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. 0382 _____ Санников В.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Санников В.А.

Группа 0382

Тема работы: Обработка PNG файла

Исходные данные:

Вариант 14

Программа **должна** иметь CLI или GUI. Более подробно тут:

Общие сведения

- **Формат картинки PNG (рекомендуем использовать библиотеку libpng)**
- файл всегда соответствует формату PNG
- 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 реализовывать следующий функционал по обработке PNG-файла

1. Рисование треугольника. Треугольник определяется
    - Координатами его вершин
    - Толщиной линий
    - Цветом линий
    - Треугольник может быть залит или нет
    - цветом которым он залит, если пользователем выбран залитый
2. Находит самый большой прямоугольник заданного цвета и перекрашивает его в другой цвет. Функционал определяется:

- Цветом, прямоугольник которого надо найти
- Цветом, в который надо его перекрасить

3. Создать коллаж размера N*M из одного либо нескольких фото -- на выбор студента (либо оба варианта по желанию). В случае с одним изображением коллаж представляет собой это же самое изображение повторяющееся N*M раз.
    - Количество изображений по “оси” Y
    - Количество изображений по “оси” X
    - Перечень изображений (если выбрана усложненная версия задания)
4. Рисование отрезка. Отрезок определяется:
    - координатами начала
    - координатами конца
    - цветом
    - толщиной

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: 18.04.2021

Дата сдачи реферата: 06.05.2021

Дата защиты реферата: 18.05.2021

Студент _____ Санников В.А.

Преподаватель _____ Берленко Т.А.

АННОТАЦИЯ

Курсовая работа представляет собой реализацию программы на языке Си в качестве решения задачи по обработке PNG изображения. Для работы с изображением использовались функции стандартных библиотек, динамическая память, структуры, библиотека pnglib и билиотека getopt.

Исходный код работы программы приведён в приложении А.

Пример работы программы приведён в приложении Б.

SUMMARY

The course work is an implementation of a program in C as a solution to the problem of processing PNG images. Standard library functions, dynamic memory, structures library libpng and getopt were used to work with images.

The source code of the program is given in Appendix A.

An example of how the program works is given in Appendix B.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является обработка PNG-файла, используя библиотеку libpng и стандартные библиотеки языка Си.

Для ввода данных, выбора способа обработки и самого изображения используется CLI.

Программа реализована на операционной системе Linux.

1. ЗАДАНИЕ

Вариант 14.

Программа **должна** иметь CLI или GUI. Более подробно тут:

Общие сведения

- **Формат картинки PNG (рекомендуем использовать библиотеку libpng)**
- файл всегда соответствует формату PNG
- 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 реализовывать следующий функционал по обработке PNG-файла

1. Рисование треугольника. Треугольник определяется
    - Координатами его вершин
    - Толщиной линий
    - Цветом линий
    - Треугольник может быть залит или нет
    - цветом которым он залит, если пользователем выбран залитый
2. Находит самый большой прямоугольник заданного цвета и перекрашивает его в другой цвет. Функционал определяется:
    - Цветом, прямоугольник которого надо найти
    - Цветом, в который надо его перекрасить
3. Создать коллаж размера N*M из одного либо нескольких фото -- на выбор студента (либо оба варианта по желанию). В случае с одним изображением коллаж представляет собой это же самое изображение повторяющееся N*M раз.
    - Количество изображений по “оси” Y
    - Количество изображений по “оси” X
    - Перечень изображений (если выбрана усложненная версия задания)

4. Рисование отрезка. Отрезок определяется:

- координатами начала
- координатами конца
- цветом
- толщиной

2. ХОД РАБОТЫ

2.1. Изучение основных теоретических положений и требований к выполнению работы.

2.2. Разработка кода.

2.2.1. Ход решения:

Используется стандартная библиотека Си, её заголовочные файлы stdio.h; stdlib.h; string.h; png.h — для взаимодействия непосредственно с PNG файлом.

Пользователь выбирает функцию которую хочет использовать, задает параметры и пишет PNG файл, который хочет изменить.

В структуру записывается данные о PNG файле (глубина каналов, пиксели, тип цвета и т.д.) с помощью функции библиотеки stdlib - read_png_file.

Далее происходит обработка изображения с помощью функции и вывод изображения на экран (функция write_png_file).

Для решения поставленной задачи написаны функции: make_collage, paint_line, print_triangle.

Также реализован CLI с помощью библиотеки getopt. Пользователь взаимодействует с программой посредством ввода ключей через «-».

В конце освобождается память и программа завершается.

Входные данные:

На вход программе поступает изображение формата PNG. Информация о нем считывается в структуру данных. Если возникают ошибки, то программа сообщает об этом.

ЗАКЛЮЧЕНИЕ

Разработана программа по обработке PNG изображения, считывающая PNG файл с цветом RBG в бинарном виде, сохраняющая информацию о файле в структуру и обрабатывающая эти данные функциями, которые выберет пользователь. Взаимодействие между пользователем и программой происходит с помощью CLI.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. http://www.libpng.org/pub/png/libpng-1.2.5-manual.html
2. https://ru.wikipedia.org/wiki/Си_(язык_программирования)
3. https://en.wikipedia.org/wiki/Getopt
4. https://habr.com/ru/

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

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

    png_structp png_ptr;
    png_infop info_ptr;
    png_byte **row_pointers;
};

void print_info(struct Png *image){
    printf("Ширина в пикселях: %d\n", image->width);
    printf("Высота в пикселях: %d\n", image->height);
    printf("Тип цвета: %u\n", image->color_type);
    printf("Глубина цвета: %u бит\n", image->bit_depth);
}

void printHelp(){
    printf("Это программа с CLI для редактирования png файлов, версия
программы 0.0001 :)\n");
    printf("Поддерживаются только файлы с глубиной цвета RGB!\n");
    printf("Формат ввода: ./bmpedit [имя исходного файла] [функция] -
[ключ1]/--[полный ключ1] [аргумент1] ...\n\n");
    printf("Функции/ключи:\n");
    printf("triangle [имя файла] - рисование треугольника с возможностью
его залить и выбрать цвет.\n");
    printf("    -f/--first  [<x-координата>.<y-координата>] - первая
вершина треугольника\n");
    printf("    -s/--second  [<x-координата>.<y-координата>] - вторая
вершина треугольника\n");
    printf("    -t/--third [<x-координата>.<y-координата>] - третья
вершина треугольника\n");
    printf("    -l/--line [<число>] - толщина сторон треугольника(в
пикселях)\n");
    printf("    -C/--color [<число>.<число>.<число>] - цвет заливки и
треугольника (RGB)\n");
    printf("    -c/--cast [<число>] - заливка треугольника (по умолчанию
без заливки) (1 - заливка, 0 - нет)\n");
    printf("line [имя файла] - рисование прямой линии.\n");
    printf("    -f/--first  [<x-координата>.<y-координата>] - начало
линии\n");
    printf("    -s/--second  [<x-координата>.<y-координата>] - конец
линии\n");
    printf("    -l/--line [<число>] - толщина линии(в пикселях)\n");
    printf("    -C/--color [<число>.<число>.<число>] - цвет линии (RGB)\
n");
```

```
    printf("collage [имя файла] - создается коллаж из изображения.\n");
    printf("    -x/--xPhoto  [<число>] - количество изображений по оси X\
n");
    printf("    -y/--yPhoto  [<число>] - количество изображений по оси Y\
n");
    printf("help - вывод справки о работе программы.\n");
    printf("[имя файла] info - вывод информации об изображении.\n");
    printf("-o/--output [путь] - файл для вывода (по умолчанию исходный
файл)\n");
}

void read_png_file(char *file_name, struct Png *image) {
    int y;
    png_byte header[8];

    FILE *fp = fopen(file_name, "rb");

    if (!fp){
        printf("Ошибка открытия файла на чтение!\n");
    }

    fread(header, 1, 8, fp);
    if (png_sig_cmp(header, 0, 8)){
        printf("Это не PNG файл!\n");
        exit(-1);
    }

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

    if (!image->png_ptr){
        printf("png_create_read_struct failed\n");
        exit(-1);
    }

    image->info_ptr = png_create_info_struct(image->png_ptr);
    if (!image->info_ptr){
        printf("png_create_info_struct failed\n");
        exit(-1);
    }

    if (setjmp(png_jmpbuf(image->png_ptr))){
        printf("error during init_io\n");
        exit(-1);
    }

    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);
```

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

    if (setjmp(png_jmpbuf(image->png_ptr))){
        printf("error during read_image\n");
        exit(-1);
    }

    image->row_pointers = (png_byte **) malloc(sizeof(png_bytep) * image-
>height * image->width);
    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);
}

int process_file(struct Png *image) {

    if (png_get_color_type(image->png_ptr, image->info_ptr) ==
PNG_COLOR_TYPE_RGB){
        return 3;
    }

    printf("Такой тип цвета не поддерживается!\n");
    exit(-1);
}

void write_png_file(char *file_name, struct Png *image) {
    FILE *fp = fopen(file_name, "wb");

    if (!fp){
        printf("Ошибка открытия результирующего файла!\n");
        exit(-1);
    }

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

    if (!image->png_ptr){
        printf("png_create_write_struct failed\n");
        exit(-1);
    }

    image->info_ptr = png_create_info_struct(image->png_ptr);
    if (!image->info_ptr){
        printf("png_create_info_struct failed\n");
        exit(-1);
    }

    if (setjmp(png_jmpbuf(image->png_ptr))){
        printf("error during init_io\n");
        exit(-1);
    }

    png_init_io(image->png_ptr, fp);
```

```
    if (setjmp(png_jmpbuf(image->png_ptr))){
        printf("error during writing header\n");
        exit(-1);
    }

    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("Ошибка чтения байтов!\n");
        exit(-1);
    }

    png_write_image(image->png_ptr, image->row_pointers);
    if (setjmp(png_jmpbuf(image->png_ptr))){
        printf("error during end of write\n");
    }

    png_write_end(image->png_ptr, NULL);

    fclose(fp);
}

void draw_Circle(struct Png *image, int x0, int y0, int line_fat, int
width_pixel, int Red, int Green, int Blue) {
    int x = 0;
    int radius = line_fat / 2;
    int y = radius;
    int start = y0 - radius;
    int end = y0 + radius;
    int delta = 1 - 2 * radius;
    int error;
    while(y >= 0) {
        png_byte *row = image->row_pointers[y0 + y];
        png_byte *ptr = &(row[(x0 + x) * width_pixel]);
        ptr[0] = Red;
        ptr[1] = Green;
        ptr[2] = Blue;

        png_byte *row1 = image->row_pointers[y0 - y];
        png_byte *ptr1 = &(row1[(x0 + x) * width_pixel]);
        ptr1[0] = Red;
        ptr1[1] = Green;
        ptr1[2] = Blue;

        png_byte *row2 = image->row_pointers[y0 + y];
        png_byte *ptr2 = &(row2[(x0 - x) * width_pixel]);
        ptr2[0] = Red;
        ptr2[1] = Green;
        ptr2[2] = Blue;

        png_byte *row3 = image->row_pointers[y0 - y];
        png_byte *ptr3 = &(row3[(x0 - x) * width_pixel]);
```

```
            ptr3[0] = Red;
            ptr3[1] = Green;
            ptr3[2] = Blue;
            error = 2 * (delta + y) - 1;
            while(start <= y0) {
                for (int i = abs(x - x0); i < (x + x0); i++) {
                    png_byte *row4 = image->row_pointers[start];
                    png_byte *ptr4 = &(row4[i * width_pixel]);
                    ptr4[0] = Red;
                    ptr4[1] = Green;
                    ptr4[2] = Blue;

                    png_byte *row5 = image->row_pointers[end];
                    png_byte *ptr5 = &(row5[i * width_pixel]);
                    ptr5[0] = Red;
                    ptr5[1] = Green;
                    ptr5[2] = Blue;
                }
                if(error > 0) {
                    start++;
                    end--;
                }
                break;
            }
            if(delta < 0 && error <= 0) {
                ++x;
                delta += 2 * x + 1;
                continue;
            }
            error = 2 * (delta - x) - 1;
            if(delta > 0 && error > 0) {
                --y;
                delta += 1 - 2 * y;
                continue;
            }
            ++x;
            delta += 2 * (x - y);
            --y;
        }
    }

    void paint_line(struct Png *image, int width_pixel, int x0, int y0, int
    x1, int y1, int line_fat, int Red, int Green, int Blue){
        int A, B, sign;
        A = y1 - y0;
        B = x0 - x1;
        if (abs(A) > abs(B)) sign = 1;
        else sign = -1;
        int signa, signb;
        if (A < 0) signa = -1;
        else signa = 1;
        if (B < 0) signb = -1;
        else signb = 1;
        int f = 0;
        png_byte *row = image->row_pointers[y0];
        png_byte *ptr = &(row[x0 * width_pixel]);
        ptr[0] = Red;
        ptr[1] = Green;
```

```
    ptr[2] = Blue;
    draw_Circle(image, x0, y0, line_fat, width_pixel, Red, Green, Blue);
    int x = x0, y = y0;
    if (sign == -1){
        do {
            f += A * signa;
            if (f > 0)
            {
                f -= B * signb;
                y += signa;
            }
            x -= signb;
            png_byte *row1 = image->row_pointers[y];
            png_byte *ptr1 = &(row1[x * width_pixel]);
            ptr1[0] = Red;
            ptr1[1] = Green;
            ptr1[2] = Blue;
            draw_Circle(image, x, y, line_fat, width_pixel, Red, Green,
Blue);
        } while (x != x1 || y != y1);
    }
    else
    {
        do {
            f += B * signb;
            if (f > 0) {
                f -= A * signa;
                x -= signb;
            }
            y += signa;
            png_byte *row2 = image->row_pointers[y];
            png_byte *ptr2 = &(row2[x * width_pixel]);
            ptr2[0] = Red;
            ptr2[1] = Green;
            ptr2[2] = Blue;
            draw_Circle(image, x, y, line_fat, width_pixel, Red, Green,
Blue);
        } while (x != x1 || y != y1);
    }
}

void paint_line_for_triangle(struct Png *image, int width_pixel, int x0,
int y0, int x1, int y1,
        int line_fat, png_byte **coordinates, int Red, int Green, int
Blue){
    int A, B, sign;
    A = y1 - y0;
    B = x0 - x1;
    if (abs(A) > abs(B)) sign = 1;
    else sign = -1;
    int signa, signb;
    if (A < 0) signa = -1;
    else signa = 1;
    if (B < 0) signb = -1;
    else signb = 1;
    int f = 0;
    png_byte *row = image->row_pointers[y0];
    png_byte *ptr = &(row[x0 * width_pixel]);
```

```
    ptr[0] = Red;
    ptr[1] = Green;
    ptr[2] = Blue;
    coordinates[y0][x0] = 1;
    draw_Circle(image, x0, y0, line_fat, width_pixel, Red, Green, Blue);
    int x = x0, y = y0;
    if (sign == -1){
        do {
            f += A * signa;
            if (f > 0)
            {
                f -= B * signb;
                y += signa;
            }
            x -= signb;
            png_byte *row1 = image->row_pointers[y];
            png_byte *ptr1 = &(row1[x * width_pixel]);
            ptr1[0] = Red;
            ptr1[1] = Green;
            ptr1[2] = Blue;
            coordinates[y][x] = 1;
            draw_Circle(image, x, y, line_fat, width_pixel, Red, Green,
Blue);
        } while (x != x1 || y != y1);
    }
    else
    {
        do {
            f += B * signb;
            if (f > 0) {
                f -= A * signa;
                x -= signb;
            }
            y += signa;
            png_byte *row2 = image->row_pointers[y];
            png_byte *ptr2 = &(row2[x * width_pixel]);
            ptr2[0] = Red;
            ptr2[1] = Green;
            ptr2[2] = Blue;
            coordinates[y][x] = 1;
            draw_Circle(image, x, y, line_fat, width_pixel, Red, Green,
Blue);
        } while (x != x1 || y != y1);
    }
}

void print_triangle(struct Png *image, int width_pixel, int x0, int y0,
int x1, int y1, int x2, int y2,
        int line_fat, int flag, int Red, int Green, int Blue){
    int y = 0, x, start_x, end_x;

    png_byte **coordinates = (png_byte **) malloc(sizeof(png_bytep *) *
image->height);
    for (int i = 0; i < image->height; i++)
        coordinates[i] = (png_byte *) malloc(png_get_rowbytes(image-
>png_ptr, image->info_ptr));
```

```
    paint_line_for_triangle(image, width_pixel, x0, y0, x1, y1, line_fat,
coordinates, Red, Green, Blue);
    paint_line_for_triangle(image, width_pixel, x1, y1, x2, y2, line_fat,
coordinates, Red, Green, Blue);
    paint_line_for_triangle(image, width_pixel, x2, y2, x0, y0, line_fat,
coordinates, Red, Green, Blue);
    //заливка треугольника
    if(flag == 1) {
        while(y < image->height){
            int count = 0;
            for(x = 0; x < image->width; x++){
                if(coordinates[y][x] == 1){
                    count += 1;
                    if(count == 1)
                        start_x = x;
                    if(count >= 2) {
                        end_x = x;
                    }
                }
            }
            if(count >= 2){
                for(int k = start_x; k < end_x; k++){
                    png_byte *row = image->row_pointers[y];
                    png_byte *ptr = &(row[k * width_pixel]);
                    ptr[0] = Red;
                    ptr[1] = Green;
                    ptr[2] = Blue;
                }
                y++;
            }else{
                y++;
            }
        }
    }
}

void replace(png_byte* buf, png_byte* for_replace, int width_pixel){
    for (int i = 0; i < width_pixel; i++){
        buf[i] = for_replace[i];
    }
}

void make_collage(struct Png *image, int width_pixel, int x_photos, int
y_photos){
    int N, M;
    int y, x, old_x, old_y;

    int new_width = image->width * x_photos;
    int new_height = image->height * y_photos;

    png_byte** new_mas = (png_byte**) malloc(sizeof(png_byte*) *
new_height);
    for (y = 0; y < new_height; y++)
        new_mas[y] = (png_byte*) malloc(sizeof(png_byte) * new_width *
width_pixel);

    for (y = 0; y < new_height; y++){
        old_y = y % image->height;
```

```
            png_byte* old_row = image->row_pointers[old_y];
            png_byte* new_row = new_mas[y];
            for (x = 0; x < new_width; x++){
                old_x = x % image->width;
                png_byte* old_pixel = &(old_row[old_x * width_pixel]);
                png_byte* new_pixel = &(new_row[x * width_pixel]);
                replace(new_pixel, old_pixel, width_pixel);
            }
        }

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

        image->row_pointers = new_mas;
        image->width = new_width;
        image->height = new_height;
}

void choice(char* func, int key, int* x0, int* y0, int* x1, int* y1, int*
x2, int* y2, int* line_fat,
            int* cast, char** output, int* Red, int* Green, int* Blue,
int* x_photos, int* y_photos){
        int i = 0;
        switch (key) {
            case 'f':
                *x0 = atoi(optarg);
                while(optarg[i] != '.'){
                    i++;
                }
                *y0 = atoi(&optarg[i+1]);
                break;
            case 's':
                *x1 = atoi(optarg);
                while(optarg[i] != '.'){
                    i++;
                }
                *y1 = atoi(&optarg[i+1]);
                break;
            case 't':
                *x2 = atoi(optarg);
                while(optarg[i] != '.'){
                    i++;
                }
                *y2 = atoi(&optarg[i+1]);
                break;
            case 'l':
                *line_fat = atoi(optarg);
                break;
            case 'c':
                *cast = atoi(optarg);
                break;
            case 'C':
                *Red = atoi(optarg);
                while(optarg[i] != '.'){
                    i++;
                }
```

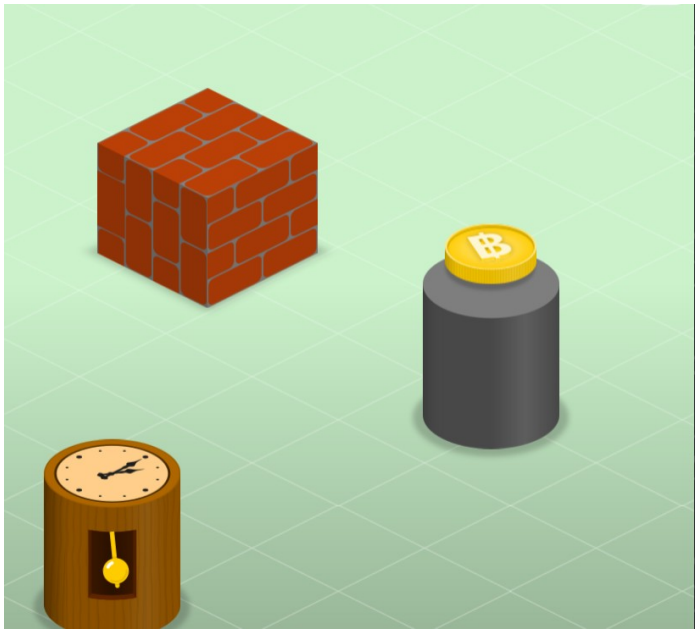
```
            *Green = atoi(&optarg[i+1]);
            while(optarg[i] != '.'){
                i++;
            }
            *Blue = atoi(&optarg[i+1]);
            break;
        case 'x':
            *x_photos = atoi(optarg);
            break;
        case 'y':
            *y_photos = atoi(optarg);
            break;
        case 'o':
            *output = optarg;
            break;
        default:
            printf("Нет такого ключа для %s!\n", func);
            exit(-1);
    }
}

int main(int argc, char **argv) {
    struct Png image;
    int index = 0, key;
    int x0 = 0, y0 = 0, x1 = 0, y1 = 0, x2 = 0, y2 = 0, line_fat = 0,
flag = 0;
    int Red = 0, Green = 0, Blue = 0;
    int x_photos = 1, y_photos = 1;
    char* output = argv[1];

    read_png_file(argv[1], &image);
    int width_pixel = process_file(&image);

    const struct option firstCordStruct = {"first", required_argument,
NULL, 'f'};
    const struct option secondCordStruct = {"second", required_argument,
NULL, 's'};
    const struct option thirdCordStruct = {"third", required_argument,
NULL, 't'};
    const struct option fatLineStruct = {"line", required_argument, NULL,
'l'};
    const struct option castStruct = {"cast", required_argument, NULL,
'c'};
    const struct option xPhotosStruct = {"xPhoto", required_argument,
NULL, 'x'};
    const struct option yPhotosStruct = {"yPhoto", required_argument,
NULL, 'y'};
    const struct option colorStruct = {"color", required_argument, NULL,
'C'};
    const struct option outputStruct = {"output", required_argument,
NULL, 'o'};

    opterr = 0;
    if(argc == 1 || !strcmp(argv[2], "help")){
        printHelp();
        return 0;
    }else if(!strcmp(argv[2], "info")){
        print_info(&image);
```

```
        return 0;
    }else if(!strcmp(argv[2], "triangle")) {
        struct option options[] = {firstCordStruct, secondCordStruct,
thirdCordStruct, fatLineStruct,
                castStruct, colorStruct, outputStruct};
        while((key = getopt_long(argc, argv, "f:s:t:l:c:C:o:", options,
&index)) != -1) {
            choice("triangle", key, &x0, &y0, &x1, &y1, &x2, &y2,
&line_fat,
                    &flag, &output, &Red, &Green, &Blue, 0, 0);
        }
        print_triangle(&image, width_pixel, x0, y0, x1, y1, x2, y2,
line_fat, flag, Red, Green, Blue);
        write_png_file(output, &image);
    }else if(!strcmp(argv[2], "line")) {
        struct option options[] = {firstCordStruct, secondCordStruct,
fatLineStruct, colorStruct,
                                    outputStruct};
        while((key = getopt_long(argc, argv, "f:s:l:C:o:", options,
&index)) != -1) {
            choice("line", key, &x0, &y0, &x1, &y1, 0, 0,  &line_fat,
                    0, &output, &Red, &Green, &Blue, 0, 0);
        }
        paint_line(&image, width_pixel, x0, y0, x1, y1, line_fat, Red,
Green, Blue);
        write_png_file(output, &image);
    }else if(!strcmp(argv[2], "collage")) {
        struct option options[] = {xPhotosStruct, yPhotosStruct,
outputStruct};
        while((key = getopt_long(argc, argv, "x:y:o:", options,
&index)) != -1) {
            choice("collage", key, 0, 0, 0, 0, 0, 0,  0,
                    0, &output, 0, 0, 0, &x_photos, &y_photos);
        }
        make_collage(&image, width_pixel, x_photos, y_photos);
        write_png_file(output, &image);
    }

    for (int y = 0; y < image.height; y++)
        free(image.row_pointers[y]);
    free(image.row_pointers);

    return 0;
}
```

ПРИЛОЖЕНИЕ Б

ПРИМЕР РАБОТЫ ПРОГРАММЫ

Фото для обработки:

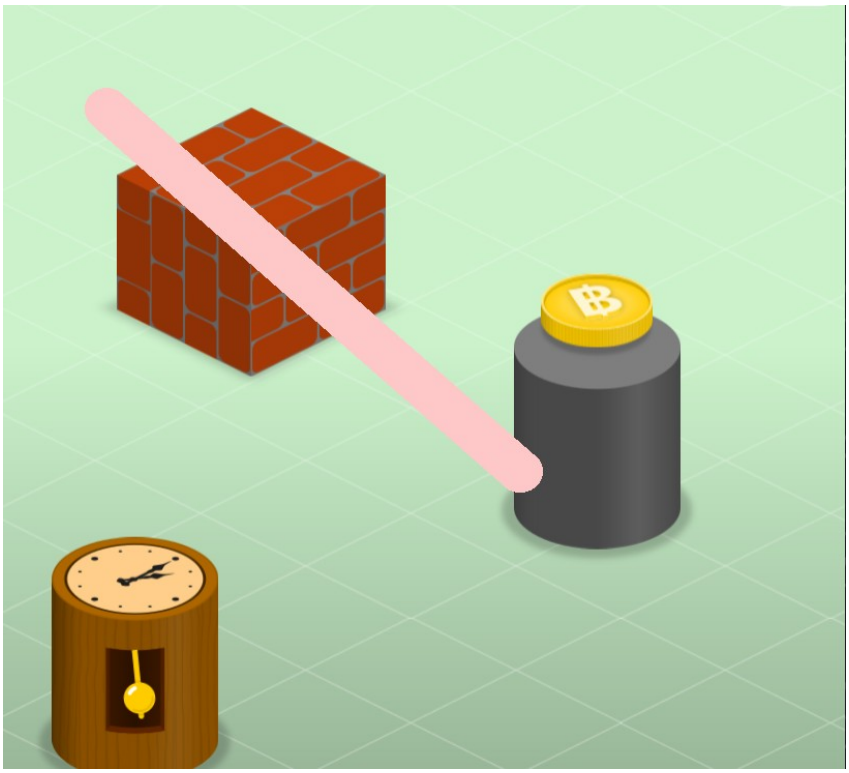


Пример 1: Вывод справки

```
Это программа с CLI для редактирования png файлов, версия программы 0.0001 :)
Поддерживаются только файлы с глубиной цвета RGB!
Формат ввода: ./bmpedit [имя исходного файла] [функция] -[ключ1]/--[полный ключ1] [аргумент1] ...

Функции/ключи:
triangle [имя файла] - рисование треугольника с возможностью его залить и выбрать цвет.
    -f/--first  [<x-координата>.<y-координата>] - первая вершина треугольника
    -s/--second  [<x-координата>.<y-координата>] - вторая вершина треугольника
    -t/--third [<x-координата>.<y-координата>] - третья вершина треугольника
    -l/--line [<число>] - толщина сторон треугольника(в пикселях)
    -C/--color [<число>.<число>.<число>] - цвет заливки и треугольника (RGB)
    -c/--cast [<число>] - заливка треугольника (по умолчанию без заливки) (1 - заливка, 0 - нет)
line [имя файла] - рисование прямой линии.
    -f/--first  [<x-координата>.<y-координата>] - начало линии
    -s/--second  [<x-координата>.<y-координата>] - конец линии
    -l/--line [<число>] - толщина линии(в пикселях)
    -C/--color [<число>.<число>.<число>] - цвет линии (RGB)
collage [имя файла] - создается коллаж из изображения.
    -x/--x_photo  [<число>] - количество изображений по оси X
    -y/--y_photo  [<число>] - количество изображений по оси Y
help - вывод справки о работе программы.
[имя файла] info - вывод информации об изображении.
-o/--output [путь] - файл для вывода (по умолчанию исходный файл)
vadim@vadim-Swift-SF314-41:~/CLionProjects/untitled$ ~
```

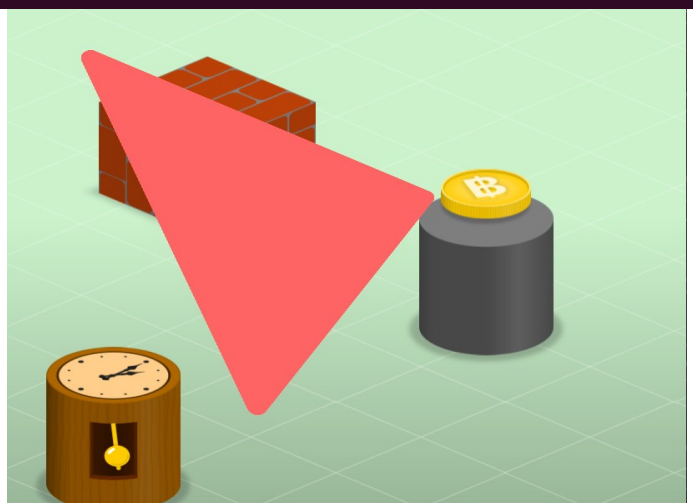
Пример 2: работа функции 1

```
ts/untitled$ ./a.out ./images/img_1001.png line -f 100.100 -s 500.450 -l 40 -C 255.200.0 -o res.png
ts/untitled$
```



Пример 3: работа функции 2

```
itled$ ./a.out ./images/img_1001.png triangle -f 100.100 -s 500.300 -t 300.600 -l 20 -C 255.100.10 -c 1 -o res.png
itled$
```



Пример 3: работа функции 3

```
vadim@vadim-Swift-SF314-41: ~/CLionProjects/untitled
untitled$ ./a.out ./images/img_1001.png collage --xPhoto 4 --yPhoto 4 --output res.png
untitled$
```