Coding standards for C Programming

1. Follow the **latest rules in the C Standard compiler documentation**
2. **Use logical variable names to avoid any confusion.**
3. Secure code is by definition clean and error-free. Security in your code has to be pervasive. That means it has to be in the forefront when you design your program, when you write it and when you test it.
4. Use **strncpy/strncat/fgets/snprintf** instead of strcpy/strcat/gets/sprintf (the latter bunch *will* get used for buffer overrun attacks, believe me).
5. The proper use of **escape sequence**s like \t or \n improves the readability of your code. Be liberal while using them as proper indentation is a good programming practice. Instead of simply using white spaces, it is better to use escape sequences as they help in displaying the output with standard formatting.
6. Use clear names for variables and functions (create your own naming style or follow someone else’s, if you like)
7. Make use of functions whenever code seems too long and the task we need to perform is operation specific and multiple
8. Don’t use too much shorthand. C allows you to write things compact. Very compact. For example: i++, ++i, i += 1, i = i + 1 all do (more or less) the same thing, but the most compact way isn’t always the clearest depending on the context. Likewise, don’t get fancy with constructs like hisFunction() && herFunction(); when a simple if-statement does the same.
9. Write some friggin’ comments, thank you very much! Unbelievable how often I had to dig through code trying to follow the original programmer(s) train of thought(s). If only they had added a line of comment or two…
10. Learn that small things matter. For example, write if (0 == value) instead of if (value == 0). This causes the compiler to complain if (no, I mean *when*) you accidentally forget one =
11. Don’t write too much redundant code. Shove it in an function (or a macro) instead.
12. Header files contain extern, functions files contain static (and not vice versa). For everything in C: make the scope as small as humanly possible. No, we don’t want an int i; in file scope because we “use i as a counter everywhere anyway”!
13. Watch out when explicitly casting from anything but void \*. Your compiler may not warn you and the bug hard to find.
14. Check all compiler warnings and make sure that they’re harmless (if needed, boost up that warning level, people!)
15. Always remember: the end goal is not to write clean code. It’s to ship a product which is secure and does what the user wants