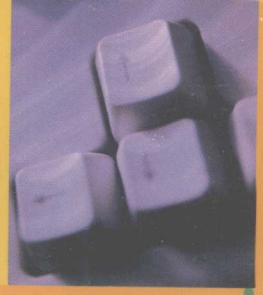
银 领 工 程

高等职业教育技能型紧缺人才培养培训工程系列教材



软件编程规范

徐人凤　孙宏伟　王　梅

高等教育出版社

高等职业教育技能型紧缺人才培养培训工程系列教材

软件编程规范

徐人凤　孙宏伟　王　梅

高等教育出版社

内容提要

本书是高等职业教育技能型紧缺人才培养培训工程系列教材，是编者在总结多年从事企业软件项目开发经验和目前一些大的软件企业的软件编程规范的基础上编写而成的。

本书内容全面、实用，按C、C++、VC++、Java、Delphi等语言分类编写，给出这些语言中关于变量命名、注释、函数/过程的书写、错误和异常处理等编写规范。本书注重培养编程人员的实际应用能力，以实例贯穿整个章节。同时，提供了很多有关编程规范方面的正例与反例，让读者对比、了解，从而提高自己的编程技能，并逐步养成良好的编程习惯。

本书可作为普通高校、高等职业学校、高等专科学校、成人高校、本科院校举办的二级职业技术学院等所有涉及软件编程课程的学生的教材和参考书，也可供示范性软件职业技术学院、继续教育学院、民办高校、技能型紧缺人才培养使用。另外，还能作为软件编程人员的参考资料和软件公司的培训教材。

图书在版编目(CIP)数据

软件编程规范/徐人凤，孙宏伟，王梅. —北京：
高等教育出版社，2005.7
ISBN 7-04-016990-8

Ⅰ.软... Ⅱ.①徐...②孙...③王... Ⅲ.软
件设计-规范-高等学校：技术学校-教材
Ⅳ.TP311.5-65

中国版本图书馆CIP数据核字(2005)第056000号

策划编辑 冯 英 **责任编辑** 张海波 **封面设计** 王凌波
版式设计 胡志萍 **责任校对** 殷 然 **责任印制** 韩 刚

出版发行	高等教育出版社	购书热线	010-58581118
社 址	北京市西城区德外大街4号	免费咨询	800-810-0598
邮政编码	100011	网 址	http://www.hep.edu.cn
总 机	010-58581000		http://www.hep.com.cn
经 销	北京蓝色畅想图书发行有限公司	网上订购	http://www.landraco.com
印 刷	廊坊市文峰档案文化用品有限公司		http://www.landraco.com.cn
开 本	787×1092 1/16	版 次	2005年7月第1版
印 张	14	印 次	2005年7月第1次印刷
字 数	330 000	定 价	17.90元

物料号 16990-00

出版说明

为了认真贯彻《国务院关于大力推进职业教育改革与发展的决定》，落实《2003—2007年教育振兴行动计划》，缓解国内劳动力市场技能型人才紧缺现状，为我国走新型工业化道路服务，自2001年10月以来，教育部在永州、武汉和无锡连续三次召开全国高等职业教育产学研经验交流会，明确了高等职业教育要“以服务为宗旨，以就业为导向，走产学研结合的发展道路”，同时明确了高等职业教育的主要任务是培养高技能人才。这类人才，既要能动脑，更要能动手，他们既不是白领，也不是蓝领，而是应用型白领，是“银领”。从而为我国高等职业教育的进一步发展指明了方向。

培养目标的变化直接带来了高等职业教育办学宗旨、教学内容与课程体系、教学方法与手段、教学管理等诸多方面的改变。与之相应，也产生了若干值得关注与研究的新课题。对此，我们组织有关高等职业院校进行了多次探讨，并从中遴选出一些较为成熟的成果，组织编写了“银领工程”丛书。本丛书围绕培养符合社会主义市场经济和全面建设小康社会发展要求的“银领”人才的这一宗旨，结合最新的教改成果，反映了最新的职业教育工作思路和发展方向，有益于固化并更好地推广这些经验和成果，很值得广大高等职业院校借鉴。我们的这一想法和做法也得到了教育部领导的肯定，教育部副部长吴启迪专门为首批“银领工程”丛书提笔作序。

我社出版的高等职业教育各专业领域技能型紧缺人才培养培训工程系列教材也将陆续纳入“银领工程”丛书系列。

“银领工程”丛书适用于高等职业学校、高等专科学校、成人高校及本科院校举办的二级职业技术学院、继续教育学院和民办高校使用。

高等教育出版社

2004年9月

自　序

软件产业是我国重点支柱产业之一，软件生产规范化、标准化同国际软件生产接轨是软件产业迅速扩大生产规模的必经之路。

印度之所以能建成拥有上万名员工的“软件工厂”，将软件作为产品，从软件生产线上生产出来，成为软件出口大国，一个非常重要的原因就是，印度的很多软件企业将质量认证作为进军国际软件市场的通行证，其软件编程实现了规范化。

我国许多软件工程师往往片面追求软件编程中算法结构的技巧，或按照自己的编程习惯书写程序代码，如程序格局、变量名的定义、大、小写字母的使用等随自己的喜欢而定。这都与国外软件行业认可的软件编程标准相违背，其结果必然造成编写出的程序代码可读性差，其产品也很难向国际市场推广。因为从规范化角度讲，这样的程序代码本身就不是产品。另外，研制一个大型的软件系统时，团队成员按照各自习惯编写出来的程序代码，在进行系统集成时，会出现很多问题，甚至使整个系统的开发前功尽弃。

看一看用微软、Borland 等公司工具自动生产的代码和具有丰富经验的程序员编写的程序，你就会发现，它确实让人感觉很“舒服”。这是为什么呢？

原因其实很简单，那就是这些程序代码遵循相同的软件编程规范，具有统一的编程风格。只有按照一定的规范进行编程，才能使编写出的程序代码能让其他人读得懂，也只有这样的程序代码才有可能进行系统集成。

我国软件产品要想走向世界，必须要遵循软件编程规范。目前，国内一些大的软件公司已经意识到这个问题，并且在公司内部建立起自己的软件编程规范。而一些中、小型软件公司即使意识到这个问题，也由于多种原因的制约无法在短期内实现软件编程的规范化。

编写本软件编程规范的目的，是为了使程序员具有良好的程序代码编写风格，将规范作为企业编程中要遵守的“法规”，以保证软件产品的高质量，提高开发效率。使用软件编程规范，可以使团队遵守相同的编程规则、具有统一的编程风格，多个人写出的程序代码看上去就像是同一个程序员编写的一样。代码易于为他人所理解，从而会在很大程度上提高代码的可维护性，也降低了以后的维护成本。

软件的生命周期中有 80% 的时间用于维护，而且维护者往往不是代码编写者本人。遵循软件编程规范编程可以提高代码的可读性，方便其他人员理解和读懂，从而在很大程度上提高代码的可维护性、降低维护成本。另外，在开发系列软件产品时，软件编程规范的应用将能保证所有的软件产品具有一致的风格。

软件编程规范的使用简化了程序员的工作，“简化”的含义不是减少代码量（相反，很多时候遵从规范会带来更多的代码），而是减少程序员在维护代码时的劳动量。另外，使用规范在很大程度上也是为了减少程序员的记忆负担。

前　言

目前，在国内各院校的软件人才培养中，涉及“如何设计和编写高质量的程序”的课程并不多，市场上相关书籍也十分匮乏。因此，教师在教学过程中以及学生在学习和编程实践过程中都很少能自觉地关注软件的质量和代码的风格。一些工作在中、小型企业的程序员，只有靠勤奋好学，并有意识地总结程序开发中的经验和教训，才能积累一些有关规范化编程的注意事项。

编写本书的目的就是为在校学生和初入软件行业的新手提供帮助，使他们掌握软件编程规范，养成良好的编程风格，以编写出高质量的程序。

本书由一条条的规则组成，基本上每条规则都是相对独立的，所以可以按任何顺序阅读或在需要时查阅，以掌握软件编程规范。

本书讲述 C、C++、VC++、Delphi、Java 等 5 种语言的软件编程规范，共八百多条规范，它由 4 个部分构成。

第一部分为 C/C++ 语言编程规范。该部分由 11 章组成。主要内容包括：程序约定；变量、常量与数据类型；表达式和基本语句；函数和过程；内存和指针；类和类函数；类的继承；可测试性；程序效率和质量保证；错误和异常处理规范以及其他规范。

第二部分为 Java 语言编程规范。该部分由 8 章组成。主要内容包括：程序组织规则；命名约定；注释约定；变量、常量；表达式和基本语句；类和类方法；代码规范以及其他规范。

第三部分为 Delphi 语言编程规范。该部分由 7 章组成。主要内容包括：程序约定；变量、常量与类型；语句；函数和过程；类和类方法；界面设计及其他规范。

第四部分为 VC++ 编程规范。该部分由 4 章组成。内容包括：编程环境设置；布局及变量；头文件、注释及其他；优化编码等。

本书附录的主要内容为 C/C++ 规范检查表、Delphi 标准控件的命名参考及 VC++ 优化编码实现过程示例。

规范并不是永远不变的，可以按照“易于使用”的原则来修改。许多大公司在某些细节上都制定了自己的规则。同时，程序设计工具对编程规范也有很大的影响，这个影响来源于开发商的程序设计风格。同样基于 C++，在 Microsoft Visual C++ 和 Borland C++ Builder 中就使用了不完全相同的编程规范。Microsoft 和 Borland 有着各自不同的、而且十分鲜明的风格。作为用户，我们可以在此基础上有所改变，但是这是有限度的。其实，在用户做出对供应商和开发工具的选择时，同时也就确定了其开发程序的风格。

另外需要强调的是，本书源于有关程序开发组织的约定及许多程序员的经验，所以本书试图为读者提供一个参考，希望读者在理解每一条规则的基础上，在自己的工作实践中加以灵活运用。

关于本书的使用方式，建议有以下三种方法：

第一种，作为软件技术专业、网络技术专业、计算机应用等专业学生的必读教材。在学习 C 语言程序设计、C++、Delphi、Java 时，可随时学习和查阅。

第二种，可作为有关选修课程的教材，建议学时为32学时。

第三种，也可作为软件企业的编程人员的工具书，在学习和编程实践中随时翻阅。

在本书编写的过程中，得到许多人的热情支持和帮助。

在此非常感谢聂哲先生和袁梅冷女士，他们两位对Java语言精髓的深入理解和编程经验，对本书Java语言部分进行了认真的审核，提出了许多宝贵的建议。

同时也要感谢李俊平和曾建华两位先生，他们多年来对Delphi语言的使用积累了非常丰富的软件开发经验，并提供给本书，无私地与大家分享。

在这里，也要感谢钟剑龙先生，他为本书提供了部分素材。

最后，非常感谢李俊平先生，他在繁忙的工作中抽出时间审阅了全稿。

真诚感谢在本书编写过程中，给予我们帮助和支持的家人和朋友，感谢他们的理解和配合。

本书由深圳职业技术学院徐人凤（高级工程师）、孙宏伟（工程师）、王梅（讲师）共同编写，由徐人凤对全书进行了统稿。徐人凤和孙宏伟均具有多年在企业从事软件开发的实际经验。

由于时间仓促和编者水平有限，书中难免有错漏之处，敬请读者批评指正，在此表示诚挚的谢意。编者E-Mail:xurf@oa.szpt.net。

编　者

2005年3月于深圳

目　录

第一部分　C/C++ 语言编程规范

第二部分　Java 语言编程规范

第三部分　Delphi 语言编程规范

第四部分 VC++ 编程规范

第一部分

C/C++ 语言编程规范

第1章 程序约定

通常，一个 C/C++ 程序分为两个文件。一个文件用于保存程序的声明(declaration)代码，称为头文件。另一个文件用于保存程序的实现(implementation)代码，称为定义文件。

C/C++ 程序的头文件以“.H”为后缀，C 程序的定义文件以“.C”为后缀，C++ 程序的定义文件通常以“.CPP”为后缀(也有一些系统以“.CC”或“.CXX”为后缀)。

1.1 程序排版布局规则

程序排版布局的目的是显示出程序良好的逻辑结构，提高程序的准确性、连续性、可读性和可维护性。更重要的是，统一的程序布局和编程风格，有助于提高整个软件的开发质量，提高开发效率，降低开发成本。同时，对于普通程序员来说，养成良好的编程习惯有助于提高自己的编程水平和编程效率。

【规则 1-1-1】程序块应采用缩进格式编写，缩进的空格数为 4 个。

说明 对于由开发工具自动生成的代码可不遵循此规则。

【规则 1-1-2】程序的分界符“{”和“}”应独占一行并且位于同一列，同时与引用它们的语句左对齐。{ }之内的代码块使用缩进规则对齐。

说明 这样能使代码便于阅读，并且方便注释。但 do while 语句和结构的类型化时可以例外，while 条件和结构名可与“}”在同一行。

反例

```
void Func (int iVar){
    while (condition){
        DoSomething( );
    }}
```

正例

```
void Func (int iVar)
{                                   // 独占一行并与引用语句左对齐
    while (condition)
    {
        DoSomething( );     // 以{、}为准缩进 4 格
    }
}
```

【规则 1-1-3】一行代码只完成一个功能，即一个变量定义占一行，一个语句占一行。

反例

```
double width, height, depth;        // 定义宽度、高度及深度变量
```

正例

```
double width;     // 定义宽度变量
double height;    // 定义高度变量
double depth;     // 定义深度变量
```

【规则 1-1-4】相对独立的程序块之间、变量说明之后必须加空行。

反例

```
if (a > b)
{
        … // program code
          max = a;
}
repssn_ind = ssn_data[index].repssn_index;
repssn_ni = ssn_data[index].ni;
```

正例

```
if (!valid_ni(ni))
{
        … // program code
}

repssn_ind = ssn_data[index].repssn_index;
repssn_ni = ssn_data[index].ni;
```

【规则 1-1-5】程序中一行代码及其注释的总字符数不能超过 80 列。

说明　包括空格在内的该行总字符数不超过 80 列。

【规则 1-1-6】较长的语句（超过 80 字符）要分成多行书写，长表达式应在低优先级操作符处拆分新行，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。

说明　① 条件表达式的续行在第一个条件处对齐。

② for 循环语句的续行在初始化条件语句处对齐。

③ 函数调用和函数声明的续行在第一个参数处对齐。

④ 赋值语句的续行应在赋值号处对齐。

正例1

```
perm_count_msg.head.len = NO7_TO_STAT_PERM_COUNT_LEN
                          + STAT_SIZE_PER_FRAM * sizeof( _UL );

act_task_table[frame_id * STAT_TASK_CHECK_NUMBER + index].occupied
        = stat_poi[index].occupied;

act_task_table[taskno].duration_true_or_false
        = SYS_get_sccp_statistic_state( stat_item );

report_or_not_flag = ((taskno < MAX_ACT_TASK_NUMBER)
                      && (n7stat_stat_item_valid (stat_item))
                      && (act_task_table[taskno].result_data ! = 0));
```

正例2

```
if ((iFormat == CH_A_Format_M)
    && (iOfficeType == CH_BSC_M)) // 条件表达式的续行在第一个条件处对齐
{
    DoSomething();
}

for (long_initialization_statement;
     long_condiction_statement;          // for循环语句续行在初始化条件语句处对齐
     long_update_statement)
{
    DoSomething();
}

// 函数声明的续行在第一个参数处对齐
BYTE ReportStatusCheckPara(HWND hWnd,
                           BYTE ucCallNo,
                           BYTE ucStatusReportNo);

// 赋值语句的续行应在赋值号处对齐
fTotalBill = fTotalBill + faCustomerPurchases[iID]
           + fSalesTax(faCustomerPurchases[iID]);
```

【规则1-1-7】循环、判断等语句中若有较长的表达式或语句，则要进行适当的拆分，长表达式应在低优先级操作符处拆分成新行，操作符放在新行之首。

正例

```
if ((taskno < max_act_task_number)
        && (n7stat_stat_item_valid (stat_item)))
{
        … // program code
}

for (i =0, j = 0; (i < BufferKeyword[word_index].word_length)
                        && (j < NewKeyword.word_length); i++, j++)
{
        … // program code
}

for (i = 0, j = 0;
        (i < first_word_length) && (j < second_word_length);
        i++, j++)
{
        … // program code
}
```

【规则 1-1-8】定义指针类型的变量时，应使“*”紧挨着变量名。对“&”也采取同样的处理规则。

反例

```
float *    pfBuffer;
```

正例

```
float    *pfBuffer;
int    *x, y; // 此处 y 不会被误解为指针
```

【规则 1-1-9】若函数或过程中的参数较长，则要进行适当的拆分。

正例

```
n7stat_str_compare((BYTE *) & stat_object,
                        (BYTE *) & (act_task_table[taskno].stat_object),
                        sizeof (_STAT_OBJECT));

n7stat_flash_act_duration( stat_item, frame_id * STAT_TASK_CHECK_NUMBER
+ index, stat_object );
```

【规则 1-1-10】不允许把多个短语句写在一行中，即一行只写一条语句。

反例

```
rect. length = 0; rect. width = 0;
```

正例

```
rect. length = 0;
rect. width = 0;
```

【规则 1-1-11】 if、else、else if、for、while、do、case、switch、default 等语句独占一行，执行语句不得紧跟其后。不论执行语句有多少条都要加{}。

说明　这样可以防止书写错误，也易于阅读。

反例　下面的代码执行语句紧跟 if 的条件之后，而且没有加{}，违反规则。

```
if (width < height) dosomething();
```

正例

```
if (width < height)
{
dosomething();
}
```

【规则 1-1-12】 源程序中关系较为紧密的代码应尽可能相邻。

说明　这样便于程序阅读和查找。

反例

```
iLength = 10;
strCaption = "Test";
iWidth = 5;
```

正例

```
iLength = 10;
iWidth = 5;      // 矩形的长与宽关系较密切，放在一起
StrCaption = "Test";
```

【规则 1-1-13】 尽可能在定义变量的同时初始化该变量。

说明　如果变量的引用处和其定义处相隔比较远，变量的初始化工作很容易被忘记。如果引用了未被初始化的变量，可能导致程序错误。

正例

```
int width = 10; // 定义并初始化 width
int height = 10; // 定义并初始化 height
int depth = 10; // 定义并初始化 depth
```

【规则 1-1-14】 禁止使用 Tab 键，必须使用空格键进行缩进。缩进量为 4 个空格。

说明　在使用不同的编辑器阅读程序时，应避免使用 Tab 键来设置空格，以免因空格数目的

不同而造成程序布局的不整齐。不要使用 BC 作为编辑器，因为 BC 会自动将 8 个空格变为一个 Tab 键，因此使用 BC 编辑器时会使程序缩进变乱。有的代码编辑器可以设置用空格代替 Tab 键。

【规则 1-1-15】函数或过程的开始、结构的定义及循环、判断等语句中的代码都要采用缩进格式。在 switch 语句中，每一个 case 分支和 default 都应用{ }括起来，而且{ }中的内容也需要缩进。

说明　这样做的目的是使程序可读性更好。

正例

```
switch (iCode)
{
    case 1:
    {
        DoSomething();      // 缩进 4 格
        break;
    }
    case 2:
    {                       // 每一个 case 分支和 default 要用{ }括起来
        DoOtherThing();
        break;
    }
    …                       // 其他 case 分支
    default:
    {
        DoNothing();
        break;
    }
}
```

【规则 1-1-16】声明类的时候，public、protected、private 等关键字应与分界符“{”、“}”对齐，这些部分的内容要进行缩进处理。

正例

```
class CCount
{
public:                         // 与 { 对齐
    CCount (void);              // 缩进处理
    ~CCount (void);
    int GetCount(void);
    void SetCount(int iCount);
```

```
private:
    int m_iCount;
}
```

【规则1-1-17】在对两个以上的关键字、变量、常量进行对等操作时，它们之间的操作符之前、之后或者前后都要加空格；进行非对等操作时，如果是关系密切的立即操作符（如->），后面不应加空格。

说明　采用这种松散方式编写代码的目的是使代码更加清晰。由于留空格所产生的清晰性是相对的，所以，在已经非常清晰的语句中没有必要再留空格。如果语句已足够清晰，则括号内侧（即左括号后面和右括号前面）不需要加空格，多重括号间不必加空格，因为在C/C++语言中括号已经是最清晰的标志了。在长语句中，如果需要加的空格非常多，那么应该保持整体清晰，而在局部不加空格。给操作符留空格时不要连续留两个以上的空格。

正例　① 只能在逗号、分号的后面加空格。

```
int a, b, c;
```

② 在比较操作符、赋值操作符（“=”、“+=”）、算术操作符（“+”、“%”）、逻辑操作符（“&&”、“&”）及位域操作符（“<<”、“^”）等双目操作符的前后都应加空格。

```
if (current_time >= MAX_TIME_VALUE)
a = b + c;
a *= 2;
a = b ^ 2;
```

③ 应在if、for、while、switch等关键字与后面的括号间加空格，这样可使关键字更为突出、明显。

```
if (a >= b && c > d)
```

④ 应在const、virtual、inline、case等关键字之后至少留一个空格，否则无法辨认这些关键字。

【规则1-1-18】对于结构型数组及多维数组，如果在定义时进行初始化，则应按照数组的矩阵结构分行书写。

正例

```
int aiNumbers[4][3] =
{
        1, 1, 1,
        2, 4, 8,
        3, 9, 27,
        4, 16, 64
}
```

【规则1-1-19】相关的赋值语句等号应对齐。

正例

```
tPDBRes.wHead    =  0;
```