**1.头文件**

**1.1使用头文件保护**

使用#define进行头文件保护，而不使用微软的#pragma once。

为了保证唯一性，头文件保护的命名需要基于项目代码路径，比如Project\Src\Area\File.h 则文件的保护应该像这样：

#ifndefine         PROJECT\_SRC\_AREA\_FILE\_H

#define              PROJECT\_SRC\_AREA\_FILE\_H

…

#endif

**1.2头文件依赖**

使用前置声明减少头文件的所要包含的文件数量，也就减少了需要重新编译文件的几率。

对于需要在头文件里使用其他文件中定义的类时，如果只是使用类的声明而不是具体定义，应该是用前置声明代替包含整个文件，如下：

使用class SomeClass;

不使用 #include “SomeClass”。

**1.3包含头文件的顺序**

顺序如下：C库、C++库、其他库.h、项目内的.h，如果次文件是cpp文件，那么要首先包含其对应的头文件，然后再按前述顺序。每个层级用空行分隔，同一层级的文件顺序按英文字母先后顺序排列，如下：

#include“MyFile.h”  (如果次文件是cpp文件，首先包含自己的头文件)

#include<stdio.h>

#include<unistd.h>

#include<iostream>

#include<map>

#include<string>

#include“lib/Alpha.h”

#include“lib/Beta.h”

#include“other/zoo.h”

#include“bar.h”

#include“foo.h”

1. **作用域**

**2.1全局变量**

尽量不要使用全局变量，根据经验，全局变量起的作用往往仅是一个静态变量。如果非要使用全局变量，考虑使用单例模式代替。

多线程中使用的全局变量，不要使用class类型（包括STL容器），避免class内部的实现非线程安全而导致bug。

1. **类**

**3.1声明顺序**

成员先后顺序：typedef、enum、常量、构造析构函数、成员函数，成员变量。

域先后顺序：public、protected、private。

比如：

Class MyClass

{

public:

public:

         voidFoo();

protected:

         voidBar();

private:

         voidFunc();

public:

         intm\_nCommonVar;

private:

         intm\_nMyVar;

};

**3.2构造函数**

构造函数只负责简单的初始化功能，如果需要进行有意义的初始化，请使用Init()函数。

\*严禁在构造函数中调用虚函数。

**3.3拷贝构造函数**

仅需要拷贝一个对象时才定义拷贝构造函数，否则明确在private域中使用声明DISALLOW\_COPY\_AND\_ASSIGN，避免编译器隐式声明的拷贝构造函数导致的不确定行为。如下：

private:

DISALLOW\_COPY\_AND\_ASSIGN(MyClass);

**3.4结构体和类**

只有数据时，使用结构体，否则使用类。

**3.5继承**

尽量使用组合代替继承。

如果类中有一个virtual函数，则析构函数必须声明为virtual。

所有继承必须为public，私有继承使用其它实现方式替代。

**3.6多重继承**

尽量避免使用多重继承，除非是COM。

**3.7模板**

尽量不要使用模板，确定使用模板时确保一年后回头你还能看懂你的代码，并且调试这段模板无压力。

**3.8操作符重载**

尽量避免使用操作符重载。

**4.其他特性**

**4.1异常处理**

不要使用异常处理。

**4.2RTTI**

不要使用RTTI，也就是dynamic\_cast，非要使用只能说明设计有缺陷。

原因：dynamic\_cast会失败，而且不安全，难维护。

**4.3前置自增和自减**

不考虑结果使用的情况下，尽量使用++i代替i++，因为++i效率更高。

**4.4const使用**

尽量使用const修饰变量。

**5.注释**

**5.1头文件注释**

头文件开头部分必须使用注释，说明此文件的作者、作用等信息。

即可证明你有多叼，也可让后人叼你。

格式如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* @description: XLBrowserAppSupport的Lua接口封装

\*

\* @author:      Gaoyunxiang

\* @date:        2019.06.01

\*

\* @last modified author:

\* @last modified date:

\*

\* @ copyright Xunlei Network 2015 -

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**5.2类注释**

声明类之前需要使用类注释，说明此类的使用用途及注意事项

如下：

// 浏览器类。

// 实现浏览器的常用功能，如导航等。

// 需由浏览器工厂类来创建实例。

class Browser{};

**5.3函数注释**

重要或难理解的函数，需要在函数声明前加上注释。

比如：

// 做加法

// nVar1：第一个加数

// nVar2：第二个加数

// 返回值：相加结果

int Add(int nVar1, int nVar2);

**5.4逻辑注释**

对于很难理解、重要、实现精彩的部分必须加上注释。

**5.5TODO注释**

对于那些来不及实现、暂未实现、需要去实现的地方加上TODO注释。

如下：

// TODO: Fly to the moon.

1. **编码风格**

**6.1编码命名**

尽量不要使用缩写，严禁使用不明单词的缩写。

类名：大写字母开头，每个单词首字母大写。

函数名：大写字母开头，每个单词首字母大写

变量名：首字母（通常为前缀）小写

**6.2变量前缀**

前缀为小写，后面第一个字母为大写

前缀的结尾为\_时（如m\_..），后面第一个字母为小写

short/int/size\_t..，加n，如int nVal =1;

long，加l

float/double..，加f

bool/BOOL..，加b

DWORD，加dw

char\*，加sz，如char \*szVal= “abc”;

wchar\_t\*，加wsz

string，加str

wstring，加wstr

指针，加p

long指针，加lp

函数指针，加pfunc

vector，加vec

list，加list

map，加map

句柄类，加h

类成员变量，加m\_，如HWND m\_hParentWnd= 0x66666666;

全局变量，加g\_

静态变量，加s\_

常量，全部字母大写

类名，加C

…

**6.3缩进和换行**

使用空格代替table，在VS里将table设置成4个空格。每个级别进行4个空格的缩进。

不要连续使用多于一次的换行。

**6.4花括号**

另起一行使用。

**6.5  条件、循环、开关等语句**

关键字后面加空格再使用。

比如：

If ();

while ();

switch语句后的case:部分加作用域{}。

**7 .其它注意事项**

**7.1    堆分配和释放**

new/delete等堆操作，必须成对出现。

尽量遵从谁分配谁释放的原则。

**7.2    引用计数**

AddRef()/Release()必须成对出现。

尽量遵从谁分配谁释放的原则。

**7.3    指针、句柄使用完释放后必须设置成NULL。**

**7.4    函数功能要单一，避免耦合多功能，严禁CreateAndNavigate这种函数。**

**7.5    指针的空值比较使用NULL，而不是0。**

**7.6    浮点数的零值比较严禁使用== 0。**

**7.7    避免直接使用微软二次定义的数据类型，比如DWORD，使用其原始的unsigned long，方便跨平台。**