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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка изображений на языке Си

Студентка гр. 0382 _____ Здобнова К.Д.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студентка Здобнова К.Д.

Группа 0382

Тема работы: Обработка строк на языке Си.

Исходные данные:

Программе на вход подается изображение (в формате PNG). Необходимо преобразовать изображение в соответствии с заданием. Для работы с аргументами командной строки требуется реализовать утилиту CLI (Command Line Interface).

Содержание пояснительной записки:

Обязательны разделы «Содержание», «Введение», «Ход работы», «Заключение», «Тестирование»

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 5.04.2021

Дата сдачи курсовой: 24.05.2021

Дата защиты курсовой: 24.05.2021

Студентка _____ Здобнова К.Д.

Преподаватель _____ Берленко Т.А.

АННОТАЦИЯ

Курсовая работа представляет собой программу на языке Си для обработки изображения в формате PNG. Программа должна поддерживать работу с утилитой CLI (библиотека getopt.h). Реализуемые функции для работы с файлом: нарисовать отрезок, сделать изображение черно-белым, инвертировать цвета в заданной окружности, обрезать изображение.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью работы является написание корректной программы для работы с изображением и функций для его обработки. Для поддержания формата png используется библиотека libpng. Написанная программа поддерживается на операционной системе Linux. Итогом курсовой работы является программа, реализующая считывание, обработку и запись png-изображения.

1. ЗАДАНИЕ

Вариант 13

Программа должна иметь CLI или GUI. Более подробно тут: http://se.moevm.info/doku.php/courses:programming:rules_extra_kurs

Общие сведения:

Формат картинки PNG (рекомендуем использовать библиотеку libpng)

файл всегда соответствует формату PNG

обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями.

все поля стандартных PNG заголовков в выходном файле должны иметь те же значения что и во входном (разумеется кроме тех, которые должны быть изменены).

Программа должна реализовывать следующий функционал по обработке PNG-файла:

- Преобразовать в Ч/Б изображение (любым простым способом).

Функционал определяется

Координатами левого верхнего угла области

Координатами правого нижнего угла области

Алгоритмом, если реализовано несколько алгоритмов преобразования изображения (по желанию студента)

- Рисование отрезка. Отрезок определяется:

координатами начала

координатами конца

цветом

толщиной

- Инвертировать цвета в заданной окружности. Окружность определяется либо координатами левого верхнего и правого нижнего угла квадрата, в который она вписана, либо координатами ее центра и радиусом

•Обрезка изображения. Требуется обрезать изображение по заданной области. Область определяется:

Координатами левого верхнего угла

Координатами правого нижнего угла

2. ХОД РАБОТЫ

2.1. Функции main():

В функции *main()* происходит обработка параметров, переданных в *argv*. С помощью функций библиотеки *getopt.h* определяется опция для редактирования изображения или вызов вспомогательной информации. Благодаря оператору switch программа выполняет в дальнейшем заданные пользователем функции.

2.2. Структуры

Структура *Png*– структура для хранения изображения и данных о нем, имеющая следующие поля:

width (тип *int)*– число, равное ширине (количеству пикселей) изображения;

height (тип *int*) – число, равное высоте (количеству пикселей) изображения;

color_type (тип *png_byte*) – число, обозначающее тип цвета изображения;

bit_depth (тип *png_byte*) – число, обозначающее глубину цвета изображения;

png_ptr (тип *png_structp*) – указатель на структуру;

info_ptr (тип *png_infop)* – сведения об изображении;

row_pointers (тип **png_bytep*) – массив указателей на строки.

2.3. Функции

Функция *print_PNG_info ()*

Функция, предназначенная для считывания печати на экран информации об изображении: ширину, высоту, тип цвета и глубину.

Функция *read_png_file ()*

Функция предназначена для чтения изображения. В функции прописаны всевозможные ошибки чтения и инициализации входного файла.

Функция *write_png_file ()*

Функция предназначена для записи изображения. В функции прописаны всевозможные ошибки при записи выходного файла.

Функция *paint_pixel ()*

Принимает аргументом координаты и цвет для перекрашивания конкретного пикселя.

Функция *cut_off()*

Функция предназначена для обрезки изображения. С помощью цикла *for* нужная область перезаписывается в левый верхний угол картинки, затем обрезается высота и ширина по заданной области.

Функция *black_and_white ()*

Функция меняет цвет пикселей заданной области на черный или белый цвета. Если сумма rgb пикселя превышает 480 (из формулы 255 / brightness / 2 * 3), то пиксель закрашивается белым, в противном случае – черным.

Функция *inversion ()*

Функция инвертирует цвета в заданной области. На вход подаются координаты центра и радиус окружности. С помощью уравнения окружности определяется подходящий пиксель. Инвертированный цвет – это разность 255 и исходного цвета.

Функция *draw_line ()*

Функция рисует линию по заданным координатам начала и конца. Реализован алгоритм брезенхема, так как в нем используются только целочисленные значения и выполняется корректное закрашивание. Высчитываются значения absDeltaX и absDeltaX, в зависимости от большего значения выполняется цикл относительно y или x. Также высчитывается направление (int *direction)* относительно оси y или x. Если значение переменной 1 происходит передвижение в верх по оси, при -1 – вниз, 0 – вертикальное или горизонтальное положение.

2.4 Тестирование

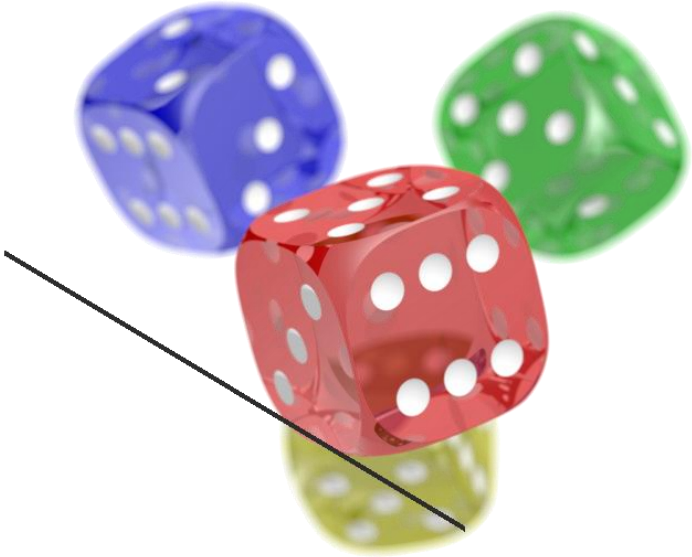
1.

```
kseniya@kseniya-VirtualBox:~/Desktop/cw2$ ./a.out test.png -i
Ширина изображения: 800
Высота изображения: 600
Тип цвета: 6
Глубина цвета: 8
```

```
kseniya@kseniya-VirtualBox:~/Desktop/cw2$ ./a.out test.png -h
Поддерживаются файлы с глубиной цвета RGB_ALPHA
Формат ввода: ./a.out <filename.png> -<option> <arguments> <filename.png>
Сделать изображение черно-белым
    -b <x верхнего левого угла> <y  верхнего левого угла> <x нижнего правого угла> <y н
ижнего правого угла>
draw_line [] -...
    -l <x начала> <y начала> <x конца> <y конца> <R> <G> <B> <A> <ширина линии в пиксел
ях (1, 3 или 5)>
Инвертировать цвета в заданной окружности
    -n <x центра> <y центра> <радиус окружности>
Обрезать изображение
    -c <x верхнего левого угла> <y  верхнего левого угла> <x нижнего правого угла> <y н
ижнего правого угла>
Получить информацию об изображении
    -i
Вызвать справку
```

```
kseniya@kseniya-VirtualBox:~/Desktop/cw2$ ./a.out test.png -b 155 400 466 100 re
s.png
kseniya@kseniya-VirtualBox:~/Desktop/cw2$
```

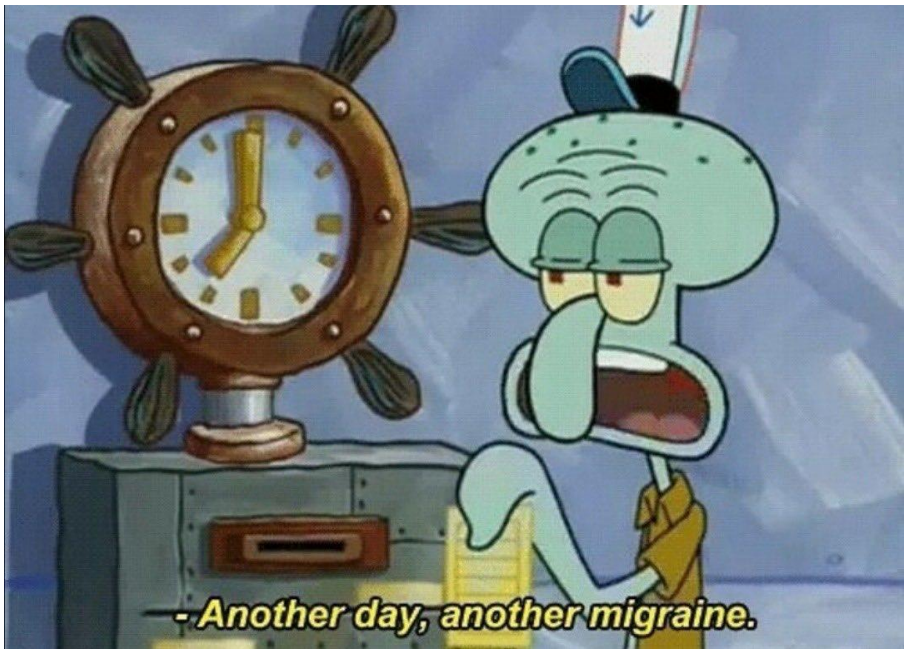
```
kseniya@kseniya-VirtualBox:~/Desktop/cw2$ ./a.out test.png -l 50 350 466 100 45
45 45 255 5 res.png
kseniya@kseniya-VirtualBox:~/Desktop/cw2$
```



```
kseniya@kseniya-VirtualBox:~/Desktop/cw2$ ./a.out test.png -n 350 200 150 res.pn
g
kseniya@kseniya-VirtualBox:~/Desktop/cw2$
```

```
kseniya@kseniya-VirtualBox:~/Desktop/cw2$ ./a.out test.png -c 360 450 700 70 res
.png
kseniya@kseniya-VirtualBox:~/Desktop/cw2$
```

2.



```
kseniya@kseniya-VirtualBox:~/Desktop/cw2$ ./a.out skw.png -i
Данный тип файла не поддерживается.
kseniya@kseniya-VirtualBox:~/Desktop/cw2$
```

ЗАКЛЮЧЕНИЕ

Была изучена библиотека libpng для работы с файлами png типа. Научились считывать, обрабатывать и записывать изображения. Программа поддерживает работу с утилитой CLI (библиотека getopt.h).

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ

1. файл main.c

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <getopt.h>
#include <png.h>

struct Png{
    int width, height;
    png_byte color_type;
    png_byte bit_depth;
    png_structp png_ptr;
    png_infop info_ptr;
    png_bytep *row_pointers;
};


void print_PNG_info(struct Png *image){
    printf("Ширина изображения: %d\n", image->width);
    printf("Высота изображения: %d\n", image->height);
    printf("Тип цвета: %u\n", image->color_type);
    printf("Глубина цвета: %u\n", image->bit_depth);
}

void print_info(){
    printf("Поддерживаются файлы с глубиной цвета RGB_ALPHA\n");
    printf("Формат ввода: ./a.out <filename.png> -<option> <arguments>
<filename.png>\n");
    printf("Сделать изображение черно-белым\n");
    printf("    -b <x верхнего левого угла> <y  верхнего левого угла> "
           "<x нижнего правого угла> <y нижнего правого угла>\n");
    printf("draw_line [] -...\n");
    printf("    -l <x начала> <y начала> <x конца> <y конца> <R> <G> <B>
<A> "
           "<ширина линии в пикселях (1, 3 или 5)>\n");
    printf("Инвертировать цвета в заданной окружности\n");
    printf("    -n <x центра> <y центра> <радиус окружности>\n");
    printf("Обрезать изображение\n");
    printf("    -c <x верхнего левого угла> <y  верхнего левого угла> "
           "<x нижнего правого угла> <y нижнего правого угла>\n");
    printf("Получить информацию об изображении\n");
    printf("    -i\n");
    printf("Вызвать справку\n");
    printf("    -h\n");
}

void read_png_file(char *file_name, struct Png *image) {
    int x,y;
    char header[8];

    FILE *fp = fopen(file_name, "rb");
    if (!fp){
```

```
        printf("Фйал не читается.\n");
        exit(-1);
    }

    fread(header, 1, 8, fp);
    if (png_sig_cmp(header, 0, 8)){
        printf("Неверный формат файла: допускается только PNG.\n");
        exit(-1);
    }
    image->png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL,
NULL, NULL);

    if (!image->png_ptr){
        printf("Ошибка инициализации.\n");
        exit(-1);
    }

    image->info_ptr = png_create_info_struct(image->png_ptr);
    if (!image->info_ptr){
        printf("png_create_info_struct failed.\n");
        exit(-1);
    }

    if (setjmp(png_jmpbuf(image->png_ptr))){
        printf("Some error handling: error during init_io.\n");
        exit(-1);
    }

    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);

    png_read_update_info(image->png_ptr, image->info_ptr);

    if (setjmp(png_jmpbuf(image->png_ptr))){
        printf("error during read_image.\n");
        exit(-1);
    }

    if (png_get_color_type(image->png_ptr, image->info_ptr) !=
PNG_COLOR_TYPE_RGB_ALPHA){
        printf("Данный тип файла не поддерживается.\n");
        exit(-1);
    }

    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, struct Png *image) {
    int x,y;
    FILE *fp = fopen(file_name, "wb");
    if (!fp){
        printf("Ошибка открытия файла на запись.\n");
        exit(-1);
    }

    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(-1);
    }

    image->info_ptr = png_create_info_struct(image->png_ptr);
    if (!image->info_ptr){
        printf("png_create_info_struct failed.\n");
        exit(-1);
    }

    if (setjmp(png_jmpbuf(image->png_ptr))){
        printf("error during init_io.\n");
        exit(-1);
    }

    png_init_io(image->png_ptr, fp);


    if (setjmp(png_jmpbuf(image->png_ptr))){
        printf("error during writing header.\n");
        exit(-1);
    }

    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);


    if (setjmp(png_jmpbuf(image->png_ptr))){
        printf("error during writing bytes.\n");
        exit(-1);
    }
```

```
    png_write_image(image->png_ptr, image->row_pointers);


    if (setjmp(png_jmpbuf(image->png_ptr))){
        printf("error during end of write.\n");
        exit(-1);
    }

    png_write_end(image->png_ptr, NULL);
    for (y = 0; y < image->height; y++)
        free(image->row_pointers[y]);
    free(image->row_pointers);
    fclose(fp);
}

void paint_pixel(struct Png *image, int x, int y, int R, int G, int B,
int A){
    if ((x < 0) || (x >= image->width) || (y < 0) || (y >= image-
>height))
        return;
    png_byte *row = image->row_pointers[y];
    png_byte *ptr = &(row[x * 4]);
    ptr[0] = R;
    ptr[1] = G;
    ptr[2] = B;
    ptr[3] = A;
}

void cut_off(struct Png *image, int x1, int y1, int x2, int y2) {
    int x, y;
    for (y = 0; y <= y2 - y1; y++) {
        png_byte *cut_row = image->row_pointers[y + y1];
        for (x = 0; x <= x2 - x1; x++) {
            png_byte *cut_ptr = &(cut_row[(x + x1) * 4]);
            paint_pixel(image, x, y, cut_ptr[0], cut_ptr[1], cut_ptr[2],
cut_ptr[3]);
        }
    }
    for (y = y2 - y1 + 1; y < image->height; y++)
        free(image->row_pointers[y]);
    image->width = x2 - x1;
    image->height = y2 - y1;
}

void black_and_white(struct Png *image, int x1, int y1, int x2, int y2) {
    int x,y;
    for (y = 0; y < image->height; y++) {
        png_byte *row = image->row_pointers[y];
        for (x = 0; x < image->width; x++) {
            png_byte *ptr = &(row[x * 4]);
            if (x1 <= x && x <= x2 && y1 <= y && y <= y2) { //прописать
ошибку выхода за границы картинки
                if (ptr[0] + ptr[1] + ptr[2] > 480) {
                    paint_pixel(image, x, y, 255, 255, 255, ptr[3]);
                } else {
                    paint_pixel(image, x, y, 0, 0, 0, ptr[3]);
                }
```

```
            }
        }
    }
}


void inversion(struct Png *image, int x1, int y1, int r){
    int x, y;
    for (y = 0; y < image->height; y++) {
        png_byte *row = image->row_pointers[y];
        for (x = 0; x < image->width; x++) {
            png_byte *ptr = &(row[x * 4]);
            if (pow((x1 - x), 2) + pow((y1 - y), 2) <= pow(r, 2)) {
//прописать ошибку выхода за границы картинки
                paint_pixel(image, x, y, 255 - ptr[0], 255 - ptr[1], 255
- ptr[2], ptr[3]);
            }
        }
    }
}

void draw_line(struct Png *image, int x1, int y1, int x2, int y2, int R,
int G, int B, int A, int width){
    int DeltaX = x2 - x1;
    int DeltaY = y2 - y1;
    int absDeltaX = abs(x2 - x1);
    int absDeltaY = abs(y2 - y1);
    int accretion = 0;
    if (absDeltaX >= absDeltaY){
        int y = y1;
        int direction = DeltaY != 0 ? (DeltaY > 0 ? 1 : -1) : 0;
        for (int x = x1; DeltaX > 0 ? x <= x2 : x >= x2; DeltaX > 0 ? x++
: x--){
            for (int i = 0; width != 1 ? (width == 3 ? i <= 1 : i <= 2) :
i < 1; i++){
                paint_pixel(image, x, y + i, R, G, B, A);
                paint_pixel(image, x, y - i, R, G, B, A);
            }
            accretion += absDeltaY;
            if (accretion >=absDeltaX) {
                accretion -= absDeltaX;
                y += direction;
            }
        }
    } else {
        int x = x1;
        int direction = DeltaX != 0 ? (DeltaX > 0 ? 1 : -1) : 0;
        for (int y = y1; DeltaY > 0 ? y <= y2 : y >= y2; DeltaY > 0 ? y++
: y--){
            for (int i = 0; width != 1 ? (width == 3 ? i <= 1 : i <= 2) :
i < 1; i++){
                paint_pixel(image, x + i, y, R, G, B, A);
                paint_pixel(image, x - i, y, R, G, B, A);
            }
            accretion += absDeltaX;
            if (accretion >= absDeltaY){
                accretion -= absDeltaY;
```

```
                x += direction;
            }
        }
    }
}

int main(int argc, char **argv) {
    struct Png image;
    read_png_file(argv[1], &image);
    char *output = argv[2];
    struct option longOpts[] = {
            {"black_and_white", required_argument, NULL, 'b'},
            {"draw_line", required_argument, NULL, 'l'},
            {"inversion", required_argument, NULL, 'n'},
            {"cut_off", required_argument, NULL, 'c'},
            {"info", no_argument, NULL, 'i'},
            {"help", no_argument, NULL, 'h'},
            {"NULL", no_argument, NULL, 'o'},
    };
    int opt, longIndex;
    char *opts = "b:l:n:c:iho";
    opt = getopt_long(argc, argv, opts, longOpts, &longIndex);
    while(opt != -1){
        switch(opt){
            case 'b':
                if (!(atoi(argv[3])) || !(atoi(argv[4])) ||
!(atoi(argv[5])) || !(atoi(argv[6]))){
                    printf("Все значения должны быть целочисленными.\n");
                    return 0;
                }
                if (atoi(argv[3]) < 0 || atoi(argv[3]) >= image.width ||
atoi(argv[4]) < 0 || atoi(argv[5]) >= image.height) {
                    printf("Левая координата выходит за границы
изображения.\n");
                    return 0;
                }
                if (atoi(argv[5]) < 0 || atoi(argv[5]) >= image.width ||
atoi(argv[6]) < 0 || atoi(argv[6]) >= image.height) {
                    printf("Правая координата выходит за границы
изображения.\n");
                    return 0;
                }
                if (atoi(argv[3]) > atoi(argv[5]) || atoi(argv[4]) <
atoi(argv[6])) {
                    printf("Некорректые кординаты: сначала введите левую
верхнюю координату, затем нижнюю правую.\n");
                    return 0;
                }
                black_and_white(&image, atoi(argv[3]), image.height -
atoi(argv[4]), atoi(argv[5]), image.height - atoi(argv[6]));
                write_png_file(argv[7], &image);
                break;
            case 'l':
                if (!(atoi(argv[3])) || !(atoi(argv[4])) ||
!(atoi(argv[5])) || !(atoi(argv[6]))){
                    printf("Все значения должны быть целочисленными.\n");
                    return 0;
```

```
                }
                if (atoi(argv[3]) < 0 || atoi(argv[3]) >= image.width ||
atoi(argv[4]) < 0 || atoi(argv[4]) >= image.height) {
                    printf("Левая координата выходит за границы
изображения.\n");
                    return 0;
                }
                if (atoi(argv[5]) < 0 || atoi(argv[5]) >= image.width ||
atoi(argv[6]) < 0 || atoi(argv[6]) >= image.height) {
                    printf("Правая координата выходит за границы
изображения.\n");
                    return 0;
                }
                if (atoi(argv[7]) < 0 || atoi(argv[8]) < 0 ||
atoi(argv[9]) < 0
                || atoi(argv[7]) > 255 || atoi(argv[8]) > 255 ||
atoi(argv[9]) > 255
                || atoi(argv[10]) < 0 || atoi(argv[10]) > 255){
                    printf("Некорректный цвет, значения должны быть в
пределах от 0 до 255.\n");
                    return 0;
                }
                if (atoi(argv[11]) != 1 && atoi(argv[11]) != 3 &&
atoi(argv[11]) != 5){
                    printf("Некорректная толщина линии, поддерживается
ширина в 1, 3 или 5 пикселей.\n");
                    return 0;
                }
                draw_line(&image, atoi(argv[3]), image.height -
atoi(argv[4]), atoi(argv[5]), image.height - atoi(argv[6]),
                          atoi(argv[7]), atoi(argv[8]), atoi(argv[9]),
atoi(argv[10]), atoi(argv[11]));
                write_png_file(argv[12], &image);
                break;
            case 'n':
                if (!(atoi(argv[3])) || !(atoi(argv[4])) ||
!(atoi(argv[5]))){
                    printf("Все значения должны быть целочисленными.\n");
                    return 0;
                }
                if (atoi(argv[3]) < 0 || atoi(argv[3]) >= image.width ||
atoi(argv[4]) < 0 || atoi(argv[4]) >= image.height) {
                    printf("Координаты выходят за пределы
изображения.\n");
                    return 0;
                }
                if ((atoi(argv[5])) < 0 ){
                    printf("Радиус не может быть отрицательным.\n");
                    return 0;
                }
                inversion(&image, atoi(argv[3]), image.height -
atoi(argv[4]), atoi(argv[5]));
                write_png_file(argv[6], &image);
                break;
            case 'c':
                if (!(atoi(argv[3])) || !(atoi(argv[4])) ||
!(atoi(argv[5])) || !(atoi(argv[6]))){
```

```
                    printf("Все значения должны быть целочисленными.\n");
                    return 0;
                }
                if (atoi(argv[3]) < 0 || atoi(argv[3]) >= image.width ||
atoi(argv[4]) < 0 || atoi(argv[4]) >= image.height) {
                    printf("Левая координата выходит за границы
изображения.\n");
                    return 0;
                }
                if (atoi(argv[5]) < 0 || atoi(argv[5]) >= image.width ||
atoi(argv[6]) < 0 || atoi(argv[6]) >= image.height) {
                    printf("Правая координата выходит за границы
изображения.\n");
                    return 0;
                }
                if (atoi(argv[3]) > atoi(argv[5]) || atoi(argv[4]) <
atoi(argv[6])) {
                    printf("Некорректые кординаты: сначала введите левую
верхнюю координату, затем нижнюю правую.\n");
                    return 0;
                }
                cut_off(&image, atoi(argv[3]), image.height -
atoi(argv[4]), atoi(argv[5]), image.height - atoi(argv[6]));
                write_png_file(argv[7], &image);
                break;
            case 'i':
                print_PNG_info(&image);
                break;
            case 'h':
                print_info();
                break;
        }
        opt = getopt_long(argc, argv, opts, longOpts, &longIndex);
    }
    return 0;
}
```