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. 1382 ____ Мамин Р.А.

Преподаватель ____ Жангиров Т.Р.

Санкт-Петербург

2022

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Мамин Р.А.

Группа 1382

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: 22.03.2022

Дата сдачи реферата: 26.05.2022

Дата защиты реферата: 09.06.2022

Студент _____ Мамин Р.А.

Преподаватель Жангиров Т. Р.

_____

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, draw_line, draw_triangle, is_rectangle.*

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: cw.c

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <math.h>
#include <png.h>

#define PNG_DEBUG 3

#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;
    int number_of_passes;
    png_bytep *row_pointers;
};

struct Configs {
    int x1, y1;
    int x2, y2;
    int x3, y3;
    int line_fat;
    int line_color_r, line_color_g, line_color_b;
    int paint_color_r, paint_color_g, paint_color_b;
    int is_pour;
    int xPhoto, yPhoto;
    char *output;
};


void printHelp() {
    printf("Это программа с CLI для редактирования png файлов\n");
    printf("Формат ввода: [имя исходного файла] ./a.out  [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/--lineFat [<число>- толщина сторон треугольника(в
пикселях)\n");
```

```
    printf("    -r/--range [<число>.<число>.<число>] - цвет сторон
треугольника (RGB)\n");
    printf("    -C/--color [<число>.<число>.<число>] - цвет заливки
треугольника (RGB)\n");
    printf("    -c/--cast - заливка треугольника (по умолчанию без
заливки)\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/--lineFat [<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("rectangle [имя файла] - поиск самого большого прямоугольника
заданного цвета и его "
           "перекраска в заданный цвет.\n");
    printf("    -C/--color [<число>.<число>.<число>] - цвет перекраски
(RGB)\n");
    printf("    -r/--range [<число>.<число>.<число>] - цвет искомого
прямоугольника (RGB)\n");
    printf("-h/--help - вывод справки о работе программы.\n");
    printf("-o/--output [путь] - файл для вывода (по умолчанию исходный
файл)\n");
}

void information(struct Png *image) {
    printf("info about input picture:\n");
    printf("width:\t%d\n", image->width);
    printf("height:\t%d\n", image->height);
    printf("bit depth:\t%u\n", image->bit_depth);
    printf("color type:\t%u\n", image->color_type);
    printf("\ttypes:\n"
           "\t0 - grayscale\n"
           "\t2 - RGB\n"
           "\t4 - grayscale with alpha\n"
           "\t6 - RGBA\n");

}


void read_png_file(char *file_name, struct Png *image) {
    int x, y;
    char header[8];    // 8 is the maximum size that can be checked

    /* open file and test for it being a png */
    FILE *fp = fopen(file_name, "rb");
    if (!fp) {
        // Some error handling: file could not be opened
        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);
    }

    /* initialize stuff */
    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_read_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);

    image->number_of_passes = png_set_interlace_handling(image->png_ptr);
    png_read_update_info(image->png_ptr, image->info_ptr);

    /* read file */
    if (setjmp(png_jmpbuf(image->png_ptr))) {
        // Some error handling: error during read_image
        printf("error during read_image\n");
        exit(-1);
    }

    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, struct Png *image) {
    int x, y;
    FILE *fp = fopen(file_name, "wb");
```

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

    /* initialize stuff */
    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) {
        // Some error handling: png_create_info_struct failed
        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);


    /* write header */
    if (setjmp(png_jmpbuf(image->png_ptr))) {
        // Some error handling: error during writing header
        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);


    /* write bytes */
    if (setjmp(png_jmpbuf(image->png_ptr))) {
        printf("Ошибка чтения байтов!\n");
        exit(-1);
    }

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


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

    png_write_end(image->png_ptr, NULL);
```

```
    /* cleanup heap allocation */
    for (y = 0; y < image->height; y++)
        free(image->row_pointers[y]);
    free(image->row_pointers);

    fclose(fp);
}

void paint_pixel(struct Png *image, int x, int y, int width_pixel, int Red,
int Green, int Blue) {
//    printf("x: %d y: %d\n", x, y);
    if (x < 0 || x >= image->width || y < 0 || y >= image->height)
        return;
    printf("x: %d y: %d\n", x, y);
    png_byte *row = image->row_pointers[y];
    png_byte *ptr = &(row[x * width_pixel]);
    ptr[0] = Red;
    ptr[1] = Green;
    ptr[2] = Blue;
}

// Рисование круга(для толщины линии)
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) {
        paint_pixel(image, x0 + x, y0 + y, width_pixel, Red, Green, Blue);

        paint_pixel(image, x0 + x, y0 - y, width_pixel, Red, Green, Blue);

        paint_pixel(image, x0 - x, y0 + y, width_pixel, Red, Green, Blue);

        paint_pixel(image, x0 - x, y0 - y, width_pixel, Red, Green, Blue);

        error = 2 * (delta + y) - 1;
        while (start <= y0) {
            for (int i = abs(x - x0); i < (x + x0); i++) {
                paint_pixel(image, i, start, width_pixel, Red, Green,
Blue);

                paint_pixel(image, i, end, width_pixel, Red, Green, 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;

    paint_pixel(image, x0, y0, width_pixel, Red, Green, 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;
            paint_pixel(image, x, y, width_pixel, Red, Green, 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;
            paint_pixel(image, x, y, width_pixel, Red, Green, Blue);
            draw_Circle(image, x, y, line_fat, width_pixel, Red, Green,
Blue);
        } while (x != x1 || y != y1);
    }
}
```



```
// проверка, находится ли текущий пиксель внутри треугольника
```

```
int is_inside(int x0, int y0, int x1, int y1, int x2, int y2, int x3, int
y3) {
    int one = (x1 - x0) * (y2 - y1) - (x2 - x1) * (y1 - y0);
    int two = (x2 - x0) * (y3 - y2) - (x3 - x2) * (y2 - y0);
    int three = (x3 - x0) * (y1 - y3) - (x1 - x3) * (y3 - y0);
    if ((one < 0 && two < 0 && three < 0) || (one > 0 && two > 0 && three >
0)) {
        return 1;
    }
    return 0;
}

//проверка, покрашен ли уже пиксель при заливке треугольника (внутри и на
линии)
int is_not_painted(struct Png *image, int width_pixel, int x, int y, int
Red_p, int Green_p, int
Blue_p, int Red_l, int Green_l, int Blue_l) {
    if (x < 0 || x >= image->width || y < 0 || y >= image->height)
        return 0;
    png_byte *row = image->row_pointers[y];
    png_byte *ptr = &(row[x * width_pixel]);
    int one = (ptr[0] == Red_p) && (ptr[1] == Green_p) && (ptr[2] ==
Blue_p);
    int two = (ptr[0] == Red_l) && (ptr[1] == Green_l) && (ptr[2] ==
Blue_l);
    if (one || two) {
        return 0;
    }
    return 1;
}

// рекурсивная заливка треугольника
void pouring_triangle(struct Png *image, int width_pixel, int x_c, int y_c,
int x1, int y1, int x2,
                      int y2,
                      int x3, int y3, int Red, int Green, int Blue, int
Red_l, int Green_l,
                      int Blue_l) {
    if (is_inside(x_c, y_c, x1, y1, x2, y2, x3, y3) &&
is_not_painted(image, width_pixel, x_c,
                                                                    y_c,
Red, Green, Blue, Red_l,

Green_l, Blue_l)) {
        paint_pixel(image, x_c, y_c, width_pixel, Red, Green, Blue);
        pouring_triangle(image, width_pixel, x_c + 1, y_c, x1, y1, x2, y2,
x3, y3, Red, Green, Blue,
                         Red_l,
                         Green_l, Blue_l);
        pouring_triangle(image, width_pixel, x_c, y_c + 1, x1, y1, x2, y2,
x3, y3, Red, Green, Blue,
                         Red_l,
                         Green_l, Blue_l);
        pouring_triangle(image, width_pixel, x_c - 1, y_c, x1, y1, x2, y2,
x3, y3, Red, Green, Blue,
                         Red_l,
                         Green_l, Blue_l);
        pouring_triangle(image, width_pixel, x_c, y_c - 1, x1, y1, x2, y2,
x3, y3, Red, Green, Blue,
                         Red_l,
```

```
                        Green_l, Blue_l);
    }
}


void draw_triangle(struct Png *image, int width_pixel, int x1, int y1, int
x2, int y2, int x3, int
y3, int line_fat, int is_pour, int Red_l, int Green_l, int Blue_l, int
Red_p, int Green_p,
                   int Blue_p) {
    paint_line(image, width_pixel, x1, y1, x2, y2, line_fat, Red_l,
Green_l, Blue_l);
    paint_line(image, width_pixel, x2, y2, x3, y3, line_fat, Red_l,
Green_l, Blue_l);
    paint_line(image, width_pixel, x3, y3, x1, y1, line_fat, Red_l,
Green_l, Blue_l);
    if (is_pour) {
        int x_c = (x1 + x2 + x3) / 3;
        int y_c = (y1 + y2 + y3) / 3;
        pouring_triangle(image, width_pixel, x_c, y_c, x1, y1, x2, y2, x3,
y3, Red_p, Green_p,
                         Blue_p, Red_l,
                         Green_l, Blue_l);
    }
}

// перенос пикселя при создании коллажа
void replace(png_byte *new_pixel, png_byte *old_pixel, int width_pixel) {
    for (int i = 0; i < width_pixel; i++) {
        new_pixel[i] = old_pixel[i];
    }
}

void make_collage(struct Png *image, int width_pixel, int x_photos, int
y_photos) {
    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 (int y = 0; y < new_height; y++)
        new_mas[y] = (png_byte *) malloc(sizeof(png_byte) * new_width *
width_pixel);
    for (int y = 0; y < new_height; y++) {
        int old_y = y % image->height;
        png_byte *old_row = image->row_pointers[old_y];
        png_byte *new_row = new_mas[y];
        for (int x = 0; x < new_width; x++) {
            int 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 (int 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;
}

int check_X_Y(struct Png *image, int x0, int y0, int *x1, int *y1, int *x2,
int *y2, int *x3, int
*y3, int *x4, int *y4, int width_pixel, int red0, int green0, int blue0) {
    int x1t = 0, y1t = 0, x2t = 0, y2t = 0, x3t = 0, y3t = 0, x4t = 0, y4t
= 0;
    int flag = 0;
    png_byte *row = image->row_pointers[y0];
    for (int x = x0; x < image->width; x++) {
        png_byte *ptr = &(row[x * width_pixel]);
        if (ptr[0] == red0 && ptr[1] == green0 && ptr[2] == blue0) {
            if (flag == 0) {
                x1t = x, y1t = y0;
                flag++;
            }
        } else {
            if (x != (x0 + 1)) {
                x2t = x - 1, y2t = y0;
                flag++;
                break;
            } else {
                return 0;
            }
        }
    }
    for (int y = y0; y < image->height; y++) {
        png_byte *row = image->row_pointers[y];
        png_byte *ptr = &(row[x0 * width_pixel]);
        if (ptr[0] != red0 || ptr[1] != green0 || ptr[2] != blue0) {
            if (y != (y0 + 1) && flag == 2) {
                y3t = y - 1, x3t = x0;
                flag++;
            }
        }
    }
    for (int y = y0; y < image->height; y++) {
        png_byte *row = image->row_pointers[y];
        png_byte *ptr = &(row[x1t * width_pixel]);
        if (ptr[0] != red0 || ptr[1] != green0 || ptr[2] != blue0) {
            if (y != (y0 + 1) && flag == 3) {
                y4t = y - 1, x4t = x2t;
                flag++;
            }
        }
    }
    if (y4t != y3t)
        return 0;
    row = image->row_pointers[y3t];

    for (int x = x3t; x < x4t; x++) {
        png_byte *ptr = &(row[x * width_pixel]);
        if (ptr[0] != red0 || ptr[1] != green0 || ptr[2] != blue0) {
            return 0;
        }
    }
    *x1 = x1t, *y1 = y1t;
    *x2 = x2t, *y2 = y2t;
```

```
    *x3 = x3t, *y3 = y3t;
    *x4 = x4t, *y4 = y4t;
    return 1;

}

//перекраска прямоугольника
void pouring_rectangle(struct Png *image, int width_pixel, int x, int y,
int Red, int Green, int
Blue, int Red_l, int Green_l, int Blue_l) {
    if (!is_not_painted(image, width_pixel, x, y, Red, Green, Blue, Red,
Green, Blue)) {
        paint_pixel(image, x, y, width_pixel, Red, Green, Blue);
        pouring_rectangle(image, width_pixel, x + 1, y, Red, Green, Blue,
Red_l, Green_l, Blue_l);
        pouring_rectangle(image, width_pixel, x - 1, y, Red, Green, Blue,
Red_l, Green_l, Blue_l);
        pouring_rectangle(image, width_pixel, x, y + 1, Red, Green, Blue,
Red_l, Green_l, Blue_l);
        pouring_rectangle(image, width_pixel, x, y - 1, Red, Green, Blue,
Red_l, Green_l, Blue_l);
    }
}

// поиск прямоугольника макс площади
int is_rectangle(struct Png *image, int width_pixel, int red0, int green0,
int blue0, int red1, int
green1, int blue1) {
    int x1, y1, x2, y2, x3, y3, x4, y4;
    int square, square_max = 0;
    int x1_max, y1_max, x2_max, y2_max, x3_max, y3_max, x4_max, y4_max;
    for (int y = 0; y < image->height; y++) {
        png_byte *row = image->row_pointers[y];
        for (int x = 0; x < image->width; x++) {
            png_byte *ptr = &(row[x * width_pixel]);
            if (ptr[0] == red0 && ptr[1] == green0 && ptr[2] == blue0) {
                if (check_X_Y(image, x, y, &x1, &y1, &x2, &y2, &x3, &y3,
&x4, &y4,
                              width_pixel, red0, green0, blue0)) {
                    int square = sqrt((x2 - x1) * (x2 - x1)) * sqrt((y3 -
y1) * (y3 - y1));
                    if (square > square_max) {
                        square_max = square;
                        x1_max = x1, y1_max = y1, x2_max = x2, y2_max = y2,
x3_max = x3, y3_max =
                                y3, x4_max = x4, y4_max = y4;
                    }
                }
            }
        }
    }
    pouring_rectangle(image, width_pixel, x1_max, y1_max, red0, green0,
blue0, red1, green1, blue1);
    return 0;
}

// расстановка конфигураций, переданных через флаги в getopt
void cfg(int **arr, int kl, char *opt) {
    char *str1;
    str1 = strtok(opt, ".");
```

```
    *(arr[0]) = atoi(str1);
    for (int i = 1; i < kl; i++) {
        str1 = strtok(NULL, ".");
        *(arr[i]) = atoi(str1);
    }
}


void choice(struct Configs *config, int opt) {
    char *str1;
    int **arr = malloc(3 * sizeof(int *));
    switch (opt) {
        case 'r':
            arr[0] = &(config->line_color_r), arr[1] = &(config-
>line_color_g), arr[2] = &
                    (config->line_color_b);
            cfg(arr, 3, optarg);
            break;
        case 'f':
            arr[0] = &(config->x1), arr[1] = &(config->y1);
            cfg(arr, 2, optarg);
            break;
        case 's':
            arr[0] = &(config->x2), arr[1] = &(config->y2);
            cfg(arr, 2, optarg);
            break;
        case 't':
            arr[0] = &(config->x3), arr[1] = &(config->y3);
            cfg(arr, 2, optarg);
            break;
        case 'l':
            config->line_fat = atoi(optarg);
            break;
        case 'C':
            arr[0] = &(config->paint_color_r), arr[1] = &(config-
>paint_color_g), arr[2] = &
                    (config->paint_color_b);
            cfg(arr, 3, optarg);
            break;
        case 'c':
            config->is_pour = 1;
            break;
        case 'x':
            config->xPhoto = atoi(optarg);
            break;
        case 'y':
            config->yPhoto = atoi(optarg);
            break;
        case 'o':
            config->output = optarg;
            break;
        case 'h':
            printHelp();
            exit(-1);
            break;
    }
}

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

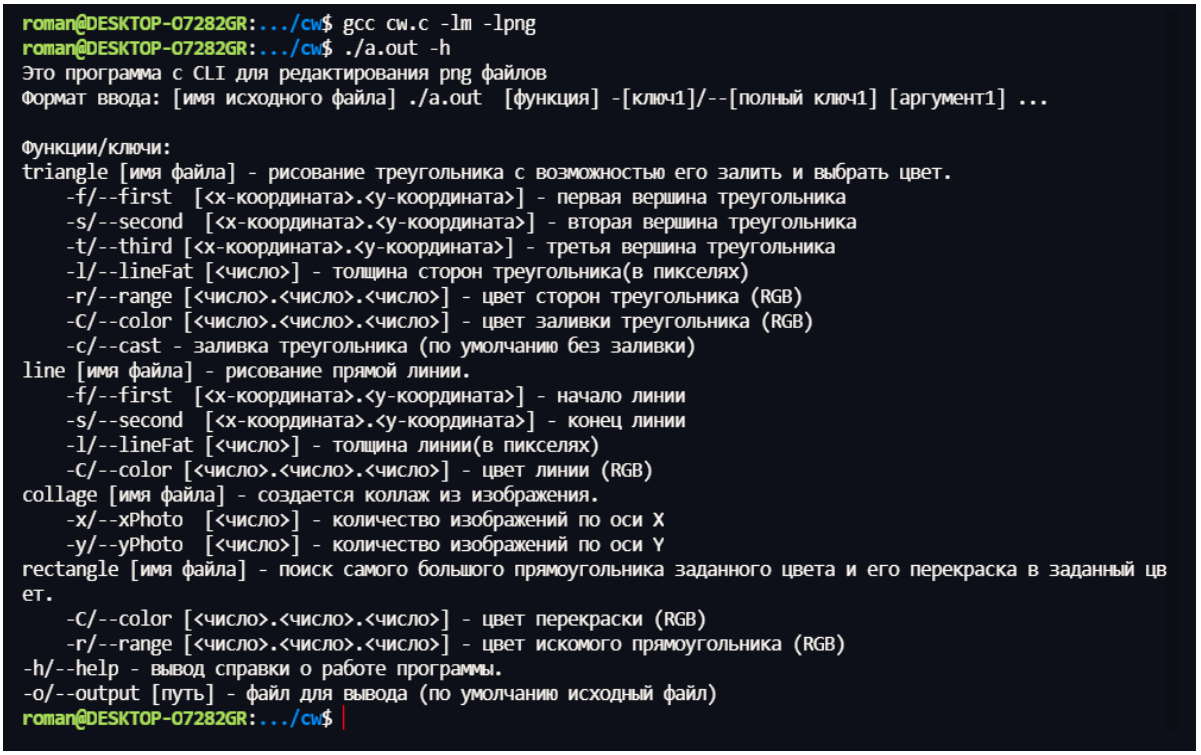
```
    char *opts = "f:s:t:l:C:cx:y:o:hr:";
    char *output = argv[1];
    struct Configs config = {0, 0, 0, 0, 0, 0, 0, 0, 0,
                             0, 0, 0, 0, 0,
                             0, 0, output};
    struct Png image;
    char *func = argv[2];
    char *file = argv[1];
    struct option longOpts[] = {
            {"first",   required_argument, NULL, 'f'},
            {"second",  required_argument, NULL, 's'},
            {"third",   required_argument, NULL, 't'},
            {"lineFat", required_argument, NULL, 'l'},
            {"color",   required_argument, NULL, 'C'},
            {"cast",    no_argument,       NULL, 'c'},
            {"xPhoto",  required_argument, NULL, 'x'},
            {"yPhoto",  required_argument, NULL, 'y'},
            {"output",  required_argument, NULL, 'o'},
            {"help",    no_argument,       NULL, 'h'},
            {"range",   required_argument, NULL, 'r'},
            {NULL,      no_argument,       NULL, 0}
    };
    int opt;
    int longIndex;
    opt = getopt_long(argc, argv, opts, longOpts, &longIndex);
    while (opt != -1) {
        choice(&config, opt);
        opt = getopt_long(argc, argv, opts, longOpts, &longIndex);
    }
    read_png_file(file, &image);
    information(&image);
    if (!strcmp(func, "triangle")) {
        draw_triangle(&image, 3, config.x1, config.y1, config.x2,
config.y2, config.x3,
                      config.y3, config.line_fat, config.is_pour,
config.line_color_r,
                      config.line_color_g,
                      config.line_color_b, config.paint_color_r,
config.paint_color_g,
                      config.paint_color_b);
    }
    if (!strcmp(func, "line")) {
        paint_line(&image, 3, config.x1, config.y1, config.x2, config.y2,
config.line_fat,
                   config.paint_color_r, config.paint_color_g,
config.paint_color_b);
    }
    if (!strcmp(func, "collage")) {
        make_collage(&image, 3, config.xPhoto, config.yPhoto);
    }
    if (!strcmp(func, "rectangle")) {
        is_rectangle(&image, 3, config.line_color_r, config.line_color_g,
config
                .line_color_b, config.paint_color_r, config.paint_color_g,
config.paint_color_b);
    }


    write_png_file(config.output, &image);
    return 0;}
```

ПРИЛОЖЕНИЕ Б

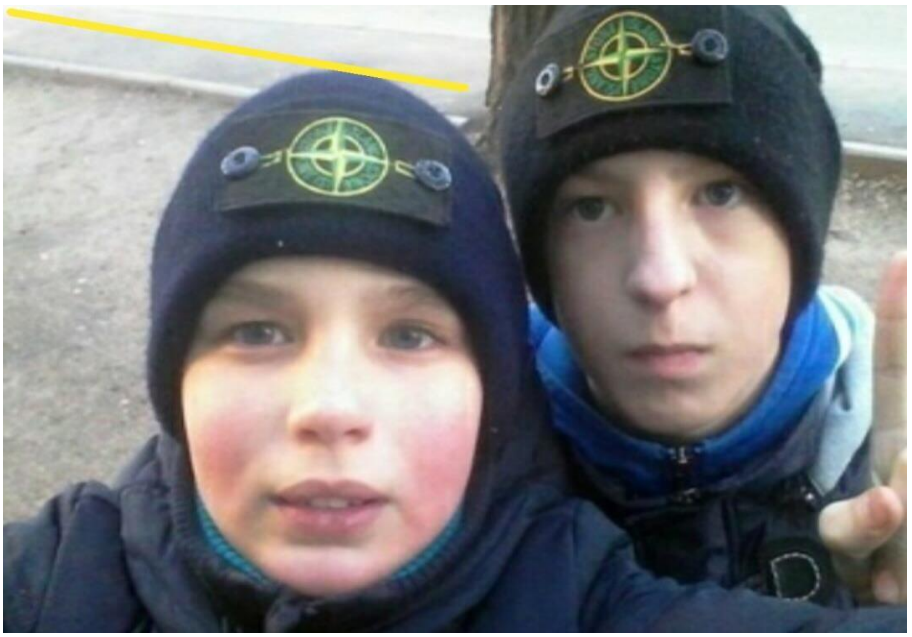
ПРИМЕР РАБОТЫ ПРОГРАММЫ

Фото для обработки:



Пример 1: Вывод справки

```
roman@DESKTOP-07282GR:.../cw$ ./a.out kek.png line -f 10.10 -s 50.50 -l 4 -C 255.234.10 -o lol.png
```

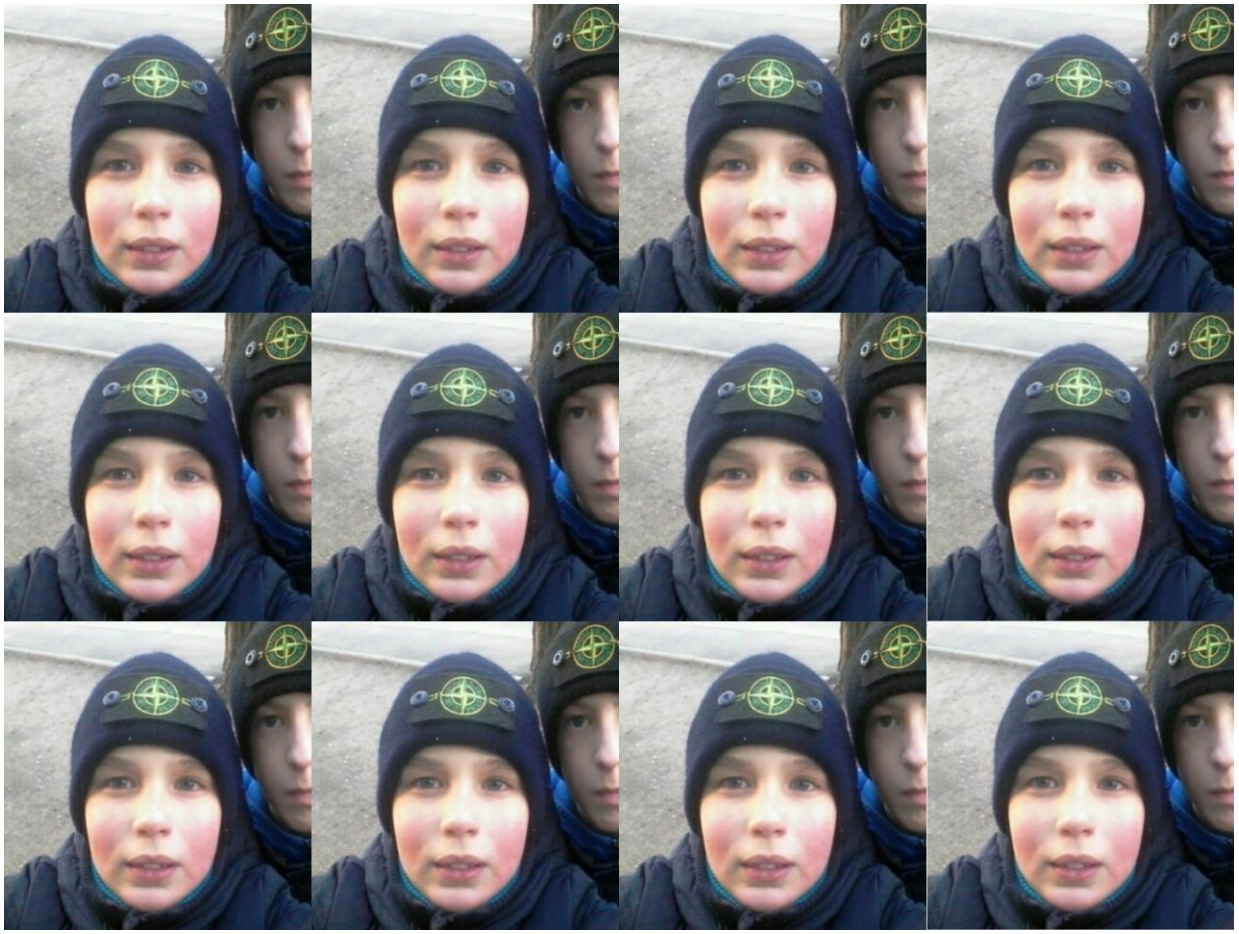


Пример 2: работа функции 1

```
roman@DESKTOP-07282GR:.../cw$ ./a.out kek.png triangle -f 10.10 -s 200.30 -t 100.200 -l 9 -r 255.0.0 -C 0.255.2 -c -o lol.png
```

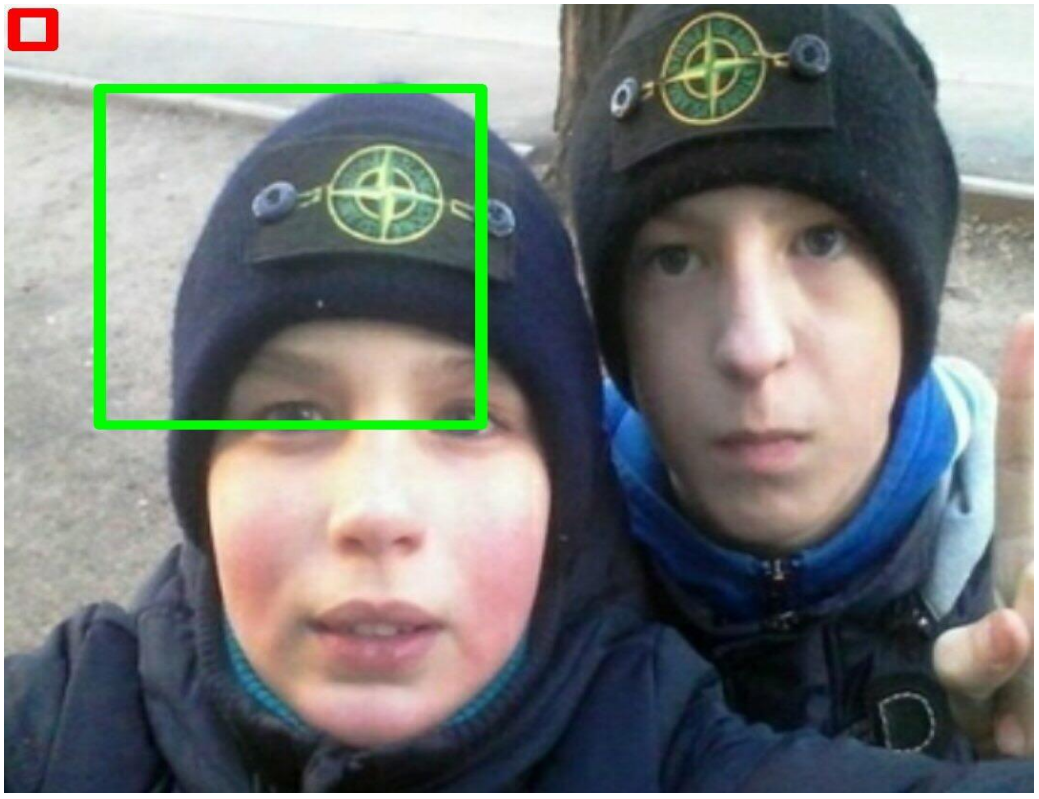
Пример 3: работа функции 2

```
roman@DESKTOP-07282GR:.../cw$ ./a.out kek.png collage -x 4 -y 3 -o lol.png
```



Пример 3: работа функции 3

```
roman@DESKTOP-O7282GR:.../cw$ ./a.out kek.png rectangle -C 0.255.0 -r 255.0.0 -o lol.png
info about input picture:
```



Пример 4: работа функции 4