**头文件**

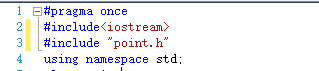
**函数参数顺序**

　　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（注：值被修改时）。输入参数一般传值或常数引用（const references），输出参数戒输入/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，会使得规则难以遵守。

**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（hidden dependencies，注：隐藏依赖主要是指包含的文件编译），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.（当前目录）和..（父目录）。



**作用域**

**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。

**C++类**

**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(non-trivial)的数据。

**拷贝构造函数**

　　仅在代码中需要拷贝一个类的对象的时候使用拷贝构造函数，不需要拷贝时使用DISALLOW\_COPY\_AND\_ASSIGN这个宏（关于这个宏的内容，可以在网上搜到，我这里就不写了）。C++中对象的隐式拷贝是导致很多性能问题和bugs的根源。拷贝构造函数降低了代码可读性，相比按引用传递，跟踪按值传递的对象更加困难，对象修改的地方变得难以捉摸。

**继承**

　　虽然C++的继承很好用，但是在实际开发中，尽量多用组合少用继承，不懂的去看GoF的《Design Patterns》。

　　但重定义派生的虚函数时，在派生类中明确声明其为virtual。

**多重继承**

　　虽然允许，但是只能一个基类有实现，其他基类是接口，这样一来和JAVA一样了。这些东西在C#和JAVA中都进行了改进，直接从语法上解决问题。

**接口**

　　虚基类必须以Interface为后缀，方便阅读。阅读方便。

**重载操作符**

　　除少数特定情况外，不要重载操作符！！！“==”和“=”的操作Euqals和CopyFrom函数代替，这样更直观，也不容易出错。

**声明次序**

　　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。

**其他 C++ 特性**

**引用参数**

　　函数形参表中，所有的引用必须的const！

**缺省参数**

　　禁止使用函数缺省参数！

**异常**

　　不要使用 C++异常。

**流**

除了记录日志，不要使用流，使用printf之类的代替。

**const 的使用**

　　在任何可以的情况下都要使用const。

**命名约定**

　　1、总体规则：不要随意缩写，如果说 ChangeLocalValue 写作ChgLocVal还有情可原的话，把ModifyPlayerName写作MdfPlyNm就太过分了，除函数名可适当为动词外，其他命名尽量使用清晰易懂的名词；

　　2、宏、枚举等使用全部大写+下划线；

　　3、变量（含类、结构体成员变量）、文件、命名空间、存取函数等使用全部小写+下划线，类成员变量以下划线结尾，全局变量以g\_开头；

　　4、普通函数、类型（含类与结构体、枚举类型）、常量等使用大小写混合，不含下划线；

**格式**

　　1、行宽原则上不超过80列；

　　2、尽量不使用非ASCII字符，如果使用的话，参考 UTF-8 格式，尽量不将字符串常量耦合到代码中；

3、UNIX/Linux下无条件使用空格;

4、函数参数、逻辑条件、初始化列表：要么所有参数和函数名放在同一行，要么所有参数并排分行；

　　5、除函数定义的左大括号可以置于行首外，包括函数/类/结极体/枚举声明、各种语句的左大括号置于行尾，所有右大括号独立成行；

　　6、./->操作符前后丌留空格，\*/&不要前后都留，一个就可，靠左靠右依各人喜好；

　　7、预处理指令/命名空间不使用额外缩进，类/结构体/枚举/函数/语句使用缩进；

　　8、return不要加()；

　　9、水平/垂直留白不要滥用，怎么易读怎么来。