CODE STYLE

Content

1. CPP version 2
2. Headers 2
3. Local variables 2
4. Casting 2
5. Preincrement and predecrement 3
6. Use of constexpr 3
7. Naming rules 3
8. File names 3
9. Type names 3
10. Variable names 3
11. Gloabal variable names 4
12. Constant names 4
13. Function names 4
14. Enumeration names 4
15. Macro names 4
16. Comments 5

1. CPP version

Code should target C++17, i.e., should not use C++2x features. Do not use non-standard extension by your compiler.

2. Headers

Every single .cpp file should have an associated .h file. All header files should have a header guard and include all other headers it needs. Prefer placing definition for templates and inline functions in the same file as their declarations. All declaring constructs must have definition in associated .cpp file.

Header guard implemented by #define statement. The format if symbol name is _<FILENAME>_H_. For header file common.h it will be looks like this:

```
#ifndef _COMMON_H_
#define _COMMON_H_
…
#endif // _COMMON_H_
```

3. Local variables

Place a function's variables in the narrowest scope possible, and initialize variables in the declaration:

```
size_t size = get_size();
std::vector<int> v = {1,2,3};
```

4. Casting

Use C++-style casts like static_cast<float>(double_value), or brace initialization for conversion of arithmetic types like int64 y = int64{1} << 42. Do not use cast formats like (int)x unless the cast is to void.

5. Preincrement and predecrement

Use the prefix form (++i) of the increment and decrement operators unless you absolutely need postfix semantics.

6. Use of constexpr

Use constexpr to define true constants or to ensure constant initialization.

7. Naming rules

Optimize for readability using names that would be clear even to other people especially your comrades. Use names that describe the purpose or intent of the object. Minimize the use of abbreviations that would likely be unknown to a new reader. Do not abbreviate by deleting letters within a word. Generally speaking, descriptiveness should be proportional to the name's scope of visibility.

8. File names

File names should be all lowercase and can include underscores (_). C++ files should end in .ccp and header files should end in .h.

9. Type names

Type names start with a capital letter and have a capital letter for each new word, with no underscores: MyExcitingClass, MyExcitingEnum.

10. Variable names

The names of variables (including function parameters) and data members are all lowercase, with underscores between words. Data members of classes (but not structs) additionally have trailing underscores. For instance: a_local_variable, a_struct_data_member, a_class_data_member_.

11. Gloabal variable names

Global variables are named with a leading ‘g’ followed by mixed case: gMyGlobalVar.

12. Constant names

Global variables declared as constexpr or const, whose value is fixed for the duration of the program, are named with a leading "c" followed by mixed case: cMyConstant.

13. Function names

Regular functions have mixed case: MyFunction()

14. Enumeration names

Enumerators (for both scoped and unscoped enums) should be named like constants, not like macros. Use the C++style of enumerations:

```
enum class MyEnum
{
cOK = 0,
cError = 1,
….
}
```

15. Macro names

In general macros should *not* be used. However, if they are absolutely needed, then they should be named with all capitals and underscores #define MY_MACRO

16. Comments

Write comments only in English! When writing your comments, write for your audience: the next contributor who will need to understand your code. Giving sensible names to types and variables is much better than using obscure names that you must then explain through comments. Avoid useless comments like “it is magic”.