**1**。模块描述

模块是为了实现某一功能的函数的集合，文件名使用缺省的后缀，在每一模块的开头

应有如下的描述体：

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

\*\*\*

\* PROJECT CODE ：项目代号或名称 \*

\* CREATE DATE ：创建日期 \*

\* CREATED BY ：创建人 \*

\* FUNCTION ：模块功能 \*

\* MODIFY DATE ：修改日期 \*

\* DOCUMENT ：参考文档 \*

\* OTHERS ：程序员认为应做特别说明的部分，如特别的编译开关 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*/

不同的修改人应在修改的地方加上适当的注释，包括修改人的姓名。另外，如有必要，

要注明模块的工作平台，如单板OS、DOS、WINDOWS 等。注明适用的编译器和编译模式。

**2**。函数描述

函数是组成模块的单元，一般用来完成某一算法或控制等。在每一函数的开头应有如下

的描述体：

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

\*\*\*

\* FUNCTION NAME：函数名称 \*

\* CREATE DATE ：创建日期 \*

\* CREATED BY ：创建人 \*

\* FUNCTION ：函数功能 \*

\* MODIFY DATE ：修改日期 \*

\* INPUT ：输入参数类型(逐个说明) \*

\* OUTPUT ：输出参数类型(逐个说明) \*

\* RETURN ：返回信息 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*/

可选的描述有：

\* RECEIVED MESSAGES：收到的消息 \*

\* SENT MESSAGES ：发送的消息 \*

\* DATABASE ACCESS ：存取的数据库 \*

\* CALLED BY ：该函数的调用者 \*

\* PROCEDURES CALLED：调用的过程 \*

\* RECEVED PRIMITIVES : 收到的原语 \*

\* SENT PRIMITIVES : 发送的原语 \*

及其它程序员认为应有的描述。标题可以只大写第一个字母。例如：Function Name：

**3**。命名规则：

A) **函数**： 函数名应能体现该函数完成的功能，关键部分应采用完整的单词，辅助部

分若太长可采用缩写，缩写应符合英文的规范。每个单词的第一个字母大写。如：ShowPoints，

CtrlDestBoard，SendResetMsg 等。

B) **变量**： 变量的命名规则部分采用匈牙利命名规则(鼓励完全使用匈牙利名规则)。

变量的第一个或前两个字母小写，表示其数据类型，其后每个词的第一个字母大写。

推荐的类型前缀如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 前缀 | 含义 | 前缀 | 含义 |
| a | 数组 | n | Short int |
| b | BOOL | np | 短指针 |
| by | BYTE | p | 指针 |
| c | char | l | Long |
| cb | 字节记数 | lp | 长指针 |
| cr | 颜色参考值 | s | 串 |
| cx,cy | 短型（x,y长度的记数） | sz | 以零结尾的串 |
| dw | DWORD | tm | 文本 |
| fn | 函数 | w | WORD |
| h | HANDLE | x,y | 短型（x或y的坐标） |
| i | int | g\_ | 全局变量 |
| m | 类的数据成员 | uc | unsigned char |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

如iCurrentValue，uTransitionCount 等。对于其他复合类型或自定义类型，请用适当的前缀

yf-f4-06-cjy

来表示。除局部循环变量外，不鼓励单个字母的变量名。

对于常用的类型定义，尽量使用WORD、BOOL、LPWORD、VOID、FAR、NEAR 等

惯用写法，避免使用char、long、void、far、near 等小写格式。不使用\_UC、 \_UL 等XXXX

公司以前一些人的习惯写法。

C) 结构：结构的定义有两个名称，一个是该结构的类型名，一个是变量名。按照C 语

言的语法，这两个名称都是可选的，但二者必有其一。我们要求写类型名，类型名以tag 做

前缀。下面是一个例子：struct tagVBXEVENT

{

HCTL hControl;

HWND hWindow;

int nID;

int nEventIndex;

LPCSTR lpEventName;

int nNumParams;

LPVOID lpParamList;

}veMyEvent;

tagVBXEVENT veMyEvent[MAXEVENTTYPE], \*lpVBXEvent;

对于程序中常用的结构，希望能使用 typedef 定义，格式如下：

typedef struct tagMYSTRUCT

{

struct members .......

} TMYSTRUCT,\* PTMYSTRUCT,FAR \* LPTMYSTRUCT;

struct 后的类型名有tag前缀，自定义的结构名称一律用大写字母，前面可以加一大写的T。

而结构类型变量定义则可以写为：

TMYSTRUCT variablename;

并可在定义 MYSTRUCT\_S 同时根据需要，定义其指针，远指针和尺寸常量：

typedef tagMYSTRUCT

{

struct members **… …**

}TMYSTRUCT,\* PTMYSTRUCT,FAR \* LPTMYSTRUCT;

#define MYSTRUCTSIZE sizeof ( TMYSTRUCT );

结构变量的命名，建议采用如下方式：

从结构名中，取出二至三个词的首字母作为代表此结构的缩写，小写作为变量前

缀。例如：（取自Microsoft Windows示例）

struct OPENFILENAME ofnMyFile

strcut CHOOSECOLOR ccScreenColor

D) **联合**：联合的命名规则和结构相似，如：

union tagMYEXAMPLE

{

int iInteger;

long lLongInt;

} myExample;

typedef tagMYEXAMPLE

{

yf-f4-06-cjy

int iInteger;

long lLongInt;

} MYEXAMPLE,\* PMYEXAMPLE,FAR \* LPMYEXAMPLE;

**4**。书写风格：

A) 函数：函数的返回类型一定要写，不管它是否默认类型，函数的参数之间应用一逗

号加一空格隔开,若有多个参数，应排列整齐。例如：

int SendResetMsg( PTLAPENTITY pLAPEntity, int iErrorNo )

{

int iTempValue;

.

.

.

}

函数的类型和上下两个括号应从第一列开始，函数的第一行应缩进一个**TAB**，不得用

空格缩进。(按大多数程序范例，**TAB** 为四个字符宽，我们规定：**TAB** 为四个字符宽。)

B) 语句：循环语句和**if** 语句等块语句的第一个大括号‘**{**’可跟在第一行的后面，接

下来的语句应缩进一个**TAB**，如：

for ( count = 0 ; count < MAXLINE ; count++ ) {

if ( (count % PAGELINE) == 0 ) {

.

.

.

}

.

.

}

也可另起一行，如：

for ( count = 0 ; count < MAXLINE ; count++ )

{

if ( (count % PAGELINE) == 0 )

{

.

.

.

}

.

.

}

两种写法在世界著名的程序员手下均可见到，我们尊重个人的习惯，但推荐使用后一种

写法。

复杂表达式(两个运算符以上，含两个)必须用括号区分运算顺序，运算符的前后应各有

一空格，习惯写在一行的几个语句(如IF 语句)，中间应有一空格，其它语句不鼓励写在同

一行。

空格加在适当的地方，如 **if (** ; **for (** ; **) {**;

语句的上下对齐也可使程序便于阅读，如：

myStruct.iFirstNumber = 0;

myStruct.lSecondNumber = 1;

myStruct.pThePoint = NULL;

C) 常量：常量一般情况下可用宏定义，用大写的方式，单词之间用下划线隔开 如：

yf-f4-06-cjy

#define MAX\_LINE 100

#define PI 3.1415926

不鼓励在程序中出现大量的数字常数。

注：对于一些有必要说明的缩写，可以在模块描述内加以说明。

**5**。头文件：头文件一般包括了数据结构的定义，函数原形的说明，宏定义等，不许包含函

数体和变量实体，文件名使用缺省的后缀.h，不使用类似.DEF 等非标准的后缀名，头文件

的开始可包括如下的注释：

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

\*\*

\* CREATE DATE：创建日期 \*

\* CREATED BY ：创建人 \*

\* MODIFIED BY ：修改人 \*

\* USED BY ：由哪些模块使用 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*/

为了避免重编译，应加上条件编译语句，如文件headfille.h 应包含下列语句：

#ifndef \_\_HEADFILE\_H

#define \_\_HEADFILE\_H

.

.

.

#endif

**6¡£** 预编译宏：对于代码中使用到的预编译宏，应在文件的适当处加以说明。对于连续多

行的宏，在书写上也采用缩进的格式。例如：

#ifdef \_BORLANDC\_

#define FAR far

#define NEAR near

#ifdef OS\_DEBUG

#define HHH 0x11

#define JJJ 0x22

#undef KKK

#endif

#else

#define FAR

#define NEAR

#endif

对于夹在代码中的编译开关，在书写格式上当作源代码一样处理。例如：

**…**

lpPrim = (LPTL2L1PRIM)GetMessage( &queL2L1 );

if( lpPrim == NULL )

{

#if OS\_DEBUG

Printf(**“**Queue L2->L1 is empty!**”** );

#endif

return;

}

**…**

**7**。注释：注释是源码中非常重要的部分，不应少于源码行数的15%，我们希望能达到25%

或更多。注释用中英文都可以，但应易读易懂。不要在一个语句的中间插入注释。

本规范内容符合公司有关文档要求，参照了北京研究部的相关文档，并听取了大家的意

yf-f4-06-cjy

见，希望诸位在软件开发过程中，能主动遵照本规范，写出高质量的源码！

附页

本附页的内容不是“软件编程规范”的一部分，但对于较大的软件，值得推荐。

1。接口函数（被其它模块调用的函数）及接口变量（被其它模块引用的变量），其命名可增

加前缀表明其所归属的模块。

如：NLS\_MsgProc, NLP\_GetPHTime,

NLS\_LogcnToConnid ...

由于C 的函数都是全程的，其它函数也建议增加前缀来避免重名或混淆。

2。头文件分开定义

A) 自用头文件：定义模块自用的函数，结构，变量等。

B) 输出头文件：用于该模块向外输出，即别的模块在使用该模块提供功能时需要引用的

头文件，包含该模块的输出函数和输出数据结构，输出常量。

3。建议在使用有一定关联的成组常量时使用枚举定义：

typedef enum

{

MONDAY,

.....

SUNDAY,

} WEEKDAY;

在有特殊数值要求时在枚举内定义：

typedef enum

{

MONDAY = 1,

.......

SUNDAY = 7,

} WEEKDAY;

好处：容易理解有关联的常量；枚举变量可以当作整型变量使用。