Hardwin Bui

Nick Henderson

Ryan McCrory

Ryan Watkins

Kenneth Yang

Style Guide

Coding Style to Follow to read easily

1. Variable names, function names, class names, etc., must be sensible, and reflect the purpose.
2. Loop-counter variables are the only variables that may be a single letter, but avoid excessively long variable names too
3. Avoid excessive and uninformative comments: “set x to 0” is not helpful; instead, use comments to explain what the code aims to achieve and, if necessary, why. If you code is so complex that you must explain how it works, consider breaking it into simpler components.
4. Encapsulate or use local variables instead. Imagine creating an implementation of a hash table that uses a global variable for the table itself
5. Variables must be declared in the smallest scope possible, and must be declared as close to their use in the function as possible
6. Compile your code with warnings enabled.
7. “Magic numbers” in your code must be specified as named constants, not as bare numbers. These are numbers that have a meaning, occur in one more more locations in the code, and which could be replaced by a named constant to clarify the code and prevent errors later
8. Commit your code through git regularly, and in small, logical chunks
9. Keep consistent colors and fonts
10. Standard headers for different modules:
    a. For better understanding and maintenance of the code, the header of different modules should follow some standard format and information.

11. Proper indentation is very important to increase the readability of the code. For making the code readable, programmers should use White spaces properly
    a. There must be a space after giving a comma between two function arguments.
    b. Each nested block should be properly indented and spaced.
12. Each variable should be given a descriptive and meaningful name indicating the reason behind using it. This is not possible if an identifier is used for multiple purposes and thus it can lead to confusion to the reader. Moreover, it leads to more difficulty during future enhancements.
13. Length of functions should not be very large
14. Your comments should not explain step by step what the code is doing. Your excellent variable/method/class naming skills will take care of that.
15. Every string should be a reference to a string resource, an enum or a constant
16. Check Values From Unreliable Sources
17. Always Write the Braces Around Single Line Blocks
18. It is considered good practice to keep the lengths of source code lines at or below 80 characters
19. Program statements should be limited to one per line. Also, nested statements should be avoided when possible.
20. It is better to use parentheses liberally. Even in cases where operator precedence unambiguously dictates the order of evaluation of an expression, often it is beneficial from a readability point of view to include parentheses anyway