/\*\*

\* copy.c

\*

\* Computer Science 50

\* Problem Set 4

\*

\* Copies a BMP piece by piece, just because.

\*/

#include <stdio.h>

#include <stdlib.h>

#include "bmp.h"

int main(int argc, char\* argv[])

{

// ensure proper usage

if (argc != 4)

{

printf("Usage: ./copy n infile outfile\n");

return 1;

}

// remember filenames

char\* n = argv[1];

char\* infile = argv[2];

char\* outfile = argv[3];

// Проверка, что 0 < n < 100

if (atoi(n) > 100 || atoi(n) < 1)

{

printf("n should be from 1 to n\n");

return 1;

}

// open input file

FILE\* inptr = fopen(infile, "r");

if (inptr == NULL)

{

printf("Could not open %s.\n", infile);

return 2;

}

// open output file

FILE\* outptr = fopen(outfile, "w");

if (outptr == NULL)

{

fclose(inptr);

fprintf(stderr, "Could not create %s.\n", outfile);

return 3;

}

// read infile's BITMAPFILEHEADER

BITMAPFILEHEADER bf;

fread(&bf, sizeof(BITMAPFILEHEADER), 1, inptr);

// read infile's BITMAPINFOHEADER

BITMAPINFOHEADER bi;

fread(&bi, sizeof(BITMAPINFOHEADER), 1, inptr);

//информация из старого файла

int biWidth\_old = bi.biWidth;

int biHeight\_old = bi.biHeight;

int padding\_old = (4 - (bi.biWidth \* sizeof(RGBTRIPLE)) % 4) % 4;

printf("%i,%i,%i,%i,%i\n", bf.bfSize, bi.biSizeImage, biWidth\_old, biHeight\_old, padding\_old);

//информация в новый файл

bi.biWidth \*= atoi(n);

int padding\_new = (4 - (bi.biWidth \* sizeof(RGBTRIPLE)) % 4) % 4;

bi.biHeight \*= atoi(n);

bi.biSizeImage = ((sizeof(RGBTRIPLE) \* bi.biWidth) + padding\_new) \* abs(bi.biHeight);

bf.bfSize = bi.biSizeImage + sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER);

printf("%i,%i,%i,%i,%i\n", bf.bfSize, bi.biSizeImage, bi.biWidth, bi.biHeight, padding\_new);

// ensure infile is (likely) a 24-bit uncompressed BMP 4.0

if (bf.bfType != 0x4d42 || bf.bfOffBits != 54 || bi.biSize != 40 ||

bi.biBitCount != 24 || bi.biCompression != 0)

{

fclose(outptr);

fclose(inptr);

fprintf(stderr, "Unsupported file format.\n");

return 4;

}

// write outfile's BITMAPFILEHEADER

fwrite(&bf, sizeof(BITMAPFILEHEADER), 1, outptr);

// write outfile's BITMAPINFOHEADER

fwrite(&bi, sizeof(BITMAPINFOHEADER), 1, outptr);

for (int i = 0, biHeight = abs(biHeight\_old); i < biHeight; i++)

{

for (int p = 0; p < atoi(n)-1; p++)

{

// растягиваем по горизонтали

for (int j = 0; j < biWidth\_old; j++)

{

RGBTRIPLE triple;

// read RGB triple from infile

fread(&triple, sizeof(RGBTRIPLE), 1, inptr);

// write RGB triple to outfile n times

for (int k = 0; k < atoi(n); k++)

{

fwrite(&triple, sizeof(RGBTRIPLE), 1, outptr);

}

}

for (int k = 0; k < padding\_new; k++)

{

fputc(0x00, outptr);

}

fseek(inptr, -(biWidth\_old)\*sizeof(RGBTRIPLE), SEEK\_CUR);

}

for (int j = 0; j < biWidth\_old; j++)

{

RGBTRIPLE triple;

// read RGB triple from infile

fread(&triple, sizeof(RGBTRIPLE), 1, inptr);

// write RGB triple to outfile n times

for (int k = 0; k < atoi(n); k++)

{

fwrite(&triple, sizeof(RGBTRIPLE), 1, outptr);

}

}

for (int k = 0; k < padding\_new; k++)

{

fputc(0x00, outptr);

}

fseek(inptr, padding\_old, SEEK\_CUR);

}

// close infile

fclose(inptr);

// close outfile

fclose(outptr);

// that's all folks

return 0;

}