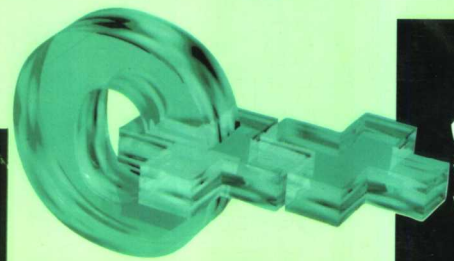
王世同 李 强 等 编著

C++

Visual

# Visual C++ 6.0

# 编程基础





清华大学出版社
http://www.tup.tsinghua.edu.cn

(京)新登字 158 号

## 内 容 简 介

本书介绍了 Microsoft 公司的 Visual C++ 6.0 的程序设计技术，内容包括：Visual C++ 6.0概述，创建简单的应用程序，制作编辑框、滚动条、复选框、单选按钮、组合框、菜单、对话框、工具条和状态条、单文档/多文档界面应用程序，创建并使用动态链接库，ActiveX 控件基本知识，MFC 类库一览，使用 MFC 编写 ActiveX 控件，程序调试技术等。

本书内容精练、循序渐进、实用性强，主要通过实例说明如何使用 Visual C++ 6.0，并插入适量图片，使得内容精确、完整，即使是初学者也不难掌握。



**图书在版编目(CIP)数据**

Visual C++ 6.0 编程基础/王世同等编著 .—北京：清华大学出版社，1999.6
ISBN 7-302-03578-4

Ⅰ.V… Ⅱ.王… Ⅲ.C 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(1999)第 19506 号

**出版者**：清华大学出版社(北京清华大学校内，邮编 100084)
http://www.tup.tsinghua.edu.cn
**印刷者**：北京市清华园胶印厂
**发行者**：新华书店总店北京发行所
**开　本**：787×1092 1/16　**印张**：14.75　**字数**：347 千字
**版　次**：1999 年 6 月第 1 版　1999 年 10 月第 2 次印刷
**书　号**：ISBN 7-302-03578-4/TP・1971
**印　数**：6001～14000
**定　价**：19.80 元

# 前　言

Microsoft 公司于 1998 年推出了一套软件开发工具集——Microsoft Visual Studio 6（Microsoft 可视化创作室，也称 Microsoft Visual Studio 98），Visual C++ 6.0 就是其中的一个套件，它是一个彻底的程序员级的开发环境，“可视化”的设计减少了不少编程的工作量。利用 Visual C++ 6.0 几乎可以设计一切，小至应用系统，大至开发工具。作为 Windows 环境下优秀、正宗的 C++ 编译器的 Visual C++ 6.0 进一步成为软件开发人员强有力的工具。Visual C++ 是汇集 Microsoft 公司技术精华的主流产品，它不仅全面贯彻了面向对象技术，而且在编译优化技术方面较其他同类产品具有明显的优势。

Visual C++ 6.0 在用户界面上最大、最受欢迎的改进就是引入智能感应（IntelliSense）技术。该技术最早出现在 Visual Basic 5.0 中，它可以根据编辑时代码的输入状态自动把属性、参数信息、数据类型信息和代码信息显示在一个列表框中，供开发者选择并自动完成单词的输入，或者给开发者以提示。由于在使用 Visual C++ 编程时需要用到很多类及其成员和 Win32 函数，以往开发者不得不记忆大量的信息，或者频繁地查阅联机帮助，严重影响了编程的效率。现在有了智能感应技术后，开发者可以摆脱一些琐碎的细节问题，把精力更好地集中在程序设计之上，从而提高了工作效率。

在 Visual C++ 这个集成环境中开发应用程序时，不一定要手工设计用户界面，而只需选取菜单命令，Visual C++ 系统就会生成一个可实际运行的 Windows 应用程序框架。此后，程序员就可利用基于 Windows 的 C++ 编辑器，借助 AppWizard 建立面向对象的应用程序。

除了 AppWizard 外，Visual C++ 还包含 C++ 应用程序生成器、基于 Windows 的编辑器、基于 Windows 的面向图形的编程工具、使 C++ 代码和 Windows 消息及类成员函数相联系的交互式工具、可用来编写 Visual C++ 文本程序的 QuickWin 库以及预编译的头文件和源文件。当然，类库丰富更是 Visual C++ 的最大特色之一。

参与本书编写的有王世同、李强、宗毅、郑洁、胡荣华、张爱玲、周爽、徐卫、李毅军、王朋、陈敏等；本书由杨国雍、赵明华策划；徐力军负责资料整理；全书由王世同和李查德审校；录入排版工作由张强、邱海燕负责。

作者

1999 年 2 月

# 目　录

# 第1章 Visual C++ 6.0概述

众所周知，Visual C++是目前开发效率最高的C++ 系统。使用Visual C++可以编写出性能卓越的Windows和Web应用程序。Visual C++拥有一个优秀的集成开发环境，集编辑、编译、连接、调试、向导等多项功能于一体，并且提供了目前已成为业界标准的MFC(Microsoft Application Foundation Classes)类库。

更令诸多喜爱Visual C++的用户兴奋的是，新近推出的Visual C++ 6.0，在保持了5.0版本优点的基础上，在用户界面、编译调试技术、语言特性等方面都有了新的改进和发展。

## 1.1 Visual C++ 6.0的新特点

1. 引入智能感应技术

Visual C++ 6.0在用户界面上最大、最受欢迎的改进就是引入智能感应(IntelliSense)技术。该技术最早出现在Visual Basic 5.0中，它可以根据编辑时代码的输入状态自动把属性、参数信息、数据类型信息和代码信息显示在一个列表框中，供开发者选择并自动完成单词的输入，或者给开发者以提示。由于在使用Visual C++编程时需要用到很多类及其成员和Win32函数，以往开发者不得不记忆大量的信息，或者频繁地查阅联机帮助，严重地影响了编程的效率。现在有了智能感应技术后，开发者可以摆脱一些琐碎的细节问题，把精力更好地集中在程序设计之上，从而提高工作效率。

2. 支持动态更新视图

在类中增加一个变量或一个方法时，会立刻反映在类视图之中。

3. 支持多显示器

在Windows 98或Windows NT 5.0系统中，开发者可以在一个显示器上运行程序，在另一个显示器上调试代码，或者把集成开发环境中数量众多的工具条或窗口拖动放置在不同的显示器上，以扩大各个窗口的视野范围。

4. 联机帮助

在新的HTML帮助系统中，联机帮助从集成开发环境中独立出来了，与其他Visual Tools的帮助一起放在了MSDN文库之中。Visual Studio 6.0所带的MSDN文库以Compiled HTML的形式存在，可由集成开发环境在需要时调用。

5. 提高编译效率

与5.0版相比，Visual C++ 6.0的编译效率更高，调试版工程效率提高了30%，非调

试版工程效率提高了15%。通过引入4个新的关键字_assume、inline、__inline和_forceinline,它允许开发者灵活地控制编译时代码的优化过程,从而达到更好的优化效果。

6. 先进的调试技术

在调试技术方面,Visual C++ 6.0引入了另一项深受好评的技术:即编即调(Edit and Continue)。程序员可以在调试过程中编辑源代码,而无需像以前那样,先退出调试状态,编辑,重新编译连接,重启动调试器,再跟踪到问题发生的地方。调试过程的缺省设置支持即编即调,当更改源代码之后,集成环境会自动编译修改过的部分,在可能的情况下直接修改处于断点状态的可执行文件,然后继续运行该程序。不过,当修改涉及到全局变量、类定义等影响程序整体的地方,则必须重新生成一次可执行文件,即编即调在此就无效了。总的来说,即编即调技术能够有效地减少调试所需时间,提高开发者的工作效率。Visual C++ 6.0还支持对DLL的延迟加载,除了必须加载才能继续执行下去这种情况之外,这样可以提高应用程序的运行速度。

7. 壮大的MFC类库

Visual C++ 6.0的MFC类库新增了11个类,并对22个类进行了修改。新类COleDocObjectItem提供了对Active Document Containment的支持;CHtmlView封装了DHTML控件,全面支持动态HTML,可用于开发Web浏览器式的应用程序;CReBar,CRebarCtrl,CComboBoxEx,CDateTimeCtrl,CIPAddressCtrl和CMonthCalCtrl则封装了IE 4所引入的一些通用控件。

8. 集成的数据库功能

在数据库方面,Visual C++ 6.0引入了OLE DB(OLE数据库)供求双方模板、ADO数据绑定对话向导、OLE DB和ADO的数据绑定控件等新特性和新功能。

## 1.2 Visual C++ 6.0——面向对象的程序设计语言

Visual C++是一种面向对象的程序设计语言。要想充分理解这种程序设计语言的优势,需要将它与传统的结构化程序语言加以比较。

### 1.2.1 传统的结构化程序设计模式

结构化程序设计(Structured Programming),即SP模式,在过去的几十年中,一直是程序开发设计的主流思想。具体可以这样解释:首先为解决某个实际问题而确定一个算法,然后为该算法构造适当的数据结构,通过算法的操作过程体现算法的思想。也就是说程序是在数据的某种特定的表示方式和结构的基础上,对抽象算法的具体实现。

这种程序设计方法力求算法描述准确,对每一个子模块容易进行程序正确性证明,对数据进行编译检查在一定程度上增强了程序的可靠性,相对于以前的标准编程来说,确实是一个很大的进步。但是,在本质上这种程序设计方法是面向过程的,不能直接反映人们解决问题的思路,因此这种模式存在固有的缺陷,主要表现在:

1. 程序的可重用性差

现在的应用程序变得越来越大,越来越复杂,但其中有很多重复性的工作,代码重用

成为提高效率的关键。而采用传统的SP模式，每次程序员进行一个新系统的开发，几乎要从零开始做起，并且要针对具体问题做大量重复而繁琐的工作。即使重用代码，也只是进行简单的拷贝，若稍有不同的话，还必须逐字修改。也就是说，这种模式不能直接继承引用已编好的应用程序的某些部分。

2. 维护程序一致性差

传统的SP模式是面向过程的模式，数据和方法是分开的，这样很可能产生问题空间和方法空间在结构上的不一致。对程序运行起重要作用的数据一般要做全局处理，若为了新的需要而改变某一数据结构，则所有处理数据的过程要重新考虑，才能保证与数据的一致性。这样的话，维护数据和过程的一致性要花费大量的精力，而且还可能产生不少错误。

### 1.2.2 面向对象的程序设计模式

为了克服结构化程序设计的这些弊端，产生了面向对象的程序设计模式(Object-Oriented Programming)，即OOP模式。它摆脱了传统过程模式的束缚，使应用程序开发跃上了一个新的台阶。

1. 对象的概念

在面向对象模式中，将“对象”作为系统中最基本的运行实体，“对象”中封装了描述该对象的特殊属性(数据)和行为方式(方法)。整个应用程序由许多不同类型的对象组成，各对象既是独立的实体，又可以通过消息相互作用，对象中的方法决定要向哪个对象发消息，发什么消息，以及收到消息后如何处理等。

也就是说，面向对象模式以对象或数据为中心，这时的数据与传统的被动的数据不同，它具有“行动”的功能，而这种行动是在对象接到消息时发生的。由于对象反映了应用领域的模块性，具有一定的稳定性，因此可以被当作一个组件，去构成更复杂的应用。又由于对象一般封装的是某一具体的实际工作的各种成分，因此某一对象改变时，对整个系统几乎没有影响。

为了描述功能相近的对象，引入了“类(class)”的概念。类与类之间的关系是层次结构，属于某个类的对象除了具有该类的全部特征外，还具有层次结构中该类上层所有类的全部性质，这种机制称为继承。

面向对象程序设计的方法的继承性和模块性，使得新的应用程序可以在原来对象的基础上，通过重用和扩展来进行，而不必从头做起或者拷贝原有代码。这就大大提高了程序开发的效率，减少了重新编写新代码的工作量，同时降低了程序设计过程中出错的可能性。

2. 对象、方法和消息之间的关系

面向对象的程序是由若干“对象”有机结合而成。这里的“对象”指的是将某些数据代码和对该数据的操作代码封装起来的模块，是有着特殊属性(数据)和行为方式(方法)的逻辑实体。

对象包含了数据和方法，每个对象就是一个微小的程序。由于其他对象不能直接操

纵该对象的私有数据，只有对象自身的方法才能得到它，这就使得对象具有很强的独立性。因此，可以把对象当作应用程序的基本组件，就像电器中的电子元件一样，可以使用若干元件组成不同的电路。在面向对象的程序设计中，可使用若干对象来建立所需要的各种复杂的应用软件，即通过对象组合，创建具体的应用。

对象的这种应用程序组件作用使它具有很强的可重用性，用户在开发应用程序时可以调用系统库中的各种对象，作为自己的应用程序的基本组件。这将使新增代码明显减少，而且还增加了程序的可靠性和可维护性。

在面向对象的程序设计中是通过消息来请求对象进行动作的，对象间的联系（或称相互作用）也是通过消息来完成的。消息中只包括消息发送者的要求，不指示接收者具体该如何去处理这些消息。对象接收一个消息后该如何动作的过程称为“方法”，接收者所含的方法决定该对象如何处理消息。

一个对象可以接收不同形式、不同内容的多个消息；相同形式的消息可以送给不同的对象；不同的对象对于形式相同的消息可以有不同的解释，做出不同的反应。因此只要给出对象的所有消息模式及对应于每一个消息的处理方法，也就定义了一个对象的外部特征。

3. 类

类是对具有公共的方法和一般特性的一组基本相同对象的描述。在面向对象程序设计中，对象是构成程序的基本单位。每个对象都应该属于某一类，这好比传统程序设计语言中，每个变量都应具有一定的类型 。一个类定义的是一种对象类型，由数据和方法组成，描述了属于该类型的所有对象的性质。

在程序执行过程中，由类动态生成相应的对象。一个类可以生成多个不同的对象，这些对象具有相同的属性，当然，由于对象所接收的消息形式可以不同，它们可能有不同的内部状态。对象也可称为类的一个实例。

一个类可以由其他已存在的类派生出来，类与类之间按具体情况以层次结构组织起来，在这种层次结构中，处于上层的类称为父类，处于下层的类称为子类或派生类。

例如，可将类按下列层次关系组织起来（见图 1.1）。

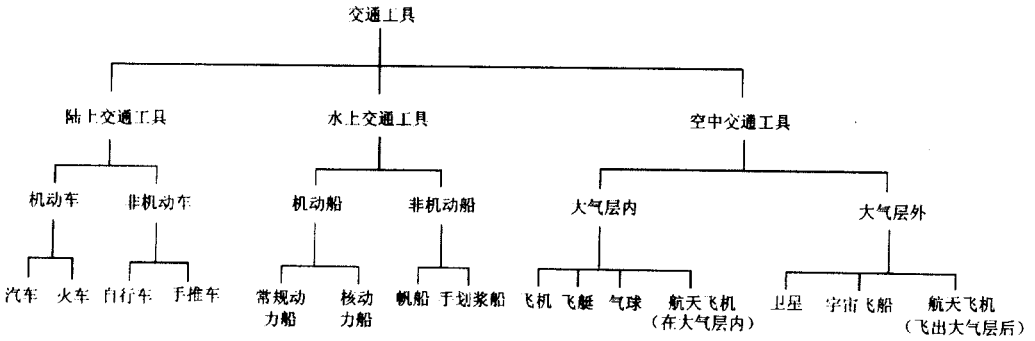



**图 1.1　类层次的例子**

父类、子类之间具有继承性，一个子类可自动继承其父类的全部描述，而这种继承性

具有传递性，任何一个类均继承了层次结构中上层所有类的全部特性。从这个意义上讲，一个子类既是由新增的属性和操作对父类进行的补充和改进，也是父类特性的具体化和完善。如图 1.1 的层次关系中，每一个子类都是对其父类的进一步说明，都比父类更具体。而对于这一层次，又可以细分出它们的子类。

在类的层次结构中，一个类可以有多个子类。如果要求一个类至多有一个父类，则一个类至多只能直接继承某一类的方法和数据，这种继承方式称为简单继承（或单一继承）；如果允许一个类有多个父类，则一个子类可以从多个父类中得到方法和数据，这种继承为多重继承。

使用面向对象的程序设计语言进行编程时，总会用到系统提供的类库，类库是用于特定程序任务的类的集合，熟悉和掌握类库对程序员很关键。通常情况下类库中定义的类可分为两种类型，一种是通用类，它们可以被几乎所有的应用程序所用。这些类主要包括一些处理字符串、数组、链表、队列等数据结构的子类。另一种类是专门为某种应用而设计的。如类库中与对象链接与嵌入（OLE）有关的类。程序员创建新类其实就是对那些被精心测试过的类库中提供的类进行扩充，以满足应用程序的特殊要求，而对父类中已有的描述不必重新定义。或者说程序员可以从已存在的类中继承数据和方法，设计子类时只考虑与父类不同的地方，在不更改父类的情况下，增强原程序的功能。继承也是面向对象程序设计方法的一个重要机制，传统语言不支持继承。

### 1.2.3 Visual C++ 6.0 的编程特征

1. 封装性

封装指的是方法和数据放在同一个对象中，对数据的存取只能通过该对象本身的方法来进行。其他的对象不能直接作用于该对象中的数据，对象间的相互作用只能通过消息进行。

2. 继承性

继承性指的是一个新类可以从现有的类中派生出来，新类具有父类中所有的特性，直接继承了父类的方法和数据，新类的对象可以调用该类及父类的成员变量和成员函数。

3. 多态性

多态性指的是同一个消息被不同的对象接收时解释为不同意义的能力。也就是说，同样的消息被不同的类对象接收时产生完全不同的行为。利用多态性，用户可发送一般形式的消息，而将所有实现的细节留给接收消息的对象去解决。

## 1.3 建立开发 Visual C++ 6.0 应用程序的整体概念

本节将展示使用 Visual C++ 6.0 开发面向对象应用程序的整体概念。无论是初学者，还是编程老手，都会从中得到不少启示，并可以将其直接应用到以后的开发工作中。

### 1.3.1 Visual C++6.0 应用程序的组成

通常，编写 Visual C++ 6.0 应用程序包括以下两个部分：

1. 可视化编程部分

可视化编程部分利用 Visual C++ 6.0 提供的强大的软件开发工具向导，不需要手工编写代码，只要根据自己的设计思想，用鼠标或键盘进行操作即可。可视化编程部分主要用于制作 Windows 风格的图形用户界面和各种控件，如果用代码编写相应的内容，将是一件很困难和繁琐的事情。

2. 代码编程部分

代码编程部分使用 Visual C++ 6.0 提供的文本编辑器，用面向对象的 Visual C++ 编程语言来编写。

### 1.3.2 面向对象应用程序的维护

用面向对象的方法开发应用程序已渐成潮流。人们也普遍认为面向对象应用程序的维护应不成问题，但事实却并非如此。随着面向对象技术的广泛使用，面向对象应用程序不易维护（原因是维护者不易分析、理解这类应用程序）的问题已越来越突出。下面将在讨论面向对象技术对应用程序维护影响的基础上，探讨面向对象应用程序的维护问题，并提出解决的对策。

所有应用程序都会有一个代价高昂的维护阶段。很多人的经验表明：少数维护是改正性维护，多数维护属适应性（或改善性）维护。不管一个应用程序有多么好，用户都会要求有更强的功能、更好的适应性。多年的实践表明：只有不断的维护，应用程序才有生命力。

同所有应用程序一样，面向对象应用程序的维护需要两个基本条件：

（1）待维护的应用程序可以理解。

（2）待维护的应用程序可以修改。

理解程序的关键问题是维护人员能重构“源设计图”，并掌握原设计人的设计思想和方法、策略。如果这些思想、方法、策略都集中在一段不大的程序里，那么理解起来还容易一些。但用面向对象技术开发出来的程序往往存在于不连续的程序段中（比如一个方法可能有一段程序，而一个对象中可能有多个各自独立的方法），这时维护者就容易产生错误的理解，并据此做出错误的修改。

面向对象技术的要点，就是把问题抽象成各个对象并封装它们。对象内部的消息传递机制、靠消息激活方法的手段，以及对象间的并行、继承、传递、激活等特性，必然会导致应用程序各部分之间存在大量的复杂关系。这些面向对象所固有的特色，使程序员们已习惯的阅读、分析、理解程序的方法失去了作用，这会大大增加理解应用程序的难度。

而对一个已理解应用程序的修改却容易得多，因为面向对象应用程序中的对象能够比较完整地反映客观事物的静态属性和动态行为，继承机制又可以使修改量降至最低，而且对象封装具有屏蔽作用（对一个对象的修改不影响其他对象）。

综上所述，我们可以得知：面向对象的应用程序不易理解，但理解之后容易修改（与其他类型的应用程序相比）。

### 1.3.3 开发面向对象应用程序的一般过程

只有充分理解面向对象应用程序的开发方法，才能更好地对其进行维护，而重构“源设计图”则是理解的基础。在重构“源设计图”时，因为对象的多态性、继承机制以及动态连接等特性，给理解原设计增加了困难。为了日后维护的方便，应在开发时就使应用程序的设计思想易于理解（但由于开发中难以避免的变动，很难保证最终应用程序的易理解性），应用程序开发环境也应提供能帮助维护者理解原应用程序设计思路和策略的工具（眼下这种工具太少）。

按一般习惯理解一个程序段时，应先完成对程序的静态分析，在查阅文档及程序代码的同时，要努力搞清这个程序段的用途及设计思想。维护人员需要查清这个程序段从何处被调用，这段代码所调用的其他过程、调用的前提条件、这些过程的功能及数据流动情况，这样才能真正了解、掌握这段代码。对于面向对象的代码，还必须考虑它的一些特殊点。

为了理解面向对象的应用程序，应在获取该应用程序文档的前提下，分析、研究各对象的组织结构，以及各对象间的相互作用关系，在此基础上逐步识别出程序中的对象、对象间的关系，并掌握各对象中各方法被激活的条件、消息的来源及传递的途径。

### 1.3.4 使应用程序中的继承关系清晰化

为了理解程序，就必须搞清楚程序的依赖性（包括其中的调用及数据流的相互关系）。面向对象程序中广泛使用的继承机制为查找和分析程序的依赖性增加了困难。在分析一段程序时，必须考虑它是否存在于某些特定的上下文之中，以及是否依赖于某个（些）类中的方法。让人头痛的是，这个方法有可能是由其子类的一个成员所操纵的，而操纵该方法的实例变元既可以在类中进行说明，也可以在其子类的一个成员中进行说明。

在面向对象应用程序的开发过程中，随着新对象的不断增加，继承关系的层次也可能会增加，这往往会使继承关系变得更加复杂。例如，维护人员如果需要检查一个消息发送，就需要检查若干层，才能确定该消息的接收者（即某个对象）。这种检查无疑增加了理解程序的难度。

在开发应用程序时，为避免减弱数据的封装性，同时保证类的功能，程序员很自然地把对象内的方法做得短小而简单，小方法的数量因此而大大增加。这些大量存在的小方法往往使得系统中的继承关系链变得很长，显然，这也会增加分析的难度。

### 1.3.5 正确对待应用程序中的动态链接及多态性

对一个应用程序来说，多态性和动态链接具有两面性：优点是使程序具有灵活性（这是面向对象设计的目标之一，也是其特有的技术魅力）；缺点是给人们理解应用程序带来了困难，从而增加了维护的难度。

动态链接及多态性会给应用程序维护带来分析上的困难。如果只使用静态链接，而且代码的上下文提供了相应的信息，人们可以据此推导出相应的类，但仍有被误导的可能。因为处理一个消息可以用库中的多种方法，一个人又很难记住这些方法对消息的响

应,这时出现理解失误的可能性极大。

其实,产生困难的关键是各对象的语义。如果开发者能保证多态一致性,使所有方法对某个消息都产生相同的反映,那么问题就会少一些。但如果方法的名字不一样,维护者仍会产生误解,因为一个方法有少量不同的行为,它们会改变整个代码段的含义。

维护人员可以通过比较"外部依赖图"来发现这种不一致。外部依赖图是一种比较有效的检测工具。当一个方法被启动时,它可以自动地用数据流图的方法来确定可能建立起来的数据项依赖关系。若设计人员能正确地使用多态性,那么具有相同名字的方法就会有相同形式的外部依赖图,这样应用程序工具就能比较由相同消息启动的方法的外部依赖图,并对不同点作标记,方便维护人员的分析。

### 1.3.6 使用 Visual C++ 6.0 开发应用程序应注意的问题

总的来看,面向对象的方法具有很多优点,但在维护上却产生了一些难点,主要是增加了维护者理解、分析应用程序的难度。为了减轻面向对象应用程序维护的难度,综上所述,建议采用以下措施:

(1) 应用程序开发人员在使用面向对象的某些技术(如继承、动态链接等)时要特别小心。因为这些应用程序机制类似于传统方法中的 GOTO 语句,虽然有其好处,但使用不当也会带来维护及调试上的困难。

(2) 开发人员要在文档中作好记录,特别要记录好具有密切关系的类的活动及测试过程,使文档能尽量全面地反映应用程序的情况。