**1：添加文件修改、版权信息**

在每一个头文件和。pp文件的开头，请以注释的方式注明本文件所属的模块、本文件的功能等，如：

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

＊　Copyright(c)2010 Teleca company.

＊　All rights reserved.

＊

＊　文件名称：filename.h/ filename.cpp

＊　简要描述： 简要描述文件的内容和功能

＊

＊　当前版本：1.1

＊　作者/修改者：

＊　完成日期：

＊　修订说明：

＊

＊　取代版本：1.0

＊　修改人：

＊　完成日期：

＊　修订说明：

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

**2 函数注释分三部分：功能、参数和返回值**

每个程序都应该以一段简短地、说明其功能的注释开头。请为每个函数书写注释以说明函数做了些什么。如果有参数和返回值，也需要对其进行解释。如下：

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

\* 函数名称：函数名

\* 功能描述：简要描述改函数实现的功能

\* 参数列表：param1——描述；

\* param2——描述；

\* param3——描述；

\* 返回结果：描述返回值的情况

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

参数名采用第一个单词首字母小写而后面的单词首字母大写的单词组合（即**骆驼风格**），如: int leftValue

**3.变量的定义和注释**

变量名的定义采用骆驼风格,应当使用“名词”或者“形容词＋名词”。

例如：

float value;

float oldValue;

float newValue;

对于每个静态变量和全局变量，请添加相应的注释语句。对于重要的变量、定义等进行注释。

**4. 变量和函数的命名原则**

1)全局（包括类中的）变量用长名字，局部变量用短名字。类成员变量前应加上 ｍ＿，静态变量加上S\_,全局变量加上 ｇ＿.

* 类的数据成员加前缀m\_（表示member），这样可以避免数据成员与成员函数的参数同名。

例如：

void Object::SetValue(int width, int height)

{

m\_width = width;

m\_height = height;

}

* 静态变量加前缀s\_（表示static）。

例如：

void Init(…)

{

static int s\_initValue; // 静态变量

…

}

* 如果不得已需要全局变量，则使全局变量加前缀g\_（表示global）。

例如：

int g\_howManyPeople; // 全局变量

int g\_howMuchMoney; // 全局变量

2)局部变量应尽量易懂简洁，使用常见的变量，如: nCount，i，j，k，len, pos, ect.

3)自定义信号以sig＿开头，自定义槽以slt＿开头。sig＿和slt＿后面第一个单词（或其缩写）要以小写字母开头，第二个以后的单词（或缩写）要以大写字母开头。

5)如果程序中出现了两个类名缩写一样的问题，如：QToolBar 和 QToolButton 的缩写都是tb，这时应将其中一个类的缩写进行变动，变动的准则以能够避免冲突，同时缩写能够表达类名为准。如这里可以将QToolBar 的缩写为tbar，QToolButton仍用tb，这样就可以避免命名冲突。

6) 类名用大写字母开头的单词组合而成，而函数名用骆驼风格。

例如：

class Node; // 类名

class LeafNode; // 类名

void drawRect(void); // 函数名

void setValue(int value); // 函数名

**5. 头文件结构及类声明体中的编排及**

1) 头文件由三部分内容组成：

<1>头文件开头处的版权和版本声明（参见示例1-1）。

<2>预处理块。

<3>函数和类结构声明等。

其中，头文件以内部包含卫哨开始（#define\*\*\*, #ifndef\*\*\*），以内部包含卫哨结束(#endif //\*\*\*)。

2)在类的声明体中，请按照：Q\_OBJECT、public：、siganls：、slots：、protected：、priavte:的顺序将成员变量和函数进行排列。如果需要在这个类（class）中声明某种数据类型（结构、枚举等），请将这个声明放在所有成员变量和成员函数的前面。如果在一种类型的声明中，即有成员变量，又有函数声明，请使用两个相同的类型声明将它们隔开，如：

private：

void function（）；

…

private：

int m\_num；

…

3)对一般的界面类，如果使用了信号／槽，则class声明的第一行必须是Q\_OBJECT。

4)建议避免使用protected 类型的成员函数或者成员变量，因为在 Qt 中，大量的事件处理函数（槽）均是以protected 类型出现的。

**6:界面布局原则**

　 制作Qt界面的时候，请尽量使用布局管理器(QlayOut)进行布局管理，避免使用绝对坐标，除非你能够绝对肯定这个界面的尺寸是完全不会变换的。如果某个区域出现了很多控件，在保证不会影响程序控制结构的前提下，请将这些控件放在一个可以包含其它窗口控件

的盒子（如 QWidget、QFrame、QGroupBox）中，然后再让这个盒子出现在界面上（即这些控件以整体形式出现）。书写界面上的英文长度的时候，请与对应的中文进行比较，确保汉化后中文能够被完全显示。

　 代码中如果出现了需要在界面上出现的文字，必须用英文表示，并且使用tr 函数将其包含。如：tr(“message！”),不能在程序中将这些文字直接用中文替换。

**7.调试信息原则**

代码中如果需要输出调试信息，请使用条件编译QT\_NO\_DEBUG\_OUTPUT宏将其包围，如：

#ifndef QT\_NO\_DEBUG\_OUTPUT

qDebug(“debug message”);

QMessageBox::warning(0, “warning”, “message”);

#endif // QT\_NO\_DEBUG\_OUTPUT