Coding Guide Lines for HiFlow

Philipp Gerstner

August 8, 2021

Contents

1 Naming Conventions

1.1 Classes, functions and variables

1. class names: ThisIsAClassName
2. function names: this_is_a_function_name
3. member variables get a underscore: i_am_a_member_variable_
4. integer variables and functions returning a number of something: nb_XY

2 Optimization

2.1 Function Calls

1. avoid call-by-value for non-built-in types. Use reference or pointers instead: void functionXY(int a, T& b) or void functionXY(int a, T* b) instead of void functionXY(int a, T b)
2. use const for reference input variables, that are not changed inside the function: void functionXY(int a, const T& b) instead of void functionXY(int a, T& b)
3. avoid return values for non-built-in types in performance critical parts. Instead, pass output object by reference: void functionXY(int a, T& b) instead of T functionXY(int a)

2.2 Memory Allocation

Dynamic memory allocation is expensive. Mind this fact every time you use new or initializing a std::vector, or other STL containers.

1. try to avoid dynamic memory allocation inside loops. Instead, initialize the variables outside the loop and call clear.

3 Class Design

3.1 Constructors, destructors

1. Don't use non-virtual destructors in base classes
2. Don't use virtual functions in constructors of base classes

3.2 Public, protected, private

1. data members should be private (also not protected)

3.3 Inheritance

1. In almost all cases, public inheritance is the way to go
2. Use virtual inheritance only, if multiple inheritance is used

4 C++11 and beyond Features

4.1 auto

4.2 Pointers

1. use Smart pointers like std::unique_ptr, std::shared_ptr, std::weak_ptr when dynamically allocint memory through new

2. use nullptr instead of NULL or 0 for setting a pointer to zero