Coding Style

Bill Marjerison ([bill.marjerison@computools-jp.com](mailto:bill.marjerison@computools-jp.com))

August, 2024

# Overview

Coding style depends on the programming language being used and personal preferences for that particular language, and ultimiately by the rules or preferences of the organization paying for the project.

If you have some flexibility in the style, I suggest you look at some existing rules. Some rule sets include:

* Chromium https://chromium.googlesource.com/chromium/src/+/HEAD/styleguide/styleguide.md
* Google https://google.github.io/styleguide/
* LLVM https://llvm.org/docs/CodingStandards.html
* Mozilla Firefox https://firefox-source-docs.mozilla.org/code-quality/coding-style/index.html
* WebKit <https://webkit.org/code-style-guidelines/>

If you are using Visual Studio, you can configure a specifc style set including some of those listed above at Tools 🡪 Settings 🡪 Text Editor 🡪 C++ (or other language) 🡪 Code Style 🡪 Formatting. When configured, the editor and formatter will automatically apply the appropriate rules such as indentation and alignment.

These rule sets deal with:

* Item naming.
* Indentation and brace alignment.
* Use of tab characters (in most cases you should use spaces instead).
* Comments.
* Dos and don’ts for the project or the organization.

The entire point of having coding rules is to have clean, consistent code so that anyone looking at the code now or in the future can read and navigate the code with a minium of fuss.

# My General Rules

My basic coding rules for my own projects include:

* I generally use Pascal-style item naming for class and function names where the first letter of each word is capitalized. There are exceptions depending on the language (especially Java) and some database code.
* Do not economize on item names. If you can understand what a variable does by its name, you need fewer comments.
* Code in alphabetical order so a reader can easily find something. An exception is to put constructors and destructors near the top.
* Include a space between an if, do, or while and the control list.
* Align else statements with the if.
* Do not use tab characters.
* In SQL, always capitalize the SQL keywords. For field names, check the database configuration because in some environments you can use mixed cases, but in others you might need to match the case exactly (in which case a correct name but wrong case will blow up your code).
* Comment as you go, and keep the comments up to date.
* Include system and library files before project files.
* Use “#if 0 . . .#endif” to temporarily comment out blocks of code.