**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

КУРСОВАЯ РАБОТА

**по дисциплине «Программирование»**

**Тема: Программа на языке C для обработки изображений PNG**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Коршков А.А. |
| Преподаватель |  | Государкин Я.С. |

Санкт-Петербург

2024

## ЗАДАНИЕ

**НА КУРСОВУЮ РАБОТУ**

Студент: Коршков Александр

Группа: 3343

Тема: Обработка PNG изображения

Условия задания (Вариант 4.20):

Программа должна иметь следующую функции по обработке изображений:

1. Рисование отрезка. Флаг для выполнения данной операции: `--line`. Отрезок определяется:

* координатами начала. Флаг `--start`, значение задаётся в формате `x.y`, где x – координата по x, y – координата по y
* координатами конца. Флаг `--end` (аналогично флагу `--start`)
* цветом. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет)
* толщиной. Флаг `--thickness`. На вход принимает число больше 0

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

1. Рисование пентаграммы в круге. Флаг для выполнения данной операции: `--pentagram`. Пентаграмма определяется:

* координатами ее центра и радиусом. Флаги `--center` и `--radius`. Значение флаг `--center` задаётся в формате `x.y`, где x – координата по оси x, y – координата по оси y. Флаг `--radius` На вход принимает число больше 0
* толщиной линий и окружности. Флаг `--thickness`. На вход принимает число больше 0
* цветом линий и окружности. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет)

Предполагаемый объем пояснительной записки:

Не менее 33 страниц.

Дата выдачи задания: 18.03.2024

Дата сдачи реферата: 23.05.2024

Дата защиты реферата: 23.05.2024

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Коршков А.А. |
| Преподаватель |  | Государкин Я.С. |

**АННОТАЦИЯ**

В ходе курсовой работы реализована программа, осуществляющая обработку PNG изображения. Для взаимодействия с программой реализован интерфейс командной строки (CLI).

Программа реализует следующие функции: рисование линии по заданным координатам, цвету и толщине, отражение заданной области по оси «x» или «y», рисование пентаграммы в окружности с заданными координатами центра, радиуса и толщины линии. Сборка проекта осуществляется с помощью утилиты make.

**ВВЕДЕНИЕ**

Цель работы: разобрать структуру PNG изображать, научиться работать с PNG изображением на языке программирования C с помощью библиотеки libpng, реализовать программу, реализующую несколько функций по обработке изображения, его считыванию и записи, взаимодействие с которой должно осуществляться с помощью интерфейса командной строки.

**1. ОПИСАНИЕ СТРУКТУРЫ ПРОГРАММЫ**

Описание структур:

1. *Png* – структура, содержащая данные для работы с PNG изображением: ширина (weight), высота (height), тип цветопередачи (color\_type, программа работает с RGB), количество бит на пикселей (bit\_depth), количество проходов по изображению для обработки (number\_of\_passes), указатель на массив указателей на строки пикселей изображения (row\_pointers).
2. *info\_file* – структура, содержащая информацию о входном (input\_file) и выходном (output\_file) изображениях и переменную info (для показа информации изображении).
3. *RGB* – структура, которая представляет собой цвет, кодируемый тремя компонентами: *r* (красный), *g* (зеленый), *b* (синий).
4. *Point* – структура, содержащая координаты пикселя. Используется для рисования линии и пентаграммы.
5. *info\_line* – структура, содержащую информацию о линии: p0 p1 – координаты линии, color – цвет, thickness – толщина, p – показывает, что надо рисовать линию.
6. *info\_mirror* – структура, содержащую информацию о линии: p0 p1 – координаты левого верхнего и правого нижнего углов прямоугольной области, axis – ось отражения, p – показывает, что надо отразить область.
7. *info\_ pentagram* – структура, содержащую информацию о линии: center – координаты центра, radius – радиус окружности, color – цвет, thickness – толщина линий пентаграммы и окружности, p – показывает, что надо рисовать пентаграмму.

Описание функций:

1. *int is\_digit(char \*line)*– проверка строки, что это число.
2. *void free\_png(Png \*png);* – освобождает память из-под изображения.
3. *void set\_pixel(int x, int y, RGB color, Png \*png)*–устанавливает цвет пикселя в заданных кординатах.
4. *RGB get\_color(int x, int y, Png \*png)* – возвращает цвет пикселя.
5. *int get\_size\_pixel(Png \*png)*– вовращает размер пикселя изображения.
6. *void print\_info(char \*input\_file, Png \*png)* – выводит информацию по изображению.
7. *void print\_help()*–выводит справку.
8. *void read\_png\_file(char \*file\_name, Png \*image)* – чтение изображения из файла.
9. *void write\_png\_file(char \*file\_name, Png \*image))* запись изображения в файл.
10. *void draw\_mirror(int x0, int y0, int x1, int y1, char axis, Png \*png);*– вызывает функцию отражения области.
11. *int check\_mirror(info\_mirror \*mirror)*– проверяет, что все параметры отражения присутствуют.
12. *void set\_left\_up(char \*xy, info\_mirror \*mirror)* – задаёт верхний левый угол.
13. *void set\_right\_down(char \*xy, info\_mirror \*mirror)* – задаёт правый нижний угол.
14. *void set\_axis(char \*axis, info\_mirror \*mirror)* – задаёт ось отражения.
15. *void draw\_line(int x0, int y0, int x1, int y1, int thickness, RGB color, Png \*png)* – вызывает функцию рисования линии.
16. *int check\_line(info\_line \*line)*– проверяет, что все параметры линии присутствуют.
17. *int main(int argc, char\* argv[])* – главная функция программы, осуществляет обработку аргументов командной строки и вызывающую необходимые функции.
18. *void set\_start\_cords(char \*xy, info\_line \*line)* – задаёт начальные координаты линии.
19. *void set\_end\_cords(char \*xy, info\_line \*line)* – задаёт конечные координаты линии.
20. *void set\_color\_line(char \*rgb, info\_line \*line);* – задаёт цвет линии.
21. *void set\_thickness\_line(char \*thickness, info\_line \*line)*– задаёт толщину линии.
22. *void set\_square(int x0, int y0, int size, RGB color, Png \*png)* – рисует квадрат.
23. *void draw\_pentagram(int x0, int y0, int radius, int thickness, RGB color, Png \*png)* – вызывает функцию рисования пентаграммы в круге.
24. *int check\_pentagram(info\_pentagram \*pentagram)* – проверяет, что все параметры пентаграммы присутствуют.
25. *void set\_color\_pentagram(char \*color, info\_pentagram \*pentagram)* – задаёт цвет линий пентаграммы и окружности.
26. *void set\_center(char \*center, info\_pentagram \*pentagram)* – задаёт центр окружности и пентаграмма.
27. *void set\_radius(char \*radius, info\_pentagram \*pentagram)* – задаёт радиус.
28. *void draw\_circle(int x0, int y0, int radius, int thickness, RGB color, Png \*png)* – рисует окружность с толщиной.

Созданная программа разделена на модули, что хорошо сказывается на масштабируемости кода и возможности развития программы в целом. Все функции распределены по соответствующим файлам, отвечающим за какой-либо аспект действий. Программа собирается с использованием Makefile, что обеспечивает как легкость в редактировании зависимостей между модулями, так и удобство в управлении процессом компиляции. Разработанный программный код см. в приложении А.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы была создана программа на языке программирования C, осуществляющая обработку PNG изображения. В зависимости от выбранных опций, программа выполняет одну из поддерживаемых функций. Сборка проекта осуществляется с помощью утилиты make. Запуск программы и выбор опций осуществляется через CLI (command line interface).

# Приложение А Исходный код программы

Название файла: main.c

#include "../include/png\_objects.h" /\* объекты и структуры для PNG \*/

#include "../include/read\_write.h" /\* функции чтения и записи файла \*/

#include "../include/print\_help\_info.h" /\* функции вывода информации о программе и изображении\*/

#include "../include/draw\_line.h" /\* функции рисования линии \*/

#include "../include/draw\_pentagram.h" /\* функции рисования пентаграммы \*/

#include "../include/draw\_mirror.h" /\* функции отражения области \*/

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

/\* вывод информации о курсовой работе и авторе \*/

puts("Course work for option 4.20, created by Alexander Korshkov.");

/\* проверка, переданы ли аргументы \*/

if (argc == 1) {

print\_help();

return 0;

}

opterr = 0;

/\* структура, содержащая длинная флаги \*/

struct option long\_opt[] = {

{"help", no\_argument, NULL, 'h'}, /\* справка \*/

{"info", no\_argument, NULL, '!'}, /\* информация о файле \*/

{"input", required\_argument, NULL, 'i'}, /\* входной файл \*/

{"output", required\_argument, NULL, 'o'}, /\* выходной файл \*/

{"line", no\_argument, NULL, 300}, /\* рисование линии \*/

{"start", required\_argument, NULL, 301}, /\* начальная точка \*/

{"end", required\_argument, NULL, 302}, /\* конечная точка \*/

{"color", required\_argument, NULL, 303}, /\* цвет линии \*/

{"thickness", required\_argument, NULL, 304}, /\* толщина линии \*/

{"mirror", no\_argument, NULL, 400}, /\* отражение \*/

{"axis", required\_argument, NULL, 401}, /\* ось отражения \*/

{"left\_up", required\_argument, NULL, 402}, /\* верхний левый угол области \*/

{"right\_down", required\_argument, NULL, 403}, /\* нижний правый угол области \*/

{"pentagram", no\_argument, NULL, 404}, /\* рисование пентаграммы в круге\*/

{"center", required\_argument, NULL, 405}, /\* центр круга \*/

{"radius", required\_argument, NULL, 406}, /\* радиус круга \*/

/\* color \*/ /\* цвет линии пентаграмма \*/

/\* thickness \*/ /\* толщина линии пентаграмма \*/

{0, 0, 0, 0}

};

/\* информация о фото \*/

Png input\_image;

/\* информация о входном и выходном файле \*/

info\_file information = {.output\_file = "out.png", .info = 0};

/\* информация о линии, области для отражения и рисования пентаграммы \*/

info\_line line = {.p = 0, .p0 = {.x = -1, .y = -1}, .p1 = {.x = -1, .y = -1},

.thickness = -1, .color = {.r = -1, .g = -1, .b = -1}};

info\_mirror mirror = {.p = 0, .p0 = {.x = -1, .y = -1}, .p1 = {.x = -1, .y = -1}, .axis = 'n'};

info\_pentagram pentagram = {.p = 0, .center = {.x = -1, .y = -1}, .radius = -1};

/\* цикл, обрабатывающий флаги и аргументы \*/

int opt;

while ((opt = getopt\_long(argc, argv, "hi:o:", long\_opt, NULL)) != -1) {

switch (opt) {

/\* справка \*/

case 'h': {

print\_help();

return 0;

}

/\* информация о файле \*/

case '!': {

information.info = 1;

break;

}

/\* входной файл \*/

case 'i': {

information.input\_file = optarg;

break;

}

/\* выходной файл \*/

case 'o': {

information.output\_file = optarg;

break;

}

/\* рисование линии \*/

case 300: {

line.p = 1;

break;

}

/\* начальная координата линии \*/

case 301: {

set\_start\_cords(optarg, &line);

break;

}

/\* конечная координата линии \*/

case 302: {

set\_end\_cords(optarg, &line);

break;

}

/\* цвет линии (для пентаграммы тоже)\*/

case 303: {

char\* color = strdup(optarg);

set\_color\_line(optarg, &line);

set\_color\_pentagram(color, &pentagram);

break;

}

/\* толщина линии (для пентаграммы тоже) \*/

case 304: {

char\* thickness = strdup(optarg);

set\_thickness\_line(optarg, &line);

set\_thickness\_pentagram(thickness, &pentagram);

break;

}

/\* отражение области \*/

case 400: {

mirror.p = 1;

break;

}

/\* ось отражённой области \*/

case 401: {

set\_axis(optarg, &mirror);

break;

}

/\* верхний левый угол области \*/

case 402: {

set\_left\_up(optarg, &mirror);

break;

}

/\* нижний правый угол области \*/

case 403: {

set\_right\_down(optarg, &mirror);

break;

}

/\* рисование пентаграммы в круге \*/

case 404: {

pentagram.p = 1;

break;

}

/\* центр круга \*/

case 405: {

set\_center(optarg, &pentagram);

break;

}

/\* радиус круга \*/

case 406: {

set\_radius(optarg, &pentagram);

break;

}

default: {

puts("Unknown option!");

exit(47);

}

}

}

if (line.p + mirror.p + pentagram.p == 0) {

printf("You don't specify any function!\n");

exit(40);

}

if (line.p + mirror.p + pentagram.p != 1) {

printf("You specify more than one function!\n");

exit(40);

}

/\* проверка, что имя файла было как-то передано (через опцию или в конце отдельным аргументом) \*/

if (!information.input\_file) {

if (optind < argc) {

information.input\_file = argv[argc - 1];

} else {

printf("You don't specify input file!\n");

exit(40);

}

}

/\* чтение и получение информации об изображении \*/

read\_png\_file(information.input\_file, &input\_image);

/\* проверка, ввёл ли пользователь флаг --info \*/

if (information.info == 1) {

print\_info(information.input\_file, &input\_image);

return 0;

}

/\* проверка на корректность и количество введённых параметров линии \*/

if (line.p) {

if (check\_line(&line))

draw\_line(line.p0.x, line.p0.y, line.p1.x, line.p1.y, line.thickness, line.color, &input\_image);

else {

printf("You don't specify all parameters for line!\n");

free\_png(&input\_image);

exit(40);

}

}

/\* проверка на корректность и количество введённых параметров отражённой области \*/

if (mirror.p) {

if (check\_mirror(&mirror))

draw\_mirror(mirror.p0.x, mirror.p0.y, mirror.p1.x, mirror.p1.y, mirror.axis, &input\_image);

else {

printf("You don't specify all parameters for mirror!\n");

free\_png(&input\_image);

exit(40);

}

}

/\* проверка на корректность и количество введённых параметров пентаграммы в круге \*/

if (pentagram.p) {

if (check\_pentagram(&pentagram))

draw\_pentagram(pentagram.center.x, pentagram.center.y, pentagram.radius, pentagram.thickness, pentagram.color, &input\_image);

else {

printf("You don't specify all parameters for pentagram!\n");

free\_png(&input\_image);

exit(40);

}

}

/\* запись файла \*/

write\_png\_file(information.output\_file, &input\_image);

return 0;

}

Название файла: png\_objects.h

#ifndef PNG\_OBJECTS

#define PNG\_OBJECTS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include <png.h>

#include <math.h>

#include <getopt.h>

typedef struct Point {

int x;

int y;

} Point;

typedef struct RGB {

int r;

int g;

int b;

} RGB;

typedef struct Png {

int width;

int height;

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 info\_file {

int info;

char \*input\_file;

char \*output\_file;

} info\_file;

typedef struct info\_line {

Point p0;

Point p1;

RGB color;

int p;

int thickness;

} info\_line;

typedef struct info\_mirror {

Point p0;

Point p1;

int p;

char axis;

} info\_mirror;

typedef struct info\_pentagram {

Point center;

RGB color;

int p;

int radius;

int thickness;

} info\_pentagram;

#endif

Название файла: print\_help\_info.h

#ifndef PRINT\_HELP\_INFO\_H

#define PRINT\_HELP\_INFO\_H

#include "png\_objects.h"

void print\_info(char \*input\_file, Png \*png);

void print\_help();

#endif

Название файла: print\_help\_info.c

#include "../include/print\_help\_info.h"

#include "../include/add\_operations.h"

/\* вывод информации об изображении \*/

void print\_info(char \*input\_file, Png \*png) {

printf("Information about PNG file \"%s\":\n", input\_file);

printf("Width: %d\n", png->width);

printf("Height: %d\n", png->height);

printf("Color type: %d\n", png->color\_type);

printf("Bit depth: %d\n", png->bit\_depth);

free\_png(png); /\* освобождение памяти \*/

}

/\* Вывод информации о программе, как её использовать \*/

void print\_help() {

puts("Usage: cw [OPTIONS]\n"

"OPTIONS:\n"

"-h, --help - print information about flags\n"

"--info - print information about PNG file\n"

"-i, --input <argument> - set input file name\n"

" -o, --output <argument> - set output file name\n"

"--line <arguments> - set line drawing mode\n"

"\t--start <x.y> - set start point of line\n"

"\t--end <x.y> - set end point of line\n"

"\t--color <r.g.b> - set color of line\n"

"\t --thickness <argument> - set thickness of line\n"

"--mirror <arguments> - set mirror mode\n"

"\t --axis <argument> - set axis of mirror (must me 'x' or 'y')\n"

"\t--left\_up <x.y> - set left\_up point of mirror's edge\n"

"\t--right\_up <x.y> - set right\_up point of mirror's edge\n"

"--pentagram <arguments> - set pentagram in circle mode\n"

"\t--center <x.y> - set center point of pentagram\n"

"\t--radius <argument> - set radius of pentagram\n"

"\t--color <r.g.b> - set color of pentagram\n"

"\t--thickness <argument> - set thickness of pentagram");

}

Название файла: read\_write.h

#ifndef READ\_WRITE\_C

#define READ\_WRITE\_C

#include <stdio.h>

#include <stdlib.h>

#include <png.h>

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

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

#endif

Название файла: read\_write.c

#ifndef READ\_WRITE\_C

#define READ\_WRITE\_C

#include <stdio.h>

#include <stdlib.h>

#include <png.h>

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

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

#endif

#include "../include/png\_objects.h"

#include "../include/read\_write.h"

#include "../include/add\_operations.h"

/\* функция чтения из файла \*/

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

{

int x, y; /\* размеры изображения \*/

FILE \*fp = fopen(file\_name, "rb"); /\* открытие файла в бинарном режиме \*/

if (!fp)

{

printf("File \"%s\" not found!\n", file\_name); /\* файл не найден/не существует \*/

exit(43);

}

char header[8]; /\* заголовок файла \*/

fread(header, 1, 8, fp); /\* чтение заголовка \*/

/\* проверка заголовка \*/

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

{

printf("Probably, file \"%s\" is not a png!\n", file\_name);

fclose(fp); /\* закрытие файла \*/

exit(42);

}

/\* создание структуры для чтения изображения\*/

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

if (!image->png\_ptr)

{

printf("Error in png structure!\n"); /\* структура не создана \*/

fclose(fp); /\* закрытие файла \*/

exit(44);

}

/\* создание структуры с информацией об изображении \*/

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

if (!image->info\_ptr)

{

printf("Error in png info-structure (information is broken)!\n"); /\* структура не создана \*/

png\_destroy\_read\_struct(&image->png\_ptr, NULL, NULL); /\* удаление структур \*/

fclose(fp); /\* закрытие файла \*/

exit(44);

}

/\* неизвестная ошибка \*/

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

{

printf("Unknown Error!\n");

png\_destroy\_read\_struct(&image->png\_ptr, &image->info\_ptr, NULL); /\* удаление структур \*/

fclose(fp);

exit(49);

}

png\_init\_io(image->png\_ptr, fp); /\* открытие потока ввода/вывода \*/

png\_set\_sig\_bytes(image->png\_ptr, 8); /\* установка заголовка \*/

png\_read\_info(image->png\_ptr, image->info\_ptr); /\* чтение информации об изображении \*/

image->width = png\_get\_image\_width(image->png\_ptr, image->info\_ptr); /\* получение ширины изображения \*/

image->height = png\_get\_image\_height(image->png\_ptr, image->info\_ptr); /\* получение высоты изображения \*/

image->color\_type = png\_get\_color\_type(image->png\_ptr, image->info\_ptr); /\* получение типа изображения \*/

if (image->color\_type != PNG\_COLOR\_TYPE\_RGB && image->color\_type != PNG\_COLOR\_TYPE\_RGBA) {

puts("The color type of image isn't RGB or RGBA.");

png\_destroy\_read\_struct(&image->png\_ptr, &image->info\_ptr, NULL);

exit(46);

}

image->bit\_depth = png\_get\_bit\_depth(image->png\_ptr, image->info\_ptr); /\* получение глубины изображения (бит на пиксель) \*/

image->number\_of\_passes = png\_set\_interlace\_handling(image->png\_ptr); /\* получение количества проходов для инициализации \*/

png\_read\_update\_info(image->png\_ptr, image->info\_ptr); /\* чтение информации об изображении \*/

image->row\_pointers = (png\_bytep \*)malloc(sizeof(png\_bytep) \* image->height); /\* выделение памяти под указатели на строки \*/

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

{

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

}

png\_read\_image(image->png\_ptr, image->row\_pointers); /\* чтение изображения \*/

fclose(fp); /\* закрытие файла \*/

}

/\* функция записи в файл \*/

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

{

int x, y; /\* размеры изображения \*/

FILE \*fp = fopen(file\_name, "wb"); /\* открытие файла для записи \*/

if (!fp)

{

printf("The file \"%s\" cannot be created!\n", file\_name); /\* файл не может быть создан \*/

exit(43);

}

/\* создание структуры для записи \*/

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

if (!image->png\_ptr)

{

printf("It\'s impossible to write the structure of file \"%s\"!\n", file\_name); /\* структура не создана \*/

fclose(fp); /\* закрытие файла \*/

exit(44);

}

/\* создание структуры с информацией об изображении \*/

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

if (!image->info\_ptr)

{

printf("It\'s impossible to write the information of file \"%s\"!\n", file\_name);

png\_destroy\_write\_struct(&image->png\_ptr, NULL);

fclose(fp);

exit(44);

}

/\* неизвестная ошибка \*/

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

{

png\_destroy\_write\_struct(&image->png\_ptr, &image->info\_ptr);

fclose(fp);

exit(49);

}

png\_init\_io(image->png\_ptr, fp); /\* открытие потока для записи \*/

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); /\* установка заголовка \*/

/\*запись информации об изображении \*/

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

/\* ошибка при записи информации об изображении \*/

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

{

png\_destroy\_write\_struct(&image->png\_ptr, &image->info\_ptr); /\* удаление структур \*/

fclose(fp); /\* закрытие файла \*/

exit(45);

}

/\* запись изображения \*/

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

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

{

png\_destroy\_write\_struct(&image->png\_ptr, &image->info\_ptr);

fclose(fp);

exit(45);

}

png\_write\_end(image->png\_ptr, NULL); /\* завершение записи\*/

free\_png(image);

fclose(fp);

}

Название файла: draw\_mirror.h

#ifndef DRAW\_MIRROR\_H

#define DRAW\_MIRROR\_H

#include "../include/add\_operations.h"

void draw\_mirror(int x0, int y0, int x1, int y1, char axis, Png \*png);

int check\_mirror(info\_mirror \*mirror);

void set\_left\_up(char \*xy, info\_mirror \*mirror);

void set\_right\_down(char \*xy, info\_mirror \*mirror);

void set\_axis(char \*axis, info\_mirror \*mirror);

#endif

Название файла: draw\_mirror.c

#include "../include/draw\_mirror.h"

/\* функция отражения области \*/

void draw\_mirror(int x0, int y0, int x1, int y1, char axis, Png \*png) {

int width = abs(x1 - x0);

int height = abs(y1 - y0);

if (axis == 'x') {

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

for (int x = x0; x < x0 + width / 2; ++x) {

int mirroredX = x1 - (x - x0);

RGB tempColor = get\_color(x, y, png);

RGB newColor = get\_color(mirroredX, y, png);

set\_pixel(x, y, newColor, png);

set\_pixel(mirroredX, y, tempColor, png);

}

}

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

for (int y = y0; y < y0 + height / 2; ++y) {

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

int mirroredY = y1 - (y - y0);

RGB tempColor = get\_color(x, y, png);

RGB newColor = get\_color(x, mirroredY, png);

set\_pixel(x, y, newColor, png);

set\_pixel(x, mirroredY, tempColor, png);

}

}

}

}

int check\_mirror(info\_mirror \*mirror) {

if (mirror->p0.x == -1 || mirror->p0.y == -1) {

puts("You don't specify start cords of mirror's edge!");

return 0;

}

if (mirror->p1.x == -1 || mirror->p1.y == -1) {

puts("You don't specify end cords of mirror's edge!");

return 0;

}

if (mirror->axis == 'n') {

puts("You don't specify axis of mirror!");

return 0;

}

return 1;

}

void set\_left\_up(char \*xy, info\_mirror \*mirror) {

char \*x = strtok(xy, ".");

char \*y = strtok(NULL, ".");

if (x == NULL) {

puts("You don't specify X for left\_up point of mirror's edge!");

exit(40);

}

if (y == NULL) {

puts("You don't specify Y for left\_up point of mirror's edge!");

exit(40);

}

if (is\_digit(x))

mirror->p0.x = atoi(x);

else {

puts("Your X coord for left\_up point isn't integer number!");

exit(41);

}

if (is\_digit(y)) {

mirror->p0.y = atoi(y);

} else {

puts("Your Y coord for left\_up point isn't integer number!");

exit(41);

}

}

void set\_right\_down(char \*xy, info\_mirror \*mirror) {

char \*x = strtok(xy, ".");

char \*y = strtok(NULL, ".");

if (x == NULL) {

puts("You don't specify X for right\_down point of mirror's edge!");

exit(40);

}

if (y == NULL) {

puts("You don't specify Y for right\_down point of mirror's edge!");

exit(40);

}

if (is\_digit(x))

mirror->p1.x = atoi(x);

else {

puts("Your X coord for right\_down point isn't integer number!");

exit(41);

}

if (is\_digit(y)) {

mirror->p1.y = atoi(y);

} else {

puts("Your Y coord for right\_down point isn't integer number!");

exit(41);

}

}

void set\_axis(char \*axis, info\_mirror \*mirror) {

if (strcmp(axis, "x") == 0) {

mirror->axis = 'x';

} else if (strcmp(axis, "y") == 0) {

mirror->axis = 'y';

} else {

puts("Your axis isn't 'x' or 'y'!");

exit(41);

}

}

Название файла: draw\_line.h

#ifndef DRAW\_LINE\_H

#define DRAW\_LINE\_H

#include "../include/png\_objects.h"

#include "../include/add\_operations.h"

void draw\_line(int x0, int y0, int x1, int y1, int thickness, RGB color, Png \*png);

int check\_line(info\_line \*line);

void set\_start\_cords(char \*xy, info\_line \*line);

void set\_end\_cords(char \*xy, info\_line \*line);

void set\_color\_line(char \*rgb, info\_line \*line);

void set\_thickness\_line(char \*thickness, info\_line \*line);

void set\_square(int x0, int y0, int size, RGB color, Png \*png);

#endif

Название файла: draw\_line.c

#include "../include/draw\_line.h"

#include "../include/draw\_pentagram.h"

/\* функция для рисования линия \*/

void draw\_line(int x0, int y0, int x1, int y1, int thickness, RGB color, Png \*png) {

int dx = abs(x1 - x0);

int dy = abs(y1 - y0);

int sx = x0 < x1 ? 1 : -1;

int sy = y0 < y1 ? 1 : -1;

int err = dx - dy;

while (1) {

int rectX = x0 - thickness / 2;

int rectY = y0 - thickness / 2;

int rectWidth = thickness;

int rectHeight = thickness;

if ((rectX >= 0 && rectX + rectWidth < png->width &&

rectY >= 0 && rectY + rectHeight < png->height) ||

(x0 >= 0 && x0 < png->width && y0 >= 0 && y0 < png->height)) {

for (int y = rectY; y < rectY + rectHeight; ++y) {

for (int x = rectX; x < rectX + rectWidth; ++x) {

set\_pixel(x, y, color, png);

}

}

}

if (x0 == x1 && y0 == y1) {

break;

}

int e2 = 2 \* err;

if (e2 > -dy) {

err -= dy;

x0 += sx;

}

if (e2 < dx) {

err += dx;

y0 += sy;

}

}

}

/\* проверка введённых параметров для линии на существование (все ли существуют) \*/

int check\_line(info\_line \*line) {

if (line->p0.x == -1 || line->p0.y == -1) {

puts("You don't specify start cords of line!"); /\* нет начальных координат \*/

return 0;

}

if (line->p1.x == -1 || line->p1.y == -1) {

puts("You don't specify end cords of line!"); /\* нет конечных координат \*/

return 0;

}

if (line->color.r == -1 || line->color.g == -1 || line->color.b == -1) {

puts("You don't specify color of line!"); /\* нет конечных координат \*/

return 0;

}

if (line->thickness == -1) {

puts("You don't specify thickness of line!"); /\* нет толщины \*/

return 0;

}

return 1;

}

/\* функция для задания начальных координат \*/

void set\_start\_cords(char \*xy, info\_line \*line) {

char \*x = strtok(xy, ".");

char \*y = strtok(NULL, ".");

if (x == NULL) {

puts("You don't specify X for start point of line!");

exit(40);

}

if (y == NULL) {

puts("You don't specify Y for start point of line!");

exit(40);

}

if (is\_digit(x))

line->p0.x = atoi(x);

if (is\_digit(y))

line->p0.y = atoi(y);

else {

puts("Your Y coord for start point isn't integer number!");

exit(41);

}

}

/\* функция для задания конечных координат \*/

void set\_end\_cords(char \*xy, info\_line \*line) {

char \*x = strtok(xy, ".");

char \*y = strtok(NULL, ".");

if (x == NULL) {

puts("You don't specify X for end point of line!");

exit(40);

}

if (y == NULL) {

puts("You don't specify Y for end point of line!");

exit(40);

}

if (is\_digit(x))

line->p1.x = atoi(x);

else {

puts("Your X coord for end point isn't integer number!");

exit(41);

}

if (is\_digit(y))

line->p1.y = atoi(y);

else {

puts("Your Y coord for end point isn't integer number!");

exit(41);

}

}

/\* функция для задания цвета линии \*/

void set\_color\_line(char \*rgb, info\_line \*line) {

char \*r = strtok(rgb, ".");

char \*g = strtok(NULL, ".");

char \*b = strtok(NULL, ".");

if (r == NULL) {

puts("You don't specify R (red) component of line's color!");

exit(40);

}

if (g == NULL) {

puts("You don't specify G (green) component of line's color!");

exit(40);

}

if (b == NULL) {

puts("You don't specify B (blue) component of line's color!");

exit(40);

}

if (is\_digit(r)) {

if ((atoi(r) < 0) || (atoi(r) > 255)) {

puts("Your R (red) component of line's color must be in the range from 0 to 255!");

exit(42);

}

line->color.r = atoi(r);

} else {

puts("Your R (red) component of line's color isn't integer number!");

exit(41);

}

if (is\_digit(g)) {

if ((atoi(g) < 0) || (atoi(g) > 255)) {

puts("Your G (green) component of line's color must be in the range from 0 to 255!");

exit(42);

}

line->color.g = atoi(g);

} else {

puts("Your G (green) component of line's color isn't integer number!");

exit(41);

}

if (is\_digit(b)) {

if ((atoi(b) < 0) || (atoi(b) > 255)) {

puts("Your B (blue) component of line's color must be in the range from 0 to 255!");

exit(42);

}

line->color.b = atoi(b);

} else {

puts("Your B (blue) component of line's color isn't integer number!");

exit(41);

}

}

/\* функция для задания толщины линии \*/

void set\_thickness\_line(char \*thickness, info\_line \*line) {

if (thickness == NULL) {

printf("You don't specify thickness of line!\n");

exit(40);

}

if (is\_digit(thickness)) {

if (atoi(thickness) < 1) {

printf("Your line's thickness must be greater than 0!\n");

exit(42);

}

line->thickness = atoi(thickness);

} else {

printf("Your line's thickness isn't integer number!\n");

exit(41);

}

}

void set\_square(int x0, int y0, int size, RGB color, Png \*png) {

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

for (int j = -size; j <= size; j++) {

if (!(x0 + i < 0 || x0 + i >= png->width || y0 + j < 0 || y0 + j >= png->height)) {

set\_pixel(x0 + i, y0 + j, color, png);

}

}

}

}

Название файла: draw\_pentagram.h

#ifndef DRAW\_PENTAGRAM\_H

#define DRAW\_PENTAGRAM\_H

#include "../include/draw\_pentagram.h"

#include "../include/draw\_line.h"

void draw\_pentagram(int x0, int y0, int radius, int thickness, RGB color, Png \*png);

int check\_pentagram(info\_pentagram \*pentagram);

void set\_radius(char \*radius, info\_pentagram \*pentagram);

void set\_center(char \*center, info\_pentagram \*pentagram);

void set\_color\_pentagram(char \*color, info\_pentagram \*pentagram);

void set\_thickness\_pentagram(char \*thickness, info\_pentagram \*pentagram);

void draw\_circle(int x0, int y0, int radius, int thickness, RGB color, Png \*png);

void draw\_fill\_circle(int x0, int y0, int radius, RGB color, Png \*png);

#endif

Название файла: draw\_pentagram.c

#include "../include/draw\_pentagram.h"

#include <time.h>

void draw\_pentagram(int x0, int y0, int radius, int thickness, RGB color, Png \*png) {

// time\_t start\_time = clock();

draw\_circle(x0, y0, radius, thickness, color, png);

// time\_t end\_time = clock();

// printf("%lf\n", -difftime(start\_time, end\_time) / CLOCKS\_PER\_SEC);

Point p2 = {x0, y0 - radius};

Point p5 = {x0 + radius \* cos(-18 \* M\_PI / 180), y0 + radius \* sin(-18 \* M\_PI / 180)};

Point p3 = {x0 + radius \* cos(54 \* M\_PI / 180), y0 + radius \* sin(54 \* M\_PI / 180)};

Point p1 = {x0 + radius \* cos(126 \* M\_PI / 180), y0 + radius \* sin(126 \* M\_PI / 180)};

Point p4 = {x0 + radius \* cos(198 \* M\_PI / 180), y0 + radius \* sin(198 \* M\_PI / 180)};

// start\_time = clock();

draw\_line(p1.x, p1.y, p2.x, p2.y, thickness, color, png);

// end\_time = clock();

// printf("%lf\n", -difftime(start\_time, end\_time) / CLOCKS\_PER\_SEC);

// start\_time = clock();

draw\_line(p2.x, p2.y, p3.x, p3.y, thickness, color, png);

// end\_time = clock();

// printf("%lf\n", -difftime(start\_time, end\_time) / CLOCKS\_PER\_SEC);

// start\_time = clock();

draw\_line(p3.x, p3.y, p4.x, p4.y, thickness, color, png);

// end\_time = clock();

// printf("%lf\n", -difftime(start\_time, end\_time) / CLOCKS\_PER\_SEC);

// start\_time = clock();

draw\_line(p4.x, p4.y, p5.x, p5.y, thickness, color, png);

// end\_time = clock();

// printf("%lf\n", -difftime(start\_time, end\_time) / CLOCKS\_PER\_SEC);

// start\_time = clock();

draw\_line(p5.x, p5.y, p1.x, p1.y, thickness, color, png);

// end\_time = clock();

// printf("%lf\n", -difftime(start\_time, end\_time) / CLOCKS\_PER\_SEC);

}

int check\_pentagram(info\_pentagram \*pentagram) {

if (!pentagram->radius) {

puts("You don't specify radius of pentagram!");

return 0;

}

if (!pentagram->center.x || !pentagram->center.y) {

puts("You don't specify center of pentagram!");

return 0;

}

if (!pentagram->thickness) {

puts("You don't specify thickness of pentagram!");

return 0;

}

return 1;

}

void set\_radius(char \*radius, info\_pentagram \*pentagram) {

if (radius == NULL) {

printf("You don't specify radius of pentagram!\n");

exit(41);

}

if (is\_digit(radius)) {

if (atoi(radius) < 1) {

printf("Your pentagram's radius must be greater than 0!\n");

exit(42);

}

pentagram->radius = atoi(radius);

} else {

printf("Your pentagram's radius isn't an integer number!\n");

exit(0);

}

}

void set\_center(char \*center, info\_pentagram \*pentagram) {

char \*x = strtok(center, ".");

char \*y = strtok(NULL, ".");

if (x == NULL) {

puts("You don't specify X for center of pentagram!");

exit(40);

}

if (y == NULL) {

puts("You don't specify Y for center of pentagram!");

exit(40);

}

if (is\_digit(x))

pentagram->center.x = atoi(x);

else {

puts("Your X coord for pentagram's center isn't integer number!");

exit(41);

}

if (is\_digit(y))

pentagram->center.y = atoi(y);

else {

puts("Your Y coord for pentagram's center isn't integer number!");

exit(41);

}

}

void set\_thickness\_pentagram(char \*thickness, info\_pentagram \*pentagram) {

if (thickness == NULL) {

printf("You don't specify thickness of pentagram!\n");

exit(40);

}

if (is\_digit(thickness)) {

if (atoi(thickness) < 1) {

printf("Your line's thickness must be greater than 0!\n");

exit(42);

}

pentagram->thickness = atoi(thickness);

} else {

printf("Your pentagram's thickness isn't integer number!\n");

exit(40);

}

}

void set\_color\_pentagram(char \*rgb, info\_pentagram \*pentagram) {

char \*r = strtok(rgb, ".");

char \*g = strtok(NULL, ".");

char \*b = strtok(NULL, ".");

if (r == NULL) {

puts("You don't specify R (red) component of pentagram's color!");

exit(40);

}

if (g == NULL) {

puts("You don't specify G (green) component of pentagram's color!");

exit(40);

}

if (b == NULL) {

puts("You don't specify B (blue) component of pentagram's color!");

exit(40);

}

if (is\_digit(r)) {

if ((atoi(r) < 0) || (atoi(r) > 255)) {

puts("Your R (red) component of pentagram's color must be in the range from 0 to 255!");

exit(41);

}

pentagram->color.r = atoi(r);

} else {

puts("Your R (red) component of pentagram's color isn't integer number!");

exit(41);

}

if (is\_digit(g)) {

if ((atoi(g) < 0) || (atoi(g) > 255)) {

puts("Your G (green) component of pentagram's color must be in the range from 0 to 255!");

exit(41);

}

pentagram->color.g = atoi(g);

} else {

puts("Your G (green) component of pentagram's color isn't integer number!");

exit(41);

}

if (is\_digit(b)) {

if ((atoi(b) < 0) || (atoi(b) > 255)) {

puts("Your B (blue) component of pentagram's color must be in the range from 0 to 255!");

exit(41);

}

pentagram->color.b = atoi(b);

} else {

puts("Your B (blue) component of pentagram's color isn't integer number!");

exit(41);

}

}

//void draw\_circle(int x0, int y0, int radius, int thickness, RGB color, Png \*png) {

// int D = 3 - 2 \* radius;

// int x = 0;

// int y = radius;

// while (x <= y) {

// set\_square(x + x0, y + y0, thickness / 2, color, png);

// set\_square(y + x0, x + y0, thickness / 2, color, png);

// set\_square(-y + x0, x + y0, thickness / 2, color, png);

// set\_square(-x + x0, y + y0, thickness / 2, color, png);

// set\_square(-x + x0, -y + y0, thickness / 2, color, png);

// set\_square(-y + x0, -x + y0, thickness / 2, color, png);

// set\_square(y + x0, -x + y0, thickness / 2, color, png);

// set\_square(x + x0, -y + y0, thickness / 2, color, png);

// if (D < 0) {

// D += 4 \* x + 6;

// x++;

// } else {

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

// x++;

// y--;

// }

// }

//}

void draw\_circle(int x0, int y0, int radius, int thickness, RGB color, Png \*png) {

int yCenter = y0;

int halfThickness = thickness / 2;

int outerRadius = radius + halfThickness;

int innerRadius = radius - halfThickness;

int outerRadiusSquared = outerRadius \* outerRadius;

int innerRadiusSquared = innerRadius \* innerRadius;

for (int y = yCenter - outerRadius; y <= yCenter + outerRadius; y++) {

for (int x = x0 - outerRadius; x <= x0 + outerRadius; x++) {

int dx = x - x0;

int dy = y - y0;

int distance = dx \* dx + dy \* dy;

if (distance <= outerRadiusSquared && distance >= innerRadiusSquared) {

set\_pixel(x, y, color, png);

}

}

}

}

void draw\_fill\_circle(int x0, int y0, int radius, RGB color, Png \*png) {

int yCenter = y0;

int halfThickness = 0;

int outerRadius = radius + halfThickness;

int innerRadius = radius - halfThickness;

int outerRadiusSquared = outerRadius \* outerRadius;

int innerRadiusSquared = innerRadius \* innerRadius;

for (int y = yCenter - outerRadius; y <= yCenter + outerRadius; y++) {

for (int x = x0 - outerRadius; x <= x0 + outerRadius; x++) {

int dx = x - x0;

int dy = y - y0;

int distance = dx \* dx + dy \* dy;

if (distance <= outerRadiusSquared && distance >= innerRadiusSquared) {

set\_pixel(x, y, color, png);

}

else if (distance < innerRadiusSquared) {

set\_pixel(x, y, color, png);

}

}

}

}

Название файла: Makefile

CC = gcc

CFLAGS = -g -Werror -std=gnu99

LIBS = -lpng -lm

SRCDIR = sources

OBJDIR = objects

INCDIR = include

DOCSDIR = docs/html

DOCSINDEX = ./docs/html/index.html

SOURCES = $(wildcard $(SRCDIR)/\*.c)

OBJECTS = $(patsubst $(SRCDIR)/%.c, $(OBJDIR)/%.o, $(SOURCES))

EXECUTABLE = cw

DOXYGEN\_CONFIG = Doxyfile

all: $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)

$(CC) $(CFLAGS) $^ -o $@ $(LIBS)

$(OBJDIR)/%.o: $(SRCDIR)/%.c

@mkdir -p $(OBJDIR)

$(CC) $(CFLAGS) -c $< -o $@

clean\_objects:

rm -rf $(OBJDIR)

clean:

rm -rf $(OBJDIR) $(EXECUTABLE)

clean\_docs:

rm -rf $(DOCSDIR)

docs:

doxygen $(DOXYGEN\_CONFIG)

docs\_view:

doxygen $(DOXYGEN\_CONFIG)

xdg-open $(DOCSINDEX)

.PHONY: all, clean, clean\_docs, clean\_objects, docs, docs\_view

**Приложение Б**

**ТЕСТИРОВАНИЕ**



Рисунок 1 – изображение для тестирования

1. Тестирование функции line:

Аргументы для запуска: ./cw -i imgs/input.png --line --start 25.75 --end 175.150 --thickness 6 --color 100.100.100



Рисунок 2 – результат работы функции line

1. Тестирование функции mirror:

Аргументы для запуска: ./cw -i imgs/input.png --mirror --axis x --left\_up 75.75 --right\_down 175.175



Рисунок 3 – результат работы функции *mirror*

1. Тестирование функции pentagram:

Аргументы для запуска: ./cw -i imgs/input.png --pentagram --center 100.75 --radius 60 --thickness 6 --color 100.100.100



Рисунок 4 – результат работы функции *pentagram*