COP4338

Fall 2016

Caryl Rahn

## Assignment #2: BMP Image Processing (due by 11:55 PM on 11/16)

Through this programming assignment, the students will learn to do the following:

1. Learn to read and write binary files.
2. Gain more experience with using pointers.
3. Gain more experience with dynamic memory allocation and deallocation.

In this assignment, you are asked to manipulate image from a 24-bit uncompressed bmp file. (The format of bmp files is given in http://en.wikipedia.org/wiki/BMP\_file\_format). To help with reading and writing the image file, you are given library functions and an example that deal with the bmp file (in a zip file on my faculty website). In the example, the image is flipped horizontally. You are asked to implement the enlarge() and rotate() functions: the former is used to enlarge the image by an integer scale factor; the latter is to rotate the image either clockwise or counter-clockwise by a certain degree (which must be a multiple of 90) depending whether the rotation degree is positive or negative.

The program should take the follow command-line options:

% bmptool [-s scale | -r degree | -f ] [-o output\_file] [input\_file]

Where -s means to scale the image by the given 'scale' (a positive integer: 1, 2, 3, ...), -r means to rotate the image by 'degree' (which must be a multiple of 90, clockwise if positive, counter-clockwise if negative), and -f means to flip the image horizontally. You can assume for each type (-s, -r, or -f), the command line has at most one option. That is, the user is not supposed to do something like "./bmptool -s 2 -s 4". If that happens, you can either prompt error, or take either 2 or 4 as the scale. However, the user may present a combination, say, './bmptool -r 90 -s 4'. If multiple option types are present, the order for processing the image is that you do scale first, then rotate, and then flip.

You are required to use getopt() to process the command-line. If '-o output\_file' is missing, use standard output. If 'input\_file' is missing, use standard input. The options to use standard input or standard output will only be used when chaining commands.

Make sure the program returns 0 on success. In that case, one can 'chain' the commands using pipes, like:

% bmptool -s 4 1.bmp | bmptool -r -90 | bmptool -f -o 2.bmp

Your program needs to provide necessary sanity-check for command line arguments and handle various error conditions and prompt the user with helpful information. You need to use getopt() to process the command line arguments. Test your program with various combinations to make sure it works as expected. You must use dynamic memory to store the content of the new image before writing out to file. You need to reclaim memory afterwards to prevent from memory leaks.

Please submit your work through moodle as one zip file. Follow the instructions below carefully (to avoid unnecessary loss of grade):

To start, first create a directory for this homework and name it FirstnameLastnameA2 (of course, you'd use your real name here).  You should place the source code and the Makefile in the directory. One should be able to create the executable by simply 'make'. The Makefile should also contain a 'clean' target for cleaning up the directory (removing all temporary files, object files and executable files). Make sure you don't include intermediate files: \*.o, executables, \*~, etc., in your submission. (There'll be a penalty for including unnecessary intermediate files).

Please make sure you submit homework before the deadline.

NOTE THAT ALL HOMEWORK IS INDIVIDUAL WORK. NO CODE SHARING AND CO-DEVELOPMENT IS EVER ALLOWED. REFER TO CODE OF CONDUCT STATED IN SYLLABUS. ZERO TOLERANCE FOR ETHIC VIOLATIONS IN THIS COURSE.