Visual Basic 高级编程技术

——从 VB 6.0 向 VB.NET 过渡

李鸿吉　编著

科 学 出 版 社

北　京

内 容 简 介

本书通过大量的示例，按近年来的程序设计思想，系统地介绍了面向对象、ActiveX、OLE、Windows API、多媒体、数据库、资源文件、注册表、串行通信、Internet 等较有深度的编程方法。同时本书还着重探讨了如何从 VB 6.0 向 VB.NET 的过渡以及使用升级向导的种种问题，并以示例的形式介绍了 VB.NET 的继承性实现和 VB.NET 调用 VB 6.0 组件的方法。本书在结构和内容的编排上注重深入浅出、循序渐进。

本书可以作为大专院校、培训班的教学参考书，也可以作为有 VB 开发经验的专业人士的提高读物和工具书。对于承担项目的科研人员、教学人员以及研究生，在将科技成果软件化时本书是很有实用价值的参考资料。

图书在版编目（CIP）数据

Visual Basic 高级编程技术：从 VB 6.0 向 VB.NET 过渡/李鸿吉编著.
—北京：科学出版社，2003
ISBN 7-03-011009-9
I . V… II . 李… III . BASIC 语言—程序设计 IV. TP312
中国版本图书馆 CIP 数据核字（2002）第 094184 号

责任编辑：孟战龙／责任校对：都 岚
责任印制：吕春珉／封面设计：王 浩

科 学 出 版 社出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

印刷
科学出版社总发行 各地新华书店经销
*
2003 年 1 月第 一 版 开本：720×1000 1/16
2003 年 1 月第一次印刷 印张：55 3/4
印数：1－4000 字数：1096 000

定价：85.00 元（含光盘）

（如有印装质量问题，我社负责调换<环伟>）

前　言

1991 年 VB 1.0 问世，开始了 RAD（Rapid Application Development，快速应用程序开发）模式的新纪元。

在 VB 6.0 获得空前成功的背景下，VB 又以.NET 的全新姿态亮相。VB.NET 的推出在计算机界引起轩然大波，反对者与拥护者各执一词。对于新世纪之初软件界的这场大论战，历史将会做出最客观的结论。

坐等必然会失去良机。惟有静观其变，以一种平和的心态，回顾叱咤风云的 VB 6.0，展望如雷贯耳的 VB.NET，然后再决定下一步的行动，是比较明智的选择。

本书的宗旨

VB 6.0 是以 COM（Component Object Model，组件对象模型）为基础的，VB.NET 则完全建立在.NET 框架之上，核心技术完全不同。将 VB.NET 看成是 VB 6.0 的升级很是牵强，倒是将它作为一种全新的语言也许更为贴切。

现在的 VB.NET 还没有完全成熟，在某些功能方面赶不上 VB 6.0。

VB.NET 的开发人员不仅没有排斥 VB 6.0，而且表示要不断观察 VB 6.0 的技术领域并吸收素材充实 VB.NET 系统。

以上这些基本事实说明，VB 6.0 不会很快退出历史舞台，在相当长的一段时间内 VB.NET 要与 VB 6.0“和平共处，共同繁荣”。

在这段过渡时期内，已经开发的 VB 6.0 系统是否升级到 VB.NET，新的开发项目是使用 VB 6.0 还是 VB.NET，这是 VB 程序员和有关决策者必然关注的问题。本书正是针对从 VB 6.0 向 VB.NET 过渡的特殊性，在介绍 Visual Basic 高级编程技术的同时，探索从 VB 6.0 向 VB.NET 转化所可能遇到的一些问题，并提出一些建议。

本书的主要内容

本书一共分 14 章。

前 13 章包括面向对象编程、ActiveX 编程、OLE、连接 Windows API、多媒体编程、数据库编程、Visual SourceSafe、资源文件、注册表、拖放、串行通信、Internet 编程、应用程序发布等内容。以上各章在介绍编程方法的同时，着重讨论从 VB 6.0 升级到 VB.NET 所出现的一些问题。

第 14 章可以看作是前 13 章的总结。包括对 VB 6.0 和 VB.NET 进行全方位的

比较，以示例的形式着重对 VB.NET 如何实现继承性作了说明，并给出 VB.NET 调用 VB 6.0 组件的方法。最后分析在过渡时期可以采用的策略。

本书配有光盘，包括各个章节所提供示例的源程序。源程序分成两大部分：VB 6.0（置于 VB_6 文件夹）和 VB.NET（置于 VB_NET 文件夹）。

本书的特点

• 本书既不是泛泛介绍 VB 6.0 的书，也不是 VB.NET 的入门教材。像如何安装开发、如何进入开发环境、如何使用联机帮助等，诚然这些都很重要，但都不是本书讨论的对象。有关这类问题的书籍很多，读者可以从中取得帮助。

• 通俗易懂、循序渐进是本书的法宝之一，从最简单的内容开始，一步一个脚印，逐步地深入。

• 计算机编程有很强的实践性，因此示例贯穿本书的始终。读者在动脑的过程中，动手在计算机上试一试，会取得事半功倍的效果。

本书的读者对象

• 略有一些 VB 6.0 编程经验，而又想进一步提高的 VB 应用程序开发人员。

• 有一定的 VB 6.0 编程经验，想升级换代到 VB.NET 的应用程序开发人员。

• 已经开发了 VB 6.0 应用程序，在 VB.NET 面前还拿不定主意的应用程序开发人员和决策人员。

如何使用本书

• 对于 VB 编程经验不多的读者，应以示例为前导，从头到尾阅读本书。

• 对于 VB 编程经验比较多的读者，可以选择自己感兴趣的部分阅读。

• 想通过本书了解如何从 VB 6.0 过渡到 VB.NET 的读者，第 14 章将是重点。第 14 章也着重讲解了 VB.NET 热点项目的编程方法。

• 阅读文字和运行示例要穿插进行。示例是编程方法的具体化，通过运行示例更可以获得感性认识。

• 本书中提到的“详解”，是指 2001 年由科学出版社出版的笔者的另一本著作《Visual Basic 6.0 中文版编程方法详解》。这本书对 VB 编程的基础方法进行了详细介绍，可以作为缺少 VB 编程经验的读者的参考书。

• 笔者在编制 VB 6.0 的示例时操作系统是 Windows 98。在扩展到 VB.NET 时不得不改用 Windows 2000 和 Windows XP，原因是 Windows 98 不再支持 VB.NET。

如何使用光盘

本书所附光盘提供示例的源代码，在运行时需要注意：

• 属于 VB_6 文件夹的，需要在 VB 6.0 的开发环境下运行（.DLL、.EXE、控件等组件不能直接运行）。属于 VB_NET 文件夹的，需要在 VB.NET 的开发环境下运行。

• 如果读者的计算机有 E 盘，建议将光盘上的文件夹 VB_6 和 VB_NET 复制到 E 盘，以避免出现路径的麻烦。

• 如果运行时出现“不是有效路径”等类的错误，请读者按书中所介绍的步骤变换成读者的路径。对 VB 6.0 应用程序来说，这类问题要少一些，原因是本书大量使用相对路径（App.Path &“/文件名”），这一招对 VB.NET 则行不通。可以按示例所提供的步骤重新建立，在 VB 6.0 程序正常运行之后，再试着升级为 VB.NET。

• 对用到.DLL、.EXE、控件等组件的应用程序需要在开发环境中对组件重新编译，这个过程可以将组件在读者的系统注册表登记，只有登记后才能调用。

感谢

李玉书研究员和万永革博士参加部分工作。

如果没有全家人的热情鼓励和鼎立支持，本书是不可能完成的。

一年半以来，笔者不敢懈怠，全身心地投入到本书的编著中，力求完美。但由于水平有限，书中错漏在所难免，恳请专家和广大读者指正。

作　者

目　　录

第 1 章　面向对象编程

20 世纪 70 年代后曾盛行结构化的程序设计技术，其编程思想是将课题分解成一个一个的过程，程序由过程组合而成，程序的运行形成过程流。这是面向过程的编程技术。

面向对象编程 OOP（Object Oriented Programming）是 20 世纪 80 年代后出现的，现已成为软件开发的时尚。这种技术隐含了操作对象的大量细节，通过操作对象属性和调用对象方法建立应用程序，从而使程序员能够最大限度地重复使用现有代码，将精力集中于对象的使用方面，从而大大提高了软件开发效率。

从面向过程编程转向面向对象编程，是软件开发思想的一次巨大飞跃。利用 OOP 方法可以在较短的时间内，使用较短的代码设计出新颖、简洁、可以重复使用并便于维护的程序。

我们周围的一切事物，包括房子、汽车、山脉、河流等，都可以说成是对象。每个对象都有自己独特的性质，各个对象之间可以存在互相联系、互相依存的关系。这种“对象普遍性”的认识反映到程序设计中，就形成 OOP 的基本观点：窗体、控件、文件、数据库、打印机等一切都是对象，对象无所不在。对象是 OOP 认识论的核心。

OOP 涉及的另一个基本概念是“类”。VB 工具箱中的控件就是类。窗体中的控件则是对象。对象是由类“克隆”出来的，对象就是类的实例。类是建立可重复使用代码的基础。

OOP 有四大特征：抽象性、封装性、多态性、继承性。这四大特征是衡量某种语言是否是真正的面向对象语言的尺度。

VB 6.0 算不上是真正的 OOP 语言，原因是 VB 6.0 缺乏继承性。但这并不影响 VB 6.0 使用类将程序组织成一个真正的可重复使用的模块。而 VB.NET 是一个真正的面向对象的编程语言，抽象性、封装性、多态性、继承性一应俱全。

毋庸讳言，对习惯于传统编程方法的学者来说，面向对象编程是一块硬骨头。面向对象编程的难点在于必须建立全新的编程思路。

就全书而言，本章是比较难懂的一章，有些概念会一而再、再而三地提到，读者可能会感到有些凌乱。不要紧，只要钻进去，乱到一定程度，就会产生豁然开朗的感觉。

本章的主要内容是：

- 描述对象、类、对象变量、面向对象特征等基本概念。
- 介绍自定义类的创建方法，特别是一些实用技术。

本章一直在使用名字 clsEmployee 代表某一种类，但 clsEmployee 的内容一直在改变。最初的 clsEmployee 仅包括一个公有属性（1.3.6 节），以后改变为私有属性，然后用属性过程公布（1.3.7 节）。1.3.8～1.3.10 节又在 clsEmployee 类中增加了方法、事件和文档。1.4.4 节和 1.4.5 节在 clsEmployee 中又考虑对象集合，在窗体模块中引入集合运算。clsEmployee 的演变体现了由浅入深、由简单到复杂的历程。

- 集合类进入更深的层次，分析如何将集合的属性和方法封装从而构成类。
- 给出创建文件处理类的方法，可以将类中的数据做成文件保存起来。
- 提供外部事件的生成和响应方法。
- 对接口进行全面介绍，然后探讨如何通过接口技术实现多态性和继承性。

1.1 OOP 概述

对象与类是 OOP 中两个最重要的基本概念。

1.1.1 对象

对象是将数据和代码包装起来的实体，例如放在窗体上的文本框就是一个对象。

VB 对象具有属性、方法和事件。属性是描述对象的数据；方法告诉对象应该做的事情；事件是对象所引发的事情，事件发生时可以编写代码进行处理。

在应用程序的集成开发环境下，属性窗口可以显示对象所属的类和 Name 等属性。例如，在 VB 6.0 的窗体上安置一个文本框对象，此时可以在属性窗口显示该对象所属的类是 TextBox 和缺省的 Name 属性为 Text1，参见图 1-1（在 VB.NET 中对应的缺省名为 TextBox1）。

1.1.2 VB 6.0 的全局对象与 VB.NET 的等效项

1. VB 6.0 的全局对象

VB 6.0 的对象可以分成程序对象和系统对象。程序对象又有预定义和自定义之分。例如，在窗体上加入文本框，该文本框是预定义的程序对象，原因是系统对文本框的属性、方法和事件都预先做了规定。如果在窗体上加入 3 个文本框，尽管其属性值可能各不相同，但都是工具箱中同一个类——文本框类的实例，属性的个数和意义都是预定好的。

自定义程序对象的属性、方法和事件需要由程序员自行设置。

系统对象也是预定义的，是应用程序层之外的对象。前面提到的文本框对象是预定义的，但在将其纳入窗体之前该对象并不真正存在，预定义的程序对象是依赖于应用程序的，是属于应用程序内的。而系统对象则不同，是独立于应用程

序存在的，是先天的，在编程时可以直接引用而无须声明和实例化。系统对象又称为全局对象。VB 6.0 的全局对象一共有 6 个，表 1-1 提供了全局对象及其方法。

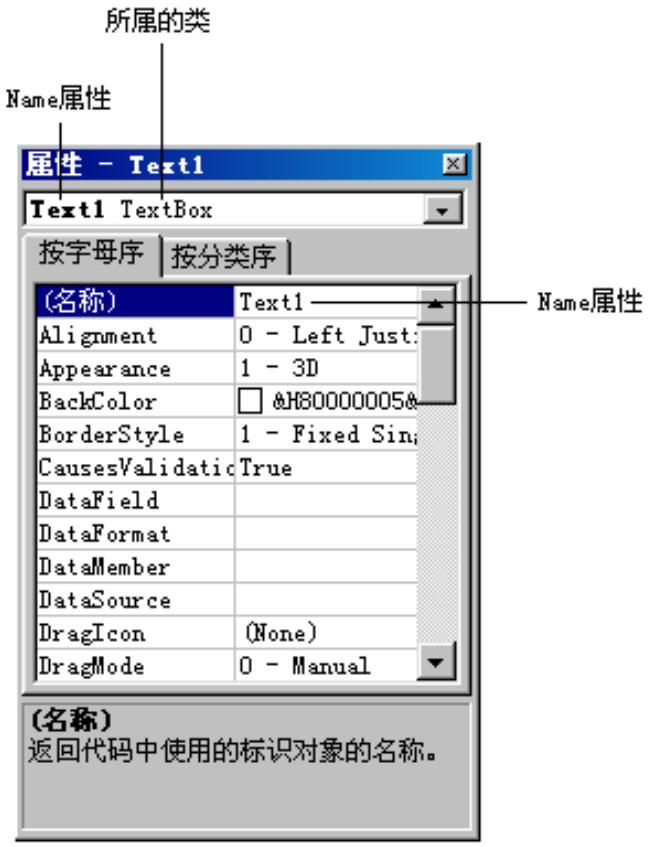

图 1-1　对象的类名和属性

表 1-1　VB 6.0 的全局对象及其方法

对　　象	描　　述	方　　法
App	当前应用程序	EXEName 返回应用程序文件名 Path 返回应用程序路径 Title 返回初始启动窗体标题 PrevInstance 返回应用程序的前一个实例是否在运行
ClipBoard	剪贴板	Clear 清除剪贴板 GetData 返回剪贴板内的图形 GetText 返回剪贴板内的文本 SetData 将图形放到剪贴板 SetText 将文本放到剪贴板 SelStart 返回或设置所选文本的第一个字符的位置 SelLength 返回或设置所选文本的长度 SelText 返回或设置所选文本
Debug	程序调试	在开发环境下运行非.EXE 程序时在立即窗口显示信息
Printer	打印机	提供将输出信息发送到打印机的打印功能
Screen	屏幕	FontCount 返回当前屏幕支持的字体数目 Fonts 提供所有可能的屏幕字体名称列表 Height 返回屏幕的高度 Width 返回屏幕的宽度

续表

对　象	描　述	方　　法
		TwipsPerPixelX 返回水平度量的对象的每像素的 Twip 数 TwipsPerPixelY 返回垂直度量的对象的每像素的 Twip 数 MousePointer 指定鼠标形状
Err	保存出错信息	Raise 生成运行时错误 Clear 清除 Err 对象的所有属性设置

全局对象 App 可以取得可执行文件的路径。例如，一个.WAV 文件与一个可执行应用程序在同一个文件夹，这时可以有：

```
MMControl1.FileName = App.Path & "\WAV300.WAV"
```

用这种方法加载文件比使用绝对路径要优越的多，可以避免因更换计算机而造成的找不到文件的弊端。

又如，判断是否有前一个应用程序的实例运行，可以使用

```
If App.PrevInstance Then
MsgBox "有以前的程序实例运行！"
Else
MsgBox "没有以前的程序实例运行！"
End If
```

如果想使窗体 Form1 位于屏幕的中心，可以使用屏幕对象 Screen，在 Form_Load 事件中加入

```
Form1.Left = (Screen.Width - Form1.Width) / 2
Form1.Top = (Screen.Height - Form1.Height) / 2
```

利用 Screen 对象还可以设定鼠标指针形状，例如：

```
Screen.MousePointer = vbHourglass
```

将鼠标指针设定为沙漏状，通常表示等待。而

```
Screen.MousePointer = vbDefault
```

将鼠标指针设定为与对象有关的缺省状态。

表 1-2 给出 VB 6.0 的 MousePointer 的设置值。

表 1-2　VB 6.0 的 MousePointer 的设置值

常　　数	值	说　　明
vbDefault	0	由对象决定的缺省设置
vbArrow	1	箭头
vbCrosshair	2	十字线
vbIbeam	3	I 型
vbIconPointer	4	矩形内的小矩形
vbSizePointer	5	东、南、西、北四方向箭头
vbSizeNESW	6	东北和西南方向的箭头

续表

常　　数	值	说　　明
vbSizeNS	7	向南和向北方向的箭头
vbSizeNWSE	8	向东南和西北方向的箭头
vbSizeWE	9	向东和向西方向的箭头
vbUpArrow	10	向上的箭头
vbHourglass	11	表示等待的沙漏状
vbNoDrop	12	不允许放下
vbArrowHourglass	13	箭头和沙漏
vbArrowQuestion	14	箭头和问号
vbSizeAll	15	四向尺寸线
vbCustom	99	自定义图标

2. 全局对象在 VB.NET 中的等效项

在 VB 6.0 应用程序中大量使用全局对象，例如 App、Screen、Clipboard 等，很多程序员都用得滚瓜烂熟。可惜 VB.NET 都不再继续支持。在从 VB 6.0 向 VB.NET 升级的过程中这些全局对象可能出现如下三种情况。

（1）有对应的等效项，运行结果也正确

例如：

```
Form1.Left = (Screen.Width - Form1.Width) / 2
Form1.Top = (Screen.Height - Form1.Height) / 2
```

在 VB.NET 中的等效项为

```
Form1.DefInstance.Left = VB6.TwipsToPixelsX((VB6.PixelsToTwipsX _
    (System.Windows.Forms.Screen.PrimaryScreen.Bounds.Width) - _
    VB6.PixelsToTwipsX(Form1.DefInstance.Width)) / 2)
Form1.DefInstance.Top = VB6.TwipsToPixelsY((VB6.PixelsToTwipsY _
   (System.Windows.Forms.Screen.PrimaryScreen.Bounds.Height) - _
   VB6.PixelsToTwipsY(Form1.DefInstance.Height)) / 2)
```

又如：

```
If App.PrevInstance Then
    MsgBox "有以前的程序实例运行！"
Else
    MsgBox "没有以前的程序实例运行！"
End If
```

在 VB.NET 中的等效项为

```
If (UBound(Diagnostics.Process.GetProcesses _
    ByName(Diagnostics.Process.GetCurrentProcess.ProcessName))>0) _
    Then
```

```
        MsgBox("有以前的程序实例运行！")
    Else
        MsgBox("没有以前的程序实例运行！")
End If
```

（2）有对应的等效项，但执行时没有反应

例如：

· Screen.MousePointer = vbHourglass

升级向导变换为

```
System.Windows.Forms.Cursor.Current = _
    System.Windows.Forms.Cursors.WaitCursor
```

· Screen.MousePointer = vbDefault

则变换为

```
System.Windows.Forms.Cursor.Current = _
    System.Windows.Forms.Cursors.Default
```

可惜执行后都没有反应。

又如，在 VB 6.0 窗体代码中包括的 APP.Path：

· strINIFileName = APP.Path & "\GPS 数据.ini"

升级向导将其升级为

```
strINIFileName = VB6.GetPath & "\GPS 数据.ini"
```

但在运行时却不能获取当前应用程序的路径，不得已用手工改成绝对路径：

```
strINIFileName = "C:\LHJ\VB_6\B\ GPS 数据.ini"
```

如此一来，由 App.Path 提供路径所具有的优越性丧失殆尽。

（3）没有对应的等效项

这时会出现语法错误，升级将失败，这类情况比较多见。对付的办法是按 VB.NET 的要求耐心修改程序。例如，在 VB 6.0 中常常用 Clear 方法清除剪贴板，即

```
Clipboard.Clear
```

升级到 VB.NET 则会出现如下的语法错误：

```
"Clear"不是 "System.Windows.Forms.Clipboard" 的成员
```

1.1.3 类

俗话说，物以类聚，人以群分。根据事物的相似性将事物归类，例如，张志和刘华都是某公司的雇员，可以归并为“雇员类”，但张志是公司的司机，又可以归并为“司机类”，刘华的特长是编软件，又可以归并为“程序员类”。将相似的事物聚在一起，这是类的第一层意思，即从类的特征来描述类。

类还有第二层意思，即类是“制造对象者”，打个比方，类就像是能繁殖工

蜂的蜂王，可以“克隆”对象。这是从功能上来认识类。

工具箱中的每个控件就是一个类，是由 VB 系统所规定好的预定义类。

类本身不具备属性值，也没有行为可以执行。但类可以定义属性并包括行为的说明。真正具备属性值且执行行为的是由类所创建的对象，就像专门干活的是工蜂一样。我们也可设想类为制造汽车的工厂，所制造的汽车是对象。我们无法也没有必要规定汽车工厂的外形具有蓝色、红色、黑色、白色等属性值，但由汽车工厂所生产的汽车则具备车体颜色的属性值。同样，我们不能把汽车制造厂开来开去，但汽车则可以，原因是汽车是汽车工厂制造出来的“对象”，可以执行由工厂“类”所规定的行为。

简言之，对象是由类所创建的，对象是类的实例。例如，标准工具箱中的命令按钮是“命令按钮类”，在把命令按钮加载到窗体之后，便创建了命令按钮类中的一个实例——“命令按钮对象”。加载窗体上的命令按钮、文本框、标签等这一类对象属于预定义对象，VB 已经完成对其封装的任务。

程序员可以根据编程的需要建立自定义的类。自定义的类大大增强了程序员的编程能力，具体来说自定义类在编程时可以起如下作用：

（1）可以利用类封装程序段，这可以提高程序段的可读性并易于重复使用。

（2）建立 ActiveX DLL 和 ActiveX EXE 文件。例如，可以将环境保护条例放在一个对象中，并编译为 ActiveX DLL 文件。通过引用这个 DLL 文件，其他程序员可以很方便地将这些条例纳入自己的程序中。

（3）用类建立插入功能，可以增强 VB 的开发环境。插入功能可以用来建立程序向导、提供辅助程序、制作使程序设计简单的工具。

1.1.4 类的特征

【VB 6.0】

（1）一个类对应一个.CLS 文件，或者说一个.CLS 文件只包括一个类。

（2）用扩展名.CLS 来识别类。

（3）类与类之间在创建时没有明确的关系。

（4）类的限定符有：

· Private：类只能由创建类的工程使用。

· PublicNotCreatable：类可以由工程之外的代码使用，但必须由工程内代码创建。

· SingleUse：类可以由工程之外的代码使用和创建。

· GlobalSingleUse：类可以由工程之外的代码使用和创建。

· MultiUse：类可以由工程之外的代码使用和创建。

· GlobalMultiUse：类可以由工程之外的代码使用和创建。

【VB.NET】

（1）不再使用.CLS 作为类模块的扩展名，不管是类、窗体，还是模块其扩展名都是.VB。

（2）在一个文件中可以包括多个类。

（3）在.VB 文件中用关键字来识别类，例如：

```
Public Class XClass
...
...
End Class
```

（4）在类中可以嵌套类，例如：

```
Public Class XClass
...
    Public Class YClass
    ...
    ...
    End Class
...
End Class
```

类 YClass 是类 XClass 的子类。

（5）类的限定符有：

· Public：没有访问限制。

· Private：只能由当前命名空间的代码访问，仅适用于类中的代码。

· Protected：可由派生类访问，适用于类的继承类。

· Friend：可以由相同命名空间的不同类型的类访问，对工程或组件中的代码有效。

· Protected Friend：Protected 加 Friend。

1.1.5 结构

【VB 6.0】

在 VB 中有一种组合的数据类型称为“结构”，例如：

```
Public Type 天气信息
    气温 As Single
    风向 As String
    风力 As Single
End Type
```

就是 VB 6.0 的结构。

利用“Dim udtA As 天气信息”可以声明一个“天气信息”型的结构数据。以后我们就可以这样为其赋值：

```
udtA.气温 = 10.5 : udtA.风向 = "南转北" : udtA.风力 = 3
```

也可以按如下方式在立即窗口显示结构数据：

```
Debug.Print udtA.气温, udtA.风向, udtA.风力
```

VB 6.0 的“结构”确实是一种方便、灵活、功能强大的数据类型。但 VB 6.0 的“结构”与 VB 6.0 的“类”相比，则是“小巫见大巫”了。原因是“类”不仅以属性的形式封装了数据项，还以过程的形式封装了方法、事件等，可以说“类”是将“结构”和“过程”组织在一起的整体。

【VB.NET】

与 VB 6.0 的结构相比，VB.NET 的结构有两点重要变化：一是不再使用 Type 而使用 Structure；二是既包括数据项，也包括属性过程和方法，在功能上向类靠拢。

例如，VB.NET 的“天气信息”结构是：

```
Public Structrue 天气信息
    气温 As Single
    风向 As String
    风力 As Single
    降雨强度 As Single

    ReadOnly Property FullCase
        Get
            Return 气温 & " " & 风向 & " " & 风力
        End Get
    End Property

    Public Function 降雨量(ByVal 面积 As Decimal) As Decimal
        Return 面积 * 降雨强度
    End Function
End Structure
```

后面的函数“降雨量”就是方法。

1.1.6 对象和类

对象是运行时的实体，需要占用内存资源。当对象卸载或应用程序结束，对象就会消失。

类是设计时的实体，是应用程序设计人员的工具。应用程序的用户看不到类，只能看到由类所创建的对象。

将某个类赋予对象变量时，类的一个实例——对象就被创建并被初始化。

1.1.7 类的实例化

【VB 6.0】

在 VB 6.0 中有两种方法可以将类赋给对象。

（1）早期绑定

将对象变量声明为某特定类型的变量后，然后使其实例化：

```
Private objA As XClass
Set objA = New XClass
```

这种方式分两步走。第一行代码将对象变量 objA 声明为特定的类 XClass，实际上是存储 XClass 类的一个对象的引用。第二行代码将类 XClass 的一个实例赋给对象变量 objA。

早期绑定的基本点是使用特定的类名，或者说使用显式类名。

对早期绑定，VB 6.0 在编译时就能够把类和对象绑定在一起，就能够找到对象的属性、方法并存储对该对象的一个引用。如果存在不支持的属性或方法，编译器就会立即捕获到错误。运行时，能够使用该引用立即调用对象的属性、方法，而不用再去查询该对象的方法或方法的参数。

在声明对象变量时，允许使用 New 关键字，这时是一步走。例如：

```
Private objA As New Xclass
```

由这种方式所声明的对象变量立即可以使用，被称为“自动实例对象变量”。

对 VB 6.0 而言，“自动实例对象变量”有两个问题。其一，在每次运行到使用该类型变量的语句时，VB 6.0 就必须检查该对象是否已经创建，影响运行效率。其二，如果不小心终止了该对象，之后又再次引用，这时将创建类的一个新实例，如果再从终止的变量中提取某种属性的话，将发生意想不到的结果。鉴于上述弊端，在 VB 6.0 中使用“自动实例对象变量”时应该谨慎。

（2）晚期绑定

不是将对象变量声明为某特定类型，而是声明为一般类型，例如：

```
Private objA As Object
Set objA = New XClass
```

这种方式也是分两步走。第一行代码将对象变量 objA 声明为一般的类 Object。第二行代码建立了对 XClass 类的新对象的引用和实例化。

也可以使用 CreateObject 函数，例如：

```
Dim ObjA As Object
Set objA = CreateObject("Scripting.FileSystemObject")
```

对于将对象变量声明为 Object 或 Variant 等来代替类名的晚期绑定，编译时不进行错误检查，编译器不能推断出这个变量将包括哪种类型的对象，只能存储有关的属性或方法的名字和参数的信息，直到运行时对象变量和对象才能绑定。绑定之后，VB 6.0 就有了对该对象的一个引用，该引用存储在一个对象变量中。然后就可以用对象变量来引用对象、设置对象的属性值或者执行对象的方法。这种晚期绑定在运行时必须先找到对象的属性、方法，然后才能调用。以动态的方式处理对象，提供了极大的灵活性。

绑定的类型与访问对象的速度有关，访问早期绑定的变量比晚期绑定的速度快。较为粗略的估计，VB 6.0 利用晚期绑定访问一个简单的属性有可能比利用有效的早期绑定要慢上大约 200 倍，这确实是一个不容忽视的数字。在编制 VB 6.0 应用程序时建议尽量使用早期绑定。

表 1-3 列出 VB 6.0 晚期绑定所使用的一般的对象类型。

表 1-3　VB 6.0 中一般的对象类型

对象类型	可引用的对象
Form	应用程序中的任何窗体
Control	应用程序中的任何控件
MDIForm	应用程序中的 MDI 窗体
Object	应用程序中的任何对象
Variant	应用程序中的任何变量

【VB.NET】

在 VB.NET 中也可以先声明一个对象变量，然后再创建实例：

```
Dim objA As XClass
objA = New XClass()
```

也可以一步走，例如：

```
Dim objA As New XClass()
```

可以使用 VB 6.0 所没有的方式，例如：

```
Dim objA As XClass = New XCLass()
objA = Assembly.CreateInstance("XClass")
objA = System.Activator.CreateInstance("XClass","URIToAssembly")
```

VB.NET 不再使用 VB 6.0 中的 Set 语句，原因是在 VB.NET 中对象的处理与其他类型的数据是一样的。

VB.NET 不再使用 VB 6.0 中的 CreateObject 语句，原因是 CreateObject 语句与 COM（组件对象模型，Component Object Model）连带有关，VB.NET 不再使用 COM。

在 VB.NET 中创建类实例也存在早期绑定和晚期绑定。只要 Option Strict 设置为 On（缺省设置），所有对象都是早期绑定。如果 Option Strict 被设置成 Off，在整个代码中都使用晚期绑定。

“自动实例对象变量”在 VB 6.0 中的负面作用，在 VB.NET 中不再存在，可以大胆使用。

1.1.8 OOP

OOP 的基本思想是，使用可重复使用的程序段来编程。通过程序段的重复使用，可以减少很多不必要的重复劳动，降低出错机会，从而大大提高应用程序的开发效率。作法是提取各可视化对象、数据对象及数学对象中的共性归纳成类，由类再“克隆”出对象，实现程序段的重复使用。

OOP 的四大特征：抽象性、封装性、多态性、继承性。

1. 抽象性

抽象是对对象进行归纳的能力。面向对象语言借助类来进行抽象。类定义对象的属性和方法。在抽象过程中先集中考虑属性和方法，而不管具有这些属性和方法的具体对象。对事物进行抽象是在事物的个性中寻求共性，实质上是集中事物要点的过程。也就是说先超脱一些，眼光放得广一些、远一些，而不是只盯住具体的事物不放。

在不同的条件下会有不同的抽象，例如在发工资的时候，张志和刘华都应该抽象为“雇员类”，因为这时共同的注重点是雇员的薪水。而在技术考核时，不能去考核司机张志编程序的能力，他应该抽象为“司机类”；也不能去考核程序员刘华开车的能力，她应该抽象为“程序员类”。

使用抽象可以首先将精力集中在应用程序的功能上，考虑需要做些什么，而先不必探讨怎样由计算机来实现，由此可以先从繁琐的计算机处理技术中解脱出来。这样做的重要性之一是便于最终用户的参与，最终用户可以不理会编程的具体技术而对系统功能横挑鼻子竖挑眼。

2. 封装性

确保对象的有关信息和信息处理方式都存储在对象的定义内，对象的细节在外面是看不见的，也称为信息隐藏。封装好的对象的所有数据都保存在一个内存区域，不能被别的应用程序直接访问，所有的赋值和读取都通过对象本身所具有的属性、方法和事件进行。封装性将对象的接口（属性、方法、事件）与对象的实现完全分开。

可以用“黑箱”来加深对封装性的理解。当代的程序员用不着琢磨正弦函数是如何计算出来的，只要将自变量送入正弦函数的“黑箱”，出来的就是正弦函数

值。我们说正弦函数的计算细节被封装到“黑箱”中。一个封装好的对象也等同于一个“黑箱”。当然，正弦函数的“黑箱”与对象的“黑箱”是有区别的，前者只封装算法，而后者则包括算法和数据。

封装性是一个颇受程序员欣赏的特性，至少有以下两个重要作用：

· 可以对所有赋给对象属性的值进行检查，并立即剔除无效的值。

· 可以自由地修改对象内的数据，而无须改变程序的其他部分。

3. 多态性

指不同类的对象对外界显现相同或相似的特征。窗体和控件都具有多态性。例如，窗体上的文本框和图片框是不同类的对象，但它们具有相同的属性和方法，包括 Name、Left、BackColor 属性和 Move 方法等等。多态性处理的是具有同样接口的不同的类。

编程时利用多态性的好处是：

· 可以少记很多参数和语法格式。

· 可以利用单独的变量管理一组控件，可以创建窗体上管理所有控件的一般过程。

总的来说多态性可以大大减少编程时使用类的代码量。

4. 继承性

前面反复提到，由类可以实现程序段的重复使用。继承性也涉及到重复使用，但继承性说的是由类到类的重复使用，即从基类派生出继承类，或者说从父类生成子类。继承类（子类）具有基类（父类）的属性和方法。由继承性可以大大减少类创建者的工作量。

VB 6.0 不支持继承性，这就意味着其他程序员不能直接在已经建立的类的基础上创建子类，但仍然可以使用一些技术模拟继承性。VB.NET 则实现了完全的继承，成为真正面向对象的编程语言。

1.2 对象变量

变量可以有整型、单精度、货币型、字符型等多种类型。同样，对象也可以成为变量。从广泛的意义上说，放置在窗体上的文本框、标签、命令按钮等都是对象变量的具体值。

如果一个程序只是用常量参与运算，常量值的每次变化都要修改程序，这种程序是很糟糕的。编程中使用变量的必要性在于提高程序的灵活性、通用性和可读性，使用对象变量的必要性也如此。此外，将对象赋给对象变量还有如下优点：