**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 *PNG*-файла | 8 |
| 2.2 | Дополнительные, вспомогательные функции | 9 |
| 2.3 | Основные функции | 15 |
| 2.4 | Функция *main* | 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 структурy *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 следующую 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`)

1. 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

1. 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 выведение символов в поток вывода; после выведения 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 стоке и с помощью *if* и функции *isdigit* проверяет каждый элемент, для 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 заданному координатами пикселю 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:ь 8.05.2024).

**приложение А**

**тестированиЕ**

1. Вывод справки (./cw):

|  |  |  |
| --- | --- | --- |
| № | Ввод | Вывод |
| 1 | ./cw —ornament —pattern circle —color 0.255.0 -i input.png -o output.png  https://sun1-22.userapi.com/impg/KjPSNG2BDCNKbBtgNIacSjJ-Dor46iTSDOioGg/yVFDJjEGwSk.jpg?size=305x299&quality=96&sign=32af1b2ee7c4e1f39790259a3d7c8bcf&type=album | output.png:  https://sun9-3.userapi.com/impg/vKPQtY9yUf4qGVb917Q9B45JcP5ofCJtzfcTww/J8dIfyb3SQk.jpg?size=305x299&quality=96&sign=b7e81709d0c0b1ba8b97748e567df34e&type=album |
| 2 | ./cw —rect —left\_up 54.21 —right\_down 400.321 —color 0.128.128 -i input.png -o output.png —thickness 5  https://sun9-78.userapi.com/impg/h-F-hG75q0rOmND4PFil9yp4s10KODSW0ODdBw/YhwGZADbUJs.jpg?size=496x500&quality=96&sign=fbb83a61f256b2b2af7e64e23716e935&type=album | output.png:  https://sun9-25.userapi.com/impg/QZL06qgTRsYrDcJ2t6xOBT2qIDovwU8tDG_cVw/p-k8tOExQQU.jpg?size=496x500&quality=96&sign=bf82adaff5f81d240f6d02f68768503d&type=album |
| 3 | ./cw —left\_up 318.180 —right\_down 470.320 —output ./output.png —angle 90 —input input.png —rotate  https://sun9-78.userapi.com/impg/h-F-hG75q0rOmND4PFil9yp4s10KODSW0ODdBw/YhwGZADbUJs.jpg?size=496x500&quality=96&sign=fbb83a61f256b2b2af7e64e23716e935&type=album | output.png:  https://sun9-56.userapi.com/impg/W8TMTRtsO81PiUOOHAWWD6SSsbDHkVvqzzsLIQ/SwAa7Ki-EBE.jpg?size=496x500&quality=96&sign=dad10b346cd7f7c2f50ee78cb98e0e63&type=album |

**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\_color);

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%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\_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, 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,NULL,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]){

/\* Сhecking 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 угл 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