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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка PNG файла

Студент гр. 3341 _____ Пчелкин Н.И.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Пчелкин Н.И.

Группа 3341

Вариант 19

Программа обязательно должна иметь CLI (опционально дополнительное использование GUI). Более подробно тут:
http://se.moevm.info/doku.php/courses:programming:rules_extra_kurs

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 треугольника. Флаг для выполнения данной операции: `--triangle`. Треугольник определяется
Координатами его вершин. Флаг `--points`, значение задаётся в формате `x1.y1.x2.y2.x3.y3` (точки будут (x1; y1), (x2; y2) и (x3; y3)), где x1/x2/x3 – координаты по x, y1/y2/y3 – координаты 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)
Треугольник 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) Находит самый большой прямоугольник заданного цвета и перекрашивает его в другой цвет. Флаг для выполнения данной операции: `--biggest_rect`. Функционал определяется:
Цветом, прямоугольник которого надо найти. Флаг `--old_color` (цвет задаётся строкой `rrr.ggg.bbb`, где rrr/ggg/bbb – числа, задающие цветовую компоненту. пример `--old_color 255.0.0` задаёт красный цвет)
Цветом, в который надо его перекрасить. Флаг `--new_color` (работает аналогично флагу `--old_color`)
(3) Создать коллаж размера N*M из одного изображения. Флаг для выполнения данной операции: `--collage`. Коллаж представляет собой это же самое изображение повторяющееся N*M раз.
Количество изображений по “оси” Y. Флаг `--number_y`. На вход принимает число больше 0
Количество изображений по “оси” X. Флаг `--number_x`. На вход принимает число больше.
Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции.

Содержание пояснительной записки:
разделы «Аннотация», «Содержание», «Введение», «Ход работы», «Пример работы программы», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 18.03.2024

Дата сдачи реферата: 13.05.2024

Дата защиты реферата: 15.05.2024

Студент		Пчелкин Н.И.
Преподаватель		Глазунов С.А.

АННОТАЦИЯ

В данной курсовой работе была реализована программа, обрабатывающая PNG изображения, не имеющие сжатия. Программа проверяет тип изображения, его версию, при соответствии требованиям в дальнейшем обрабатывает его и подаёт на выход изменённую копию изображения. Взаимодействие с программой осуществляется с помощью CLI (интерфейс командной строки).

SUMMARY

In this course has been created a program that processes uncompressed PNG images. The program checks the type of image, its version, if it meets the requirements, it further processes it and outputs a modified copy of the image. Interaction with the program is performed using CLI (command line interface).

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является создание программы на языке Си для обработки PNG изображений.

Для достижения поставленной цели потребовалось решить ряд задач:

- изучить, как устроены PNG файлы, что они в себе содержат;

- научиться распознавать PNG файлы среди прочих и проверять их прочие характеристики;

- научиться считывать и записывать PNG изображения;

- разработать функцию рисования треугольника на изображении, его заливки;

- разработать функцию поиска наибольшего прямоугольника заданного цвета на изображении и его перекрашивания;

- разработать функцию создания коллажа из исходного изображения по заданным количествам повторений изображения по оси x и по оси y;

- изучить библиотеку *getopt.h*;

- научиться работать с аргументами командной строки, длинными и короткими флагами;

- создать *Makefile* для сборки программы;

- протестировать разработанную программу.

1. РАБОТА С ФАЙЛАМИ

Работа с файлом происходит при помощи библиотеки *png.h*. Функции по считыванию файла, проверки его и заполнения соответствующей структуры, а также функция по созданию PNG файла и записыванию в него полученную структуру описаны в мануале *pnglib*.

2. ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ

Ввод аргументов в программу происходит по средству флагов. Считать их можно благодаря библиотеке getopt.h, в которой описывается функция getopt_long(). Получаемые значения проходят сквозь ветку switch-case, записывая необходимые аргументы в соответствующие переменные и запуская необходимые для обработки изображения функции.

Валидность введенных аргументов происходит с помощью библиотеки regex.h. При введении аргумента ему сопоставляется соответствующая маска. При совпадении – аргумент вводится корректно, при несовпадении – выдается ошибка чтения аргумента.

3. ОСНОВНЫЕ ФУНКЦИИ

3.1 Вспомогательные функции

get_points() и get_color() обеспечивают ввод точек и цвета соответственно.

point_is_in() производит проверку, входит ли какая-то конкретная точка в изображение или нет. Если входит, то возвращает 1, в противном случае – 0.

set_pixel() проверяет, входит ли точка в изображение и если входит, то задает поданный в функцию цвет конкретному пикселю по его координатам.

draw_thin_line() рисует отрезок толщины один по методу Брезенхема.

draw_line() рисует линию произвольной толщины(в основе функции лежит предыдущая функция).

create_new_image() создает новое пустое изображение произвольных размеров.

is_in_triangle() проверяет, входит ли конкретная точка в заданный координатами точек треугольник (в основе лежит исчесление и анализ векторного произведения).

check_color() проверяет на соответствие цвет пикселя и заданный цвет. Если соответствует, то возвращает 1, в противном случае – 0.

find_biggest_rect() находит самый большой прямоугольник заданного цвета и возвращает координаты его верхнего левого и нижнего правого углов.

3.2 Рисование треугольника

Рисование треугольника осуществляется с помощью функции *draw_triangle(),* которая принимает на вход координаты вершин треугольника, ширину контура, структуру изображения, цвет контура, а также флаг fill, отвечающий за необходимость заливки фигуры и цвет заливки. Для рисования треугольника функция последовательно вызывает функцию *draw_line()* для рисования трёх сторон фигуры по заданным точкам. Если была необходима заливка, то программа красит в необходимый цвет каждый пиксель, входящий в

заданный треугольник. Проверка осуществляется при помощи функции is_in_triangle().

3.3 Поиск самого большого прямоугольника заданного цвета и перекрашивание его в другой цвет

За эту опцию отвечает функция *recolor_biggest_rect()*, на вход которой требуется структура изображения, цвет искомого прямоугольника и цвет, в который его необходимо перекрасить. Функция ищет самый большой прямоугольник, с помощью функции find_biggest_rect(), а затем перекрашивает все пиксели этого прямоугольника в новый цвет.

3.4. Создание коллажа из исходного изображения

За создание коллажа отвечает функция *make_collage()*, которая принимает на вход количество повторений исходного изображения по осям x и y, а также структуру изображения. Функция выделяет память под новое изображение большего размера, затем копирует пиксели из исходного изображения в новое в соответствии с заданными параметрами повторения, а затем присваивает указатель новой структуры старой.

3.5 Вызов помощи

За вызов справки о функционале программы отвечает функция print_help(), которая выводит текст справки на экран.

3.6 Отображение информации об изображении

За это в программе отвечает функция info(), которая последовательно выводит каждый параметр структуры изображения, кроме массива пикселей.

3.7 Функции ошибок

raise_error() – печатает текст ошибки и предлагает ознакомиться со справкой о функционале программы. Функция также останавливает программу с кодом ошибки.

raise_reading_error() создана для обработки ошибок во время считывания и записывания файла, работает аналогично предыдущей функции.

ЗАКЛЮЧЕНИЕ

Разработана программа на языке программирования Си, обрабатывающая PNG изображения и имеющая CLI. В ходе выполнения работы было изучено устройство PNG файлов; изучены методы считывание и записи файлов; получены навыки обработки изображений; разработаны функции для рисования треугольника и его заливки; поиска наибольшего прямоугольника заданного цвета и его перекрашивания; создания коллажа из исходного изображения; изучена библиотека *getopt.h;* изучена работа с аргументами командной строки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1.https://ru.wikipedia.org/wiki/ PNG - структура файла PNG;

2.https://se.moevm.info/lib/exe/fetch.php/courses:programming:programming_cw_metoda_2nd_course_last_ver.pdf.pdf - методические материалы для написания курсовой работы

3.https://www.r5.org/files/books/computers/languages/c/kr/Brian_Kernighan_Dennis_Ritchie-The_C_Programming_Language-RU.pdf – язык программирования Си

4. https://habr.com/ru/articles/55665/ - принцип работы getopt_long

5. https://ru.wikibooks.org/wiki/Реализации_алгоритмов/Алгоритм_Брезенхэма - алгоритм Брезенхэма

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <stdio.h>
#include <getopt.h>

#include "image_processing.h"
#include "file_manager.h"

int main(int argc, char* argv[]){
    printf("Course work for option 4.19, created by Nikita Pchelkin.\n");

    opterr = 0;
    Png image;
    char* input_file = strdup(find_input_file(argc, argv));
    char* output_file = calloc(MAX_STRING, sizeof(char));

    read_png_file(input_file, &image);
    image_processing(argc, argv, &image, &output_file, input_file);
    write_png_file(output_file, &image);

    return 0;
}
```

Название файла: struct.h

```
#pragma once
#include <png.h>

typedef struct {
    int width, height;
    png_byte color_type;
    png_byte bit_depth;

    png_structp png_ptr;
    png_infop info_ptr;
    png_bytep *row_pointers;
    png_byte channels;
} Png;
```

Название файла: file_manager.h

```
#pragma once

#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>

#include "struct.h"
#include "help.h"

#define PNG_DEBUG 3

char* find_input_file(int argc, char* argv[]);
void read_png_file(char* file_name, Png* image);
void write_png_file(char *file_name, Png* image);
```

Название файла: file_manager.c

```
#include "file_manager.h"

char* find_input_file(int argc, char* argv[]){
    optind = 0;
    int opt;
    int option_index = 0;
    struct option long_option[] = {
        {"input", 1, NULL, 'i'},
        {"help", 0, NULL, 'h'},
        {0, 0, 0, 0}
    };

    while((opt = getopt_long(argc, argv, "-hi:", long_option, &option_index)) !=
-1){
        if(opt == 'i')
            return optarg;
        if(opt == 'h'){
            print_help();
            exit(0);
        }
    }

    return argv[argc-1];
}

void read_png_file(char* file_name, Png* image) {

    int y;
    char header[8];

    FILE *fp = fopen(file_name, "rb");
    if (!fp){
       raise_reading_error("file could not be opened");
    }

    fread(header, 1, 8, fp);
    if (png_sig_cmp(header, 0, 8)){
       raise_reading_error("file is not recognized as a PNG");
    }

    image->png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL, NULL,
NULL);

    if (!image->png_ptr){
       raise_reading_error("png_create_read_struct failed");
    }

    image->info_ptr = png_create_info_struct(image->png_ptr);
    if (!image->info_ptr){
        raise_reading_error("png_create_info_struct failed");
    }

    if (setjmp(png_jmpbuf(image->png_ptr))){
        raise_reading_error("error during init_io");
    }

    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->channels = png_get_channels(image->png_ptr, image->info_ptr);

    png_read_update_info(image->png_ptr, image->info_ptr);

    if (setjmp(png_jmpbuf(image->png_ptr))){
       raise_reading_error("error during read_image");
    }
    if (png_get_color_type(image->png_ptr, image->info_ptr) !=
PNG_COLOR_TYPE_RGB && png_get_color_type(image->png_ptr, image->info_ptr) !=
PNG_COLOR_TYPE_RGBA){
        raise_reading_error("Can't support this color type");
    }
    image->row_pointers = (png_bytep *) malloc(sizeof(png_bytep) * image-
>height);
    for (y = 0; y < image->height; y++)
        image->row_pointers[y] = (png_byte *) malloc(png_get_rowbytes(image-
>png_ptr, image->info_ptr));

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

    fclose(fp);
}

void write_png_file(char *file_name, Png* image) {
    int y;
    FILE *fp = fopen(file_name, "wb");
    if (!fp){
        raise_reading_error("file could not be opened");
    }

    image->png_ptr = png_create_write_struct(PNG_LIBPNG_VER_STRING, NULL, NULL,
NULL);

    if (!image->png_ptr){
        raise_reading_error("png_create_write_struct failed");
    }

    image->info_ptr = png_create_info_struct(image->png_ptr);
    if (!image->info_ptr){
       raise_reading_error("png_create_info_struct failed");
    }

    if (setjmp(png_jmpbuf(image->png_ptr))){
        raise_reading_error("error during init_io");
    }

    png_init_io(image->png_ptr, fp);

    if (setjmp(png_jmpbuf(image->png_ptr))){
        raise_reading_error("error during writing header");
    }

    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))){
```

```
        raise_reading_error("error during writing bytes");
    }

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

    if (setjmp(png_jmpbuf(image->png_ptr))){
        raise_reading_error("error during end of write");
    }

    png_write_end(image->png_ptr, NULL);

    for (y = 0; y < image->height; y++)
        free(image->row_pointers[y]);
    free(image->row_pointers);

    fclose(fp);
}
```

Название файла: image_processing.h

```
#pragma once

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

#include "help.h"
#include "struct.h"

#define BLANK_STRING "\0"
#define MAX_STRING 256

void get_points(char* string, int* x1, int* y1, int* x2, int* y2, int* x3, int*
y3, regmatch_t groupArray[]);
int get_color(char* string, int buffer[3], regmatch_t groupArray[]);
int point_is_in(Png* image, int x, int y);
void set_pixel(Png* image, int x, int y, int color[3]);
void draw_thin_line(Png* image, int x1, int y1, int x2, int y2, int color[3]);
void draw_line(Png* image, int x1, int y1, int x2, int y2, int color[3], int
thickness);
Png* create_new_image(Png* image, int width, int height);
int is_in_triangle(int x0, int y0, int x1, int y1, int x2, int y2, int x3, int
y3);
void draw_triangle(Png* image, int x1, int y1, int x2, int y2, int x3, int y3,
int thickness, int color[3], int fill, int fill_color[3]);
int check_color(Png* image, int x, int y, int color[3]);
int* find_biggest_rect(Png* image, int color[3]);
void recolor_biggest_rect(Png* image, int old_color[3], int new_color[3]);
void make_collage(Png* image, int number_x, int number_y);
void image_processing(int argc, char* argv[], Png* image, char** output_file,
char* input_file);
```

Название файла: image_processing.c

```
#include "image_processing.h"

void get_points(char* string, int* x1, int* y1, int* x2, int* y2, int* x3, int*
y3, regmatch_t groupArray[]){
    char* number;
    int group_number = 1;
```

```
    int buffer[3][2];
    for(int i = 0; i < 3; i++)
        for(int j = 0; j < 2; j++){
            number = calloc(MAX_STRING, sizeof(char));
            for(int k = groupArray[group_number].rm_so; k <
groupArray[group_number].rm_eo; k++)
                number[k - groupArray[group_number].rm_so] = string[k];
            buffer[i][j] = atoi(number);
            group_number++;
        }
    *x1 = buffer[0][0];
    *y1 = buffer[0][1];
    *x2 = buffer[1][0];
    *y2 = buffer[1][1];
    *x3 = buffer[2][0];
    *y3 = buffer[2][1];
}

int get_color(char* string, int buffer[3], regmatch_t groupArray[]){
    char* number;
    int index;
    for(int i = 0; i < 3; i++){
        number = calloc(MAX_STRING, sizeof(char));
        index = 0;
        for(int k = groupArray[i+1].rm_so; k < groupArray[i+1].rm_eo; k++)
            number[index++] = string[k];
        buffer[i] = atoi(number);
        if(buffer[i] < 0 || buffer[i] > 255){
            return 1;
        }
    }
    return 0;
}

int point_is_in(Png* image, int x, int y){
    if(x >= 0 && x < image->width && y >= 0 && y < image->height)
        return 1;
    return 0;
}

void set_pixel(Png* image, int x, int y, int color[3]){
    if(point_is_in(image, x, y)){
        image->row_pointers[y][x * image->channels + 0] = color[0];
        image->row_pointers[y][x * image->channels + 1] = color[1];
        image->row_pointers[y][x * image->channels + 2] = color[2];
        if(image->channels == 4)
            image->row_pointers[y][x * image->channels + 3] = 255;
    }
}

void draw_thin_line(Png* image, int x1, int y1, int x2, int y2, int color[3]){
    int sign_y = y2 < y1 ? -1 : 1;
    int sign_x = x2 < x1  ? -1 : 1;
    int dy = abs(y2 - y1);
    int dx = abs(x2 - x1);

    set_pixel(image, x2, y2, color);
    int delta = dx - dy;
    while(x1*sign_x <= x2*sign_x || y1*sign_y <= y2*sign_y){
        set_pixel(image, x1, y1, color);
        if(2 * delta + dy > 0){
            delta -= dy;
```

```
            x1 += sign_x;
        }
        if(2 * delta - dx < 0){
            delta += dx;
            y1 += sign_y;
        }
    }
}

void draw_line(Png* image, int x1, int y1, int x2, int y2, int color[3], int
thickness){
    draw_thin_line(image, x1, y1, x2, y2, color);
    for(int i = 1; i <= (thickness - 1) / 2; i++){
        if(abs(x1-x2) >= abs(y1-y2)){
            draw_thin_line(image, x1, y1 + i, x2, y2 + i, color);
            draw_thin_line(image, x1, y1 - i, x2, y2 - i, color);
        } else{
            draw_thin_line(image, x1 + i, y1, x2 + i, y2, color);
            draw_thin_line(image, x1 - i, y1, x2 - i, y2, color);
        }
    }
    if(thickness % 2 == 0){
        if(abs(x1-x2) >= abs(y1-y2)){
            draw_thin_line(image, x1, y1 + 1, x2, y2 + 1, color);
        } else{
            draw_thin_line(image, x1 + 1, y1, x2 + 1, y2, color);
        }
    }
}

Png* create_new_image(Png* image, int width, int height){
    Png* new_image = malloc(sizeof(Png));

    new_image->png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL,
NULL, NULL);
    if (!new_image->png_ptr) {
        printf("Error in make_empty_image function: png_create_read_struct
failed.\n");
        exit(20);
    }
    new_image->info_ptr = png_create_info_struct(new_image->png_ptr);

    new_image->width = width;
    new_image->height = height;
    new_image->color_type = image->color_type;
    new_image->bit_depth = image->bit_depth;
    new_image->channels = image->channels;

    png_read_update_info(new_image->png_ptr, new_image->info_ptr);

    new_image->row_pointers = (png_bytep*)malloc(sizeof(png_bytep) * new_image-
>height);
    for (int y = 0; y < new_image->height; y++)
        new_image->row_pointers[y] = (png_byte*)malloc(png_get_rowbytes(image-
>png_ptr, image->info_ptr) / image->width * new_image->width);

    return new_image;
}

int is_in_triangle(int x0, int y0, int x1, int y1, int x2, int y2, int x3, int
y3){
```

```
    if(((x1-x0)*(y2-y1)-(x2-x1)*(y1-y0))*((x2-x0)*(y3-y2)-(x3-x2)*(y2-y0))*((x3-
x0)*(y1-y3)-(x1-x3)*(y3-y0)) == 0)
        return 0;
    int sign_1 = (x1-x0)*(y2-y1)-(x2-x1)*(y1-y0) > 0 ? 1 : -1;
    int sign_2 = (x2-x0)*(y3-y2)-(x3-x2)*(y2-y0) > 0 ? 1 : -1;
    int sign_3 = (x3-x0)*(y1-y3)-(x1-x3)*(y3-y0) > 0 ? 1 : -1;
    if(sign_1 == sign_2 && sign_2 == sign_3)
        return 1;
    else
        return -1;
}

void draw_triangle(Png* image, int x1, int y1, int x2, int y2, int x3, int y3,
int thickness, int color[3], int fill, int fill_color[3]){
    if(fill)
        for(int i = 0; i < image->height; i++)
            for(int j = 0; j < image->width; j++)
                if(is_in_triangle(j, i, x1, y1, x2, y2, x3, y3) == 1)
                    set_pixel(image, j, i, fill_color);

    draw_line(image, x1, y1, x3, y3, color, thickness);
    draw_line(image, x1, y1, x2, y2, color, thickness);
    draw_line(image, x3, y3, x2, y2, color, thickness);
}

int check_color(Png* image, int x, int y, int color[3]){
    if(!point_is_in(image, x, y))
        return 0;
    for(int i = 0; i < 3; i++)
        if(image->row_pointers[y][x*image->channels + i] != color[i])
            return 0;
    return 1;
}

int* find_biggest_rect(Png* image, int color[3]){
    int max_area = 0;
    int current_area, max_current_area;
    int a, b;
    int x1, y1, x2, y2;
    int a_max, b_max;

    for(int x = 0; x < image->width; x++)
        for(int y = 0; y < image->height; y++)
            if(check_color(image, x, y, color) && (!check_color(image, x-1, y,
color) || !check_color(image, x, y-1, color))){
                int b_current_max = image->height;
                max_current_area = 0;
                a = 0, b = 0;
                while(point_is_in(image, x + a, y) && check_color(image, x + a,
y, color)){
                    while(point_is_in(image, x + a, y + b) && check_color(image,
x + a, y + b, color)){
                        current_area += a+1;
                        b++;
                        if(b > b_current_max)
                            break;
                    }
                    if(current_area > max_current_area){
                        max_current_area = current_area;
                        a_max = a;
                        b_max = b - 1;
                        b_current_max = b - 1;
```

```
                    }
                    current_area = 0;
                    b = 0;
                    a++;
                }

                if(max_current_area > max_area){
                        max_area = max_current_area;
                        x1 = x, y1 = y;
                        x2 = x + a_max, y2 = y + b_max;
                }
            }

    if(max_area == 0){
        int* result = malloc(4 * sizeof(int));
        result[0] = -1, result[1] = -1, result[2] = -1, result[3] = -1;
        return result;
    }

    int* result = malloc(4 * sizeof(int));
    result[0] = x1, result[1] = y1, result[2] = x2, result[3] = y2;
    return result;
}

void recolor_biggest_rect(Png* image, int old_color[3], int new_color[3]){
    int* coordinates = find_biggest_rect(image, old_color);
    for(int x = coordinates[0]; x <= coordinates[2]; x++)
        for(int y = coordinates[1]; y <= coordinates[3]; y++)
            set_pixel(image, x, y, new_color);
}

void make_collage(Png* image, int number_x, int number_y){
    Png* new_image = create_new_image(image, image->width*number_x, image-
>height*number_y);

    png_byte* new_ptr;
    png_byte* ptr;

    for(int i = 0; i < number_x; i++)
        for(int j = 0; j < number_y; j++)
            for(int y = 0; y < image->height; y++)
                for(int x = 0; x < image->width; x++){
                    new_ptr = &(new_image->row_pointers[y+j*image->height][(x +
i*image->width)*image->channels]);
                    ptr = &(image->row_pointers[y][x*image->channels]);
                    for(int k = 0; k < image->channels; k++)
                        new_ptr[k] = ptr[k];
                }

    *image = *new_image;
}

void image_processing(int argc, char* argv[], Png* image, char** output_file,
char* input_file){

    int opt;
    char* options = "-tbcfhFP:T:C:L:O:o:i:N:X:Y:?";
    int option_index = 0;
    optind = 0;
    int input_is_set = 0;
    struct option long_options[] = {
        {"triangle", 0, NULL, 't'},
```

```
        {"biggest_rect", 0, NULL, 'b'},
        {"collage", 0, NULL, 'c'},
        {"info", 0, NULL, 'f'},
        {"help", 0, NULL, 'h'},
        {"input", 1, NULL, 'i'},
        {"output", 1, NULL, 'o'},
        {"points", 1, NULL, 'P'},
        {"thickness", 1, NULL, 'T'},
        {"color", 1, NULL, 'C'},
        {"fill", 0, NULL, 'F'},
        {"fill_color", 1, NULL, 'L'},
        {"old_color", 1, NULL, 'O'},
        {"new_color", 1, NULL, 'N'},
        {"number_y", 1, NULL, 'Y'},
        {"number_x", 1, NULL, 'X'},
        {0, 0, 0, 0}
    };

    char* regexColor = "^([0-9]+)\\.([0-9]+)\\.([0-9]+)$";
    regex_t regexColorCompiled;
    size_t maxColorGroups = 4;
    regcomp(&regexColorCompiled, regexColor, REG_EXTENDED);
    regmatch_t groupArrayColor[maxColorGroups];

    char* regexPoints = "([0-9]+)\\.([0-9]+)\\.([0-9]+)\\.([0-9]+)\\.([0-
9]+)\\.([0-9]+)";
    regex_t regexPointsCompiled;
    size_t maxPointsGroups = 7;
    regcomp(&regexPointsCompiled, regexPoints, REG_EXTENDED);
    regmatch_t groupArrayPoints[maxPointsGroups];

    char key = 0;

    int x1, x2, x3, y1, y2, y3;
    int thickness;
    int color[3];
    int fill_color[3];
    int p_is_set = 0;
    int t_is_set = 0;
    int c_is_set = 0;
    int f_is_set = 0;
    int fc_is_set = 0;

    int old_color[3];
    int new_color[3];
    int O_is_set = 0;
    int N_is_set = 0;

    int x, y;
    int x_is_set = 0;
    int y_is_set = 0;

    while((opt = getopt_long(argc, argv, options, long_options, &option_index))
!= -1){
        switch(opt){
        case 't':
        case 'b':
        case 'c':
            key = opt;
            break;

        case 'P':
```

```
            if(regexec(&regexPointsCompiled, optarg, maxPointsGroups,
groupArrayPoints, 0) == REG_NOMATCH)
                raise_error("Wrong argument for --points.");
            get_points(optarg, &x1, &y1, &x2, &y2, &x3, &y3, groupArrayPoints);
            p_is_set = 1;
            break;
        case 'T':
            if(atoi(optarg) <= 0)
                raise_error("Wrong argument for --thickness.");
            thickness = atoi(optarg);
            t_is_set = 1;
            break;
        case 'C':
            if(regexec(&regexColorCompiled, optarg, maxColorGroups,
groupArrayColor, 0) == REG_NOMATCH || get_color(optarg, color, groupArrayColor))
                raise_error("Wrong argument for --color.");
            get_color(optarg, color, groupArrayColor);
            c_is_set = 1;
            break;
        case 'F':
            f_is_set = 1;
            break;
        case 'L':
            if(regexec(&regexColorCompiled, optarg, maxColorGroups,
groupArrayColor, 0) == REG_NOMATCH || get_color(optarg, fill_color,
groupArrayColor))
                raise_error("Wrong argument for --color.");
            get_color(optarg, fill_color, groupArrayColor);
            fc_is_set = 1;
            break;

        case 'O':
            if(regexec(&regexColorCompiled, optarg, maxColorGroups,
groupArrayColor, 0) == REG_NOMATCH || get_color(optarg, old_color,
groupArrayColor))
                raise_error("Wrong argument for --old_color.");
            O_is_set = 1;
            break;
        case 'N':
            if(regexec(&regexColorCompiled, optarg, maxColorGroups,
groupArrayColor, 0) == REG_NOMATCH || get_color(optarg, new_color,
groupArrayColor))
                raise_error("Wrong argument for --new_color.");
            N_is_set = 1;
            break;

        case 'X':
            if(atoi(optarg) <= 0)
                raise_error("Wrong argument for --number_x.");
            x = atoi(optarg);
            x_is_set = 1;
            break;
        case 'Y':
            if(atoi(optarg) <= 0)
                raise_error("Wrong argument for --number_y.");
            y = atoi(optarg);
            y_is_set = 1;
            break;

        case 'o':
            if(!strcmp(optarg, input_file))
```

```
                raise_error("Names of the input and output files are the
same.");
            *output_file = strdup(optarg);
            break;
        case 'f':
            info(image);
            exit(0);
        case 'i':
            input_is_set = 1;
            break;
        case 'h':
            print_help();
            exit(0);
        case '?':
            raise_error("Wrong option.");
        case 1:;
            char* message = malloc(MAX_STRING*sizeof(char));
            sprintf(message, "Argument \"%s\" was unidentified and ignored.",
optarg);
            opt = getopt_long(argc, argv, options, long_options, &option_index);
            if(opt != -1 || input_is_set == 1)
                raise_error(message);
            break;
        }
    }


    switch(key){
        case 't':
            if(p_is_set + t_is_set + c_is_set - f_is_set + fc_is_set < 3)
                raise_error("Not all options were set for --triangle.");
            draw_triangle(image, x1, y1, x2, y2, x3, y3, thickness, color,
f_is_set, fill_color);
            break;
        case 'b':
            if(N_is_set + O_is_set < 2)
                raise_error("Not all options were set for --biggest_rect.");
            recolor_biggest_rect(image, old_color, new_color);
            break;
        case 'c':
            if(x_is_set + y_is_set < 2)
                raise_error("Not all options were set for --collage.");
            make_collage(image, x, y);
            break;
        case 0:
            printf("No command was entered.\n");
            print_help();
            exit(0);
            break;
    }

    if(!strcmp(*output_file, BLANK_STRING))
        *output_file = strdup("out.png");
}
```

Название файла: help.h

```
#pragma once

#include <stdlib.h>
#include <stdio.h>
#include "struct.h"
```

```
void print_help();
void raise_error(char* reason);
void raise_reading_error(char* reason);
void info(Png* image);
```

Название файла: help.c

```
#include "help.h"

void print_help(){
    printf("Flags -o, --output are used to set name of output file (out.png by
default).\n");
    printf("Flags -i, --input are used to set name of input file (last argument
by default).\n");
    printf("Flag --info is used to get the info about recieved png image.\n");
    printf("\nThere are some functions to edit the image:\n");

    printf("Flag --triangle is used to draw a triangle. For initializing it you
need to set some values:\n");
    printf("\t-P, --points <x1.y1.x2.y2.x3.y3>:\n\t\tCoordinates of points of
the triangle.\n");
    printf("\t-T, --thickness <number>:\n\t\tThickness of the lines. Requires
positive number.\n");
    printf("\t-C, --color <rrr.ggg.bbb>:\n\t\tColor of the lines. rrr/ggg/bbb -
numbers specifying red/green/blue color component.\n");
    printf("\t-F, --fill:\n\t\tIf there is a flag, the triangle will be
filled.\n");
    printf("\t--fill_color <rrr.ggg.bbb>:\n\t\tFill color. Take values similar
to --color.\n");

    printf("\nFlag --biggest_rect is used to find the largest rectangle of a
given color and recolor it with a different color. For initializing it you need
to set some values:\n");
    printf("\t--old_color <rrr.ggg.bbb>:\n\t\tThe color of the rectangle to
find. rrr/ggg/bbb - numbers specifying red/green/blue color component.\n");
    printf("\t--new_color <rrr.ggg.bbb>:\n\t\tThe color to recolor the
rectangle. Take values similar to --old_color.\n");

    printf("\nFlag --collage is used to create collage. For initializing it you
need to set some values:\n");
    printf("\t-X, --number_x <number>:\n\t\tNumber of images vertically.
Requires positive number.\n");
    printf("\t-Y, --number_y <number>:\n\t\tNumber of images horizontally.
Requires positive number.\n");
}

void raise_error(char* reason){
    fprintf(stderr, "Error: %s\nSee '--help' or '-h' for a complete list of
options.\n", reason);
    exit(41);
}

void raise_reading_error(char* reason){
    fprintf(stderr, "Error: %s.\n", reason);
    exit(42);
}

void info(Png* image){
    printf("Image Width: %d\n", image->width);
    printf("Image Height: %d\n", image->height);
    printf("Image Bit Depth: %d\n", image->bit_depth);
    printf("Image Channels: %d\n", image->channels);
    if (image->color_type == PNG_COLOR_TYPE_RGB) {
```

```
        printf("Image Colour Type: RGB\n");
    } else {
        printf("Image Colour Type: RGB_A\n");
    }
}
```

Название файла: Makefile

```
TARGET = cw
CC = gcc
CFLAGS = -g
SRC = $(wildcard *.c)
OBJ = $(patsubst %.c, %.o, $(SRC))

all : $(TARGET)

$(TARGET) : $(OBJ)
    $(CC) $(CFLAGS) $^ $(CFLAGS) -lpng -o $@

%.o : %.c
    $(CC) $(CFLAGS) -c $< -lpng -o $@

clean :
    rm $(TARGET) *.o
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Фото для обработки – рисование треугольника

```
./cw --input vremya_na_ishode --output output.png --triangle --points
400.250.400.750.650.500 --color 0.0.0 --thickness 5 --fill --fill_color
0.0.0
```

Результат работы программы:

Фото для обработки – поиск наибольшего прямоугольника заданного цвета и перекрашивание его в новый

```
./cw --input pravda.png --biggest_rect --old_color 255.255.255 --new_color 0.255.0
```

Результат работы программы:

Фото для обработки – создание коллажа из исходного изображения

```
./cw --input voistinu --output out.png --collage --number_x 3 --number_y
2
```

Результат работы программы: