EECS 398 System Design in C++
Winter 2019 Stylesheet

Coding standards

I'll be reading your code, so I'm going to ask that you make it easy for me by following my style, heavily-influenced by my experience at Microsoft.

1. Every file should have a header at the top explaining what it does and its history, with dated entries listing who did what to the code.
2. You may use either tabs or spaces but tabs and indents are 3 spaces.
3. Please try to keep your lines to no more than 80 characters. You may line-break anywhere, but pick sensibly and indent the remainder of the line by 2 indent levels = 6 spaces.
4. Each new statement starts on a new line.
5. The nested statements inside an if/else, a for statement, and similar control structures should be indented by 3 spaces on the next line. The idiom of chaining else if on one line with no additional indenting is disallowed.
6. When using { and } around a block of code or a class definition, the braces go on separate lines, indented 3 spaces, lined up with the code they contain. I dislike extra braces around a single statement.
7. When using { and } around a class definition, public and private keywords should line up with the braces. Everything else is indented one more level.
8. When using { and } around a list on a single line, there is one space after { and one before }.
9. There is one space after `(` or `[` and one before] or `)`. One space between a keyword and a (, otherwise none.
10. I dislike extra parenthesis that add nothing to the readability.
11. There is one space before and after `+ - = += -= :` ?, etc.
12. Prefix operators `* & ++ --` have one space before and none after.
13. Postix operators `++ --` have no spaces before and one after, except when followed by a semicolon.
14. There are two spaces after `;`.
15. There are no spaces around `. -> ::`
16. I do not need an RME list but I do need useful comments with a clear explanation of how your code works.

17. I dislike SHOUTING and most use of underscores. Please avoid them anywhere in your code, except as needed, e.g., as defined by the operating system functions or API.
18. I dislike functions that stretch to more than about 100 LOC (lines of code).
19. I prefer //-style comments.
20. Variable and function names should be in either camelCase or PascalCase. camelCase names start with a lower-case letter (to help you remember, camel is an ordinary noun) and should be used for things that are local or private. PascalCase names start with an upper-case letter (Pascal was a man's name) and are used for things that public or more global.
21. Names should be unambiguous in their meaning (e.g., LightIsOn, not LightOnOrOff). Words should be spelled out. Non-standard abbreviations should not be used, e.g., elements[], not elmts[]. Common, well-known acronyms may be used. One and two letters acronyms can be all caps; otherwise, initial cap, then lower case.
22. Character string literals are disallowed in the body of your code. Literals should be defined in a table or a header and referred to symbolically so as to accommodate possible translation.
23. All your code should support Unicode or UTF-8 in filenames, paths, and content.