实验室编码规范

2016年12月13日 星期二 11:01

总体要求：

1. 绝对完全一致的编码风格，便于团队合作；
2. 在不影响可读性的情况下，用尽量少的代码函数完成功能；
3. 尽量做到代码零注释；

命名要求：

1. 变量命名规则

名称	命名规则	例子
类名	大写 C 开头，其后每个单词首字母大写，单词之间不加任何分隔符	CRenderComponent
结构名	大写 S 开头，其余规则同类名	SRectangle
枚举名	大写 E 开头，其余规则同类名	ENodeType
类的公有函数	第一个单词首字母小写，且必须为动词，其后每个单词首字母大写，单词之间不加任何分隔符	initNetwork()
类的保护函数	以单下划线开头，其余规则同公有函数	_initNetwork()
类的私有函数	以双下划线开头，其余规则同公有函数	__initNetwork()
类的虚函数	在函数名末尾添加大写字母 V	_initNetworkV()
局部普通变量	每个单词首字母大写，且单词之间不加任何分隔符，整体变量名必须为名词	LocalIP
局部指针变量	以小写字母 p 开头，其余规则同普通变量	pData
类的成员变量	以 m_开头，其余规则同普通变量	m_LocalIP
类的指针成员变量	以 m_开头，其余规则同普通指针变量	m_pData
函数输入参数	以小写字母 v 开头，不区分输入参数是否为指针，其余规则同普通变量	vLocalIP，vData
函数输出参数	以小写字母 vo 开头，其余规则同函数输入参数	voLocalIP
函数输入出参数	某个参数同时作为输入和输出参数，则以小写字母 vio 开头，其他规则同函数输入参数	vioLocalIP
指针类型的函数参数	不需要在参数前加小写字母 p	

2. 避免变量和函数命名中的拼写错误；
3. 禁止在变量和函数命名中使用拼音；
4. 函数的命名必须精确反应该函数的功能，禁止在函数内部做和函数名无关的操作；
5. 除了众所周知的缩写外，避免使用缩写；
6. get*()函数用于返回类的内部成员变量值，但外部绝对不能根据get*()的返回值来修改类的内部成员变量值；
    a. 返回类型不能为void；
    b. 所有参数必须是输入参数；
    c. get*()函数本身必须加const修饰，限制其修改类的任何成员变量；
    d. 返回值如果有必要，必须加const修饰以防止被外部修改；
7. fetch*()成员函数表示返回的值可供外部修改，因此fetch*()的返回值不用const修饰，但fetch*()本身需要用const修饰；
8. 在需要返回复杂的数据结构而无法通过get*()或fetch*()来返回结果时，使用dump*()将返回值通过函数参数的形式传入；

全局变量

1. 尽量避免使用全局变量，绝对不能在全局命名空间中使用全局变量；
2. 如果要使用全局变量，必须保证不论外部输入如何变化，该全局变量的值永远不会发生改变；

引用和const

1. 函数的返回值或传入参数是类时，尽量使用传引用而不是传值；
2. 函数的输入参数使用了引用时，同时要使用const修饰，以防止该输入参数被修改；

3. 类的成员变量为指针时，如果该成员变量的创建不是由当前类负责创建，尽量对该指针成员变量用const修饰；

注释

1. 避免为变量名增加注释去说明该变量的功能，变量名本身就该明确体现其功能；
2. 尽量避免写注释，如果要写，尽量用英文，并注意拼写错误；（注意设计文档不是注释）
3. 文件中不能有被注释掉的代码；

面向对象要求

1. 一个函数只完成一个功能，**函数函数控制在50行**；
2. **一个类的源代码行数控制在400行**；
3. 重载的虚函数必须使用关键字override进行修饰；
4. 如果类的内部成员变量是个复杂的数据结构，避免直接将该数据结构通过get*()/fetch*()函数返回；

格式要求

1. 程序中不能出现多余的空行和空格，每个空行和空格的出现必须有其存在的理由；
2. cpp文件中每个函数前加如下两行(类的构造函数和析构函数不加)

```
//*************************************************************
//FUNCTION:
```

3. 函数之间用且仅用一行空行分开；

调试辅助

1. 在函数的开头尽量使用了_ASSERT()保证输入参数的合法性，以及保证函数的前置假设都成立；
2. 获取到指针在使用前尽量通过_ASSERT()保证保证该指针不为空；
3. 避免串联式的指针调用（如pClass1->getClass2()->getClass3()->doSomething()）；
4. 调试辅助的代码放在#ifdef _DEBUG / #endif 之间；

HIVE提供的公共功能

1. HiveCommonMicro.h中提供了大量宏用于压缩代码行数，尽量使用

```
namespace hiveCommon
{
    #define _SAFE_DELETE(p)       { delete   (p); (p)=nullptr; }
    #define _SAFE_DELETE_ARRAY(p) { delete[] (p); (p)=nullptr; }
    #define _OFFSET_2D(x, y, w)   ((y) * (w) + (x))
    #define _SWAP(a, b, t)        {t=a; a=b; b=t;}
    #define _MIN(x, y)            (x)<(y)?(x):(y)
    #define _MAX(x, y)            (x)>(y)?(x):(y)
    #define _HIVE_EPSILON         (FLT_EPSILON * 10)

    #define __EXCEPTION_SITE__    __FUNCTION__, __FILE__, __LINE__
    #define __FUNCTION_SITE__     __FUNCTION__, __FILE__, __LINE__
    #define _HIVE_EARLY_RETURN(condition, prompt, return_value) if (condition) {hiveCommon::hiveOutputWarning(__EXCEPTION_SITE__, prompt); return return_value;}
    #define _HIVE_EARLY_EXIT(condition, prompt)                 if (condition) {hiveCommon::hiveOutputWarning(__EXCEPTION_SITE__, prompt); return;}
    #define _HIVE_SIMPLE_IF(condition, opTrue)                  if (condition) {opTrue;}
    #define _HIVE_SIMPLE_IF_ELSE(condition, opTrue, opFalse)    if (condition) {opTrue;} else {opFalse;}
    #define _HIVE_UNIMPLEMENTED_FUNC                            hiveCommon::hiveOutputWarning(__EXCEPTION_SITE__, "Unimplemented function!");
    #define _HIVE_INCOMPLETE_FUNC                               hiveCommon::hiveOutputWarning(__EXCEPTION_SITE__, "Incomplete function!");

    #define _BOOST_STR1(s, aug1)            boost::str(boost::format(s) % (aug1))
    #define _BOOST_STR2(s, aug1, aug2)      boost::str(boost::format(s) % (aug1) % (aug2))
    #define _BOOST_STR3(s, aug1, aug2, aug3) boost::str(boost::format(s) % (aug1) % (aug2) % (aug3))
    #define _BOOST_STR4(s, aug1, aug2, aug3, aug4) boost::str(boost::format(s) % (aug1) % (aug2) % (aug3) % (aug4))
    #define _BOOST_STR5(s, aug1, aug2, aug3, aug4, aug5) boost::str(boost::format(s) % (aug1) % (aug2) % (aug3) % (aug4) % (aug5))
    #define _BOOST_STR6(s, aug1, aug2, aug3, aug4, aug5, aug6) boost::str(boost::format(s) % (aug1) % (aug2) % (aug3) % (aug4) % (aug5) % (aug6))
    #define _BOOST_STR7(s, aug1, aug2, aug3, aug4, aug5, aug6, aug7) boost::str(boost::format(s) % (aug1) % (aug2) % (aug3) % (aug4) % (aug5) % (aug6) % (aug7))
    #define _BOOST_STR8(s, aug1, aug2, aug3, aug4, aug5, aug6, aug7, aug8) boost::str(boost::format(s) % (aug1) % (aug2) % (aug3) % (aug4) % (aug5) % (aug6) % (aug7) % (aug8))

    #define _HIVE_ASSERT(condition, prompt) {if (!(condition)) {_CrtDbgBreak(); hiveCommon::hiveOutputWarning(__EXCEPTION_SITE__, prompt);}}
    #define _HIVE_ASSERT_AND_EARLY_EXIT(condition, prompt) {if (!(condition)) {_CrtDbgBreak(); hiveCommon::hiveOutputWarning(__EXCEPTION_SITE__, prompt); return;}}
    #define _HIVE_ASSERT_AND_EARLY_RETURN(condition, prompt, return_value) {if (!(condition)) {_CrtDbgBreak(); hiveCommon::hiveOutputWarning(__EXCEPTION_SITE__, prompt); return return_value;}}

    #define _THROW_HIVE_EXCEPTION(error_info) {throw hiveCommon::HIVE_EXCEPTION() << hiveCommon::HIVE_EXCEPTION_INFO(error_info);}
}
```

2. 使用hiveCommon提供的日志功能来完成日志输出，禁止在程序中自己调用printf()函数或往std::cout输出内容；
3. 和配置文件相关的功能，使用HIVE自带的配置文件解析功能；

头文件

1. 尽量避免包含无用的头文件，特别是在头文件中，尽量使用前置声明来避免在头文件中包含其他头文件；
2. 头文件的包含顺序应该为：
    a. 基类头文件或当前类对应的头文件（用双引号）；
    b. C++自带的头文件（用尖括号）；
    c. boost的头文件（用尖括号）
    d. 其他源码提供的头文件（用尖括号）；
    e. 本项目的头文件（用双引号）；

```cpp
#include "OSGCameraManipulator.h"
#include <boost/algorithm/string.hpp>
#include <boost/format.hpp>
#include "common/CommonInterface.h"
#include "common/HiveCommonMicro.h"
#include "../AbstractKernel/RenderEngineInterface.h"
#include "OSGEngineInterface.h"
#include "OSGEngine.h"
```

其他

1. 双重循环使用循环变量i和k，而不是i和j
2. 变量的定义尽量靠近其第一次使用；
3. 禁止程序中出现0之外的任何数字；

```cpp
    DWORD *index = new DWORD[(row - 1)*(col - 1)*2*3];    //为什么要乘以2再乘以3？
```

来自 <http://192.168.18.250:8090/pages/viewpage.action?pageId=5210440>