**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

КУРСОВАЯ РАБОТА

**по дисциплине «Программирование»**

**Тема: Обработка PNG изображения**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Силяев Р.А. |
| Преподаватель |  | Государкин Я.С. |

Санкт-Петербург

2024

## ЗАДАНИЕ

**НА КУРСОВУЮ РАБОТУ**

Студент: Силяев Руслан

Группа: 3343

Тема: Обработка PNG изображения

Условия задания (Вариант 4.16):

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)

**АННОТАЦИЯ**

Для создания программы, которая может рисовать квадраты, менять местами области, заменять один цвет на другой, на языке C была разработана программа, осуществляющая работу с png изображениями с помощью библиотеки libpng. Также необходимым условием было добавление CLI.

**ВВЕДЕНИЕ**

Целью курсовой работы является создание программы по обработке png изображений, овладение работы с png изображениями на языке C.

**1. ОПИСАНИЕ СТРУКТУРЫ ПРОГРАММЫ**

Описание структур:

1. *RGB* – структура, содержащая информацию о цвете.
2. *Parametrs* – структура, содержащая информацию о флагах
3. *Png* – структура, содержащая информацию о изображении
4. *Rectangle –* структура, содержащая информацию о координатах квадрата

Описание функций:

1. *int main(int argc, char\*\* argv)* – главная функция программы, из нее осуществляется вызов остальных функций.
2. *void throwError(const char\* message, int error)* – выводит ошибку, завершает работу программы.
3. *void checkExtraArguments(Parametrs\* par)* – проверяет наличие лишних флагов, аргументов.
4. *void checkColor(RGB color)* – проверяет правильность заданного цвета
5. *void checkFreq(Png\* image, Parametrs\* par)* – проверяет верность заданных флагов для *FreqColor*.
6. *void checkSquare(Png\* image, Parametrs\* par)* – проверяет верность заданных флагов для *Square*.
7. *void checkExchange(Png\* image, Parametrs\* par)* – проверяет верность заданных флагов для *Exchange*.
8. *Parametrs\* initParametrs(Parametrs\* par) –* инициализирует структуру *Parametrs* пустышками.
9. *void Help()* – вывод справочной информации о программе и достуаных флагах.
10. *void Info(Png\* image)* – вывод информации о изображении.
11. *char\*\* toCorrect(char\* opt, int size)* – приводит *opt* к правильному виду для записи в *Parametrs*.
12. *Parametrs\* CLI(int argc, char\*\* argv)* – исполняет роль *CLI*.
13. *void read\_png\_file(char \*file\_name, Png \*image)* – записывает изображение.
14. *void write\_png\_file(char \*file\_name, Png \*image)* – сохраняет изменения в изображении.
15. *void changeColor(png\_byte\* ptr, RGB color)* – в выбранном пикселе меняется цвет.
16. *void Square(Png\* image, Parametrs\* par)* – выполняется рисование квадрата с заданными характеристиками.
17. *int check(int x, int y, int W, int H)* – проверяет верность заданных координат.
18. *void drawLine(Png\* image, int x1, int y1, int x2, int y2, int line\_thickness, RGB color)* – рисует линию на изображении.
19. *void drawCircle(Png\* image ,int x0, int y0, int radius, RGB color)* – рисует круг на изображении.
20. *void Exchange(Png\* image, Parametrs\* par)* – делит заданную область на 4 части, меняет их местами в зависимости от выбранного типа замены.
21. *void swapAreas(Png\* image, Rectangle\* first, Rectangle\* second)* – меняет местами 2 области.
22. *void FreqColor(Png\* image, Parametrs\* par)* – заменяет самый часто встречаемый цвет на выбранный.

Разработанный код см. в приложении А.

**ТЕСТИРОВАНИЕ**

 Рисунок 1 – изображение для тестирования

1. Задание Square

Строка с флагами: ./cw --square --left\_up 100.100 --side\_size 50 --thickness 10 --color 255.0.0 --fill true --fill\_color 0.255.0 --input test.png --output out.png

Рисунок 2 – результат работы для задания 0w0

1. Задание Exchange

Строка с флагами: ./cw --exchange --left\_up 100.100 --right\_down 300.300 --exchange\_type diagonals --input test.png --output out.png

Рисунок 3 – результат работы для задания

1. Задание Freq\_color

Строка с флагами: ./cw --freq\_color --color 255.0.0 --input test.png --output out.png

Рисунок 4 – результат работы для задания

**ЗАКЛЮЧЕНИЕ**

В ходе курсовой работы была разработана программа на языке C, которая с помощью библиотеки libpng обрабатывает png изображения, в программу добавлен интерфейс CLI.

# Приложение А Исходный код программы

Название файла: checkErr.h

#ifndef CHECKERR\_H

#define CHECKERR\_H

#include "main.h"

#include <stdio.h>

#include <stdlib.h>

#include <png.h>

extern const char\* inputError;

extern const char\* outputError;

extern const char\* multipyFuncError;

extern const char\* argsError;

extern const char\* colorError;

extern const char\* typeError;

extern const char\* thicknessError;

extern const char\* fileTypeError;

void throwError(const char\* message, int error);

void checkExtraArguments(Parametrs\* par);

void checkColor(RGB color);

void checkFreq(Png\* image, Parametrs\* par);

void checkSquare(Png\* image, Parametrs\* par);

void checkExchange(Png\* image, Parametrs\* par);

#endif

Название файла: main.h

#ifndef MAIN\_H

#define MAIN\_H

#include <stdio.h>

#include <stdlib.h>

#include <png.h>

#include <unistd.h>

#include "struct.h"

#include "work.h"

#include "checkErr.h"

#include "Parametrs.h"

#endif

Название файла: Parametrs.h

#ifndef PARAMETRS\_H

#define PARAMETRS\_H

#include "main.h"

#include <getopt.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

Parametrs\* initParametrs(Parametrs\* par);

void Help();

void Info(Png\* image);

char\*\* toCorrect(char\* opt, int size);

Parametrs\* CLI(int argc, char\*\* argv);

#endif

Название файла: struct.h

#ifndef STRUCTURES\_H

#define STRUCTURES\_H

#include <png.h>

#include <stdbool.h>

typedef struct{

int height, width;

png\_byte color\_type;

png\_byte bit\_depth;

png\_structp png\_ptr;

png\_infop info\_ptr;

int number\_of\_passes;

png\_bytep \*row\_pointers;

} Png;

typedef struct{

int r;

int g;

int b;

} RGB;

typedef struct {

int x0;

int y0;

int x1;

int y1;

} Rectangle;

typedef struct {

char\* input;

char\* output;

bool info;

bool square;

int left;

int up;

int side\_size;

int thickness;

bool fill;

RGB fill\_color;

bool exchange;

int right;

int down;

int exchange\_type;

bool freq\_color;

RGB color;

} Parametrs;

#endif

Название файла: work.h

#ifndef WORK\_H

#define WORK\_H

#include "main.h"

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <unistd.h>

#include <png.h>

void read\_png\_file(char \*file\_name, Png \*image);

void write\_png\_file(char \*file\_name, Png \*image);

void changeColor(png\_byte\* ptr, RGB color);

void Square(Png\* image, Parametrs\* par);

int check(int x, int y, int W, int H);

void drawLine(Png\* image, int x1, int y1, int x2, int y2, int line\_thickness, RGB color);

void drawCircle(Png\* image ,int x0, int y0, int radius, RGB color);

void Exchange(Png\* image, Parametrs\* par);

void swapAreas(Png\* image, Rectangle\* first, Rectangle\* second);

void FreqColor(Png\* image, Parametrs\* par);

#endif

Название файла: checkErr.c

#include "../include/checkErr.h"

const char\* inputError = "Error: Invalid input<3"; //41

const char\* outputError = "Error: Invalid output<3"; //42

const char\* multipyFuncError = "Error: More than one function were called<3"; //43

const char\* argsError = "Error: Invalid args<3"; //44

const char\* colorError = "Error: Invalid color<3"; //45

const char\* typeError = "Error: Invalid type<3"; //46

const char\* thicknessError = "Error: Invalid thickness<3"; //47

const char\* fileTypeError = "Error: Invalid type of file<3"; //48

void throwError(const char\* message, int error) {

printf("%s\n", message);

exit(error);

}

void checkExtraArguments(Parametrs\* par){

int flag = 0;

if(par->square == true || par->side\_size != -1 || par->thickness != -1 || par->fill != false || par->fill\_color.r != -1){

flag = 1;

}

if(par->exchange == true || par->right != -1 || par->down != -1 || par->exchange\_type != -1){

if(flag != 0){

throwError(argsError, 44);

}

flag = 2;

}

if(par->freq\_color == true){

if(flag != 0){

throwError(argsError, 44);

}

flag = 3;

}

if(par->color.r != -1){

if(flag == 2){

throwError(argsError, 44);

}

}

if(par->left != -1){

if(flag == 3){

throwError(argsError, 44);

}

}

}

void checkColor(RGB color) {

if (color.r > 255 || color.g > 255 || color.b > 255){

throwError(colorError, 45);

}

if (color.r < 0 || color.g < 0 || color.b < 0){

throwError(colorError, 45);

}

}

void checkFreq(Png\* image, Parametrs\* par){

checkColor(par->color);

}

void checkSquare(Png\* image, Parametrs\* par){

if(par->left == -1 || par->up == -1 || par->side\_size == -1 || par->thickness == -1 || par->color.r == -1){

throwError(argsError, 44);

}

if(par->thickness <= 0){

throwError(thicknessError, 47);

}

checkColor(par->color);

if(par->fill){

checkColor(par->fill\_color);

}

}

void checkExchange(Png\* image, Parametrs\* par){

if(par->left == -1 || par->up == -1 || par->right == -1 || par->down == -1 || par->exchange\_type == -1){

throwError(argsError, 44);

}

if(par->left < 0){

par->left = 0;

}

if(par->up < 0){

par->up = 0;

}

if(par->right < par->left){

throwError(argsError, 44);

}

if(par->down < par->up){

throwError(argsError, 44);

}

if(par->exchange\_type < 0 || par->exchange\_type > 2){

throwError(typeError, 46);

}

}

Название файла: main.c

#include "../include/main.h"

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

Parametrs\* par = CLI(argc, argv);

checkExtraArguments(par);

Png image;

read\_png\_file(par->input, &image);

if (par->info)

Info(&image);

if (par->square) {

checkSquare(&image, par);

Square(&image, par);

}

if (par->exchange) {

checkExchange(&image, par);

Exchange(&image, par);

}

if (par->freq\_color) {

checkFreq(&image, par);

FreqColor(&image, par);

}

write\_png\_file(par->output, &image);

free(par);

return 0;

}

Название файла: Parametrs.c

#include "../include/Parametrs.h"

void Help(){

printf("Course work for option 4.16, created by Silyaev Ruslan.\n"

"Usage: ./cw [FLAGS]\n\n"

"Options:\n"

"-h --help: Вывод справочной информации.\n"

"-i --input: Изменение входного файла.\n"

"-o --output: Изменение выходного файла.\n"

"--info: Вывод информации изображения.\n\n"

"--square: Рисование квадрата.\n"

"--left\_up: Координаты левого верхнего угла.\n"

"--side\_size: Размер стороны.\n"

"--thickness: Толщина линий.\n"

"--color: Цвет линий.\n"

"--fill: Залит или нет.\n"

"--fill\_color: Цвет заливки.\n"

"--exchange: Поменять местами 4 куска области.\n"

"--left\_up: Координаты левого верхнего угла области.\n"

"--right\_down: Координаты правого нижнего угла области.\n"

"--exchange\_type: Способ обмена частей.\n"

"--freq\_color: Находит самый часто встречаемый цвет и заменяет его на другой заданный цвет.\n"

"--color: Цвет, в который надо перекрасить самый часто встречаемый цвет.\n"

);

}

void Info(Png\* image){

printf("Image settings:\n"

"Width: %d.\n"

"Height: %d.\n"

"Color type: %d.\n"

"Bit depth: %d\n",

image->width, image->height, image->color\_type, image->bit\_depth);

}

Parametrs\* initParametrs(Parametrs\* par) {

par->input = NULL;

par->output = NULL;

par->info = false;

par->square = false;

par->left = -1;

par->up = -1;

par->side\_size = -1;

par->thickness = -1;

par->color.r = par->color.g = par->color.b = -1;

par->fill = false;

par->fill\_color.r = par->fill\_color.g = par->fill\_color.b = -1;

par->exchange = false;

par->right = -1;

par->down = -1;

par->exchange\_type = -1;

par->freq\_color = false;

return par;

}

char\*\* toCorrect(char\* par, int size){

char\*\* correctpar = malloc(52 \* sizeof(char\*));

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

correctpar[i] = malloc(strlen(par));

}

int curlen = 0;

int curel = 0;

for (int i = 0; i < strlen(par); i++){

if (par[i-1] == '.' && i > 0){

correctpar[curel++][curlen-1] = '\0';

curlen = 0;

}

correctpar[curel][curlen++] = par[i];

}

correctpar[curel][curlen] = '\0';

return correctpar;

}

Parametrs\* CLI(int argc, char\*\* argv){

Parametrs\* par = malloc(sizeof(Parametrs));

initParametrs(par);

opterr=0;

const char\* short\_options = "hi:o:";

const struct option long\_options[] = {

{"help", 0, NULL, 'h'},

{"input", 1, NULL, 'i'},

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

{"info", 0, NULL, 52},

{"square", 0, NULL, 510},

{"left\_up", 1, NULL, 511},

{"side\_size", 1, NULL, 512},

{"thickness", 1, NULL, 513},

{"color", 1, NULL, 514},

{"fill", 0, NULL, 515},

{"fill\_color", 1, NULL, 516},

{"exchange", 0, NULL, 520},

{"right\_down", 1, NULL, 521},

{"exchange\_type", 1, NULL, 522},

{"freq\_color", 0, NULL, 530},

{0, 0, 0, 0}

};

int res;

char\*\* args;

while((res = getopt\_long(argc, argv, short\_options, long\_options, NULL)) != -1){

switch (res)

{

case 'h': //--help

Help();

break;

case 'i': //--input

par->input = optarg;

break;

case 'o': //--output

par->output = optarg;

break;

case 52: //--info

par->info = true;

break;

case 510:

if(par->exchange || par->freq\_color){

throwError(multipyFuncError, 43);

}

par->square = true;

break;

case 511:

args = toCorrect(optarg, 2);

par->left = strtol(args[0], NULL, 10);

par->up = strtol(args[1], NULL, 10);

break;

case 512:

par->side\_size = strtol(optarg, NULL, 10);

break;

case 513:

par->thickness = strtol(optarg, NULL, 10);

break;

case 514:

args = toCorrect(optarg, 3);

par->color.r = strtol(args[0], NULL, 10);

par->color.g = strtol(args[1], NULL, 10);

par->color.b = strtol(args[2], NULL, 10);

break;

case 515:

par->fill = true;

break;

case 516:

args = toCorrect(optarg, 3);

par->fill\_color.r = strtol(args[0], NULL, 10);

par->fill\_color.g = strtol(args[1], NULL, 10);

par->fill\_color.b = strtol(args[2], NULL, 10);

break;

case 520:

if(par->square || par->freq\_color){

throwError(multipyFuncError, 43);

}

par->exchange = true;

break;

case 521:

args = toCorrect(optarg, 2);

par->right = strtol(args[0], NULL, 10);

par->down = strtol(args[1], NULL, 10);

break;

case 522:

if (strcmp(optarg, "clockwise") == 0)

par->exchange\_type = 0;

else if (strcmp(optarg, "counterclockwise") == 0)

par->exchange\_type = 1;

else if (strcmp(optarg, "diagonals") == 0)

par->exchange\_type = 2;

else

throwError(typeError, 46);

break;

case 530:

if(par->square || par->exchange){

throwError(multipyFuncError, 43);

}

par->freq\_color = true;

break;

case '?':

throwError(argsError, 44);

break;

default:

break;

}

}

if (argc == 2 && (strcmp(argv[1], "--help") == 0 || strcmp(argv[1], "-h") == 0))

exit(0);

if (par->input == NULL && optind == argc - 1) {

par->input = malloc(strlen(argv[argc - 1]) + 1);

strncpy(par->input, argv[argc - 1], strlen(argv[argc - 1]) + 1);

}

if (par->input == NULL)

throwError(inputError, 41);

if (par->output == NULL) {

par->output = malloc(strlen("out.png") + 1);

par->output = "out.png";

}

if (strcmp(par->input, par->output) == 0)

throwError(outputError, 42);

return par;

}

Название файла: work.c

#include "../include/work.h"

void read\_png\_file(char \*file\_name, Png \*image) {

int x,y;

char header[8]; // 8 is the maximum size that can be checked

/\* open file and test for it being a png \*/

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

if (!fp){

printf("Invalid file name: %s\n", file\_name);

exit(10);

}

fread(header, 1, 8, fp);

if (png\_sig\_cmp(header, 0, 8)){

printf("File %s is not recognized as a PNG\n", file\_name);

exit(11);

}

/\* initialize stuff \*/

image->png\_ptr = png\_create\_read\_struct(PNG\_LIBPNG\_VER\_STRING, NULL, NULL, NULL);

if (!image->png\_ptr){

printf("png\_create\_read\_struct failed\n");

exit(10);

}

image->info\_ptr = png\_create\_info\_struct(image->png\_ptr);

if (!image->info\_ptr){

printf("png\_create\_info\_struct failed\n");

exit(10);

}

if (setjmp(png\_jmpbuf(image->png\_ptr))){

printf("Error during init\_io\n");

exit(10);

}

png\_init\_io(image->png\_ptr, fp);

png\_set\_sig\_bytes(image->png\_ptr, 8);

png\_read\_info(image->png\_ptr, image->info\_ptr);

image->width = png\_get\_image\_width(image->png\_ptr, image->info\_ptr);

image->height = png\_get\_image\_height(image->png\_ptr, image->info\_ptr);

image->color\_type = png\_get\_color\_type(image->png\_ptr, image->info\_ptr);

image->bit\_depth = png\_get\_bit\_depth(image->png\_ptr, image->info\_ptr);

image->number\_of\_passes = png\_set\_interlace\_handling(image->png\_ptr);

png\_read\_update\_info(image->png\_ptr, image->info\_ptr);

/\* read file \*/

if (setjmp(png\_jmpbuf(image->png\_ptr))){

printf("Error during read\_image\n");

exit(10);

}

image->row\_pointers = (png\_bytep \*) malloc(sizeof(png\_bytep) \* image->height);

for (y = 0; y < image->height; y++)

image->row\_pointers[y] = (png\_byte \*) malloc(png\_get\_rowbytes(image->png\_ptr, image->info\_ptr));

png\_read\_image(image->png\_ptr, image->row\_pointers);

fclose(fp);

}

void write\_png\_file(char \*file\_name, Png \*image) {

int x,y;

/\* create file \*/

FILE \*fp = fopen(file\_name, "wb");

if (!fp){

printf("File %s could not be opened\n", file\_name);

exit(10);

}

/\* initialize stuff \*/

image->png\_ptr = png\_create\_write\_struct(PNG\_LIBPNG\_VER\_STRING, NULL, NULL, NULL);

if (!image->png\_ptr){

printf("png\_create\_write\_struct failed\n");

exit(10);

}

image->info\_ptr = png\_create\_info\_struct(image->png\_ptr);

if (!image->info\_ptr){

printf("png\_create\_info\_struct failed\n");

exit(10);

}

if (setjmp(png\_jmpbuf(image->png\_ptr))){

printf("Error during init\_io\n");

exit(10);

}

png\_init\_io(image->png\_ptr, fp);

/\* write header \*/

if (setjmp(png\_jmpbuf(image->png\_ptr))){

printf("Error during writing header\n");

exit(10);

}

png\_set\_IHDR(image->png\_ptr, image->info\_ptr, image->width, image->height,

image->bit\_depth, image->color\_type, PNG\_INTERLACE\_NONE,

PNG\_COMPRESSION\_TYPE\_BASE, PNG\_FILTER\_TYPE\_BASE);

png\_write\_info(image->png\_ptr, image->info\_ptr);

/\* write bytes \*/

if (setjmp(png\_jmpbuf(image->png\_ptr))){

printf("Error during writing bytes\n");

exit(10);

}

png\_write\_image(image->png\_ptr, image->row\_pointers);

/\* end write \*/

if (setjmp(png\_jmpbuf(image->png\_ptr))){

printf("Error during end of write\n");

exit(10);

}

png\_write\_end(image->png\_ptr, NULL);

/\* cleanup heap allocation \*/

for (y = 0; y < image->height; y++)

free(image->row\_pointers[y]);

free(image->row\_pointers);

fclose(fp);

}

void changeColor(png\_byte\* tmp, RGB color) {

tmp[0] = color.r;

tmp[1] = color.g;

tmp[2] = color.b;

}

/\*Задание №1\*/

int check(int x, int y, int W, int H){

return x >= 0 && x < W && y >= 0 && y < H;

}

void drawCircle(Png\* image ,int x0, int y0, int radius, RGB color){

int D = 3 - 2 \* radius;

int x = 0;

int y = radius;

int W = image->width;

int H = image->height;

while (x <= y) {

if (check(x+x0,y+y0,W,H)) {

png\_byte\* ptr = &(image->row\_pointers[y+y0][(x+x0) \* 3]);

changeColor(ptr, color);

}

if (check(y+x0,x+y0,W,H)) {

png\_byte\* ptr = &(image->row\_pointers[x+y0][(y+x0) \* 3]);

changeColor(ptr, color);

}

if (check(-y+x0,x+y0,W,H)) {

png\_byte\* ptr = &(image->row\_pointers[x+y0][(-y+x0) \* 3]);

changeColor(ptr, color);

}

if (check(-x+x0, y+y0,W,H)) {

png\_byte\* ptr = &(image->row\_pointers[y+y0][(-x+x0) \* 3]);

changeColor(ptr, color);

}

if (check(-x+x0,-y+y0,W,H)) {

png\_byte\* ptr = &(image->row\_pointers[-y+y0][(-x+x0) \* 3]);

changeColor(ptr, color);

}

if (check(-y+x0,-x+y0,W,H)) {

png\_byte\* ptr = &(image->row\_pointers[-x+y0][(-y+x0) \* 3]);

changeColor(ptr, color);

}

if (check(y+x0,-x+y0,W,H)) {

png\_byte\* ptr = &(image->row\_pointers[-x+y0][(y+x0) \* 3]);

changeColor(ptr, color);

}

if (check(x+x0,-y+y0,W,H)) {

png\_byte\* ptr = &(image->row\_pointers[-y+y0][(x+x0) \* 3]);

changeColor(ptr, color);

}

if (D < 0) {

D += 4 \* x + 6;

x++;

} else {

D += 4 \* (x - y) + 10;

x++;

y--;

}

}

}

void drawLine(Png\* image, int x1, int y1, int x2, int y2, int line\_thickness, RGB color){

int dx = abs(x2 - x1);

int dy = abs(y2 - y1);

int sx = x1 < x2 ? 1 : -1;

int sy = y1 < y2 ? 1 : -1;

int err = dx - dy;

int h = image->height;

int w = image->width;

while(1){

if (y1 >= 0 && y1 <= h && x1 >= 0 && x1 <= w){

if (line\_thickness == 1){

png\_byte\* ptr = &(image->row\_pointers[y1][x1 \* 3]);

changeColor(ptr, color);

}

}

if(line\_thickness > 1 && x1 - (line\_thickness/2) < w && y1 - (line\_thickness/2) < h && x1 + (line\_thickness/2) >= 0 && y1 + (line\_thickness/2) >= 0){

drawCircle(image, x1, y1, line\_thickness/2 ,color);

}

if (x1 == x2 && y1 == y2){

break;

}

int e2 = 2 \* err;

if (e2 > -dy) {

err -= dy;

x1 += sx;

}

if (e2 < dx) {

err += dx;

y1 += sy;

}

}

}

void Square(Png\* image, Parametrs\* par){

int x0=par->left;

int x1=x0 + par->side\_size;

int y0=par->up;

int y1=y0 + par->side\_size;

drawLine(image, x0, y0, x1, y0, par->thickness, par->color);

drawLine(image, x0, y1, x1, y1, par->thickness, par->color);

drawLine(image, x1, y0, x1, y1, par->thickness, par->color);

drawLine(image, x0, y0, x0, y1, par->thickness, par->color);

if(par->fill){

int i = y0+par->thickness/2;

while(i > 0 && i < image->height && i <= y1-par->thickness/2){

png\_byte\* row = image->row\_pointers[i];

int j = x0+par->thickness/2;

while (j > 0 && j < image->width && j <= x1-par->thickness/2){

png\_byte\* ptr = &(row[j \* 3]);

changeColor(ptr, par->fill\_color);

j++;

}

i++;

}

}

}

/\*Задание №2\*/

void swapAreas(Png\* image, Rectangle\* first, Rectangle\* second) {

RGB canvas1[first->y1-first->y0][first->x1-first->x0];

RGB canvas2[second->y1-second->y0][second->x1-second->x0];

for (int y = first->y0; y < first->y1; y++) {

png\_byte\* row = image->row\_pointers[y];

for (int x = first->x0; x < first->x1; x++) {

png\_byte\* ptr = &(row[x \* 3]);

canvas1[y-first->y0][x-first->x0].r = ptr[0];

canvas1[y-first->y0][x-first->x0].g = ptr[1];

canvas1[y-first->y0][x-first->x0].b = ptr[2];

}

}

for (int y = second->y0; y < second->y1; y++) {

png\_byte\* row = image->row\_pointers[y];

for (int x = second->x0; x < second->x1; x++) {

png\_byte\* ptr = &(row[x \* 3]);

canvas2[y-second->y0][x-second->x0].r = ptr[0];

canvas2[y-second->y0][x-second->x0].g = ptr[1];

canvas2[y-second->y0][x-second->x0].b = ptr[2];

}

}

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

png\_byte\* row = image->row\_pointers[y];

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

png\_byte\* ptr = &(row[x \* 3]);

if (y >= first->y0 && y < first->y1 && x >= first->x0 && x < first->x1) {

ptr[0] = canvas2[y-first->y0][x-first->x0].r;

ptr[1] = canvas2[y-first->y0][x-first->x0].g;

ptr[2] = canvas2[y-first->y0][x-first->x0].b;

}

if (y >= second->y0 && y < second->y1 && x >= second->x0 && x < second->x1) {

ptr[0] = canvas1[y-second->y0][x-second->x0].r;

ptr[1] = canvas1[y-second->y0][x-second->x0].g;

ptr[2] = canvas1[y-second->y0][x-second->x0].b;

}

}

}

}

void Exchange(Png\* image, Parametrs\* par){

int width = par->right-par->left;

int height = par->down-par->up;

if(width%2 != 0)

par->right = par->right - 1;

width = par->right-par->left;

if(height%2 != 0)

par->down = par->down - 1;

height = par->down-par->up;

Rectangle rectLeftUp;

Rectangle rectLeftDown;

Rectangle rectRightUp;

Rectangle rectRightDown;

rectLeftUp.x0 = par->left;

rectLeftUp.y0 = par->up;

rectLeftUp.x1 = par->right - (width / 2);

rectLeftUp.y1 = par->down - (height / 2);

rectLeftDown.x0 = par->left;

rectLeftDown.y0 = par->down - (height / 2);

rectLeftDown.x1 = par->right - (width / 2);

rectLeftDown.y1 = par->down;

rectRightUp.x0 = par->right - (width / 2);

rectRightUp.y0 = par->up;

rectRightUp.x1 = par->right;

rectRightUp.y1 = par->down - (height / 2);

rectRightDown.x0 = par->right - (width / 2);

rectRightDown.y0 = par->down - (height / 2);

rectRightDown.x1 = par->right;

rectRightDown.y1 = par->down;

switch (par->exchange\_type)

{

case 0:{ //clockwise

swapAreas(image, &rectLeftUp, &rectLeftDown);

swapAreas(image, &rectLeftDown, &rectRightDown);

swapAreas(image, &rectRightDown, &rectRightUp);

break;

}

case 1:{ //counterclockwise

swapAreas(image, &rectLeftUp, &rectRightUp);

swapAreas(image, &rectRightUp, &rectRightDown);

swapAreas(image, &rectRightDown, &rectLeftDown);

break;

}

case 2:{ //diagonals

swapAreas(image, &rectLeftUp, &rectRightDown);

swapAreas(image, &rectLeftDown, &rectRightUp);

break;

}

default:

break;

}

}

/\*Задание №3\*/

void FreqColor(Png\* image, Parametrs\* par){

int color = 3;

int\*\*\* colors = calloc(256, sizeof(int\*\*));

for(int i = 0; i < 256; i++){

colors[i] = calloc(256, sizeof(int\*));

for(int j = 0; j < 256; j++){

colors[i][j] = calloc(256, sizeof(int));

}

}

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

png\_byte\* row = image->row\_pointers[y];

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

png\_byte\* ptr = &(row[x \* color]);

colors[ptr[0]][ptr[1]][ptr[2]]++;

}

}

RGB old\_color;

int maxColor = 0;

for(int r = 0; r < 256; r++){

for(int g = 0; g < 256; g++){

for(int b = 0; b < 256; b++){

if(colors[r][g][b] > maxColor){

maxColor = colors[r][g][b];

old\_color.r = r;

old\_color.g = g;

old\_color.b = b;

}

}

}

}

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

png\_byte\* row = image->row\_pointers[y];

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

png\_byte\* ptr = &(row[x \* color]);

if (old\_color.r == ptr[0] && old\_color.g == ptr[1] && old\_color.b == ptr[2])

changeColor(ptr, par->color);

}

}

}