

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

КУРСОВАЯ РАБОТА
по дисциплине «Программирование»
Тема: Разработка программы для обработки PNG изображений

Студент гр. 3344

Тукалкин В.А.

Преподаватель

Глазунов С.А.

Санкт-Петербург

2024

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Тукалкин В.А.

Группа 3344

Тема работы: «Разработка программы для обработки PNG изображений»

Общие сведения

- Формат картинки PNG (рекомендуем использовать библиотеку *libpng*)
- файл всегда соответствует формату *PNG*
- Реализация интерфейса должна быть с использованием *getopt*
- все поля стандартных *PNG* заголовков в выходном файле должны иметь те же значения что и во входном (разумеется, кроме тех, которые должны быть изменены).
- Программа обязательно должна иметь CLI

Содержание пояснительной записки:

- Содержание
- Введение
- Описание работы
- Описание работы программы
- Тестирование
- Заключение
- Список использованных источников.

Предполагаемый объем пояснительной записки:

Не менее 30 страниц.

Дата выдачи задания: 18.03.2024

Дата сдачи реферата: 22.05.2024

Дата защиты реферата: 24.05.2023

Студент

Тукалкин В.А.

Преподаватель

Глазунов С.А.

АННОТАЦИЯ

Курсовая работа представляет собой программу на языке Си, которая обрабатывает *PNG*-изображение. Программа имеет *CLI* (интерфейс командной строки) для ввода параметров обработки *PNG*-файла пользователем.

Для чтения и записи изображения была использована библиотека *libpng*; для обработки изображения использовались функции стандартных библиотек; для анализов аргументов командной строки использовалась библиотека *getopt*.

Методы

исследования включают анализ стандартных *PNG* заголовков, обработку входных параметров. Результатом работы является функциональная программа,

способная обрабатывать изображения в соответствии с заданными параметрами, сохраняя структуру и характеристики исходного файла.

СОДЕРЖАНИЕ

	Введение	4
1.	Подключаемые библиотеки, макроопределения, структуры	7
2.	Функции	8
2.1	Функции чтения и записи <i>PNG</i> -файла	8
2.2	Дополнительные, вспомогательные функции	9
2.3	Основные функции	15
2.4	Функция <i>main</i>	18
	Заключение	19
	Список использованных источников	20
	Приложение А. Результаты тестирования	21
	Приложение Б. Исходный код программы	24

ВВЕДЕНИЕ

Целью данной работы является разработка программы, обрабатывающей *PNG*-изображение, на языке Си.

Для достижения поставленной цели требуется решить следующие задачи:

- разработать функции чтения и записи *PNG*-файла, реализовать записи в структуру *Png*;
- разработать функцию рисования треугольника на изображении;
- разработать функцию поиска наибольшего прямоугольника заданного цвета и перекрашивания в другой цвет;
- разработать функцию, создающую коллаж размера $N*M$ из изображения;
- разработать функцию рисования линии;
- написать *Makefile* для сборки программы;
- протестировать разработанную программу.

1. ОПИСАНИЕ ВАРИАНТА РАБОТЫ

Задание

Программа обязательно должна иметь CLI (опционально дополнительное использование GUI).

Программа должна реализовывать весь следующий функционал по обработке png-файла

Общие сведения

- Формат картинки PNG (рекомендуем использовать библиотеку libpng)
- без сжатия
- файл может не соответствовать формату PNG, т.е. необходимо проверка на PNG формат. Если файл не соответствует формату PNG, то программа должна завершиться с соответствующей ошибкой.
- обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями.
- все поля стандартных PNG заголовков в выходном файле должны иметь те же значения что и во входном (разумеется кроме тех, которые должны быть изменены).

Программа должна 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) Сделать рамку в виде узора. Флаг для выполнения данной операции: `-ornament`. Рамка определяется:
- Узором. Флаг `--pattern`. Обязательные значения: `rectangle` и `circle`, `semicircles`. Также можно добавить свои узоры (красивый узор можно получить, используя фракталы). Подробнее здесь: https://se.moevm.info/doku.php/courses:programming:cw_spring_ornament
 - Цветом. Флаг `--color` (цвет задаётся строкой `rrr.ggg.bbb`, где `rrr/ggg/bbb` – числа, задающие цветовую компоненту. пример `--color 255.0.0` задаёт красный цвет)
 - Шириной. Флаг `--thickness`. На вход принимает число больше 0
 - Количеством. Флаг `--count`. На вход принимает число больше 0
 - При необходимости можно добавить дополнительные флаги для необозначенных узоров
- 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'`
- Углом поворота

Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции.

2. ПОДКЛЮЧАЕМЫЕ БИБЛИОТЕКИ, МАКРООПРЕДЕЛЕНИЯ, СТРУКТУРЫ

Для корректной работы программы подключены стандартные библиотеки языка Си: *stdlib.h*, *stdio.h*, *math.h*, *ctype.h*, *string.h*, *ctype.h*.

Также подключена библиотека *png.h* для чтения и записи *PNG*-файла и библиотека *getopt.h* для анализа аргументов командной строки.

Определены две структуры:

- Структура *Png*, хранящая параметры изображения: высоту *height* и ширину *width*, цветовой тип *color_type*, битовую глубину *bit_depth*, количество каналов *channels*, *png_ptr* (указатель на *png_struct*), *info_ptr* (указатель на *png_info*), *row_pointers* (указатель на сетку пикселей).

3. ФУНКЦИИ

3.1. Функции чтения и записи PNG-файла

Функция *read_png_file()* принимает на вход указатель на строку *file_name* – имя *PNG*-файла, который нужно считать, а также указатель на структуру *Png image*; с помощью функций из библиотеки *libpng* данные из *IHDR* считываются и записываются в структуру *image*; также происходит динамическое выделение памяти для сетки пикселей с последующей записью в структуру *image*; если на каком-либо этапе считывания *PNG*-файла возникает ошибка, то выводится сообщение о том, какую именно часть файла не удалось считать, и программа завершается.

Функция *write_png_file()* принимает на вход указатель на строку *file_name* – имя *PNG*-файла, куда требуется записать изображение, а также указатель на структуру *Png image*; с помощью функций из библиотеки *libpng* информация о изображении, а также сетка пикселей записывается в *PNG*-файл; если на этапе записи *PNG*-файла возникает ошибка, то она корректно обрабатывается: выводится сообщение о том, что именно не удалось записать, и программа завершается. В конце функция очищается динамически выделенная память для изображения.

3.2. Дополнительные, вспомогательные функции

Функция *print_help()* обращается к файлу *help*, расположенному в директории программы, и печатает в поток вывода справку по работе с программой (принцип работы: открывает файл *help* (в нём расположена справка по работе с программой), находящийся в рабочей директории; циклом *while* происходит посимвольное считывание файла и выводение символов в поток вывода; после вывода всех символов файл закрывается).

Функция *print_info()* принимает на вход указатель *img* на структуру *Png* и печатает в поток вывода основную информацию о *PNG*-файле (принцип работы: получает данные из структуры *Png img* и печатает их в поток вывода).

Функция *is_number()* принимает на вход указатель *number* на *char*, проверяет, является ли данная строка числом, и если является, то возвращает значение 1, иначе – 0 (принцип работы: циклом *for* проходит по строке и с помощью *if* и функции *isdigit* проверяет каждый элемент).

Функция *check_color_and_coordinates()* принимает на вход указатель *string* на *char* и число *number_dot*, проверяет корректность значения цвета и координат, и если корректно, то возвращает значение 1, иначе – 0 (принцип работы: циклом *for* проходит по строке и с помощью *if* и функции *isdigit* проверяет каждый элемент, для каждого элемента происходит проверка полноты параметром для цвета).

Функция *set_pixel()* принимает на вход указатель *img* на структуру *Png*, координаты пикселя *x* и *y*, указатель на массив с цветом *color* длины 3; изменяет цвет пикселя на переданный в функцию цвет (принцип работы: переходит в сетке пикселей к заданным координатам пикселя и изменяет значения трёх каналов (*RGB*) в соответствии с переданным цветом *color*).

Функция *draw_ring()* принимает на вход указатель *image* на структуру *Png*, координаты центра окружности $x0$ и $y0$, радиус окружности *radius*, толщину линии число *thickness*, массив с цветом *color[3]*; изменяет цвет пикселей, находящихся на расстоянии *radius* от центра окружности(предназначение: вспомогательная функция для *function_ornament* опции *semicircles*).

Функция *draw_circle()* принимает на вход указатель *image* на структуру *Png*, координаты центра окружности $x0$ и $y0$, толщину линии число *thickness*, массив с цветом *color[3]*; изменяет цвет пикселей, находящихся на расстоянии не более *radius* от центра окружности(предназначение: вспомогательная функция для *draw_ring*).

3.3. Основные функции

Функция *function_rect()* принимает на вход указатель *image* на структуру *Png*, координаты левого верхнего *left_up[2]* и правого нижнего *right_down[2]* углов области, толщину линии число *thickness*, массив с цветом *color[3]*, для заливки прямоугольника цветом число *fill*, массив с цветом заливки *fill_color[3]*; изменяет цвет пикселей, находящихся в заданной области и пиксели на расстоянии *thickness/2* от области (принцип работы: делает координаты корректными, с помощью циклов *for* проходит по всем пикселям, находящимся на сторонах заданной области и меняет их цвет на *color*, если *fill=1* при помощи циклов *for* проходит по каждому пикселю в области от *left_up+thickness/2* по *right_down-thickness/2* для *x* и *y*, меняя цвет на *fill_color*).

Функция *function_ornament()* принимает на вход указатель *image* на структуру *Png*, номер опции *pattern*, массив с цветом *color[3]*, толщину линии число *thickness*, число повторений *count*; изменяет цвет пикселей, находящихся на изображении, в итоге рисуя рамку (принцип работы: вычисляется центр изображения и радиус, при помощи *switch* выбирается нужная рамка: 1 – циклом *for* до значения *count* вычисляются координаты левого верхнего и правого нижнего углов, затем рисуются линии; 2 – циклами *for* от центра до края изображения проверяется каждый пиксель и если он находится на расстоянии *radius* от центра пиксели изменяют цвет на *color*, пиксели имеют зеркальные координаты относительно центра; 3 – первым циклом *for* со значением до *count* вычисляются координаты центров и функций *draw_ring* рисуются окружности по вертикале, второй цикл *for* аналогично первому, но для горизонтали).

Функция *function_rotate()* принимает на вход указатель *image* на структуру *Png*, координаты левого верхнего *left_up[2]* и правого нижнего *right_down[2]* углов области, угл поворота *angle*; поворачивает изображение (часть) относительно центра заданной области на 90/180/270 градусов (принцип работы: проверяются координаты на корректность, создаётся двумерный массив

arr и в него копируется заданная область с запасом, вычисляется центр области, при помощи *switch* выбирается угол и поворачивается область: 90 – вычисляются координаты новой точки, которая будет левым верхнем углом, относительно области, циклами *for* совершается проход по всем пикселям области, вычисляются новые координаты и меняются цвета в соответствии с массивом *arr*, если координаты имеют значение меньше 0, то цвет чёрный; 180 – координаты новой точки, которая будет левым верхнем углом является правый нижний угол, циклами *for* совершается проход по всем пикселям области, вычисляются новые координаты и меняются цвета в соответствии с массивом *arr*; 270 – вычисляются координаты новой точки, которая будет левым верхнем углом, относительно области, циклами *for* совершается проход по всем пикселям области, вычисляются новые координаты и меняются цвета в соответствии с массивом *arr*, если координаты имеют значение меньше 0, то цвет чёрный).

3.4. Функция *main*

В функции *main()* реализовано управление программой с помощью аргументов командной строки. С помощью функций из библиотеки *getopt* происходит считывание параметров обработки *PNG*-изображения после соответствующих флагов, проверка на корректность передаваемых значений (в случае ввода некорректного параметра, неверного флага или недостаточного количества аргументов для обработки программа напишет в поток вывода соответствующую информацию об ошибке и завершит работу), и последующая запись параметров. Далее происходит чтение *PNG*-файла и обработка *PNG*-изображения в соответствии с передаваемыми флагами и параметрами. Далее обработанное изображение записывается в *PNG*-файл, и программа завершается.

Результаты тестирования см. в приложении А.

Разработанный программный код см. в приложении Б.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана программа, управляемая с помощью аргументов командной строки, считывающая *PNG*-изображение, обрабатывающая изображение (виды обработки изображения: рисование прямоугольника, рисование рамки у изображения, поворот изображения (части) на 90/180/270 градусов) в соответствии с передаваемыми аргументами и записывающая обработанное изображение в *PNG*-файл. В ходе выполнения работы были развиты навыки работы с изображением, библиотеками *libpng* и *getopt*.


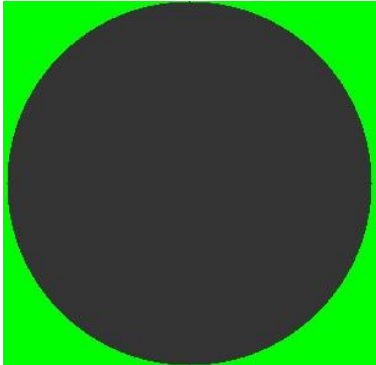


СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Керниган Б., Ритчи Д., Язык программирования Си.: Издательство Москва, Вильямс, 2015 г. 304 с.
2. Мануал по работе с библиотекой libpng // libpng.org. URL: <http://www.libpng.org/pub/png/libpng-1.2.5-manual.html> (дата обращения 12.05.2024).
3. Электронный учебник по программированию на языках Си и С++ // cppstudio. URL: <http://cppstudio.com/> (дата обращения 13.05.2024).
4. Документация libpng // libpng docs URL: <http://www.libpng.org/pub/png/libpng-1.2.5-manual.html> (дата обращения: 8.05.2024).

ПРИЛОЖЕНИЕ А

ТЕСТИРОВАНИЕ

1. Вывод справки (./cw):

№	Ввод	Вывод
1	<pre>./cw —ornament — pattern circle —color 0.255.0 -i input.png -o output.png</pre> 	<p>output.png:</p> 
2	<pre>./cw —rect —left_up 54.21 —right_down 400.321 —color 0.128.128 -i input.png -o output.png —thickness 5</pre> 	<p>output.png:</p> 
3	<pre>./cw —left_up 318.180 —right_down 470.320 —output ./output.png —</pre>	<p>output.png:</p>

angle 90 —input
input.png —rotate



ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: **cw.c**

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

#define RED "\033[1;31m"
#define RESET "\033[0m"
#define GREEN "\e[0;32m"

typedef struct{
    int width, height;
    png_byte color_type;      //{aka int}
    png_byte bit_depth;      //{aka int}
    png_structp png_ptr;     //{aka struct png_struct_def *}
    png_infop info_ptr;      //{aka struct png_info_def *}
    int number_of_passes;
    png_bytep *row_pointers; //{aka unsigned char *}
} Png;

int is_number(char* number);
int check_color_and_coordinates(char* string, int number_dot);
void print_info(Png* image);
void print_help();
void read_png_file(char* file_name, Png* image);
void write_png_file(char* file_name, Png* image);

void set_pixel(Png* image, int x, int y, int color[3]);
void function_rect(Png* image, int left_up[2], int right_down[2],
intthickness, int color[3], int fill, int fill_color[3]);
void draw_ring(Png* image, int color[3], int thickness, int radius, int
x0, int y0);
void draw_circle(Png* image, int color[3], int radius, int x0, int y0);
void function_ornament(Png* image, int pattern, int color[3], int
thickness, int count);
void function_rotate(Png* image, int left_up[2], int right_down[2], int
angle);

int main(int argc, char* argv[]){
    printf("Course work for option 4.15, created by Vladimir
Tukalkin.\n");

    if(argc<2){
        print_help();
        return 0;
    }

    Png img;
    int number_function=-1;
```

```

char input_filename[1000000];
char output_filename[1000000];
int parameters_ornament=0;
int flag_print_info=0;
int flag_left_up=0;
int flag_right_down=0;
int flag_input_filename=0;

int left_up[2]={0,0};
int right_down[2]={0,0};
int thickness=-1;
int color[3]={-1,-1,-1};
int fill=0;
int fill_color[3]={-1,-1,-1};
int pattern=-1;
int count=-1;
int angle=-1;

char* optstring="i:o:fnhrztQ:W:E:R:T:Y:U:I:";
int option_index=0;
struct option long_options[]={
    {"left_up",    required_argument, NULL, 'Q'},
    {"right_down", required_argument, NULL, 'W'},
    {"thickness",  required_argument, NULL, 'E'},
    {"color",      required_argument, NULL, 'R'},
    {"fill_color", required_argument, NULL, 'T'},
    {"pattern",    required_argument, NULL, 'Y'},
    {"count",      required_argument, NULL, 'U'},
    {"angle",      required_argument, NULL, 'I'},
    {"input",      required_argument, NULL, 'i'},
    {"output",     required_argument, NULL, 'o'},
    {"fill",       no_argument, NULL, 'f'},
    {"info",       no_argument, NULL, 'n'},
    {"help",       no_argument, NULL, 'h'},
    {"rect",       no_argument, NULL, 'r'},
    {"ornament",   no_argument, NULL, 'z'},
    {"rotate",     no_argument, NULL, 't'},
    {0,0,0,0}};

int
opt=getopt_long(argc,argv,optstring,long_options,&option_index);

while(opt!=-1){
    switch(opt){
        case 'i':
            strcpy(input_filename,optarg);
            flag_input_filename=1;
            break;
        case 'o':
            strcpy(output_filename,optarg);
            break;
        case 'n':
            flag_print_info=1;
            break;
        case 'f':
            fill=1;
            break;
        case 'r':

```

```

        number_function=1;           //function rect
        break;
    case 'z':
        number_function=2;           //function ornament
        break;
    case 't':
        number_function=3;           //function rotate
        break;
    case 'Q':
        if(!check_color_and_coordinates(optarg,1)){
            printf("%sParameter left_up is not a
number.%s\n", RED, RESET);
            return 41;
        }
        left_up[1]=atoi(strtok(optarg, "."));
        left_up[0]=left_up[1];
        left_up[1]=atoi(strtok(NULL, "."));
        flag_left_up=1;
        break;
    case 'W':
        if(!check_color_and_coordinates(optarg,1)){
            printf("%sParameter right_down is not a
number.%s\n", RED, RESET);
            return 41;
        }
        right_down[1]=atoi(strtok(optarg, "."));
        right_down[0]=right_down[1];
        right_down[1]=atoi(strtok(NULL, "."));
        flag_right_down=1;
        break;
    case 'E':
        if(!is_number(optarg)){
            printf("%sParameter thickness is not a
number.%s\n", RED, RESET);
            return 41;
        }
        thickness=atoi(optarg);
        break;
    case 'R':
        if(!check_color_and_coordinates(optarg,2)){
            printf("%sParameter color is not a
number.%s\n", RED, RESET);
            return 41;
        }
        color[2]=atoi(strtok(optarg, "."));
        color[0]=color[2];
        color[2]=atoi(strtok(NULL, "."));
        color[1]=color[2];
        color[2]=atoi(strtok(NULL, "."));
        if(color[0]>255 || color[1]>255 ||
color[2]>255){
            printf("%sParameter color has [0-
255].%s\n", RED, RESET);
            return 41;
        }
        parameters_ornament+=1;
        break;

```

```

        case 'T':
            if(!check_color_and_coordinates(optarg,2)){
                printf("%sParameter fill_color is not a
number.%s\n",RED,RESET);
                return 41;
            }
            fill_color[2]=atoi(strtok(optarg, "."));
            fill_color[0]=fill_color[2];
            fill_color[2]=atoi(strtok(NULL, "."));
            fill_color[1]=fill_color[2];
            fill_color[2]=atoi(strtok(NULL, "."));
            if(fill_color[0]>255 || fill_color[1]>255 ||
fill_color[2]>255){
                printf("%sParameter fill_color has [0-
255].%s\n",RED,RESET);
                return 41;
            }
            break;
        case 'Y':
            if(strcmp(optarg,"rectangle")==0){
                pattern=1;
            }else if(strcmp(optarg,"circle")==0){
                pattern=2;
            }else if(strcmp(optarg,"semicircles")==0){
                pattern=3;
            }else{
                printf("%sParameter pattern is
incorrect.%s\n",RED,RESET);
            }
            parameters_ornament+=1;
            break;
        case 'U':
            if(!is_number(optarg) || atoi(optarg)<0){
                printf("%sParameter count is
incorrect.%s\n",RED,RESET);
                return 41;
            }
            count=atoi(optarg);
            parameters_ornament+=1;
            break;
        case 'I':
            if(atoi(optarg)==90 || atoi(optarg)==180 ||
atoi(optarg)==270){
                angle=atoi(optarg);
                break;
            }else{
                printf("%sParameter angle is
incorrect.%s\n",RED,RESET);
                return 41;
            }
        case 'h':
            print_help();
            return 0;
        case '?':
            printf("%sArguments entered
incorrectly.%s\n",RED,RESET);
            return 41;

```



```

        default:
            break;
    }

    opt=getopt_long(argc,argv,optstring,long_options,&option_index);
    }

    if(strlen(output_filename)==0){
        printf("%sMissing output filename.\n%s",RED,RESET);
    }

    if(flag_input_filename==0)    strcpy(input_filename,argv[argc-
1]);
    read_png_file(input_filename, &img);

    if(flag_print_info==1){
        print_info(&img);
        return 0;
    }

    switch (number_function){
        case 1:
            if(flag_left_up==1    &&    flag_right_down==1    &&
color[0]!=-1 && thickness!=-1){
                if((fill==1 && fill_color[0]==-1) || (fill==0
&& fill_color[0]!=-1)){
                    printf("%sMissing    parameter    for
fill.%s\n",RED,RESET);
                    return 42;
                }
            }

            function_rect(&img,left_up,right_down,thickness,color,fill,fill_col
or);

            break;
        }else{
            printf("%sThere are not enough parameters for
the function rect.%s\n",RED,RESET);
            return 42;
        }
        case 2:
            if(parameters_ornament>=2){
                if(pattern==2 && color[0]!=-1){

                    function_ornament(&img,pattern,color,0,1);
                    break;
                }
                if(pattern!=2 && pattern!=-1 && color[0]!=-1 &&
thickness!=-1 && count!=-1){

                    function_ornament(&img,pattern,color,thickness,count);
                    break;
                }
                printf("%sThere are not enough parameters for
the function ornament.%s\n",RED,RESET);
                break;
            }else{

```

```

        printf("%sThere are not enough parameters for
the function ornament.%s\n",RED,RESET);
        return 42;
    }
    case 3:
        if(angle!=-1        &&        flag_left_up==1        &&
flag_right_down==1){

            function_rotate(&img,left_up,right_down,angle);
            break;
        }else{
            printf("%sThere are not enough parameters for
the function rotate.%s\n",RED,RESET);
            return 42;
        }
    default:
        printf("%sIncorrectly        function
name.%s\n",RED,RESET);
        return 43;
    }

    if(strlen(output_filename)!=0){
        write_png_file(output_filename,&img);
    }
    printf("%sDONE%s\n",GREEN,RESET);
    return 0;
}

int is_number(char* number){
    for(size_t i=0;i<strlen(number);i++){
        if(!isdigit(number[i])){
            return 0;
        }
    }
    return 1;
}

int check_color_and_coordinates(char* string, int number_dot){
    int count_dot=0;
    int len=0;
    for(size_t i=0;i<strlen(string);i++){
        if(isdigit(string[i])        ||        (string[i]=='-'        &&
number_dot==1)){
            len++;
        }else if(string[i]=='.' && count_dot<number_dot){
            len++;
            count_dot++;
        }
    }
    if(count_dot==2){
        int k=0;
        for(size_t i=0;i<len;i++){
            if(string[i]=='.'){
                k+=1;
            }else{
                k=0;
            }
        }
    }
}

```

```

        if(k==2){
            printf("%sIncorrect          color          or
fill_color.%s\n",RED,RESET);
            exit(41);
        }
    }
    if(len==strlen(string) && count_dot==number_dot) return 1;
    return 0;
}

void print_info(Png* image){
    printf("---Info---\n");
    printf("Width: %d\n", image->width);
    printf("Height: %d\n", image->height);
    printf("Bit Depth: %d\n", image->bit_depth);
    printf("Number of passes: %d\n", image->number_of_passes);
}

void print_help(){
    FILE* file=fopen("help.txt","r");
    if(!file){
        printf("%sFile help.txt not found.%s\n",RED,RESET);
        exit(40);
    }
    char symbol=(char) fgetc(file);
    while(symbol!=EOF){
        printf("%c",symbol);
        symbol=(char) fgetc(file);
    }
    fclose(file);
}

void read_png_file(char* file_name, Png* image){
    unsigned char header[8];
    /* Открыть и проверить, что файл png */
    FILE *fp = fopen(file_name, "rb");
    if(!fp){
        if(strlen(file_name)!=0){
            printf("%sCannot          read
file: %s\n",RED,file_name,RESET);
            exit(40);
        }else{
            printf("%sCannot read file.%s\n",RED,RESET);
            exit(40);
        }
    }
    fread(header, 1, 8, fp);
    if(png_sig_cmp(header, 0, 8)!=0){
        printf("%sProbably,          %s          is          not          a
png.%s\n",RED,file_name,RESET);
        exit(40);
    }

    /* Инициализация структуры PNG */
    image->png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING,
NULL, NULL, NULL);

```

```

        if(!image->png_ptr){
            // Блок исполнится при ошибке в структуре PNG
            printf("%sError in png structure.%s\n",RED,RESET);
            exit(40);
        }
        image->info_ptr = png_create_info_struct(image->png_ptr);
        if(!image->info_ptr){
            printf("%sError in png info-structure.%s\n",RED,RESET);
            exit(40);
        }
        if(setjmp(png_jmpbuf(image->png_ptr))){
            printf("%sError during init_io.%s\n",RED,RESET);
            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);

        /* Проверка типа PNG */
        if(image->color_type==PNG_COLOR_TYPE_RGBA){
            printf("%sThe program does not support working with the
PNG_COLOR_TYPE_RGBA color type.%s\n",RED,RESET);
            exit(40);
        }else if(image->color_type==PNG_COLOR_TYPE_GRAY){
            printf("%sThe program does not support working with the
PNG_COLOR_TYPE_GRAY color type.%s\n",RED,RESET);
            exit(40);
        }else if(image->color_type==PNG_COLOR_TYPE_GRAY_ALPHA){
            printf("%sThe program does not support working with the
PNG_COLOR_TYPE_GRAY_ALPHA color type.%s\n",RED,RESET);
            exit(40);
        }else if(image->color_type==PNG_COLOR_TYPE_PALETTE){
            printf("%sThe program does not support working with the
PNG_COLOR_TYPE_PALETTE color type.%s\n",RED,RESET);
            exit(40);
        }
        png_read_update_info(image->png_ptr, image->info_ptr);

        /* чтение файла */
        image->row_pointers = (png_bytep *) malloc(sizeof(png_bytep)*
image->height);
        for(int y=0;y<image->height;y++){

            image->row_pointers[y]=(png_bytep)malloc(png_get_rowbytes(image->pn
g_ptr,image->info_ptr));
        }
        png_read_image(image->png_ptr, image->row_pointers);

```

```

        fclose(fp);
    }

    void write_png_file(char* file_name, Png* image){
        /* создание файла */
        FILE* fp=fopen(file_name, "wb");
        if(!fp){
            printf("%sError in write_png_file function: file could not
opened.%s\n", RED, RESET);
            exit(40);
        }

        /* Инициализация структуры */
        image->png_ptr=png_create_write_struct(PNG_LIBPNG_VER_STRING,N
ULL,NULL,NULL);
        if(!image->png_ptr){
            printf("%sError in write_png_file function:
png_create_write_struct failed.%s\n", RED, RESET);
            exit(40);
        }
        image->info_ptr=png_create_info_struct(image->png_ptr);
        if(!image->info_ptr){
            printf("%sError in write_png_file function:
png_create_info_struct failed.%s\n", RED, RESET);
            exit(40);
        }
        if(setjmp(png_jmpbuf(image->png_ptr))){
            printf("%sError in write_png_file function: error during
init_io.%s\n", RED, RESET);
            exit(40);
        }
        png_init_io(image->png_ptr, fp);

        /* Запись заголовка */
        if(setjmp(png_jmpbuf(image->png_ptr))){
            printf("%sError in write_png_file function: error during
writing header.%s\n", RED, RESET);
            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))){
            printf("%sError in write_png_file function: error during
writing bytes.%s\n", RED, RESET);
            exit(40);
        }
        png_write_image(image->png_ptr, image->row_pointers);

        /* Конец записи */
        if(setjmp(png_jmpbuf(image->png_ptr))){
            printf("%sError in write_png_file function: error during
writing end of file.%s\n", RED, RESET);
            exit(40);
        }
    }

```

```

        png_write_end(image->png_ptr, NULL);

        /* Очистка памяти */
        for(int y = 0; y < image->height; y++)
            free(image->row_pointers[y]);
        free(image->row_pointers);
        fclose(fp);
    }

    void set_pixel(Png* image, int x, int y, int color[3]){
        if(x>=0 && y>=0 && x<image->width && y<image->height){
            image->row_pointers[y][x*3+0]=color[0]; //Red
            image->row_pointers[y][x*3+1]=color[1]; //Green
            image->row_pointers[y][x*3+2]=color[2]; //Blue
        }
    }

    void function_rect(Png* image, int left_up[2], int right_down[2], int
thickness, int color[3], int fill, int fill_color[3]){
        /* Checking the correctness of coordinates */
        int k;
        if(left_up[0]>right_down[0]){
            k=right_down[0];
            right_down[0]=left_up[0];
            left_up[0]=k;
        }
        if(left_up[1]>right_down[1]){
            k=right_down[1];
            right_down[1]=left_up[1];
            left_up[1]=k;
        }
        /* Draw up */
        for(int y=left_up[1]-
thickness/2;y<left_up[1]+thickness/2;y++){
            for(int x=left_up[0];x<right_down[0];x++){
                set_pixel(image,x,y,color);
            }
        }
        /* Draw left */
        for(int y=left_up[1];y<right_down[1];y++){
            for(int x=left_up[0]-
thickness/2;x<left_up[0]+thickness/2;x++){
                set_pixel(image,x,y,color);
            }
        }
        /* Draw down */
        for(int y=right_down[1]-
thickness/2;y<right_down[1]+thickness/2;y++){
            for(int x=left_up[0];x<right_down[0];x++){
                set_pixel(image,x,y,color);
            }
        }
        /* Draw right */
        for(int y=left_up[1];y<right_down[1];y++){
            for(int x=right_down[0]-
thickness/2;x<right_down[0]+thickness/2;x++){
                set_pixel(image,x,y,color);
            }
        }
    }

```

```

        }
    }
    /* Fill */
    if(fill==1){
        for(int y=left_up[1]+thickness/2;y<right_down[1]-
thickness/2;y++){
            for(int x=left_up[0]+thickness/2;x<right_down[0]-
thickness/2;x++){
                set_pixel(image,x,y,fill_color);
            }
        }
    }
}

void draw_ring(Png* image, int color[3], int thickness, int radius,
int x0, int y0){
    thickness=thickness/2;
    for(int y=y0-radius-2*thickness;y<y0+radius+2*thickness;y++){
        for(int x=x0-radius-
2*thickness;x<x0+radius+2*thickness;x++){
            if(((radius)*(radius)-2*thickness*radius-
2)<((x-x0)*(x-x0)+(y-y0)*(y-y0)) &&
((thickness+radius)*(thickness+radius))>((x-x0)*(x-x0)+(y-y0)*(y-y0))) {
                set_pixel(image,x,y,color);
            }
        }
    }
}

void draw_circle(Png* image, int color[3], int radius, int x0, int
y0){
    for(int y=y0-radius;y<y0+radius;y++){
        for(int x=x0-radius;x<x0+radius;x++){
            if(((radius)*(radius))>((x-x0)*(x-x0)+(y-
y0)*(y-y0))) {
                set_pixel(image,x,y,color);
            }
        }
    }
}

void function_ornament(Png* image, int pattern, int color[3], int
thickness, int count){
    int width=image->width/count/2+1;
    int height=image->height/count/2+1;
    int radius=image->width/2;
    if(radius>image->height/2) radius=image->height/2;
    switch(pattern){
        case 1:
            if(thickness>image->width)
thickness=image->width/2;
            if(thickness>image->height)
thickness=image->height/2;
            if(thickness*4*count>image->width)
count=1+image->width/thickness/4;
            if(thickness*4*count>image->height)
count=1+image->height/thickness/4;

```

```

        for(size_t i=0;i<count;i++){
            int
left_up[2]={0+i*2*thickness,0+i*2*thickness};
            int right_down[2]={image->width-
i*2*thickness,image->height-i*2*thickness};
            /* Draw up */
            for(int
y=left_up[1];y<left_up[1]+thickness;y++){
                for(int
x=left_up[0];x<right_down[0];x++){
                    set_pixel(image,x,y,color);
                }
            }
            /* Draw left */
            for(int y=left_up[1];y<right_down[1];y++){
                for(int
x=left_up[0];x<left_up[0]+thickness;x++){
                    set_pixel(image,x,y,color);
                }
            }
            /* Draw down */
            for(int y=right_down[1]-
thickness;y<right_down[1];y++){
                for(int
x=left_up[0];x<right_down[0];x++){
                    set_pixel(image,x,y,color);
                }
            }
            /* Draw right */
            for(int y=left_up[1];y<right_down[1];y++){
                for(int x=right_down[0]-
thickness;x<right_down[0];x++){
                    set_pixel(image,x,y,color);
                }
            }
        }
        break;
    case 2:
        for(int y=image->height/2;y<image->height;y++){
            for(int x=image->width/2;x<image->width;x++){
                if((radius*radius)<((x-
image->width/2)*(x-image->width/2)+(y-image->height/2)*(y-
image->height/2))){
                    set_pixel(image,x,y,color); //right
down
                    set_pixel(image,image->width-x-
1,image->height-y-1,color); //left up
                    set_pixel(image,image->width-x-
1,y,color); //left_down
                    set_pixel(image,x,image->height-y-
1,color); //right_up
                }
            }
        }
        break;
    case 3:
        for(size_t i=0;i<count;i++){

```



```

        draw_ring(image,color,thickness+1,width,width+i*2*width,0);

        draw_ring(image,color,thickness+1,width,width+i*2*width,image->height);
    }
    for(size_t i=0;i<count;i++){

        draw_ring(image,color,thickness+1,height,0,height+2*i*height);

        draw_ring(image,color,thickness+1,height,image->width,height+2*i*height);
    }
    break;
default:
    printf("КАК ТЫ СЮДА ПОПАЛ?!\n");
    exit(41);
}
}

void function_rotate(Png* image, int left_up[2], int right_down[2],
int angle){
    if((left_up[0]>right_down[0]) || (left_up[1]>right_down[1])){
        int k1=left_up[0], k2=left_up[1];
        left_up[0]=right_down[0];
        left_up[1]=right_down[1];
        right_down[0]=k1;
        right_down[1]=k2;
    }
    if(right_down[0]>image->width || right_down[1]>image->height){
        printf("%sInvalid size image.%s\n",RED,RESET);
        exit(41);
    }
    int arr[right_down[1]+1][(right_down[0]+1)*3];
    for(int y=0;y<right_down[1];y++){
        for(int x=0;x<right_down[0];x++){
            if(x>=0 && y>=0 && x<right_down[0] &&
y<right_down[1]){
                arr[y][x*3+0]=image->row_pointers[y][(x)*3+0];
                arr[y][x*3+1]=image->row_pointers[y][(x)*3+1];
                arr[y][x*3+2]=image->row_pointers[y][(x)*3+2];
            }
        }
    }

    int x0=(right_down[0]-left_up[0])/2+left_up[0];
    if((right_down[0]-left_up[0])%2!=0) x0-=1;
    int y0=(right_down[1]-left_up[1])/2+left_up[1];
    //if((right_down[1]-left_up[1])%2==0) y0-=1;
    int x1,y1;
    switch(angle){
        case 90:
            x1=x0 - (right_down[1]-left_up[1])/2;
            y1=y0 + (right_down[0]-left_up[0])/2;
            if((right_down[1]-left_up[1])%2==0) y1-=1;
            for(int y=left_up[1];y<right_down[1];y++){
                for(int x=left_up[0];x<right_down[0];x++){

```

```

        int xx=x1+y-left_up[1];
        int yy=y1-x+left_up[0];
        if(x>0 && y>0){
            int
color1[3]={arr[y][x*3+0],arr[y][x*3+1],arr[y][x*3+2]};
            set_pixel(image,xx,yy,color1);
        }else{
            int color[3]={0,0,0};
            set_pixel(image,xx,yy,color);
        }
    }
    }
    break;
case 180:
    for(int y=left_up[1];y<right_down[1];y++){
        for(int x=left_up[0];x<right_down[0];x++){
            int xx=right_down[0]+left_up[0]-x-1;
            int yy=right_down[1]+left_up[1]-y-1;
            int color1[3]={0,0,0};
            if(x>=0 && y>=0 && xx>0 && yy>0 &&
xx>=left_up[0] && xx<right_down[0] && yy>=left_up[1] && yy<right_down[1]){
                color1[0]=arr[y][x*3+0];
                color1[1]=arr[y][x*3+1];
                color1[2]=arr[y][x*3+2];
            }
            set_pixel(image,xx,yy,color1);
        }
    }
    break;
case 270:
    x1=x0+(right_down[1]-left_up[1])/2;
    y1=y0-(right_down[0]-left_up[0])/2;
    for(int y=left_up[1]+1;y<right_down[1];y++){
        for(int x=left_up[0];x<right_down[0];x++){
            int xx=x1-y+left_up[1];
            int yy=y1+x-left_up[0];
            if(x>0 && y>0){
                int
color1[3]={arr[y][x*3+0],arr[y][x*3+1],arr[y][x*3+2]};
                set_pixel(image,xx,yy,color1);
            }else{
                int color[3]={0,0,0};
                set_pixel(image,xx,yy,color);
            }
        }
    }
    break;
}
}
}

```

Файл: **Makefile**

all: cw

cw:

gcc Tukalkin_Vladimir_cw.c -lpng -lm -o cw

Файл: **help.txt**

---СПРАВКА---

Программа для обработки изображений в формате PNG.

Функционал:

- 1) Рисование прямоугольника.
- 2) Сделать рамку в виде узора.
- 3) Поворот изображения (части) на 90/180/270 градусов.

Флаги:

--help (-h): вывести справку.

--info (-n): вывести информацию об изображении.

--rect (-r) {--left_up, --right_down, --thickness, --color, --fill, --fill_color}: рисование прямоугольника.

--ornament (-z) {--pattern, --color, --thickness, --count}: сделать рамку в виде узора.

--rotate (-t) {--left_up, --right_down, --angle}: поворот изображения (части) на 90/180/270 градусов.

--input (-i) {filename}: задать имя входного изображения. Если флаг отсутствует, то предполагается, что имя входного изображения передаётся последним аргументом.

--output (-o) {filename}: задать имя выходного изображения. Если флаг отсутствует, то предполагается, что имя выходного изображения является аналогичным входному.

--fill (-f): залить прямоугольник цветом флага --fill_color. Работает как бинарное значение: флага нет - false, флаг есть - true.

--left_up (-Q) {x.y}: задать координаты левого верхнего угла. Значение задаётся в формате `left.up`, где left - координата по x, up - координата по y.

--right_down (-W) {x.y}: задать координаты правого нижнего угла. Значение задаётся в формате `right.down`, где right - координата по x, down - координата по y.

--thickness (-E) {N}: задать толщину линии. На вход принимает число больше 0.

--color (-R) {r.g.b}: задать цвет линии. Цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb - числа, задающие цветовую компоненту.

--fill_color (-T) {r.g.b}: задать цвет заливки. Цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb - числа, задающие цветовую компоненту.

--pattern (-Y) {pattern}: задать узор рамки. Имеет значения: rectangle и circle, semicircles.

`--count (-U) {N}`: задать количество для узора рамки. На вход принимает число больше 0.

`--angle (-I) {N}`: задать угол поворота. Возможные значения: 90, 180, 270.

Пример корректной передачи аргументов:

```
./a.out -h

./a.out --info --input input_filename.png

./a.out --rect --left_up 10.20 --right_down 100.20 --thickness 5 --
color 255.0.0 --input input_filename.png --output output_filename.png

./a.out --rect --left_up 10.20 --right_down 100.20 --thickness 5 --
color 255.0.0 --input input_filename.png --output output_filename.png --
fill --fill_color 0.255.0

./a.out --ornament --pattern circle --color 127.255.0 -i
input_filename.png -o output_filename.png

./a.out --ornament --semicircles --color 0.255.0 --thickness 10 --
count 7 --input input_filename.png -o output_filename.png

./a.out --rotate --left_up 0.0 --right_down 100.100 --angle 180 -i
input_filename.png -o output_filename.png
```