C、C++、C#编程规范

ver 1.0.0

2006-10-13

1. 目的

为了保证公司编写出的程序都符合相同的规范，保证一致性、统一性而建立的程序编码规范。

2. 范围

适用于部门内所有 C、C++、C#程序，包括基于.NET 平台的软件开发工作。编写其他程序也可以适当参考。

3. 规范的总体要求

3.1. 基本要求

(1) 程序结构清晰，简单易懂，单个函数/方法的程序行数不得超过 100 行。
(2) 打算干什么，要简单，直截了当，代码精简，避免垃圾程序。
(3) 尽量使用标准库函数和公共函数。
(4) 不要随意定义全局变量，尽量使用局部变量。
(5) 使用括号以避免二义性。

3.2. 可读性要求

(1) 可读性第一，效率第二。避免太过于技巧性，尽量不采用递归模式。
(2) 保持注释与代码完全一致。
(3) 每个源程序文件，都有文件头说明，说明规格见规范。
(4) 每个函数，都有函数头说明，说明规格见规范。
(5) 主要变量（结构、联合、类或对象）定义或引用时，注释能反映其含义。
(6) 常量定义（DEFINE）有相应说明。
(7) 处理过程的每个阶段都有相关注释说明。
(8) 在典型算法前都有注释。
(9) 利用缩进来显示程序的逻辑结构，缩进量一致并以 Tab 键为单位
(10) 循环、分支层次不要超过五层。
(11) 注释可以与语句在同一行，也可以在上行。
(12) 空行和空白字符也是一种特殊注释。
(13) 一目了然的语句不加注释。
(14) 注释的作用范围可以为：定义、引用、条件分支以及一段代码。
(15) 注释行数（不包括程序头和函数头说明部份）应占总行数的 1/5 到 1/3 。

3.3. 结构化要求

(1) 禁止出现两条等价的支路。
(2) 禁止 goto 语句。
(3) 用 if 语句来强调只执行两组语句中的一组。禁止 else goto 和 else return。
(4) 用 case 实现多路分支。
(5) 避免从循环引出多个出口。
(6) 函数只有一个出口。
(7) 不使用条件赋值语句。
(8) 避免不必要的分支。
(9) 不要轻易使用条件分支去替换逻辑表达式。

3.4. 正确性与容错性要求

(1) 程序首先是正确，其次是优美。
(2) 无法证明你的程序没有错误，因此在编写完一段程序后，应先回头检查。
(3) 改一个错误时可能产生新的错误，因此在修改前首先考虑对其它程序的影响。
(4) 所有变量在调用前必须被初始化。
(5) 对所有的用户输入，必须进行合法性检查。
(6) 不要比较浮点数的相等，如： 10.0 * 0.1 == 1.0 ， 不可靠
(7) 程序与环境或状态发生关系时，必须主动去处理发生的意外事件，如文件能否逻辑锁定、打印机是否联机等。
(8) 单元测试也是编程的一部份，提交联调测试的程序必须通过单元测试。

3.5. 可重用性要求

(1) 重复使用的完成相对独立功能的算法或代码应抽象为公共控件或类。
(2) 公共控件或类应考虑 OO 思想，减少外界联系，考虑独立性或封装性。
(3) 公共控件或类应建立使用模板。

4. 变量命名规范

程序中变量名称 = 变量的前缀 +代表变量含意的英文单词或单词缩写。

(1) 命名必须具有一定的实际意义，即使对于可能仅出现在几个代码行中的生存期很短的变量，仍然使用有意义的名称。仅对于短循环索引使用单字母变量名，如 i 或 j。

(2) 局部变量中可采用如下几个通用变量：nTemp，nResult，i，j（一般用于循环变量）。

(3) 鉴于大多数名称都是通过连接若干单词构造的，请使用大小写混合的格式以简化它们的阅读。每个单词的第一个字母都是大写.

(4) 在变量名中使用互补对，如 min/max、begin/end 和 open/close。

(5) 不要使用原义数字或原义字符串，如 for i = 1 to 7。而是使用命名常数，如 for i = 1 To NUM_DAYS_IN_WEEK 以便于维护和理解。

具体例程:
Char 或者 TCHAR 类型 与 Windows API 有直接联系的用 szAppName[10]形式，
　　否则用 FileName[10]形式,单个字符也可用小写字母表示;
BOOL 类型 bEnable;
int 类型 iCmdShow;
float 类型 fTmp
LONG 类型 lParam;
UINT 类型 uNotify;

DWORD 类型 dwStart;
PSTR 类型 pszTip;
LPSTR 类型 lpCmdLine
LPTSTR 类型 lpszClassName;
LPVOID 类型 lpReserved
WPARAM 类型 wParam,
LPARAM 类型 lParam
HWND 类型 hDlg;
HDC 类型 hDC;
HINSTANCE 类型 hInstance
HANDLE 类型 hInstance,
HICON 类型 hIcon;
DWORD 类型 dwWord
String 或者 AnsiString 类型 strMyString

类成员变量以_m_开头： m_nVal, m_bFlag
全局变量以 g_开头 g_nMsg, g_bFlag

资源名字定义格式:
菜单:IDM_XX 或者 CM_XX
位图:IDB_XX
对话框:IDD_XX
字符串:IDS_XX
DLGINIT:DIALOG_XX
ICON:IDR_XX

4.1. 结构体（typedef struct）命名

结构体类型命名必须全部用大写字母；结构体变量命名必须用大小写字母组合，第一个字母必须使用大写字母，必要时可用下划线间隔。对于私有数据区，必须注明其所属的进程。全局数据定义只需注意其用途。

示例如下：

```
typedef struct
{
    char szProductName[20];
    char szAuthor[20];
    char szReleaseDate[16];
    char szVersion[10];
    unsigned long MaxTables;
    unsigned long UsedTables;
}DBS_DATABASE;
```

DBS_DATABASE GdataBase;

4.2. 控件命名规则

控件命名=控件缩写前缀 + “_” +变量名

控件	缩写
Label	lbl
TextBox	txt
CheckBox	chk
Button	cmd
ListBox	lst
DropDownList	ddl
窗口	fm
下拉式列表框 combo	cob
图象 image	img
picture	pic
网格 Grid	grd
滚动条	scr
fram	frm

4.3. 常量命名和宏定义规范

(1) 常量和宏定义必须具有一定的实际意义，常量名均为大写，字之间用下划线分隔。
(2) 每一条定义的右侧必须有一简单的注释，说明其作用。
(3) 常量和宏定义在#include 和函数定义之间。
(4) 变量名和常量名最多可以包含 255 个字符，但超过 30 个字符的名称比较笨拙。

例：

```
private const bool   WEB_ENABLEPAGECACHE_DEFAULT          = true;
private const int    WEB_PAGECACHEEXPIRESINSECONDS_DEFAULT = 3600;
define NUM_DAYS_IN_WEEK 7
```

4.4. 类（Class）命名规范

(1) 名字尽量不使用缩写，除非它是众所周知的。
(2) 名字可以有两个或三个单词组成，但通常不应多于三个，不要使用下划线。
(3) 使用名词或名词短语命名类。

例：

```
public class FileStream
public class Button
public class String
```

(4) 类模块级的变量请用“m_”作前缀

```
public class hello
{
            private string m_Name;
            private DateTime m_Date;
}
```

(5) 类的属性所对应的变量，采用属性名前加“m_”前缀的形式

```
public class hello
{
   private string m_Name;
       public string Name ()
      {
                    return m_Name;
       }
}
```

(6) 过程级的变量不使用前缀

```
public class hello
{
      void say()
      {
            string SayWord;
       }
}
```

(7) 过程的参数使用“p_”作为参数

```
  public class hello
  {
          void say(string p_SayWord)
          {
           }
  }
```

4.5.接口（Interface）命名规范

和类命名规范相同，唯一区别是　接口在名字前加上“I”前缀。例如：

```
interface IDBCommand;
interface IButton;
```

4.6.方法（Method）和函数命名规范

第一个字母必须使用大写字母,要求用大小写字母组合规范函数命名,必要时可用下划线间隔。函数原型说明包括引用外来函数及内部函数，外部引用必须在右侧注明函数来源： 模块名及文件名，如是内部函数，只要注释其定义文件名。

示例如下：

```
void UpdateDB_Tfgd (TRACK_NAME); //Module Name :r01/sdw.c
void PrintTrackData (TRACK_NAME); //Module Name :r04/tern.c
void ImportantPoint (void); //Module Name :r01/sdw.c
void ShowChar (int , int , chtype); //Local Module
void ScrollUp_V (int , int); //Local Module
```

4.7. 命名空间（NameSpace）命名规范

和类命名规范相同。

4.8. 异常捕获的一些规则

针对异常捕获过程中的 Exception 变量命名，在没有冲突的情况下，统一命名为 ex。

```
Try
{
       //your code
       try
       {
                //code
       }
       catch(Exception ex)
       {
                //your code
       }
  }
catch(Exception ex)
{
       //your code
  }
```

如果捕获异常不需要作任何处理，则不需要定义 Exception 实例

例：

```
try
{
       //your code
}
catch( Exception )
{
}
```

5. 注释

(1) 原则上注释要求使用中文，或者浅显的英文。
(2) 文件开始注释内容包括:公司名称、版权、作者名称、时间、模块用途、背景介绍等,复杂的算法需要加上流程说明。
(3) 函数注释包括:输入、输出、函数描述、流程处理、全局变量、调用样例等,复杂的函数需要加上变量用途说明;
(4) 程序中注释包括:修改时间和作者、方便理解的注释等。

5.1. 文件开头的注释模板

```
/*********************************************************************
** 文件名:
** Copyright (c) 1998-2006  亚信科技（成都）有限公司 CMCC-SW-PSO
** 创建人:
** 日 期:
** 修改人:
** 日 期:
** 描 述:
**
** 版 本:
**
*********************************************************************/
```

5.2. 函数开头的注释模板

```
/*********************************************************************
** 函数名:
** 输 入: a,b,c
** a---
** b---
** c---
** 输 出: x---
** x 为 1, 表示...
** x 为 0, 表示...
** 功能描述:
** 全局变量:
** 调用模块:
** 作 者:
** 日 期:
** 修 改:
```

** 日 期:
** 版本
**********/

5.3. 程序中的注释模板

/*----------*/
/* 注释内容 */
/*----------*/

5.4. 单行注释

提倡使用//

5.5. 其他

对于比较大的函数，每个 block 和特殊的函数调用，都必须注明其功能，举例如下：

```
count.divisor = 1193280 / freq; // compute the proper count
OutByte((unsigned short)67, (unsigned char)182); // tell 8253 that a count is coming
OutByte((unsigned short)66, count. c[0]); // send low-order byte
OutByte((unsigned short)66, count. c[1]); // send high-order byte
```