**QT Coding Convention**

| Version | Date | Status | Comments | Author |
| --- | --- | --- | --- | --- |
| 0.1 | 29.06.2010 |  | Combine Nokia, QT and C++ coding conventions | Forrest Chen,  Johnny Liu |
|  |  |  |  |  |
|  |  |  |  |  |

**Basic coding convention from Nokia and QT official web**



**Additional conventions which are not mentioned by Nokia and QT**

**1：Add file and version information.**

At the beginning of header files and .cpp files, need to use comment make simple notes about Copyright, Function description ,Version, Author,ect. Function description is a little more important, we often simplify it over or forget to update when some big changes are made.

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

\*　Copyright(c)2010 Teleca company.

\*　All rights reserved

\*

\*　Name: filename.h/ filename.cpp

\*　Function: Brief description files contents and features.

\*

\*　Version: 1.1

\*　Author: Johnny Liu

\*　Date: June 28, 2010

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

**2. Function header comments: Features, Parameters and Return value**

Every program should begin with a brief comment to explain the features of function. And all of parameters and return values should add comments for brief description too. E.g:

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

\* Function: Briefly describe the features of function

\* Parameters: param1——Description；

\* param2——Description；

\* param3——Description；

\* Return : Briefly describe the return value

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

**3. Variables definition and comment.**

1)The definition of variables use Hungarian style(The compound words or phrases in which the elements are joined without spaces, the first letter is low case and the first part is abbreviation of variable category, the words behind are all start with upper case). The name should use “Noun” or “Adj. + Noun”. E.g.

float fValue;

int iOldValue;

QString strNewValue;

All variables need comments if additional comments is needed, it means that name of variable can’t say what it is or not completely, it need to tell developer more information.

Variables name are use Hungarian notation, the details of Hungarian notation please check the **Appendix\_1,** Category abbreviation, please see **Appendix\_2 at the end**

2)Global variables and class data members use long name, local variables use short name. The name of class data members beginning with m\_, static variables beginning with s\_, global variables beginning with g\_.

* The prefix of class data members start with m\_(‘m’ means member) can avoid confuse with another variables. E.g.

Class ClassA

{

private:

int m\_iWidth;

int m\_iHeight;

}

* The prefix of static variables is s\_(‘s’means static), E.g.

void init(…)

{

static int s\_iInitValue; // static variables

…

}

* If we have to use global variables, the prefix of it is g\_ (means global), E.g.

int g\_iHowManyPeople; // global variables

int g\_iHowMuchMoney; // global variables

**Notes:** We should try to avoid use global variables as we can.

3)Local variables should simple and easy to understand, use common variables, E.g. nCount，i，j，k，len, pos, ect.

4)In program, if two or more classes have same abbreviation, like QToolBar and QToolButton, both abbreviation are tb, then we need to change the abbreviation in one of them, the change principle is avoid conflict and can express the mean of class. E.g. The abbreviation of QToolBar can be change to ‘tbar’, but QToolButton still use ‘tb’ for arrreviation.

5) Class name start with the combination of words that all start with upper case, but the function name use camel style. E.g.

class Node; // class name

class LeafNode; // class name

void drawRect(void); // function name

void setValue(int value); // function name

**4. Header files structure and class declaration arrangement.**

1) The header files consist with three parts.

<1>The version declaration at the beginning of header files.（Reference the first rule）.

<2> Preprocessing block.

<3>Function and class declaration, ect.

In it, the header files start with**(#define\*\*\*, #ifndef\*\*\***)，end with**(#endif //\*\*\***).

2)In class declaration, the order is: **Q\_OBJECT-> public-> siganls-> slots->protected-> priavte**. If need to declaration some another data types(Structure,Enumerate,etc.), should put those declaration before the data members and function members. If in same class, both data variables and function declaration are use same type declaration , split both of them. E.g.

**private：**

void function();

…

**private：**

int m\_iNum;

…

3)At usual, if we use the Signals/Slots, the first sentence of class declaration is Q\_OBJECT.

**5:UI layout principle.**

1)When make Qt UI, use QLayout as more as you can for layout management, try to avoid use absolute coordinates, unless you are certain that the UI won’t changed it’s coordinate forever.

2)If one area have many widgets, try to put those widgets in a window box(e.g. QWidget, QFrame,QGroupBox,etc.), then put those window boxes in the UI.

**6.Debug information**

At first, in order to debug our program, we should add debug information in our codes. We should define a macro wrapped by QT\_NO\_DEBUG\_OUTPUT in \*.h file, E.g.

#ifndef QT\_NO\_DEBUG\_OUTPUT

#define PRINT(s) qDebug(s)

#else

#define PRINT(s)

#endif // QT\_NO\_DEBUG\_OUTPUT

**Appendix:**

* 1. Appendix\_1: Hungarian notation 
  2. Appendix\_2: Contorls abbreviation 