# 软件编码规范说明书

# 西北师范大学宿舍管理系统

# **软件编码规范说明书**

# 1 介绍

## 1.1 为什么要有编码规范

编码规范对于程序员而言尤为重要，有以下几个原因：

- 一个软件的生命周期中，80%的花费在于维护  
- 几乎没有任何一个软件，在其整个生命周期中，均由最初的开发人员来维护  
- 编码规范可以改善软件的可读性，可以让程序员尽快而彻底地理解新的代码  
- 如果你将源码作为产品发布，就需要确任它是否被很好的打包并且清晰无误，一如你已构建的其它任何产品

# 2 编码风格

## 2.1 代码缩进

程序块（包括函数、过程、结构的定义及循环、判断等语句）要严格采用缩进风格编写，对齐只使用空格键，不使用TAB键，所有的缩进为4个空格。

## 2.2 变量声明

程序块（包括函数、过程、结构的定义及循环、判断等语句）要严格采用缩进风格编写，对齐只使用空格键，不使用TAB键，所有的缩进为4个空格。

## 2.3 代码块长度

单个函数的程序行数不得超过200行。一个程序文件的长度不得超过5000行代码。

## 2.4 代码换行

1. 较长的语句（>80字符）要分成多行书写，长表达式要在低优先级操作符处划分新行，操作符放在新行之首，划分出的新行要进行适当的缩进，使排版整齐，语句可读。

2. 不允许把多个短语句写在一行中，即一行只允许写一条语句（如if 、for 、do 、while、case 、switch等语句自占一行）。

3. 程序块的分界符必须各自独占一行并且位于同一列，同时与引用它们的语句左对齐。

## 2.5 空行和空格

1. 通过摘要注释分隔的两个代码块之间、局部变量和它后边的语句之间、函数之间留一个空行。

2. 在所有关键字和逗号之后要留一个空格，方法名之后不要留空格，紧跟左括号“（”，以与关键字区别。

3. 如果“;”不是一行的结束符号，其后要留空格，如for (initialization; condition; update)

4. 赋值操作符、比较操作符、算术操作符、逻辑操作符、位域操作符，如“=”、“+=” “>=”、“<=”、“+”、“\*”、“%”、“&&”、“||”、“<<”，“^”等二元操作符的前后要加空格。

5. 一元操作符如“!”、“~”、“++”、“--”、“&”（地址运算符）等前后不加空格。诸如“［］”、“.”、“->”这类操作符前后不加空格。

## 2.6 变量命名

1. 变量的名字须使用“名词”或者“形容词＋名词”。 变量的命名要求符合匈牙利命名法则，即开头字母用变量的类型（第一个字母小写），其余部分用变量的英文意思或其英文意思的缩写 （尽量避免用中文的拼音） ，中间单词的第一个字母要大写。 即： 变量名 =变量类型 +变量的英文意思（或缩写）。

2. 变量名称须准确、完整地描述变量的含义。循环计数变量的名称须有含义。如果循环语句的长度超过了两行或者存在着嵌套循环， 避免使用 i 、j 或 k 之类的变量，须使用有意义的变量，有利于程序的理解。（单字符的变量名一般只用于生命期非常短暂的变量）。

3. 对于所有布尔型变量的命名，能够直接从名称上看出为真的条件。枚举类型的变量名称须包含基础类型，以方便分辨变量的类型。例如，用Color 变量表示 ColorRed 和 ColorGreen 枚举类型的值。

## 2.7 常量命名

常量的命名须代表抽象的实体， 而非它们所代表的值， 即对于涉及物理状态或者含有物理意义的常量， 不允许直接使用不易理解的数字， 必须用有意义的枚举或宏来代替。所有常量名均全部大写， 由英文或其缩写， 单词间以‘ \_’隔开，如int MAX\_NUM

## 2.8注释

2.8.1 代码注释

注释将增加代码的清晰度。 注释需简洁、明了，含义准确，防止注释二义性，避免在注释中使用缩写，特别是非常用缩写。注释不允许包括其他的特殊字符。可以先写注释， 后写代码， 但对代码的注释不可放在代码下面， 须放在代码上方或右方相邻位置。

2.8.2 变量注释

1. 对于变量的注释必须紧跟在变量的后面说明变量的作用。对于所有有物理含义的变量、常量、数据结构声明 ( 包括数组、结构、类、枚举等 ) ，如果其命名不是充分自注释的，在声明时都必须加以注释，说明其物理含义。原则上对于每个变量都应该注释，但对于意义非常明显的变量，可以不注释。

2. 对于全局变量必须有较详细的注释，包括对其功能、取值范围、哪些函数或过程存取它以及存取时注意事项等进行说明。

2.8.3 函数注释

1. 对于函数，须在函数头部从函数的目的、功能、输入参数、输出参数、返回值、调用关系（函数、表）、“算法”、 “日期”等方面进行注释。不允许在一行代码或表达式的中间插入注释。在定义函数原型时，须对每一个参数加以注释。

2. 对于函数体内的分支语句（条件分支、循环语句等）必须编写注释，对于前后顺序不能颠倒的情况，建议在注释中增加序号。对于从一个 case子句进入后续的 case 子句，须用注释“ Fall through ”明确标记。当代码段较长，特别是多重嵌套时，这样做可以使代码更清晰，更便于阅读。

3. 如果部分代码不再使用，但还要保存，须使用 #if UNUSED标识。如果部分代码目前不能使用，但最后肯定要用，须使用 #if LATER 标识。使用“TODO: comment”标记未完成代码， “ BUG: comment”标记错误代码。

# 3 错误、异常处理

## 3.1 错误处理

系统在正常状态下以及无重载和硬件失效状态下，不应产生任何异常。发生异常或错误时， 对于应用服务层不允许抛出异常中断服务程序的运行， 须由系统自行处理保持程序正常运行， 必须采用日志机制来报告异常， 包括异常发生的时刻，错误说明，可能的原因、严重程度，发生异常或错误的位置等。

凡是涉及到表操作 (insert,update,select,delete) 的 sql 语句, 都必须进行错误捕捉 , 不能将错误带到后面的语句。

从表中取数的语句 , 应严格区分 NO\_DATA\_FOUND （无数据返回，用户级错误）和 TOO\_MANY\_ROWS （数据异常导致，系统级错误）的错误 , 并将相应错误信息填入错误信息。

## 3.2 异常处理

不允许使用异常实现来控制程序的正常流程， 异常实现只用于异常和错误发生时如何使程序回到正常流程。 在捕获语句的抛出异常子句中， 必须抛出原始异常，用以维护原始错误的堆栈分配。

所有存储过程 ( 函数) 在入口处统一先将返回错误代码 (errCode) 设置为 42, 出口一律在存储过程的结束部分 , 不允许中间返回，功能处理成功结束后再将错误代码 (errCode) 设置为 0( 成功), 避免程序过程中因错误未能正确捕获 ,导致功能未能完成 , 而程序却成功返回的情况出现，所有存储过程（函数）结束前应统一捕获系统异常， 所有捕获异常处都要定义 WHEN OTHER 子程序，以便捕获所有没有显示处理或其他类型的异常。

# 4 注意事项

## 4.1 变量的使用

1. 公共变量是增大模块间耦合的重要原因之一， 在开发过程中能不用公共变量尽量不要使用。

2. 临时或局部变量，使用时必须首先初始化，严禁使用未经初始化的变量作为右值。初始化类的实例时，除非十分必要，否则不要赋 null 值，在创建新对象时必须检查返回值，或者采用标准方法处理内存管理错误。

3. 尽量不要提供 public 和 protected 的成员变量，应使用属性来代替。

## 4.2 代码实现

1. 不要在程序中使用运算符的默认优先级， 应使用括号明确表达式的操作顺序。

2. 对于要返回指针但可能出错的函数， 应返回一个 BOOL值和指向参数的指针。

3. 每个接口不应该只有一个成员，最多也不应超过 20 个，尽量使每个接口中包含 3－5个成员，并且接口成员中不要包含事件。

4. 如果只有成员函数或操作符，应该使用类，而不用结构。

5. 定义类成员时，应明确定义其属性 public 、protected 或 private ，并按照此顺序分组定义。在类中不要说明全局数据成员，可以使用内嵌访问函数，以提高性能。

6. 不要使用虚函数，如多态，除非必须使用，因为虚函数的开销较大，如果使用多态，其基类的构造函数必须是虚函数。为了清楚其间，重载虚方法时，应明确说明为 virtual 。

7. 在实现构造函数时，最好不要实现复杂费时的功能，但要确保初始化所有数据成员。同时要保证构造函数、析构函数不能失败，应在一个方法中实现内存分配，或潜在的错误处理。如果部分资源在析构函数执行之前释放，要确保全部域复位，如释放对象并将指针置为 NULL，以避免多次释放出现异常。

8. 每个类最好存放在单独的文件中，并且一个文件只能有一个命名空间，避免将多个命名空间放在同一个文件里面，每个命名空间对应一个目录。