**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

КУРСОВАЯ РАБОТА

**по дисциплине «Программирование»**

**Тема:** Обработка изображения в формате PNG

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Жучков О.Д. |
| Преподаватель |  | Государкин Я.С. |

Санкт-Петербург

2024

## ЗАДАНИЕ

**НА КУРСОВУЮ РАБОТУ**

Студент: Жучков Олег

Группа: 3343

Тема: Обработка изображения в формате PNG

Условия задания (Вариант 4.21):

Программа должна иметь следующие функции по обработке изображений:

1. Рисование прямоугольника. Флаг для выполнения данной операции: `--rect`. Он определяется:
   * Координатами левого верхнего угла. Флаг `--left\_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y.
   * Координатами правого нижнего угла. Флаг `--right\_down`, значение задаётся в формате `right.down`, где right – координата по x, down – координата по y.
   * 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).
   * Прямоугольник может быть залит или нет. 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. Рисование правильного шестиугольника. Флаг для выполнения данной операции: `--hexagon`. Шестиугольник определяется:
   * Координатами его центра и радиусом в который он вписан. 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. 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).
   * Шестиугольник может быть залит или нет. Флаг `--fill`. Работает как бинарное значение: флага нет – false , флаг есть – true.
   * Цветом которым залит шестиугольник, если пользователем выбран залитый. Флаг `--fill\_color` (работает аналогично флагу `--color`).
3. Копирование заданной области. Флаг для выполнения данной операции: `--copy`. Функционал определяется:
   * Координатами левого верхнего угла области-источника. Флаг `--left\_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y.
   * Координатами правого нижнего угла области-источника. Флаг `--right\_down`, значение задаётся в формате `right.down`, где right – координата по x, down – координата по y.
   * Координатами левого верхнего угла области-назначения. Флаг `--dest\_left\_up`, значение задаётся в формате `left.up`, где left – координата по x, up – координата по y.

Дата выдачи задания: 18.03.2024

Дата сдачи реферата: 23.05.2024

Дата защиты реферата: 23.05.2024

**АННОТАЦИЯ**

В процессе работы создан проект на языке программирования C, обрабатывающий PNG изображения. Для работы с изображениями в программе используется библиотека libpng. В программе реализованы следующие задачи: рисование прямоугольника, рисование правильного шестиугольника, копирование области. Программа имеет интерфейс командной строки (используется библиотека getopt). Функции программы разделены по нескольким файлам, для сборки проекта используется утилита make.

**ВВЕДЕНИЕ**

Цель работы заключается в изучении структуры PNG файлов и освоении работы с ними. Необходимо разработать программу с функциями обработки изображения, таких как рисование геометрических фигур, и для взаимодействия с пользователем необходимо реализовать интерфейс командной строки. Программа должна обрабатывать возможные ошибки, такие как некорректный пользовательский ввод.

**1. ОПИСАНИЕ СТРУКТУР ДАННЫХ**

**1.1. Структура png\_t**

Данная структура предназначена для хранения информации об изображении и его содержании. Используется при чтении и записи PNG файла, при обработке изображения.

**1.2. Структура rgb\_t**

Данная структура хранит цвет в формате RGB (значения красного, зеленого, красного цветов). Используеся для рисования на изображении, копирования пикселей.

**1.3. Структура params\_t**

Структура содержит обработанные аргументы, введённые пользователем с помощью интерфейса командной строки.

**2. Описание структуры программы**

**2.1. params.c**

* *void print\_help()* – вывод справки.
* *void init\_params(params\_t\* params)* – инициализирует поля params\_t.
* *void raise\_arg\_error() –* выводит сообщение об ошибке при некорректном вводе.
* *void parse\_coords(char\* arg, int\* x, int\* y) –* обработка введённых координат формата “x.y”.
* *void parse\_rgb(char\* arg, rgb\_t\* rgb)* – обработка введённого цвета формата “r.g.b”.
* *void parse\_params(params\_t\* optparams, int argc, char\*\* argv)* – обработка пользовательского ввода через интерфейс командной строки.

**2.2. read\_write\_png.c**

* *void read\_png(char\* file\_name, png\_t\* image)* – открытие и считывание PNG файла.
* *void write\_png(char\* file\_name, png\_t\* image)* – запись обработанного изображения в PNG файл.

**2.3. png\_edit.c**

* *rgb\_t int\_to\_rgb(int r, int g, int b)* – создание структуры rgb\_t из 3 значений int.
* *rgb\_t png\_byte\_to\_rgb(png\_byte r, png\_byte g, png\_byte b* – создание структуры rgb\_t из 3 значений png\_byte.
* *bool check\_coords(png\_t\* image, int x, int y) –* проверка координат на вхождение в размеры изображения*.*
* *void set\_pixel(png\_t\* image, int x, int y, rgb\_t rgb) –* закрашивание данного пикселя данным цветом.
* *rgb\_t get\_pixel(png\_t\* image, int x, int y) –* получить цвет данного пикселя и вернуть в виде rgb\_t.
* *void copy\_pixel\_to(png\_t\* image, int x0, int y0, int x1, int y1) –* копирование значение цвета одного пикселя в другой.

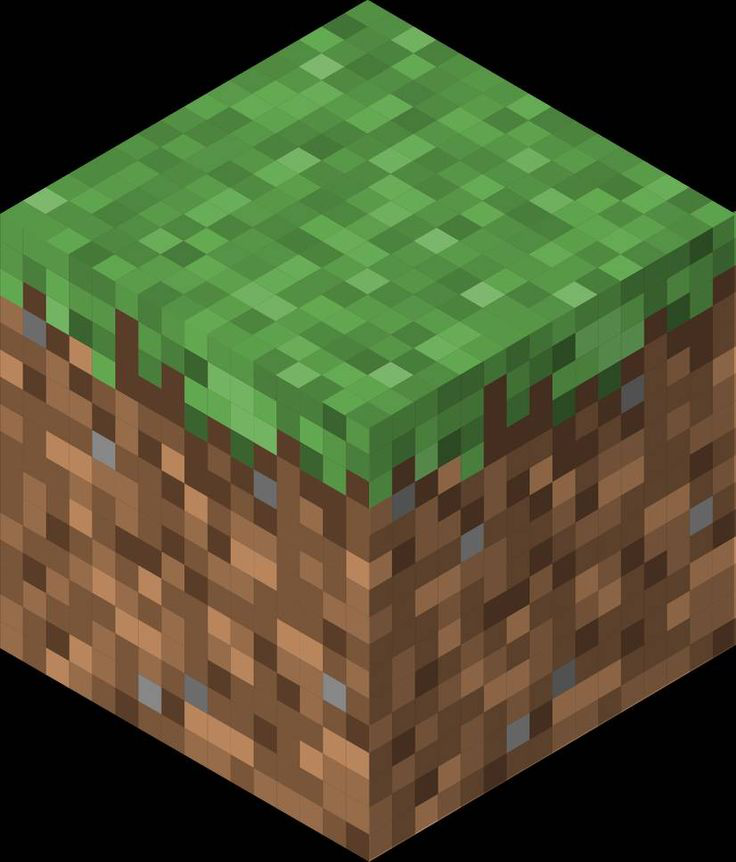
**2.4. png\_draw**

* *void draw\_line(png\_t\* image, int x0, int y0, int x1, int y1, rgb\_t rgb, int thickness)* – рисование линии заданной толщины с использованием алгоритма Брезенхэма.
* *void fill\_circle(png\_t\* image, int x0, int y0, rgb\_t rgb, int radius)* – рисование закрашенного круга (используется для толщины).
* *void draw\_circle(png\_t\* image, int x0, int y0, rgb\_t rgb, int radius) –* рисование окружности с использованием алгоритма Брезенхэма*.*
* *void draw\_rectangle(png\_t\* image, int x0, int y0, int x1, int y1, rgb\_t rgb, int thickness, bool fill, rgb\_t fillrgb) –* рисование прямоугольника.
* *void fill\_rectangle(png\_t\* image, int x0, int y0, int x1, int y1, rgb\_t rgb) –* закрашивание прямоугольника.
* *void draw\_hexagon(png\_t\* image, int x0, int y0, rgb\_t rgb, int radius, int thickness, bool fill, rgb\_t rgbfill) –* рисование шестиугольника.
* *void fill\_hexagon(png\_t\* image, int x0, int y0, rgb\_t rgb, int radius) –* закрашивание шестиугольника.
* *void copy\_area\_to(png\_t\* image, int x0, int y0, int x1, int y1, int x2, int y2) –* копирование выделенной области*.*

**2.5. main.c**

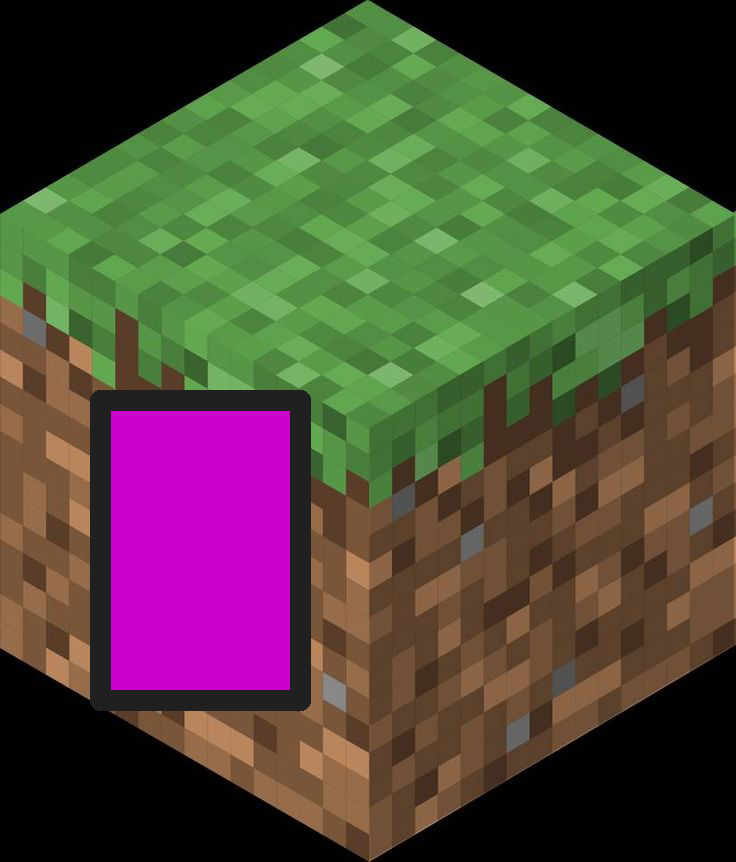
В функции main() вызывется функция обработки введённых аргументов и вызываются функции открытия, обработки, записи изображения*.*

**ТЕСТИРОВАНИЕ**

 Рисунок 1 – изображение для тестирования

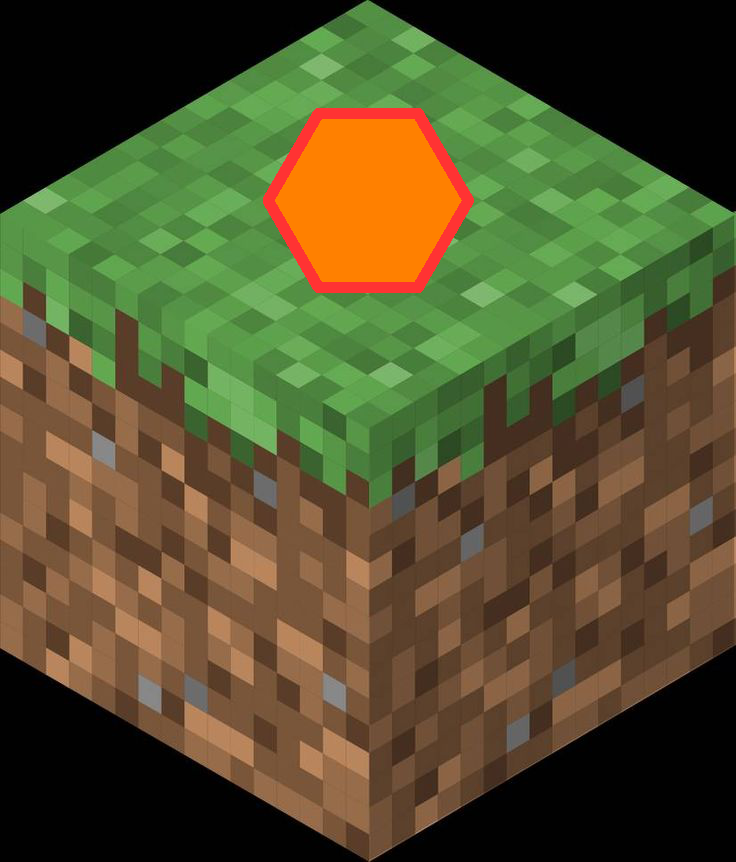
1. Задание *rectangle*:

Аргументы запуска: ./cw --rect --left\_up 100.400 --right\_down 300.700 --color 32.32.32 --thickness 20 --fill --fill\_color 204.0.204 in.png

Рисунок 2 – результат работы для задания *rectangle*

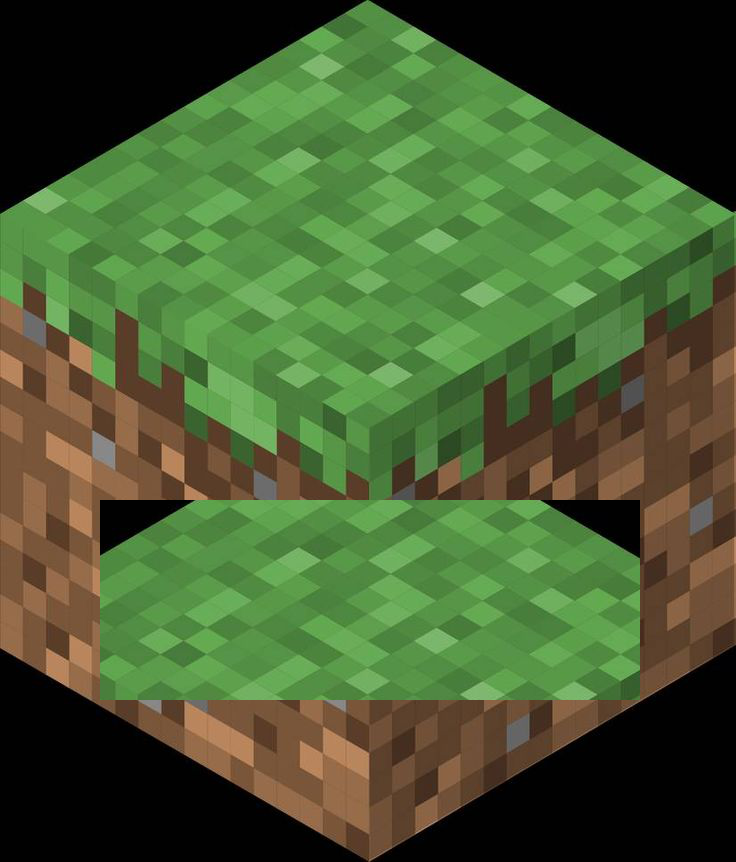
1. Задание *hexagon*:

Аргументы запуска: ./cw --hexagon --center 368.200 --radius 100 --color 255.51.51 --fill --fill\_color 255.128.0 --thickness 10 in.png

Рисунок 3 – результат работы для задания *hexagon*

1. Задание *copy*:

Аргументы запуска: ./cw --copy --left\_up 100.100 --right\_down 640.300 --dest\_left\_up 100.500 in.png

Рисунок 4 – результат работы для задания *ornament*

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы создана программа на языке программирования C, осуществляющая обработку PNG изображения с помощью библиотеки libpng. Программа может выполнять такие задачи, как: рисование прямоугольника, шестиугольника, копирование области. Выбор задачи и ввод аргументов производится через интерфейс командной строки. Проект разделен на несколько файлов и компилируется с помощью make.

# Приложение А Исходный код программы

Название файла: params.h

#ifndef param

#define param

#include <getopt.h>

#include <structs.h>

void init\_params(params\_t\* params);

void raise\_arg\_error();

void parse\_coords(char\* arg, int\* x, int\* y);

void parse\_rgb(char\* arg, rgb\_t\* rgb);

void parse\_params(params\_t\* optparams, int argc, char\*\* argv);

#endif

Название файла: png\_draw.h

#ifndef draw

#define draw

#include <structs.h>

#include <png.h>

#include <stdbool.h>

void draw\_line(png\_t\* image, int x0, int y0, int x1, int y1, rgb\_t rgb, int thickness);

void draw\_line\_symmetrical\_point(png\_t\* image, int x0, int y0, int x1, int y1, int x2, int y2, rgb\_t rgb, int thickness);

void fill\_circle(png\_t\* image, int x0, int y0, rgb\_t rgb, int radius);

void draw\_circle(png\_t\* image, int x0, int y0, rgb\_t rgb, int radius);

void draw\_rectangle(png\_t\* image, int x0, int y0, int x1, int y1, rgb\_t rgb, int thickness, bool fill, rgb\_t fillrgb);

void fill\_rectangle(png\_t\* image, int x0, int y0, int x1, int y1, rgb\_t rgb);

void draw\_hexagon(png\_t\* image, int x0, int y0, rgb\_t rgb, int radius, int thickness, bool fill, rgb\_t rgbfill);

void fill\_hexagon(png\_t\* image, int x0, int y0, rgb\_t rgb, int radius);

void copy\_area\_to(png\_t\* image, int x0, int y0, int x1, int y1, int x2, int y2);

#endif

Название файла: png\_edit.h

#ifndef pngedit

#define pngedit

#include <png.h>

#include <structs.h>

#include <stdbool.h>

rgb\_t int\_to\_rgb(int r, int g, int b);

rgb\_t png\_byte\_to\_rgb(png\_byte r, png\_byte g, png\_byte b);

bool check\_coords(png\_t\* image, int x, int y);

void set\_pixel(png\_t\* image, int x, int y, rgb\_t rgb);

rgb\_t get\_pixel(png\_t\* image,int x, int y);

void copy\_pixel\_to(png\_t\* image, int x0, int y0, int x1, int y1);

#endif

Название файла: read\_write\_png.h

#ifndef read\_write

#define read\_write

#include <png.h>

#include <structs.h>

void read\_png(char\* file\_name, png\_t\* image);

void write\_png(char\* file\_name, png\_t\* image);

#endif

Название файла: structs.h

#ifndef structs

#define structs

#include <png.h>

#include <stdbool.h>

typedef struct {

png\_byte r, g, b;

} rgb\_t;

typedef struct {

png\_structp png\_ptr;

png\_infop info\_ptr;

png\_byte bit\_depth, color\_type;

int number\_passes;

int width, height;

png\_bytepp rows;

} png\_t;

typedef struct{

char\* input;

char\* output;

bool info;

bool help;

bool rect;

int left\_up\_x;

int left\_up\_y;

int right\_down\_x;

int right\_down\_y;

int thickness;

rgb\_t color;

bool fill;

rgb\_t fill\_color;

bool color\_input;

bool fcolor\_input;

bool hexagon;

int center\_x;

int center\_y;

int radius;

bool copy;

int dest\_left\_up\_x;

int dest\_left\_up\_y;

} params\_t;

#endif

Название файла: main.c

#include <stdio.h>

#include <getopt.h>

#include <stdlib.h>

#include <string.h>

#include "structs.h"

#include "read\_write\_png.h"

#include "png\_draw.h"

#include "png\_edit.h"

#include "params.h"

void print\_info(png\_t\* image){

printf("Image info:\nHeight: %d; Width: %d; Color type: %d; Bit depth: %d\n", image->height, image->width, image->color\_type, image->bit\_depth);

}

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

params\_t params;

init\_params(&params);

parse\_params(&params, argc, argv);

png\_t image;

if (strcmp(params.input,params.output)==0){

puts("Input and output can't be the same file!");

exit(49);

}

read\_png(params.input, &image);

int cnt = 0;

if (params.rect) cnt++;

if (params.hexagon) cnt++;

if (params.copy) cnt++;

if (cnt > 1){

raise\_arg\_error();

}

if (params.rect){

if (params.left\_up\_x != -1 && params.left\_up\_y != -1 &&

params.right\_down\_x != -1 && params.right\_down\_y != -1 &&

params.color\_input &&

params.thickness != -1)

if (params.fill && (!params.fcolor\_input))

raise\_arg\_error();

else

draw\_rectangle(&image, params.left\_up\_x, params.left\_up\_y, params.right\_down\_x, params.right\_down\_y, params.color, params.thickness, params.fill, params.fill\_color);

else

raise\_arg\_error();

}

if (params.hexagon){

if (params.center\_x != -1 && params.center\_y != -1 && params.radius != -1 &&

params.color\_input &&

params.thickness != -1)

if (params.fill && (!params.fcolor\_input))

raise\_arg\_error();

else

draw\_hexagon(&image, params.center\_x, params.center\_y, params.color, params.radius, params.thickness, params.fill, params.fill\_color);

else

raise\_arg\_error();

}

if (params.copy){

if (params.left\_up\_x!= -1 && params.left\_up\_y != -1 && params.right\_down\_x != -1 &&

params.right\_down\_y != -1 && params.dest\_left\_up\_x != -1 && params.dest\_left\_up\_y != -1)

copy\_area\_to(&image, params.left\_up\_x, params.left\_up\_y, params.right\_down\_x, params.right\_down\_y, params.dest\_left\_up\_x, params.dest\_left\_up\_y);

else

raise\_arg\_error();

}

if (params.info) print\_info(&image);

write\_png(params.output, &image);

}

Название файла: params.c

#include <params.h>

#include <getopt.h>

#include <stdlib.h>

#include <string.h>

#include <png\_edit.h>

void print\_help(){

printf("Course work for option 4.21, created by Zhuchkov Oleg.\n"

"-h --help: Display parameters information\n"

"-i --input: Name of the input png. REQUIRED (if not using --input put the filename at the end of command line)\n"

"-o --output: Name for the output png. out.png by default\n"

"--info: Display png information\n"

"--rect: Draw a rectangle\n"

"--hexagon: Draw a hexagon\n"

"--copy: Copy a rectangular area\n"

"--left\_up: First corner of an area (for rectangle and copy)\n"

"--right\_down: Second corner of an area\n"

"--center: Center of the hexagon\n"

"--radius: Radius of the hexagon\n"

"--thickness: Thickness of drawn figures\n"

"--color: Color of drawn figures\n"

"--fill: Fill drawn shape\n"

"--fill\_color: Color to fill drawn shape\n"

"Coords format: x.y Color format: r.g.b\n");

}

void init\_params(params\_t\* params){

params->input = NULL;

params->output = "out.png";

params->info = false;

params->help = false;

params->rect = false;

params->left\_up\_x = -1;

params->left\_up\_y = -1;

params->right\_down\_x = -1;

params->right\_down\_y = -1;

params->thickness = -1;

params->color.r = -1;

params->color.g = -1;

params->color.b = -1;

params->fill = false;

params->fill\_color.r = -1;

params->fill\_color.g = -1;

params->fill\_color.b = -1;

params->hexagon = false;

params->center\_x = -1;

params->center\_y = -1;

params->radius = -1;

params->copy=false;

params->dest\_left\_up\_x=-1;

params->dest\_left\_up\_y=-1;

params->color\_input=false;

params->fcolor\_input=false;

}

void raise\_arg\_error(){

puts("Incorrect or missing argument(s)!");

exit(42);

}

void parse\_coords(char\* arg, int\* x, int\* y){

char xx[30], yy[30];

int i=0, j=0, l=0;

while (arg[i] != '\0'){

if (arg[i]== '.'){

xx[l] = '\0';

if (j==1) raise\_arg\_error();

j++;

i++;

l = 0;

continue;

}

if ((arg[i]>'9'||arg[i]<'0')&&arg[i]!='-') raise\_arg\_error();

if (j==0){

xx[l] = arg[i];

}

else{

yy[l] = arg[i];

}

i++;

l++;

}

yy[l] = '\0';

if(strlen(xx) == 0 || strlen(yy) == 0) raise\_arg\_error();

\*x = strtol(xx,NULL,10);

\*y = strtol(yy,NULL,10);

}

void parse\_rgb(char\* arg, rgb\_t\* rgb){

char r[30], g[30], b[30];

int i=0, j=0, l=0;

while (arg[i] != '\0'){

if (arg[i]== '.'){

if (j==2) raise\_arg\_error();

if (j==1){

g[l] = '\0';

}

if (j==0){

r[l] = '\0';

}

j++;

i++;

l=0;

continue;

}

if (arg[i]>'9'||arg[i]<'0') raise\_arg\_error();

if (j==0){

r[l] = arg[i];

}

if (j==1){

g[l] = arg[i];

}

else

b[l] = arg[i];

i++;

l++;

}

b[l] = '\0';

if (strlen(r) == 0 || strlen(g) == 0 || strlen(b) == 0) raise\_arg\_error();

int rr = strtol(r,NULL,10);

int gg = strtol(g,NULL,10);

int bb = strtol(b,NULL,10);

if (rr > 255 || gg > 255 || bb > 255) raise\_arg\_error();

\*rgb = int\_to\_rgb(rr,gg,bb);

}

void parse\_params(params\_t\* optparams, int argc, char\*\* argv){

opterr = 0;

static struct option options[] = {

{"input", 1, NULL, 'i'},

{"output", 1, NULL, 'o'},

{"info", 0, NULL, 400},

{"help", 0, NULL, 'h'},

{"rect", 0, NULL, 401},

{"left\_up", 1, NULL, 402},

{"right\_down", 1, NULL, 403},

{"thickness", 1, NULL, 404},

{"color", 1, NULL, 405},

{"fill", 0, NULL, 406},

{"fill\_color", 1, NULL, 407},

{"hexagon", 0, NULL, 408},

{"center", 1, NULL, 409},

{"radius", 1, NULL, 410},

{"copy", 0, NULL, 411},

{"dest\_left\_up", 1, NULL, 412},

{0,0,0,0}

};

int c = 1;

int arg;

while(c != -1){

c = getopt\_long(argc, argv, "hi:o:",options,NULL);

if (c == -1) break;

switch (c){

case 'h':{

optparams->help = true;

break;}

case 'i':{

optparams->input = optarg;

break;

}

case 'o':{

optparams->output = optarg;

break;

}

case 400:{

optparams->info = true;

break;

}

case 401:{

optparams->rect = true;

break;

}

case 402:{

parse\_coords(optarg, &(optparams->left\_up\_x), &(optparams->left\_up\_y));

break;

}

case 403:{

parse\_coords(optarg, &(optparams->right\_down\_x), &(optparams->right\_down\_y));

break;

}

case 404:{

arg = strtol(optarg, NULL, 10);

if (arg <= 0) raise\_arg\_error();

optparams->thickness = arg;

break;

}

case 405:{

parse\_rgb(optarg, &(optparams->color));

optparams->color\_input=true;

break;

}

case 406:{

optparams->fill = true;

break;

}

case 407:{

parse\_rgb(optarg, &(optparams->fill\_color));

optparams->fcolor\_input=true;

break;

}

case 408:{

optparams->hexagon = true;

break;

}

case 409:{

parse\_coords(optarg, &(optparams->center\_x), &(optparams->center\_y));

break;

}

case 410:{

arg = strtol(optarg, NULL, 10);

if (arg <= 0) raise\_arg\_error();

optparams->radius = arg;

break;

}

case 411:{

optparams->copy = true;

break;

}

case 412:{

parse\_coords(optarg, &(optparams->dest\_left\_up\_x), &(optparams->dest\_left\_up\_y));

break;

}

case '?':{

raise\_arg\_error();

break;

}

default:{

raise\_arg\_error();

break;

}

}

}

if (optparams->help){

print\_help();

exit(0);

}

if (optparams->input == NULL && optind == argc - 1)

{

optparams->input = calloc(strlen(argv[argc - 1]) + 1, sizeof(char));

strncpy(optparams->input, argv[argc - 1], strlen(argv[argc - 1]) + 1);

}

if (optparams->input == NULL){

puts("No input file!");

exit(49);

};

}

Название файла: png\_draw.c

#include <png.h>

#include <stdbool.h>

#include <math.h>

#include <stdlib.h>

#include <png\_draw.h>

#include <png\_edit.h>

#include <structs.h>

void draw\_line(png\_t\* image, int x0, int y0, int x1, int y1, rgb\_t rgb, int thickness){

int deltax = abs(x1 - x0);

int deltay = abs(y1 - y0);

int temp;

bool swap = false;

if(deltay > deltax){

temp = x0;

x0 = y0;

y0 = temp;

temp = x1;

x1 = y1;

y1 = temp;

temp = deltax;

deltax = deltay;

deltay = temp;

swap = true;

}

if (y0 > y1){

temp = x0;

x0 = x1;

x1 = temp;

temp = y0;

y0 = y1;

y1 = temp;

}

int error = 0;

int deltaerr = (deltay + 1);

int y = y0;

int diry = 1;

if (x0 > x1)

for (int x = x0; x>=x1; x--){

if (thickness==1)

if (swap)

set\_pixel(image,y,x,rgb);

else

set\_pixel(image,x,y,rgb);

else

if (swap)

fill\_circle(image,y,x,rgb,(thickness+1)/2);

else

fill\_circle(image,x,y,rgb,(thickness+1)/2);

error = error + deltaerr;

if (error >= (deltax + 1))

{

y = y + diry;

error = error - (deltax + 1);

}

}

else

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

if (thickness==1)

if (swap)

set\_pixel(image,y,x,rgb);

else

set\_pixel(image,x,y,rgb);

else

if (swap)

fill\_circle(image,y,x,rgb,(thickness+1)/2);

else

fill\_circle(image,x,y,rgb,(thickness+1)/2);

error = error + deltaerr;

if (error >= (deltax + 1))

{

y = y + diry;

error = error - (deltax + 1);

}

}

}

void draw\_line\_symmetrical\_point(png\_t\* image, int x0, int y0, int x1, int y1, int x2, int y2, rgb\_t rgb, int thickness){

int deltax = abs(x1 - x0);

int deltay = abs(y1 - y0);

int temp;

bool swap = false;

if(deltay > deltax){

temp = x0;

x0 = y0;

y0 = temp;

temp = x1;

x1 = y1;

y1 = temp;

temp = deltax;

deltax = deltay;

deltay = temp;

swap = true;

}

if (y0 > y1){

temp = x0;

x0 = x1;

x1 = temp;

temp = y0;

y0 = y1;

y1 = temp;

}

int error = 0;

int deltaerr = (deltay + 1);

int y = y0;

int diry = 1;

int ax, ay;

if (x0 > x1)

for (int x = x0; x>=x1; x--){

if (swap){

ax = y;

ay = x;

}

else{

ax = x;

ay = y;

}

if (thickness==1){

set\_pixel(image,ax,ay,rgb);

set\_pixel(image,x0\*2 - ax,ay,rgb);

set\_pixel(image,ax,y0\*2 - ay,rgb);

set\_pixel(image,x0\*2 - ax,y0\*2 - ay,rgb);

}

else{

fill\_circle(image,ax,ay,rgb,thickness/2);

fill\_circle(image,x0\*2 - ax,ay,rgb,thickness/2);

fill\_circle(image,ax,y0\*2 - ay,rgb,thickness/2);

fill\_circle(image,x0\*2 - ax,y0\*2 - ay,rgb, thickness/2);

}

error = error + deltaerr;

if (error >= (deltax + 1))

{

y = y + diry;

error = error - (deltax + 1);

}

}

else

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

if (swap){

ax = y;

ay = x;

}

else{

ax = x;

ay = y;

}

if (thickness==1){

set\_pixel(image,ax,ay,rgb);

set\_pixel(image,x0\*2 - ax,ay,rgb);

set\_pixel(image,ax,y0\*2 - ay,rgb);

set\_pixel(image,x0\*2 - ax,y0\*2 - ay,rgb);

}

else{

fill\_circle(image,ax,ay,rgb,thickness/2);

fill\_circle(image,x0\*2 - ax,ay,rgb,thickness/2);

fill\_circle(image,ax,y0\*2 - ay,rgb,thickness/2);

fill\_circle(image,x0\*2 - ax,y0\*2 - ay,rgb, thickness/2);

}

error = error + deltaerr;

if (error >= (deltax + 1))

{

y = y + diry;

error = error - (deltax + 1);

}

}

}

void fill\_circle(png\_t\* image, int x0, int y0, rgb\_t rgb, int radius){

if (x0-radius > image->width) return;

if (y0-radius > image->height) return;

if (x0+radius < 0) return;

if (y0+radius < 0) return;

draw\_circle(image, x0, y0, rgb, radius);

for(int y=-radius; y<=radius; y++){

if (y0+y < 0) continue;

if (y0+y > image->height) break;

for(int x=-radius; x<=radius; x++){

if(x0+x < 0) continue;

if (x0+x > image->width) break;

if(x\*x+y\*y <= radius\*radius)

set\_pixel(image, x0+x, y0+y, rgb);

}

}

}

void draw\_circle(png\_t\* image, int x0, int y0, rgb\_t rgb, int radius){

int x = 0;

int y = radius;

int delta = 3 - 2 \* radius;

while (y >= x){

set\_pixel(image,x0 + x, y0 + y,rgb);

set\_pixel(image,x0 + x, y0 - y,rgb);

set\_pixel(image,x0 - x, y0 + y,rgb);

set\_pixel(image,x0 - x, y0 - y,rgb);

set\_pixel(image,x0 + y, y0 + x,rgb);

set\_pixel(image,x0 + y, y0 - x,rgb);

set\_pixel(image,x0 - y, y0 + x,rgb);

set\_pixel(image,x0 - y, y0 - x,rgb);

if (delta < 0)

delta = delta + 4\*x++ + 6;

else

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

}

}

void draw\_rectangle(png\_t\* image, int x0, int y0, int x1, int y1, rgb\_t rgb, int thickness, bool fill, rgb\_t fillrgb){

if (fill) fill\_rectangle(image, x0, y0, x1, y1, fillrgb);

draw\_line(image, x0, y0, x1, y0, rgb, thickness);

draw\_line(image, x0, y1, x1, y1, rgb, thickness);

draw\_line(image, x0, y0, x0, y1, rgb, thickness);

draw\_line(image, x1, y0, x1, y1, rgb, thickness);

}

void fill\_rectangle(png\_t\* image, int x0, int y0, int x1, int y1, rgb\_t rgb){

int temp;

if (x0 > x1){

temp = x0;

x0 = x1;

x1 = temp;

}

if (y0 > y1){

temp = y0;

y0 = y1;

y1 = temp;

}

if (x0 >= image->width) return;

if (y0 >= image->height) return;

if (x1 > image->width) x1 = image->width;

if (y1 > image->height) y1 = image->height;

if (x0 < 0) x0 = 0;

if (y0 < 0) y0 = 0;

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

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

set\_pixel(image, x, y, rgb);

}

void draw\_hexagon(png\_t\* image, int x0, int y0, rgb\_t rgb, int radius, int thickness, bool fill, rgb\_t rgbfill){

int x1, y1;

x1 = x0 - radius / 2;

y1 = y0 - sqrt(3) / 2 \* radius;

if (fill) fill\_hexagon(image, x0, y0, rgbfill, radius);

draw\_line(image, x1,y1, x0-radius,y0, rgb, thickness);

draw\_line(image, x1,y1, x0\*2-x1,y1, rgb, thickness);

draw\_line(image, x0\*2-x1,y1, x0+radius,y0, rgb, thickness);

draw\_line(image, x0+radius,y0, x0\*2-x1,y0\*2-y1, rgb, thickness);

draw\_line(image, x0\*2 - x1,y0\*2-y1, x1,y0\*2-y1, rgb, thickness);

draw\_line(image, x1,y0\*2-y1, x0-radius, y0, rgb, thickness);

}

void fill\_hexagon(png\_t\* image, int x0, int y0, rgb\_t rgb, int radius){

int x1, y1, x2, y2;

x1 = x0 - radius;

y1 = y0;

x2 = x0 - radius / 2;

y2 = y0 - sqrt(3) / 2 \* radius;

fill\_rectangle(image, x2, y2, x0 \* 2 - x2, y0 \* 2 - y2, rgb);

int deltax = abs(x2- x1);

int deltay = abs(y2 - y1);

int error = 0;

int deltaerr = (deltax + 1);

int x = x2;

int dirx = -1;

for (int y = y2; y<=y1; y++){

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

set\_pixel(image,x,yy,rgb);

set\_pixel(image,x0\*2 - x,yy,rgb);

set\_pixel(image,x,y0\*2 - yy,rgb);

set\_pixel(image,x0\*2 - x,y0\*2 - yy,rgb);

}

error = error + deltaerr;

if (error >= (deltay + 1))

{

x = x + dirx;

error = error - (deltay + 1);

}

}

}

void copy\_area\_to(png\_t\* image, int x0, int y0, int x1, int y1, int x2, int y2){

int temp;

if (x0 > x1){

temp = x0;

x0 = x1;

x1 = temp;

}

if (y0 > y1){

temp = y0;

y0 = y1;

y1 = temp;

}

for (int x = 0; x < x1 - x0; x++)

for (int y = 0; y < y1 - y0; y++)

copy\_pixel\_to(image, x0+x, y0+y, x2 + x, y2 + y);

}

Название файла: png\_edit.c

#include <png.h>

#include <png\_edit.h>

#include <structs.h>

rgb\_t int\_to\_rgb(int r, int g, int b){

rgb\_t color;

color.r = (png\_byte)r;

color.g = (png\_byte)g;

color.b = (png\_byte)b;

return color;

}

rgb\_t png\_byte\_to\_rgb(png\_byte r, png\_byte g, png\_byte b){

rgb\_t color;

color.r = r;

color.g = g;

color.b = b;

return color;

}

bool check\_coords(png\_t\* image, int x, int y){

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

}

void set\_pixel(png\_t\* image, int x, int y, rgb\_t rgb){

if (check\_coords(image, x, y)) return;

png\_bytep pixel = (png\_bytep)(image->rows)[y]+x\*3;

pixel[0] = rgb.r;

pixel[1] = rgb.g;

pixel[2] = rgb.b;

}

rgb\_t get\_pixel(png\_t\* image,int x, int y){

png\_byte r,g,b;

r = ((png\_bytep)(image->rows[y]))[x\*3];

g = ((png\_bytep)(image->rows[y]))[x\*3+1];

b = ((png\_bytep)(image->rows[y]))[x\*3+2];

return png\_byte\_to\_rgb(r,g,b);

}

void copy\_pixel\_to(png\_t\* image, int x0, int y0, int x1, int y1){

if (check\_coords(image, x0, y0) || check\_coords(image, x1, y1)) return;

rgb\_t color;

color = get\_pixel(image, x0, y0);

set\_pixel(image,x1,y1,color);

}

Название файла: read\_write\_png.c

#include <read\_write\_png.h>

#include <stdlib.h>

#include <stdio.h>

#include <png.h>

void read\_png(char\* file\_name, png\_t\* image) {

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

if (fp == NULL){

puts("Cannot open file!");

exit(40);

}

png\_byte header[8];

fread(header, sizeof(png\_byte), 8, fp);

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

puts("Wrong file signature! Is it a png?");

exit(40);

}

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

if (image->png\_ptr == NULL){

puts("Cannot create read struct!");

exit(40);

}

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

if (image->png\_ptr == NULL){

puts("Cannot create info struct!");

exit(40);

}

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

puts("Error when creating read struct!");

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

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

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

puts("Color type is not RGB!");

exit(40);

}

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

image->rows = (png\_bytepp)malloc(sizeof(png\_bytepp) \* image->height);

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

image->rows[i] = (png\_bytep)calloc(image->width \* 3, sizeof(png\_byte));

png\_read\_image(image->png\_ptr, image->rows);

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

fclose(fp);

}

void write\_png(char\* file\_name, png\_t\* image){

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

if (fp == NULL){

puts("Cannot open write file!");

exit(41);

}

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

if (image->png\_ptr == NULL){

puts("Cannot create write struct!");

exit(41);

}

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

if (image->png\_ptr == NULL){

puts("Cannot create info struct!");

exit(41);

}

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

puts("Error when creating write struct!");

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

exit(41);

}

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

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);

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

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

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

puts("Error when writing PNG!");

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

exit(41);

}

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

fclose(fp);

}