MIT EECS 6.815/6.865: Assignment 0:

# Introduction to C++ and the 6.815/6.865 Image Class

Due Wednesday February 11 at 9pm

## 1 Summary

- Learning C++
- Compiling, Debugging and Submitting Code
- C++ (Source/Header, Static Typing, Text I/O, Classes)
- The Image Class
- Brightness and Contrast

## 2 Installation

Throughout this course, you will develop a C++ library for computational photography that you can use in future projects. Like Java, C++ is an object-oriented programming language. Its syntax is pretty similar to both C and Java. For those familiar with Java, http://www.cprogramming.com/java/c-and-c++-for-java-programmers.html can be a great resource. We also suggest using http://www.cplusplus.com/. C++ is one of the most widely used programming languages in the world. Therefore, if you come across an error it is very likely that someone else has already encountered this problem. Just Googling your errors can often be the best first step in debugging.

C++ is a compiled language, which means C++ code must be compiled before it can be executed. In this section, we will go over several compilers you might use. Regardless of which compiler you choose to use, your code must compile on the online submission system.

### 2.1 Compilers

The code you write should be portable enough to compile in any compiler. The submission system will use `g++` on Linux. You can use this compiler on either your own machine running Linux (to install on Ubuntu: sudo apt-get install build-essential) or use machines in the Athena cluster to compile your code. If you have `g++` installed, you can compile your code by going to the code directory and typing

```
make
```

on the command line. This will compile the starter code into an executable called `a0`. You can run this by typing

```
make run
```

If you are successful, you should get a message that says
`Congratulations, you have compiled the starter code!`

You can use the same setup on Mac OS by installing command line tools either through Xcode (Preferences → Downloads) or the terminal (http://osxdaily.com/2014/02/12/install-command-line-tools-mac-os-x/). The process may be different for different versions of OS X; similar tutorials can be easily found online. You can use Cygwin on Windows to create a unix-like shell and use the same compilation setup (https://www.cygwin.com/). The alternative is using Visual Studio (https://www.dreamspark.com/Student/Default.aspx) on Windows. We can try to help you with compilation problems in office hours.

1) Make sure you can compile the starter code without problems. There is nothing to turn in for this problem, but if you can't compile the code, you will be unlikely to complete the remainder of the assignment.

## 2.2 Submission System

The online submission system will compile your code on our servers, execute it and then display the output. All text written to standard out and standard error is printed. Any png images written to the `asst/Output` directory will be displayed by the submission system, so you can verify that your code is working through the system. Keep all of your code in the `asst` directory of the zip file you submit. The submission system will ignore files outside of this directory.

## 2.3 Debugging

You are welcome to use a debugger, such as the one that comes with your IDE. However, we also suggest you use assert statements or print statements to make sure conditions you think are true are actually true. For example, if you are performing a division, you may want to assert that the divisor is not zero.

```
float safeDivision(float dividend, float divisor) {
  assert(divisor!=0, "Divisor is zero");
  return dividend/divisor;
}
```

If the divisor is zero, the program will abort and tell you which assertion failed. Otherwise, the returned value would be `Inf` or `NaN`, which might be undesired.

### 2.4 Image Input/Output

Our Image class supports reading PNG files only. All the sampled images we give you will be in the PNG format and you can use one of a handful of tools to convert your own images to this format (e.g. http://image.online-convert.com/convert-to-png). In addition, the Image class can only write to PNG files. You can use your favorite image viewer to view the images.

## 3 C++

In this section, we will introduce a few C++ language features that are different from previous languages you may have used. C++ is most similar to Java. The big differences are that C++ has explicit memory management, distinguishes between references and pointers and organizes code into header and source files. You can find more information in this C++ tutorial for Java programmers (http://www.cprogramming.com/java/c-and-c++-for-java-programmers.html) and this reference website (http://www.cplusplus.com/).

### 3.1 Headers and Source Files

C++ programs are usually organized into header and source files. Actual executable code is in source files, while function and class declarations are in headers. Open the attached `a0.h` and `a0.cpp` files in a text editor or IDE of your choice. The function

```
void helloworld(float a, float b);
```

is in both the header file (`a0.h`) and the source file (`a0.cpp`). We will give you key function definitions in the header file, and you will implement them in the source file.

When you compile and execute your code, the program will run all commands in the `main` function located in `a0_main.cpp`. We will not grade the contents of this function and will replace it by our own to run our unit tests, but it will allow you to execute your own functions and verify that your code is correct.

### 3.2 Static Typing

All C++ variables must have a type. In 6.815/6.865, we will use IEEE single-precision floats to represent the value of a pixel.

2a) Declare a floating point variable `c` that is the sum of `a` and `b` in the function `void helloworld(float a, float b)`.

## 3.3 Text Input/Output

You may find it useful to print values to screen for the purposes of debugging or getting information about what you are working on. You can do this using the syntax

```
cout << "Hello World!" << endl;
```

You can also use `cout` to display variables using the same syntax.

2b) Write statements in `void helloworld(float a, float b)` that prints the following

```
Hello World!
The value of a is _.
The value of b is _.
The sum of a and b is _.
```

where the underscores are replaced by the actual numbers that make the sentences true.

## 3.4 Classes and Functions

We have provided the specification for a class to represent Images (`Image.h`). The specification contains a number of methods and variables that belong to each instantiation of the Image class. Most of these methods are implemented in the source file (`Image.cpp`).

In a later part of this pset, you are going to implement some of the definitions in the source file `Image.cpp`. For now, just look over the header file and look for the two constructors on lines 18 and 21 of `Image.h`. They are

```
Image(int width_, int height_ = 0, int channels_ = 0,
   const std::string &name="");
Image(const std::string & filename);
```

You can create an instance of the Image class using either constructor. The first creates a blank image of dimensions `width_` $\times$ `height_` $\times$ `channels_`. You can use the first constructor to create an Image variable `my_im` that is $100 \times 100$ pixels with three color channels by typing

```
Image my_im(100,100,3);
```

3) Implement the function
`Image readAnImage(const std::string &filename)` in `a0.cpp`, which returns an Image created using the second constructor taking `filename` as an input.

# 4 The Image Class

The Image class specification is given in `Image.h`. Images are three dimensional arrays (width by height by color channel) that store pixels as a vector of floats called `image_data`. In memory, the pixels in the image are stored sequentially in row-major order in three adjacent color planes. That is, the distance or *stride* between adjacent values in the same row in `image_data` is equal to 1. The stride between adjacent values in the same column is equal to the width of the image and the stride between the different color channels at the same pixel is width×height. You can use the method `int stride(int dimension)` to compute these values.

`image_data` is a C++ vector, which is essentially an array which manages its own memory. You can access elements of it using brackets. For instance, `image_data[0]` returns the front element in the vector. More information about vectors can be found at http://www.cplusplus.com/reference/vector/vector/.

For images that correspond to pictures, the floating points in `image_data` will be between 0 and 1. There is nothing guaranteeing that the values of `image_data` stay in this range and there may be times when you want to use the Image class to store intermediate data that doesn't correspond to pictures, in which case the range is not meaningful. When the image is written to a file, it will assume the data lies in this range and will round values outside of the range to one of the endpoints.

You can use the Image `write` method to write images to png files. For example,

```
my_im.write("./my_image.png");
```

Then, you can view them in your favorite image viewer. This may be useful for debugging. Alternatively, if you don't feel like providing a filename, you can use `my_im.debug_write()` to write an image to an automatically named file. This might make debugging easier

## 4.1 Pixel Accessors and Setters

You are going to implement the accessor and setter operators for pixel values in the image. We adopt the convention that the elements are accessed via the () operator. This is contrary to C++ convention, but will allow us to match the syntax of Halide (http://halide-lang.org/), which we will use at the end of

the semester to write fast image processing code. That is, the pixel at location $(x, y)$ in the third color channel of `my_im` is accessed via

```
my_im(x,y,2);
```

The third color channel is accessed via the number 2, not 3. That is because we want to use 0-indexing, in which the first element in a given index is specified by 0.

Implement three accessors with 0-indexing and bounds checking to make sure the input is valid. If the input is not valid, throw an exception using the command `throw OutOfBoundsException();`.

In `Image.cpp` implement:

4a) `number_of_elements()` : returns the number of elements in the image. An RGB (3 color channel) image of $100 \times 100$ pixels has 30000 elements.

4b) `my_im(x)` : returns the value of `image_data` at location $x$ as long as $x$ is less than ($<$) the total number of pixels and at least ($>=$) 0. You need to implement the functions
`const float & operator()(int x) const`
`float & operator()(int x)`
with identical code. Use `number_of_elements()` for bounds checking.

4c) `my_im(x,y,c)` : returns the value at location $(x, y)$ in the $c$th color channel. You need to implement the functions
`const float & operator()(int x, int y, int z) const`
`float & operator()(int x, int y, int z)`
with identical code. Use `width()`, `height()` and `channels()` for bounds checking.

4d) `my_im(x,y)` : returns the value at location $(x, y)$ in the 0th color channel. You need to implement the functions
`const float & operator()(int x, int y) const`
`float & operator()(int x, int y)` with identical code.

# 5 Brightness and Contrast

Now for the fun part. Once we have these accessors, we can perform simple operations like increasing the brightness or contrast of an image.

5a) Implement the function `brightness` in `a0.cpp`, which multiplies the pixels in an image by the value `factor`. Make sure to create a new image and return that rather than modifying the input image. This is good practice in case you want to use the input again.

5b) Implement the function `contrast`, which increases the contrast of an image around a specified `midpoint` by the value `factor`. That is, you should apply the following function to every pixel's value

$$I_{out} = \texttt{factor} \times (I_{in} - \texttt{midpoint}) + \texttt{midpoint}.$$

**Extra:** Try handling the values going out of the range 0 and 1 in a sensible way. You can test your functions on the included `Input/Boston_low_contrast.png`. This input image has very low contrast. You can use your `contrast` function to increase it.

# 6 Submission

Turn in your files to the online submission system (link is on Stellar) and make sure all your files are in the `asst` directory under the root of the zip file. If your code compiles on the submission system, it is organized correctly. The submission system will run code in your main function, but we will not use this code for grading.

In the submission system, there will be a form in which you should answer the following questions:

- How long did the assignment take?
- Potential issues with your solution and explanation of partial completion (for partial credit)
- Any extra credit you may have implemented
- Collaboration acknowledgment (you must write your own code)
- What was most unclear/difficult?
- What was most exciting?