MP 1 Image Manipulation

Due: ~~9/1 @ 11:59 PM~~ **Tuesday, September 8 at 11:59 PM**
Doxygen for MP 1

Lab machines

We will be grading the MPs for this course using the EWS Linux machines. Any errors that you run into as a result of developing your code on other machines are your own responsibility.

No SVN

If you do not have an account on the svn server, you can still work on this MP by downloading the source files from mp1.tar.gz.

Goals and Overview

In this MP (machine problem) you will:

- learn about the course programming environment
- learn to use a C++ library designed for image manipulation
- learn about pointers

In addition, you'll get more chance to mess with some of the things in C++ identical to those in Java — namely loops and conditionals — so that you can see for yourself that they work the same way.

The Assignment, Part 1: Reading and Understanding the Policies

Read and understand the following information:

- Course Info
- Collaboration Policy
- Coding Style Policy

Important Information

The information in these links is relevant to not just this MP, but every future MP as well, and you **will** be held responsible for having read and understood all of the information. So, part of this MP assignment is to read over this part 1 material, and to gain a complete understanding of it before moving on to part 2 of this assignment. This means that you should ASK QUESTIONS ON ANY OF THIS PART 1 MATERIAL THAT YOU DO NOT UNDERSTAND.

The Assignment, Part 2: Tutorials

Linux Environment

If you have not read through and/or watched the material we have on Linux and Vim, and are generally not comfortable with either, please go to the Resources page and review those tutorials.

SVN Reference

If you have not already done so, please review the SVN Reference. We will be using SVN as our source code control mechanism throughout the course.

Makefile Tutorial

Go through the Makefile Tutorial before you begin the assignment, to familiarize yourself with the idea of Makefiles and how to use them. You will be using them in part 3 of this assignment.

Make sure you check out your subversion directory before doing the Makefile Tutorial.

The Assignment, Part 3: Coding

This portion of the MP has two requirements. You will first define a pixel class called `RGBAPixel` that is used by the `PNG` class to store the data associated with the pixels in an image. In addition you will write a bit of client code that modifies a `PNG`.

Checking Out the Code

To check out the provided code for mp1 from the class repository, go to the directory you want your work to be downloaded to and type the following into the command console:

```
TERMINAL
svn co https://subversion.ews.illinois.edu/svn/fa15-cs225/NETID/ cs225
```

where `NETID` is your University NetID. If you’re on the VM or you’re own machine it will ask you to login as your local username first. Just hit enter, and when it prompts you for a username / password give it your NetID / AD password.

This will make a directory called `cs225` which has all your course code in it. To get new assignments you just have to run `svn up` from that directory.

In there will be a directory called `mp1` and it will contain the following:

- The files `png.cpp` and `png.h`, from the EasyPNG image library.
- Some PNG image files: `in_01.png`, `out_01.png`, `in_02.png`, `out_02.png`, `in_03.png`, and `out_03.png`. If you are physically on one of the lab machines, you can view a PNG file by opening it in Mozilla Firefox, or another application as you prefer. Otherwise, if you are accessing remotely, it may be easier to just transfer the files to your own system and open them with a viewer of your choice.

Loading Clang

If you're on EWS you'll need to make sure that you can access Clang. Run the following command:

```
TERMINAL
source /class/cs225/setup
```

Clang

We are using Clang as our compiler / linker this semester and libc++ as our standard library. See the Makefile Tutorial or the lab_intro `Makefile` for what compiler / linker flags you need.

Background: PNG image handling

In this assignment, you will be loading and saving images stored in the PNG image file format. You won't need to know the details of how an PNG image is stored in a file; instead, you will rely on the EasyPNG code that we have provided for you. A reference for it is provided for you in Doxygen form.

Please read the Doxygen documentation. It will explain what functions are available for you to use for this MP.

If you still have questions about how to use EasyPNG, post on Piazza or ask course staff after you've read the Doxygen documentation.

Problem statement

1. Create a class called `RGBAPixel` whose functionality is described in this Doxygen documentation. Take care to note that all the members of the class are public. Following convention, the class definition should be placed in a file called `rgbapixel.h`, and the member function implementations should be placed in a file called `rgbapixel.cpp`.

2. Write a program (the function `main`), contained in file `main.cpp`, that uses the functions from the EasyPNG library to open a bitmap image named `in.png`, rotate the image by 180 degrees, and write the resulting image to the file `out.png`. You should assume that

in.png is located in the working directory (i.e., the directory from which your program was run) and place out.png there as well. The output image should have the exact same dimensions as the input image (i.e. the two images should be of the exact same width and height). You may use the given files in_01.png, out_01.png, in_02.png, out_02.png, in_03.png, and out_03.png to test your program. First, copy in_01.png to in.png by typing

```
TERMINAL
cp in_01.png in.png
```

at the command line on a lab system. Once your program has run, type

```
TERMINAL
diff out.png out_01.png
```

to compare your program's output to out_01.png, which is the output file produced by the solution program written by the course staff. If both files are the same, your program is likely running correctly. You may test your program a second time on the other pairs of images in the same manner. Your program will be graded using a similar procedure on a series of other bitmap images. Although we don't provide you with these images, you are welcome to test your program on additional images of your own, looking at each output image to see whether it is a perfect 180 degree rotation of the corresponding input image.

Compilation

You need to write your own Makefile for this MP, and hand it in with your code:

- We will be compiling your code with the Makefile you hand in, and if we can't do so, you'll end up with something close to zero for your MP grade — i.e. if your Makefile doesn't work, that will be treated the same as if your code had a compiler error.
- The name of the executable file that your main produces must be mp1 — that is, to run your program, you should have to type a '.', then a '/', then a lowercase 'm', then a lowercase 'p', then '1', then hit return. Do not name your executable MP1, Mp1, mp1.out, mp1.exe, or anything else, other than the above-mentioned mp1. If your executable is not named correctly, that will be treated the same as if your code had a compile-time error, namely, your grade will be something close to a zero.
- Your Makefile should compile together your own solution file, namely main.cpp along with the given files png.cpp and png.h. Do not have typos in your file names or your Makefile!! For example, make sure your Makefile compiles a file named main.cpp, not a file named main.C or test.cpp or any other such thing. Also, make sure your Makefile does not compile extra files that are not part of this MP. For example, do not add in files from the Makefile tutorial by mistake; the only files the Makefile should be dealing with are the few listed in the paragraph above.
- You need to make sure your filenames and function names are correct in order for your code to compile.

Output Formatting Issues

We have provided your output format in the MP specification above, and you need to match that *exactly*. Testing will be done by comparing your output PNG image file to our own (using the diff utility). If your image file does not match ours, that is considered incorrect output; you do not get partial credit for getting close. So we recommend you use the tools you learned in part 1 of this MP. Furthermore, it is advisable to confine yourself to the use of the EasyPNG library functions listed above when manipulating images, just to be safe; just because two images look the same doesn't mean they really are.

Implementation Notes

- You will be responsible for testing your code before handing it in. Using the three test images we have provided to check your program, while not required, is strongly recommended.

Handing in your code

To facilitate anonymous grading, **do not** include any personally-identifiable information (like your name, your UIN, or your NetID) in any of your source files. Instead, before you hand in this assignment, create a file called partners.txt that contains only the NetIDs of people in your collaboration group (if it exists), one per line. If you worked alone, include only your own NetID in this file. We will be automatically processing this information, so do not include anything else in the file. (If we must manually correct your submission, you may lose points.) As always, if you're working in a group, each group member must hand in the assignment. (Failure to cite collaborators violates our academic integrity policy; we will be aggressively pursuing such violators.)

We will be using Subversion as our hand-in system this semester. Our grading system will checkout your most recent (pre-deadline) commit for grading. Therefore, to hand in your code, all you have to do is commit it to your Subversion repository.

Be sure your working directory is the `mp1` folder that was created when you checked out the code. To hand in your code, you first need to add the new files you created to the working copy of your repository by typing:

```
TERMINAL
svn add rgbapixel.h
svn add rgbapixel.cpp
svn add main.cpp
svn add Makefile
svn add partners.txt
```

To commit your changes to the repository type:

```
TERMINAL
svn ci -m "mp1 submission"
```

The ci command tells SVN that you want to commit your changes to the repository.

The `-m` flag tells SVN that the quoted message following it will be a description of the changes you made. This is not a requirement, but it is a good practice to add comments to your revisions. In the event of needing to revert to previous revision, the comments will help you differentiate between the revisions (and in group projects, it can help identify who made the changes).

Grading Information

The following files are used to grade mp1:

- `rgbapixel.h`
- `rgbapixel.cpp`
- `main.cpp`
- `Makefile`
- `partners.txt`

All other files will not be used for grading.

Video FAQ

- What's the deal with `operator()`?
- Pointers on Pointers

Good luck!