C++ coding style

1. Document project with **doxygen** comments. Use `@tag` notation. Provide at least brief description for every file, class, public method and instance variable.

2. Use `#include "filename"` for own files and `#include <filename>` for library files.
3. Do not put more than one class definition into a header file.
4. Minimize `#includes` in header files with the proper use of declarations: it is sufficient to have only a declaration of class or struct when it is used in a header only by pointer or reference. It is implementation file which `#includes` actual definitions of used classes.

5. Do not use global variables. Use objects or structures with minimal appropriate scope of visibility.
6. Minimize usage of free functions, use static class methods instead.
7. Do not use `#define` for code definitions (use `inline` functions) or constants definitions (use enumerations or constants).

8. Organize definitions of classes and structs in the following order: `public`, `protected`, `private`.
9. Do not use `public`/`protected` instance or class variables. Use properties (`private` variable + `get`Variable and `set`Variable methods). When the only purpose of your "class" is grouping of the variables, use keyword `struct`.
10. Do not write implementation of a method in a class definition. For short methods (or the situations when it is required, e.g. template definitions), you may put implementation in the header file after the class definition, using keyword `inline`. Place non-inline / long method implementations in the respective cpp file.
11. Use specifier `const` (placed after name and parameters list, before opening curly bracket) for instance methods that do not modify instance variables.
12. In implementation, place all specifiers together with the return type of a function/method on a separate line.

13. Name instance variables and instance methods starting with a lowercase letter, class and struct names starting with a capital letter. If name consists of several words, write them together with the first letter of each word capitalized. Write names of constants (`enum`, but also `const`) all-capitalized with the words separated by an underscore. Do not start names with an underscore.
14. Add a prefix `p` to names of pointer variables, follow it by a capitalized letter of the first word of variable's name.

15. Use `const` for instance variables, method parameters and local variables which are not changed after initializations.
16. Do not use numerical values in code, wrap them into respective constants and use their symbolic name.
17. Initialize every variable before use.

18. Write opening and closing curly brackets for each loop and conditional operator, even when body consists of one operator or is empty.
19. Write opening bracket for a block on the same line, closing bracket on a separate line. Intend all lines of the block's body with one tab relative to the block preceding operator.
20. For very long methods, class definitions and namespaces, provide a comment for closing curly bracket telling what it is closing.

21. If you feel that you are coding/seeing the same thing for the second time, stop and think if it is possible to extract the required behavior and implement it in a reusable way.