Windows API 函数在 Visual Basic 中的应用实例

姚巍　编著

人 民 邮 电 出 版 社

图书在版编目（CIP）数据

Windows API 函数在 Visual Basic 中的应用实例 / 姚巍编著.
—北京：人民邮电出版社，2003.12
ISBN 7-115-11736-5

Ⅰ. W… Ⅱ. 姚… Ⅲ. BASIC 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2003）第 100994 号

内容提要

本书在众多的 API 函数中选取了经常使用的 109 个用于完成不同操作任务的 API 函数，介绍了在 Visual Basic 编程环境中，使用这些 API 函数进行程序设计的方法。书中以编程实例的方式，详细说明了这些 API 函数在程序中的使用。全书共提供了 89 个完整的程序。

本书除重点介绍使用 API 函数进行程序设计外，作为比较还选择了少量与某个 API 函数功能相同或相近的控件进行了介绍。除 API 函数的基础知识外，书中介绍的内容按工作对象的不同分为 Windows 系统操作、工作界面操作、文件磁盘操作、多媒体操作、注册表操作、网络操作和非 Windows 动态连接库的使用。

本书内容丰富、思路清晰、实用性强，适合有一定 Visual Basic 编程经验，且有进一步提高编程水平要求的程序设计人员阅读，也可以供大专院校相关专业的师生参考。

Windows API 函数在 Visual Basic 中的应用实例

◆ 编　　著　姚　巍
责任编辑　马　嘉

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
读者热线　010-671940942
北京汉魂图文设计有限公司制作
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：19
字数：459 千字　　2003 年 12 月第 1 版
印数：1 –　000 册　　2003 年 12 月北京第 1 次印刷

ISBN 7-115-11736-5/TP・3657

定价：35.00 元

本书如有印装质量问题，请与本社联系　电话：（010）67129223

前　言

使用 Visual Basic 编写程序的读者应该都能体会到这种语言简单、方便、快捷的优点。程序设计人员经常需要使用 Visual Basic 提供的、能实现不同操作功能的、被称为方法的语句、控件和函数。使用这些方法可以实现通常所需的操作，但是 Visual Basic 提供的方法也有局限性，例如使用 Visual Basic 提供的对注册表进行操作的函数只能对注册表的指定位置进行操作，而无法对注册表的任意键进行操作。对于此类操作，就有必要使用 API 函数来实现。

API 是“应用程序编程接口”的简称，这是一系列很复杂的函数。编程人员可以用不同的编程语言调用 API 函数，编写出在 Windows 操作系统中运行的应用程序。API 函数有成百上千个，这些函数有的可以单独在程序中使用，有的需要几个 API 函数配合使用。在数量众多的 API 函数中，只要熟练地掌握和运用其中的百十个函数，即可以较大程度地提高程序设计的水平，并使应用程序在功能上更为完善。

一、本书内容

Visual Basic 本身提供了许多与 API 函数功能相同或相近的操作，如 CopyFile，对于这类函数本书不作介绍。书中选择的 API 函数是在编程实践中经常使用的，能完成 Visual Basic 本身提供的方法所不能或不便完成的操作。将这些 API 函数的具体使用方法以具体实例的方式介绍给读者，希望能对读者有所帮助。

本书介绍的内容涉及到以下几个方面：API 函数的基础知识、Windows 系统操作、工作界面操作、文件及磁盘操作、多媒体操作、注册表操作、网络操作和对非 Windows 动态链接库的使用。本书共分为 8 章，各章的主要内容如下：

第 1 章　API 函数的基础知识

介绍了有关 API 函数的基础知识，给出了 4 个使用 API 函数进行简单操作的实例，最后介绍了查看 API 函数的简单方法。设置这一章的目的是使读者能对 API 函数的构成及使用有个初步的了解。

第 2 章　Windows 系统操作

Windows 的系统操作涉及的问题很多，本章选取了一些与系统操作有关的内容介绍给读者。如获得操作系统版本、得到系统目录路径和计算机名、CPU 和内存测试、程序运行、程序快捷方式的建立等。

第 3 章　工作界面操作

程序的工作界面是为用户提供的操作平台，Visual Basic 解决了程序工作界面设计的许多问题，但是一些比较特殊的操作，如隐藏鼠标光标等还需要使用 API 函数实现。在这一章介绍了使用 API 函数进行有关鼠标操作、窗体显示方式及窗体颜色设置、特殊的文字显示效果、多种形式的图片显示效果及屏幕分辨率调整、屏幕的亮度控制等程序的设计。

第 4 章　文件及磁盘操作

本章介绍与文件操作有关的内容，如对 INI 文件的操作、文件的复制和移动及在文件复制移动时的动画显示，对.hlp 和.chm 格式帮助文件的调用，并简单介绍了.chm 格式的帮助

文件的制作。在磁盘操作中介绍了诸如获得磁盘类型、磁盘格式化、磁盘容量的测试和磁盘序列号及磁盘文件类型等操作。

第 5 章 多媒体操作

对多媒体文件，可以直接使用 Visual Basic 提供的多媒体控件进行操作。如果应用程序仅需要对多媒体文件进行简单的播放操作，还是使用 API 函数较好。在这一章中介绍了使用 API 函数播放 WAV 文件、MIDI 文件、AVI 文件和 CD 的方法及怎样进行音量控制等问题。

第 6 章 注册表操作

注册表是计算机软、硬件信息的集合。Visual Basic 用于操作注册表的函数及操作注册表的功能有限，如需要操作注册表的任意键及数据项必须使用 API 函数。在这一章比较详细地介绍了注册表的作用及组成，常用的操作注册表的 API 函数，并给出了有关对程序运行方式的操作、数据备份、密码读写、显示设置和窗体颜色配置及 ODBC 配置等有关注册表操作的实例。

第 7 章 网络操作

计算机网络能否正常工作，涉及的问题不少，如 IP 地址、网卡和网络连接等。本章介绍了有关网卡物理地址、IP 地址、网络驱动器的映射、网络链接测试、Internet 的连接等有关网络操作的问题。

第 8 章 对非 Windows 动态链接库的使用

除 Windows 所提供的动态链接库外,有时还需要使用其他的动态链接库来实现某种操作。本章通过高精度定时器、操作 I/O 接口和读取条形码，介绍了如何在程序中使用其他的动态链接库。

二、本书的特点

本书不仅介绍了使用一个或几个 API 函数进行的单项操作，如获取本机操作系统版本、CPU 使用情况的测试等，还介绍了如何使用相同的 API 函数进行多项操作，如使用 BitBlt 函数实现不同形式的图片显示效果，使用 mciSendString 函数播放不同形式的多媒体文件等。

作为比较，书中选择了一些与某个 API 函数功能相同或相近的控件进行了介绍，如 AVI 文件的播放。除重点介绍使用 Windows API 函数进行程序设计外，书中还介绍了少量非 Windows 动态链接库提供的 API 函数的使用方法。

在本书的配套光盘中，对书中介绍的所有操作实例都给出了完整的程序，供读者参考。

三、本书的读者对象

在程序中使用 API 函数，对程序设计人员的编程水平要求较高。本书适合有一定编程经验，且有进一步提高编程水平要求的程序设计人员阅读，也可供大专院校相关专业的师生参考。

由于作者水平有限，错误或疏漏之处在所难免，欢迎广大读者批评指正。

作者邮箱：yaowei@up369.com

作者

目　录

第 1 章　API 函数的基础知识 · · · · · 1
1.1　API 函数的基本概念 · · · · · 1
1.1.1　什么是 API 函数 · · · · · 1
1.1.2　API 函数的组成 · · · · · 1
1.2　API 函数的使用方法 · · · · · 2
1.2.1　API 函数的声明 · · · · · 2
1.2.2　API 函数的变量 · · · · · 4
1.2.3　句柄 · · · · · 5
1.2.4　在 Visual Basic 中使用 API 函数的两个实例 · · · · · 7
1.3　API 函数的查看 · · · · · 10
1.4　本章使用的 API 函数列表 · · · · · 14
第 2 章　Windows 系统操作 · · · · · 15
2.1　系统信息 · · · · · 15
2.1.1　获得操作系统版本 · · · · · 15
2.1.2　系统目录路径 · · · · · 18
2.1.3　获得计算机名和用户名 · · · · · 19
2.1.4　设置计算机名 · · · · · 20
2.1.5　CPU 使用情况和内存容量测试 · · · · · 21
2.2　关闭 Ctrl+Alt+Del 组合功能 · · · · · 36
2.3　延时操作 · · · · · 38
2.4　得到系统当前运行的程序名 · · · · · 40
2.5　停止指定程序运行 · · · · · 42
2.6　退出系统 · · · · · 45
2.6.1　Windows 98 系统的关机操作 · · · · · 45
2.6.2　Windows 2000 系统的关机操作 · · · · · 46
2.7　建立快捷方式 · · · · · 53
2.8　使 PC 喇叭播放音乐 · · · · · 54
2.9　本章使用的 API 函数列表 · · · · · 57
第 3 章　工作界面操作 · · · · · 58
3.1　鼠标操作 · · · · · 58
3.1.1　限制鼠标移动 · · · · · 58
3.1.2　捕捉鼠标光标坐标 · · · · · 62
3.1.3　显示和隐藏鼠标光标 · · · · · 65

3.2 显示和隐藏任务栏 · · · · · 67
3.3 保持在前面的窗体 · · · · · 69
3.4 将程序图标加入到 Windows 系统托盘 · · · · · 71
3.5 椭圆窗体 · · · · · 75
3.6 使关闭按钮失效 · · · · · 77
3.7 自动关闭消息框 · · · · · 79
3.8 使窗体标题栏闪烁 · · · · · 81
3.9 改变窗体颜色 · · · · · 84
3.10 实现 ABOUT 窗体 · · · · · 86
3.11 空心文字 · · · · · 89
3.12 多种多样的图片显示效果 · · · · · 92
3.12.1 BitBlt 函数 · · · · · 92
3.12.2 图片的简单复制 · · · · · 93
3.12.3 从上到下效果 · · · · · 95
3.12.4 中间开幕效果 · · · · · 98
3.12.5 百叶窗效果 · · · · · 99
3.12.6 马赛克效果 · · · · · 101
3.12.7 上下对接效果 · · · · · 108
3.12.8 图像显示过程的程序模拟 · · · · · 110
3.13 屏幕分辨率的动态调整 · · · · · 114
3.14 改变屏幕亮度 · · · · · 118
3.15 本章使用的 API 函数列表 · · · · · 125
第 4 章 文件及磁盘操作 · · · · · 127
4.1 文件操作 · · · · · 127
4.1.1 INI 文件操作 · · · · · 127
4.1.2 文件操作及工作进度显示 · · · · · 131
4.1.3 清空回收站 · · · · · 135
4.1.4 帮助文件调用 · · · · · 141
4.2 磁盘操作 · · · · · 150
4.2.1 获得磁盘类型 · · · · · 150
4.2.2 磁盘格式化 · · · · · 151
4.2.3 获得指定磁盘的容量 · · · · · 153
4.2.4 获得磁盘序列号和文件系统类型 · · · · · 158
4.3 本章使用的 API 函数列表 · · · · · 161
第 5 章 多媒体操作 · · · · · 162
5.1 WAV 播放 · · · · · 162
5.2 MIDI 播放 · · · · · 164
5.3 音量控制 · · · · · 166
5.4 CD 播放 · · · · · 171

5.4.1 光驱仓门的弹出与关闭 · · · · · 174
5.4.2 CD 唱片总播放时间的测定 · · · · · 174
5.4.3 各曲目播放时间的测定 · · · · · 175
5.4.4 乐曲的播放、暂停和停止 · · · · · 177
5.4.5 曲目跳转 · · · · · 178
5.4.6 显示当前曲目的播放时间 · · · · · 179
5.4.7 工作界面设计 · · · · · 184
5.5 AVI 播放 · · · · · 184
5.5.1 使用 API 函数播放 AVI 文件 · · · · · 185
5.5.2 使用控件播放 AVI 文件 · · · · · 186
5.6 本章使用的 API 函数列表 · · · · · 187
第 6 章 注册表操作 · · · · · 188
6.1 注册表的作用及组织结构 · · · · · 188
6.1.1 注册表的作用 · · · · · 188
6.1.2 注册表的结构 · · · · · 189
6.1.3 注册表中各主键的作用 · · · · · 190
6.2 注册表操作函数概述 · · · · · 192
6.3 主键及项类型的对应常数 · · · · · 193
6.3.1 在对注册表操作时常数的作用 · · · · · 193
6.3.2 主键常数 · · · · · 193
6.3.3 数据类型常数 · · · · · 194
6.4 对程序运行的操作 · · · · · 194
6.4.1 程序的开机运行 · · · · · 194
6.4.2 防止同一程序的重复运行 · · · · · 199
6.5 数据的定期备份 · · · · · 207
6.6 使用注册表进行密码操作 · · · · · 211
6.6.1 密码输入的一般操作 · · · · · 211
6.6.2 在注册表中保存密码和读出密码时使用的 API 函数 · · · · · 212
6.6.3 在注册表中读取密码的程序设计 · · · · · 212
6.6.4 修改注册表中的密码 · · · · · 215
6.7 对显示选项和窗体颜色的操作 · · · · · 219
6.7.1 窗体颜色的改变和恢复 · · · · · 220
6.7.2 控制改变墙纸和显示选项 · · · · · 225
6.8 在注册表中直接配置客户端 ODBC · · · · · 229
6.8.1 配置 ODBC 的一般操作过程 · · · · · 230
6.8.2 ODBC 配置完成后注册表中的对应数据 · · · · · 233
6.8.3 用于在注册表配置 ODBC 的 API 函数 · · · · · 234
6.8.4 直接配置 ODBC 的程序 · · · · · 235
6.8.5 手动配置 ODBC 的程序 · · · · · 241

6.9 本章使用的 API 函数列表 · · · · · 245
第 7 章 网络操作 · · · · · 246
7.1 获得网卡物理地址 · · · · · 246
7.2 读取和设置 TCP/IP 协议属性 · · · · · 250
7.2.1 读取 IP 地址 · · · · · 252
7.2.2 设置 IP 地址 · · · · · 253
7.3 在程序中映射和断开网络驱动器 · · · · · 256
7.3.1 调用“映射网络驱动器”对话框 · · · · · 256
7.3.2 自动连接和断开网络驱动器 · · · · · 258
7.4 网络连接测试 · · · · · 260
7.5 Internet 的连接与断开连接 · · · · · 266
7.5.1 拨打默认连接 · · · · · 266
7.5.2 连接状态测试 · · · · · 268
7.5.3 连接到指定网站 · · · · · 269
7.5.4 断开拨号连接 · · · · · 270
7.6 本章使用的 API 函数列表 · · · · · 272
第 8 章 对非 Windows 动态链接库的使用 · · · · · 273
8.1 高精度定时器 · · · · · 273
8.2 操作 I/O 接口 · · · · · 274
8.3 读取条形码 · · · · · 275
8.4 本章使用的 API 函数列表 · · · · · 277
附录 1 各章使用的 API 函数总表 · · · · · 278
附录 2 本书使用的 API 函数索引 · · · · · 283
附录 3 各章附盘程序 · · · · · 287
附录 4 键值和 ASCII 码 · · · · · 290

第 1 章　API 函数的基础知识

这一章主要介绍与 API 函数有关的一些基础知识。为使读者初步了解 API 函数的使用，本章安排了 4 个用 API 函数进行简单操作的小实例。最后介绍了使用 Visual Basic 为用户提供的 API 阅读器，查看 API 函数的简单方法。

1.1　API 函数的基本概念

1.1.1　什么是 API 函数

API 的全称是“Application Programming Interface”，可以称为“应用程序编程接口”。API 是一些用 C 和 C++编写的，由操作系统或其他程序使用的函数。程序设计人员可以在不同的程序设计语言中使用 API 函数，编写运行于 Windows 操作系统的应用程序。

使用 API 函数可以在很大程度上扩展 Visual Basic 的功能。如果遇到使用 Visual Basic 本身提供的语句、函数和控件不便甚至无法实现的操作时，可以考虑使用 API 函数。API 函数在使用上很方便，在程序中只要事先对 API 进行了声明，就可以同使用 Visual Basic 自身的函数一样，在程序中使用这些函数。

API 函数涉及面广，数量众多。程序设计人员如果能掌握比较常用的百十个 API 函数，即可以解决很多问题并可以使程序的功能更加完善。

1.1.2　API 函数的组成

API 函数由许多能完成不同操作的动态链接库（Dynamic Link Library）组成，动态链接库文件的扩展名为“DLL”，这是 Windows 系统中的一种特殊的可执行文件。在动态链接库中包含有将某些函数预先编译成目标文件的程序模块，当程序访问所需的动态链接库时，Windows 才能确定被调函数的地址并将其连接到应用程序之中。使用动态链接库可以使应用程序实现代码和资源的共享。

本书主要使用的 API 函数动态链接库如下。

- gdi32.dll——对系统的图形图像进行操作；
- kernel32.dll——对计算机系统资源进行操作；
- user32.d11——对操作系统进行操作；
- mpr.dll——对计算机接口进行操作；
- user32.dll——对用户接口进行操作；
- winmm.dll——对多媒体进行操作；
- rasapi32.dll——远程访问连接。

在这些动态链接库中均包含数量不等的 API 函数。

1.2 API 函数的使用方法

API 函数在使用上并不神秘，下面这个程序的功能是使用 API 函数在工作界面上绘制一个圆弧。

程序清单：

```
Option Explicit
Private Declare Function Arc Lib "gdi32" (ByVal hdc As Long, ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long, ByVal X3 As Long, ByVal Y3 As Long, ByVal X4 As Long, ByVal Y4 As Long) As Long
Private Sub Form_Load()
    Me.AutoRedraw = True
    Arc Me.hdc, 10, 10, 120, 120, 1, 120, 120, 10
End Sub
```

程序运行效果如图 1.1 所示。

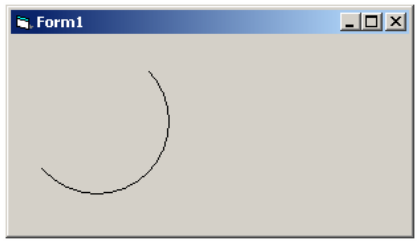


图 1.1 使用 API 函数绘制的圆弧

本节程序参见附盘：程序\第一章\绘制圆弧.vbp

1.2.1 API 函数的声明

在 Visual Basic 程序中调用 API 函数时，必须进行 API 函数的声明。声明 API 函数的作用是确定将要使用的 API 函数的名称（有时还需要写出所用的 API 函数的别名）、该 API 函数所在的文件、函数中使用的参数及参数类型和数据传输方式及所用函数本身的函数类型。

在 Visual Basic 应用程序中进行 API 函数声明后，即可在程序中调用 API 函数，其调用方法与在 Visual Basic 中调用其他函数的方法相同。

声明 API 函数的语句格式如下：

[Public/Private] Declare Function API 函数名 Lib "该函数所在的文件名" [Alias "该函数的别名"][（变量名及变量类型说明）] As API 函数类型说明

这里的 **Public** 和 **Private** 用于确定该 API 函数的使用范围。如果将 API 函数写在模块里作为公用函数使用，则在对 API 函数进行说明时，需要使用 **Public** 说明符或不用任何说明符。如果不进行函数使用范围的说明，系统默认该函数为公用函数。下面是一个声明 API 函数的具体实例。

```
Declare Function FlashWindow Lib "user32" (ByVal hwnd As Long, ByVal bInvert As Long) As Long
```

这个声明 API 函数的语句的作用如下：

- Declare 前面没有 Private 说明本函数可以被工程中的任何程序调用。
- 函数名是 FlashWindow，不需使用别名。
- 该函数被包含在名为 user32 的动态链接库中。
- 函数使用了两个变量 hwnd 和 bInvert，这两个变量的类型为 Long，参数的传递方式是 ByVal（按值传送）。
- FlashWindow 本身的类型是 Long 型。

如果仅在本窗体程序中使用 API 函数，需要将 API 函数声明为 **Private**，此时如果不进行 API 函数使用范围的声明或将其使用范围声明为 **Public**，程序运行时系统会给出如图 1.2 所示的出错提示。



图 1.2　错误声明 API 函数使用范围时的提示

一般，对于供单一程序使用的 API 函数，直接将所用的 API 函数放在程序里并将其作用范围声明为 **Private** 即可。对于供多个程序共用的 API 函数将其放在模块中并将其使用范围声明为 **Public** 较好。

如果某 API 函数没有返回值，可以将其声明为 Sub，如：

```
Declare  Sub  aaa  Lib "aaa.dll"() As Long
```

对于引用 Windows 核心动态链接库的 API 函数，在进行函数声明时可以不包含所用到的动态链接库的文件扩展名.dll。如果需要调用其他的动态链接库，则在 Lib 子句中必须指定这个文件的所在路径及扩展名。如：

```
Declare Function Abc Lib "D:\XXX\Abc.dll" (ByVal X As Long, ByVal Y As Long) As Long
```

有的 API 函数在调用时需要使用函数别名，这时需要使用 Alias 说明函数的别名，函数别名的最后一个字符通常是“A”。如 SetWindowsText 函数的别名是 SetWindowsTextA。字母“A”是设计人员的一种习惯的别命名约定，表示这个函数属于 ANSI 版本。如：

```
Private Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" (ByVal hwnd As
Long, ByVal nIndex As Long) As Long
```

当需要使用函数别名，而在声明 API 函数时没有对该 API 函数的别名进行说明，则在调用这个 API 函数时，会产生函数调用错误，如图 1.3 所示。



图 1.3　函数调用错误

许多 API 函数在工作时，有时还要涉及到一些常数。例如，在调用 API 函数对注册表中名为 HKEY_LOCAL_MACHINE 主键中的数据项进行操作时，可以安排相应的语句如下：

```
RegCreateKey &H80000002, "SOFTWARE\ODBC\ODBC.INI\人员简历", hKeyHandle
```

这条语句中的“&H80000002”称为主键常数，用来确定所需操作的注册表中主键。与主键有关的问题在本书“6.3　主键及项类型的对应常数”中有详细介绍。

为提高程序的可读性和便于程序的维护，可以使用字符串与这些常数相对应，以便明确

其在程序中的作用。这些字符串称为字符常数或字符标识。如果需要使用字符常数时，除了对 API 函数本身进行声明外，还要对这些字符常数进行说明。

对于前面的这条语句，可以事先设定好一个表示该主键的字符串，这里不妨直接使用主键名称 HKEY_LOCAL_MACHINE。然后使用语句 Const 将这个准备用于表示主键的字符串说明为与 HKEY_LOCAL_MACHINE 主键对应的常数&H80000002。当这个字符串被说明之后，在程序中直接使用该字符串代替相应的常数即可。

与语句

```
RegCreateKey &H80000002, "SOFTWARE\ODBC\ODBC.INI\人员简历", hKeyHandle
```

相对应的常数说明和语句调用如下：

```
' 常数说明
Const HKEY_LOCAL_MACHINE = &H80000002
...
' API 函数调用
RegCreateKey HKEY_LOCAL_MACHINE, "SOFTWARE\ODBC\ODBC.INI\人员简历", hKeyHandle
```

这个操作不需要 API 函数的返回值，所以不必用括号将函数参数括起来。如果需要 API 函数的返回值，则需按照下面的格式设置函数调用。

变量＝API 函数（函数参数）

用于接收 API 函数返回值的变量类型，必须与所使用的 API 函数的类型相同。

1.2.2 API 函数的变量

1．变量类型及数据的传送方式

在进行 API 函数声明时，除需要说明该函数的函数名、函数别名及这个函数所在的文件外，还要说明在该 API 函数中使用的变量类型及数据的传送方式。API 中使用的变量的个数和类型及数据的传送方式等是在编写这个 API 函数时确定的。

例如对于名为“ ChangeDisplaySettings ”的 API 函数，其函数声明如下：

```
Private Declare Function Lib ChangeDisplaySettings "user32" Alias "ChangeDisplaySettingsA"
(lpDevMode As Any, ByVal dwFlags As Long) As Long
```

这个函数使用的变量名称、类型和数据传输方式如表 1.1 所示。

表 1.1　　ChangeDisplaySettings 函数的变量设置

变量名称	类型	数据传输方式
lpDevMode	Any	按地址
dwFlags	Long	ByVal

如果某个变量需要按地址进行数据传输，其变量名前面用于说明数据传输方式的说明符可以省略。在这个函数中变量 lpDevMode 要求按地址传输数据，在进行函数声明时省略了对应的传输方式说明符“ ByRef ”。

API 函数中的变量与普通函数中的变量相同都是“ 形式参数 ”，其变量名可以是任意有效字符，但是使用能表示变量作用的字符较好。

2．结构型变量

对于某些操作，有时需要将不同类型的变量组合成一个有机的整体，以便于在程序中引用。在这种情况下设置的变量是一种特殊类型的变量，这种变量一般称为“结构型”变量。

声明结构型变量的语句格式如下：

变量作用范围说明 Type 结构名称

**　变量列表**

End Type

例如：

```
Private Type Student
  Name As String
  Age As Integer
End Type
```

该语句的作用如下：

- 通知系统有一个名为“Student”的结构型变量。
- 这个结构型变量的作用范围是“Private”，即在本模块中有效。如果在声明结构型变量时不作变量作用范围的说明，则这个结构型变量的作用范围是“Public”。
- 在这个结构型变量中有名为“Name”和“Age”两个具体的变量。
- 其中变量“Name”为字符型变量，“Age”为整型变量。

结构型变量的引用格式为：**结构名．结构中的变量名＝数据**

这里的“.”可以理解为“的”。例如：Student.Name＝"李　明"，语句执行的操作是将字符“李　明”赋值给 Student 中“的”Name。

这种数据类型类似于数据库中的字段名和字段类型。在其他的程序设计语言中，也可以设置这种类型的变量。在不同的程序设计语言中，对“结构型”变量的称呼不完全相同，比如“Turbo C”称这种类型的变量为“结构体”。

1.2.3　句柄

在 MSDN 中将句柄解释为：由操作环境定义的一个惟一的整数值，它被程序用来标识或者切换到对象，如窗体或控件等。

简单地说，这句话的意思是“**用来标识或切换对象的整数值**”。句柄的作用是指定 API 函数的工作对象，如窗体或控件等。在 API 中常用的句柄如表 1.2 所示。

表 1.2　　常用句柄

句柄名称	应用对象
hbitmap	位图
hbrush	填充区域
hDC	设备
hfont	字体
句柄名称	应用对象
hicon	图标

续表

句柄名称	应用对象
hmen	内存
hmenu	菜单
hwnd	窗体对象
hdlg	对话框

如：Private Declare Function ShowScrollBar Lib "user32" (ByVal hwnd As Long, ByVal wBar As Long, ByVal bShow As Long) As Long

即表示该函数对窗体进行操作。

假设现在需要在窗体上设置一个文本框，在文本框中显示窗体的标题，如图 1.4 所示。

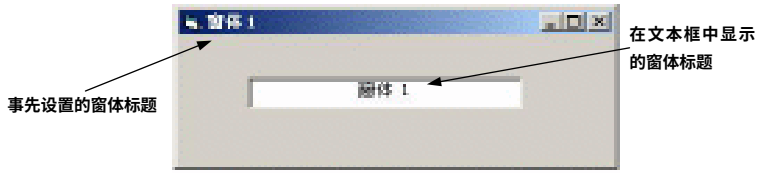


图 1.4　在文本框中显示窗体标题

可以设置窗体属性如表 1.3 所示。

表 1.3　　窗体属性

属性	值
Caption	窗体 1

直接使用对象编程时的程序如下：

```
Option Explicit
Private Sub Form_Load()
  Text1 = Form1.Caption
End Sub
```

程序执行时通过 Form1.Caption 得到窗体标题。执行 Text1=Form1.Caption 操作时，将这个窗体标题赋值给文本框 Text1，这样就在文本框 Text1 中显示出表示窗体标题的文字。

如果使用 API 函数进行上述操作，程序如下：

```
Option Explicit
' API函数声明
Private Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal hwnd As Long, ByVal lpString As String, ByVal cch As Long) As Long
Private Sub Form_Load ()
    Dim Title As String
    Title = String(100, Chr$(0))
```

```
    GetWindowText  Me.hwnd, Title, 100
    Title = Left$(Title, InStr(Title, Chr$(0)) - 1)
    Text1 = Title
End Sub
```

程序中的“Me.hwnd”即为窗口句柄。

本节程序可参见附盘：程序\第一章\窗口句柄.vbp

这里函数中对变量的声明如表 1.4 所示。

表 1.4　　变量声明

变量名	作用	类型	数据传输方式
hwnd	句柄	Long	ByVal
lpString	字符串	Long	ByVal
cch	字符串长度	Long	ByVal

在进行函数调用时的语句如下：

```
GetWindowText Me.hwnd, Title, 100
```

这条语句的作用是将窗口句柄，即本窗体的标题文字传输给变量 Title，并限制其长度为 100 个字符。

Title 得到该句柄后，去掉其中的空字符。最后将其显示在文本框 Text1 中。

句柄实际上是指向指针的指针。指针是一种内存地址，应用程序启动后，这个程序中的所有对象都是驻留在内存中的。表面上看只要获知这个内存的首地址，就可以随时使用这个地址访问对象。但是 Windows 是一个以虚拟内存为基础的操作系统。在这种系统环境下，Windows 内存管理器经常在内存中移动对象，以此来满足各种应用程序的内存需要。对象被移动，该对象的地址就变化了。本程序运行 3 次时其句柄数值的变化如图 1.5 所示。

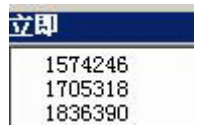


图 1.5　Me.hwnd 在数值上的变化

为了解决应用对象在内存中的地址变化问题，Windows 操作系统为应用程序设置了一些内存存储地址，用来专门登记应用对象在内存中的地址变化，而这个地址（存储单元的位置）本身是不变的。当 Windows 内存管理器移动了对象在内存中的位置后，把对象新的地址告知这个句柄地址并保存。这样系统只需记住这个句柄地址就可以间接地知道对象具体在内存中的哪个位置。这个地址是在对象装载（Load）时由系统分配的，当系统卸载时（Unload）又释放给系统。由于句柄表示的是对象在内存中的地址，在程序中不要对地址进行数学运算。

1.2.4　在 Visual Basic 中使用 API 函数的两个实例

下面举两个在 Visual Basic 中使用 API 函数的实例。

1．获得系统名称

通过调 Windows 的系统信息程序可以显示计算机硬件系统和软件系统的有关信息，计算机系统信息中的系统摘要信息如图 1.6 所示。

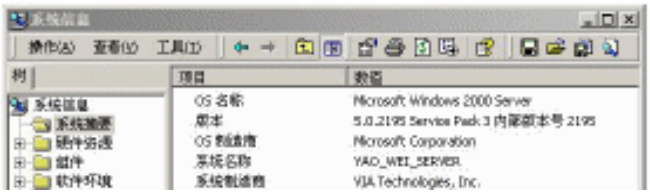


图 1.6　在系统信息中显示的系统名称

在程序中通过调用相关的 API 也可以将有关信息显示出来，显示“系统名称”的程序的工作界面如图 1.7 所示。



图 1.7　程序运行后获得的系统名称

获得系统名称的 API 函数为 GetComputerName，这个函数的情况将在第 2 章中再进行详细介绍，具体情况见第 2 章 “2.1.3 获得计算机名和用户名”一节。

程序清单：

```
Option Explicit
' API函数声明
Private Declare Function GetComputerName Lib "kernel32" Alias "GetComputerNameA" (ByVal
lpBuffer As String, nSize As Long) As Long
Private Sub Form_Load()
    Dim dwLen As Long
    Dim strString As String
    dwLen = 20
    strString = String(dwLen, " ")
'  调用 GetComputerName 得到系统名称
    GetComputerName strString, dwLen
'  在文本框 Text1 中显示系统名称
    Text1 = strString
 End Sub
```

本节程序可参见附盘：程序\第一章\获得系统名称.vbp

2．在文本框中加入滚动条

文本框常用于文字的输入或显示，将文本框的 Multiline 的属性设为“True”时可以显示多行文字，如图 1.8 所示。这时需要使用方向键进行文字的换行操作。在常用的文字处理软件中，其文字编辑区的右侧和下侧都设有滚动条用于调整文字的显示，如图 1.9 所示。



图 1.8　普通的多行显示的文本框



图 1.9　记事本的工作界面

在将文本框控件的 Multiline 的属性设为“ True ”之后，再进行文本框的滚动条的设置，即可在文本框中加入滚动条，如图 1.10 所示。



（*a*）选择滚动条样式　　（*b*）滚动条被加入到窗体

图 1.10　滚动条的选择和加入窗体

除了通过属性调整在文本框中设置滚动条外，还可以在程序中使用 API 函数对文本框的滚动条进行设置。在文本框中加入滚动条的函数如表 1.5 所示。

表 1.5　　在文本框中加入滚动条的 API 函数

函数名称	作用
ShowScrollBar	在文本框中加入滚动条

在表 1.5 中所列出的 API 函数的函数声明和变量使用如下：

✓　在文本框中加入滚动条函数 ShowScrollBar

函数声明：

```
Private Declare Function ShowScrollBar Lib "user32" (ByVal hwnd As Long, ByVal wBar As Long, ByVal bShow As Long) As Long
```

函数变量设置如表 1.6 所示。

表 1.6　　Sho wScrollBar 函数的变量设置

变量名称	作用	类型	数据传输方式
hwnd	对象句柄	Long	ByVal
wBar	滚动条形式	Long	ByVal
bShow	是否显示	Long	ByVal

滚动条的显示形式由常数确定，确定滚动条显示形式的常数如表 1.7 所示。

表 1.7　　用于确定滚动条显示形式的常数

常数值	字符标识	作用
0	SB_HORZ	设置水平滚动条
1	SB_VERT	设置垂直滚动条
3	SB_BOTH	水平滚动条和垂直滚动条同时设置

假设在文本框 Text1 中加入垂直滚动条，则设置 ShowScrollBar Text1.hwnd,SB_VERT,True 即可。在文本框中加入滚动条后的程序执行结果如图 1.11 所示，