Лабораторная работа № 1 Работа с растровыми изображениями	Студент	Думалаков И.В.
	Группа	ИВТ-463
	Дата	
	Преподаватель	
	Подпись преподавателя	

Цель работы: Разработать программу с использованием низкоуровневых библиотек декодирования изображений для трансформации изображения заданного формата.

Вариант: Трансформация изображения: поворот на 180 градусов; Формат изображения: PNG.

Ход работы:

Рисунок 1. Исходное изображение.

Рисунок 2. Поворот на 180 градусов.

Рисунок 3. Компрессия 0%.

Рисунок 4. Компрессия 60%.

Рисунок 5. Компрессия 90%.

Компрессия, %	Размер, МБ	Наличие видимых артефактов	Оригинальный размер, кб
90	1.88	Нет	123
60	0.122	Нет	
30	0.118	Нет	

Код программы:

main.cpp:

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include "lib.h"

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

    char* src = "inImg.png";
    char* dest0 = "out0.png";
    char* dest6 = "out6.png";
    char* dest9 = "out9.png";

    read_png_file(src);
    process_file();
    write_png_file(dest0, 0);
    write_png_file(dest6, 6);
    write_png_file(dest9, 9);

    return 0;
}
```

lib.cpp:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include <png.h>

#define PNG_DEBUG 3

extern "C" {
#include <png.h>
}

void abort_(const char* s, ...) {
    va_list args;
    va_start(args, s);
    vfprintf(stderr, s, args);
    fprintf(stderr, "\n");
    va_end(args);
    abort();
```

```
}

int x, y;

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;

void saveWithCompression(int compression) {
    png_set_IHDR(png_ptr, info_ptr, width, height,
        bit_depth, color_type, PNG_INTERLACE_NONE,
        PNG_COMPRESSION_TYPE_BASE, PNG_FILTER_TYPE_BASE);

    png_set_compression_level(png_ptr, compression);
    png_write_info(png_ptr, info_ptr);

    if (setjmp(png_jmpbuf(png_ptr)))
        abort_("[write_png_file] Error during writing bytes");

    png_write_image(png_ptr, row_pointers);

    if (setjmp(png_jmpbuf(png_ptr)))
        abort_("[write_png_file] Error during end of write");

    png_write_end(png_ptr, NULL);

}

void read_png_file(char* file_name) {
    char header[8];

    FILE* fp = fopen(file_name, "rb");
    if (!fp)
        abort_("[read_png_file] File %s could not be opened for reading", file_name);
    fread(header, 1, 8, fp);
    png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL, NULL, NULL);

    if (!png_ptr)
        abort_("[read_png_file] png_create_read_struct failed");

    info_ptr = png_create_info_struct(png_ptr);
    if (!info_ptr)
        abort_("[read_png_file] png_create_info_struct failed");

    if (setjmp(png_jmpbuf(png_ptr)))
        abort_("[read_png_file] Error during init_io");

    png_init_io(png_ptr, fp);
    png_set_sig_bytes(png_ptr, 8);

    png_read_info(png_ptr, info_ptr);

    width = png_get_image_width(png_ptr, info_ptr);
    height = png_get_image_height(png_ptr, info_ptr);
    color_type = png_get_color_type(png_ptr, info_ptr);
    bit_depth = png_get_bit_depth(png_ptr, info_ptr);

    number_of_passes = png_set_interlace_handling(png_ptr);
    png_read_update_info(png_ptr, info_ptr);
```

```
    if (setjmp(png_jmpbuf(png_ptr)))
        abort_("[read_png_file] Error during read_image");

    row_pointers = (png_bytep*)malloc(sizeof(png_bytep) * height);
    for (y = 0; y < height; y++)
        row_pointers[y] = (png_byte*)malloc(png_get_rowbytes(png_ptr, info_ptr));

    png_read_image(png_ptr, row_pointers);

    fclose(fp);

}

void write_png_file(char* file_name, int compression) {

    FILE* fp = fopen(file_name, "wb");
    if (!fp)
        abort_("[write_png_file] File %s could not be opened for writing", file_name);



    png_ptr = png_create_write_struct(PNG_LIBPNG_VER_STRING, NULL, NULL, NULL);

    if (!png_ptr)
        abort_("[write_png_file] png_create_write_struct failed");

    info_ptr = png_create_info_struct(png_ptr);
    if (!info_ptr)
        abort_("[write_png_file] png_create_info_struct failed");

    if (setjmp(png_jmpbuf(png_ptr)))
        abort_("[write_png_file] Error during init_io");

    png_init_io(png_ptr, fp);

    if (setjmp(png_jmpbuf(png_ptr)))
        abort_("[write_png_file] Error during writing header");


    saveWithCompression(compression);

}

void process_file(void) {

    int colorType = png_get_color_type(png_ptr, info_ptr);

    int colorCount = colorType == PNG_COLOR_TYPE_RGB ? 3 : 4;


    for (int y = 0; y < height / 2; y++) {
        for (int x = 0; x < width; x++) {
            for (int c = 0; c < colorCount; c++) {
                auto temp = row_pointers[height - y - 1][(width - x - 1) * colorCount + c];
                row_pointers[height - y - 1][(width - x - 1) * colorCount + c] =
row_pointers[y][x * colorCount + c];
                row_pointers[y][x * colorCount + c] = temp;
            }
        }
    }

}
```

lib.h:

```
#pragma once
#ifndef MYLIBPNG_LIB_H
#define MYLIBPNG_LIB_H
#define _CRT_SECURE_NO_WARNINGS
#include <string>

void read_png_file(char* file_name);
void saveWithCompression(int compression);
void write_png_file(char* file_name, int compression);
void process_file();

#endif
```

Ссылка на репозиторий:

https://github.com/SkifHungry/Multimedia_technology/tree/main/Lab1