**Coding Guidelines**

**#1 – Indent your code**

The indentation is something most people don’t care about, but it actually makes your code much easier to read, to understand and to maintain. PLEASE indent your code ☺. You should use either the [K&R style](https://en.wikipedia.org/wiki/Indent_style#K.26R_style) or the [Allman style](https://en.wikipedia.org/wiki/Indent_style#Allman_style), but be consistent. And avoid mixing tab and spaces, as some editors handle them differently and your code will look messed up.

**#2 – Write in English**

The main reason is that not everyone is able to speak your language. This is not specific to this project; in general it’s a good habit to write your code entirely in English. And this includes the documentation (*especially* the documentation).

**#3 – Class members are private**

And they start with “m\_”.

In every OOP language, the class members should not be accessible from other classes, except (possibly) children. The reason is simple: imagine you have some data stored as a Matrix, set as public. Now you want (e.g. for performance reasons) to switch to a 1D array. But everyone in your team accesses directly your Matrix, so if you change it, everything will have to be rewritten. Instead, if your Matrix is private, people access it through a getter “getData()” and if you decide to use a 1D array, you just have to change the “getData()” function, which will convert your Matrix in a 1D array: the change will thus be completely invisible for everyone except you. So class members are ALWAYS PRIVATE (or maybe protected).

In C++, it’s a good habit to prefix your class members with “m\_”, as nothing differentiates them from normal / global variables.

**#4 – The headers contain only declarations**

You should not write plain code in the headers. The reason is, if your header is included several times, your code will be compiled several times, thus resulting in extra memory usage and performance losses. For a small project of a dozen of files, it’s not important but for bigger projects it could be critical.

**#5 – Avoid defines and global variables**

In C++, you have the possibility to declare typed constants in your classes, so do it ☺. The defines are not typed so they should not be used to declare constants and the global variables are a bad habit in general. The constants should be defined are public static const.

**#6 – Use const keyword whenever possible**

This keyword is often forgotten, although it is very powerful. When a method does not modify the class data (typically a getter), use const at the end: void\* MyClass::getData() const. When a function needs (returns) a pointer/reference to a piece of data which is not (should not be) modified, declare a const: const int\* getInfo(const char\* fixedStr, char\* normalStr). Here the fixedStr cannot be modified in the function, and the returned integer cannot be modified by the caller. However, the normalStr can be modified in the function. A pointer/reference declared as const prevents the modification of the data it points to, thus avoiding mistakes / unwanted modifications.

**#7 – Organize your declarations**

Don’t hide your constructor in the middle of all your methods. Generally, we use the following order: Structures/enums, then static stuff, then variables, then constructors/destructors, then getters/setters, then other methods. The public always goes before the private.

**#8 – Use CamelCase**

Naming your objects as follows: anObjectWhichIsWellWritten is a wide spread convention on all programming languages, so it’s a good habit to follow it. The classes, the stand-alone functions, the structures and the enums should start with an uppercase letter while the variables and the class members should start with a lowercase letter. For the class methods, we found both conventions, so choose one (but stay consistent). The constants should be written only with uppercase letters.

**#9 – Don’t write giant statements / functions**

It’s literally impossible to understand/check a condition when it is split over three lines with && and || everywhere and enough parenthesis to build a parachute. Instead, you can use intermediate variables, functions, macros, you have the choice. Similarly, a function with more than 200 lines is very difficult to read, it is generally much better to split it into sub functions.

**#10 – Don’t copy and paste**

If you have to copy and paste more than three consecutive lines, then create a function / method. It will probably be much easier to read after that, and maybe you will use this function somewhere else later.

**#11 – Additional remarks**

1. Don’t implement empty destructors. If you don’t need it, then don’t create it ☺
2. Put the function parameter names in the declaration as well. Think to the other programmers which only have access to your function declaration, the name of the variable can be very informative.
3. Don’t hesitate to put spaces everywhere to make your code more readable. Especially in mathematic formulas.
4. Don’t hesitate to use line breaks to make the structure of your code clearer. But don’t exaggerate…
5. Avoid multiple statements on the same line. Generally, it makes your code more difficult to read.
6. Don’t flood the console with 1000 log messages / sec. It’s useless and prevents other team members to use the console to do their own debug ^^”