

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

КУРСОВАЯ РАБОТА
по дисциплине «Программирование»
Тема: Обработка PNG изображений

Студент гр. 1304

Поршнеv Р.А.

Преподаватель

Чайка К.В.

Санкт-Петербург

2022

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Поршнев Р.А.

Группа 1304

Тема работы: Обработка PNG изображений

Исходные данные:

Набор ключей, которые управляют обработкой исходного изображения, а также имя выходного файла, где будет храниться обработанное изображение.

Содержание пояснительной записки:

Введение.

Основные теоретические положения.

Реализация программы.

Тестиро

Заключение.

Список используемых источников.

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 23.03.2022

Дата сдачи реферата: 23.05.2022

Дата защиты реферата: 25.05.2022

Студент

Поршнев Р.А.

Преподаватель

Чайка К.В.

АННОТАЦИЯ

В данной курсовой работе была реализована программа, которая имеет следующий функционал по обработке PNG изображения:

- (1) Рисование отрезка. Отрезок определяется:
 - координатами начала
 - координатами конца
 - цветом
 - толщиной
- (2) 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
- (3) Обрезка изображения. Требуется обрезать изображение по заданной области. Область определяется:
 - Координатами левого верхнего угла
 - Координатами правого нижнего угла

СОДЕРЖАНИЕ

	Введение	4
1.	Основные теоретические положения	5
1.1.	Используемые библиотеки	6
1.2.	Применение getopt	6
1.3.	Принцип хранения PNG - файла	6
2.	Реализация программы	8
2.1.	Функция входа в программу	8
2.2.	Функция считывания входного файла	8
2.3.	Функция записи в выходной файл	8
2.4.	Функция рисования линий	8
2.5.	Функция утолщения линии	9
2.6.	Функция инвертирования пикселей в окружности	9
2.7.	Функция обрезки изображения	9
3.	Тестирование	10
	Заключение	17
	Список использованных источников	18
	Приложение А. Исходный код программы и инструкция по запуску	19
	Приложение Б. Примеры работы программы и обработки ошибок	30

ВВЕДЕНИЕ

Целью данной работы является разработка программы, которая имеет функционал по считыванию, записи и обработке изображений формата PNG.

Программа должна получать исходные данные из командной строки с помощью ключей, то есть с помощью *getopt*.

Для реализации данной программы предстоит решить следующие задачи:

- Изучить *getopt*
- Изучить функции библиотеки *libpng*
- Считать входные данные в виде файла с изображением
- Обработать файл, исходя из условия задания
- Записать изменения в новый файл
- Протестировать работу программы

1. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

1.1. Используемые библиотеки

В данной курсовой работе применялись следующие библиотеки: `<stdio.h>`, `<unistd.h>`, `<getopt.h>`, `<stdlib.h>`, `<string.h>`, `<math.h>`, `<png.h>`. Если использование таких библиотек, как `<stdio.h>`, `<stdlib.h>`, `<string.h>` довольно привычно, то присутствие библиотек `<unistd.h>`, `<getopt.h>`, `<math.h>`, `<png.h>` уже требует пояснений. Из библиотеки `<unistd.h>` понадобятся функции для работы с файлами, их считыванием, закрытием и так далее. Библиотека `<getopt.h>` понадобится для работы с ключами. Библиотека `<math.h>` содержит большое количество математических функций, которые тоже нужны для правильного функционирования программы. Для непосредственной работы с PNG – файлами потребуется подключить библиотеку `<png.h>`.

1.2. Применение getopt

Существует большое количество программ на linux, работающих с помощью CLI(command line interface), поэтому было принято решение реализовать взаимодействие пользователя и программу с помощью данного интерфейса.

1.3. Принцип хранения PNG – файла

Для удобства работы с PNG – изображениями в работе будет использована библиотека `libpng`. После считывания файла для обработки изображения будет предоставлен доступ к двумерному массиву пикселей.

2. РЕАЛИЗАЦИЯ ПРОГРАММЫ

2.1. Функция входа в программу

В функции *main* объявляется структура *Png image*, в которую будут записываться данные об изображении. Структура *Png* имеет следующие поля: ширина, высота, реальная высота (потребуется при обрезании изображения), тип цвета, указатель на структуру, которая хранит некоторые данные об изображении, указатель на структуру, которая тоже хранит некоторые данные об изображении, а также указатель на указатель строку пикселей.

Далее происходит инициализация структуры *Configs*, которая хранит наименование функции, которую нужно выполнить и остальные поля для её работы, а так же другие поля для работы остальных функций.

Инициализируется массив *opts*, который хранит ключи и информацию присутствия обязательного аргумента для данного ключа. Далее инициализируется структура длинных ключей, информации об обязательном аргументе ключа, а также соответствующий короткий ключ.

Следующим этапом происходит считывание ключей, их аргументов, а так же заполнение структуры *Configs*. Функционал реализован с помощью оператора *switch* и *getopt*. В случае, если ключ имеет обязательный аргумент, который вводится через запятые, то сначала аргумент делится по запятой, а потом происходит перенос аргументов в *Configs*. В случае неопознанных аргументов или неверных параметров выводится справка.

После *getopt* следует отсев считанных ключей. Исходя из данных справки, можно сделать вывод, что несчитанными останутся 2 аргумента – это имя входного файла и выходного. В ином случае выводится справка.

Далее следует считывание PNG – файла. Если в ходе считывания произошла ошибка, то функция возвращает 1 и программа завершается.

Далее следуют проверки на корректность ввода пользовательских значений, а также на тип цвета. В случае некорректности выводятся соответствующие сообщения.

Следующим шагом происходит вызов функций на обработку, а так же проводятся дополнительные проверки на корректность. В конце функции *main()* происходит запись результата в выходной файл с изображением.

2.2. Функция считывания входного файла

Функция считывания начинается с открытия с входного файла. Далее следует проверка на его открытие. Затем происходит считывание подписи файла. Если подпись не соответствует файлу PNG, то выводится соответствующее сообщение и функция завершается. Далее происходит инициализация указателей на структуры, а так же проверки на успешность произведённой инициализации. Следующим шагом инициализируется функции ввода/вывода по умолчанию для файла PNG в стандартные потоки. В случае неудачи выводится сообщение с помощью функции *jmpbuf*. Затем функция *png_set_sig_bytes()* сохраняет количество байтов подписи файла PNG, которые были считаны из потока PNG. Далее следует функция *png_read_info()*, которая считывает информацию о высоте, ширине, типе цвета, глубине цвета. Затем следует инициализация полей структуры *image* и обновление информации. Наконец, происходит выделение памяти под строки и столбцы матрицы пикселей. Далее следует считывание новой матрицы пикселей, в случае неудачи, с помощью *jmpbuf* выводится сообщение о неудаче. В конце данной функции файл считывания закрывается.

2.3. Функция записи в выходной файл

Функция *write_png_file* отличается от функции записи тем, что отсутствуют проверки подписи PNG – файла, а также появляется заполнение заголовка выходного изображения функцией *png_set_IHDR*. Так же происходит запись информации, хранящейся в полях *info_ptr* и *png_ptr* в PNG – файл. Затем с помощью *png_write_image* записываются пиксели в выходной файл. Далее *png_write_end* записывает конец файла PNG, в который уже были записаны данные изображения. В конце функции происходит очистка памяти, которая выделялась под хранение пикселей, а так же закрытие выходного файла.

2.4. Функция рисования линий

Функция *line* обеспечивает рисование линий толщиной 1. Линия рисуется по алгоритму Брезенхэма. Суть заключается в том, что программа определяет, какой пиксель нужно закрасить на следующей итерации, чтобы получившийся набор пикселей напоминал линию.

2.5. Функция утолщения линии

Функция *thickness* обеспечивает рисование линий заданной толщины и вызывается на каждой итерации закрашивания пикселя в функции *line*. Суть заключается в том, что из каждого пикселя, который является точкой отрезка, рисуется окружность радиусом толщина линии целочисленное поделённое на 2.

2.6. Функция инвертирования пикселей в окружности



Функция *circle* нужна для инверсии пикселей в окружности. Если пользователь на вход подал координаты квадрата, который описан около окружности, то в функции *main()* рассчитывается центр и радиус данной окружности и подаётся данной функции на вход. В самой функции перебирается массив пикселей. Если пиксель лежит внутри окружности, то есть $(x-x_0)^2 + (y-y_0)^2 < R^2$, где (x_0, y_0) – центр окружности, тогда для каждой RGB – компоненты производится инверсия цвета.



2.7. Функция обрезки изображения

Функция *trim* обрезает изображение по прямоугольнику. Для обрезки изображения вся область переносится в левый верхний угол, а так же меняется высота и ширина изображения с помощью двух циклов, один из которых вложенный. Программа учитывает, что произойдёт утечка памяти, поэтому при записи в новый файл память очищает старую карту пикселей, на основании старого значения высоты, которое хранится в поле *true_height*.

3. ТЕСТИРОВАНИЕ


№	Входные данные	Выходные данные	Комментарии
1	./a.out	<справка>	Ответ правильный, не введены опции
2	./a.out -f test.png out.png	./a.out: invalid option -- 'f' <справка>	Ответ правильный, ключа 'f' нет в функционале
3	./a.out -h	<справка>	Ответ правильный, введена опция справки
4	./a.out -l apple.png out.png	Error!!! File couldn't be open!!!	Ответ правильный, такого файла нет на диске
5	./a.out -l -w 100 dog.jpg out.png	Error!!! File is not recognized as a PNG!!!	Ответ правильный, dog.jpg имеет формат JPG
6	./a.out -l pic.png out.png	Error!!! Type of Color is RGB, but must be RGBA	Ответ правильный, у изображения pic.png отсутствует альфа-канал
7	./a.out -l -s 0,0 -e 1000,1000 test.png out.png	Error!!! Invalid values of coordinates!!! Minimal size of coordinates = 0 Maximal size of coordinates of x = 599 Maximal size of coordinates of y = 449	Ответ правильный, ширина картинки равна 600 пикс., высота – 450 пикс.
8	./a.out -l -s 0,0 -e 100,100	Error!!! Invalid values of colors!!! Minimal value of colors = 0	Ответ правильный, цвет

	—rgb 300,200, 200 test.png out.png	Maximal value of colors = 255	задаётся числом от 0 до 255
9	./a.out -l -s 0,0 -e 100,100 —rgb 200,200, 200 -v 1000 test.png out.png	Error!!! Invalid value of transparency!!! Minimal value of transparency = 0 Maximal value of transparency = 255	Ответ правильный, прозрачност ь задаётся числом от 0 до 255
1 0	./a.out -o 1 -s 100,100 - e 200,199 test.png out.png	Error!!! It's not a quadrate!!!	Ответ правильный, заданные координаты не задают квадрат
1 1	./a.out -l -s 134,240 - e 400,400 —rgb 130,180, 90 — width 50 test.png out.png		Ответ правильный
1 2	./a.out -l -s 59,95 - e 2000,100 0 —rgb 100,200, 0 — transpare ncy 100 —width 100 myduck. png out.png		Ответ правильный

1 3	./a.out -l -s 400,500 - e 1000,200 0 --width 1000 mydog.p ng out.png		Ответ правильный
1 4	./a.out -o 1 -s 200,200 - e 300,300 test.png out.png		Ответ правильный

1 5	./a.out -o 1 -s 500,500 - e 2000,200 0 mydog.p ng out.png		Ответ правильный
1 6	./a.out -o 2 —start 800,750 - r 500 myduck. png out.png		Ответ правильный

1 7	./a.out — circle 2 —start 250,50 — radius 5000 test.png out.png		Ответ правильный
1 8	./a.out — trim -s 300,258 - e 400,400 test.png out.png		Ответ правильный
1 9	./a.out -t -s 1000,50 —end 2000,200 0 mydog.p ng out.png		Ответ правильный

2 0	./a.out -t -s 566,1000 —end 2000,125 6 myduck. png out.png		Ответ правильный
--------	--	--	---------------------

ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы была изучена работа с getopt, обработка изображений PNG формата, работа с массивами пикселей, написан механизм обработки некорректных данных.

Реализована программа, которая принимает на вход ключи и параметры, исходя из которых производит обработку PNG изображений по заданному условию.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Язык программирования СИ / Керниган Б., Ритчи Д. СПб.: Издательство “Невский диалект”, 2001. 352 с.
2. Основы программирования на языках С и С++ [Электронный ресурс]
URL: <http://cplusplus.com>
3. https://ru.wikipedia.org/wiki/Алгоритм_Брезенхэма
4. <https://www.linuxhowtos.org/manpages/3/libpng.htm>
5. https://ru.wikibooks.org/wiki/Реализации_алгоритмов/Алгоритм_Брезенхэма
6. <https://refspecs.linuxbase.org>

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Для запуска программы:

- Напишите в терминале `gcc main.c -lpng -lm`
- Напишите в терминале `./a.out --help`
- Изучите справку и на основании её сделайте ту обработку изображения, которая вам потребуется

Файл: main.c

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

void printHelp(){
    printf("\nRange of values of colors and transparency: [0,\n255]\n\n");
    printf("\nAvailable functions: \n\n");
    printf("-l --line -s --start <valueX>,<valueY> -e --end\n<valueX>,<valueY> -c --rgb <valueRED>,<valueGREEN>,<valueBLUE> -v --\ntransparency <value> -w --width <value> <filename_input>\n<filename_output>\n\n");
    printf("-o --circle <l> -s --start <valueX>,<valueY> -e --end\n<valueX>,<valueY> <filename_input> <filename_input>\n\t\t\t\ttor\n");
    printf("-o --circle <2> -s --start <valueX>,<valueY> -r --radius\n<value> <filename_input> <filename_output>\n\n");
    printf("-t --trim s --start <valueX>,<valueY> -e --end\n<valueX>,<valueY> <filename_input> <filename_output>\n\n");
    printf("-h --help -? - help\n\n");
}

struct Png{
    int width, height, true_height;
    png_byte color_type;
    png_byte bit_depth;

    png_structp png_ptr;
    png_infop info_ptr;
    png_bytep *row_pointers;
};

struct Configs{
    char func;
    int x0;
    int y0;
```

```

    int x1;
    int y1;
    int red;
    int green;
    int blue;
    int transparency;
    int width;
    int rradius;
    int type;
    int activate_s;
    int activate_e;
    int activate_c;
    int activate_v;
    int activate_w;
    int activate_r;
};

int read_png_file(char *file_name, struct Png *image) {
    int x,y;
    char header[8];    // 8 is the maximum size that can be checked

    /* open file and test for it being a png */
    FILE *fp = fopen(file_name, "rb");
    if (!fp){
        printf("Error!!!\nFile couldn't be open!!!\n");
        return 1;
    }

    fread(header, 1, 8, fp);
    if (png_sig_cmp(header, 0, 8)){
        // Some error handling: file is not recognized as a PNG
        printf("Error!!!\nFile is not recognized as a PNG!!!\n");
        return 1;
    }

    /* initialize stuff */
    image->png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL,
    NULL, NULL);

    if (!image->png_ptr){
        // Some error handling: png_create_read_struct failed
        printf("Error!!!\npng_create_read_struct failed!!!\n");
        return 1;
    }

    image->info_ptr = png_create_info_struct(image->png_ptr);
    if (!image->info_ptr){
        // Some error handling: png_create_info_struct failed
        printf("Error!!!\npng_create_info_struct failed!!!\n");
        return 1;
    }

    if (setjmp(png_jmpbuf(image->png_ptr))){
        // Some error handling: error during init_io
        printf("Error!!!\nError during png_init_io!!!\n");
        return 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->>true_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);

    /* read file */
    if (setjmp(png_jmpbuf(image->png_ptr))) {
        // Some error handling: error during read_image
        printf("Error!!!\nError during png_read_image!!!\n");
        return 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);
}

int write_png_file(char *file_name, struct Png *image) {
    int x, y;
    /* create file */
    FILE *fp = fopen(file_name, "wb");
    if (!fp) {
        printf("Error!!!\nFile couldn't be open!!!\n");
        return 1;
    }

    /* initialize stuff */
    image->png_ptr = png_create_write_struct(PNG_LIBPNG_VER_STRING, NULL,
NULL, NULL);

    if (!image->png_ptr) {
        printf("Error!!!\npng_create_write_struct failed!!!\n");
        return 1;
    }

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

```

```

    }

    if (setjmp(png_jmpbuf(image->png_ptr))) {
        printf("Error!!!\nError during png_init_io!!!\n");
        return 1;
    }

    png_init_io(image->png_ptr, fp);

    /* write header */
    if (setjmp(png_jmpbuf(image->png_ptr))) {
        printf("Error!!!\nError during png_write_info!!!\n");
        return 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);

    /* write bytes */
    if (setjmp(png_jmpbuf(image->png_ptr))) {
        printf("Error!!!\nError during png_write_image!!!\n");
        return 1;
    }

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

    /* end write */
    if (setjmp(png_jmpbuf(image->png_ptr))) {
        printf("Error!!!\nError during png_write_end!!!\n");
        return 1;
    }

    png_write_end(image->png_ptr, NULL);

    /* cleanup heap allocation */
    for (y = 0; y < image->>true_height; y++)
        free(image->row_pointers[y]);
    free(image->row_pointers);

    fclose(fp);
}

```

```

void thickness(struct Png *image, int x, int y, int radius, int red, int
green, int blue, int transparency){

```

```

    for(int i = 0; i < image->height; i++){
        for(int j = 0; j < image->width; j++){
            double r = (x - j) * (x - j) + (y - i) * (y - i);

```

```

        if (r <= (float)(radius * radius)){
            png_byte *row = image->row_pointers[i];
            png_byte *ptr = &(row[j * 4]);
            ptr[0] = red;
            ptr[1] = green;
            ptr[2] = blue;
            ptr[3] = transparency;
        }
    }
}

void line(struct Png *image, int x0, int y0, int x1, int y1, int red, int
green, int blue, int transparency, int wd){

    int addx, addy;
    float newwd = (float)wd / 2;
    int x = x0, y = y0;
    int deltaX = abs(x1 - x0);
    int deltaY = abs(y1 - y0);
    int signX = x0 < x1 ? 1 : -1;
    int signY = y0 < y1 ? 1 : -1;
    int error = deltaX - deltaY;
    while(x0 != x1 || y0 != y1){
        png_byte *row = image->row_pointers[y0];
        png_byte *ptr = &(row[x0 * 4]);
        ptr[0] = red;
        ptr[1] = green;
        ptr[2] = blue;
        ptr[3] = transparency;

        thickness(image, x0, y0, (wd - 1) / 2, red, green, blue,
transparency);

        int error2 = error;
        if(error2 > -deltaY)
        {
            error -= deltaY;
            x0 += signX;
        }
        if(error2 < deltaX)
        {
            error += deltaX;
            y0 += signY;
        }
    }
    png_byte *row = image->row_pointers[y0];
    png_byte *ptr = &(row[x0 * 4]);
    ptr[0] = red;
    ptr[1] = green;
    ptr[2] = blue;
    thickness(image, x0, y0, (wd - 1) / 2, red, green, blue,
transparency);
}

void circle(struct Png *image, int x, int y, int radius){

```

```

        for(int i = 0; i < image->height; i++){
            for(int j = 0; j < image->width; j++){
                double r = (x - j) * (x - j) + (y - i) * (y - i);
                if (r < (float)(radius * radius)){
                    png_byte *row = image->row_pointers[i];
                    png_byte *ptr = &(row[j * 4]);
                    ptr[0] = 255 - ptr[0];
                    ptr[1] = 255 - ptr[1];
                    ptr[2] = 255 - ptr[2];
                }
            }
        }
    }

void fill(int **arr, int x0, int y0, int x1, int y1){

    for(int i = y0; i <= y1; i++)
        for(int j = x0; j <= x1; j++)
            arr[i][j] = 1;

}

void trim(struct Png *image, int x0, int y0, int x1, int y1){

    for(int i = y0; i <= y1; i++)
        for(int j = x0; j <= x1; j++){
            png_byte *row1 = image->row_pointers[i];
            png_byte *ptr1 = &(row1[j * 4]);
            png_byte *row = image->row_pointers[i-y0];
            png_byte *ptr = &(row[(j - x0) * 4]);
            ptr[0] = ptr1[0];
            ptr[1] = ptr1[1];
            ptr[2] = ptr1[2];
            ptr[3] = ptr1[3];
        }
    image->height = abs(y1-y0);
    image->width = abs(x1-x0);

}

int main(int argc, char* argv[]){
    int k = 0;
    char *pch;
    int xs, ys;
    struct Png image;
    struct Configs config = {' ', 0, 0, 0, 0, 255, 255, 255, 255, 1, 0,
-1, 0};
    char *opts = "lto:s:e:c:r:h?w:v:i";
    struct option longOpts[]={
        {"line", no_argument, NULL, 'l'},
        {"start", required_argument, NULL, 's'},
        {"end", required_argument, NULL, 'e'},
        {"rgb", required_argument, NULL, 'c'},
        {"transparency", required_argument, NULL, 'v'},
        {"width", required_argument, NULL, 'w'},
        {"circle", required_argument, NULL, 'o'},
        {"radius", required_argument, NULL, 'r'},

```

```

        {"trim", no_argument, NULL, 't'},
        {"help", no_argument, NULL, 'h'},
        {"info", no_argument, NULL, 'i'},
        {NULL, 0, NULL, 0}
    };
    int opt;
    int longIndex;
    opt = getopt_long(argc, argv, opts, longOpts, &longIndex);
    while(opt != -1){
        switch(opt){

            case 'l':
                config.func = 'l';
                break;

            case 'o':
                config.func = 'o';
                if (strcmp(optarg, "0") != 0 && atoi(optarg) == 0){
                    printHelp();
                    return 0;
                }
                config.type = atoi(optarg);
                break;

            case 't':
                config.func = 't';
                break;

            case 's':
                k = 0;
                pch = strtok(optarg, ",");
                while(pch != NULL){
                    if (strcmp(pch, "0") != 0 && atoi(pch) == 0){
                        printHelp();
                        return 0;
                    }
                    if (k == 0)
                        config.x0 = atoi(pch);
                    if (k == 1)
                        config.y0 = atoi(pch);
                    k++;
                    pch = strtok (NULL, ",");
                }
                if (k != 2){
                    printHelp();
                    return 0;
                }
                config.activate_s = 1;
                break;

            case 'e':
                k = 0;
                pch = strtok(optarg, ",");
                while(pch != NULL){
                    if (strcmp(pch, "0") != 0 && atoi(pch) == 0){
                        printHelp();
                        return 0;
                    }

```



```

        }
        if (k == 0)
            config.x1 = atoi(pch);
        if (k == 1)
            config.y1 = atoi(pch);
        k++;
        pch = strtok (NULL, ",");
    }
    if (k != 2){
        printHelp();
        return 0;
    }
    config.activate_e = 1;
    break;

case 'c':
    k = 0;
    pch = strtok(optarg, ",");
    while(pch != NULL){
        if (strcmp(pch, "0") != 0 && atoi(pch) == 0){
            printHelp();
            return 0;
        }
        if (k == 0)
            config.red = atoi(pch);
        if (k == 1)
            config.green = atoi(pch);
        if (k == 2)
            config.blue = atoi(pch);
        pch = strtok (NULL, ",");
        k++;
    }
    if (k != 3){
        printHelp();
        return 0;
    }
    config.activate_c = 1;
    break;

case 'v':
    if (strcmp(optarg, "0") != 0 && atoi(optarg) == 0){
        printHelp();
        return 0;
    }
    config.transparency = atoi(optarg);
    config.activate_v = 1;
    break;

case 'r':
    if (strcmp(optarg, "0") != 0 && atoi(optarg) == 0){
        printHelp();
        return 0;
    }
    config.radius = atoi(optarg);

```

```

        config.activate_r = 1;
        break;

    case 'w':
        if (strcmp(optarg, "0") != 0 && atoi(optarg) == 0){
            printHelp();
            return 0;
        }
        config.width = atoi(optarg);
        config.activate_w = 1;
        break;

    case 'h':
    case '?':
        printHelp();
        return 0;
    }

    opt = getopt_long(argc, argv, opts, longOpts, &longIndex);
}

argc -= optind;
argv += optind;

if (argc != 2){
    printHelp();
    return 0;
}

if (read_png_file(argv[0], &image))
    return 0;
if (png_get_color_type(image.png_ptr, image.info_ptr) ==
PNG_COLOR_TYPE_RGB){
    printf("Error!!!\nType of color is RGB, but must be RGBA!!!\n");
    return 0;
}

if (png_get_color_type(image.png_ptr, image.info_ptr) !=
PNG_COLOR_TYPE_RGBA){
    printf("Error!!!\nType of color is not RGBA!!!\n");
    return 0;
}

if ((config.x0 < 0) || (config.x0 > image.width - 1) || (config.y0 <
0) || (config.y0 > image.height - 1) ||
    (config.x1 < 0) || (config.x1 > image.width - 1) || (config.y1
< 0) || (config.y1 > image.height - 1)){
    printf("Error!!!\n");
    printf("Invalid values of coordinates!!!\n");
    printf("Minimal size of coordinates = 0\n");
    printf("Maximal size of coordinates of x = %d\n",
image.width - 1);
    printf("Maximal size of coordinates of y = %d\n",
image.height - 1);
    return 0;
}

```

```

        if ((config.red < 0) || (config.red > 255) || (config.blue < 0) ||
            (config.blue > 255) || (config.green < 0) || (config.green >
255)){
            printf("Error!!!\n");
            printf("Invalid values of colors!!!\n");
            printf("Minimal value of colors = 0\n");
            printf("Maximal value of colors = 255\n");
            return 0;
        }

        if ((config.transparency < 0) || (config.transparency > 255)){
            printf("Error!!!\n");
            printf("Invalid value of transparency!!!\n");
            printf("Minimal value of transparency = 0\n");
            printf("Error!!! Maximal value of transparency = 255\n");
            return 0;
        }

        if ((config.width < 0)){
            printf("Error!!!\n");
            printf("Invalid value of width of line!!!\n");
            printf("Minimal value of width = 0\n");
            return 0;
        }

        if (config.func == 'l'){
            if ((config.activate_s == 0) && (config.activate_e == 0)){
                line(&image, 0, 0, image.width - 1, image.height - 1,
config.red, config.green, config.blue, config.transparency, config.width);
            }
            else if ((config.activate_s == 1) && (config.activate_e == 1)){
                line(&image, config.x0, config.y0, config.x1, config.y1,
config.red, config.green, config.blue, config.transparency, config.width);
            }
            else{
                printHelp();
                return 0;
            }
        }

        if (config.func == 'o'){
            if (config.type == 1){
                if ((config.activate_s == 0) && (config.activate_e ==
0)){
                    config.x0 = 0; config.y0 = 0;
                    config.x1 = image.width - 1; config.y1 =
image.height - 1;
                    xs = (config.x0 + config.x1) / 2;
                    ys = (config.y0 + config.y1) / 2;
                    if (abs(config.x0 - config.x1) != abs(config.y0 -
config.y1)){
                        printf("Error!!!\nIt's not a quadrate!!!\n");
                    }
                }
                else if ((config.activate_s == 1) && (config.activate_e
== 1)){

```

```

        xs = (config.x0 + config.x1) / 2;
        ys = (config.y0 + config.y1) / 2;
        if (abs(config.x0 - config.x1) != abs(config.y0 -
config.y1)){
            printf("Error!!!\nIt's not a quadrate!!!\n");
        }
    }
    else{
        printHelp();
        return 0;
    }
    circle(&image, xs, ys, abs(xs - config.x0));
}

else if (config.type == 2){
    if (config.activate_s == 0){
        config.x0 = (image.width - 1) / 2;
        config.y0 = (image.height - 1) / 2;
        circle(&image, config.x0, config.y0,
config.radiou);
    }
    else if (config.activate_s == 1){
        circle(&image, config.x0, config.y0,
config.radiou);
    }
    else{
        printHelp();
        return 0;
    }
}

else{
    printHelp();
    return 0;
}

}

if (config.func == 't'){
    trim(&image, config.x0, config.y0, config.x1, config.y1);
}

if (write_png_file(argv[1], &image))
    return 0;
return 0;
}

```

ПРИЛОЖЕНИЕ Б

ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ И ОБРАБОТКИ ОШИБОК

```

omagnum@VirtualBox:~$ ./a.out
Range of values of colors and transparency: [0, 255]

Available functions:
-l --line <x> --start <valueX>, <valueY> --end <valueX>, <valueY> --c --rgb <valueRGB>, <valueGRI>, <valueBLU> --v --transparency <value> --w --width <value> <filename_input> <filename_output>
-o --circle <x> --s --start <valueX>, <valueY> --end <valueX>, <valueY> <filename_input> <filename_output>
    or
-o --circle <x> --s --start <valueX>, <valueY> --r --radius <value> <filename_input> <filename_output>
-t --trn <x> --start <valueX>, <valueY> --end <valueX>, <valueY> <filename_input> <filename_output>
-h --help -f - help

omagnum@VirtualBox:~$ ./a.out -b
Range of values of colors and transparency: [0, 255]

Available functions:
-l --line <x> --start <valueX>, <valueY> --end <valueX>, <valueY> --c --rgb <valueRGB>, <valueGRI>, <valueBLU> --v --transparency <value> --w --width <value> <filename_input> <filename_output>
-o --circle <x> --s --start <valueX>, <valueY> --end <valueX>, <valueY> <filename_input> <filename_output>
    or
-o --circle <x> --s --start <valueX>, <valueY> --r --radius <value> <filename_input> <filename_output>
-t --trn <x> --start <valueX>, <valueY> --end <valueX>, <valueY> <filename_input> <filename_output>
-h --help -f - help

omagnum@VirtualBox:~$ ./a.out -f test.png out.png
./a.out: invalid option -- 'f'

Range of values of colors and transparency: [0, 255]

Available functions:
-l --line <x> --start <valueX>, <valueY> --end <valueX>, <valueY> --c --rgb <valueRGB>, <valueGRI>, <valueBLU> --v --transparency <value> --w --width <value> <filename_input> <filename_output>
-o --circle <x> --s --start <valueX>, <valueY> --end <valueX>, <valueY> <filename_input> <filename_output>
    or
-o --circle <x> --s --start <valueX>, <valueY> --r --radius <value> <filename_input> <filename_output>
-t --trn <x> --start <valueX>, <valueY> --end <valueX>, <valueY> <filename_input> <filename_output>
-h --help -f - help

omagnum@VirtualBox:~$ █

```

```

omagnum@VirtualBox:~$ ./a.out -t dog.png out.png
Error!!!
File couldn't be open!!!
omagnum@VirtualBox:~$ ./a.out --line dog.png out.png
Error!!!
File couldn't be open!!!
omagnum@VirtualBox:~$ ./a.out --line dog.jpg out.png
Error!!!
File is not recognized as a PNG!!!
omagnum@VirtualBox:~$ ./a.out --line dash.png out.png
Error!!!
Type of color is RGB, but must be RGBA!!!
omagnum@VirtualBox:~$ ./a.out --line -s -100,100 --end 1000,1000 test.png out.png
Error!!!
Invalid values of coordinates!!!
Minimal size of coordinates = 0
Maximal size of coordinates of x = 599
Maximal size of coordinates of y = 499
omagnum@VirtualBox:~$ ./a.out --line -s 0,100 --end 10,200 --rgb 200,0,5 test.png out.png
Error!!!
Invalid values of colors!!!
Minimal value of colors = 0
Maximal value of colors = 255
omagnum@VirtualBox:~$ ./a.out -l -s 0,0 --e 100,100 --rgb 200,200,200 --v 1000 test.png out.png
Range of values of colors and transparency: [0, 255]

Available functions:
-l --line <x> --start <valueX>, <valueY> --end <valueX>, <valueY> --c --rgb <valueRGB>, <valueGRI>, <valueBLU> --v --transparency <value> --w --width <value> <filename_input> <filename_output>
-o --circle <x> --s --start <valueX>, <valueY> --end <valueX>, <valueY> <filename_input> <filename_output>
    or
-o --circle <x> --s --start <valueX>, <valueY> --r --radius <value> <filename_input> <filename_output>
-t --trn <x> --start <valueX>, <valueY> --end <valueX>, <valueY> <filename_input> <filename_output>
-h --help -f - help

omagnum@VirtualBox:~$ ./a.out -l -s 0,0 --e 100,100 --rgb 200,200,200 --v 1000 test.png out.png
Error!!!
Invalid value of transparency!!!
Minimal value of transparency = 0
Maximal value of transparency = 255
omagnum@VirtualBox:~$ ./a.out -o 1 -s 100,100 --e 200,199 test.png out.png
Error!!!
It's not a quadrat!!!
omagnum@VirtualBox:~$ █

```