МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Программа на языке С 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
2. 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
3. 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 ли-
нии */

            {"mirror",     no_argument,       NULL, 400}, /* отражение */
            {"axis",       required_argument, NULL, 401}, /* ось отраже-
ния */
            {"left_up",    required_argument, NULL, 402}, /* верхний ле-
вый угол области */
            {"right_down", required_argument, NULL, 403}, /* нижний пра-
вый 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 от-
ражённой области */
    if (mirror.p) {
        if (check_mirror(&mirror))
            draw_mirror(mirror.p0.x, mirror.p0.y, mirror.p1.x, mir-
ror.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 пента-
граммы в круге */
    if (pentagram.p) {
        if (check_pentagram(&pentagram))
```

```
            draw_pentagram(pentagram.center.x, pentagram.center.y, penta-
gram.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 режи-
ме */
    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, im-
age->info_ptr); /* получение высоты изображения */
    image->color_type = png_get_color_type(image->png_ptr, im-
age->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, im-
age->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) * im-
age->height); /* выделение памяти под указатели на строки */
    for (y = 0; y < image->height; y++)
    {
        image->row_pointers[y] = (png_byte *)malloc(png_get_rowbytes(im-
age->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, im-
age->height, image->bit_depth, image->color_type,
                 PNG_INTERLACE_NONE, PNG_COMPRESSION_TYPE_BASE, PNG_FIL-
TER_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 суще-
ствуют) */
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 num-
ber!");
        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 num-
ber!");
        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 num-
ber!");
        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 num-
ber!");
        exit(41);
    }
    if (is_digit(y))
        pentagram->center.y = atoi(y);
    else {
        puts("Your Y coord for pentagram's center isn't integer num-
ber!");
        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 >= innerRa-
diusSquared) {
                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 >= innerRa-
diusSquared) {
                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

2. 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*

3. 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*