### Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Лабораторная работа №6

«Низкоуровневое программирование»

Выполнил студент группы P33111

Ярошевский М.С.

Преподаватель Жирков И.О.

2020

Задание

Реализовать поворот картинки bmp на 90 градусов.

Код

bmp\_io.c  
#include "bmp\_io.h"

#include <stdlib.h>

#include <stdio.h>

BMP\_File \*readBMP(char \*filename) {

FILE \*fp = fopen(filename, "rb");

if (fp == NULL) {

printf("File %s is not opened", filename);

return NULL;

}

BitMapFileHeader fileHeader = {};

fread(&fileHeader, sizeof(BitMapFileHeader), 1, fp);

if (fileHeader.bfType != 0x4d42 && fileHeader.bfType != 0x4349 && fileHeader.bfType != 0x5450) {

printf("File %s is not BMP format", filename);

fclose(fp);

return NULL;

}

fseek(fp, 0, SEEK\_END);

long filesize = ftell(fp);

fseek(fp, sizeof(BitMapFileHeader), SEEK\_SET);

BitMapInfoHeader infoHeader = {};

fread(&infoHeader, sizeof(BitMapInfoHeader), 1, fp);

if(fileHeader.bfFileSize != filesize) {

printf("BMP file has invalid filesize field");

return NULL;

}

if(infoHeader.biSize != 40 && infoHeader.biSize != 108 && infoHeader.biSize != 124) {

printf("BMP file has unknown size info header");

fclose(fp);

return NULL;

}

if(infoHeader.biWidth < 1 || infoHeader.biWidth > 10000 ||

infoHeader.biHeight < 1 || infoHeader.biHeight > 10000) {

printf("BMP file has not supported width or height ( > 10000 or < 1)");

fclose(fp);

return NULL;

}

if (fileHeader.bfReserved1 != 0 || infoHeader.biPlanes != 1 ||

fileHeader.bfOffBits != 14 + infoHeader.biSize) {

printf("BMP file contains some errors in structure");

fclose(fp);

return NULL;

}

if (infoHeader.biBitCount != 24 || infoHeader.biCompression != 0) {

printf("BMP file is only full colored (24 bits) and not compressed");

fclose(fp);

return NULL;

}

fseek(fp, fileHeader.bfOffBits, SEEK\_SET);

RGBQuad \*pixels = malloc(infoHeader.biHeight \* infoHeader.biWidth \* sizeof(RGBQuad));

if (pixels == NULL) {

printf("Cannot create array of pixel");

fclose(fp);

exit(1);

}

int kr = (infoHeader.biWidth \* 3) % 4;

if (kr != 0) { kr = 4 - kr; }

for (size\_t i = 0; i < infoHeader.biHeight; i++) {

for (size\_t j = 0; j < infoHeader.biWidth; j++) {

size\_t index = i \* infoHeader.biWidth + j;

fread(&(pixels[index]), sizeof(RGBQuad), 1, fp);

}

for (int k = 0; k < kr; k++) { getc(fp); }

}

fclose(fp);

BMP\_File \*bmpFile = malloc(1 \* sizeof(BMP\_File));

bmpFile->fileHeader = fileHeader;

bmpFile->infoHeader = infoHeader;

bmpFile->rgbQuad = pixels;

return bmpFile;

}

void writeBMP(BMP\_File \*bmp, char \*newFilename) {

FILE \*fw = fopen(newFilename, "w+");

if (fw == NULL) {

printf("File %s is not opened", newFilename);

exit(1);

}

BitMapFileHeader fileHeader = bmp->fileHeader;

fwrite(&fileHeader, sizeof(BitMapFileHeader), 1, fw);

BitMapInfoHeader infoHeader = bmp->infoHeader;

fwrite(&infoHeader, sizeof(BitMapInfoHeader), 1, fw);

fseek(fw, fileHeader.bfOffBits, SEEK\_SET);

int kr = (infoHeader.biWidth \* 3) % 4;

if (kr != 0) { kr = 4 - kr; }

RGBQuad \*pixels = bmp->rgbQuad;

for (size\_t i = 0; i < infoHeader.biHeight; i++) {

for (size\_t j = 0; j < infoHeader.biWidth; j++) {

size\_t index = i \* infoHeader.biWidth + j;

fwrite(&(pixels[index]), sizeof(RGBQuad), 1, fw);

}

for (int k = 0; k < kr; k++) { putc(0, fw); }

}

fclose(fw);

}

bmp\_io.h

//

// Created by lanolin on 31.10.2020.

//

#ifndef LLP\_LAB\_6\_\_BMP\_IO\_H

#define LLP\_LAB\_6\_\_BMP\_IO\_H

#include "bmp\_struct.h"

/\*\*

\*

\* @return BMP\_File struct with all information about bmp file

\*/

BMP\_File\* readBMP(char\* filename);

/\*\*

\*

\* @return write changes in new file on filename

\*/

void writeBMP(BMP\_File \*bmp, char\* newFilename) ;

#endif

bmp\_struct.h

#ifndef LLP\_LAB\_6\_\_BMP\_STRUCT\_H

#define LLP\_LAB\_6\_\_BMP\_STRUCT\_H

#include <stdint.h>

typedef struct \_\_attribute\_\_((packed)){

uint16\_t bfType;

uint32\_t bfFileSize;

uint32\_t bfReserved1;

uint32\_t bfOffBits;

} BitMapFileHeader;

typedef struct \_\_attribute\_\_((packed)){

uint32\_t biSize;

int32\_t biWidth;

int32\_t biHeight;

uint16\_t biPlanes;

uint16\_t biBitCount;

uint32\_t biCompression;

uint32\_t biSizeImage;

uint32\_t biXPelsPerMeter;

uint32\_t biYPelsPerMeter;

uint32\_t biClrUsed;

uint32\_t biClrImportant;

} BitMapInfoHeader;

typedef struct \_\_attribute\_\_((packed)){

uint8\_t rgbBlue;

uint8\_t rgbGreen;

uint8\_t rgbRed;

} RGBQuad;

typedef struct {

BitMapFileHeader fileHeader;

BitMapInfoHeader infoHeader;

RGBQuad\* rgbQuad;

} BMP\_File;

#endif

bmp\_transform.c

#include "bmp\_transform.h"

#include <stdio.h>

#include <stdlib.h>

#include <memory.h>

void printInformationBMP(BMP\_File \*bmpFile) {

BitMapFileHeader header = bmpFile->fileHeader;

printf("File Header: \n");

printf("\tSignature %d (0x%x)\n", header.bfType, header.bfType);

printf("\tFileSize %d (0x%x)\n", header.bfFileSize, header.bfFileSize);

printf("\tReserved\_1 %d (0x%x)\n", header.bfReserved1, header.bfReserved1);

printf("\tDataOffset %d (0x%x)\n", header.bfOffBits, header.bfOffBits);

#ifdef BIT\_MAP\_INFO\_HEADER\_PRINT\_INF

BitMapInfoHeader info = bmpFile->infoHeader;

printf("Info Header: \n");

printf("\tSize %d (0x%x)\n", info.biSize, info.biSize);

printf("\tWidth %u (0x%x)\n", info.biWidth, info.biWidth);

printf("\tHeight %u (0x%x)\n", info.biHeight, info.biHeight);

printf("\tPlanes %d (0x%x)\n", info.biPlanes, info.biPlanes);

printf("\tBits Per Pixel %d (0x%x)\n", info.biBitCount, info.biBitCount);

printf("\tCompression %d (0x%x)\n", info.biCompression, info.biCompression);

printf("\tImageSize %d (0x%x)\n", info.biSizeImage, info.biSizeImage);

printf("\tXpixelsPerM %u (0x%x)\n", info.biXPelsPerMeter, info.biXPelsPerMeter);

printf("\tYpixelsPerM %u (0x%x)\n", info.biYPelsPerMeter, info.biYPelsPerMeter);

printf("\tColors Used %u (0x%x)\n", info.biClrUsed, info.biClrUsed);

printf("\tImportant Colors %d (0x%x)\n", info.biClrImportant, info.biClrImportant);

#endif

#ifdef BIT\_MAP\_PIXEL\_PRINT\_INF

for(long i = 0; i < bmpFile->infoHeader.biWidth; i++) {

printf("Pixel %ld: r=%d g=%d b=%d \n", i,

bmpFile->rgbQuad[i].rgbRed,

bmpFile->rgbQuad[i].rgbGreen,

bmpFile->rgbQuad[i].rgbBlue

);

}

#endif

}

BMP\_File\* rotate90Clockwise(BMP\_File \*bmpFile) {

int width = bmpFile->infoHeader.biWidth;

int height = bmpFile->infoHeader.biHeight;

BMP\_File \*new\_bmp = copyEmptyBMP(bmpFile);

new\_bmp->infoHeader.biWidth = height;

new\_bmp->infoHeader.biHeight = width;

for(int y = new\_bmp->infoHeader.biHeight; y >= 0 ; y--) {

for(int x = 0; x < new\_bmp->infoHeader.biWidth; x++) {

RGBQuad \*newPix = getPixel(new\_bmp, x, y);

RGBQuad \*oldPix = getPixel(bmpFile, width - y - 1, x);

memcpy(newPix, oldPix, sizeof(RGBQuad));

}

}

return new\_bmp;

}

BMP\_File\* rotate180(BMP\_File \*bmpFile) {

int width = bmpFile->infoHeader.biWidth;

int height = bmpFile->infoHeader.biHeight;

BMP\_File \*new\_bmp = copyEmptyBMP(bmpFile);

new\_bmp->infoHeader.biWidth = width;

new\_bmp->infoHeader.biHeight = height;

for(int y = new\_bmp->infoHeader.biHeight; y >= 0 ; y--) {

for(int x = 0; x < new\_bmp->infoHeader.biWidth; x++) {

RGBQuad \*newPix = getPixel(new\_bmp, x, y);

RGBQuad \*oldPix = getPixel(bmpFile, width - x - 1, height - y - 1);

memcpy(newPix, oldPix, sizeof(RGBQuad));

}

}

return new\_bmp;

}

BMP\_File\* rotate90CounterClockwise(BMP\_File \*bmpFile) {

int width = bmpFile->infoHeader.biWidth;

int height = bmpFile->infoHeader.biHeight;

BMP\_File \*new\_bmp = copyEmptyBMP(bmpFile);

new\_bmp->infoHeader.biWidth = height;

new\_bmp->infoHeader.biHeight = width;

for(int y = new\_bmp->infoHeader.biHeight-1; y >= 0 ; y--) {

for(int x = 0; x < new\_bmp->infoHeader.biWidth; x++) {

RGBQuad \*newPix = getPixel(new\_bmp, x, y);

RGBQuad \*oldPix = getPixel(bmpFile, y, height - x - 1);

memcpy(newPix, oldPix, sizeof(RGBQuad));

}

}

return new\_bmp;

}

RGBQuad\* getPixel(BMP\_File \*bmp, int x, int y) {

size\_t index = bmp->infoHeader.biWidth \* y + x;

return &(bmp->rgbQuad[index]);

}

BMP\_File\* copyEmptyBMP(BMP\_File \*bmpFile) {

BMP\_File \*newBMP = malloc(1 \* sizeof(BMP\_File));

newBMP->fileHeader = bmpFile->fileHeader;

newBMP->infoHeader = bmpFile->infoHeader;

RGBQuad \*pixels = malloc(newBMP->infoHeader.biHeight \* newBMP->infoHeader.biWidth \* sizeof(RGBQuad));

newBMP->rgbQuad = pixels;

return newBMP;

}

BMP\_File\* rotate(BMP\_File \*bmpFile, int angelOfDegrees) {

if(angelOfDegrees == 0) {

return bmpFile;

}else if(angelOfDegrees < 0) {

angelOfDegrees = -((-angelOfDegrees) % 360);

}else {

angelOfDegrees = angelOfDegrees % 360;

}

if(angelOfDegrees == 90 || angelOfDegrees == -270) {

return rotate90Clockwise(bmpFile);

}else if(angelOfDegrees == 180 || angelOfDegrees == -180) {

return rotate180(bmpFile);

}else if (angelOfDegrees == 270 || angelOfDegrees == -90) {

return rotate90CounterClockwise(bmpFile);

}else {

printf("ERROR: Angel degrees must be equals +/- 90 or +/- 180 or +/- 270");

return NULL;

}

}

bmp\_transform.h

#ifndef LLP\_LAB\_6\_\_BMP\_TRANSFORM\_H

#define LLP\_LAB\_6\_\_BMP\_TRANSFORM\_H

#include "bmp\_struct.h"

void printInformationBMP(BMP\_File \*bmp);

BMP\_File\* rotate(BMP\_File \*bmpFile, int angelOfDegrees);

RGBQuad\* getPixel(BMP\_File \*bmp, int x, int y);

BMP\_File\* copyEmptyBMP(BMP\_File \*bmpFile);

#endif

bmp.h

#ifndef LLP\_LAB\_6\_\_BMP\_H

#define LLP\_LAB\_6\_\_BMP\_H

#include "bmp\_io.h"

#include "bmp\_struct.h"

#include "bmp\_transform.h"

#endif

main.c

#include <stdlib.h>

#include "lib/bmp\_io.h"

#include "lib/bmp\_transform.h"

int main() {

char\* filename = "/home/maximyarosh/lab6/llp-lab-6-master/src/RS01.bmp";

BMP\_File\* bmpFile = readBMP(filename);

if(bmpFile == NULL) {

exit(1);

}

char\* newFilename\_1 = "/home/maximyarosh/lab6/llp-lab-6-master/src/RS01\_1.bmp";

BMP\_File\* newBMP\_1 = rotate(bmpFile, 90);

writeBMP(newBMP\_1, newFilename\_1);

free(newBMP\_1);

char\* newFilename\_2 = "/home/maximyarosh/lab6/llp-lab-6-master/src/RS01\_2.bmp";

BMP\_File\* newBMP\_2 = rotate(bmpFile, 180);

writeBMP(newBMP\_2, newFilename\_2);

free(newBMP\_2);

char\* newFilename\_3 = "/home/maximyarosh/lab6/llp-lab-6-master/src/RS01\_3.bmp";

BMP\_File\* newBMP\_3 = rotate(bmpFile, 270);

writeBMP(newBMP\_3, newFilename\_3);

free(newBMP\_3);

free(bmpFile);

return 0;

}

Вывод

В процессе выполнения данной лабораторной работы вертел картинки bmp.