C++编码规范

目录

1. 头文件
2. 作用域
3. 类
4. 其他
5. 命名约定

六、注释

七、格式

一、头文件

1、#define的保护

所有头文件都应该使用#define防止头文件被多重包含（multiple inclusion），命名格式当是： <PROJECT> \_ <PATH> \_ <FILE> \_H\_

为保证唯一性，头文件的命名应基于其所在项目源代码树的全路径。例如，项目 foo 中的头 文件 foo/src/bar/baz.h 按如下方式保护：

#ifndef FOO\_BAR\_BAZ\_H\_

#define FOO\_BAR\_BAZ\_H\_ ...

#endif // FOO\_BAR\_BAZ\_H\_

2、头文件依赖

使用前置声明（forward declarations）尽量减少.h 文件中#include 的数量。

当一个头文件被包含的同时也引入了一项新的依赖（dependency），只要该头文件被修改 ， 代码就要重新编译。如果你的头文件包含了其他头文件，这些头文件的任何改变也将导致那 些包含了你的头文件的代码重新编译。因此，我们宁可尽量少包含头文件，尤其是那些包含 在其他头文件中的。

使用前置声明可以显著减少需要包含的头文件数量。举例说明：头文件中用到类 File，但不 需要访问 File 的声明，则头文件中只需前置声明 class File;无需#include"file/base/file.h"。

3、函数参数顺序（Function Parameter Ordering）

定义函数时，参数顺序为：输入参数在前，输出参数在后。

C/C++函数参数分为输入参数和输出参数两种，有时输入参数也会输出（译者注：值被修 改时）。输入参数一般传值或常数引用（const references），输出参数或输入/输出参数 为非常数指针（non-const pointers）。对参数排序时，将所有输入参数置于输出参数之 前。不要仅仅因为是新添加的参数，就将其置于最后，而应该依然置于输出参数之前。

这一点并不是必须遵循的规则，输入/输出两用参数（通常是类/结构体变量）混在其中，会 使得规则难以遵循。

4、包含文件的名称及次序

将包含次序标准化可增强可读性、避免隐藏依赖, ，次序如下：C 库、C++库、其他库的.h、项目内的.h。

项目内头文件应按照项目源代码目录树结构排列，并且避免使用 UNIX 文件路径.

二、作用域

1、命名空间

在.cc 文件中，提倡使用不具名的命名空间，使用具名命名空间时，其名称可 基于项目或路径名称，不要使用 using 指示符。

定义：命名空间将全局作用域细分为不同的、具名的作用域，可有效防止全局作用域的命名冲突。

不能在.h 文件中使用不具名命名空间。

不要声明命名空间 std 下的任何内容，包括标准库类的前置声明。声明 std 下的实体会导 致不明确的行为，如，不可移植。声明标准库下的实体，需要包含对应的头文件。

2、嵌套类

当公开嵌套类作为接口的一部分时，虽然可以直接将他们保持在全局作用域中，但将嵌套类 的声明置于命名空间中是更好的选择。

3、非成员函数、静态成员函数和全局函数

使用命名空间中的非成员函数或静态成员函数，尽量不要使用全局函数。

4、局部变量

将函数变量尽可能置于最小作用域内，在声明变量时将其初始化。

5、全局变量

class 类型的全局变量是被禁止的，内建类型的全局变量是允许的，当然多线程代码中非常 数全局变量也是被禁止的。永远不要使用函数返回值初始化全局变量。

三、类

1、构造函数的职责

构造函数中只进行那些没有实际意义的）初始化，可能的话 ， 使用 Init()方法集中初始化为有意义的（non-trivial）数据。

2、默认构造函数

如果一个类定义了若干成员变量又没有其他构造函数，需要定义一个默认构造函数，否则编 译器将自动生产默认构造函数。

3、明确的构造函数

通常，只有一个参数的构造函数可被用于转换。

4、拷贝构造函数

仅在代码中需要拷贝一个类对象的时候使用拷贝构造函数；不需要拷贝时应使用 DISALLOW\_COPY\_AND\_ASSIGN。

定义：通过拷贝新建对象时可使用拷贝构造函数（特别是对象的传值时）。

5、结构体和类

仅当只有数据时使用 struct，其它一概使用 class。

struct 被用在仅包含数据的消极对象（passive objects）上，可能包括有关联的常量，但 没有存取数据成员之外的函数功能，而存取功能通过直接访问实现而无需方法调用，这儿提 到的方法是指只用于处理数据成员的，如构造函数、析构函数、Initialize()、Reset()、 Validate()。

如果需要更多的函数功能，class 更适合，如果不确定的话，直接使用 class。

6、继承

使用组合通常比使用继承更适宜，如果使用继承的话，只使用公共继承。

7、多重继承

真正需要用到多重实现继承）的时候非常少，只有当最多一个基类中含有实现，其他基类都是以 Interface 为后缀的纯接口类时才会使用多重继承。

只有当所有超类（superclass）除第一个外都是纯接口时才能使用多重继承。为确 保它们是纯接口，这些类必须以 Interface 为后缀。

8、接口

接口类不能被直接实例化，因为它声明了纯虚函数。为确保接口类的所有实现可被正确销毁 ， 必须为之声明虚析构函数。

9、操作符重载

一般不要重载操作符，尤其是赋值操作（operator=）比较阴险，应避免重载。如果需要 的话，可以定义类似 Equals()、CopyFrom()等函数。

10、存取控制

将数据成员私有化，并提供相关存取函数，如定义变量 foo\_及取值函数 foo()、赋值函数 set\_foo()。

11、声明次序

在类中使用特定的声明次序：public:在 private:之前，成员函数在数据成员（变量）前。

定义次序如下：public:、protected:、private:，如果那一块没有，直接忽略即可。

每一块中，声明次序一般如下：

1) typedefs 和 enums；

2) 常量；

3) 构造函数；

4) 析构函数；

5) 成员函数，含静态成员函数；

6) 数据成员，含静态数据成员。

宏 DISALLOW\_COPY\_AND\_ASSIGN 置于 private:块之后，作为类的最后部分。参考拷 贝构造函数。

.cc 文件中函数的定义应尽可能和声明次序一致。

12、编写短小函数

倾向于选择短小、凝练的函数。

长函数有时是恰当的，因此对于函数长度并没有严格限制。如果函数超过 40 行，可以考虑 在不影响程序结构的情况下将其分割一下。

四、其他

1、引用参数

按引用传递的参数必须加上 const。

2、函数重载

仅在输入参数类型不同、功能相同时使用重载函数（含构造函数），不要使用函数重载模仿 缺省函数参数。

3、缺省函数

禁止使用缺省函数

4、异常

不要使用C++异常

5、运行时类型识别

禁止使用RTTI

6、类型转换

使用static\_cast<>()等C++的类型转换，不要使用int y = (int)x 或int y = int(x);。

7、流

只能在记录日志时使用流。

8、const的使用

强烈建议在任何可以使用的情况下都要使用const

9、整形

C++内建整形中，唯一用到的是int,如果程序中需要不同大小的变量，可以使用<stdint.h>中得精确宽度的整形，如int16\_t。

10、0和NULL

整数用0，实数用0.0，指针用NULL，字符（串）用’\0’。

11、sizeof

尽可能用sizeof(vername)代替sizeof(type)。

五、命名约定

1、通用命名规则

函数命名，变量命名，文件命名应具有描述性，不要过度缩写，类型和变量应该都是名词，函数名可以用‘命令性’动词。

尽可能给出描述性名称，不要节约空间，让别人很快理解你的代码更重要。

除非放到项目外也非常明了，否则不要使用缩写。

不要省略字母的缩写。

2、文件命名

文件名要全部小写，可以包含下划线（\_）或短线(-)，按项目约定来。

3、类型命名

类型命名每个单词以大写字母开头，不包含下划线。

所有类型命名——类、结构体、类型定义、枚举——使用相同约定。

4、变量命名

变量名一律小写，单词间以下划线相连，类的成员变量以下划线结尾。

5、常量命名

在名称前加k。所有编译时常量和其他变量保持些区别，k后接大写字母开头的单词。

6、函数命名

普通函数大小写混合，存取函数则要求与变量名匹配。

普通函数函数名以大写字母开头，每个单词首字母大写，没有下划线。

存取函数要与存取的变量名匹配。

7、命名空间

命名空间的名称是全小写的，其命名基于项目名称和目录结构。

1. 注释
2. 注释风格

使用//或/\* \*/，统一就好。

1. 文件注释

在每个文件开头加入版权公告，然后是文件内容描述。

1. 类注释

每个类的定义要附着类的功能和用法的注释。

1. 函数注释

函数声明出注释描述函数功能，定义出描述函数实现。

1. 变量注释

通常变领命本身足以很好说明变量用途，特定情况下，需要额外注释说明。

1. 实现注释

对于实现代码中巧妙的、晦涩的、有趣的、重要的地方加以注释。

1. 标点、拼写和语法

留着标点、拼写和语法，写的好的注释比差的要易读的多。

1. 格式
2. 行长度

每一行代码字符数不超过80。

1. 非ASCII字符

尽量不使用非ASCII字符，使用时必须使用UTF-8格式。

1. 空格还是制表位

只使用空格，每次缩进2个空格。

1. 函数声明与定义

返回类型和函数名在同一行，合适的话，参数也放在同一行。

1. 函数调用

尽量放在同一行，否则，将实参封装在圆括号中。

1. 条件语句

更提倡不在圆括号中添加空格，关键字else另起一行。

1. 循环和开关选择语句

Switch语句可以使用大括号分块：空循环体应使用{}或continue

1. 指针和引用表达式

句点（.）或箭头（->）前面不要有空格，指针/地址操作符后不要有空格。

1. 函数返回值

Return表达式中不要使用圆括号。

1. 变量及数组初始化

选择=还是()(。

1. 类格式

声明属性依次序是public、protected、private,每次缩进一个空格

1. 初始化列表

构造函数初始化列表放在同一行或按四格缩进并排几行。

1. 命名空间格式化

命名空间内容不缩进。

1. 水平留白

水平留白的使用因地制宜，不要在行尾添加无谓的留白。

1. 垂直留白

垂直留白越少越好。