

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

**КУРСОВАЯ РАБОТА**  
**по дисциплине «Программирование»**  
**Тема: PNG-ФАЙЛЫ.**

Студент гр. 7383

\_\_\_\_\_

Зуев Д. В.

Преподаватель

\_\_\_\_\_

Берленко Т. А.

Санкт-Петербург

2018

## ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Зуев Д. В.

Группа 7383

Тема работы: PNG-ФАЙЛЫ.

Содержание пояснительной записки:

- Содержание
- Введение
- Описание функций для работы со списком
- Тестирование программы
- Заключение
- Список использованных источников
- Приложение А. Исходный код программы

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент

Зуев Д. В.

Преподаватель

Берленко Т. А.

## **АННОТАЦИЯ**

В данной курсовой работе была разработана программа на языке Си, которая позволяет зеркально отразить заданную область PNG-файла по горизонтали. Набор функций включает в себя функции чтения изображения, зеркального отражения области изображения, записи изображения, а также проверки введенных данных. Приведено полное описание исходного кода.

## **SUMMARY**

In this course work was developed a program in C, which allows you to mirror the specified area of the PNG file horizontally. A set of functions includes the functions of image reading, mirroring the image area, recording the image, as well as checking the data entered. The full description of the source code is given.

## СОДЕРЖАНИЕ

	Введение	5
1.	Теоретические сведения	6
2.	Функции и структуры для работы с PNG-файлами.	7
2.1.	Структура struct Png	7
2.2.	Функция считывания	7
2.3.	Функция записи	7
2.4.	Функция зеркального отражения области изображения	7
2.5.	Функция, проверяющая корректность имени файла	7
2.6.	Функция, проверяющая корректность координат	7
2.7.	Основная функция	8
3.	Тестирование программы	9
	Заключение	10
	Список использованных источников	11
	Приложение А. Тестовые случаи	12
	Приложение Б. Исходный код программы	14

## ВВЕДЕНИЕ

**Цель работы:** ознакомиться с библиотекой `png.h`, освоить на практике считывание и запись PNG-файла, научиться совершать простые операции над пикселями PNG-файла.

**Задание:** требуется написать программу, которая отражает заданную область PNG-файла по горизонтали (симметрия относительно горизонтальной оси). Результат сохраняет в новом файле.

Программа получает параметры из входного потока и должна проверить их корректность. Параметры:

- `input_file`
- `x0`
- `y0`
- `x1`
- `y1`

`input_file` - имя PNG файла

`x0 y0` левый верхний угол области (отсчет с точки 0, 0)

`x1 y1` правый нижний угол области

В случае, если программа получила некорректные параметры, то:

- не создается выходного файла
- выводится сообщение об ошибке “Fail with <имя параметра>”.

## 1.ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Файл с расширением png – файл изображения, хранящегося в Portable Network Graphic (PNG) формате. Как GIF файл, содержит битовую карту индексированных цветов с сжатием без потерь, но без ограничений авторского права, обычно используется для хранения графики для веб-изображений.

Изображения типа PNG это растровые изображения. Растровое изображение — изображение, представляющее собой сетку пикселей — цветных точек (обычно прямоугольных) на мониторе, бумаге и других отображающих устройствах.

Основные характеристики растрового изображения:

- Размер изображения в пикселях.
- Количество используемых цветов или глубина цвета.
- Цветовое пространство.
- Разрешение изображения.

## 2. ФУНКЦИИ И СТРУКТУРЫ ДЛЯ РАБОТЫ С PNG-ФАЙЛАМИ.

**2.1. Структура struct Png** - структура содержит поля `int width, height` — в данных переменных хранится размер изображения в пикселях по горизонтали и вертикали. Поле `png_byte color_type` хранит информацию о типе цвета в пикселе. Поле `png_byte` хранит количество бит на сэмпл. Поле `png_structp png_ptr` хранит данные png файла. Поле `png_infor info_ptr` хранит информацию о png файле. Поле `int number_of_passes` хранит количество проходов для считывания изображения. Поле `png_bytep* row_pointers` хранит строку пикселей.

**2.2. Функция считывания** - `int readPNG(char* filename, pngfile* image)`. Функция считывает изображение и все данные о нем из файла в структуру png с помощью функций из библиотеки `png.h`, проверяя успешность считывания на каждом шаге.

**2.3. Функция записи** - `int writePNG(char* filename, pngfile* image)`. Функция записывает изображение и все данные о нем из структуры png в файл с помощью функций из библиотеки `png.h`, проверяя успешность выполнения операций.

**2.4. Функция зеркального отражения области изображения** - `void reflectionOfTheArea(pngfile* image, int x0, int y0, int x1, int y1)`. Отражает заданную область зеркально по горизонтали, проходя по каждому пикселю из области.

**2.5. Функция, проверяющая корректность имени файла** - `int filenameCheck(char* filename)`. Использует библиотеку `regex.h` для проверки имени. В случае наличия сочетания «.png» в конце имени файла возвращает 1, иначе возвращает 0.

**2.6. Функция, проверяющая корректность координат** - `int coordsCheck(pngfile image, int x0, int y0, int x1, int y1)`. Сравнивает соответствующие координаты между собой и с размерами изображения. В случае неверных данных возвращает 0, иначе 1.

**2.7. Основная функция** - `main.c`, считывает данные, введенные пользователем, вызывает поочередно вышеперечисленные функции для выполнения поставленной задачи.



### **3. ТЕСТИРОВАНИЕ ПРОГРАММЫ**

Программа собрана в операционной системе Ubuntu 16.04 LTS с использованием компилятора gcc.

Тестовые случаи представлены в приложении А.

В ходе тестирования программы было выявлено, что программа работает корректно.

## **ЗАКЛЮЧЕНИЕ**

В ходе работы была реализована программа на языке Си для работы с PNG-файлом. Были написаны функции чтения, записи и обработки PNG-файла. Были освоены методы работы с PNG-файлами.

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. [https://en.wikipedia.org/wiki/Portable\\_Network\\_Graphics](https://en.wikipedia.org/wiki/Portable_Network_Graphics)
2. [http://refspecs.linuxbase.org/LSB\\_4.0.0/LSB-Desktop-generic/LSB-Desktop-generic/libpng12man.html](http://refspecs.linuxbase.org/LSB_4.0.0/LSB-Desktop-generic/LSB-Desktop-generic/libpng12man.html)
3. <http://www.libpng.org/pub/png/libpng-1.0.3-manual.html>

## ПРИЛОЖЕНИЕ А

### ТЕСТОВЫЕ СЛУЧАИ

Таблица 1. Проверка работы с корректными данными.







Введенные данные	Исходное изображение	Новое изображение
./main 1.png 100 200 300 400		
./main 2.png 0 50 350 400		
./main 3.png 50 50 250 150		

Таблица 2. Проверка работы с некорректными данными.

Ввод в терминал	Вывод в терминал	Комментарий
./main 4.png 0 0 150 150	File could not be opened	Файла 4.png в дирректории нет
./main 4.pg 0 0 150 150	Fail with input_file	Тип файла неверный
./main 3.png 50 50 250 0	Fail with y0 and y1	$y_0 > y_1$
./main 3.png 50 50 1000 300	Fail with x1	Высота файла меньше 1000
./main 3.png 50 50 1000 300 90	The number of entered parameters is incorrect	Последний параметр лишний

## ПРИЛОЖЕНИЕ Б

### ИСХОДНЫЙ КОД ПРОГРАММЫ

#### **main.c:**

```
#include <stdio.h>
#include <string.h>
#include <png.h>
#include <stdlib.h>
#include "png_func.h"

int main(int argc, char** argv)
{
    if(argc != 6)
    {
        printf("The number of entered parameters is incorrect\n");
        return 0;
    }
    char filename[FILE_NAME_LEN];
    strncpy(filename, argv[1], FILE_NAME_LEN);
    char result[256] = "result";
    strcat(result, filename, FILE_NAME_LEN);
    int x0 = atoi(argv[2]);
    int y0 = atoi(argv[3]);
    int x1 = atoi(argv[4]);
    int y1 = atoi(argv[5]);
    filename[strlen(filename)] = '\0';
    if(!filenameCheck(filename))
    {
        printf("Fail with input_file\n");
        return 0;
    }
    pngfile image;
    if(!readPNG(filename, &image))
        return 0;
    if(!coordsCheck(image, x0, y0, x1, y1))
        return 0;
    reflectionOfTheArea(&image, x0, y0, x1, y1);
    if(!writePNG(result, &image))
        return 0;
    return 0;
}
```

#### **png\_func.c:**

```

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

#include "png_func.h"

int readPNG(char* filename, pngfile* image)
{
    int x,y;
    char header[8];
    FILE* f = fopen(filename, "rb");
    if(!f)
    {
        printf("File could not be opened\n");
        return ERROR;
    }
    fread(header, 1, 8, f);
    if(png_sig_cmp(header, 0, 8))
    {
        printf("File is not recognized as a PNG\n");
        fclose(f);
        return ERROR;
    }
    image->png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING,
    NULL, NULL, NULL);

    if(!image->png_ptr)
    {
        printf("Allocation and initializion of a png_struct structure for reading
PNG file is failed\n");
        fclose(f);
        return ERROR;
    }

    image->info_ptr = png_create_info_struct(image->png_ptr);
    if(!image->info_ptr)
    {
        printf("Allocation and initializion of a png_info structure is failed\n");
        fclose(f);
        return ERROR;
    }
}

```

```

    if(setjmp(png_jmpbuf(image->png_ptr)))
    {
        printf("Initializion of input/output for the PNG file is failed\n");
        fclose(f);
        return ERROR;
    }

    png_init_io(image->png_ptr, f);

    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);

    if(setjmp(png_jmpbuf(image->png_ptr)))
    {
        printf("Reading of file is failed\n");
        fclose(f);
        return ERROR;
    }

    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(f);
    return 1;
}

int writePNG(char* filename, pngfile* image)

```



```

{
    int x,y;

    FILE *f = fopen(filename, "wb");
    if (!f)
    {
        printf("File could not be opened\n");
        return ERROR;
    }

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

    if (!image->png_ptr)
    {
        printf("Allocation and initialization of a png_struct structure for writing
    PNG file is failed\n");
        fclose(f);
        return ERROR;
    }

    image->info_ptr = png_create_info_struct(image->png_ptr);
    if (!image->info_ptr)
    {
        printf("Allocation and initialization of a png_info structure is failed\n");
        fclose(f);
        return ERROR;
    }

    if (setjmp(png_jmpbuf(image->png_ptr)))
    {
        printf("Initialization of input/output for the PNG file is failed\n");
        fclose(f);
        return ERROR;
    }

    png_init_io(image->png_ptr, f);

    if (setjmp(png_jmpbuf(image->png_ptr)))
    {
        printf("Error during writing header\n");
    }
}

```

```

        fclose(f);
        return ERROR;
    }

    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("Writing of image data is failed\n");
        fclose(f);
        return ERROR;
    }

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

    if (setjmp(png_jmpbuf(image->png_ptr)))
    {
        printf("Writing the end of a PNG file is failed\n");
        fclose(f);
        return ERROR;
    }

    png_write_end(image->png_ptr, NULL);

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

    fclose(f);
    return 1;
}

void reflectionOfTheArea(pngfile* image, int x0, int y0, int x1, int y1)
{
    int x,y;
    int ptr0[image->width][image->height];
    int ptr1[image->width][image->height];

```

```

int ptr2[image->width][image->height];
int ptr3[image->width][image->height];
for (y = y0; y <= y1; y++)
{
    png_byte *row = image->row_pointers[y];
    for (x = x0; x <= x1; x++)
    {
        png_byte *ptr = &(row[x * 4]);
        ptr0[x][y] = ptr[0];
        ptr1[x][y] = ptr[1];
        ptr2[x][y] = ptr[2];
        ptr3[x][y] = ptr[3];
    }
}
for (y = y0; y <= y1; y++)
{
    png_byte *row = image->row_pointers[y];
    for (x = x0; x <= x1; x++)
    {
        png_byte *ptr = &(row[x * 4]);
        ptr[0] = ptr0[x][y1 + y0 - y];
        ptr[1] = ptr1[x][y1 + y0 - y];
        ptr[2] = ptr2[x][y1 + y0 - y];
        ptr[3] = ptr3[x][y1 + y0 - y];
    }
}
}

int filenameCheck(char* filename)
{
    char* regexString = ".png$";
    regex_t regexComp;
    regcomp(&regexComp, regexString, REG_EXTENDED);
    return regexec(&regexComp, filename, 0, NULL, 0) == 0;
}

int coordsCheck(pngfile image, int x0, int y0, int x1, int y1)
{
    if(x0 < 0)
    {
        printf("Fail with x0\n");
        return ERROR;
    }
    if(y0 < 0)

```

```

    {
        printf("Fail with y0\n");
        return ERROR;
    }
    if(x1 > image.width)
    {
        printf("Fail with x1\n");
        return ERROR;
    }
    if(y1 > image.height)
    {
        printf("Fail with y1\n");
        return ERROR;
    }
    if(x0 > x1)
    {
        printf("Fail with x0 and x1\n");
        return ERROR;
    }
    if(y0 > y1)
    {
        printf("Fail with y0 and y1\n");
        return ERROR;
    }
    return 1;
}

```

**png\_func.h:**

#pragma once

```

#include <png.h>
#define FILE_NAME_LEN 250
#define ERROR 0

typedef 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;
}pngfile;

int readPNG(char* filename, pngfile* image);

int writePNG(char* filename, pngfile* image);

void reflectionOfTheArea(pngfile* image, int x0, int y0, int x1, int y1);

int filenameCheck(char* filename);

int coordsCheck(pngfile image, int x0, int y0, int x1, int y1);

```

### **Makefile:**

```

OBJ = png_func.o main.o

all: $(OBJ)
    gcc $(OBJ) -lpng -o main
%.o: %.c png_func.h
    gcc -c $<
clear:
    rm *.o main

```