

Style Rules for CS216

1. **Documenting your program**. Document your program *as you write it*. For the most part this documentation should be naturally occurring, in that you have made use of good variable and function names, const values, etc in writing your code. This sort of documentation is a big win in that it: will never become inconsistent with the code (it IS the code!) and keeps the amount of material that must be read to a minimum.

To aid the graders of your homework, however, often it is useful to include comments that describe a bit more of what you are doing. Code from the CS201 text (C++ Program Design, Cohoon and Davidson) and our texts (although sometimes I think code in our text could use more comments) are reasonable examples. In particular, if you use an algorithm that is tricky, it is useful to add comments explaining what you are doing. For all but the simplest of functions I would expect to see a comment or two.

For the sanity of the TAs, to help reunite any unstapled pages, and for your own future reference, please be sure to include at the top of each file the following: the name of the file, your name, lab section #, Assignment #, and the date.

```
// complex.h - interface to Complex number data type.  
// Elmer Fudd  
// CS216 – Lab Section #4  
// Lab #3  
// 2/3/2002
```

2. **Consistency and Readability**. Be consistent in indentation and documentation throughout your program. If you are adding to existing code, then your additions should be in harmony with the existing code.

Indent the body of the if-else, switch, for, while, and other nested statements to articulate the logical flow of your program. Use blank lines to separate and group logically related sections of your program.

Use const as much as possible; but not excessively. Leave a space after punctuation marks: semicolons, colons, and commas. Leave space before and after the binary operators =, <, <=, ==, +, -, *, /, but not period or arrow and other unary operators.

Avoid writing lines so long (> 80 characters) that they will not be visible on a printout. At the very least you will lose style points for this, in the worst case the TA will not be able to figure out what your code is doing.

In general, functions should be no longer than one page. If they are longer than this then you should think about breaking them up into multiple functions.

The CS201 Style guide offers more specific advice on formatting your code.