(leftX < 0 || leftY < 0 || rightX < 0 || rightY < 0 || rightX > width || rightY > height){

std::cout << "Invalid data" << std::endl;

exit(41);

return;

}

int midX = (leftX + rightX) / 2;

int midY = (leftY + rightY) / 2;

int exchange\_width = rightX - leftX + 1;

int exchange\_height = rightY - leftY + 1;

std::vector<png\_bytep> topLeft(exchange\_height / 2);

std::vector<png\_bytep> topRight(exchange\_height / 2);

std::vector<png\_bytep> bottomLeft(exchange\_height / 2);

std::vector<png\_bytep> bottomRight(exchange\_height / 2);

for (int i = 0; i < exchange\_height / 2; ++i) {

topLeft[i] = new png\_byte[exchange\_width / 2 \* 4];

topRight[i] = new png\_byte[exchange\_width / 2 \* 4];

bottomLeft[i] = new png\_byte[exchange\_width / 2 \* 4];

bottomRight[i] = new png\_byte[exchange\_width / 2 \* 4];

}

for (int y = 0; y < exchange\_height / 2; ++y) {

std::copy(newPng[leftY + y] + leftX \* 4, newPng[leftY + y] + midX \* 4, topLeft[y]);

std::copy(newPng[leftY + y] + midX \* 4, newPng[leftY + y] + rightX \* 4, topRight[y]);

std::copy(newPng[midY + y] + leftX \* 4, newPng[midY + y] + midX \* 4, bottomLeft[y]);

std::copy(newPng[midY + y] + midX \* 4, newPng[midY + y] + rightX \* 4, bottomRight[y]);

}

if (exchangeType == "clockwise") {

for (int y = 0; y < exchange\_height / 2; ++y) {

std::copy(bottomLeft[y], bottomLeft[y] + exchange\_width / 2 \* 4, newPng[leftY + y] + leftX \* 4);

std::copy(topLeft[y], topLeft[y] + exchange\_width / 2 \* 4, newPng[leftY + y] + midX \* 4);

std::copy(bottomRight[y], bottomRight[y] + exchange\_width / 2 \* 4, newPng[midY + y] + leftX \* 4);

std::copy(topRight[y], topRight[y] + exchange\_width / 2 \* 4, newPng[midY + y] + midX \* 4);

}

} else if (exchangeType == "counterclockwise") {

for (int y = 0; y < exchange\_height / 2; ++y) {

std::copy(topRight[y], topRight[y] + exchange\_width / 2 \* 4, newPng[leftY + y] + leftX \* 4);

std::copy(bottomRight[y], bottomRight[y] + exchange\_width / 2 \* 4, newPng[leftY + y] + midX \* 4);

std::copy(topLeft[y], topLeft[y] + exchange\_width / 2 \* 4, newPng[midY + y] + leftX \* 4);

std::copy(bottomLeft[y], bottomLeft[y] + exchange\_width / 2 \* 4, newPng[midY + y] + midX \* 4);

}

} else if (exchangeType == "diagonals") {

for (int y = 0; y < exchange\_height / 2; ++y) {

std::copy(bottomRight[y], bottomRight[y] + exchange\_width / 2 \* 4, newPng[leftY + y] + leftX \* 4);

std::copy(bottomLeft[y], bottomLeft[y] + exchange\_width / 2 \* 4, newPng[leftY + y] + midX \* 4);

std::copy(topRight[y], topRight[y] + exchange\_width / 2 \* 4, newPng[midY + y] + leftX \* 4);

std::copy(topLeft[y], topLeft[y] + exchange\_width / 2 \* 4, newPng[midY + y] + midX \* 4);

}

}

}

Rgb findMostFrequentColor(int &height, int &width, png\_bytep\* &newPng) {

std::map<int, std::map<int, std::map<int, int>>> colorCount;

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

png\_bytep row = newPng[y];

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

png\_bytep px = &(row[x \* 3]);

int r = px[0];

int g = px[1];

int b = px[2];

colorCount[r][g][b]++;

}

}

int maxCount = 0;

Rgb mostFrequentColor;

for (const auto &r : colorCount) {

for (const auto &g : r.second) {

for (const auto &b : g.second) {

if (b.second > maxCount) {

mostFrequentColor.R = r.first;

mostFrequentColor.G = g.first;

mostFrequentColor.B = b.first;

maxCount = b.second;

}

}

}

}

return mostFrequentColor;

}

void processFreqCommand(int &height, int &width,png\_bytep\* & newPng, int argc, char\* argv[]){

int opt;

int keyIndex;

std::string color = "";

int arrayColor[3];

while(true){

opt = getopt\_long(argc, argv, "", freqKeys, &keyIndex);

if(opt == -1){

opt = 0;

break;

}

switch(opt){

case 'l':

if(std::regex\_match(optarg, std::regex

("(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9]).(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9]).(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])"))){

color = optarg;

getRgb(color, arrayColor);

}

break;

}

}

if(color == ""){

std::cout << "Invalid data\n";

exit(41);

}

Rgb freqColor = findMostFrequentColor(height, width, newPng);

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

for(int k = 0; k < height; k++){

if(newPng[k][4\*i] == freqColor.R &&

newPng[k][4\*i + 1] == freqColor.G &&

newPng[k][4\*i + 2] == freqColor.B){

newPng[k][4\*i] = arrayColor[0];

newPng[k][4\*i + 1] = arrayColor[1];

newPng[k][4\*i + 2] = arrayColor[2];

}

}

}

}

bool processCommand(png\_bytep\* &newPng, int &height, int &width, png\_byte &colorType, png\_byte &bitDepth, int argc, char\* argv[], std::string &outputName, std::string &inputName){

int opt;

int keyIndex;

opterr = 0;

bool validFileName = false;

bool square = false;

bool exchange = false;

bool freq\_color = false;

bool printInfo = false;

int count = 0;

char\*\* argvCopy = new char\*[argc];

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

size\_t len = strlen(argv[i]) + 1;

argvCopy[i] = new char[len];

strncpy(argvCopy[i], argv[i], len);

}

if(argc == 1) outputHelp();

while(true){

opt = getopt\_long(argc, argvCopy, "ho:i", selectAction, &keyIndex);

if(opt == -1){

opt = 0;

break;

}

switch(opt){

case 'h':

outputHelp();

break;

case 'o':

if(checkValidName(optarg)) outputName = optarg;

break;

case 'i':

if(checkValidName(optarg)){

inputName = optarg;

if(read\_png\_file(inputName, newPng, height, width, colorType, bitDepth)) validFileName = true;

}

break;

case 'd':

printInfo = true;

break;

case 'r':

square = true;

count++;

break;

case 'x':

exchange = true;

count++;

break;

case 'c':

freq\_color = true;

count++;

break;

}

}

if(printInfo){

if(validFileName) outputInfo(height, width);

else{

std::cout << "Error: the input file name is invalid or the input file is corrupted" << std::endl;

exit(41);

}

}

optind = 1;

if(count > 1){

std::cout << "Error: too many arguments" << std::endl;

exit(41);

}

else{

if(validFileName&&square) processSquareCommand(height, width, newPng, argc, argv);

else if(validFileName&&exchange) processExchangeCommand(height, width, newPng, argc, argv);

else if(validFileName&&freq\_color) processFreqCommand(height, width, newPng, argc, argv);

else if(square || exchange || freq\_color){

std::cout << "Invalid input file name" << std::endl;

exit(41);

}

}

return validFileName;

};

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

png\_bytep\* newPng = nullptr;

int height;

int width;

png\_byte colorType;

png\_byte bitDepth;

char filename[] = "fd.png";

std::string outputName = "out.png";

std::string inputName = "";

std::cout << CW << std::endl;

if(findUnknownKey(argc, argv)){

exit(41);

return 0;

}

if(processCommand(newPng, height, width, colorType, bitDepth, argc, argv, outputName, inputName)){

if(outputName == inputName){

std::cout << "Error: the names of the input and output files are the same" << std::endl;

exit(41);

} else write\_png\_file(outputName, newPng, height, width);

return 0;

}

}

**приложение б**

**Тестирование**

Рисование квадрата: ./a.out --square --left\_up 220.300 --side\_size 70 --thickness 10 --color 255.20.147 --input FunnyMonkey.png --fill\_color 255.20.147 --fill

Рисунок 1. Входное изображение



Рисунок 2. Выходное изображение

Отражение области: ./a.out --exchange --left\_up 200.200 --right\_down 600.600 --exchange\_type clockwise --input SadMonkey.png

Рисунок 3. Входное изображение



Рисунок 4. Выходное изображение



Замена цвета: ./a.out --freq\_color --color 0.0.255 --input CuteMonkey.png

Рисунок 5. Входное изображение



Рисунок 6. Выходное изображение

