**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

КУРСОВАЯ РАБОТА

**по дисциплине «Программирование»**

**Тема: Обработка PNG изображения**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3343 |  | Лобова Е.И. |
| Преподаватель |  | Государкин Я.С. |

Санкт-Петербург

2024

## ЗАДАНИЕ

**НА КУРСОВУЮ РАБОТУ**

Студентка: Лобова Екатерина

Группа: 3343

Тема: Обработка PNG изображения

Условия задания (Вариант 4.24):

Программа должна иметь следующую функции по обработке изображений:

* (1) Рисование квадрата с диагоналями. Флаг для выполнения данной операции: `--squared\_lines`. Квадрат определяется:
  + Координатами левого верхнего угла. Флаг `--left\_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y
  + Размером стороны. Флаг `--side\_size`. На вход принимает число больше 0
  + Толщиной линий. Флаг `--thickness`. На вход принимает число больше 0
  + Цветом линий. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет)
  + Может быть залит или нет (диагонали располагаются “поверх” заливки). Флаг `--fill`. Работает как бинарное значение: флага нет – false , флаг есть – true.
  + Цветом которым он залит, если пользователем выбран залитый. Флаг `--fill\_color` (работает аналогично флагу `--color`)
* (2) Фильтр rgb-компонент. Флаг для выполнения данной операции: `--rgbfilter`. Этот инструмент должен позволять для всего изображения либо установить в диапазоне от 0 до 255 значение заданной компоненты. Функционал определяется
  + Какую компоненту требуется изменить. Флаг `--component\_name`. Возможные значения `red`, `green` и `blue`.
  + В какой значение ее требуется изменить. Флаг `--component\_value`. Принимает значение в виде числа от 0 до 255
* (3) Поворот изображения (части) на 90/180/270 градусов. Флаг для выполнения данной операции: `--rotate`. Функционал определяется
  + Координатами левого верхнего угла области. Флаг `--left\_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y
  + Координатами правого нижнего угла области.  Флаг `--right\_down`, значение задаётся в формате `right.down`, где right – координата по x, down – координата по y
  + Углом поворота. Флаг `--angle`, возможные значения: `90`, `180`, `270`

Дата выдачи задания: 18.03.2024

Дата сдачи реферата: 22.05.2024

Дата защиты реферата: 22.05.2024

**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 квадрата с диагоналями, фильтр rgb-компонент, поворот изображения (части) на 90/180/270 градусов. Сборка проекта осуществляется с помощью утилиты make.

**ВВЕДЕНИЕ**

Цель работы:  
• Изучить структуру изображений в формате PNG.  
• Освоить работу с PNG-изображениями на языке программирования C с использованием библиотеки libpng.  
• Разработать программу на C, реализующую различные функции обработки изображений, такие как считывание, запись и модификация.  
• Обеспечить взаимодействие с программой через интерфейс командной строки для удобства пользователей.

**1. ОПИСАНИЕ СТРУКТУРЫ ПРОГРАММЫ**

Описание структур:

1. *Png* - структура, содержащая данные для работы с PNG изображением: высоту и ширину изображения, тип цветовой модели изображения, глубина цвета, указатели на структуру управления PNG и структуру информации об изображении, количество проходов, необходимых для полной обработки изображения, указатель на массив указателей на строки пикселей изображения.
2. *Сolor* – структура, которая представляет собой цвет, задаваемый тремя компонентами: *r* (красный), *g* (зеленый), *b* (синий).
3. *Squared\_lines* – структура, содержащая информации об аргументах флагов, связанных с функцией *squared\_lines*.
4. *Rgbfilter* – структура, содержащая информации об аргументах флагов, связанных с функцией *rgbfilter*.
5. *Rotate* – структура, содержащая информации об аргументах флагов, связанных с функцией *rotate*.
6. *Configs* – структура, в которую заносятся считанные аргументы командной строки.

Описание функций:

1. *int is\_number(const char\* str)* – проверяет является ли поданная строка числом. Если да, то возвращает 1, иначе 0.
2. *void raise\_error(Configs\* config, char\* message, int return\_code)* – очищает память выделенную в программе динамически, выдаёт сообщение об ошибке и завершает программу.
3. *void init\_configs(Configs\* config)* – инициализирует структуру *config*.
4. *void free\_memory(Configs\* config)* – очищает память, выделенную для элементов структуры *config*.
5. *void parsing\_coordinates(const char\* flag, Configs\* config, const char\* optarg)* - используется для разбора и проверки координат, указанных в аргументах командной строки, и для обновления структуры c*onfig* в соответствии с этими координатами.
6. *void parsing\_angle(Configs\* config, const char\* optarg)* - используется для проверки данных на соответствие углу и для обновления структуры *config* в соответствии с этим аргументом.
7. *void parsing\_number(const char\* flag, Configs\* config, const char\* optarg)* – используется для проверки аргумента командной строки на соответствие неотрицательному числу и для обновления структуры *config* в соответствии с этим аргументом.
8. *void parsing\_color(const char\* flag, Configs\* config, const char\* optarg)* - используется для разбора и проверки цветов, указанных в аргументах командной строки, и для обновления структуры *configs* в соответствии с этими цветами.
9. *void parsing\_component\_name(Configs\* config, const char\* optarg)* – используется для проверки аргумента на соответствие названию компоненте цвета и для обновления структуры *configs* в соответствии с этим аргументом.
10. *void parsing\_component\_value(Configs\* config, const char\* optarg) -* используется для проверки и обновления значения компонента цвета в структуре *config* на основе аргумента командной строки.
11. *void read\_png\_file(char \*file\_name, Png \*image)* – считывает PNG изображение.
12. *void write\_png\_file(char \*file\_name, struct Png \*image)* – записывает PNG изображение.
13. *void rgbfilter(Png \*image, char component\_name, int component\_value) -* устанавливает в диапазоне от 0 до 255 значение заданной компоненты для всего изображения.
14. *void squared\_lines(Configs\* config, Png\* image) -* рисует квадрат с диагоналями.
15. *int rotate(canvas\_t\* canvas, int x0, int y0, int x1, int y1, int angle)* – поворачивает область изображения на 90/180/270 градусов.
16. *void draw\_line(Png\* image, int left, int up, int right, int down, int thickness, Color\* color, int num\_color) –* рисует линию, используя алгоритм Безенхэма.
17. *void set\_pixel(Png\* image, int y, int x, int num\_color, Color\* color) -* изменяет цвет отдельных пикселей в изображении PNG.
18. *int RGB\_or\_RGBA(Png \*image) –* возвращает количество байт на пиксель.
19. *void process\_file(Configs\* config, Png\* image)* – выполняет вызов функций обработки изображения.
20. *int main(int argc, char \*\*argv)* – главная функция программы, осуществляет обработку аргументов командной строки, вызывает функцию *process\_file.*

Функции разделены на тематические модули, каждый из которых отвечает за определенную часть функциональности. Такая структура облегчает добавление, изменение или удаление функций в будущем. Кроме того, использование Makefile для сборки программы упрощает управление зависимостями между модулями и автоматизирует процесс компиляции. Разработанный программный код см. в приложении А.

**ТЕСТИРОВАНИЕ**



Рисунок 1 – изображение для тестирования

1. Тестирование функции *rgbfilter*:

Аргументы для запуска: ./cw --rgbfilter --component\_name red --component\_value 45 -i mitsuri.png -o out1.png



Рисунок 2 – результат работы функции *rgbfilter*

1. Тестирование функции *squared\_lines*:

Аргументы для запуска: ./cw --squared\_lines --left\_up 500.300 --side\_size 123 --thickness 5 --color 0.0.0 --fill\_color 255.255.255 -i mitsuri.png -o out3.png



Рисунок 3 – результат работы функции *squared\_lines*

1. Тестирование функции *rotate*:

Аргументы для запуска:

./cw --rotate --angle 90 --left\_up 500.450 --right\_down 780.600 -i mitsuri.png -o out2.png



Рисунок 4 – результат работы функции *rotate*

1. Тестирование функции *info*:

Аргументы для запуска: ./cw -i mitsuri.png --info

Результат:

Course work for option 4.24, created by Ekaterina Lobova.

height: 1337;

width: 1077;

color\_type: 6;

bit\_depth: 8

**ЗАКЛЮЧЕНИЕ**

В рамках курсовой работы была разработана программа на языке C для обработки изображений в формате PNG. Программа поддерживает набор функций, которые можно выбрать с помощью командной строки (CLI). Сборка проекта осуществляется с помощью утилиты make, что упрощает управление зависимостями и автоматизирует процесс компиляции.

# Приложение А Исходный код программы

Название файла: image\_proccesing.h

#ifndef PROCESSING\_H

#define PROCESSING\_H

#define INFO "height: %d;\nwidth: %d;\ncolor\_type: %d;\nbit\_depth: %d\n"

#include "main.h"

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <png.h>

void rgbfilter(Png \*image, char component\_name, int component\_value);

void squared\_lines(Configs\* config, Png\* image);

void rotate(Configs\* config, Png\* image);

void draw\_line(Png\* image, int left, int up, int right, int down, int thickness, Color\* color, int num\_color);

void set\_pixel(Png\* image, int y, int x, int num\_color, Color\* color);

int RGB\_or\_RGBA(Png \*image);

void process\_file(Configs\* config, Png\* image);

#endif

Название файла: main.h

#ifndef MAIN\_H

#define MAIN\_H

#include "structs.h"

#include "parsing.h"

#include "png\_file\_io.h"

#include "image\_processing.h"

#include <getopt.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#define HELP \

"--squared\_lines - drawing a square with diagonals. Parametrs:\n --left\_up - coordinates of the upper-left corner.the value\

is set in the format left.up, where left is the x coordinate, up is the y coordinate\n --side\_size - the size of the side\

of the square.Accepts a number greater than 0 as input.\n --thickness - line thickness.Accepts a number greater than 0 as in\

put.\n --color - line color.The color is set by the string rrr.ggg.bbb, where rrr/ggg/bbb are the numbers specifying the\

color component. The example --color 255.0.0 sets the color red.\n --fill - a binary flag that determines whether the square will\

be filled in.\n --fill\_color - works similarly to the '--color` flag.\n--rgbfilter - allows you to either set the value of a given\

component for the entire image in the range from 0 to 255. Parametrs:\n --component\_name - which component needs to be changed.\

Possible values are red, green and `blue'.\n --component\_value - to which value the component needs to be changed.Takes\

a value as a number from 0 to 255.\n--rotate - rotate the image (part) by 90/180/270 degrees. Parametrs:\n --left\_up - coordi\

nates of the upper-left corner.the value is set in the format left.up, where left is the x coordinate, up is the y coordinate\n\

--left\_up - coordinates of the lower-right corner of the area. The value is set in the format right.down, where right is \

the x coordinate, down is the y coordinate.\n --angle - angle of rotation. Possible values: 90, 180, 270.\n"

#define START\_SENTENCE "Course work for option 4.24, created by Ekaterina Lobova.\n"

#define DEFAULT\_VALUE "out.png"

#endif

Название файла: parsing.h

#ifndef PARSING\_H

#define PARSING\_H

#include "main.h"

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

int is\_number(const char\* str);

void raise\_error(Configs\* config, char\* message, int return\_code);

void init\_configs(Configs\* config);

void free\_memory(Configs\* config);

void parsing\_coordinates(const char\* flag, Configs\* config, const char\* optarg);

void parsing\_angle(Configs\* config, const char\* optarg);

void parsing\_number(const char\* flag, Configs\* config, const char\* optarg);

void parsing\_color(const char\* flag, Configs\* config, const char\* optarg);

void parsing\_component\_name(Configs\* config, const char\* optarg);

void parsing\_component\_value(Configs\* config, const char\* optarg);

#endif

Название файла: png\_file\_io.h

#ifndef PNGIO\_H

#define PNGIO\_H

#include "main.h"

#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, struct Png \*image);

#endif

Название файла: structs.h

#ifndef STRUCTS\_H

#define STRUCTS\_H

#include <png.h>

typedef struct Png{

int width, 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 Color{

size\_t r;

size\_t g;

size\_t b;

}Color;

typedef struct Squared\_lines{

int n;

int coordinates;

int left, up;

int size;

int thickness;

Color\* thickness\_color;

size\_t fill;

Color\* fill\_color;

}Squared\_lines;

typedef struct Rgbfilter{

int n;

char component\_name;

size\_t component\_value;

}Rgbfilter;

typedef struct Rotate{

int n;

int coordinates;

int left, up, right, down;

size\_t angle;

}Rotate;

typedef struct Configs{

int info;

char\* input;

Squared\_lines\* squared\_lines;

Rgbfilter\* rgbfilter;

Rotate\* rotate;

char\* output;

}Configs;

#endif

Название файла: image\_processing.c

#include "../include/image\_processing.h"

void process\_file(Configs\* config, Png\* image){

int flag = 1;

if (config->input != NULL && config->info == 1){

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

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

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

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

free(image->row\_pointers);

}

if (config->input != NULL && config->output != NULL){

if ((config->rgbfilter)->n == 1){

if ((config->rgbfilter)->component\_name != ' ' && (config->rgbfilter)->component\_value != 256){

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

rgbfilter(image, (config->rgbfilter)->component\_name, (config->rgbfilter)->component\_value);

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

} else flag = 0;

}

if ((config->squared\_lines)->n == 1){

if ((config->squared\_lines)->coordinates != 0 && (config->squared\_lines)->thickness\_color != NULL && (config->squared\_lines)->thickness != -1){

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

squared\_lines(config, image);

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

} else flag = 0;

}

if ((config->rotate)->n == 1){

if ((config->rotate)->coordinates!=0 && (config->rotate)->angle!=0){

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

rotate(config, image);

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

} else flag = 0;

}

}

if (!flag)

raise\_error(config, "there are not enough flags to set parameters\n", 42);

free\_memory(config);

}

int RGB\_or\_RGBA(Png \*image){

if (png\_get\_color\_type(image->png\_ptr, image->info\_ptr) == PNG\_COLOR\_TYPE\_RGB){

return 3;

}

if (png\_get\_color\_type(image->png\_ptr, image->info\_ptr) == PNG\_COLOR\_TYPE\_RGBA){

return 4;

}

}

void rgbfilter(Png \*image, char component\_name, int component\_value) {

int x,y,color;

color = RGB\_or\_RGBA(image);

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 \* color]);

switch(component\_name){

case 'r':

ptr[0] = component\_value;

break;

case 'g':

ptr[1] = component\_value;

break;

case 'b':

ptr[2] = component\_value;

break;

}

}

}

}

void set\_pixel(Png\* image, int y, int x, int num\_color, Color\* color){

if (y < 0 || y >= image->height || x < 0 || x >= image->width)

return;

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

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

ptr[0] = color->r;

ptr[1] = color->g;

ptr[2] = color->b;

}

void draw\_line(Png\* image, int left, int up, int right, int down, int thickness, Color\* color, int num\_color) {

int deltaX = abs(right - left);

int deltaY = abs(down - up);

int signX = left < right ? 1 : -1;

int signY = up < down ? 1 : -1;

int error = deltaX - deltaY;

while(1)

{

int rectX = left - thickness / 2;

int rectY = up - thickness / 2;

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

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

set\_pixel(image, y, x, num\_color, color);

}

}

if (left == right && up == down) break;

int error2 = error \* 2;

if(error2 > -deltaY)

{

error -= deltaY;

left += signX;

}

if(error2 < deltaX)

{

error += deltaX;

up += signY;

}

}

}

void squared\_lines(Configs\* config, Png\* image){

int num\_color = RGB\_or\_RGBA(image);

int left = (config->squared\_lines)->left;

int up = (config->squared\_lines)->up;

int size = (config->squared\_lines)->size;

int thickness = (config->squared\_lines)->thickness;

if (thickness % 2==0) thickness--;

if ((config->squared\_lines)->fill\_color != NULL){

for (int y = up; y < up + size - 1; y++){

for (int x = left; x < left+size-1; x++){

set\_pixel(image, y, x, num\_color, (config->squared\_lines)->fill\_color);

}

}

}

draw\_line(image, left, up, left+size-1, up, thickness, (config->squared\_lines)->thickness\_color, num\_color);

draw\_line(image, left, up+size-1, left+size-1, up+size-1, thickness, (config->squared\_lines)->thickness\_color, num\_color);

draw\_line(image, left, up, left, up+size-1, thickness, (config->squared\_lines)->thickness\_color, num\_color);

draw\_line(image, left+size-1, up, left+size-1, up+size-1, thickness, (config->squared\_lines)->thickness\_color, num\_color);

draw\_line(image, left, up, left+size-1, up+size-1, thickness, (config->squared\_lines)->thickness\_color, num\_color);

draw\_line(image, left+size-1, up, left, up+size-1, thickness, (config->squared\_lines)->thickness\_color, num\_color);

}

void rotate(Configs\* config, Png\* image){

int num\_color = RGB\_or\_RGBA(image);

int left = (config->rotate)->left;

int up = (config->rotate)->up;

int right = (config->rotate)->right;

int down = (config->rotate)->down;

if (left < 0 || left > image->width || up < 0 || up > image->height || right < 0 || right > image->width || down < 0 || down > image->height){

raise\_error(config, "Rotation area is not on image.\n", 43);

}

int height\_copy = down - up;

int width\_copy = right - left;

//копирование области

png\_bytep \*row\_pointers\_copy = (png\_bytep \*) malloc(sizeof(png\_bytep) \* height\_copy);

for (int y = 0; y < height\_copy; y++)

row\_pointers\_copy[y] = (png\_byte \*) malloc(sizeof(png\_byte) \* num\_color \* width\_copy);

for (int y = up; y < down; y++){

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

for (int x = left; x < right; x++){

png\_byte \*ptr = &(row\_copy[x \* num\_color]);

for (int i = 0; i < 3; i++){

row\_pointers\_copy[y - up][(x - left)\*num\_color + i] = ptr[i];

}

}

}

//поворот скопированной области

int height\_rotate, width\_rotate;

int count = 4 - (config->rotate)->angle / 90;

for (int i = 1; i<=count; i++){

height\_rotate = width\_copy;

width\_rotate = height\_copy;

png\_bytep \*row\_pointers\_rotate = (png\_bytep \*) malloc(height\_rotate \* sizeof(png\_bytep));

for (int y = 0; y < height\_rotate; y++)

row\_pointers\_rotate[y] = (png\_byte \*) malloc(num\_color \* width\_rotate\* sizeof(png\_byte));

for (int y = 0; y < height\_copy; y++){

png\_byte \*row\_copy = row\_pointers\_copy[y];

for (int x = 0; x < width\_copy; x++){

png\_byte \*ptr = &(row\_copy[x \* num\_color]);

for (int i = 0; i < 3; i++){

row\_pointers\_rotate[x][(width\_rotate - 1 - y)\*num\_color + i] = ptr[i];

}

}

}

free(row\_pointers\_copy);

row\_pointers\_copy = row\_pointers\_rotate;

width\_copy = width\_rotate;

height\_copy = height\_rotate;

}

//вставка на исходное изображение

int rotate\_left = (left+right)/2 - width\_rotate/2;

int rotate\_up = (up+down)/2 - height\_rotate/2;

Color color;

for (int y = rotate\_up; y < rotate\_up+height\_rotate; y++){

for (int x = rotate\_left; x < rotate\_left + width\_rotate; x++){

color.r = row\_pointers\_copy[y-rotate\_up][(x-rotate\_left)\*num\_color];

color.g = row\_pointers\_copy[y-rotate\_up][(x-rotate\_left)\*num\_color + 1];

color.b = row\_pointers\_copy[y-rotate\_up][(x-rotate\_left)\*num\_color + 2];

set\_pixel(image, y, x, num\_color, &color);

}

}

free(row\_pointers\_copy);

}

Название файла: main.c

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

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

printf(START\_SENTENCE);

Configs config;

init\_configs(&config);

const char\* shortOpts = "i:o:h!";

const struct option longOpts[] = {

{ "help", no\_argument, NULL, 'h' },

{ "input", required\_argument, NULL, 'i' },

{ "output", required\_argument, NULL, 'o' },

{ "info", no\_argument, NULL, '!'},

{"squared\_lines", no\_argument, NULL, 0},

{"rgbfilter",no\_argument, NULL, 0},

{"rotate", no\_argument, NULL, 0},

{"component\_name", required\_argument, NULL, 0},

{"component\_value", required\_argument, NULL, 0},

{"left\_up", required\_argument, NULL, 0},

{"right\_down", required\_argument, NULL, 0},

{"angle", required\_argument, NULL, 0},

{"side\_size", required\_argument, NULL, 0},

{"thickness", required\_argument, NULL, 0},

{"color", required\_argument, NULL, 0},

{"fill", no\_argument, NULL, 0},

{"fill\_color", required\_argument, NULL, 0},

{ NULL, 0, NULL, 0 }

};

int opt;

int longIndex;

struct Png image;

while((opt = getopt\_long(argc, argv, shortOpts, longOpts, &longIndex)) != -1){

switch(opt){

case 'i':

if (config.input != NULL){

free(config.input);

}

config.input = (char\*)malloc((strlen(optarg)+1)\*sizeof(char));

strncpy(config.input, optarg, strlen(optarg));

break;

case 'o':

if (config.output != NULL){

free(config.output);

}

config.output = (char\*)malloc((strlen(optarg)+1)\*sizeof(char));

strncpy(config.output, optarg, strlen(optarg));

break;

case 'h':

printf(HELP);

break;

case '!':

config.info = 1;

break;

case '?':

printf("found unknown option\n");

free\_memory(&config);

return 0;

case 0:

if (longOpts[longIndex].name == "squared\_lines"){

(config.squared\_lines)->n = 1;

}

if (longOpts[longIndex].name == "rgbfilter"){

(config.rgbfilter)->n = 1;

}

if (longOpts[longIndex].name == "rotate"){

(config.rotate)->n = 1;

}

if (longOpts[longIndex].name == "component\_name"){

parsing\_component\_name(&config, optarg);

}

if (longOpts[longIndex].name == "component\_value"){

parsing\_component\_value(&config, optarg);

}

if (longOpts[longIndex].name == "left\_up"){

parsing\_coordinates(longOpts[longIndex].name, &config, optarg);

}

if (longOpts[longIndex].name == "right\_down"){

parsing\_coordinates(longOpts[longIndex].name, &config, optarg);

}

if (longOpts[longIndex].name == "angle"){

parsing\_angle(&config, optarg);

}

if (longOpts[longIndex].name == "color" || longOpts[longIndex].name == "fill\_color"){

parsing\_color(longOpts[longIndex].name, &config, optarg);

}

if (longOpts[longIndex].name == "side\_size"){

parsing\_number(longOpts[longIndex].name, &config, optarg);

}

if (longOpts[longIndex].name == "thickness"){

parsing\_number(longOpts[longIndex].name, &config, optarg);

}

if (longOpts[longIndex].name == "fill"){

(config.squared\_lines)->fill = 1;

}

}

}

if (config.input == NULL && optind == argc - 1){

config.input = (char\*)malloc((strlen(argv[argc - 1])+1)\*sizeof(char));

strncpy(config.input, argv[argc - 1], strlen(argv[argc - 1]));

}

if (config.output == NULL){

config.output = (char\*)malloc((strlen(DEFAULT\_VALUE)+1)\*sizeof(char));

strncpy(config.output, DEFAULT\_VALUE, strlen(DEFAULT\_VALUE));

}

process\_file(&config, &image);

return 0;

}

Название файла: parsing.c

#include "../include/parsing.h"

int is\_number(const char\* str) {

int len = strlen(str);

if (str[0] == '-') len -=1;

int has\_digit = 0;

while (\*str != '\0') {

if (\*str >= '0' && \*str <= '9') {

has\_digit+= 1;

}

str++;

}

return has\_digit == len;

}

void raise\_error(Configs\* config, char\* message, int return\_code){

free\_memory(config);

fprintf(stderr, message, return\_code);

exit(return\_code);

}

void init\_configs(Configs\* config){

config->info = 0;

config->input = NULL;

config->squared\_lines = (Squared\_lines\*)malloc(sizeof(Squared\_lines));

(config->squared\_lines)->n = 0;

(config->squared\_lines)->coordinates = 0;

(config->squared\_lines)->left = 0;

(config->squared\_lines)->up = 0;

(config->squared\_lines)->size = -1;

(config->squared\_lines)->thickness = -1;

(config->squared\_lines)->thickness\_color = NULL;

(config->squared\_lines)->fill = 0;

(config->squared\_lines)->fill\_color = NULL;

config->rgbfilter = (Rgbfilter\*)malloc(sizeof(Rgbfilter));

(config->rgbfilter)->component\_name = ' ';

(config->rgbfilter)->component\_value = 256;

config->rotate = (Rotate\*)malloc(sizeof(Rotate));

(config->rotate)->coordinates = 0;

(config->rotate)->left = 0;

(config->rotate)->up = 0;

(config->rotate)->right = 0;

(config->rotate)->down = 0;

(config->rotate)->angle = 0;

config->output = NULL;

}

void free\_memory(Configs\* config){

if (config->input != NULL) free(config->input);

if (config->output != NULL) free(config->output);

if (config->rgbfilter != NULL) free(config->rgbfilter);

if (config->squared\_lines != NULL){

if ((config->squared\_lines)->thickness\_color != NULL) free((config->squared\_lines)->thickness\_color);

if ((config->squared\_lines)->fill\_color != NULL) free((config->squared\_lines)->fill\_color);

free(config->squared\_lines);

}

if (config->rotate != NULL) free(config->rotate);

}

void parsing\_coordinates(const char\* flag, Configs\* config, const char\* optarg){

int x, y;

if (sscanf(optarg, "%d.%d", &x, &y) != 2){

raise\_error(config, "incorrect form of agrumeng for flag\n", 41);

} else {

if (strcmp(flag, "left\_up") == 0){

(config->squared\_lines)->coordinates = 1;

(config->squared\_lines)->left = x;

(config->squared\_lines)->up = y;

(config->rotate)->left = x;

(config->rotate)->up = y;

} else {

(config->rotate)->coordinates = 1;

(config->rotate)->right = x;

(config->rotate)->down = y;

}

}

}

void parsing\_angle(Configs\* config, const char\* optarg){

if (strcmp(optarg,"90") == 0 || strcmp(optarg,"180") == 0 || strcmp(optarg,"270") == 0){

(config->rotate)->angle = atoi(optarg);

} else {

raise\_error(config, "argument for --agngle must be one of 90, 180 or 270\n", 41);

}

}

void parsing\_number(const char\* flag, Configs\* config, const char\* optarg){

if (is\_number(optarg) && optarg[0]!='-'){

if (strcmp(flag, "thickness")==0)

(config->squared\_lines)->thickness = atoi(optarg);

else (config->squared\_lines)->size = atoi(optarg);

} else{

raise\_error(config, "incorrect value of the argument\n", 41);

}

}

void parsing\_color(const char\* flag, Configs\* config, const char\* optarg){

int r, g, b;

if (sscanf(optarg, "%d.%d.%d", &r, &g, &b)!=3){

raise\_error(config, "incorrect form of agrument for flag\n", 41);

} else {

if (r >= 0 && r <= 255 && g >= 0 && g <= 255 && b >= 0 && b <= 255){

if (strcmp(flag, "color") == 0){

(config->squared\_lines)->thickness\_color = (Color\*)malloc(sizeof(Color));

((config->squared\_lines)->thickness\_color)->r = r;

((config->squared\_lines)->thickness\_color)->g = g;

((config->squared\_lines)->thickness\_color)->b = b;

} else {

(config->squared\_lines)->fill\_color = (Color\*)malloc(sizeof(Color));

((config->squared\_lines)->fill\_color)->r = r;

((config->squared\_lines)->fill\_color)->g = g;

((config->squared\_lines)->fill\_color)->b = b;

}

} else {

raise\_error(config, "incorrect value of the color component\n", 41);

}

}

}

void parsing\_component\_name(Configs\* config, const char\* optarg){

if (strcmp(optarg,"red") == 0 || strcmp(optarg ,"blue") == 0 || strcmp(optarg,"green") == 0){

(config->rgbfilter)->component\_name = optarg[0];

}else{

raise\_error(config, "the argument of the component\_name flag must have one of the following values: red, green, or blue\n", 41);

}

}

void parsing\_component\_value(Configs\* config, const char\* optarg){

if (is\_number(optarg)){

int num = atoi(optarg);

if (num >= 0 && num <= 255){

(config->rgbfilter)->component\_value = num;

}else{

raise\_error(config, "incorrect value of the color component\n", 41);

}

}else{

raise\_error(config, "incorrect value of the color component\n", 41);

}

}

Название файла: png\_file\_io.c

#include "../include/png\_file\_io.h"

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

int x,y;

char header[8];

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

if (!fp){

fprintf(stderr, "file could not be opened\n");

exit(40);

}

fread(header, 1, 8, fp);

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

fprintf(stderr, "file is not recognized as a PNG\n");

fclose(fp);

exit(40);

}

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

if (!image->png\_ptr){

fprintf(stderr, "png\_create\_read\_struct failed\n");

fclose(fp);

exit(40);

}

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

if (!image->info\_ptr){

fprintf(stderr, "png\_create\_info\_struct failed\n");

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

fclose(fp);

exit(40);

}

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

fprintf(stderr, "error during init\_io\n");

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

fclose(fp);

exit(40);

}

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

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

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

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

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

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

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

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

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

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

fprintf(stderr, "error during read\_image\n");

fclose(fp);

exit(40);

}

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){

fprintf(stderr, "file could not be opened\n");

exit(40);

}

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

if (!image->png\_ptr){

fprintf(stderr, "png\_create\_write\_struct failed\n");

fclose(fp);

exit(40);

}

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

if (!image->info\_ptr){

fprintf(stderr, "png\_create\_info\_struct failed\n");

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

fclose(fp);

exit(40);

}

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

fprintf(stderr, "error during init\_io\n");

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

fclose(fp);

exit(40);

}

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

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

fprintf(stderr, "error during writing header\n");

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

fclose(fp);

exit(40);

}

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))){

fprintf(stderr, "error during writing bytes\n");

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

fclose(fp);

exit(40);

}

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

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

fprintf(stderr, "error during end of write\n");

fclose(fp);

exit(40);

}

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

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

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

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

free(image->row\_pointers);

fclose(fp);

}

Название файла: Makefile

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

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

CC = gcc

CFLAGS = -lpng

all: cw

cw: $(OBJECTS)

$(CC) $(OBJECTS) $(CFLAGS) -o cw

sources/%.o : sources/%.c

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

clean:

rm -rf sources/\*.o cw