图书在版编目(CIP)数据

计算机综合培训教程/中国 IT 培训工程编委会编.—珠海：珠海出版社，2002.1
ISBN7－80607－823－1
Ⅰ.电... Ⅱ.中... Ⅲ.电子计算机－基本知识 Ⅳ.TP3

中国版本图书馆 CIP 数据核字（2000）第 49639 号

计算机综合培训教程

作　　者 ■中国 IT 培训工程编委会
选题策划■孙建开
终　　审 ■成平
责任编辑■雷良波
封面设计■JUNGLE STUDIO

出版发行●珠海出版社
社　　址 ●珠海香洲银桦新村 47 栋 A 座二层
电　　话 ●2515348　邮政编码 ●519001
印　　刷 ▲广东科普印刷厂
开　　本 ▲787×1092mm　1/16
印　　张 ▲80.75　字数 ▲1600 千字
版　　次 ▲2002 年 1 月第 1 版
　　2002 年 1 月第 1 次印刷
印　　数 ▲1－5000 册
ISBN7－80607－823－1/TP.12
定　价：34.80 元

《中国 IT 培训工程》丛书简介

生存和发展，一直是 13 亿中国人面临的最重要的问题。这也是电脑书畅销不衰，计算机等级考试应考者每年超过五百万人,国家教育部把信息技术教育作为中小学必修科目,社会上各类电脑培训学校红红火火的最大原因。

为顺应市场，引导市场，珠海出版社与广州网垠公司和香港恒明出版有限公司共同策划投资 500 余万元，全国近千家电脑培训学校联袂推出《中国 IT 培训工程》，该工程分四大板块：

一、《中国计算机培训标准教材》。这是一块极具潜力的市场，计算机不仅在生产、科研、办公、教育、国防、影视、通讯等领域得到了广泛的应用，而且正在以人们始料不及的速度走进千家万户。为了满足非计算机专业人员及初学者的需求，满足职业高中、技工学校、中等专业学校、军民两用人才培训、下岗人员培训的广泛需要，我们编写了这一系列标准教材。它不仅是各类计算机培训学校的首选教材，还可作为大中专院校学生和各类成人教育的参考用书，更可作为广大用户实战操作的必备工具书。

本板块细分为：

《流行软件全面学习教程》20 余册；　　《电脑应用培训教程》20 余册；

《电脑综合培训教程》20 余册；　　《电脑超级培训学院》20 余册。

二、《中小学信息技术课教材及辅导读物》。清华、北大、中山大学的精英及资深专家均参与了教材及辅导读物的编写。本套教材及辅导读物力争得到教育部的认可，成为国家教育部指定的全国中小学校信息技术课的标准教材。

本板块细分为：《中小学生信息技术标准教程》共 9 册；

《中小学教师计算机培训教程》共 4 册；

《电脑小专家》（彩版）共 10 册。

三、《全国计算机等级考试完全版》。本套教材属全国计算机等级考试命题研

究组编写、为教育部考试中心指定教材辅导书及光盘，是每年五百万考生的必备书。最近，人民日报、光明日报、新民晚报、电脑报、北京青年报、新闻出版报、千龙网等多家媒体相继介绍了珠海出版社近期出版发行的《全国计算机等级考试完全版》等电脑图书及光盘，在全国各地引起了强烈的反响。

本板块细分为：

《全国计算机等级考试完全版》（配光盘），共 16 册；

《全国计算机等级考试完全版》（印刷版）共 16 册（印刷版以超低定价发行）。

中国 IT 培训工程丛书的特点是：一、权威性。二、垄断性。三、内容新。四、实用性强。五、印刷质量一流。六、定价合理。七、分印刷版和配光盘版。八、品种齐全。九、销售渠道完善。

机会和挑战

信息时代的来临，令人新奇而又陌生，兴奋而又不安，它充满了竞争，每一个中国人都必将面临挑战！新世纪里，拥有新观念、新知识、新经验，意味着机遇；否则意味着淘汰！从现在起，深谋远虑，从心态到技能，从观念到知识，主动出击，长远计划，充实自己，不断掌握的专业知识和职业技能，提高自身的综合素质和竞争能力。你，准备好了吗？残酷的竞争摆在你我的面前，在这能力本位的社会转型期，我们不能不学会电脑！掌握一技之长--学会简单的文案处理，专业的广告设计，打字排版，电脑维护，网页制作，或者高级程序设计？事实证明，你我的新人生就从《中国 IT 培训工程》开始……

中国 IT 培训工程编委会

第一部分　VB 趣味编程

第一章　轻轻松松学编程

1.1　编程入门

1.1.1　看实例学 VB 6.0(关于 VB 语言和怎样学习 VB)

VB6.0 全称为 VisualBasic6.0 ，(图 1-1)是微软公司推出的可视化编程工具 MSDN 之一，是目前世界上使用最广泛的程序开发工具。如果你是一个对编程一无所知，而又迫切希望掌握一种快捷实用的编程语言的初学者，那选择 VB 6.0 没错的 。即使考虑到 VB 程序本身编译和运行效率较低的不足(嘻嘻速度现在不是问题吧)，单是它的快捷的开发速度，简单易学的语法，体贴便利的开发环境，它仍不失一款优秀的编程工具，是初学者的首选。

图 1-1

也许你会问，我以前没学过任何一种语言，我能快速上手吗？别担心，没问题 VB 的语法的和 QBASIB 语言是基本相同的，也就是说它的语法是最容易被初学者所接受的。另外 VB 提供的是可视化的开发环境，我们可以像搭积木一样构建出程序的界面，而且 VB 提供了丰富的控件组，省去了我们自己写代码实现这些效果的麻烦，这样我们就能把更多的精力放在程序功能的实现上，所以 VB 学起来简单，用起来方便。

接着看看 VB 语言的前景，在目前各种编程语言共存的时代，VB 会不会落伍呢？当然不会了，在我写这篇文章的同时，微软已经透露了 VB7.0 将完全面向对象的消息，可以肯定下一代 VB 的功能一定会强大很多，我们这些所谓的 VB 程序员总算可以放心了 ，VB 不会落后于时代，毕竟它是使用人数最多的优秀的开发工具。

好了，侃了这么多关于 VB 的台前幕后，总之是为想学编程的你树立信心，编程一点都不

难，只要你决定了开始，就让我们一起踏上愉快的编程之旅吧。

接下来谈谈怎样学习 VB，先说说“看实例学 VB6.0”系列教程，它是面向编程初学者的 VB 入门教程，这个教程的特点是抛开晦涩难懂的概念和语法，不做内容上的堆积和罗列，而是采用了每节一个生动有趣的小例子的形式，每个小例子中会涉及到一个或几个 VB 编程的知识点(可能是控件，也许会是某个函数或编程小技巧)，使你快速入门。希望你学完此教程后能达到下述的效果：

从对编程一窍不通或从未接触过编程的状态，通过学习能够对 VB6.0 的编程环境比较熟悉，掌握 VB 开发界面的使用方法；对 VB 语言的基本语法大致了解，知道常见的语句的意义；学习 VB 常用控件的使用方法，并能将它们灵活运用到应用程序中；能开发简单的 VB 程序。到那时你已经能够继续深入的学习 VB 编程，可以继续参与到程序设计栏目其他版块的学习中去，嘻嘻，目标就是这样啦。

然后谈谈学习编程的方法，万事开头难，刚刚开始，遇到些困难没关系，慢慢来。编程是一个不断学习，不断积累的过程，编程的乐趣也正是存在于学习的过程中。我们每学一点，就赶快把它用到实际的程序中去，自己多学多用多实践，水平才能不断提高，这就是“学以致用”。

另外，编程涉及到很多的知识，像操作系统的、软件工程的、硬件系统的以及编程思想等各个方面，这就需要我们多看看这方面的资料，扩充自己的知识面。

还有如果学习过程中遇到了什么问题，或者有什么好的心得，你可以到洪恩的“网上交流”的“编程技术”版去提问求助或是发表文章，那里有许多编程高手可以为你答疑，还有许多同样的初学者一起交流。

“求知无限”是网上学习的特点，如果你觉得自己能够更深的学习ＶＢ或是其他编程的知识时，“程序设计”栏目的其他版块将是理想的去处，希望我们能在这样的学习环境中不断进步。

1.1.2 认识一下 VB 6.0 的编程环境

VB6.0 采用可视化的编程环境 ，它好学易用，运行 VB6.0 后，会出现如下图所示的窗口，呵呵，看起来蛮复杂的，好多的按钮、菜单、小窗口，别担心，待会儿我们会一起学习这些东东都是干什么用的。

先来解决一个小问题 ，怎么新建一个 VB 的工程呢？方法是这样的，在程序启动时出现的“新建工程”对话框中选择“标准 EXE”并点击确定，就能直接新建一个工程，如果你跳过了这个对话框则也可从“文件”菜单中选择“新建工程”重新调出此对话框。

下面一起来看看 VB 的编程环境，(如图 1-2)虽然看起来复杂，但可以把它分为几个部分，每个部分都有自己特定的功能，这样我们就清楚多了。其实 VB 这是通过这样一个界面把相近或同类的功能组合在一起的，它使我们在设计程序时能方便的控制程序的方方面面。

窗体的最上层是“VB 6.0 的菜单 ”和“便捷工具按钮”，(如图 1-3)菜单中包含了所有的 VB 提供的功能的选项，而其中一些常用的功能或操作选项则被提取出来放在了“便捷工具按钮”中，通过点击这些快捷按钮可以加快程序开发的速度，下图标出了常见的工具按钮的作用，记着使用它们哦。

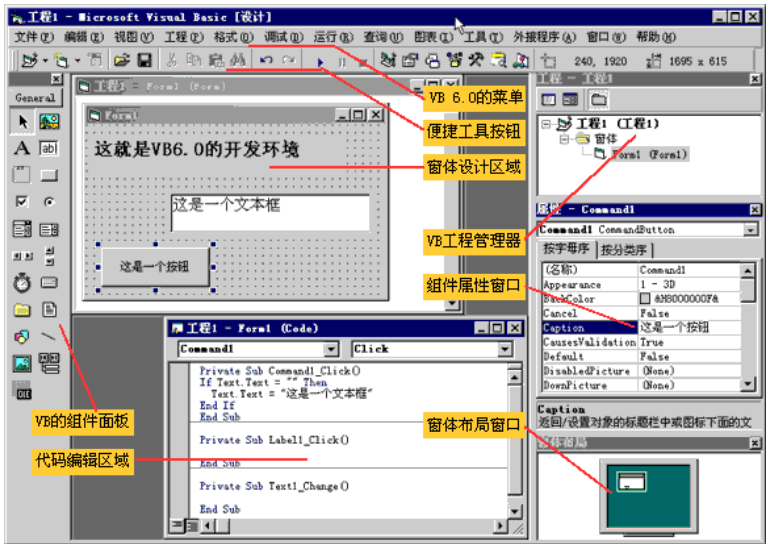

图 1-2

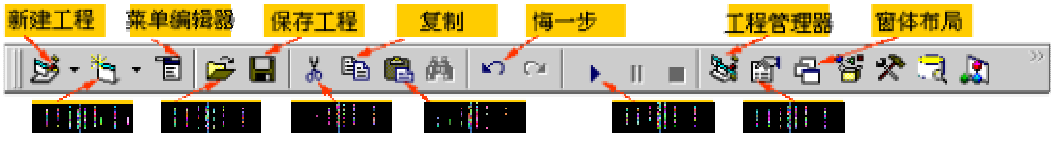

图 1-3

接着看看“VB 的组件面板”和“窗体设计区域”。VB 提供了方便的窗体设计区域，它位于整个编程窗口的中间。我们可以在这个区域中搭建出美观实用的程序界面。试着从“VB 的组件面板”中找到“按钮 CommandButton”，点击选中后，再到“窗体设计区域”的窗体上点一下。这时，窗体上会出现一个按钮。这样我们就把按钮添加到我们程序的界面中了。嘻嘻，忘了告诉你哪里是“VB 的组件面板”了，就是最左边的那一条包含许多看起来眼熟的小东西的区域。它是 VB 提供给我们的标准的编程组件(控件)。它把程序设计中常常用到的诸如按钮，图片框，列表框等等东西作好了放在那里。我们如果要使用只须添加到窗体中即可。

如图 1-4 有三个从上到下排列的小窗口，它们分别是：

图 1-4

法还不是十分了解，可以继续下一节专门对“VB 语法”的讲解，别着急，慢慢来，很快就能入门的。

1.1.3　看实例学 VB 6.0--VB 的简单语法学习

VB 语言的语法和 QBASIC 基本一致，可以说如果你能读懂 QBASIC 程序，那你读 VB 的程序是没有问题的，这一节中我们只是简单的讲一讲 VB 的语法，因为在今后的各个实例中我们会不断接触到新的语法知识，这也是一个积累的过程。

如果你还记得上节中的内容，我们提到过如果在“窗体设计区域”的某个组件(控件)上双击鼠标，会出现“代码编辑窗口”，也就是如下图的窗口，（如图 1-7），我们可以在其中输入或是修改程序的代码。在我们双击控件到写入代码的这个过程中，我们涉及到了一个很重要的概念，这是 VB 与 QBASIC 的主要差别之一，大家知道在 QBASIC 包括 Tubro C，TubroPascal 等语言中程序从一开始就需要我们一点点的写代码来实现所有的功能，比如程序运行的界面，输入输出，键盘控制等，它们在程序中体现为一个个的过程或者是函数及子程序，它们都是完全面向过程的编程语言；但是现在的 VB、VC、DELPHI 等编程语言，都提供了可视化的编程环境，备有一整套常见的组件(控件)供我们使用，这些控件可以看作是对象(VB 严格的说不是完全面向对象的编程语言)，我们写程序时会直接与这些对象打交道，而不像原来。所以在 VB 中大部分的代码是与这些相关的，例如下图所示的一段代码，它是我们在一个按钮(Command)上双击鼠标而由 VB 自动产生的一个子过程，Private Sub Command1_Click()是子过程的开始，End Sub 表示子过程的结束。然后我们可以在子过程中添加程序代码，也就是由我们自己来写当按钮被点击时，程序所做出的响应。

图 1-7

下面我们便来分析一下这段代码。

VB 中声明一个变量使用“Dim 变量名 As 变量类型”的方法，一般在程序中使用的到的变量都应该有变量声明，如上例中的 a、b、i 分别被定义为整型和字符串型变量，VB 中的变量类型有以下几种：

数据类型	说明
Byte	二进制数
Boolean	真假值
Integer	整数
Long	长整型
Single	实数
Double	双精度型
Currency	货币
Date	日期和时间
Object	对象
String	字符串
Variant	可变类型

控制结构和循环结构是程序设计中常见的两种结构。

计算机在运行程序的时候，执行语句的顺序是从上向下的。 有些简单程序可以只用单向流程来编写，但稍微复杂的程序就要靠控制语句来控制程序执行的流程。控制结构主要有两种，分支结构和循环结构。

分支结构：分支结构事实上是一种选择，在不同的条件下选择执行不同的程序段。实现分支结构的语句有很多，最常用的是 if……then……else……。例如：

……

if 条件 then

{程序段一}

else

{程序段二}

end if

……

如果条件满足了，电脑将执行程序段中的语句，然后跳过程序段二，执行下面的语句。如果条件不满足，电脑将跳过程序段一种的语句，执行程序段二，然后继续执行下面的语句。

循环结构：循环结构就是让电脑反复的执行某一程序段落若干次。 用 Do……Loop 可以循环重复执行一语句块，且重复次数不定。 在已知循环次数的条件下，用 For……next 可以反复执行统一语句块。

以上两种结构你可以参照图中所示的代码，仔细体会一下。

第二章　趣味程序 12 例

2.1　趣味程序源码注解(CommomDialog 使用全解)

CommonDialog 控件是常用的一个控件，它为我们提供了打开、另存为、字体、颜色、打印、帮助等几种类型的标准对话框，本例演示了所有这些类型的对话框的使用方法。

为了学习方便，提供的源码已经作了详细的中文注释，看看源码框中的代码：

名称	作用
CdlTest	通用对话框
CmdOpen	“打开”对话框按钮
CmdSave	“另存为”对话框按钮
CmdFont	“字体”对话框按钮
CmdColor	“颜色”对话框按钮
CmdPrint	“打印”对话框按钮
CmdHelp	“帮助”对话框按钮

```
'当“颜色”对话框按钮被按下时
Private Sub CmdColor_Click()
On Error Resume Next
CdlTest.CancelError = True
CdlTest.Flags = cdlCCRGBInit
CdlTest.ShowColor
If Err = cdlCancel Then Exit Sub
TextBoxColor.ForeColor = CdlTest.Color
End Sub
'当“字体”对话框按钮被按下时
Private Sub CmdFont_Click()
On Error Resume Next
'当用户按下“取消”按钮，返回一个错误信息，这样使我们可以对其进行控制
CdlTest.CancelError = True
'此句必须要
CdlTest.Flags = cdlCFBoth + cdlCFEffects
'显示“字体”对话框
CdlTest.ShowFont
'出现“取消”错误时，跳出
If Err = cdlCancel Then
Exit Sub
```

```
Else
将 TextBox 的字体属性根据“字体”对话框的变化作相应设置
如果用户选择了字体才将字体改变，避免字体为空的错误
If CdlTest.FontName <> "" Then
TextBoxFont.FontName = CdlTest.FontName
End If
TextBoxFont.FontSize = CdlTest.FontSize
TextBoxFont.FontBold = CdlTest.FontBold
TextBoxFont.FontItalic = CdlTest.FontItalic
TextBoxFont.FontStrikethru = CdlTest.FontStrikethru
TextBoxFont.FontUnderline = CdlTest.FontUnderline
End If
End Sub
'当“帮助”对话框按钮被按下时
Private Sub CmdHelp_Click()
On Error Resume Next
'设置 HelpCommand 属性 ， 显示 Visual Basic 帮助目录主题
CdlTest.HelpCommand = cdlHelpForceFile
'指定帮助文件
Dim fullpath As String
If Right(App.Path, 1) = "\" Then ' 若 App.Path 为根目录
fullpath = App.Path + "test.hlp"
Else
fullpath = App.Path + "\" + "test.hlp"
End If
上面是得到应用程序所在路径的小技巧
CdlTest.HelpFile = fullpath
显示“帮助”对话框
CdlTest.ShowHelp
End Sub
'当“打开”对话框按钮被按下时
Private Sub CmdOpen_Click()
'出现错误时跳到下一语句
On Error Resume Next
CdlTest.CancelError = True
'属性 DialogTitle 是要弹出的对话框的标题
CdlTest.DialogTitle = "打开文件"
'缺省的文件名为空
CdlTest.FileName = ""
```

```
'属性 Filter 是文件滤器，返回或设置在对话框的类型列表框中所显示的过滤器。
'语法 object.Filter [= 文件类型描述 1 |filter1 |文件类型描述 2 |filter2...]
CdlTest.Filter = "文本文件(.txt)|*.txt"
'Flags 属性的用法依据不同的对话框而变，详细使用需要查找联机帮助手册
CdlTest.Flags = cdlOFNCreatePrompt + cdlOFNHideReadOnly
CdlTest.ShowOpen
If Err = cdlCancel Then Exit Sub
TextBoxOPen.Text = CdlTest.FileName
End Sub
'当“打印”对话框按钮被按下时
Private Sub CmdPrint_Click()
On Error Resume Next
CdlTest.CancelError = True
'显示“打印”对话框
CdlTest.ShowPrinter
If Err = cdlCancel Then Exit Sub
End Sub
'当“保存”对话框按钮被按下时
Private Sub CmdSave_Click()
On Error Resume Next
CdlTest.CancelError = True
CdlTest.DialogTitle = "保存文件"
CdlTest.FileName = ""
'解释见上面
CdlTest.Filter = "文本文件(*.txt)|*.txt"
CdlTest.Flags = cdlOFNCreatePrompt + cdlOFNHideReadOnly
CdlTest.ShowSave
If Err = cdlCancel Then Exit Sub
TextBoxSave.Text = CdlTest.FileName
End Sub
```

如果控件面板中没有 CommonDialog **控件的小图标，必须先从“添加控件对话框”中添加，下面我们来看看怎样调用不同类型的对话框**。

CommonDialog **控件有一系列的** Show **方法，例如**：ShowOpen、ShowSave、ShowFont、ShowColor、ShowPrinter、ShowHelp **等，这些方法的使用语法是类似的，如下所示**：

object.ShowOpen，**我们只需在程序中写入这个语句，就能调出“打开”对话框，同样也能调用其它类型的对话框**。

而 CommonDialog 控件的属性是和不同的对话框类型紧密相关的，有些属性只适用于某一类对话框，有些属性在不同的对话框中的属性是有差别的，所以下面分类列出了和不同对话框相关联的属性的用法。

与 ShowOpen、ShowSave 方法相关的属性：

●FileName 属性：返回或设置所选文件的路径和文件名，如果在使用 Show 方法以前使用 FileName 属性，则设定了对话框的默认文件名；如果是在以后使用则返回选择的文件名。

使用语法是 ： CommonDialog.Filename[=pathname]

●Filter 属性：返回或设置在对话框的类型列表框中所显示的过滤器(也就是限定打开或保存为的文件类型)，它的使用语法是：

object.Filter [= 描述文字 1 |过滤标示 1 |描述文字 2 |过滤标示 2]

其中描述文字为任意文字 ，而 过滤标示则采用*.文件后缀(例如：*.bmp)的格式，描述文字和过滤标示之间用“|”隔开。

●DefaultExt 属性：为该对话框返回或设置缺省的文件扩展名，也就是当我们没有指定打开或保存的文件类型时，按 DefaultExt 属性所设置的扩展名为默认值。

与 ShowFont 方法相关的属性：

●Color 选定的颜色。为使用此属性，必须先将 Flags 属性设置为 cdlCFEffects。

●FontBold 是否选定“粗体”。

●FontItalic 是否选定“斜体”。

●FontStrikethru 是否选定删除线。

●FontUnderline 是否选定下划线。

●FontName 选定的字体名称。

●FontSize 选定的字体大小。

使用的语法是直接引用 ，比如我们要根据“字体对话框”返回的值设置文本框的字体，则直接采用语句：Text.Font=CommonDialog.FontnName

与 ShowColor 方法相关的属性：

●Color 选定的颜色。为使用此属性，必须先将 Flags 属性设置为 cdlCFEffects。

与 ShowHelp 方法相关的属性：

●HelpCommand 属性 返回或设置需要的联机帮助的类型

●HelpFile 属性 确定帮助文件的路径和文件名

语法是 ： object.HelpFile[= filename]

下面看看 CancelError 属性，它设置当选取“取消”按钮时是否认为出错，使用的语法是：CommonDialog.CancelError[= boolean] (boolean 指布尔型变量)

如果我们把它设为 True，则当使用者选取了“取消”按钮时程序会返回一个 cdlCancel 错误，通过捕捉这个错误并加以处理，我们就能避免程序的出错。

2.2　趣味程序源码注解(窗体卸载时弹出确认对话框)

有时我们希望程序在被关闭前能弹出确认信息，在按下确认按钮后才退出程序。这在一些编辑工具中十分常见，例如在退出 WORD 时，它会询问是否保存文件，这样就能避免丢失文件所造成的损失。本例就是在 VB 中实现此功能的方法。

为了学习方便，提供的源码已经作了详细的中文注释，看看源码框中的代码：

```
程序说明:
'当程序执行到 End 语句或是按下了 Alt+F4 组合键或是
右上角 X 时，VB 会先询问 Sub Form_QueryUnload 接着询问
'Sub Form_Unload(Cancel As Integer)。
如果询问时发现 Cancel 为 0（默认为 0）才会退出程序。
这样我们在 Sub Form_Unload()中把 Cancel 设为非 0 的值
就能避免直接关闭程序。
这是十分有用的，例如我们为了防止正在编辑的文件没
有存盘就退出，可以在程序中加上这段代码，然后另定
'义一个变量记录是否已经存盘，如果已经存盘就正常退
'出，否则提示存盘的信息。
这是一个保险的办法，可用 QueryUnload 事件阻止从 Windows 中的退出。
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
Dim IntR As Integer
IntR = MsgBox("确认要退出程序吗？", vbYesNo, "退出确认")
If IntR = vbNo Then Cancel = -1
End Sub
'Unload事件，当窗体从屏幕上删除时发生
说明:
将 cancel 设置为任何非零的值可防止窗体被删除，但
'不能阻止其它事件，诸如从 Microsoft Windows 操作
'环境中退出等。
'Private Sub Form_Unload(Cancel As Integer)
'Dim IntR As Integer
'IntR = MsgBox("确认要退出程序吗？", vbYesNo, "退出确认")
'If IntR = vbNo Then Cancel = -1
'End Sub
```

我们先来看看 WINDOWS 下程序退出的步骤：当程序执行到 End 语句或是按下了 Alt+F4 组合键或是右上角“X”时，系统会先询问 Sub Form_QueryUnload()接着询问 Sub Form_Unload()。如果询问时发现 Cancel 为 0（默认为 0）才会退出程序。于是我们就得到了避免直接关闭程序的基本编程思路，即在 Sub Form_Unload()中把 Cancel 设为非 0 的值。

其中 Sub Form_Unload()**的语法如下：**

Sub Form_Unload()**的完整的语句为：**Private Sub Form_Unload(Cancel As Integer)，**其中的参数** Cancel **是当系统询问时程序传递给系统的值，它的默认值为** 0，**如果我们不在程序中改变它的值，询问后窗体便自动关闭。为了实现确认后才关闭，我们可在程序中写入下面代码：**

```
Private Sub Form_Unload(Cancel As Integer)
Dim IntR As Integer
IntR = MsgBox("确认要退出程序吗？", vbYesNo, "退出确认")
If IntR = vbNo Then Cancel = -1 ’
End Sub
```

这样当我们点击“X**”或使用“**End**”语句关闭窗体时，便会出现确认对话框，只有选择“确定”才会关闭。然而** Form_Unload **的作用范围是有限的，如果直接退出** WINDOWS **操作系统，采用上面方法便失效了。为了避免这种情况的发生，我们采用** Form_QueryUnload **会更加保险一些，可以在程序中加入如下代码：**

```
Private Sub Form_QueryUnload(Cancel As Integer)
Dim IntR As Integer
IntR = MsgBox("确认要退出程序吗？", vbYesNo, "退出确认")
If IntR = vbNo Then Cancel = -1 ’
End Sub
```

这样就能保证万无一失了。

2.3 趣味程序源码注解(打开对话框中选多文件)

默认情况下，使用通用对话框打开文件时，我们同时只能选中一个文件，而很多流行的软件都支持同时选择多个文件（像 WORD、WINAMP **等）。其实对通用对话框的属性稍做改变，也能实现同样的效果，下面我们一起来做一下。**

为了学习方便，提供的源码已经作了详细的中文注释，看看源码框中的代码：

名称	作用
mnuOpen	**“打开”菜单项**
mnuExit	**“退出”菜单项**
cmdExit	**“退出”按钮**
CommonDialog1	**通用对话框**
Label1-2	**标签**
List1	**显示选中文件的列表框** ListBox

```
Private Sub cmdExit_Click()
Unload Me
End Sub
```

```
Private Sub Form_Load()
End Sub
Private Sub mnuExit_Click()
Unload Me
End Sub
Private Sub mnuOpen_Click()
Dim I As Integer
Dim Y As Integer
Dim Z As Integer
存储文件名的数组
Dim FileNames$()
CommonDialog1.FileName = ""
CommonDialog1.Filter = "All Files|*.*"
为“打开”和“另存为”对话框返回或设置选项。
语法
'object.Flags [= value]
'Flags 属性语法有下列部分:
部分　　描述
'object　　对象表达式，其值是“应用于”列表中的对象。
'value　　如“设置值”中所描述，是为“打开”和“另存为”对话框指定选项的常数或值。
'Value 的设置值是:
常数　　值　　描述
'cdlOFNAllowMultiselect　　&H200　　它指定文件名列表框允许多重选择。运行时，通过按 SHIFT 键以及使用 UP ARROW 和 DOWN ARROW 键可选择多个文件。作完此操作后，FileName 属性就返回一个包含全部所选文件名的字符串。串中各文件名用空格隔开。
'cdlOFNCreatePrompt　　&H2000　　当文件不存在时对话框要提示创建文件。该标志自动设置 cdlOFNPathMustExist 和 cdlOFNFileMustExist 标志。
'cdlOFNExplorer　　&H80000　　它使用类似资源管理器的打开一个文件的对话框模板。适用于 Windows 95 和 Windows NT 4.0。
'cdlOFNExtensionDifferent　　&H400　　它指示返回的文件扩展名与 DefaultExt 属性指定的扩展名不一致。如果 DefaultExt 属性是 Null，或者扩展相匹配，或者没有扩展时，此标志不设置。当关闭对话框时，可以检查这个标志的值。
'cdlOFNFileMustExist　　&H1000　　它指定只能输入文件名文本框已经存在的文件名。如果该标志被设置，则当用户输入非法的文件名时，要显示一个警告。该标志自动设置 cdlOFNPathMustExist 标志。
'cdlOFNHelpButton　　&H10　　使对话框显示帮助按钮。
'cdlOFNHideReadOnly　　&H4　　隐藏只读复选框。
'cdlOFNLongNames　　&H200000使用长文件名。
'cdlOFNNoChangeDir　　&H8　　强制对话框将对话框打开时的目录置成当前目录。
'cdlOFNNoDereferencelinks　　&H100000　　不要间接引用外壳链接（也称作快捷方式）。缺省时，
```

选取外壳链接会引起它被外壳间接引用。

'cdlOFNNoLongNames &H40000 无长文件名。

'cdlOFNNoReadOnlyReturn &H8000 它指定返回的文件不能具有只读属性，也不能在写保护目录下面。

'cdlOFNNoValidate &H100 它指定公共对话框允许返回的文件名中含有非法字符。

'cdlOFNOverwritePrompt &H2 使“另存为”对话框当选择的文件已经存在时应产生一个信息框，用户必须确认是否覆盖该文件。

'cdlOFNPathMustExist &H800 它指定只能输入有效路径。如果设置该标志，输入非法路径时，应显示一个警告信息。

'cdlOFNReadOnly &H1 建立对话框时，只读复选框初始化为选定。该标志也指示对话框关闭时只读复选框的状态。

'cdlOFNShareAware &H4000 它指定忽略共享冲突错误。

```
CommonDialog1.Flags = cdlOFNAllowMultiselect
CommonDialog1.Action = 1
'它指定文件名列表框允许多重选择。运行时，通过按 SHIFT 键以及
'使用 UP ARROW 和 DOWN ARROW 键可选择多个文件。作完此操作后，
'FileName 属性就返回一个包含全部所选文件名的字符串。
'串中各文件名用空格隔开。
CommonDialog1.FileName = CommonDialog1.FileName & Chr(32)
'从返回的字符串中分离出文件名
'经过分离后 FileNames(Y)数组存放着选择的文件名信息
'如果只有一个文件 FileNames(0)=“文件名”
'如果有多个文件 FileNames(0)=“路径名” FileNames(1--y)=“文件名”
'这时我们需要对数组进行处理
Z = 1
For I = 1 To Len(CommonDialog1.FileName)
'InStr函数，返回 Variant (Long)，指定一字符串在另一字符串中最先出现的位置。
'语法 InStr(起点位置, string1, string2)
I = InStr(Z, CommonDialog1.FileName, Chr(32))
If I = 0 Then Exit Sub
ReDim Preserve FileNames(Y)
'Mid函数，返回 Variant (String)，其中包含字符串中指定数量的字符。
'语法 Mid(string, start[, length])
FileNames(Y) = Mid(CommonDialog1.FileName, Z, I - Z)
Z = I + 1
Y = Y + 1
Next
'先清空列表框中已有内容
List1.Clear
```

```
如果只有一个文件 FileNames(0)=“文件名”
If Y = 1 Then
List1.AddItem FileNames(0)
如果文件个数为多个，将“路径”+“\”+“文件名”后作为完整的文件名
Else
For I = 1 To Y - 1
List1.AddItem FileNames(0) & "\" & FileNames(I)
Next
End If
End Sub
```

我们先将公共对话框控件的 Flags **属性设置为常量值** cdlOFNAllowMultiselect，**这样就使用户能够同时选择几个文件。可以通过在按住** ctrl **键的同时单击每个文件名称来选中多个文件，这些被选中的文件名称将被高亮。**Flags **属性还有许多的设置常量，你可以参考下面列表：**

常数	值	描述
cdlOFNAllowMultiselect	&H200	使文件名列表框允许多重选择。运行时，通过按 SHIFT 键以及使用 UP ARROW 和 DOWN ARROW 键可选择多个文件。作完此操作后，FileName 属性就返回一个包含全部所选文件名的字符串。串中各文件名用空格隔开。
cdlOFNCreatePrompt	&H2000	当文件不存在时对话框要提示创建文件。该标志自动设置 cdlOFNPathMustExist 和 cdlOFNFileMustExist 标志。
cdlOFNExplorer	&H80000	它使用类似资源管理器的打开一个文件的对话框模板。适用于 Windows 95 和 Windows NT 4.0。
cdlOFNExtensionDifferent	&H400	它指示返回的文件扩展名与 DefaultExt 属性指定的扩展名不一致。如果 DefaultExt 属性是 Null，或者扩展相匹配，或者没有扩展时，此标志不设置。当关闭对话框时，可以检查这个标志的值。
cdlOFNFileMustExist	&H1000	它指定只能输入文件名文本框已经存在的文件名。如果该标志被设置，则当用户输入非法的文件名时，要显示一个警告。该标志自动设置 cdlOFNPathMustExist 标志。
cdlOFNHelpButton	&H10	使对话框显示帮助按钮
cdlOFNHideReadOnly	&H4	隐藏只读复选框
cdlOFNLongNames	&H200000	使用长文件名
cdlOFNNoChangeDir	&H8	强制对话框将对话框打开时的目录置成当前目录
cdlOFNNoDereferencelinks	&H100000	不要间接引用外壳链接（也称作快捷方式）。缺省时，选取外壳链接会引起它被外壳间接引用
cdlOFNNoLongNames	&H40000	无长文件名
cdlOFNNoReadOnlyReturn	&H8000	它指定返回的文件不能具有只读属性，也不能在写保护目录下面
cdlOFNNoValidate	&H100	它指定公共对话框允许返回的文件名中含有非法字符

cdlOFNOverwritePrompt	&H2	使“另存为”对话框当选择的文件已经存在时应产生一个信息框，用户必须确认是否覆盖该文件
cdlOFNPathMustExist	&H800	它指定只能输入有效路径。如果设置该标志，输入非法路径时，应显示一个警告信息
cdlOFNReadOnly	&H1	建立对话框时，只读复选框初始化为选定。该标志也指示对话框关闭时只读复选框的状态
cdlOFNShareAware	&H4000	它指定忽略共享冲突错误

现在我们要解决关键的问题，怎样把选中文件的信息取回到程序中，加以利用。允许选择多文件后，CommonDialog 控件返回的 Filename 是按以下格式的一个字符串：

文件路径 文件名 1 文件名 2 文件名 3 ...

而如果要打开某个文件，我们需要的是该文件的完整路径，也就是“文件路径+文件名”格式，所以必须通过对字符串的操作把完整的文件名分离出来，方法如下：

1、先把返回的字符串按空格分割放在 FileNames 数组中

```
For I = 1 To Len(CommonDialog1.FileName)
'InStr函数，返回 Variant (Long)，指定一字符串在另一字符串中最先出现的位置。
'语法 InStr(起点位置, string1, string2)
I = InStr(Z, CommonDialog1.FileName, Chr(32))
If I = 0 Then Exit Sub
ReDim Preserve FileNames(Y)
'Mid函数，返回 Variant (String)，其中包含字符串中指定数量的字符。
'语法 Mid(string, start[, length])
FileNames(Y) = Mid(CommonDialog1.FileName, Z, I - Z)
Z = I + 1
Y = Y + 1
Next I
```

2、如果文件个数为多个，将“路径”+“\”+“文件名”后作为完整的文件名

```
For I=1 To Len(CommonDialog1.FileName)
FileNames(I)＝FileNames(0) & "\" & FileNames(I)
Next I
```

通过上面的处理，FileNames 数组中存放的就是我们可以直接使用的完整文件路径了，本例中我们是把它们读出显示在列表框中。

2.4 趣味程序源码注解(剪贴板的使用方法示例)

剪贴板是 WINDOWS 操作系统提供的十分有用的工具，用它可以进行文本和图形复制和粘贴操作，在 VB 中我们使用 Clipboard 对象来操作剪贴板上的文本和图形。本例只是演示了对文本类型的数据进行操作，从中可以看到 GetText、SetText、Clear 三个方法的使用。

为了学习方便，提供的源码已经作了详细的中文注释，看看源码框中的代码：

```
本例演示了剪贴板的使用方法，GetText 、SetText
'Clear 方法是剪贴板的几个方法，我们用它来复制
文本，对于图像以及其它格式的数据的操作，请参考
'VB帮助文件。
Dim DataString As String
Private Sub Command1_Click()
On Error Resume Next
If Text1.Text = "" Then
Msg = "第一个文本框内不能为空"
Exit Sub
Else
Clipboard.Clear
'SetText方法，用于复制文本到剪贴板上
'使用语法：object.SetText data, [format]
'data      必需的。是被放置到剪贴板中的字符串数据。
'Format   可选的。一个常数或数值，按照下列设置中的描述，指定 Visual Basic 识别的剪贴板格式。
'vbCFLink        &HBF00          DDE 对话信息
'vbCFRTF         &HBF01          RTF 格式
'vbCFText        1               (缺省值)文本
Clipboard.SetText (Text1.Text)
End If
End Sub
Private Sub Command2_Click()
'GetText方法，用来取得剪贴板上的文本内容
'使用语法为：Clipboard.GetText([数据的类型])
temp = Clipboard.GetText(vbCFText)
Text2.Text = temp
End Sub
Private Sub Command3_Click()
'Clear方法，用于清空剪贴板上的数据
'使用语法：Clipboard.Clear
Clipboard.Clear
End Sub
```

先来看看 GetText、SetText、Clear **三个方法的语法：**

SetText 用于向剪贴板上写入数据，使用语法如下：

object.SetText data, format

'其中 data **是被放置到剪贴板中的字符串数据；**

'format是一个常数或数值，指定 Visual Basic 识别的剪贴板格式，有三种选择 vbCFLink(&HBF00),DDE '对话信息 ；vbCFRTF(&HBF01)RTF 格式； vbCFText(1) 文本；如果不指定默认为文本。

GetText 用于从剪贴板上读入数据，使用语法如下：

object.GetText (format)

'其中 format 的用法同上

'它的返回值就是剪贴板上的数据

Clear 用于清除剪贴板上的数据，使用语法如下：

'一般情况下如果我们要使用系统剪贴板，总是先清除剪贴板上的数据，然后再写入

程序中当“复制”按钮按下时，我们把文本框 Text1 中的内容写入剪贴板，则在程序中加入如下语句：

```
Clipboard.Clear
Clipboard.SetText (Text1.Text)
```

程序中当“粘贴”按钮按下时，把剪贴板中的内容写入文本框 Text2，则在程序中加入如下语句：

```
temp = Clipboard.GetText(vbCFText)
Text2.Text = temp
```

这样就完成了一个简单的文本复制、粘贴过程，利用剪贴板还能进行各种数据格式的操作，具体的方法我们可以参照 VB 帮助。

2.5 趣味程序源码注解(简单的聊天程序)

这是一个简单的聊天程序(VB6.0 实现)，它可以实现在局域网中两台主机间的在线聊天，程序很简单，只有短短的几十行，但“麻雀虽小，五脏俱全”，它已经有了聊天程序的大体框架。我们在它的基础上稍加改进，就能做出不错的聊天小软件呢。

为了学习方便，提供的源码已经作了详细的中文注释，看看源码框中的代码：

```
Option Explicit
Private IgnoreText As Boolean
```

各控件说明

名称	类型	作用
'frmMain	Form	CHAT主窗体
'Winsock1	Winsock	连接控件
'Label1	Label	CONNECT WITH IP标签
'Label2	Label	LOCAL PORT标签
'Label3	Label	REMOTE PORT标签
'txtRemoteIP	TextBox	远程 IP 地址输入框
'txtLocalPort	TextBox	本地 PORT 输入框
'txtRemotePort	TextBox	远程 PORT 输入框

```
'cmdConnect          CommandButton     连接 CONNECT 按钮
'Label4              Label             Type your text and hit Enter to send it标签
'Frame1(remoteip)    Frame             REMOTE IP 框架
'Frame2(host ip)     Frame             HOST IP 框架
'Text1               TextBox           显示对方（远程主机）发送的 CHAT 内容
'Text2               TextBox           输入己方（本地主机）要发送的 CHAT 内容，按 ENTER
                                       键发送
'cmdClear            CommandButton     清空输入框（TEXT2）和显示框（TEXT1）中的内容
'StatusBar1          StatusBar         状态栏
'当 CLEAR 按钮按下时，清空 TEXT1 和 TEXT2 中的内容
Private Sub cmdClear_Click()
Text1 = ""
With Text2
    清空输入框
    .Text = " "
    并把焦点置于 TEXT2
    .SetFocus
End With
End Sub
'当 CONNECT 按钮按下时，进行以下操作
Private Sub cmdConnect_Click()
On Error GoTo ErrHandler
With Winsock1
    设置 RemoteHost 属性
    .RemoteHost = Trim(txtRemoteIP)
    设置 RemotePort 属性
    'RemotePort 属性的值应该等于 远程主机上的 LocalHost 属性的值
    .RemotePort = Trim(txtRemotePort)
    'LocalPort 属性的值是不能改变的,必须检查它是否已经被设置
    如果 LocalPort 属性为空（没有被设置），将其设为在 LocalPort 输入框中输入的数值
    If .LocalPort = Empty Then
        .LocalPort = Trim(txtLocalPort)
        Frame2.Caption = .LocalIP
        .Bind .LocalPort
        待查
    End If
End With
```

为了保证使用者不能改变 LocalPort 的值，将 txtLocalPort 输入框锁定

```
    txtLocalPort.Locked = True
    '在状态栏中显示“正在连接”的状态
    StatusBar1.Panels(1).Text = "    Connected to " & Winsock1.RemoteHost & "    "
    '如果连接正常，做以下设置
    Frame1.Enabled = True
    Frame2.Enabled = True
    Label4.Visible = True
    Text2.SetFocus
    Exit Sub
    '如果在连接过程中出现错误，则转向 ErrHandler:，并显示错误提示
    ErrHandler:
    MsgBox "Winsock failed to establish connection with remote server", vbCritical
    End Sub
    '当按下“F1”键时显示帮助信息
    Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = vbKeyF1 Then
    ChDir App.Path
    '调用外部程序 notepad.exe 来打开帮助文本文件
    Shell "notepad.exe readme.txt", vbNormalFocus
    End If
    End Sub
    '当窗体加载时显示提示信息并在 txtRemoteIP 框中显示本地主机的 IP
    Private Sub Form_Load()
    Show
    MsgBox "Winsock UDT Chat" & vbCrLf & "by Theo Kandiliotis (ionikh@hol.gr)" & vbCrLf
& vbCrLf & "F1 for help.", vbInformation
    txtRemoteIP = Winsock1.LocalIP
    End Sub
    '接收 TEXT2 输入框的按键，并做响应
    Private Sub Text2_KeyPress(KeyAscii As Integer)
    '定义变量 Last_Line_Feed 来记录最后输入行的位置
    Static Last_Line_Feed As Long
    '定义 New_Line 字符串记录新键入的一行文本的内容
    Dim New_Line As String
    '如果使用者按下 CLEAR 按钮对输入框内容清空，这时 TEXT2 为空，则重设最后输入
行的位置为 0
    If Trim(Text2) = vbNullString Then Last_Line_Feed = 0
    '当使用者按下 ENTER 键时
    If KeyAscii = 13 Then
```

```
        '取得最后输入行的内容并赋值给 New_Line 字符串
        New_Line = Mid(Text2, Last_Line_Feed + 1)
        '重设最后输入行的位置
        Last_Line_Feed = Text2.SelStart
        '通过 WINSOCK 发送新输入的一行文本的内容
        Winsock1.SendData New_Line
        '在状态栏显示发送信息
        StatusBar1.Panels(2).Text = "    Sent " & (LenB(New_Line) / 2) & " bytes    "
    End If
    End Sub
    '当 WINSOCK 接收到新的数据（信息）时，进行以下响应
    Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
    '定义 New_Text 字符串来记录新接收的信息
    Dim New_Text As String
    '接收信息并赋值给 New_Text
    Winsock1.GetData New_Text
    '在 TEXT1 显示框中显示新接收到的信息
    Text1.SelText = New_Text
    Frame1.Caption = Winsock1.RemoteHostIP
    '在状态栏中显示接收信息
    StatusBar1.Panels(2).Text = "    Recieved " & bytesTotal & " bytes    "
    End Sub
```

'这就是一个最简单的 CHAT 程序，你可以在它的基础上加以改进，做出更实用的 CHAT 小软件。

2.5.1　主打控件 WINSOCK 解析：

作为这个 CHAT 程序的主打控件 WINSOCK  ，我们来看看它的一些常用属性、事件和方法。利用 WinSock 控件可以与远程计算机建立连接，并通过用户数据文报协议 (UDP) 或者传输控制协议 (TCP)进行数据交换。这两种协议都可以用来创建客户与服务器应用程序。与 Timer 控件类似，WinSock 控件在运行时是不可见的。

它的主要用途有：

1、创建收集用户信息的客户端应用程序，并将收集的信息发送到某中央服务器。

2、创建一个服务器应用程序，作为多个用户的数据的汇入点。

3、创建“聊天”应用程序。

WINSOCK 的主要属性有：

●PROTOCOL（控制协议）属性，在这个例子中，我们将 PROTOCOL 属性设置为 sckUDPProtocol。这个协议一般用于简单数据交换的情况，而如果我们要编写 INTERNET 应

用程序，多采用 TCP 协议。（如图 2-1)。

●属性 RemoteHost 是远程主机的地址，LocalPort、RemotePort 分别本地主机的端口和远程主机端口，对客户来说，该属性指定发送数据的本地端口，而对于服务器来说，这是用于侦听的本地端口，我们在设置这两个属性时必须保证两台主机的端口值符合下面规则，即主机 1 的 LocalPort 等于主机 2 的 RemotePort 值，同样主机 1 的 RemotePort 等于主机 2 的 LocalPort 值，这样才能保证两机通讯的正常进行。

●LocalIP 属性用于返回本地机器的 IP 地址，格式是 IP 地址加点的字符串(xxx.xxx.xxx.xxx)。

在本例中我们用到了以下几个 WINSOCK 的方法：

●在创建 UDP 应用程序时调用了 Bind 方法，这是必须的。Bind 方法的作用是为控件“保留”一个本地端口。例如，如果将控件绑定到 1001 号端口，那么其它应用程序将不能使用该端口进行“监听”。该方法阻止其它应用程序使用同样的端口。

●SendData 方法用于发送一条数据给另一台主机，使用这个方法的语法是：WinSock.SendData [要发送的数据]

●GetData 方法。当 DataArrival 事件出现时，代码调用 GetData 方法获取数据，并将数据存储在字符串变量中。使用语法是：WinSock.GetData [接收数据的变量]

●DataArrival 事件:在本例中我们使用了 DataArrival 事件，DataArrival 事件在当新数据到达时出现使用的语法为：object_DataArrival (bytesTotal As Long) DataArrival 事件的语法包含下面部分：object 对象表达式，其值是“应用于”列表中的对象。 bytesTotal Long 型 ，可获取的数据总数量。

需要说明的是 ：如果没有获取一个 GetData 调用中的全部数据，则事件不会出现。只有存在新数据时才激活事件。可随时用 BytesReceived 属性检查可用的数据量。

这就是一个最简单的 CHAT 程序，你可以在它的基础上加以改进，做出更实用的 CHAT 小软件。

2.6 趣味程序源码注解(可以换肤的窗体)

大家都知道大名鼎鼎的 WINAMP 播放器是支持“换肤”的，也就是说，我们可以动态的改变窗体和控件的背景图案，使窗体看起来十分漂亮，而且能够不断更换新的面孔。

图 2-1

那我们能让自己的程序也实现这样的效果吗？（如图 2-1），答案是肯定的，下面我们就一起来看看用 VB6 做出的可以换肤的窗体。

程序包含一个主窗体和一个模块，为了使窗体上的控件也能以我们选择的图案作为背景，我们使用了 Microsoft Forms2.0 Object Library 的控件组，在默认状态下它不在控件面板中，我们可以在控件面板上点鼠标右键“添加”，在“添加选择对话框”中找到这个控件并选中添加进来。这时会多出几个控件，如下图所示红线圈住的部分：

这个程序中我们使用了其中的三个控件，乍一看它们和默认的标签、文本框、按钮没有什么区别，但它们的属性中多出了 BackStyle 属性，(如图 2-2)，我们正是利用了这个属性，将控件设为背景透明 0-fmBackStyleTransparent，这样控件的背景就能和窗体的背景保持一致了。

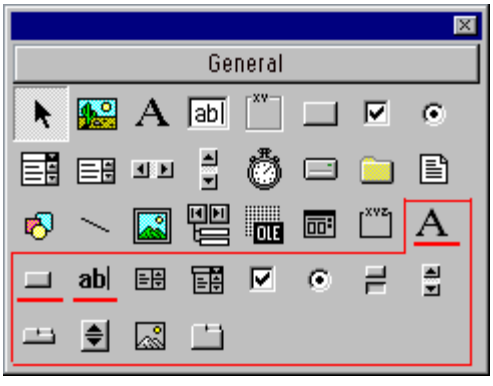

图 2-2

为了学习方便，提供的源码已经作了详细的中文注释，看看源码框中的代码：

2.7　趣味程序源码注解(使用右键菜单)

右键菜单能让软件的使用者快捷的完成操作，那么如何把右键菜单用在自己写的程序中呢？在这个小程序中我们能看到编辑并使用右键菜单的方法。

为了学习方便，提供的源码已经作了详细的中文注释，看看源码框中的代码：

名称　作用

CmdCancel　　退出按钮

mnublue　　“兰色”菜单项

mnured　　“红色”菜单项

RichTextBox1　　文本框

PopupFrm　　主窗体

mnufile　　右键菜单的名字

```
Private Sub CmdCancel_Click()
Unload Me

End Sub
```

```
'当弹出式菜单的“红色”项被点击时
Private Sub mnured_Click()
把 RichTextBox 框中的背景色设置为红色
RichTextBox1.BackColor = vbRed

End Sub

'当弹出式菜单的“兰色”项被点击时
Private Sub mnublue_Click()
把 RichTextBox 框中的背景色设置为兰色
RichTextBox1.BackColor = vbBlue

End Sub

'当文本框上出现鼠标按下的事件时
Private Sub RichTextBox1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
'MouseDown事件各种语法包含下列部分:
'button  返回一个整数，用来标识该事件的产生是按下哪个按钮
'其中左按钮（位 0），右按钮（位 2），以及中间按钮（位 4）
'shift  返回一个整数，标示是否同时有 Shift,Ctrl,Alt 键按下
'x, y  返回一个指定鼠标指针当前位置的数
'Button = 2 表示右键按下
If Button = 2 Then
'PopupMenu方法用来弹出一个菜单
语法是 object.PopupMenu menuname, flags, X, Y
'mnufile是我们在菜单编辑器中设计好的菜单
'X,Y是弹出菜单的位置，可以为数字，如果直接写为 X,Y 则是在当前鼠标位置弹出菜单
PopupFrm.PopupMenu mnufile, 0, X, Y
End If

End Sub
```

我们先打开菜单编辑器(在工具菜单中)，然后添加一个一级菜单 Popmenu，**将它的** Visible **属性设为**“False”，**这代表菜单在程序运行时是看不到的，由于一级菜单是二级菜单的上级菜单，所以二级菜单也是看不到的。然后我们来添加几个二级菜单，注意每个菜单都有** Caption **属性和** Name **属性，这两个属性是必须写的，**Caption **属性是显示在菜单项上的内容，而** Name **属性则是我们要在程序中引用菜单项的代号，类似与其它控件的** Name **属性。而菜单中分隔线是通过把** Caption **属性设为**“-”**来实现的。(如图** 2-3)。

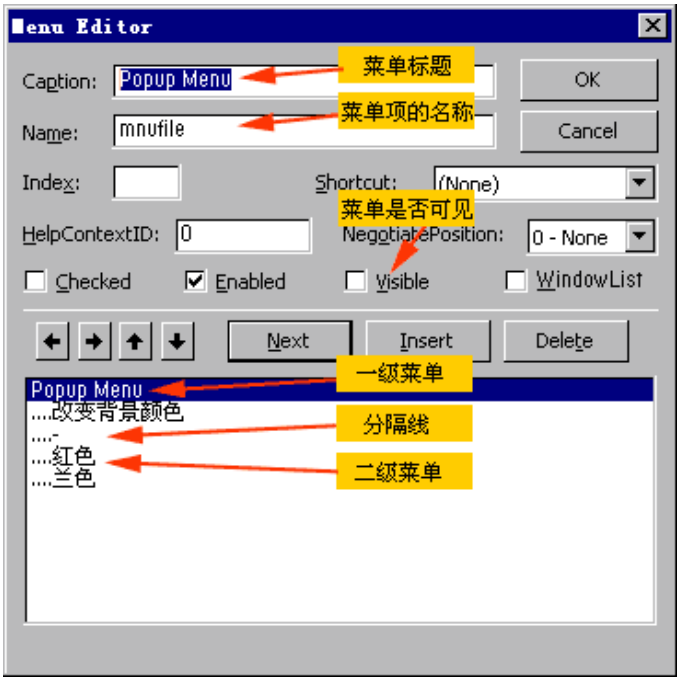

图 2-3

菜单编辑好后 ，就能在程序中引用并控制它，在本例中我们希望在文本框上点击鼠标右键时弹出这个右键菜单，则需要在文本框的 MouseDown **事件中加入控制语句：**

```
Private Sub RichTextBox1_MouseDown(Button As Integer,Shift As Integer,X As Single,Y As Single)
If Button = 2 Then 如果是鼠标右键按下时
Form1.PopupMenu mnufile, 0, X, Y 弹出右键菜单
End If
End Sub
```

其中 MouseDown **事件各个参数意义如下：**

button **返回一个整数，用来标识该事件的产生是按下哪个按键，其中左键为** 0，**右键为** 2，**中间键为** 4
shift **返回一个整数，标示是否同时有** Shift,Ctrl,Alt **键按下**
x, y **返回一个指定鼠标指针当前位置的数**

Popupmenu **是一个方法：**

PopupMenu **方法用来弹出一个菜单，语法是：**object.PopupMenu menuname, flags, X, Y
mnufile **是我们在菜单编辑器中设计好的菜单的名称**
X,Y **是弹出菜单的位置，可以为数字，如果直接写为** X,Y **则是在当前鼠标位置弹出菜单**

这样就能调出右键菜单，然后我们根据自己的需要，为每个菜单项的 Click **事件编写代码，完成一些特定的操作，右键菜单就可以使用了。最后效果如图** 2-4 **所示：**

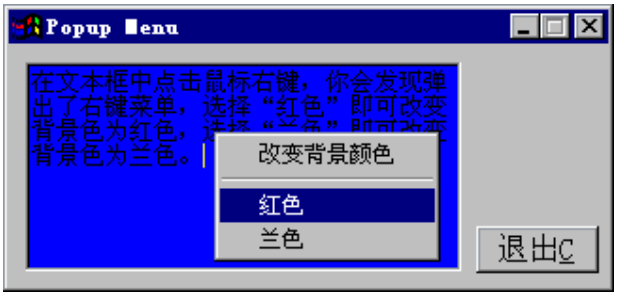

图 2-4

2.8 趣味程序源码注解（拖动无边框窗体）

一般情况下我们拖动一个窗体，必须把鼠标点在标题栏上才能拖动。但有时为了实现风格独特的窗体，我们会使用没有标题栏的窗体，这时怎么来拖动窗体呢？本例会给你答案。

为了学习方便，提供的源码已经作了详细的中文注释，看看源码框中的代码：

```
程序说明：
本例演示了怎样移动无标题栏的窗体的方法，程序中涉
及到 MouseDown、MouseMove、MouseUp 三个鼠标事件的
使用，我们可以从中学习到它们的用法。
Option Explicit
变量声明
'MoveScreen，布尔型变量，标示窗体是否处于被移动状态
Dim MoveScreen As Boolean

'鼠标位置
Dim MousX As Integer
Dim MousY As Integer
'窗体位置
Dim CurrX As Integer
Dim CurrY As Integer

'“退出”按钮
Private Sub CmdExit_Click()
    End
End Sub

'当鼠标在窗体上按下时
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
参数说明：
```

```
'button  返回一个整数，用来标识按下或释放的是哪一
个按钮。button 参数的值为相应于左按钮（1）右按钮
'（2），以及中间按钮（4）。
'shift   返回一个整数，在鼠标按钮被按下或者被释放
的同时，SHIFT,CTRL,和 ALT 键的状态，返回的 shift
参数值分别为 1，2，和 4。指示这些键的状态。
'x, y    返回一个指定鼠标指针当前位置的数。
    如果是鼠标左键按下
    If Button = 1 Then
    标示为移动状态
    MoveScreen = True
    得到鼠标在窗体上的位置（相对与窗体内部坐标）
    MousX = X
    MousY = Y
    End If
End Sub

'当鼠标在窗体上移过时
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

    如果处于鼠标左键按下的状态，即 MoveScreen = True 时
    If MoveScreen Then
        计算新的窗体坐标值
        仔细想一下，看看是不是这样
        CurrX = Form1.Left - MousX + X
        CurrY = Form1.Top - MousY + Y
        移动窗体到新的位置
        Form1.Move CurrX, CurrY
    End If

    把新的窗体坐标显示出来，是相对于屏幕的坐标
    Label3.Caption = CurrX
    Label4.Caption = CurrY
    把鼠标点击的位置显示出来，是相对与窗体的坐标
    Label7.Caption = MousX
    Label8.Caption = MousY
End Sub

如果鼠标松开，则停止拖动
```

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    MoveScreen = False
End Sub
```

程序实现的思路是在窗体的 MouseDown **事件(当鼠标在窗体上按下时发生)中获取鼠标相对于窗体的坐标，在** MouseOver **事件(当鼠标移动时发生)中获取新的鼠标坐标值，显然这两个坐标值的差加上窗体原来的坐标，就是窗体应该移动到的新的坐标值。然后采用** From.Move **方法把窗体位置移动到新坐标值处即可。**

先来看看 MouseDown、MouseMove、MouseUp **事件的使用方法，其语法如下：**

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

其中各个参数的意义如列表所示：

属性	意义
button	**返回一个整数，用来标识按下或释放的是哪一个按钮。**button **参数的值为相应于左按钮（1）右按钮（2），以及中间按钮（4）。**
shift	**返回一个整数，在鼠标按钮被按下或者被释放的同时，**SHIFT,CTRL,**和** ALT **键的状态，返回的** shift **参数值分别为** 1，2，**和** 4。**指示这些键的状态。**
x, y	**返回一个指定鼠标指针当前位置的数**

下面是实现的步骤 ：

1、**在** MouseDown **事件发生时判断按键状态并获取鼠标位置：**

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    如果是鼠标左键按下
    If Button = 1 Then
        标示为移动状态
        MoveScreen = True
        得到鼠标在窗体上的位置（相对与窗体内部坐标）
        MousX = X
        MousY = Y
    End If
End Sub
```

2、**当鼠标** MouseMove **事件发生时，计算新的窗体坐标，并移动：**

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

如果处于鼠标左键按下的状态，即 MoveScreen = True **时**

```
    If MoveScreen Then
        计算新的窗体坐标值
        CurrX = Form1.Left - MousX + X
        CurrY = Form1.Top - MousY + Y
        移动窗体到新的位置
```

```
        Form1.Move CurrX, CurrY
    End If
    End Sub
```

很简单就能移动无标题栏窗体了，这样我们也就解决了“可以换肤的窗体”一节中所提出的问题，那赶快把这节的方法运用到那里去吧。

2.9　趣味程序源码注解(文本编辑器)

今天我们一起用 RichTextBox **控件来做一个文本编辑器，(如图** 2-5),**这个文本编辑器具有比较完备的功能，例如：查找字串、设置字体等。通过这个程序，我们可以看到** RichTextBox **控件丰富的使用方法。**

图 2-5

程序中使用的 RichTextBox **控件在默认状态下不在控件面板中，我们可以在控件面板上点鼠标右键“添加”，在“添加选择对话框”中找到这个控件并添加。这时控件面板上会出现**  **图标，这就是这个小程序的主打控件** RichTextBox。

为了学习的方便，提供的源代码都提供了详细的中文注释，如下所示：

名称	作用
Form1	**主窗体**
CmdOpen	**通用对话框**
RichTextBox1	RichTextBox
mnuNew	**“新建”菜单项**
mnuOpen	**“打开”菜单项**
mnuSave	**“保存”菜单项**
mnuExit	**“退出”菜单项**
mnuFont	**“字体”菜单项**
mnuPrint	**“打印”菜单项**
mnuFind	**“查找”菜单项**
mnuNext	**“查找下一个”菜单项**

变量　　作用

sFind　　**待查找的字符串**

```
Option Explicit
Public sFind As String

Private Sub Form_Resize()

'如果窗体不处于最小化 RichTextBox1 状态，改变 RichTextBox1 大小以适应窗体大小变化
If Form1.WindowState <> 1 Then
RichTextBox1.Width = Form1.Width - 135
If Form1.Height < 1200 Then
        Form1.Height = 1200
End If
RichTextBox1.Height = Form1.Height - 675
End If
End Sub

'当“退出”菜单项被点击时
Private Sub mnuExit_Click()
Unload Me
End
End Sub

'当“查找”菜单项被点击时
Private Sub mnuFind_Click()
'InputBox('弹出的输入框的标题",[默认值], [返回的值])
'语法：InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])
sFind = InputBox("Find what?", , sFind)
'RichTextBox1.Find 是一个方法，根据给定的字符串，在 RichTextBox 控件中搜索文本
RichTextBox1.Find sFind
End Sub

'当“字体”菜单项被点击时
Private Sub mnuFont_Click()
'显示“字体”对话框
'使用指定的方法，CommonDialog 控件能够显示下列对话。
'方法　　所显示的对话框
'ShowOpen　　显示“打开”对话框
'ShowSave　　显示“另存为”对话框
```

```
    'ShowColor          显示“颜色”对话框
    'ShowFont           显示“字体”对话框
    'ShowPrinter        显示“打印”或“打印选项”对话框
    'ShowHelp           调用 Windows 帮助引擎
    CmdOpen.Flags = cdlCFBoth + cdlCFEffects
    CmdOpen.ShowFont
    将 RichTextBox1 的属性根据“字体”对话框的变化作相应设置
    要改变 RichTextBox 控件中的字体特性，可以使用
    'SelFontName、SelFontSize 和 SelFontColor 属性。
    With RichTextBox1
        .SelFontName = CmdOpen.FontName
        .SelFontSize = CmdOpen.FontSize
        .SelBold = CmdOpen.FontBold
        .SelItalic = CmdOpen.FontItalic
        .SelStrikeThru = CmdOpen.FontStrikethru
        .SelUnderline = CmdOpen.FontUnderline
    End With
    End Sub

    '当“新建”菜单项被点击时，设置为空
    Private Sub mnuNew_Click()
    RichTextBox1.Text = ""
    End Sub

    '当“查找下一个”菜单项被点击时
    Private Sub mnuNext_Click()
    'SelStart属性—返回或设置所选择的文本的起始点；如果没有文本被选中，则指出插入点的位置。
    RichTextBox1.SelStart = RichTextBox1.SelStart + RichTextBox1.SelLength + 1
    'object.Find(string, start, end, options)
    'Find 方法的语法包含下面部分:
    部分                描述
    'object         必需的。对象表达式，其值是“应用于”列表中的一个对象。
    'string         必需的。要在控件中查找的字符串表达式。
    'start          可选的。决定从哪儿开始搜索的整数字符索引。控件中的每一个字符都有一个可惟一标识
的整数索引。控件中文本的第一个字符的索引是 0。
    'end            可选的。决定在哪儿结束搜索的整数字符索引。
    'options        可选的。用来指定一个或多个可选功能常数的和。所指定的功能如“设置值”中所述。
    RichTextBox1.Find sFind, , Len(RichTextBox1)
    End Sub
```

```
'当“打开”菜单项被点击时
Private Sub mnuOpen_Click()
'参看上面 CommonDialog 方法
CmdOpen.ShowOpen
'RichTextBox的 LoadFile 方法
RichTextBox1.LoadFile (CmdOpen.FileName)
End Sub

'当“打印”菜单项被点击时
Private Sub mnuPrint_Click()
CmdOpen.Flags = cdlPDReturnDC + cdlPDNoPageNums
If RichTextBox1.SelLength = 0 Then
    CmdOpen.Flags = CmdOpen.Flags + cdlPDAllPages
Else
    CmdOpen.Flags = CmdOpen.Flags + cdlPDSelection
End If
'参看上面 CommonDialog 方法
CmdOpen.ShowPrinter
'将 RichTextBox 控件中格式化文本发送给设备进行打印。
'语法
'object.SelPrint (hDC)
'SelPrint 方法的语法包含下面部分:
'部分        描述
'object      对象表达式，其值是“应用于”列表中的一个对象。
'hdc         设备描述体，是准备用来打印控件内容的设备。
RichTextBox1.SelPrint CmdOpen.hDC
End Sub

'当“保存”菜单项被点击时
Private Sub mnuSave_Click()
CmdOpen.ShowSave
'RichTextBox的 SaveFile 方法，保存文本
RichTextBox1.SaveFile (CmdOpen.FileName)
End Sub
```

下面我们来看看 RichTextBox **控件的常用的属性、事件和方法。**

RichTextBox **控件可用于输入和编辑文本，它同时提供了比常规的** TextBox **控件更高级的格式特性。 通过这些属性，可对该控件中任何部分的文本使用不同的格式，可以将文本变**

为粗体或斜体，改变文本的颜色，创建上标或下标，可以调整段落的左右缩进值，还可以使用悬挂式缩进等。而且 RichTextBox 控件支持大于 64K 的文本，这些都是 TextBox 控件所不可比拟的。

RichTextBox 控件的主要属性有：

●SelFontName、SelFontSize 和 SelFontColor，它们用来设置文字的字体、大小和颜色。使用的语法是：

object.SelColor [= color] 需要注意的是，这个属性改变的是选中文字的属性，而不是所有的文字，这样我们可以在编辑框内出现不同的字体、颜色等。

●另外程序中还用到了 SelLength 属性－返回或设置所选择的字符数。SelStart 属性－返回或设置所选择的文本的起始点；如果没有文本被选中，则指出插入点的位置。

RichTextBox 控件的主要方法有：

●LoadFile 方法和 SaveFile 方法：它们的作用分别是从文件中导入文本和保存文本到指定的文件，这两个方法支持 txt 和 rtf 格式的文本文件，使用的语法是：

object.SaveFile(pathname, filetype)　其中 pathname 是要打开或保存的文件路径，filetype 是要打开或保存的文件类型(0 表示 txt 文件；1 表示 rtf 格式的文件)。

●Find 方法：其语法是：object.Find(string, start, end, options)

其中　string 是要查找的字符串；start, end 是查找开始和结束的位置，如果不指定的话默认是在全部文本中查找；options 是查找时匹配的模式，有三种选择，rtfWholeWord 为整个单词匹配而不是单词片段，rtfMatchCase 为是否忽略字体的差别，rtfNoHighlight 为找到的单词是否高亮显示。

●SelPrint 方法：将 RichTextBox 控件中格式化文本发送给设备进行打印，使用的语法为：object.SelPrint(hdc)，hdc 为准备用来打印控件内容的设备的句柄。

以上的方法和属性只是 RichTextBox 控件丰富用法的一小部分，如果我们很好的利用它，就能做出比较完善的文本编辑器来。

2.10　趣味程序源码注解(用 OLE 实现文件拖放)

用文件拖放可以使我们方便的打开程序所支持的文件，比如我们直接把一首歌曲(mp3)从资源管理器 Explore 中直接拖放到我们自己写的多媒体播放器上，播放器就能得到放下的文件的路径，然后自动的打开它。这也是流行软件所常用的技巧。下面介绍的小程序，采用 VB6 所提供的 OLE 拖放功能，仅用十几行代码就实现了文件拖放。

为了学习方便，提供的源码已经作了详细的中文注释，看看源码框中的代码：

```
程序说明:
本例是实现文件从 EXPLORE 到 VB 应用程序拖放的又一种
方法，与采用 API 函数实现的方法相比，这种方法更简
单易懂。
'OLE拖放是可在 Visual Basic 应用程序中添加的最强
大、最有用的功能之一就是在控件和控件之间、在控件
```

和其它 Windows **应用程序之间拖动文本和图形。有了**
'OLE，就可将这种功能引入到应用程序中。

```
Option Explicit

Private Sub Form_Load()

'经过声明 Picture1 成为接受文件拖放的一个 OLE 容器
Picture1.OLEDropMode = 1
End Sub

Private Sub Picture1_OLEDragDrop(data As DataObject, effect As Long, button As Integer, shift As Integer, x As Single, y As Single)
Dim i As Integer
  '检查放下的东西是不是文件名
  If data.GetFormat(vbCFFiles) = True Then

    Dim sFileName$

    '只读取第一条记录的信息
    sFileName = data.Files(1)

    '如果不是图片文件则转向错误处理
    On Error GoTo invalidPicture

    '依次读取各条记录，并把文件名添加在列表框中
    For i = 1 To data.Files.Count
    List1.AddItem data.Files(i)
    Next i

    '将图片显示在图片框中
    Picture1.Picture = LoadPicture(sFileName)
  End If

  Exit Sub

invalidPicture:

  '显示错误信息
```

```
    DisplayPicture1Message

End Sub
Private Sub DisplayPicture1Message()

    '清除图片框中的图片
    Picture1.Picture = LoadPicture()

    Const Msg As String = "Invalid Picture Format!"

    '在图片框中显示错误信息，这个用法很少见
    Picture1.CurrentX = (Picture1.ScaleWidth \ 2) - (Picture1.TextWidth(Msg) \ 2)
    Picture1.CurrentY = (Picture1.ScaleHeight \ 2) - (Picture1.TextHeight(Msg) \ 2)
    Picture1.Print Msg
End Sub

'当鼠标拖着东西移过图片框时
Private Sub Picture1_OLEDragOver(data As DataObject, effect As Long, button As Integer, shift As Integer, x As Single, y As Single, State As Integer)

    '检查移过图片框的是不是文件（象“回收站”就不是文件）
    If data.GetFormat(vbCFFiles) Then

        '显示可以放下的图标，是带小加号的那种
        effect = vbDropEffectCopy And effect
    Else

        '否则显示不可放下的图标，是圆圈加斜线那种
        effect = vbDropEffectNone
    End If

End Sub

-OLE 拖放简介
'可在 Visual Basic 应用程序中添加的最强大、最有用的功能之一就是在控件和控件之
'间、在控件和其它 Windows 应用程序之间拖动文本和图形。有了 OLE，就可将这种功能
'引入到应用程序中。
'使用 OLE 拖放时，并不是把一个控件拖动到另一个控件并调用代码（象本章前面讨论的
'拖放一样）；而是将数据从一个控件或应用程序移动到另一个控件或应用程序。例如，用
```

'户先选定并拖动 Excel 中的一列单元，然后将它们放到应用程序的 DBGrid 控件上。

'Visual Basic 的几乎所有控件都在某种程度上支持 OLE 拖放。（由 Visual Basic 专业版和企业版提供的）下述的标准控件和 ActiveX 控件自动支持 OLE 拖放，这意味着无论是从控件拖出还是在控件内放入都不需要编写代码:

'Apex　　DBGrid　　Picturebox　　Richtextbox

'Image　　Textbox　　Maskededitbox

为对这些控件启动自动 OLE 拖放，应将 OLEDragMode 和 OLEDropMode 设置为“自动话”。

可用下列 OLE 拖放属性、事件和方法指定已知控件响应拖放的方式。

类别　　项目　　描述

属性　　OLEDragMode　　启动控件的自动拖动或手工拖动（若控件支持手工拖动但不支持自动 OLE 拖动，则它不具有此属性，但支持 OLEDrag 方法和 OLE 拖放事件）。

'　　OLEDropMode　　指定控件如何响应放操作。

事件　　OLEDragDrop　　识别源对象何时被放到控件上。

'　　OLEDragOver　　识别源对象何时被拖动经过控件。

'　　OLEGiveFeedback　　以源对象为基础向用户提供自定义拖动图标反馈。

'　　OLEStartDrag　　在启动拖动时，源支持哪种数据格式和放效果（复制、移动或拒绝数据）。

'　　OLESetData　　在放源对象时提供数据。

'　　OLECompleteDrag　　当把对象放到目标时通知被执行的操作的源。

方法　　OLEDrag　　启动手工拖动。

本例中用到的属性和方法:

'OLEDragMode　　返回或设置是由部件还是由程序员来处理 OLE 拖放操作。

属性:　　0 表示为手工拖放　　1 表示为自动拖放

我们设为 1 是自动拖放模式，这时我们要用 OLEDragDrop，OLEDragOver 事件来控制拖放操作

'OLEDragDrop　------当源部件决定放操作能发生，且源部件被放到目标部件时，此事件发生。

语法：Private Sub object_OLEDragDrop(data As DataObject, effect As Long, button As Integer, shift As Integer, x As Single, y As Single)

'OLEDragDrop 事件语法包含下面部分:

部分　　描述

'object　　对象表达式，其值是“应用于”列表中的一个对象。

'data　　DataObject 对象，包含源提供的格式，另外也可能包含这些格式的数据。若 DataObject 不包含数据，则当控件调用 GetData 方法时提供数据。SetData 和 Clear 方法不能用在这里。

'effect　　源对象设置的长整型数，用来识别执行的动作，这样当部件被移动后允许源采取适当的动作（例如，如果源被从一个部件移到另一个部件，则执行删除数据操作）。可能的取值列于“设置值”中。

'button　　整数，当按下鼠标键时，与鼠标状态相对应。左键为位 0，右键为位 1，中键为位 2。这些位相应的值分别为 1，2 和 4，它代表了鼠标键的状态。可设置三个位中的部分、全部或根本不设置，相应地表明部分、全部按键被按下或没有按键按下。

'shift　　整数，当按下 SHIFT、CTRL 和 ALT 键时，与这些键状态相对应。

SHIFT 键为位 0，CTRL 键为位 1，ALT 键为位 2。这些位相应的值分别为 1，2 和 4，shift 参数代表了这些键的状态。可设置三个位中的部分、全部或根本不设置，相应地表明部分、全部按键被按下或没有按键按下。例如，同时按下 CTRL 和 ALT 键，shift 值为 6。CTRL

'x,y　　确定鼠标指针当前位置的数值。x 和 y 值由对象的 ScaleHeight、ScaleWidth、ScaleLeft 和 ScaleTop 属性设置的坐标系统的格式来表示。

其中 effect 参数的设置如下：

常数	值	描述
'vbDropEffectNone	0	放目标不接受数据。
'VbDropEffectCopy	1	放结果保存于从源到目标的数据拷贝中。初始数据没有被拖放操作改变。
'VbDropEffectMove	2	放结果保存于要从拖放源移到放源的数据中。移动后，拖放源要删除数据。
'OLEDragOver		当一个部件在另一个部件上拖动时发生。

其参数与 OLEDragDrop 的参数类似。

OLE 拖放是可在 Visual Basic 应用程序中添加的最强大、最有用的功能之一，利用它可以在控件和控件之间、在控件和其它 Windows 应用程序之间拖动文本和图形。VB 中的大部分控件都是支持 OLE 拖放，但默认状态下控件的 OLEDropMode 属性是设为 0，也就是不允许拖放的。如果我们要使用 OLE 拖放，必须先把 OLEDropMode 属性设为 1，这样控件才成为接受文件拖放的一个 OLE 容器。

如图 2-6 所示：我们把 PictureBox 的 OLEDropMode 属性设为 1，这样 PictureBox 就能够接受从 WINDOWS 资源管理器 Explore 或是其它程序中拖放的文件或图片。我们也可以在代码中设置这个属性。这就是上面源码框中 Picture1.OLEDropMode = 1 语句的含义。

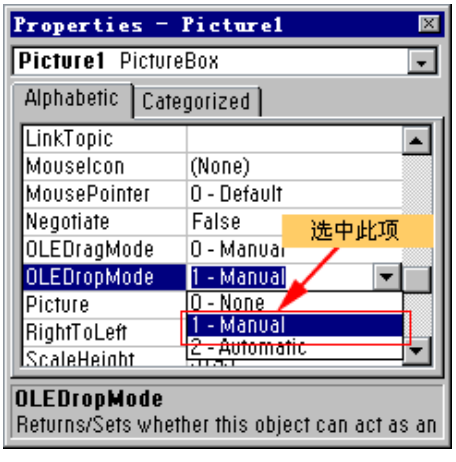

图 2-6

下面我们来看看本例中用到的 OLE 拖放的事件和方法。

●OLEDragDrop 事件：意义是当控件允许放下的动作，并且有东西在控件上放下的时刻发生。当此事件发生时程序会去执行 Sub object_OLEDragDrop()子过程的代码，所以我们必须在其中加入程序来响应此事件。在本例中，程序要判断放下东西的类型是不是图片文件，

共有几个图片文件，然后在 PictureBox 中显示图片，并把文件的路径在列表框中列出来，总之我们可以根据自己的需要写这段代码。

OLEDragDrop 事件的语法为：

```
Sub object_OLEDragDrop(
    data As DataObject,        DataObject 对象，包含放下的数据，具体的格式参见源码注释;
    effect As Long,            用来识别执行的动作 ;
    button As Integer,         整数 ，与鼠标状态相对应。左键为 0，右键 1，中键 2;
    shift As Integer,          整数，当按下 SHIFT、CTRL 和 ALT 键时，与这些键状态相对应;
    x As Single,               x,y 确定鼠标指针当前位置的数值;
    y As Single
```

●OLEDragOver 事件：意义是当鼠标拖着东西在控件上移过时发生，当此事件发生时程序会去执行 Sub object_OLEDragOver()子过程的代码，本例中我们用程序判断鼠标拖动的是不是文件，如果是则显示“允许放下”的鼠标指针，否则显示“禁止放下”的鼠标指针。

用 OLE 实现文件拖放，代码简单易懂，而采用另外一种方法 API 函数实现起来就比较复杂，但是可以实现更加灵活的功能，有兴趣的话你也可以参照一下 API 实例解析中的相关内容。

2.11 趣味程序源码注解(在状态栏中显示帮助信息)

状态栏是 VB 中的一个很有用的控件，（如图 2-7）。但是默认状态下它不在控件面板中，在我们使用它之前必须先添加，方法是按下 Ctrl+T 快捷键，在弹出的“Components”对话框中选中 MS Windows Common Controls。这时在控件面板中会出现状态栏的小图标  ，双击此图标就能把它添加到窗体中。

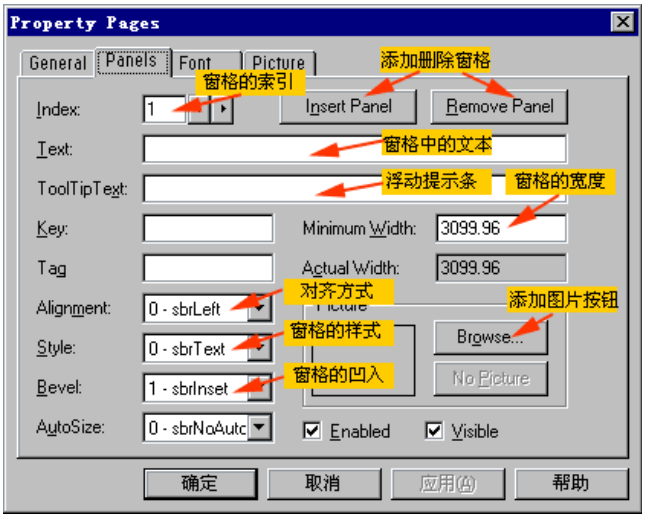

图 2-7

接下来我们根据自己的需要来编辑状态栏，选中窗体上的状态栏，在鼠标右键菜单中选择“属性 Properties”，这时会弹出如上图所示的对话框，我们选择“Panels 窗格”。其中“Insert Panel”按钮用来添加状态栏中的窗格数目，当我们添加一个窗格后“Index”的值也会自动加一，在程序中我们通过窗格的索引值来引用窗格。下面的一些项目都是窗格的属性，我们可以在此设置也可在程序中控制它。比较常用的属性有：

“Text”--在各个窗格中显示的文字；

“Alignment”--窗格中文本的对齐方式；

“Style”窗格的样式(选择 0-sbrText 为显示文本，选择 6-sbrDate 为显示日期等等)；

“Bevel”--窗格的凹凸状态。

另外我们还可以在窗格中加入图片，方法是点击“Browse”然后选中图片文件。

在本程序中我们在第一个窗格中动态显示帮助信息，在第二、三个窗格中分别显示日期和时间。那么怎样在程序中控制状态栏中显示的文本呢？

例如我们希望鼠标移动到不同的按钮上时状态栏的第一个窗格中显示不同的帮助，则我们在按钮的 OnMouseOver 事件中加入如下语句即可：

```
Private Sub CmdNew_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    staSample.Panels(1).Text = ""
    staSample.Panels(1).Text = "清空文本框"
End Sub
```

同样的我们能为其它按钮等控件添加帮助信息。

为了学习方便，提供的源码已经作了详细的中文注释，看看源码框中的代码：

名称	作用
CmdItalic	“斜体”按钮
CmdBold	“粗体”按钮
CmdExit	“退出”按钮
CmdNew	“重写”按钮
Text1	文本框
frmStatusBar	主窗体
staSample	状态栏

```
Option Explicit

'当“斜体”按钮按下时
Private Sub CmdItalic_Click()
'将文本框中的字体设为斜体
Text1.FontItalic = True
End Sub

'当鼠标移动到“斜体”按钮上时
Private Sub CmdItalic_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
'把状态栏的第一个 Panels 中的 Text 属性设为"转化为斜体"的帮助信息
staSample.Panels(1).Text = ""
staSample.Panels(1).Text = "转化为斜体"
End Sub

'当“粗体”按钮按下时
Private Sub CmdBold_Click()
'将文本框中的字体设为粗体
Text1.FontBold = True
End Sub

'当鼠标移动到“粗体”按钮上时
Private Sub CmdBold_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
'把状态栏的第一个 Panels 中的 Text 属性设为"转化为粗体"的帮助信息
staSample.Panels(1).Text = ""
staSample.Panels(1).Text = "转化为粗体"
End Sub

'当“退出”按钮按下时
Private Sub CmdExit_Click()
'退出程序
Unload Me
End Sub

'当鼠标移动到“退出”按钮上时
Private Sub CmdExit_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
'把状态栏的第一个 Panels 中的 Text 属性设为"退出程序"的帮助信息
staSample.Panels(1).Text = ""
staSample.Panels(1).Text = "退出程序"
End Sub

'当“重写”按钮按下时
Private Sub CmdNew_Click()
'将文本框中的文本清空
Text1.Text = ""
Text1.FontBold = False
Text1.FontItalic = False
End Sub
```

```
'当鼠标移动到“重写”按钮上时
Private Sub CmdNew_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
把状态栏的第一个 Panels 中的 Text 属性设为"清空文本框"的帮助信息
staSample.Panels(1).Text = ""
staSample.Panels(1).Text = "清空文本框"
End Sub

Private Sub edit_Click()
frmProperties.Show vbModal
End Sub

'当鼠标移动到文本框上时
Private Sub Text1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
把状态栏的第一个 Panels 中的 Text 属性设为"输入文字"的帮助信息
staSample.Panels(1).Text = ""
staSample.Panels(1).Text = "输入文字"
End Sub
```

2.12　味程序源码注解(自动改变控件大小)

由于我们设计窗体时，安排的各个控件的大小和位置是固定的，如果在程序运行过程中窗体大小发生了变化，就会造成控件大小与窗体大小不成比例。怎样简单快捷的解决这个问题呢？本例给出了一个不错的方法。

为了学习方便 ，提供的源码已经作了详细的中文注释，看看源码框中的代码：

本例实现一个控件大小随窗体大小改变而自动改变的文本编辑器

名称	作用
Form1	主窗体
CmdOpen	通用对话框
RichTextBox1	RichTextBox
mnuNew	“新建”菜单项
mnuOpen	“打开”菜单项
mnuSave	“保存”菜单项
mnuExit	“退出”菜单项
mnuFont	“字体”菜单项
mnuPrint	“打印”菜单项
mnuFind	“查找”菜单项
mnuNext	“查找下一个”菜单项

变量　　作用

sFind　　**待查找的字符串**

Option Explicit

Private FormOldWidth As Long

　　保存窗体的原始宽度

Private FormOldHeight As Long

保存窗体的原始高度

Public sFind As String

在程序装入时必须加入

Private Sub Form_Load()

Call ResizeInit(Me)

End Sub

当“退出”菜单项被点击时

Private Sub mnuExit_Click()

Unload Me

End

End Sub

当“查找”菜单项被点击时

Private Sub mnuFind_Click()

'InputBox('**弹出的输入框的标题**",[**默认值**], [**返回的值**])

语法：InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])

sFind = InputBox("Find what?", , sFind)

'RichTextBox1.Find **是一个方法，根据给定的字符串，在** RichTextBox **控件中搜索文本**

RichTextBox1.Find sFind

End Sub

当“字体”菜单项被点击时

Private Sub mnuFont_Click()

显示“字体”对话框

使用指定的方法，CommonDialog **控件能够显示下列对话**。

方法　　所显示的对话框

'ShowOpen　　**显示“打开”对话框**

'ShowSave　　**显示“另存为”对话框**

'ShowColor　　**显示“颜色”对话框**

'ShowFont　　**显示“字体”对话框**

'ShowPrinter　　**显示“打印”或“打印选项”对话框**

```
'ShowHelp      调用 Windows 帮助引擎
CmdOpen.Flags = cdlCFBoth + cdlCFEffects
CmdOpen.ShowFont
将 RichTextBox1 的属性根据“字体”对话框的变化作相应设置
要改变 RichTextBox 控件中的字体特性，可以使用
'SelFontName、SelFontSize 和 SelFontColor 属性。
With RichTextBox1
.SelFontName = CmdOpen.FontName
.SelFontSize = CmdOpen.FontSize
.SelBold = CmdOpen.FontBold
.SelItalic = CmdOpen.FontItalic
.SelStrikeThru = CmdOpen.FontStrikethru
.SelUnderline = CmdOpen.FontUnderline
End With
End Sub

当“新建”菜单项被点击时，设置为空
Private Sub mnuNew_Click()
RichTextBox1.Text = ""
End Sub

当“查找下一个”菜单项被点击时
Private Sub mnuNext_Click()
'SelStart属性—返回或设置所选择的文本的起始点；如果没有文本被选中，则指出插入点的位置。
RichTextBox1.SelStart = RichTextBox1.SelStart + RichTextBox1.SelLength + 1
'object.Find(string, start, end, options)

'Find 方法的语法包含下面部分:
部分              描述
'object           必需的。对象表达式，其值是“应用于”列表中的一个对象。
'string           必需的。要在控件中查找的字符串表达式。
'start            可选的。决定从哪儿开始搜索的整数字符索引。控件中的每一个字符都有一个可惟一标识的整数索引。控件中文本的第一个字符的索引是 0。
'end              可选的。决定在哪儿结束搜索的整数字符索引。
'options          可选的。用来指定一个或多个可选功能常数的和。所指定的功能如“设置值”中所述。
RichTextBox1.Find sFind, , Len(RichTextBox1)
End Sub

当“打开”菜单项被点击时
```

```
Private Sub mnuOpen_Click()
参看上面 CommonDialog 方法
CmdOpen.ShowOpen
'RichTextBox的 LoadFile 方法
RichTextBox1.LoadFile (CmdOpen.FileName)
End Sub

当“打印”菜单项被点击时
Private Sub mnuPrint_Click()
CmdOpen.Flags = cdlPDReturnDC + cdlPDNoPageNums
If RichTextBox1.SelLength = 0 Then
CmdOpen.Flags = CmdOpen.Flags + cdlPDAllPages
Else
CmdOpen.Flags = CmdOpen.Flags + cdlPDSelection
End If
参看上面 CommonDialog 方法
CmdOpen.ShowPrinter
将 RichTextBox 控件中格式化文本发送给设备进行打印。
语法
'object.SelPrint (hDC)
'SelPrint 方法的语法包含下面部分:
部分          描述
'object       对象表达式，其值是“应用于”列表中的一个对象。
'hdc          设备描述体，是准备用来打印控件内容的设备。
RichTextBox1.SelPrint CmdOpen.hDC
End Sub
当“保存”菜单项被点击时
Private Sub mnuSave_Click()
CmdOpen.ShowSave
'RichTextBox的 SaveFile 方法，保存文本
RichTextBox1.SaveFile (CmdOpen.FileName)
End Sub

确保窗体改变时控件随之改变
Private Sub Form_Resize()
Call ResizeForm(Me)
End Sub

以下为模块中的代码:
```

```
Option Explicit
定义 FormOldWidth, FormOldHeight 为全局变量，这样其他模块才能调用它
Global FormOldWidth, FormOldHeight

在调用 ResizeForm 前先调用本函数
Public Sub ResizeInit(FormName As Form)

 Control 是一个对象，表示所有 Visual Basic 内部控件的类名。
可以将一个变量标为 Control 对象，像把控件放到窗体上的一样来引用它。例如:
'Dim C As Control
'Set C = Command1
Dim Obj As Control
FormOldWidth = FormName.ScaleWidth
FormOldHeight = FormName.ScaleHeight
On Error Resume Next

'Each是一个关键字，作用是针对一个数组或集合中的每个元素，重复执行一组语句。
语法
'For Each element In Group
For Each Obj In FormName
   'Tag返回或设置一个表达式用来存储程序中需要的额外数据。
   Obj.Tag = Obj.Left & " " & Obj.Top & " " & Obj.Width & " " & Obj.Height & " "
Next Obj
On Error GoTo 0

End Sub

按比例改变表单内各元件的大小，
在调用 ReSizeForm 前先调用 ReSizeInit 函数
Public Sub ResizeForm(FormName As Form)
Dim Pos(4) As Double
 Dim i As Long, TempPos As Long, StartPos As Long
Dim Obj As Control
Dim ScaleX As Double, ScaleY As Double

保存窗体宽度缩放比例
ScaleX = FormName.ScaleWidth / FormOldWidth
 保存窗体高度缩放比例
ScaleY = FormName.ScaleHeight / FormOldHeight
```

```
    On Error Resume Next

     For Each Obj In FormName
       StartPos = 1

       读取控件的原始位置与大小
      For i = 0 To 4
       'InStr 函数，返回 Variant (Long)，指定一字符串在另一字符串中最先出现的位置。语法:InStr([start, ]string1, string2[, compare])
       TempPos = InStr(StartPos, Obj.Tag, " ", vbTextCompare)
       If TempPos > 0 Then
       'Mid函数，返回 Variant (String)，其中包含字符串中指定数量的字符。语法：Mid(string, start[, length])
           Pos(i) = Mid(Obj.Tag, StartPos, TempPos - StartPos)
       StartPos = TempPos + 1
     Else
       Pos(i) = 0
     End If

     根据控件的原始位置及窗体改变大小的比例对控件重新定位与改变大小
     'Move方法，用以移动 MDIForm、Form 或控件。语法:object.Move Left, Top, Width, Height
     Obj.Move Pos(0) * ScaleX, Pos(1) * ScaleY, Pos(2) * ScaleX, Pos(3) * ScaleY

   Next i
 Next Obj
 On Error GoTo 0

End Sub
```

先来看看与控件和对象的大小、位置相关的几个属性，已经列表如下：

属性	意义
ScaleWidth、ScaleHeight	用 ScaleLeft、ScaleTop、ScaleHeight 和 ScaleWidth 来完成基于对象内部尺寸的操作，如绘出或移动包含在该对象中的对象。
ScaleLeft、ScaleTop	
Left、Top	用 Left、Top、Height 和 Width 属性来完成基于对象外部维数的操作，如移动或改变尺寸。对于窗体内放置的控件，这几个属性代表控件相对于窗体的位置；而对于窗体应用这几个属性则表示窗体相对于屏幕的位置。
Width、Height	

程序实现的基本思路是在窗体大小改变前取得窗体的 ScaleWidth、ScaleHeight **以及窗体内各个控件的** Left、Top、Width、Height **属性，并把它们保存起来。当窗体的大小被改变时，**

用窗体的新 ScaleWidth、ScaleHeight **与改变前的值相比，得到一个缩放比例，然后把这个比例与控件的** Left、Top、Width、Height **属性的乘积作为新的控件位置和大小，重画这些控件。下面是实现的步骤：**

1、**得到原有窗体** ScaleWidth、ScaleHeight **属性：**

```
FormOldWidth = FormName.ScaleWidth
FormOldHeight = FormName.ScaleHeight
```

2、**取得各个控件的** Left、Top、Width、Height **属性并把它们存放在控件的** Tag **属性中：**

```
Dim Obj As Control    'Control是一个对象，表示所有 Visual Basic 内部控件的类名
For Each Obj In FormName
'Tag返回或设置一个表达式用来存储程序中需要的额外数据。
Obj.Tag = Obj.Left & " " & Obj.Top & " " & Obj.Width & " " & Obj.Height & " "
Next Obj
```

3、**当窗体大小改变时计算缩放比例：**

```
'保存窗体宽度缩放比例
ScaleX = FormName.ScaleWidth / FormOldWidth
'保存窗体高度缩放比例
ScaleY = FormName.ScaleHeight / FormOldHeight
```

4、**读取控件的原始位置与大小，并根据缩放比例重新安置控件：**

```
For Each Obj In FormName StartPos = 1
  For i = 0 To 4
    TempPos = InStr(StartPos, Obj.Tag, " ", vbTextCompare)
    If TempPos > 0 Then
        Pos(i) = Mid(Obj.Tag, StartPos, TempPos - StartPos)
        StartPos = TempPos + 1
    Else Pos(i) = 0
    End If
    Obj.Move Pos(0) * ScaleX, Pos(1) * ScaleY, Pos(2) * ScaleX, Pos(3) * ScaleY
  Next i
Next Obj
```

好了这样我们就不必担心窗体内的控件大小位置不成比例了，由于程序可以自动改变所有控件的大小，只要是窗体内的控件，大小都随之而变化，可谓“一劳永逸”的好办法。

第三章　VB 编程技巧大派送

3.1　列表框使用技巧

1、排列列表框中的列表项

在设计或运行时，都可以通过将列表框的 SORTED 属性设置为 TRUE（默认为 FALSE）来对列表框的各项按字母顺序进行排列。

2、怎样返回被选中的列表项的文本

可以使用 list.text 属性 或 list.list(list.listindex)来得到列表项。

3、使用多选列表框

多选列表框允许用户一次选择多个列表项 。通过对 MULTISELECT 属性的设置，就可以把一个列表框变成多选列表框。用户可以用 SHIFT 和 CTRL 键选择多个列表框。下面是设置它时可能用到的值：

0--不允许进行多选（默认）

1--简单的多选，单击鼠标或空格键可在列表框中选中一项或取消选择

2--扩展的多选。按下 SHIFT 键并单击鼠标或按下 SHIFT 键和一个箭头键

也许你会说，我已经把上面的 MULTISELECT 属性设置好了，而且也能够在列表框中实现多选，但是怎么才能返回选中的条目呢？是这样的，我们可以使用一个循环来找到被选中的项： 循环中利用了 SELECTED 属性：

```
Dim intloopindex as integer                 'intloopindex为循环变量
for intloopindex =0 to list.listcount-1     'list.listcount-1是列表框中最大的列表项序号
    if list.selected(intloopindex) then     'selected属性为列表项的选中状态，为布尔型
      list2.additem list.list(intloopindex) 将选中的列表项添加到另外一个列表框中
    end if                                  我们可根据自己需要写这段代码
next intloopindex
```

4、使列表框具有水平滚动条：只须用 COLUMNS 属性将列表框划分为多列即可，默认情况下属性的值为 0，不允许多列；设置为其他值时，列表框就会将他的列表项显示为多列，而我们对每一项的操作方法不变。

5、在列表框中使用复选框：可通过设置 STYLE 属性，将一个列表框变成使用复选框的列表框，下面是设置 STYLE 属性时所用的值：0--标准列表框（默认）1--带有复选框的列表框

6、清空列表框 LIST.CLEAR

其实上面的这些技巧对于和列表框类似的控件例如 ：文件列表框(FileListBox)、文件夹列表框(DirListBox)等也是适用的，赶快拿去试一试吧。

3.2　工具栏使用技巧

1、在工具栏中添加复选（切换）按钮

工具栏中的复选按钮是指当按钮被按下以后就保持被按下的状态 ，只有下次再按才会弹起来，这就是工具栏的复选切换状态。要实现这样的效果，必须将它的 STYLE 属性设为 tbrCheck，这个设置可以在工具栏的属性页中完成。方法是右击工具栏并选择 Properties 选项以打开属性页，单击属性页的 Buttons 选项卡，选择要用的按钮，将它的形式 STYLE 设为 tbrCheck 即可。

2 、在工具栏中添加组合框和其他控件

通过将按钮的 STYLE 属性设为 tbrPlaceholder 在工具栏中设置空间 ，可以将组合框和其他控件添加到工具栏中，我们以组合框为例来看看具体步骤：

1 ）右击工具栏并选择 Properties 选项打开属性页，单击属性页的 Buttons 选项卡；

2 ）在要添加组合框的地方添加一个新的按钮；

3 ）把新按钮的 STYLE 属性设为 tbrPlaceholder，这样按钮不会显现出来，而只是一片空白，用来设置组合框；

4 ）在 width 框中输入一个值，这是预留给组合框的空间的宽度；

5 ）点击确定后，在工具栏的空白处新画一个组合框，注意一定要新画；

6 ）其他操作与原来相同。

3 、怎样做出象 IE 一样的平面工具栏

IE 、WORD 等流行软件的工具栏在通常状态下是平面的，只有当鼠标移过时才会突起，这样的效果通过 VB 工具栏本身是无法实现的，虽然可以用贴图的方法来模拟这种效果但却十分麻烦，简便的方法是通过调用 WIN32 API 函数来实现 。其思路是用 SendMessage 函数向工具栏发送设置显示样式 TB_SETSTYLE 的消息来改变工具栏的显示效果。具体的实现请参看“API 实例解析”之“实现平面工具栏”。

3.3　文本框使用技巧

1、控制文本框中输入的内容

例如我们只要求在文本框中输入数字 ，而不允许出现其它字符，则我们使用 KEYPRESS 事件并检查 KeyAscii 参数即可，KeyAscii 参数用来保存键盘所用的 ANSI 码（不是 ASCII 码），下面是一个例子：

```
Private Sub Text_KeyPress(KeyAscii As Integer)
if KeyAscii < Asc("0") Or KeyAscii > Asc("9") then
    KeyAscii=0
end if
end sub
```

2 、从外部文本文件读入和从文本框中写出内容到文件内容：

有时我们需要从外部的文本文件中读入到文本框中，或是把输入到文本框中的内容保存到文件中，这时可以通过对文件操作的几个语句来实现，其中 Open 语句能够对文件输入/输出，LOF 函数返回一个 Long 型值，表示用 Open 语句打开的文件的大小，该大小以字节为单位。Input 函数从文件中读入数据，而 Print 函数则把数据写到文件中。具体的代码如下面的实例，需要注意的是文本框只能打开大小在 32K 以下的文本文件，否则会出现错误。

```
filename="c:\myext.txt"
Open filename For Input As #1
Text.Text=Input$(Lof(1),#1)
Close #1
filename="c:\myext.txt"
on error resume next
open "c:\file.txt" for output as #1
print #1,text1.text
close #1
```

3 、怎样把文本框中的文本全部选中

这需要使用文本框的属性 SelStart 和 SelLength 属性，SelStart 表示选择文本的开始，SelLength 表示要选中文本的长度，采用以下两句代码就行了。

```
Text1.SelStart = 0
Text1.SelLength = Len(Text1.Text)
```

另外顺便提一下 SelText 属性，它可以得到已经选中的文本。

4 、如何在已经存在的文本的 textbox 添加新的一行

```
Dim strNewText As String
Text1.strNewText = "Updated: " & Date
Text1.SelStart = Len(Text1.Text)
Text1.SelText = vbNewLine & strNewText
```

实现的关键是 vbNewLine 常数，它的意义是插入一个换行符。

3.4 组合框使用技巧

1、组合框的三种形式

组合框是用来显示一个文本框及一个下拉列表的控件，你可能认为只有一种组合框，但其实有三种，可以通过设置组合框的 STYLE 属性来选择其一，下面是设置组合框的 STYLE 属性时用的值：

VBCOMBODROPDOWN --0 ：包括一个下拉列表和一个文本框，可以选择也可输入文字；

VBCOMBOSIMPLE --1 ： 简单组合框，包括一个文本框和一个不会下拉的列表；

VBCOMBODROPDOWNLIST --2 ： 只选组合框。

2 、在组合框中添加图像

这需要使用 IMAGE 组合框 ImageCombo 和 ImageList ，我们先在“添加组件”中选中“Ms CommonControl 6.0 ”然后添加 ImageCombo 和 ImageList，并在 ImageList 中添加图片，然后可以用以下语句为下拉框添加图片。

```
ImageCombo1.ComboItems.Add 1,"key1","item1",1
ImageCombo1.ComboItems.Add 2,"key2","item2",2
```

3.5　文件操作的技巧

1、建立和删除目录

你可以用 MKDIR 语句建立一个新目录 ，可以用 RMDIR 语句删除一个目录。下面是一个例子:

```
mkdir "c:\data"
rmdir "c:\data"
```

2 、改变当前目录:

可用 CHDIR 改变当前目录为指定目录 ，例如:

```
CHDIR "C:\WINDOWS"
```

如果想改变当前目录为应用程序所在的目录 ，可以使用 APP.PATH 实现，例如:

```
chdir app.path
```

3 、拷贝、移动文件

可以用 filesystemobject 拷贝文件 ，该对象提供计算机文件系统的访问权限和类似 copyfile 的拷贝文件的方法。该方法使用的语法为:

```
filesystemobject.copyfile source,destination[,overwrite]
```

source 是源文件 （包含路径）

destination 是目标文件 ，亦包含路径

overwrite 是布尔值 ，如果是真则意味着要覆盖已存在的目标文件

使用这个对象之前必须先建立它 ，方法如下:

```
Dim FileSystemObject As Object
Set FileSystemObject=CreatObject("Scripting.FileSystemObject")
FileSystemObject.CopyFile "c:\file1.txt", "c:\file2.txt"
```

CopyFile 是一个方法 ，其作用是将源文件复制到目标位置，它的两个参数分别是源文件和目标文件的路径，另外可以使用的方法有 MoveFile 和 DeleteFile，分别用来移动和删除一个文件，它们的使用方法如下:

```
FilesysTemObject.MoveFile "c:\file1","d:\file1"
FileSystemObject.DeleteFile "c:\text.txt"
```

3.6　VB 中如何实现文本查找功能-Instr 函数使用技巧

VB 中如何实现文本查找功能

实现查找功能的关键在于使用 InStr 函数，这个函数可以找到指定的字符串在另一字符串中最先出现的位置。我们先来看一看使用这个函数的语法：

InStr([start,]string1, string2[, compare])

这个函数需要的的参数是起始位置 、主体字符串、要查找的字符串；Compare 是可选参数。指定字符串比较。此 compare 参数是可以省略的，也可以是 0, 1 或 2。指定 0（缺省）做二进制比较。指定 1 做不区分大小写的文本比较。例如我们要查找在字符串“abcdefg”中是否存在“cd”并返回其位置，则使用下面的语句就可以实现：

pos=InStr(1,"abcdefg","cd")

则 pos 会返回 3 表示查找到并且位置为第三个字符开始。这就是“查找”的实现，而“查找下一个”功能的实现就是把当前位置作为起始位置继续查找。

下面举例说明 ：

放置一个文本框 TEXT1 供用户输入文本或调入文本文件，用来做在其中查找文本的验证，放置另一个文本框 TEXT2 供用户输入要查找的字符串，放置两个命令按钮，Command1、Command2，其标题分别为“查找”、“查找下一个”。

在窗体的总体声明部分写如下代码 ：

```
Option Explicit '定义目标位置变量
Private TargetPosition As Integer

 '编写一个查找函数
Private Sub FindText(ByVal start_at As Integer)
Dim pos As Integer
Dim target As String
'获取用户输入的要查找的字符串
target = text2.Text
pos = InStr(start_at, text1.Text, target)
If pos > 0 Then
        '找到了匹配字符串
        TargetPosition = pos
        text1.SelStart = TargetPosition - 1
        '选中找到的字符串
        text1.SelLength = Len(target)
        text1.SetFocus
Else '没有找到匹配的字符串
MsgBox "没找到！"
text1.SetFocus
```

```
End If
End Sub
```

双击“查找”命令按钮:

```
Private Sub command1_Click() '从第一个字符处开始查找
FindText 1
End Sub
```

双击“查找下一个”按钮:

```
Private Sub command2_Click() '从当前位置继续查找
FindText TargetPosition + 1
End Sub
```

运行程序 ，在文本框 1 **中输入一些字符串，在文本框** 2 **中输入要查找的字符串，单击 “查找”按钮和“查找下一个”按钮进行验证**。

3.7　树视 TreeView 树视的使用技巧

1、**为树状浏览器控件添加节点和子节点**

用 ADD **方法添加一个新节点到树状浏览器的** NODES **集合时，可以声明它是和已存在的节点所联系起来的。通常使用** ADD **方法，其语法如下**:

```
Nodes.Add(relative,[relationship][,key][,text][,image][,selectedimage])
```

各个参数的意义如下 ：

relationship **参数是通过关系节点参数与新节点连接的另一个节点**;

relationship **参数可能是以下情况**:

tvwlast--1；**该节点置于所有其他的在** relative **中被命名的同一级别的节点的后面**

tvwNext--2；**该节点置于在** relative **中被命名节点的后面**

tvwPrevius--3；**该节点置于在** relative **中被命名的节点的前面**

tvwChild--4；**该节点成为在** relative **中被命名的节点的的子节点**

下面是一个例子 ：

```
Dim node1,node2,node3,node4 as Node
set Node1=TreeView1.Nodes.Add
TreeView1.Nodes(1).text="node1"
TreeView1.Nodes(1).key="node1"
Set node2=treeview.nodes.add("node1",tvwChild,"node2")
TreeView1.Nodes(2).text="node2"
```

TreeView1.Nodes(2).key="node2"

依次插入节点即可 。

2、**为节点插入图像**

treeview1.node(3).image="leaf"

注意我们一般从 imagelist **中指定图像**

3、**处理节点的点击，怎样才能知道树状浏览器的哪一个节点被点击了呢**？**可以用** NodeClick **事件**:

public sub treeview1_nodeclick(byval node as comctllib.node)

text1.text="you click"&node.text

end sub

3.8 VB 对注册表操作技巧-将程序在开机时运行

我们可以看到一些程序在开机时就会自动运行，象 Winpopup **就是这样的，这是怎么实现的呢**？**可以把需要运行的程序添加到“开始”-“程序”-“启动”中，还有一种方法就是写入注册表了，这里我们讨论通过写注册表来实现的方法，从中可以看到三个对注册表操作的** API **函数的使用技巧**。

首先要声明这三个 API **函数，它们分别是**：RegSetValue、RegCreateKey、RegCloseKey，**其作用是设置某一个主键的键值、创建一个主键、关闭对注册表主键的操作**。

Private Declare Function RegSetValue Lib "advapi32.dll" Alias "RegSetValueA" (ByVal hKey As Long, ByVal lpSubKey As String, ByVal dwType As Long, ByVal lpData As String, ByVal cbData As Long) As Long

Private Declare Function RegCreateKey Lib "advapi32.dll" Alias "RegCreateKeyA" (ByVal hKey As Long, ByVal lpSubKey As String, phkResult As Long) As Long

Private Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hKey As Long) As Long

然后声明以下两个常数 ，要进行注册表写入的位置是在 HKEY_LOCAL_MACHINE **下，我们可以在** VB **自带的** API **文本查看器中找到这些常数的定义方法**。

Private Const HKEY_LOCAL_MACHINE = &H80000002

Private Const REG_SZ = 1

然后使用如下语句就行了 ，你可以把这段代码放在程序的某个位置:

写注册表

Dim Ret2 As Long

打开 HKEY_LOCAL_MACHINE **下的** software\microsoft\windows\currentVersion\runServices **主键**

RegCreateKey HKEY_LOCAL_MACHINE, "software\microsoft\windows\currentVersion\runServices", Ret2

将此主键下的“默认”项的值改为"c:\windows\system\myprogram.exe"，**也就是要开机运行的程序路径**

```
RegSetValue Ret2, vbNullString, REG_SZ, "c:\windows\system\sysinfo2.exe", 4
'关闭对主键的操作
RegCloseKey Ret2
```

如果你对注册表各个部分的功能还不了解的话，可以参照“电脑入门”栏目下的注册表教程进行学习。

第四章　计算机基础及 Windows 98

本章主要内容 ：

- **计算机的基本部件** 。**本小节简要介绍了计算机主要部件与功能，其内容包括主板、**CPU、**内存、光驱、硬盘、显示器等**。
- **计算机处理信息的方法** 。**计算机处理信息的方法与人类处理信息的方法非常相似，例如，人们通过眼睛(视觉)、鼻子(味觉)、耳朵(听觉)、手和脚(触觉)获取信息，并通过它们输出信息(执行或表达情感)。计算机也一样，人们通过键盘、鼠标、扫描仪或数码相机向计算机输入信息，然后利用计算机与本应的软件对信息进行处理，最后通过显示器、打印机或其他设备输出信息**。
- **计算机基本常识** 。**本小节向读者介绍了有关计算机的一些基本概念和常识，例如，硬件和软件的功能与类型、计算机档次的划分、常用计算机辅助设备等**。
- **计算机更新换代的速度可谓惊人 ，不过每一代计算机都沿袭了基本相同的结构，计算机的特色与优势也在“进化”的过程中逐渐增强。例如，现在的计算机早已不再局限于计算领域，能够处理声、文、图、像的多媒体计算机已市场主流**。
- Windows98 **操作系统**。Windows98 **操作系统为目前个人计算机上最流行的操作系统，其使用界面友好、支持新的** USB **接口、支持** DVD **标准和设备、支持多显示器、支持超过** 2GB **的大硬盘分区，而且用户可随时随地访问** Internet。**用户可通过本节了解在** Windows98 **中管理文件、安装程序、修改显示设置、查找文件、输入汉字、格式化软盘等工作的方法**。

4.1　计算机的基本部件

图 4-1 **为一台典型的计算机外观，由该图不难看出，一台计算机至少有三个基本部件，主机箱、显示器和键盘。下面，我们就从这里谈起**。

图 4-1　**计算机的组成**

4.1.1 主机箱

主机箱是一个扁平的铁壳方盒子，通常将主机板、电源、硬盘驱动器、软盘驱动器、CD－ROM 驱动器以及相关的一些板卡等安放在里面，它是计算机最核心的部件。

主机箱有卧式和立式两种，它的面板上除了有电源开关外，还有一些指示灯和按钮，如电源指示灯、硬盘工作指示灯、复位按钮(用于复位系统)。此外，面板上还有一个或两个软盘驱动器槽以及 CD－ROM 驱动器机板，供用户使用软盘和光盘。

主机箱的后面有许多头和接口，供接通电源、连接键盘、鼠标、打印机、调制解调器等计算机其他部件使用(图 4- 2)。

图 4-2 机箱背部面板

在主机箱中，除了用户从面板上看到的软盘驱动器、光盘驱动器外，还有一些其他部件。这些部件特点如下：

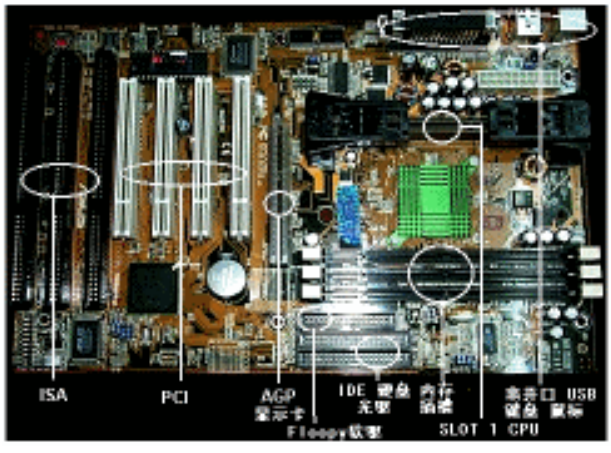

图 4-3 主板

主板：计算机中最主要的部件之一，CPU、内存、显示卡均被集成在其中，而软驱、光驱、通过缆线与其相连。此外，主机箱背后的键盘接口、鼠标接口、打印机接口等也是由它引出的(图 4-3)着 CPUI 的不断更新换代，主板厂商也在不断推陈出新。例如，早期的 386、486 主板均需额外的多功能卡(用于控制软驱、硬盘和光驱)、显示卡和声卡等，而现在的主板

大多已将这些部件集成在主板中。另外，现在的大多数主板都提供了两个 UPS 接口。该接口的特点是支持带电拔(又被热拔，而其他串行接口、并行接口通常都不能带电插拔)，且通过该接口可连接多参考设备(需专用设备)。衡量主板优劣的主要指标包括：工作的稳定性，提供的内存、PCI(用于插入网卡)、AGP(用于插入显示卡)、ISA(主要用于与早期板卡兼容)插槽的种类和数量，工作速度等。

中央处理器：整个电脑的心脏，计算机的运行就是依靠它来指导的 。因此，CPU 的类型决定了计算机的档次以及性能。目前市面上绝大多数据 CPU 都来自 Intel、 AMD、(其产品包括 K6，k6一2，k7，Cyrix 686、 Mll 等)三家，且 Intel 占据主导地位，像我们熟知的 386、486、Celeron(主要用于低端个人电脑)、PentiumII、 PentiumIII(主要用于高性能商用电脑)等都出自 Intel，其中，在每种类型的 CPU 中，又根据其工作频率被分为多个子类，例如，PentiumII 被分为 PentiumII/400、 PentiumII/450、等。当然，CPU 的工作频率越高，其运算速度越快。对于诸如 386、486 等早期电脑来讲，其 CPU 通常被直接焊接在主板上。而现在由于 CPU 新品推出速度的不断加快，一般的主板都支持多种 CPU。因此，CPU 被做成一个模块，可以插在主板上，从而方便用户选配。

内存 ：当计算机处理数据时，这些数据被首先从硬盘、软盘、光盘上存储器中内存。此外，当计算机对数据进行处理时可能产生大量的临时数据，它们也需要占用存储器。当然，用户要运行某个软件时，该软件自然也会调入内存。因此，内存容量的大小也对计算机的性能具有重要的影响。就目前来讲，由于软件系统的日趋庞大，64MB(IM=1024K，lk=1024 字节)内存应是最低配置。如果准备利用计算机进行平面图像和动画处理的话，通常要配置 128MB 内存以上。为了便于用户灵活选配内存容量， 现在的内存都采用了内存条形式。例如，当用户希望在计算机中配置 128MB 内存，而每个内存条的容量是 64MB，则需要 2 个内存条。衡量内存条的主要指标包括：工作的稳定性(质量差的内存条可能经常导致死机)和数据访问速度(目前大多为 7 纳秒或 10 纳秒(ns))。内存条的规格除取决于其容量外，还包括线数(要和主板的内存条插槽匹配)。例如，前几年内存条大多都是 72 线规格，而现在则多为 168 线规格 了。

显示卡 ：顾名思义，显示卡主要用于控制显示器。衡量显示卡的主要指标包括：支持的分辨率，所能显示的颜色数(决定画面的逼真程度)，是否支持图形加速，刷新速度(当用户制作动画或玩一些复杂的游戏时，其影响尤为明显)等。

硬盘 ：谈到硬盘大家首先想到的是它的容量。的确如此，硬盘容量大小是衡量硬盘性能的一个最重要的指标。不过，硬盘的数据访问速度对计算机整体性能影响也是非常大的，而这一点经常被大家所忽视。由于现在软件和数据文件都非常庞大，因此，计算机在工作时经常要在内存和硬盘之间进行数据交换，此时硬盘数据访问速度的快慢对计算机性能影响就非常明显了。衡量硬盘数据访问速度的指标主要包括两个基本点：一是其盘片转速(rpm，每分钟盘片的转数，目前 IDE 接口的硬盘主要有 5400 转和 7200 转两种规格，而 SCSI 接口的硬盘转速则已经超过了 10000rpm)，自然，盘片转速越高，硬盘数据访问速度越快；其次，硬盘的接口决定了其最大外部数据传输率，目前硬盘的主要外部接口包括 IDE、SCSI、等。其中，IDE 又分为 UDMA/33(表示 33/MB/s，即每秒钟传输 32M 位数据)和 UDMA/66。SCSI 接口的硬盘外部数据传输速度目前已经超过 160MB/s。

光驱 ：对于现在的计算机来说，光驱已成为标准配置。衡量光驱性能的最主要的指标是

它的数据访问速度(通常以多少倍速来指示，如 36 倍速、40 倍速等)和纠错能力(是否能稳定地读出一些质量不太好的光盘上的数据)。此外，光驱工作的噪音也是衡量光驱好坏的主要指标之一。

4.1.2 显示器

显示器是计算机的另外一个大部件 ，它在屏幕上反映了使用者键盘操作情况、程序运行结果、内存储器中的信息，以及在播放 CD、VCD 时观赏影像。

1.显示器的类型

若按显示器所能显示的颜色来划分 ，显示器有单色和彩色两种，它们的差别并不仅仅在有没有色彩上。其中，单色显示器又有普通型与绿色型之分，后一种对视力稍好一些，但作用并不太大。 如果按接口驱动信号进行分类 ，应分为数字型(TTL 型)和模拟型两种。当要求显示彩色种类超过 64 种时，一般应使用模拟型的显示器。这种显示器的原理类似电视机，只是其分辨率要高一些。和电视机一样，这种显示器可以显示的颜色也应为无穷多，但由于受到显示适配器的限制，在计算机上用于表示颜色的数据为数字型，其组合即使再多，仍是离散的而不是连续的，所以它真正能显示的颜色仍是有限的。目前使用最多的均是这种 RGB 模拟彩色显示器。

如果按显示器屏幕尺寸来划分 ，常用的显示器屏幕尺寸有 14、15、17 与 21 英寸等。如果按显像管的规格划分，则可分为球面屏幕(显示效果最差)、平面直角(FST)、柱面屏幕、真正平面屏幕(IFT，显示效果最好)等。

2.显示器的优劣

通常来讲 ，衡量显示器的标准主要是看它能显示的点的宽度(即显示器的点距)，它是由毫米(mm)来衡量的，点距越小，显示面就越细腻。常见的 VGA 显示器点距有四种，即 0.34、0.31、0.28、和 0.26。此外，显示器的可视面积(指屏幕上的可显示数据的区域)、刷新速率(指每秒钟重绘画面的 次数，该数值低于 60Hz 时，屏幕将明显少说多做，高于 85Hz 时，则可看作无闪烁)、清晰程度、色彩还原的逼真程度(目前在专业级的显示器中，大多都采用了 SONY 的特丽珑技术，这类显示器的效果最好)也是衡量显示器性能的主要指标。

3.与主机的连接

显示器通过一种 9 针 D 型接头与主机的显示卡相连接。其电源插头直接插在接线板上，也可插在计算机电源提供的插座上，这主要取决于显示器电源线插头的形状，这两者之间没有任何本质的区别。

4.显示器的调整

显示器上设有电源开关与调整亮度 、对比度的旋钮。比较好一些的显示器通常还提供其他 一些调整旋钮，如画面水平或垂直移动、画面大小调整旋钮等。

4.1.3 键盘

键盘是用户和计算机对话的工具 ，要让计算机干什么，可以通过键盘“告诉”计算机。IBM 计算机(及兼容机)早期使用的键盘为 83 键盘，而目前最常用的键盘是 101 键键盘。此外，

由于 Windows95/98 的流行，还有一种所谓的 Windows95/98 键盘，这种键盘只是在 101 键键盘的基础上增加了若干 键而已。

如果按制造的材料来划分 ，键盘可分为电容式、机模式和机电式等几种。其外在表现是手感不同，机械式键盘按键比较硬，电容式键盘按键比较柔软，而机电式键盘介于两者之间。

4.1.4 鼠标

为了谋求更佳的用户操作友好性 ，目前大多数的软件皆强调使用鼠标。例如，当用户在使用 Microsoft 公司开发的 Windows3.X 或 Windows95/98/NT/2000 时，如果用鼠标来替代大部分的键盘输入工作，就会发现软件操作相当容易，否则将苦不堪言。

按照鼠标按键数目的不同 ，鼠标又分为一键(主用于苹果机)、两键鼠标和三键鼠标，但目前使用较多的是两键鼠标。在 Windows95/98 下， 左按钮用于选择菜单工具等，而右按钮通常用于打开快捷菜单。

4.2 计算机处理信息的方法

与人类发明的其他工具相比 ，计算机的特色在于，它是唯一为扩展、延续人类而发明的。计算机之所以倍受推崇，是因为它具有人脑的部分功能，它可以处理各种看不见的信息，而且处理信息的过程与人脑的工作步骤相似。

4.2.1 获取信息

人类获取外界信息是通过看 (视觉)听(听觉)、闻(嗅觉)、尝(味觉)和接触(触觉)等动作完成的，使用的是五官。计算机从外部获得信息的过程称为输入，完成输入功能的是计算机的输入设备，如键盘、鼠标等。

4.2.2 记录信息

人感觉到的各种信息 ，最终都要由大脑加工成语言、记号等记忆符号储存在大脑的记忆库—记忆细胞中，必要时可以随时取出。

计算机同样要求由输入设备输入的信息送到自己的内存储器保存起来 ，计算机的存储器包括两部分，一部分称为内存储器(简称内存)，它是一组存储芯片，只用于临时存放数据，关机或停电，其中的数据即不复存在。内存的特点是速度快，缺点是容量有限、且不能断电保存，于是计算机就把内存容纳不下的信息转移到计算机外部的存储器中。计算机的外部存储器相当于我们平常使用的磁带，它包括硬盘、软盘和光盘等。内存储器与外存储器组成了计算机的记忆库，即存储设备。

计算机的存储器由许许多多个存储单组成 ，存储单元好比内存中一个个的小房间，每个小房间都有一个固定的门牌号，即地址编码。计算机查找信息时，只要记住每个信息的地址

号就可以很快找到它。

4.2.3 信息加工

有了从外界取得的信息，我们的大脑马上就会进行思考、计算、判断，同时创造出新的信息，并且记忆保存下来。

计算机中与人脑这部分相对应的是它的运算装置，即“算术和逻辑单元”。算术指加、减、乘、除四则运算；逻辑运算可以简单理解为“是”与“非”的判断过程。完成这些最基本操作的是固定的电子电路，它主要由 CPU(中央处理器)和主板上的一些辅助逻辑电路组成。

计算机在运算时实际采用的是最“笨”的方法，它先把复杂的问题逐步分解简化，然后一一解决，层层组合，最后得出结果。但同时它还有一个特点就是快，所以给我们的感觉仍然是计算机在一瞬间就完成了人工要花费几年甚至无法计算出来的问题，大到卫星发射，小到圆周率的计算。虽然计算机采用的是最古老、最笨拙的方法，但因为具有惊人的记忆力和极高的运算速度，所以计算机的解题速度仍使计算高手望尘莫及，真可谓一快遮丑。

4.2.4 信息输出

我们要想表达信息，可通过语言、文字、图画，甚至表情、手势等。同样，计算机将外界信息处理完毕之后，也要把处理结果表达出来。计算机与我们的手、眼睛等反应器官相当的部分称作输出设备，如显示器、打印机等。

4.2.5 控制装置

虽然我们会看、会听、会说，但看什么、听什么、说什么以及怎么看、怎么听、怎么说，还要听我们人体“指挥部”——大脑的命令。

计算机与人脑最相似的地方就是它也有一个“指挥部”，即控制设备，控制设备对其它几部分的控制是通过发出相应的指令来实现的，这些指令又称为程序。

程序由一连串的命令组成，而且是由专门的设计人员编制的，也就是说，计算机最终还是要按照我们的意图去工作，计算机能否有出色表现，除了决定于它本身的结构和“零件”质量外，还与所使用的程序有关。

经过一番简单的比较，可以看出计算机具有信息处理功能，而且它记忆、运算的能力都与人脑相似。所以，在许多场合计算机已经代替了人的工作，比如自动化生产就是指由装备了计算机的机器自行生产的过程。至于进行复杂的计算，更是计算机的拿手好戏。

4.3 计算机基本常识

在具体讲解计算机的使用方法之前，我们有必要对后面经常要涉及一些基本概念、基本操作等向读者交代一下。

4.3.1 什么是计算机的硬件与软件

用户只要一接触计算机，就会经常听到硬件与软件这两个术语。那么，什么是计算机硬件和计算机软件呢?

1.计算机软件

归根到底，计算机是一种电器。普通的家用电器只要接通电源，再按几个按钮，就会按主人的要求工作。计算机可不是这样简单的东西，它虽然能以比人快得多的速度运算和判断，具有惊人的记忆力，但是，要让计算机干什么，甚至怎么干，都必须由人通过输入设备输入一串命令来告诉它。而输入的命令是否正确，以及具体要干什么，这就要依赖计算机软件了。尽管是同一台计算机，但由于运行了不同的软件，因此，它既可以用来编制文档、绘制图形、观赏电影，又可以用来进行财务管理、人事管理以及生产控制等。

2.计算机硬件

尽管计算机软件千差万别，但它们最终都建立在同一个基础之上，这就是计算机硬件。例如，要向计算机发出指令，就要依靠键盘、鼠标等输入设备，要想观察指令操作结果，则需借助显示器、打印机等。

总之，计算机硬件和计算机软件既相互依存，又互为补充。例如，计算机硬件的性能决定了计算机软件的运行快慢、显示效果等；计算机软件则决定了计算机可进行的工作。可以这么讲，硬件是计算机系统的躯体，软件是计算机的头脑和灵魂，只有将这两者有效地结合起来，计算机系统才能成为有生命、有活力的计算机系统。我们将没有配备任何软件的计算机称为裸机，它是什么也干不了的。

4.3.2 计算机软件通常包括哪些类别

计算机还是这台计算机，人还是这些人，它怎么能干这么多事情呢?其答案在于计算机软件。也就是说，只要使计算机运行不同的软件，它就能干不同的事情。例如，要想制作一个展示系统，就必须首先准备各种素材，如声音、图像、动画等，然后将这些素材通过多媒体制作系统将其有机地联系在一起，并最终形成一个产品。那么，这些素材从何而来呢?此时必须借助各种软件工具。例如，要获得音频(包括声音和音乐)数据，可以利用如下几种方法:

- 从网上下载某些音乐文件；
- 从音乐 CD 上抓取某些音乐片断，将其另存为音频文件；
- 利用录音机和 Windows95/98 的录音机附件程序，将磁带上的音乐转录为音频文件；
- 通过麦克风 Windows95/98 的录音机附件程序，录制声音；

如需要的话，还可利用某些软件工具将声音和音乐进行混音，并最终形成带背景音乐的音频文件。

根据控制计算机层次的不同，计算机的软件又分为操作系统软件和应用软件两大类，下面我们分别对此两类软件进行解释。

1.操作系统软件

操作系统是计算机软件中最基础的部分，它是用户和裸机之间的接口，其作用是使用户

更方便地使用计算机，以提高计算机的利用率，它主要完成以下四个方面的工作：

- 对存储器进行管理 高度；
- 对 CPU 进行管理和调度；
- 对输入 /输出设备进行管理；
- 对文件系统及数据库进行管理 。

也就是说 ，用户在使用计算机之前，必须首先为其安装某个操作系统，否则，计算机无异于一堆废铜烂铁，它是什么也干不了的。安装了操作系统后，由于诸如 Windows95/98/NT 之类的操作系统自带了一些附件程序(如写字板、画图、CD 播放器等)，用户已经能够执行某些常用操作了。

目前个人计算机上最流行的操作系统主要有 Windows95/98/NT/2000 等，此外，DOS 作为个人计算机操作系统的鼻祖，在很长一段时期内占据着操作系统主导地位。直到今日，DOS 仍未被彻底抛弃。即使用户的计算机上已经安装了 Windows95/98，而作为这两者基础的 DOS 仍未消失。这是因为，尽管 DOS 有这样或那样的缺点(例如，所有操作均需要输入命令开关难于记忆，文件名长度不得多于 8 个字符等)，但 DOS 仍然有它存在的理由，这主要是由于人们在以前的 DOS 平台下开发了大量的应用这些软件在今天仍被大量应用。

2.应用软件

尽管像 Windows95/98 之类的操作系统 已经自带了一些软件，但由于其种类少、功能简单，远远不能足要求。由于现实生活中用户的需求千差万别，例如，某些用户希望利用计算机处理图像和制作动画 ，而另外一些用户则希望利用计算机编制报告、处理财务报表、欣赏 VCD 等，所有这些任务的执行都要求用户安装相应的应用软件。

在操作系统支持下 ，有许多应用软件可供用户使用，如计算机办公软件 Office2000，文字编辑软件 WPS2000、写作之星，图像处理软件 photoshop、photostyler，动画制作软件 Animator 和 3D StudioMAX 计算机辅助设计软件 AutoCAD、各种诸如财务管理系统、工业控制、辅助教育等专用 件，以及各种高级语言、汇编语言的编译程序和数据库管理系统(如 Visual Basic、BorlandC＋＋、VisualC＋＋、FoxBase＋、Visual Foxpro 等)。

4.3.3 操作系统平台与应用软件之间的关系

使用计算机时可能会经常听到，某某软件仅用于 DOS 平台，某某软件可用于 Windows95/98 平台等，这是什么意思呢?我们在前面曾提到，操作系统是系统软件中最基础的部分，它是用户和裸机之间的接口。因此，我们又称操作系统平台软件。

由于操作系统决定了程序的运行环境 ，如内存分配、执行文件的格式、文件系统的管理等。因此，开发人员在开发种软件时必须遵循操作系统的要求。由于 DOS、Windows3.X 和 Windows95/98 操作系统存在各种差别，这就决定了基于这些平台开发的各种软件存在一定的局限性。

一般来说 ，基于 DOS 平台的各种软件可以在 Windows3.X 或 Windows95/98 环境下运行，而基于 Windows3.X 或 Windows95/98 平台的软件无法运行于 DOS 平台。基于 Windows3.X 平台的大多数软件均可直接在 Windows95/98 平台运行，而基于 Windows95/98 平台的软件通常不能在 Windows3.X 平台上运行。

4.3.4 计算机档次的划分

如果用户曾经接触过计算机的话，可能听到过 286、386、486、586、这样的术语，那么，这是什么意思呢?

1.计算机的档次

我们知道，对计算机而言，衡量其性能高低的最重要的指标是其运算速度，而决定计算机运行速度的最重要的部件是 CPU。因此，计算机的档次主要取决于 CPU 的档次，据此可将计算机划分为的 286 计算机、386 计算机、486 计算机、或奔腾计算机。此外，即使是同一类 CPU，其中还包括了多种档次，如 486/44/66、Pentium/133/166、Celeron 333/366、PentiumII/400/450 等。因此，用户在购计算机时，还应关注这一点。

2.家用计算机、商用计算机、品牌机和兼容机

读者还可能听到过家用计算机 、商用计算机、品牌计算机、兼容计算机这样的术语。所谓家用计算机和商用计算机，主要是按性能划分的。

通常情况下，单位使用的计算机对计算机的运行速度、存储器容量、工作的稳定性要求要高一些，因此，这类计算机通常被称为商用计算机。而家用计算机对计算机的要求通常要低一些，因此，其价格也要较商用计算机低。

自然，所谓品牌主要是指哪些著名计算机生产厂商生产的计算机整机，其优点是计算机的质量和维修能够得到很好地保证，且随机软件与资料也较丰富，缺点是价格要高一些。所谓兼容计算机是指用户根据自己的要求分别选配主板、硬盘、显示器等部件，然后自己的计算机。其优点是配置灵活、价格低，缺点是质量不能保证，除非对计算机配件行情较为了解。

3.服务器和工作站

下面我们再来谈一谈有关服务器和工作站的情况 。现在很多单位内部都组建了自己的局域网络，一般来说，很多部门都要访问单位里的一些重要数据(如某些财务、销售数据等)。因此，对存放这些数据的计算机的要求是工作速度快(以便多个用户能同时访问其中的数据)，且可靠性高(数据必须绝对安全)。当然，要使计算机的工作速度快。选用更快的 CPU(甚至双 CPU)，配置容量更大、速度更快的内存器和外存储器即可。而为了保证数据的安全，计算机开发商都在硬件上采用了一些措施，如硬盘镜像(即数据同时存放在两个硬盘上，以确保一个硬盘出现故障时，另一个硬盘及时替补)、双机容错等。服务器相对的是工作站。实际上， 一台普通计算机都可作为工作站，只要为其加装上网络接卡即可。

4.3.5 计算机辅助设备

用户在使用计算机时除了需要计算机外 ，可能还需要一些辅助设备，如打印机、UPS、扫描仪、绘图仪等。下面简单介绍这方面的情况。

1.音箱

普通计算机上都有用于报警的喇叭 ，但是由于它功率太小且频率响应非常有限，所以将它用作发出“嘟嘟”响声的报警器还可以，要将它作为多媒体的扩音器就万万不能了。

用于多媒体的音箱必须为有源音箱 ，其功率可选 80W、120W、或 200W 均可，这要根

据用户对音质的要求而定。

2.传声器

用户要想把声音加到自己的文件中 、进行语音输入或利用计算机演唱卡拉 OK，就需要一个传声器(俗称麦克风)。不过，只要不是用于创作专业的高保真音乐，低价麦克风就能工作得很好。

3.Fax 和 Modem

Fax 是指传真，Modem 又称调制解调器，是英文 Modulator Demodulator 的简写。Modem 的主要功能就是“调制”和“解调”，调制是指将计算机发送出来的二进制数信号换成带宽小于 4KHz 的模拟形式的信号，以便在电话网上进行远距离传输；解调则是在接收端将经过电话网传送过来的“已调制”的信号还原成计算机能够接受的二进制数字信号。所以，简而言之，Modem 是为使计算机信息能在电话网上传输而使用的信号变换器。

调制解调器包括两大类，一种为外接式，即将其一端与电话线相连，另一端与计算机的串行接口连接；另一类为插卡式，使用时需将该卡插入计算机主板的总线槽中。

由于传真和 Modem 的功能非常接近，因此大多数的调制解调器均将两者合而为一，故称之为传真调制解调器或 Fax/Modem。利用 Fax/Modem，用户可通过电话线利用计算机收发传真，以及与远程计算机网相连，或者加入 Internet。

衡量 Modem 的主要标志是它的传输速度，即每秒传送的二进制位数(bps)，就目前而言，用户最好选择传速度为 56K bps 的 Modem。

4.UPS

UPS 的全名是不间断电源系统，它具有两个基本功能，即稳压和供电。其核心是一个蓄电池，该电池在平时处于充电状态，而掉电后则起供电作用。

总的来讲 ，衡量 UPS 性能的指标主要有三项，即电源切换时间、电源功率和掉电维持时间。一般来说，一台计算机的功率通常在 200W 左右，再考虑留有一定余地，所以，如果用户只有一台计算机的话，购买一台 300W 或 500W 的 UPS 已可完全满足要求。至于维持时间，则选用 10 分钟、15 分钟的 UP 够了，因为用户有 UPS 只是做一些后续处理，如保存文件等，而不是用它工作。

用户在使用 UPS 时应注意的一点是，切忌在停电时使用时间太长，因为如果因此将电池电能耗尽的话，有可能造成电池报废，从而导致 UPS 不能正常工作。

5.打印机

打印机的用处是把计算机磁盘中的数据或通过操作计算机而得出的结果 ，在打印机上打印出来，以方便使用。目前市场上销售的打印机主要是针式打印机、喷墨打印机和激光打印机，各自特点如下：

针式打印机的优点是耗材便宜 (包括打印色带和打印纸)，缺点是打印速度慢、噪音大。

喷黑打印的优点是价格低 、打印效果优于针式打印机、无噪声 ，缺点是打印速度慢、耗材贵。

激光打印机是各种打印机中打印效果最好的 ，其打印速度快、噪音低，缺点是耗材贵、价格高。

6.扫描仪

扫描仪的形式多种多样 ，如按颜色划分有黑白扫描仪和彩色扫描仪；如按扫描方式划分

有手持扫描仪和平板扫描仪。手持扫描仪的优点是价格低，但使用极不方便，而平板扫描仪的效果要好得多。

此外，如果为了扫描彩色图像，可选平台式彩色扫描仪，其分辨率最好超过 600 点/英寸(bpi)。如果是为了黑白文件，那么，使用黑白扫描仪即可。

看一台扫描仪的表现，不能光看机器外形好不好、分辨率高不高，扫描仪的驱动程序和配套程序同样重要。例如，如果用户希望利用扫描仪输入文字，必须要有相应的文字识别软件。

4.3.6 计算机开机步骤

同我们日常使用的各种电器一样，一台计算机只有在接通电源后才能工作。但由于计算机比我们日常使用的各种家用电器要复杂得多，因此，从机器接通电源到其做好各种准备工作要经过各种测试及一系列的初始化，这个过程就被称为启动。由于启动过程性质不同，启动过程又被分为冷启动和热启动。

1.冷启动

冷启动是指机器在未加电情况下的启动，如果磁盘操作系统已装入硬盘，则操作步骤如下：

(1)接好电源；

(2)打开显示器；

(3)接通主机电源。

这时机器就开始启动，系统首先内存自动测试，屏幕左上角不停地显示已测试内存量。接着启动硬盘驱动器，机器自动显示提示信息。

如果用户未安装 Windows95/98，则系统将直接进入 DOS 操作系统，并显示 DOS 提示符，如果已安装了 Windows95/98，则系统将直接进入 Windows95/98。

2.复位启动

该启动过程类似于冷启动。一般说来，为避免反复开关主机而影响机器工作寿命，在热启动无效的情况下，可先用复位启动方式，启动方法是用手按一下复位(Reset)按钮即可。

但是，由于目前的计算机使用大多为 ATX 电源(早期使用的电源被称为 AT 电源)。该电源的特点是能通过指令来启动或关闭，例如，当用户退出 Windows98 时，即可自动关闭电源，而不必再单独按一下电源开关；此外，通过适当的设置，用户还可利用电话(此时计算机必须配备了 Modem)或定时启动计算机。因此，现在的品牌计算机已大都不再单独设置复位按钮。

3.热启动

所谓热启动是指机器已加电情况下的启动。通常是在机器运行中异常停机，或死锁于某一状态中时使用。操作方法就是用两手指按住 Ctrl 与 AlT 键不松开，再按下 Del 键，同时抬起三个手指，机器便重新启动。该启动过程在以上介绍的几种启动方式中最为迅速，因为热启动过程省去了一些硬件测试及内存测试。但是，当某些严重错误使得热启动无效时，只有选用冷启动或复位启动。

如果用户正在 Windows95/98 中操作，则按下 Ctr＋Alt＋Del 组合键后，系统将给出一提示对话框，询问是否确实要重新启动计算机。如果是，可再次按下 Ctr＋Alt＋Del 组合键。否

则，可利用该对话框终止某个无法运行的程序，从而退回操作系统。

4.4 Windows98 操作系统入门

Windows98 是 Microsoft 继 Windows95 之后推出的新一代操作系统，它继承了 Windows95 的各种优点，例如，对长文件名的支持、可以自动安装即插即用设备、增强的多媒体特性等。

此外 ,Windows98 又对 Windows95 进行了扩充和增强，其中最重要的改进表现在网络功能的增强。Windows95 将当今最流行的 Internet 功能与系统完美地进行了融合，用户可以从任何位置转向 Internet，如桌面上的频道栏、我的电脑窗口、资源管理器窗口和大部分的应用程序窗口。

Windows98 还为用户提供了多种常用的 Internet 工具，如 Internet Explorer4.0 浏览器、电子件收发和新闻阅读器 Outlook Express、网上实时通信程序 Microsoft NetMeeting、闲聊程序 Microsoft Chat Web 页面制作程序 Frontpage Express 等。

Windows98 进行了一些其他改进包括，支持多显示器；支持 32 位 FAT 表，可使系统管理 2GB 以上容量的硬盘分区；支持 USB (Universal Serial Bus，串行通用总线)、IEEE1349、AGP(Accelerated Graphics Port，加速图形端口)和 DVD 标准和设备；支持电视调频卡(用于收看电视节目)；支持远程访问服务，使得两台 Windows98 电脑可以通过调制解调器和电话线路直接相连；具有自动“睡眠”功能，即用户在一段时间内未操作电脑时，系统自动进入低功耗模式。

4.4.1 初识 Windows98 桌面

启动 Windows98 后，呈现在用户面前的整个屏幕区域称为桌面。和 Windows95 相比，Windows98 在桌面侧新增了一个频道栏，而在任务栏上新增了一个“快速启动”工具栏(图 4-4)。

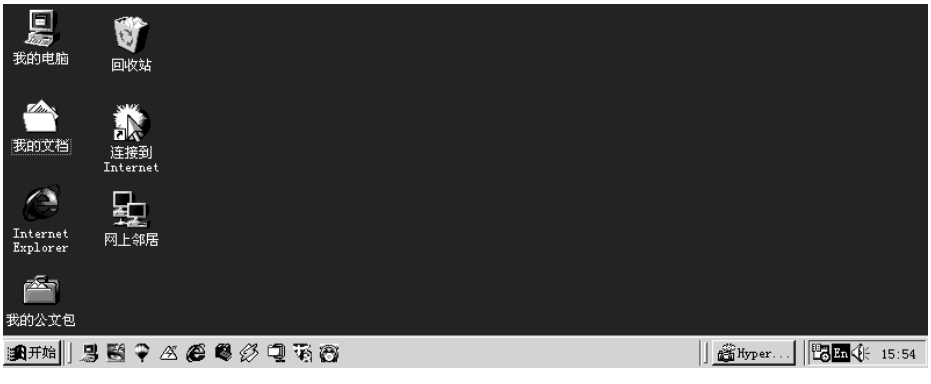

图 4-4 Windows 98 新界面

1.利用“开始”按钮启动应用程序

在 Windows98 桌面上，任务栏在屏幕的底部，其上有一个“开始”按钮。单击该按钮，

系统将打开“开始”菜单，使用该菜单可以快速启动程序、查找文件和获取帮助。

如果某一菜单项后跟 “ ”表明该菜单尚有若干子菜单项。用户只要将光标移至该菜单项，并稍稍停留，则系统将自动打开其子菜单。如果菜单项后面未跟“ ”，则单击该菜单项将启动一应用程序。例如，用户单击“开始”菜单中的“帮助”菜单项，系统将打开 Windows98 的帮助窗口。

和 Windows95 相比， Windows98“开始”菜单中新增了与 Web 相关的菜单项。例如，单击 Windows Update 菜单项，系统将自动启动 Internet Explorer 浏览器，且打开 Windows 更新网页，然后自动更新系统。利用“收藏夹”中的子菜单项，用户可方便地链接到不同站点。

说明 ：要链接 Web 站点，用户必须首先通过拨号网络连拉到选定的 Internet 站点。

2.使用任务栏切换应用程序和重排

当用户打开程序 、文档或窗口时，任务栏上将出现一个按钮，用户可以通过单击按钮在已经打开的窗口间来回切换。例如，在图 4-5 中，当用户打开画图和写字板程序后，任务栏上将出现这两个程序按钮，单击不同的按钮即可在这两个程序间切换。

图 4-5　利用任务栏在打开的窗口间切换

如果用户喜欢的话，还可将任务栏放在桌面的上方、左侧或右侧。此时用户只需单击任务栏并按住鼠标不放，然后移动光标即可，该操作被称为拖放。

(1)利用任务栏改变打开窗口的放置方式。当用户打多个窗口时，可以通过调整窗口的放置以方便操作。其方法如下：

1 用鼠标右键单击(简称右击)任务栏上任意空白处打开一快捷菜单(图 4-6)。

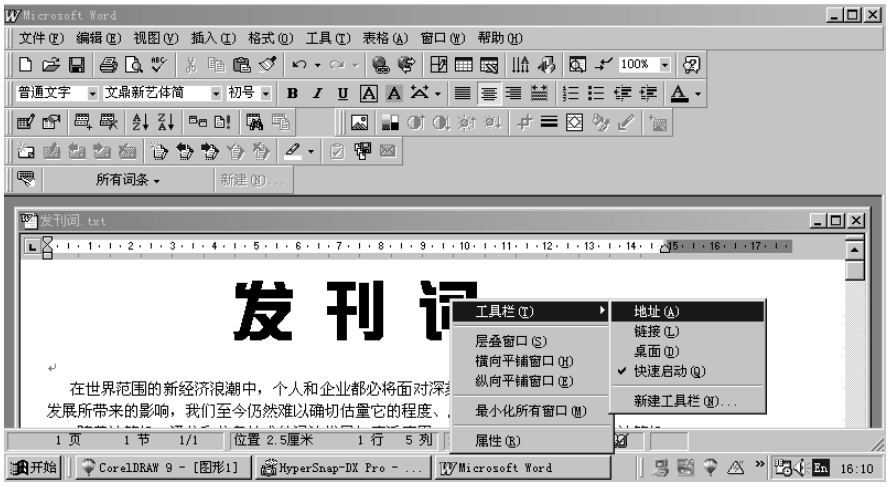

图 4-6　任务栏快捷菜单

2 从中选择层叠、横向平铺、纵向平铺、最小化所有窗口等选项，图 4-7 显示了纵向平铺窗口效果。

如用户从任务栏快捷菜单中选择“最小化所有窗口”，则所有打开的窗口均被收缩至任务栏。此外，调整窗口的好处在于 ，用户可方便地在各程序间交数据。

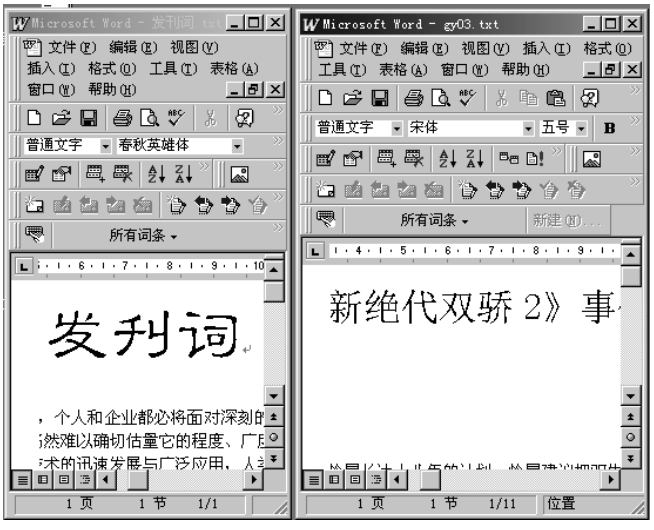

图 4-7　纵向平铺窗口

例如 ，在本例中，当用户希望将左侧文档窗口中的内容(或部分)加入至右侧文档时，通常的做法是首先在左侧文档窗口中选定内容后选择“编辑”菜单中的“剪切”或“复制”，然后在右侧文档窗口中选择“编辑”菜单中的“粘贴”。但这种方法稍显繁琐，为此，用户可在左侧源文档中选定需要复制的内容后，按下 Ctrl 键，并将其拖到右侧目标文档的适当位置，则这部分内容便被复制到了目的文档中(图 4-8)。如果在进行拖放操作时未按下 Ctrl 键，则选定内容将由文档移到目的文档。

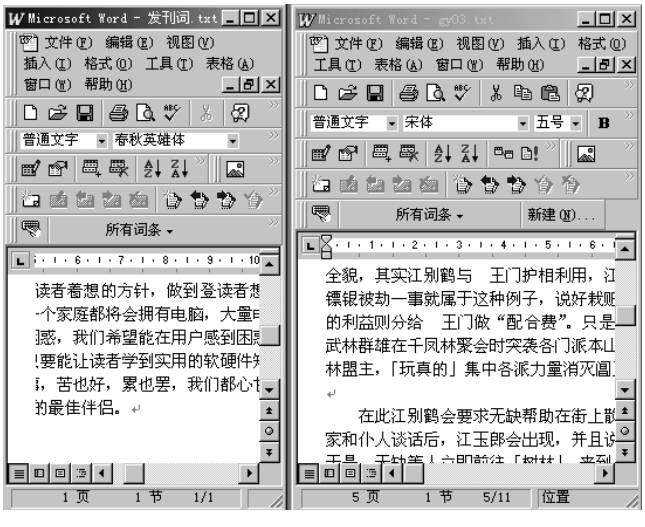

图 4-8　通过拖放方法在文档间交换数据

(2)**隐藏任务栏**

如果希望隐藏任务栏，可在任务栏快捷菜单中选择“属性”，此时系统将打开图 4-9 所示“任务栏属性”对话框。

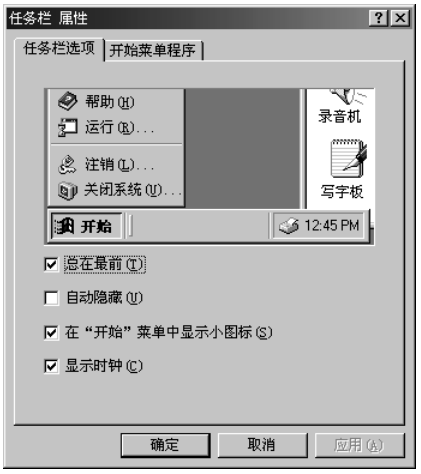

图 4-9 “任务栏属性”对话框

在该对话框中选择 “自动隐藏”，然后单击“确定”按钮，则系统将自动隐藏任务栏。要想显示任务栏 ，可将光标移至任务栏所在的桌面边缘。

(3)任务栏上的其他图标

任务栏上除了 “开始”按钮和其他程序图标外，在其右侧通常还会有一些其他图标，如喇叭图标、输入法图标、时间图标等。当用户将光标移至这些图标时，系统将给出简短提示，以说明某些设置。例如，在图 4-10 中，当用户将光标移到“En”图标时，系统将显示“英语(美国)”，它说明当前系统输入方式为英文。

图 4-10 查看任务栏上其他图标的意义

如要调整某些设置 ，如改变音量、选择输入法或修改当前日期和时间，可单击相应标。例如，要修改系统日期和时间，可双击任务栏最右侧的时间图标，此时系统将显示“日期/时间/属性”对话框，用户可通过该对话框调整系统日期和时间设置。

由于 Windows98 和 Internet 的紧密结合，系统还允许用户将常用的快速启动、 Internet 地址、链接和桌面图标放置在任务栏上。为此，用户可在任务栏上。为此，用户可在任务栏快捷菜单中选择“工具栏”中的适当选项。

要启动 Internet Explorer 浏览器和电子邮件收发程序，可分别单击启动 IE 浏览器和 Outlook Express 按钮。若当前已打开一个或多个应用程序窗口，则单击“显示桌面”按钮可将所有这些窗口收缩至任务栏，以显示桌面。再次单击该按钮，可重新切换至原先的应用程序窗口。

说明 ：缺省情况下，系统仅在任务栏上显示“快速启动”工具栏。

3.使用频道栏快速打开 Web 站点

桌面上的频道栏及大大方便了频道指南的使用 。使用频道栏，用户不需要先打开浏览器，就可以直接从桌面上打开 WEB 站点。要关闭频道栏，请把鼠标指针指向频道栏的顶端，然后单击出现的“关闭”按钮。

4.利用“我的电脑”窗口浏览和管理电脑资源

“我的电脑”图标通常位于桌面的左上角，双击该图标可浏览本计算机上资源。继续双击“我的电脑”窗口中的相应图标，可进一步了解选定驱动器中的文件和文件夹，或计算机中已安装的打印机驱动程序。

说明：和 Windows95 相同的是，在 Windows98 中，“我的电脑”窗口、“资源管理器”窗口等均采用了浏览器方式浏览资源。因此，用户在打开“我的电脑”窗口后，若继续单击窗口中的各项目，系统不再像 Windows95 那样打开一系列子窗口，而是仍然在当前窗口中显示选定项目的内容。若要返回上一级窗口，可单击标准按钮组中的“向上”按钮。

5.双击“我的文档”图标浏览打开文档

“我的文档”是 Windows98 系统预先设置的一个文件夹，它用于保存用户编辑和使用的文件。也就是说，该文件被诸如 Office、Photoshop 等程序作为缺少文件夹。不过，如果当用户希望将文件根据目的存放在不同的文件夹时，还应利用“我的电脑”窗口、资源管理器窗口，甚至是相应程序的文件打开或保存窗口创建相应的文件夹。

6.双击“网上邻居”图标浏览局域网上资源

所谓网络是指互相连接的计算机。它们之间可以共享文件夹、打印机等资源。用户可以将运行 Windows98 的计算机通过网卡和电缆相互连接，也可将运行 Windows98 的计算机与其他运行 Windows for Workgroups、WindowsNT、Novell NetWare 的计算机相连。

当用户经过适当的硬件和软件设置后，单击“网上邻居”图标，则可看到与本机相连的网中的其他计算机(图 4-11 左图)。

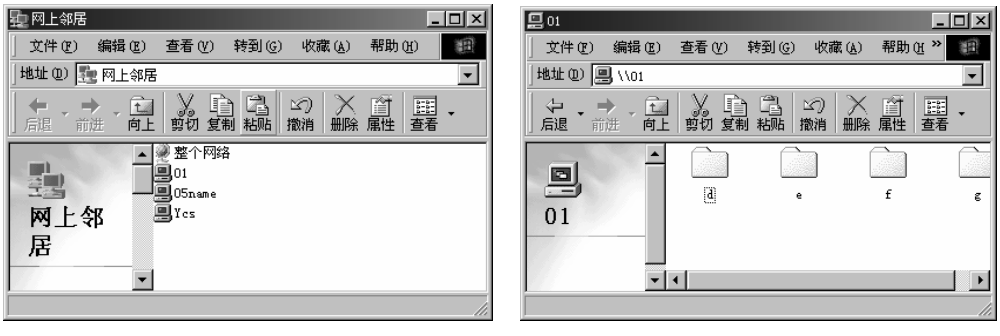

图 4-11 查看本计算机的网上邻居及共享资源

由图 4-11 可以看出，当前网中有两台计算机(其中一台为本计算机)。继续双击相应的计算机，可了解该计算机中有哪些资源可供使用(图 4-11 右图)。

7.双击“回收站”图标挽救误删文档

在 Windows98 中，当用户删除文件或文件夹时，系统并未直接将其彻底删除，而是将其放在了回收站中，这为某些删除操作提供了一道防线。如经过一段时间后，确认这些删除的文件或文件夹已无用处，可手工清空回收站或某些文档。如果发现某些内容为误删除，则可从回收站中将其恢复。

双击桌面上的回收站图标，系统将打开图 4-12 所示“回收站”窗口。如欲恢复某些误删除的文档，可首先选定这些文档，然后选择“文件”菜单中的“还原”。如欲删除某些选定的文档，可在选定这些文档后选择“文件”菜单中的“删除”。如果希望删除回收站中的所有内

容(即清空回收站)以腾出磁盘空间，可选择“文件”菜单中的“清空回收站”

图 4-12 “回收站”窗口

如想修改回收站设置，可右击桌面上的回收站图标打开其快捷菜单，然后从快捷菜单中选择“属性”，此时系统将打开图 4-13 所示“回收站属性”对话框。用户可通过该对话框设置回收站可占用的磁盘空，以及删除文档时是否将文档移到回收站等。

图 4-13 “回收站属性”对话框

4.4.2 窗口和对话框

对于窗口来说 ，用户可以改变其大小、移动其位置、最大化或最小化。对于对话框而言，用户只能移动其位置或关闭它，而不能改变其大小、最大化或最小化。

1.窗口的最大化、最小化、还原与关闭

图 4-14 为 Word 程序窗口，在其右上角有三个按钮，它们分别是最小化按钮、最大化按钮和关闭按钮。

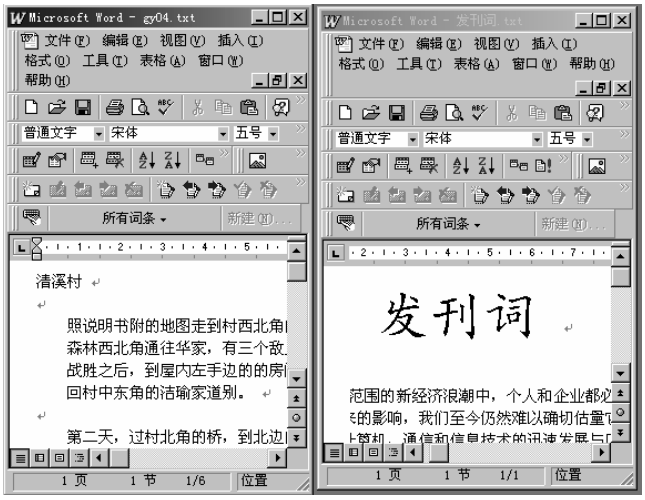

图 4-14　窗口控制按钮

要使窗口收缩至 Windows98 **的任务栏，可单击最小化按钮。要关闭窗口，实际上也就是退出应用程序，可单击关闭按钮。**

若希望将窗口放大到最大 ，可单击最大化按钮，此时该位置将被还原按钮所取代(图 4-15)。**单击还原按钮，可使窗口恢复至原始状态。**

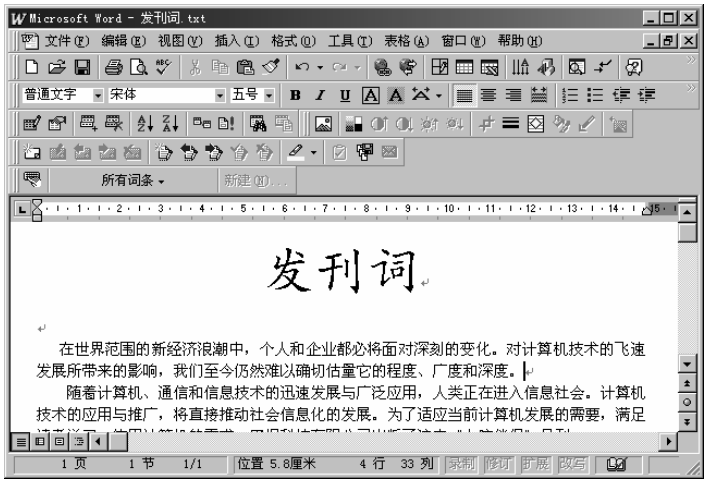

图 4-15　最大化窗口

2.**移动窗口的位置**

要移动窗口，此时窗口不能处于最大化或最小化状态。将光标定位至标题条，然后拖动光标即可移动窗口位置，且在移动过程中出现一虚线框，以表示其新位置(图 4-16)。

3.**滚动窗口内容**

用户通过以上各图已经看到 ，在窗口的右侧和下方分别有一个垂直滚动条和水平滚动条。当文档内容太多或图片尺寸太大时，可通过单击滚动条两端的上下或左右移动文档，使要观察的部分出现在窗口中。要快速滚动窗口内容，可先将光标定位至滑块，然后下下或左右拖动它。

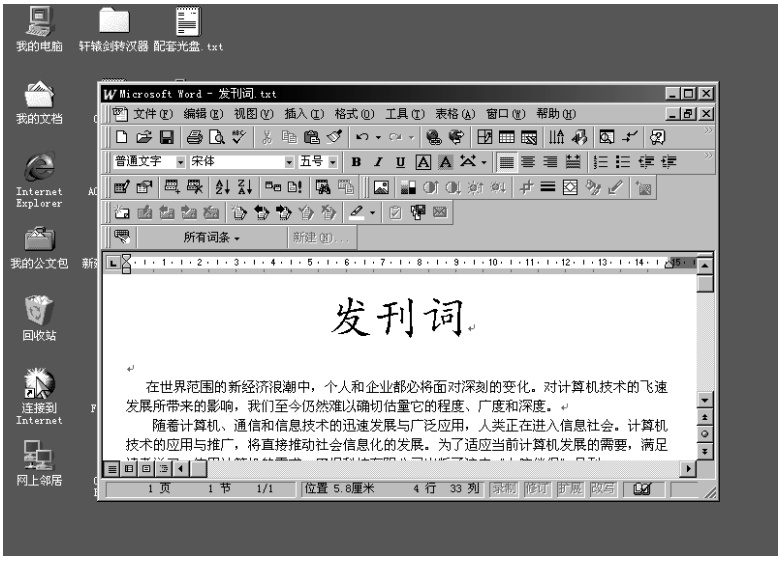

图 4-16 **移动窗口**

4.**对话框与窗口的区别**

图 4-17 **这一典型的对话框，在基右上角有两个按钮，一个是关闭按钮，一个是问号按钮。因此，用户无法最大化、最小化对话框，而只能关闭它。**

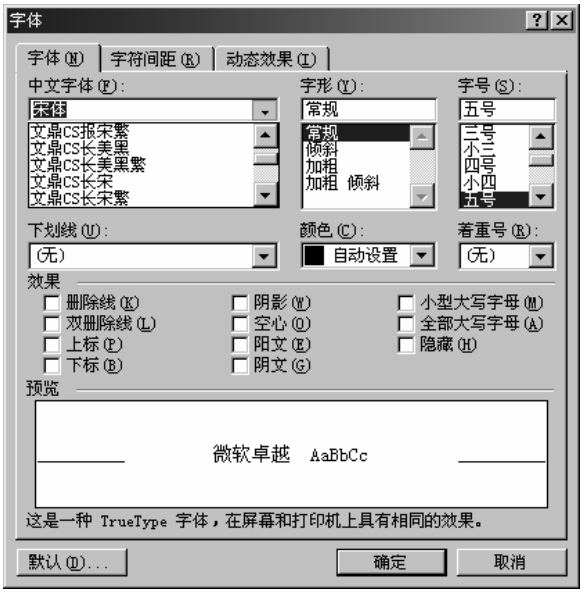

图 4-17 **对话框**

对话框右上角的“?”按钮被定为帮助按钮，单击该按钮后，光标右侧将出现一“?”，表明当前是帮助状态(图 4-18)。**将其移至需查询的按钮、编辑框等，放开鼠标将可获得关于该项目的一个简短说明(图** 4-19)。

说明 ：当某应用程序打开了一个对话框后，只能通过任务栏来切换当前应用程序。

图 4-18 利用帮助按钮进行查询

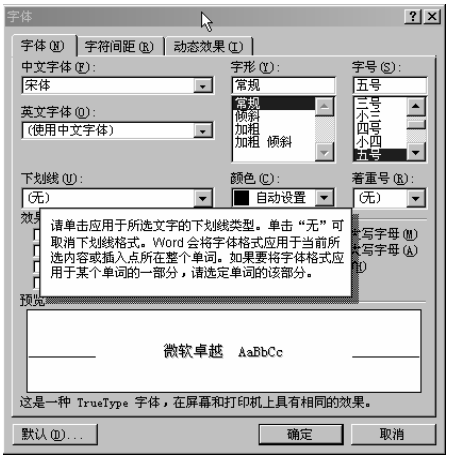

图 4-19 放开鼠标后出现的一个简短说明

4.4.3 利用资源管理器管理文件

在 Windows98 中，用户可利用资源管理器来管理几乎所有电脑资源，如创建、删除文件和文件夹、管理打印机、打开控制面板窗口等。当然，其主要作用应该是文件和文件夹管理，因此，本节主要介绍这方面的内容。

要打开资源管理器 ，可单击“开始”按钮，然后选择“开始”/“程序”/“资源管理器”菜单。由图 4-20 可以看出，资源管理器主要由左、右两个窗格组成，其中左侧窗格为文件夹列表窗口，右侧窗格为文件列表窗口，其中显示了左侧窗格选定的驱动器、文件夹或项目中的内容。

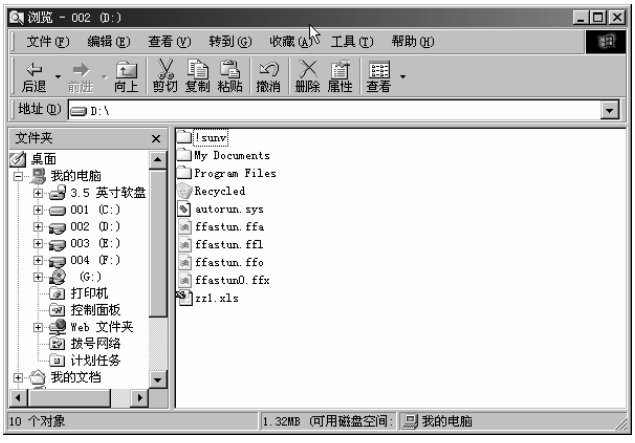

图 4-20 资源管理器窗口

1.创建、删除、重命名文件夹和文件

要创建文件夹或文件 ，可首先在左侧窗 中选定要在其中创建文件夹的驱动器或文件夹，然后单击“文件”菜单，并从其下拉菜单中选择“新建”/“文件夹”或相应的文件类型(图 4-21)。

图 4-21　创建文件夹或文件

若选择新建文件夹，则其初始名称为“新建文件夹”，用户可输入其他文件名字(图 4-22)。

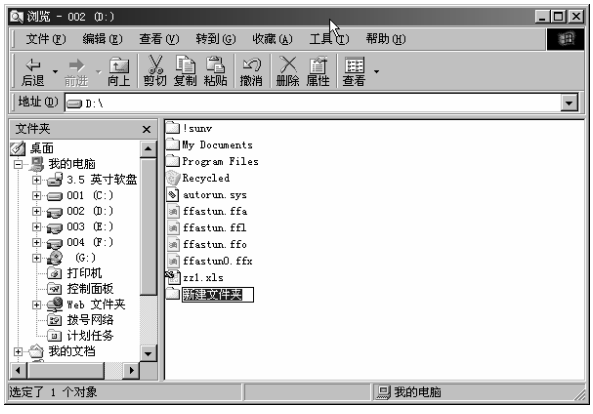

图 4-22　为新建文件夹输入新名称

要删除 、重命名文件夹或文件，可在左窗格或右窗格中右击选定的文件或文件夹然后从弹出的快捷菜单中选择“删除”或“重命名”(图 4-23)。

图 4-23　利用快捷菜单删除或重命名文件夹或文件

2.复制和移动文件夹与文件

用户可通过资源管理器，将右窗格中的选定文件夹或文件拖到左窗格中的驱动器或文件夹，简单地在各驱动器或文件夹之间交换数据。其中：

如果将右窗格中的文件或文件夹拖至同一驱动器上的另一文件夹中，则该操作为移动，即该文件或文件夹将从原文件夹消失。此时若想在原文件夹保留选定的文件或文件夹，可在拖动时按下 Ctrl 键，此时光标右下方的小方框中将出现“＋”号。

如果用户将文件或文件夹拖至不同驱动器上的另一文件，则该操作为复制(图 4-24)。如果此时希望将选定文件或文件夹移动到另一驱动器上的指定文件夹中，可在选定要移动的文件或文件夹后，单击“剪切”按钮，然后单击目标文件夹并单击“粘贴”按钮。

图 4-24　利用拖动方法复制文件夹

3.选定多个文件夹或文件

在具体操作时，用户可能经常需要删除、复制或移动多个关和文件，此时就需要首先选定这些文件夹和文件。

要选定一组连续的文件夹和文件，可首先在资源管理器右窗格中单击第一个要选定的文件夹或文件，然后按下 Shift 键单击要选定的最后一个文件夹或文件。如果按下 Ctrl 键单击文件夹或文件，则可选定或取消选任何离散的文件夹或文件。例如，在图 4-25 中，用户可首先单击 Computer.jpg 选定第一个文件，然后按下 Shift 键单击“茶杯 1.tif”，按下来按下 Ctrl 键单击“2.jpg”和“地球仪.tif”。

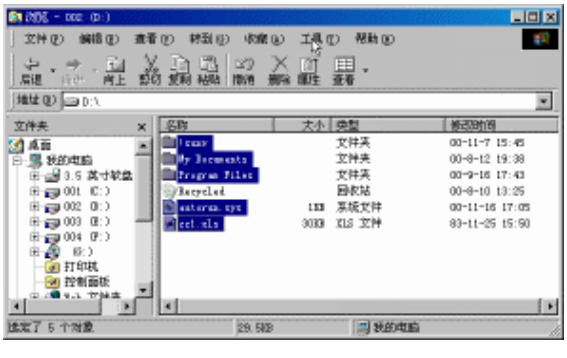

图 4-25　选定多个文件夹或文件

要选定某个文件夹中的所有内容 ，可首先在左窗格中单击该文件夹，然后单击“编辑”/“选定”菜单。

要选定当前文件夹中已选定的文件或文件夹以外的其他文件夹或文件 ，可选择“编辑”/“反向选择”菜单。

4.改变文件夹列表

通过选择 “查看”菜单中的“大图标”、“小图标”、“列表”或“详细资料”，可改变资源管理器中文件的列表方式。

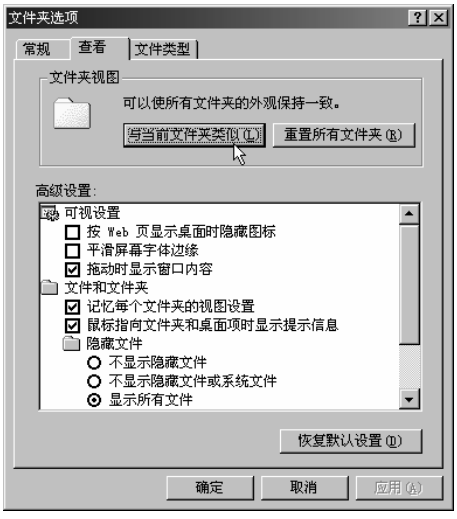

图 4-26　调整隐藏文件和扩展名显示

缺省情况下 ，在资源管理器中不显示隐藏文件和系统文件。如果希望显示所有文件，可选择“查看”/“文件夹选项”菜单打开“文件夹选项”对话框，然后选择“查看”项卡中“隐藏文件”设置区下的“显示所有文件”单选钮(图 4-26)。

此时，缺省情况下，已知文件类型的扩展名被隐藏，只是在资源管理器的“类型”列显示该文件的类型。如果希望显示所有文件的扩展名，可在图 4-26 中取消“隐藏已知文件类型的扩展名”复选框。

4.4.4　系统日常设置及操作

用户在使用 Windows98 进行工作时，经常需要对系统进行设置。例如，更改显示模式、安装字体等。此外，本节还交要向 读者介绍一些基本操作，如格式化软盘的方法、创建桌面快捷方式的方法等。

1.更改显示分辨率和颜色

在用户进行工作时 ，为了增强显示效果，经常需要调整颜色及分辨率设置。例如，为了使图片显示效果更逼真，可能需要真彩色显示；为了使屏幕上能够显示尽可能多的内容，可能需要调整显示分辨率。同时，由于目前显示卡性能指标(主要是显示内存)的提高和屏幕尺寸的增加(如 15 英寸和 17 英寸显示器价格的大幅降价)，也为显示效果的改进提供了条件。

(1)分辨率和颜色基本常识

我们知道，电脑屏幕实际上是由一个个的点组成的，这被称为显示器的分辨率。例如，假定当前显示分辨率为 640×480，则屏幕上共有 307200 个点。实际上，电脑的显示分辨率是由显示器和显示卡共同决定的。也就是说，要将当前显示分辨率设置为某种模式(如 1280×1023)，要看显示器和显示卡是否都支持这种模式。而具体设置分辨率的方法则是通过设置卡参数来实现的。

电脑所能显示的色通常有 16 色、256 色、64K 颜色和真彩色模式等。具体面言，电脑所能显示的颜色数取决于系统为每个点安排的存储空间大小。例如，假定每个点用 4 位(位是计算机中的一个特定概念，它相当于一个器件的两个状态，因此，它只有两个值，即开和关或 0 和 1)。此外，为了便于系统管理，每 8 位被称为一个字节，每 210=1024 字节被称为 1K 字节，每 220=1024K 字节被称为 IM(兆)字节)，则共有 24=16 种组合。

也就是说，当前屏幕上只能显示 16 种颜色，如果分辨率为 640×480，则共需(640×480)/2=153600 字节=150K 字节显示内存。我们知道，大自然中的颜色数目基本上是无限的，因此，无论每个点占多少位，它总是有限的。但是，当利用 3 个字节表示一个点时，每个点可显示的颜色种类为 224=1677216(约 168 万)，因此，人们将这种显示模式称为真彩色。由于目前的显示卡都配有 4M 或更高的显示内存，因此，大部分显示卡都支持 1280×1024，且颜色模式为真彩色的显示模式(此时需 1280×3=3840K 显示内存)。

说明：和分辨率不同的是，由于显示的颜色信号为模拟信号，因此，计算机所能显示的颜色数完全取决于显示卡。

(2)更改显示模式的方法

要更改显示模式，可在桌面任意空白处单击鼠标右键打开桌面快捷菜单，并从中选择“属性”，此时系统将打开“显示 属性”对话框。

单击“设置”选项卡，此时“显示 属性”对话框将如图 4-27 所示。要更改颜色显示模式，可单击“颜色”区中的向下按钮打开颜色模式下拉列表，然后从中进行选择。要调整分辨率，可拖动“屏幕区域”区中的滑块左右移动。

图 4-27 “显示属性”对话框“设置”选项卡

说明：尽管 Windows98 支持对即插即用设备自动安装驱动程序，但对于很多低价显示卡而言，系统均不能自动识别其类型，从而导致用户无法调整显示分辨率和颜色模式。为此用

户需手工安装该显示卡的启动程序，具体安装步骤可参考显示卡使用说明书。

2.调整桌面显示背景图案

如果用户认为桌面太单调的话，可在“显示 属性”对话框中选择“背景”选项卡，然后从对话框左下方的图片列表中选择合适的图或 HTML 文档，并从“显示”下拉列表中选择合适的显示方式(其中，“居中”表示按图片原尺寸将图片放在屏幕中央“平铺”表示按图片原尺寸安排一幅或多幅图片，使之充满屏幕；“拉伸”表示将图片照屏幕尺寸进行拉伸，使之充满屏幕)，此时用户可通过对话框上方的显示器图案看到显示效果(图 4-28)。

图 4-28 设置桌面背景图案

说明 ：要选择其他图片，可单击“浏览”按钮。

3.设置屏幕保护程序

用户可能经常碰到这样的情况 ，因其他事情需暂时离开，但由于不希望其他人员看见当前所进行的工作，此时便可借助屏幕保护程序来暂时更改屏幕显示了。

要设置屏幕保护程序，可在“显示 属性”对话框中选择“屏幕保护程序”选项卡，然后可从中选择所需屏幕保护程序并设置相应的参数，如是否需要密码、启动屏幕保护程序的等待时间等(图 4-29)。

图 4-29 设置屏幕保护程序

说明：所谓等待时间表示，当没有按键或鼠标操作时，需等待多少时间启动屏幕保护程序。如果未设置密码，则按任意键或移动鼠标均可终止屏幕保护程序。否则，需键入密码才可终止屏幕保护程序。

此外，利用“显示 属性”对话框中的“外观”和“效果”选项卡，用户还可设置桌面、非活动窗口、消息框用其组成元素所用字体、尺寸、颜色等，在此就不一一说明了。

4.利用电源管理功能管理显示器、硬盘等

利用 Windows98 提供的电源管理功能，用户可确定经过多长时间，由系统自动关闭主机、显示器或硬盘。

图 4-30 设置电源管理功能

要设置电源管理功能，用户可在控制面板窗口双击“电源管理”图标，此时系统将打开图 4-30 所示对话框。单击选定项目右侧的向下按钮，从下拉列表中选择适当选项即可进行相应设置。

5.创建桌面快捷方式以方便操作

用户在日常操作时，可能经常使用某个程序。如果每次都通过单击“开始”按钮，然后单击“开始/程序”菜中的适当选项，就显得有些繁琐。为此，用户可为某些常用的菜单项和程序创建桌面快捷方式。这样一来，要启动该程序，只需双击桌面上的快捷图标即可。所谓快捷方式，实际上是一个指针，它指向创建该快捷方式的菜单或程序项。因此，删除快捷方式并不会影响原程序。

要为“开始”菜单中的菜单项创建快捷方式，可首先让光标移到“开始”按钮并单击鼠标右键，然后从打开的快捷菜单中选择“打开”，此时系统将打开图 4-31 所示窗口。

Start Menu 窗口实际上对应了“开始”菜单。双击 Programs 图标，可进一步打开如图 4-31 所示的 Programs 窗口(对应“程序”菜单)。如果希望将某个菜单项放到桌面，可直接将其拖动到桌面即可。不过，此菜单项将不会再出现在“程序”菜单中。否则，如果希望使选定菜单项同时存在于“程序”菜单和桌面上，可在按下【Ctrl】键后将选定项目拖至桌面。此外，

用户也可右击选定菜单项，然后从打开的快捷菜单中选择“创建快捷方式”，为其创建一个快捷方式，然后将该快捷方式拖至桌面。

图 4-31　开始菜单窗口

说明：某些程序(如 AutoCAD2000、WPS2000 等)在安装时会同时创建桌面快捷方式和菜单项。

6.安装字体以扩充字库

由于 Windows 98 系统本身只提供了最基本的宋、仿、楷、黑等字体。因此，用户在对文档进行编排时，经常会感到很不方便。

为此，可双击控制面板中的“字体”图标打开“字体”窗口，然后单击“文件|安装新字体”(图 4-32)。

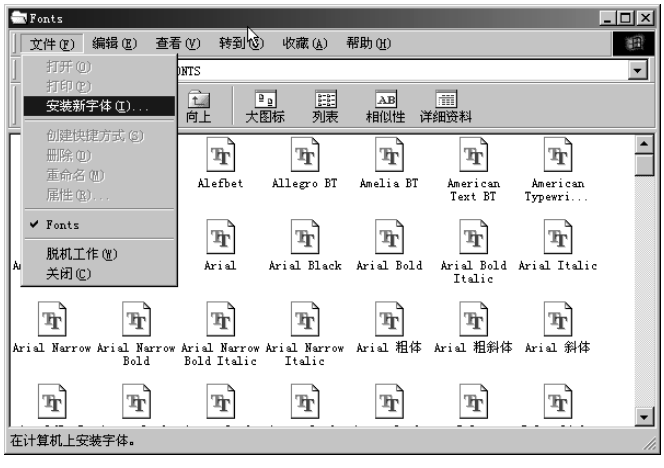

图 4-32　安装新字体

此时系统将打开图 4-33 所示“添加字体”对话框，单击“驱动器”下拉列表选中要安装的字体文件所在的驱动器，然后双击字体文件所在文件夹。此时系统将自动搜索能够安装的字体，并对其进行列表(图 4-33)。

同样，用户可按照前面介绍的选择一组和文件夹的方法，配合【Shift】和【Ctrl】键选择一组要安装的字体。要选择所有字体，可单击“全选”按钮。选定所要安装的字体后，单击“确定”按钮即可将其安装到系统中。

图 4-33 “添加字体”对话框

7.软盘格式化和复制方法

用户在使用软盘时，为了清除软盘上的数据和检验软盘的质量，可能需要格式化软盘。为此，可在“我的电脑”窗口或资源管理器中右击软盘符号，然后从弹出的快捷菜单中选择“格式化”，此时系统将打开“格式化”对话框(图 4-34)。

图 4-34 格式化软盘

在 “格式化”对话框中，可选择格式化类型以及设置卷标等。其中，“快速”表示仅初始化软盘的目录和文件分配表；“全面”表示彻底格式化软盘；“仅复制系统文件”表示仅将少量系统文件传送到软盘，使之成为可引导的系统盘。

说明 ：用户也可使用此种方法格式化硬盘。不过，由于硬盘上的数据通常非常重要，故要特别小心。

显示 ，要复制软盘可在右击软盘打开的快捷菜单中选择“复制软盘”。

8.程序的安装、启动和删除

在 Windows 98 中，程序的安装、启动和删除方法非常简单。本节就来简单介绍一下这方面的内容。

(1)安装程序

如前所述 ，对于某些具有安装程序的软件而言，当用户将相应的 CD 盘插入光驱后，系统会自动启动运行程序，用户只需简单击安装画面中的“安装”按钮即可。对于不具自动安装功能的软件，其安装方法大致有三种，一是利用资源管理器，即在资源管理器对软件所在软盘或光盘进行列表，找到其安装程序并双击之，此后只需按提示回答问题即可(图 4-35)。

图 4-35　利用资源管理器安装软件

安装软件的第二种方法是使用 “运行”对话框。首先单击“开始”按钮打开“开始”菜单，从中选择“运行”打开“运行”对话框(图 4-36)。在该对话框中单击“浏览”按钮打开“浏览”对话框，打开软件所在文件夹，选中安装程序并单击“打开”按钮(图 4-37)，则“运行”对话框将如图 4-38 所示。在运行对话框中单击“确定”按钮，即可开始安装软件了。

图 4-36　利用“运行”对话框安装软件

图 4-37　利用“浏览”对话框选定安装程序

图 4-38　选定安装程序后的“运行”对话框

(2)启动程序

要启动软件 ，最简单的方法莫过于直接使用“开始”菜单了，将光标移至选定的菜单项并单击，即可启动相应程序。

说明 ：大多数软件安装后都创建了一个程序组，即其中包含了多个程序项，如主程序、帮助、卸载程序项和其他子程序等。

(3)删除程序

如果用户希望从硬盘上删除不需要的程序项 ，通常不能使用直接将该软件所在的文件夹删除的方法来进行。这是由于，用户在安装软件时并非所有程序都安装至某一个文件夹，而

是部分文件被安装到了其他目录。此外，安装软件还涉及到应用程序向 Windows 98 注册，以及向“开始”菜单或“程序”菜单项的问题。

因此，如果某一软件的子菜单项中已有“卸载 XXXX”或“Uninstall XXXX”，则可直接选中该菜单项，则可在“添加/删除程序属性”对话框中选中所要制卸载的程序项后(在控制面板窗口中双击“添加/删除程序”图标可打开该对话框)，单击该对话框右下角的“添加/删除”按钮(图 4-39)。

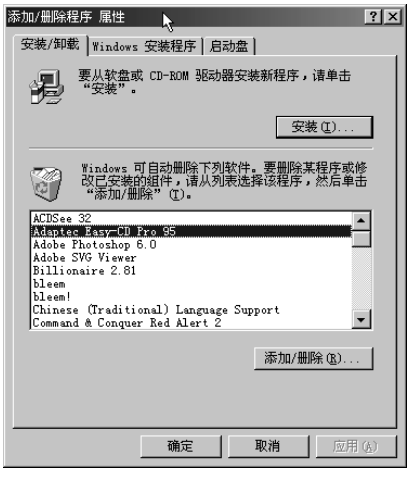

图 4-39 利用“添加/删除程序属性”对话框删除程序

如果某一软件的子菜单项中没有 “卸载 XXXX”或“Uninstall XXXX”菜单项，而且该程序项也未出现在“添加/删除程序属性”对话框中，则用户只有分别删除该软件所在目录，并手工调整“开始”菜单或“程序”菜单了。

9.查找文件、文件夹、计算机或用户

用户操作电脑时，经常会忘记了自己将文件放在哪个文件夹中，或者文件夹位于哪个驱动器上，此时便可借助 Windows 98 提供的查找工具来进行搜索。

(1)查找文件和文件夹

查找文件和文件夹的步骤如下：

1 单击“开始”按钮打开“开始”菜单，将光标移至“查找”打开“查找”子菜单。

2 从“查找”菜单中选择“文件或文件夹”打开“查找”对话框(图 4-40)。

图 4-40 “查找”对话框

3 在“名称”编辑框中输入文件或文件夹名称，然后单击“开始查找”按钮，则结果将如图 4-41 所示。

图 4-41 输入文件或文件夹名称后开始查找

图 4-42 选定要搜索的驱动器

4 如果用户希望调整待搜索的驱动器，可单击“搜索”下拉框，从中进行选择(图 4-42)。如选择“我的电脑”，则表示搜索全部驱动器。

5 如果想搜索在指定日期创建或修改的文件或文件夹，可选择“修改日期”选项卡选中“找出所有已创建的或已修改的文件”及下面适当的单选框(图 4-43)。例如，在本例中表示仅搜索创建或修改时间在 2000 年 10 月 19 日和 2001 年 1 月 17 日之间的文件或文件夹。

6 如果想搜索指定类型的文件或文件夹，可选择“高级”选项卡(图 4-44)。其中，“所属类型”下拉列表用于设置文件类型，“大小”区用于设置搜索文件的尺寸。

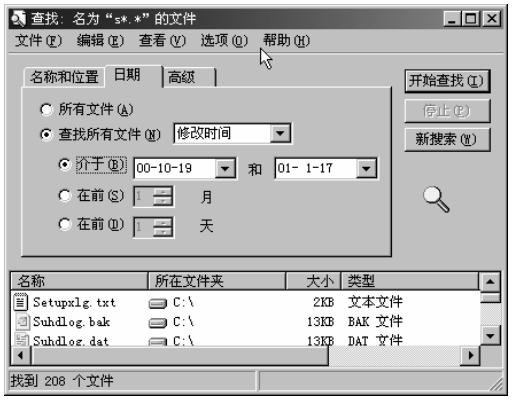

图 4-43 设置搜索指定类型的文件或文件夹

图 4-44 设置搜索指定类型的文件夹

说明 ：这三个选项卡中的条件为逻辑“与”的关系，例如，按照本例设置的条件，表示搜索创建或修改时间在 2000 年 10 月 19 和 2001 年 1 月 17 日之间，大小超过 2K 的文件。要清除搜索条件设置，可单击“新搜索”按钮。

(2)查找网络中的计算机

当本计算机连入某个网络时，可通过选择“开始”|“查找”|“计算机”查找指定计算机，此时系统将显示图 4-45 所示对话框。

图 4-45 查找网络中的计算机

说明：如果知道要搜索的共享文件夹的路径，可同时指定计算机和文件夹的名称，例如 \\marketing\peports.

(3)**查找** Internet **上的** Web **页**

图 4-46 查找 Internet 上的 Web 页

当本计算机连入 Internet **网络时，可通过选择“开始”|“查找”|“在** Internet **上”查找** Internet **上的** Web **页，此时系统将显示图** 4-46 **所示对话框。**

说明：Alta Vista、Infoseek、Lycos **等为搜索引擎，当用键入某些关键词并单击“搜索”按钮后，系统将搜索网中出现该词汇的** Web **页。**Yahoo!、Excite **等为目录站点，这些站点中分类存储了大量的** Web **站点地址、用户可同样通过输入某些关键词来进行搜索。**

(4)**查找网络中的用户**

当本计算机连入局域或 Internet **网络时，可通过选择“开始”|“查找”|“用户”查找网络中的用户，此时系统将显示图** 4-47 **所示对话框。**

图 4-47 查找网络中的用户

单击“查看”下拉列表框，系统将打开一下拉列表，用户可从中选择从何处搜索用户(图 4-48)。

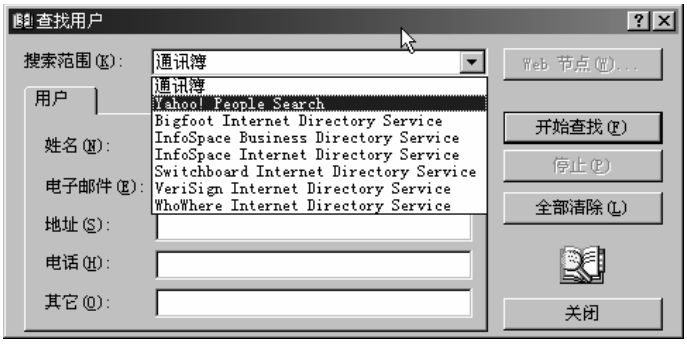

图 4-48　选择供使用的目录服务和通信簿

10.打开最近使用的文档

为了方便用户，Windows 98d "在“开始”|“文档”子菜单中保存有最近所使用的一组文档。此外，在很多应用程序的“文件”菜单中也保存有用户最近使用的若干文档。要打开最近使用的文档，可按如下步骤进行:

(1)单击“开始”按钮，将光标移至“文档”菜单项打开其子菜单(图 4-49)。

(2)单击选定文档，即可打开相应的编辑窗口。例如，在本例中单击最上方的 tu412，则系统将打开画图窗口。

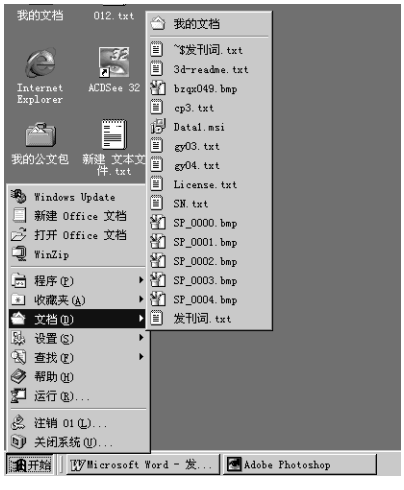

图 4-49　打开“开始”/“文档”菜单

说明：为了便于操作，用户也可将经常使用的文件夹拖至桌面。

11.在 Windows 98 中输入汉字及标点的方法

在 Windows 98 中文简体版中，系统提供了 7 种汉字输入法，它们分别是微软拼音输入法、全拼输入法、郑码输入法、智能 ABC 输入法、表形码输入法、区位输入法和双拼输入法，其中后 3 种输入法未安装。这 7 种输入法各具特色，分别适合不同习惯和水平的用户使用。

(1)启动汉字输入法

安装 Windows 98 后，在 Windows 98 工作环境中可随时使用【Ctrl＋Shift】键启动中文输入法，【Ctrl＋Shift】键还可用于从中文输入方式返回英文输入方式或切换输入法。此外，用户不可利用【Ctrl＋Space】来启动或关闭汉字输入法。

当然，选择汉字输入法最直接的方法还是通过单击任务栏上的输入法图标。此时系统将打开 2－50 所示输入法弹出菜单，单击其中适当的菜单项即可。

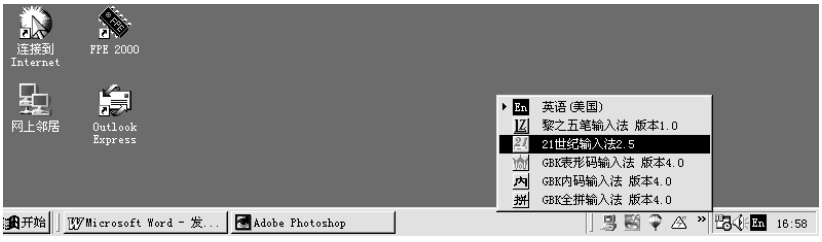

图 4-50 输入法弹出菜单

说明：在 Windows 98 中，汉字输入状态和应用程序是关联的。例如，当用户在桌面状态下选择了拼音输入方式，若转至画图、写字板或其他应用程序，仍需重新选择输入法，在图 4-50 中，五笔输入法需单独安装，Windows 98 不提供该输入法。

输入法启动后，屏幕底部出现输入法提示条(图 4-51)。同时，在提示条输入法名框中会显示当前输入法的名称，其内容可能是“标准”、“拼音”或“郑码”等。

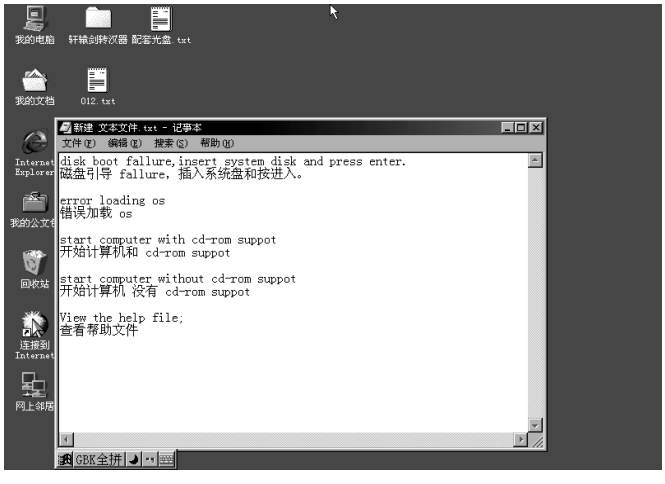

图 4-51 输入法提示条

对于该提示条，当用户将光标移至其边界时，光标将变为花十字，此时按住鼠标左键并拖动光标(简称拖动)即可移动其位置。

在图 4-51 中，如果我们希望“档案”两字，可首先输入“dang”，系统将打开一编码输入框和文字选择框(图 4-52)。按数字键 4(“档”字的序号)或单击该项，均可输入“档”字(图 4-53)。

由于 Windows 98 的拼音输入法带有联想功能(或称为字词功能)，因此，与“档”字有关的词组将被显示出来，输入“1”或单击该项可继续输入“案”字。

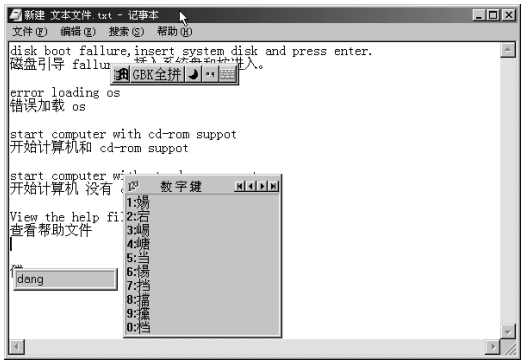

图 4-52　输入“档”字的拼音“dang”

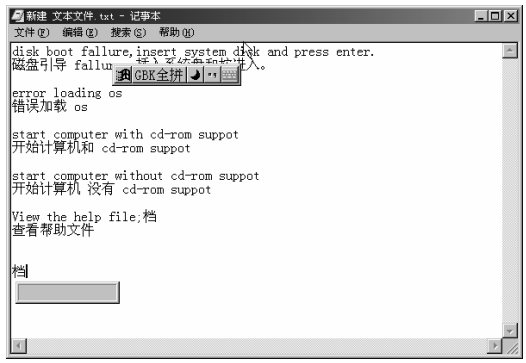

图 4-53　选择“档”字

要移动编码框位置 ，可将光标移至该框(此时光标将变为花十字，然后拖动光标。要移动文字选择框的位置，可将光标移至该框的标题条处(此时光标也呈花十字形状)，然后拖动光标即可。

(2)文字的续选和词组输入

大家都知道 ，虽然拼音输入法具有不必学习即可使用的优点，但这种输入法也具有明显的不足，那就是重码字太多。例如，当用户输入“dang”后，文字选择框中除了有我们所需的“档”字外，还有“当”、“党”、“挡”等等。假定用户要输入“裆”，则需要翻页进行选择(称为续选)。那么，如何进行续选呢。

Windows 98 中，用户可通过两种方法进行续选，一种是单击文字选择框中标题条上的选择当前编码的第一页、选择当前页的上一页、选择当前页的下一页、选择当前编码的最后一页；一种是通过按【－】和【＋】键来前翻页或后翻页。

要输入词组 ，可直接输入该词组的编码。例如，要输入“档案”，可直接键入“dangan”。

说明 ：智能 ABC 输入法同时保留了【[】和【]】键的翻页功能。

(3)输入法提示条的组成

如前面各图所示，输入法提示除了用于显示当前输入法外，还有若干按钮，下面我们分别介绍这几个按钮的含义(图 4-54)。

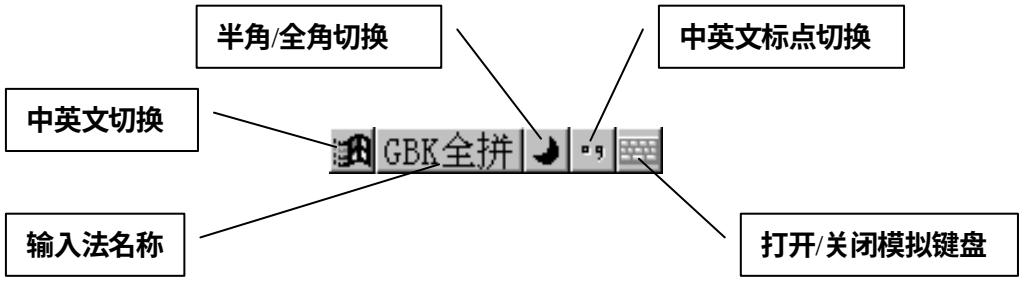

图 4-54　输入法提示条上各按钮的意义

中英文切换按钮 ：单击此按钮，该按钮标识交变为“A”，表示转至英文输入方式。再次单击该按钮 ，可重新切换至汉字输入方式。该按钮在不关闭汉字输入状态的情况下，切换中英文输入方式。

全拼 ：输入法框：表示当前的输入法类型。

全角和半角切换按钮 ：用于切换英文字符的全角和半角状态。通常情况下，如果用户选择字体为宋体、黑体等中文字体，英文字符都为半角形式，即其宽度是变化的。但有时用户希望二者同宽，则因通过单击该按钮转至全角方式。

中英文标点切换按钮：通常情况下，如果用户创建的是中文文档或中英文混合文档，可选用全身标点，即每个标点符号均占用一个字宽(称为全身标点)。如果用户编制的是纯英文文档或包含了英文段落，则可切换至英文标点。

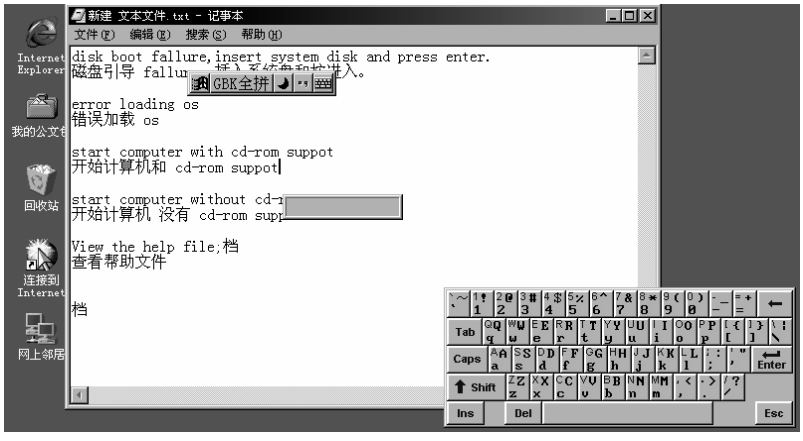

图 4-55 利用模拟键盘输入文字

说明 ：如果切换至英文标点方式，按【.】键时输入的不再是句号“。”而是“.”按【‘】键输入的也不再是单引号或双引号(同时按下【Shift】键)。

模拟键 (又称软键盘或动态键盘)开关按钮：当用户单击该按钮时，系统将打开模拟键盘(图 4-55)。用户可通过鼠标单击模拟键盘上的按键来输入汉字或特定字符，以及制表符、回车键、删除键等，其用法和真实键盘完全相同。此外，用户还要通过从右击图标打开的快捷菜单中选择适当的菜单项，选择要打开的模拟键盘类型(如标点符号、数字序号等)。

(4)删除输入编码

如果在输入过程中想取消已输入的编码时 ，可以按【Esc】键重新开始。如果想逐个取消已输入的编码，可按退格键。

(5)输入数字和中文标点符号

如果用户希望在中文输入状态下输入数字 ，可在编码输入框为空时直接按数字键。如编码框不空，应首先使用退格键或【Esc】键清除编码框。

在中文 Windows98 环境下，用户可以非常方便地使用中文标点符号。当输入法设置了中文标点符号有效时，即可用英文键直接输入中文标点符号。

(6)手工造词

用用户经常要用到某些词组 ，而该词组并未在词库中，则用户可通过手工造词加入到词库中。

要进行手工造词，可右击输入法提示条，然后从弹出的快捷菜单中选择“手工造词”，此时系统将打开图 4-56 所示对话框。

图 4-56 “手工造词”对话框

图 4-57 将手工造词添加至词库中

在“词语”框中输入词组内容，在“外码”框中输入该词组的编码，然后单击“添加”按钮即可将其添加至词库中(图 4-57)。**单击“关闭”按钮，该词组即生效**。

要修改或删除手工造词，可在该对话框上方选择“维护”单选框，然后单击“删除”或“修改”按钮。(**图** 4-58)。

(7)**安装外部输入法**

对于系统未包括的输入法，如五笔字型输入法，可通过运行其安装程序来安装它们。

图 4-58 词库维护

第二部分　VB 6.0 编程基础

第一章　Visual Basic 6.0 初接触

Visual Basic 是 Microsoft 公司开发的一系列编写 Windows 应用程序的软件。它以其制作过程简洁明了，实现功能十分强大而著称于世。这次 Visual Basic 6.0 的发行，带来了比 Visual Basic 5.0 更完善的功能，使初学者极易上手。不仅能编写一些简单的小程序，还能用 VB 做一些更深入的工作，VB 的功能是非常强大的，假如你能设想一个程序任务，我们就能用 VB 编程来实现它。

如你所想，在你成功掌握 VB6.0 之前有大量的东西需要你学习，但是一旦掌握了 VB 的基础，会发现你立即能够举一反三。

1.1　版本简介

- Visual Basic 学习版使编程人员可轻松开发 Win32 应用程序。该版本包括所有的内部控件以及网格、选项卡和数据绑定控件。学习版提供的文档有 Learn VB Now CD 和包含全部联机文档的 Microsoft Developer Network CD。
- 专业版为专业编程人员提供了一整套功能完备的开发工具。该版本包括学习版的全部功能以及 ActiveX 控件、Internet Information ServerApplication Designer、集成的 Visual Database Tools 和 DataEnvironment、Active Data Objects 和 Dynamic HTML Page Designer。专业版提供的文档有 Visual Studio Professional Features 手册和包含全部联机文档的 Microsoft Developer Network CD。
- 企业版使得专业编程人员能够开发功能强大的组内分布式应用程序。该版本包括专业版的全部功能以及 Back Office 工具，例如 SQL Server、Microsoft Transaction Server、Internet Information Server、VisualSourceSafe、SNA Server 等。企业版包括的印刷文档有 Visual StudioEnterprise Features 手册以及包含全部联机文档的 Microsoft DeveloperNetwork CD。

1.2　安装

首先介绍一下安装 Visual Basic 6.0 的软、硬件配置需求。

- Windows95/98 或 Windows NT Workstation 或 Windows NT Server 3.0 上述产品或更高版。
- 486DX/66 MHz 或更高的处理器，或任何运行于 Microsoft Windows NT Workstation 的 Alpha 处理器。

- 配置一个 CD-ROM 驱动器和鼠标。
- Windows 95 下需 16 MB RAM ，Windows NTWorkstation 下需 32 MB RAM。
- Microsoft Windows 支持的 VGA 或分辨率更高的显示器。

硬盘空间需求：

- 标准版——典型安装 48MB，完全安装要 80MB。
- 个人版——典型安装 48MB，完全安装要 80MB。
- 企业版——典型安装 128 MB，完全安装要 147 MB。
- 附加内容——MCDN 需 67 MB，IE4.0 需 66 MB。

从光盘上安装 Visual Basic 6.0 ，请执行以下步骤：

1. 在 CD-ROM 驱动器中插入光盘。
2. 在光盘根目录下运行 Setup.exe 或 Install.exe 。如果您的计算机能够在系统中运行 AutoPlay，则在插入光盘时，安装程序将被自动加载。
3. 选取“安装 Visual Basic 6.0”。
4. 选取按何种方式安装（典型，最小，自定义三种方式。），以及安装目录。

如果想要卸载 Visual Basic 6.0 可再次运行 Setup.exe 此时会出现如图 1-1 所示对话框选择“添加/删除”项，再选择 Visual Basic 6.0 ，单击下一步，即可卸载 Visual Basic 6.0。

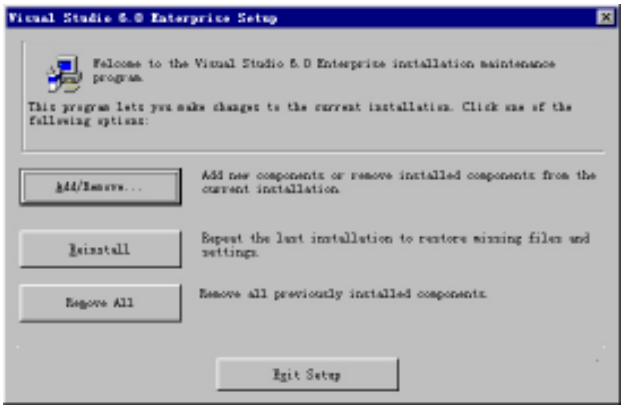

图 1-1 “添加/删除”对话框

1.3 关键概念

为了理解应用程序的开发，先要理解 Visual Basic 编程的一些关键概念。因为 Visual Basic 是开发 Win32 应用程序的语言，所以有必要与 Windows 环境保持一定的相似性。如果不熟悉 Windows 编程，那么读者就必须明白 Windows 编程和在其它环境下编程的一些根本性的差别。

Windows 的工作方式用最简单的语言来概括就是三个关键的概念：窗口、事件和消息。Windows 操作系统在管理所有的窗口时要给每一个窗口指定一个惟一的标识号（窗口句柄或 hWnd），通过它来连续地监视每一个窗口的活动或事件的信号。事件可以通过诸如单击鼠标

或按下键盘按键的操作而产生，也可以通过程序产生，甚至可以由操作另一个窗口产生。每发生一次事件，程序将引发一条消息送至操作系统。操作系统处理该消息并发送给其它窗口。然后，每一个窗口才能根据自身处理该条消息的指令而采取适当的操作。

Visual Basic 可以使您摆脱处理所有的底层消息（许多消息由 Visual Basic 自动处理。）的麻烦，您可以专心处理其它一些消息事件过程，这样可以快速创建强大的应用程序而毋需处理不必要的细节。

在传统的应用程序中，应用程序自身控制了执行哪一部分代码和按何种顺序执行代码。从第一行代码执行程序并按应用程序中预定的路径执行，必要时调用过程。而在事件驱动的应用程序中，代码不是按照预定的路径执行，而是在响应不同的事件时执行不同的代码段。事件可以由用户操作触发，也可以由来自操作系统或其它应用程序的消息触发，甚至由应用程序本身的消息触发。这些事件的顺序决定了代码执行的顺序，因此应用程序每次运行时所经过的代码的路径都是不同的。因为事件的顺序是无法预测的，所以在代码中必须对执行时的“各种状态”作一定的假设。当作出某些假设时（例如，假设在运行处理某一输入字段的过程之前，该输入字段必须包含确定的值），应该组织好应用程序的结构，以确保该假设始终有效（例如，在输入字段中有值之前禁止使用启动该处理过程的命令按钮）。在执行中代码也可以触发事件。例如，在程序中改变文本框中的文本将引发文本框的 Change 事件。如果 Change 事件中包含有代码，则将导致该代码的执行。如果原来假设该事件仅能由用户的交互操作所触发，则可能会产生意料之外的结果。正因为这一原因，所以在设计应用程序时理解事件驱动模型并牢记在心是非常重要的。

传统的应用程序开发过程可以分为三个明显的步骤：编码、编译和测试代码。但是 Visual Basic 与传统的语言不同，它使用交互式方法开发应用程序，使三个步骤之间不再有明显的界限。在大多数语言里，如果编写代码时发生了错误，则在开始编译应用程序时该错误就会被编译器捕获。此时必须查找并改正该错误，然后再次进行编译，对每一个发现的错误都要重复这样的过程。Visual Basic 在编程者输入代码时便进行解释，即时捕获并突出显示大多数语法或拼写错误。看起来就像一位专家在监视代码的输入。除即时捕获错误以外，Visual Basic 也在输入代码时部分的编译该代码。当准备运行和测试应用程序时，只需极短时间即可完成编译。如果编译器发现了错误，则将错误突出显示于代码中。这时可以更正错误并继续编译，而不需从头开始。由于 Visual Basic 的交互特性，因此可以发现在开发应用程序时，您自己正频繁地运行着您的应用程序。通过这种方式，代码运行的效果可以在开发时进行测试，而不必等到编译完成以后。

1.4　集成开发环境介绍

Visual Basic 的开发环境常常是指集成开发环境或 IDE，这是因为它在一个公共环境里集成了许多不同的功能，例如，设计、编辑、编译和调试。在大多传统开发工具中，每个功能都是以一个独立的程序运行，并都有自己的界面，如图 1-2 所示。

第一次运行 Visual Basic 6.0 后，我们将看到 Visual Basic 6.0 的集成开发环境如下。主要由以下几个部分组成：

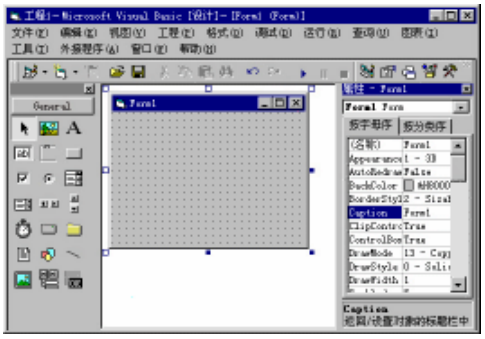

图 1-2　Visual Basic 6.0 集成开发环境

1.4.1　菜单条

它显示了所使用 Visual Basic 6.0 的命令如图 1-3 所示：

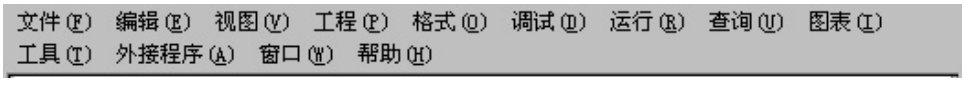

图 1-3　Visual Basic 6.0 菜单条

1.4.2　工具栏

在 IDE 下提供对常用命令的快速访问。单击工具栏上的按钮，则执行该按钮所代表的操作。按照缺省规定，启动 Visual Basic 之后，显示的是“标准”工具栏。附加的编辑、窗体设计和调试的工具栏可以从“视图”菜单上的“工具栏”命令中添加或删除如图 1-4 所示。

图 1-4　Visual Basic 6.0 工具栏

1.4.3　工具箱

提供一组工具，用于设计时在窗体中放置控件。除了缺省的工具箱布局之外，还可以通过单击“添加选项卡”并在结果选项卡中添加控件来创建自定义布局如图 1-5 所示。

图 1-5　Visual Basic 6.0 工具箱

1.4.4　工程管理器窗口

列出当前工程中的窗体和模块。工程是指用于创建一个应用程序文件的集合如图 1-6。

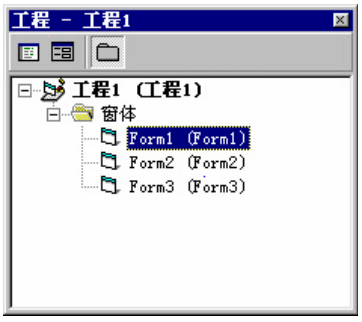

图 1-6　Visual Basic 6.0 **工程管理器窗口**

1.4.5　属性窗口

列出对选定窗体和控件的属性设置值。属性是指对象的特征，如大小、标题或颜色。如图 1-7 **所示**。

图 1-7　Visual Basic 6.0 **属性窗口**

1.4.6　对象浏览器

列出工程中有效的对象，并提供在代码中漫游的快速方法。可以使用“对象浏览器”浏览在 Visual Basic **中的对象和其它应用程序，查看对那些对象有效的方法和属性，并将代码过程粘贴进自己的应用程序如图** 1-8 **所示**。

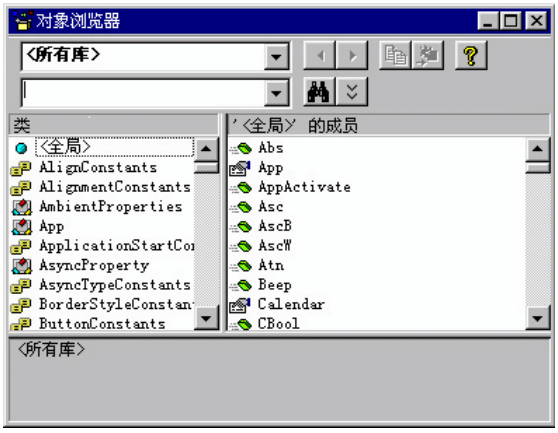

图 1-8　Visual Basic 6.0 **对象浏览器**

1.4.7　对象窗口

作为自定义窗口用来设计应用程序的界面。在窗体中添加控件、图形和图片来创建所希望的外观。应用程序中每一个窗体都有自己的窗体设计器窗口如图 1-9 **所示。**

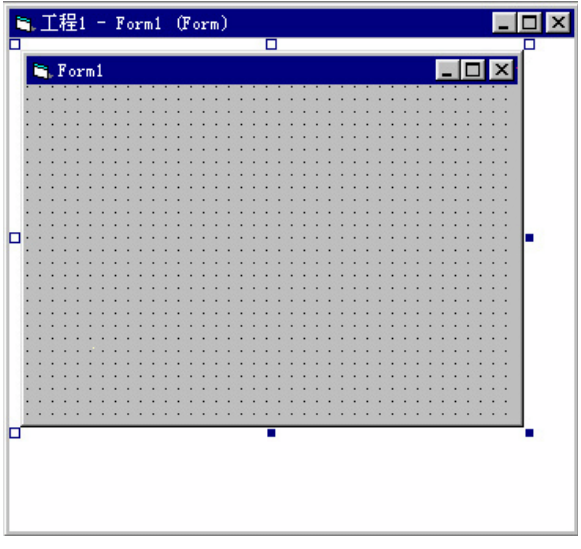

图 1-9　Visual Basic 6.0 **窗体设计器**

1.4.8　代码窗口

是输入应用程序代码的编辑器。应用程序的每个窗体或代码模块都有一个单独的代码编辑器窗口。如图 1-10 **所示。**

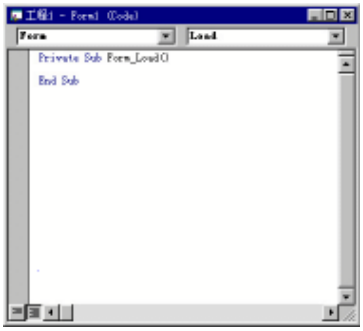

图 1-10　Visual Basic 6.0 **代码编辑器窗口**

1.4.9　窗体布局窗口

Form Layout window（**图** 1-11）**允许使用表示屏幕的小图像来布置应用程序中各窗体的位置**。

图 1-11　Visual Basic 6.0 **窗体布局窗口**

1.4.10　立即、本地和监视窗口

这些附加窗口是为调试应用程序提供的。它们只在 IDE **之中运行应用程序时才有效**。

图 1-12　Visual Basic 6.0 **立即、本地和监视窗口**

1.5　本章小结

以上我们介绍了 Visual Basic 6.0 **一些有关编程环境、版本信息、软硬件需求等基础知识，下面我们会继续介绍有关** Visual Basic 6.0 **的一些编程知识**。

第二章　设计第一个窗体

设计任何一个 Visual Basic 6.0 应用程序，都应从设计窗体开始，窗体就是一个窗口，它构成应用程序可视部分的背景。大多数应用程序都在窗体上放置控件对象，如命令按钮和文本框等，通过设置窗体和控件的属性，可以得到令人满意的用户界面。为窗体和控件编写了代码以后，它们就可以对程序运行时发生的事件作出响应。这一章将教会大家如何在运行中改变对象的属性。

本章主要内容：

- 建立满意的窗体
- 添加合适的控件
- 设置对象属性得到美观实用的界面
- 使用代码窗口
- 运行中改变对象属性

2.1　窗体的建立

建立窗体是编写 Visual Basic 6.0 应用程序的基础，这一节大家将学会如何建立自己满意的窗体。首先，向大家介绍一下图形用户界面的概念。

2.1.1　GUI 介绍

计算机发展到 70 年代，计算机上连接了显示器，称之为终端。与以前相比，带有终端的计算机，用户可以直接将命令输入计算机，而计算机也可以立即作出回答，这使用户方便了很多。

最初的用户界面是由一个黑白屏幕和一个叫做光标的闪烁点组成。为使计算机处理问题，必须输入计算机能够识别的命令，否则，如果输入不正确的命令或按了不恰当的键计算机将拒绝操作。这种简单的界面称为命令行界面，在这种界面下工作，很快就会感到使用计算机是一件很慢，很令人厌烦的事。

为了使计算机变得更友好，编程时可以选择菜单中列出的有用命令，这样人们就不必担心命令是否被正确输入，只要用键盘或鼠标进行选择就可以了。

几乎每个程序都提供菜单，不幸的是它们绝大部分各不相同，这使大多数人陷入了困境。为了方便用户，使计算机变得真正友好，程序员们定义了一套供所有程序工作的标准方法。为了代替在屏幕上枯燥无味的命令符号，这新的标准就是使用颜色菜单和图形等方法。现在，人们可以通过使用屏幕上提示的图和图标，选择要用的命令。

这些新的标准被称为是图形用户界面(Graphical User Interface，简称 GUI)。由于 GUI 的设计者不同，自然，每一个 GUI 都略有差别。

2.1.2 空白窗体

窗体是 Visual Basic 6.0 最基本的核心，若想用 Visual Basic 6.0 编程，首先要创建一个 Visual Basic 6.0 的空白窗体。启动 Visual Basic 6.0 应用程序，Visual Basic 6.0 就会自动创建一个空白窗体(Form)，如图 2-1 所示。

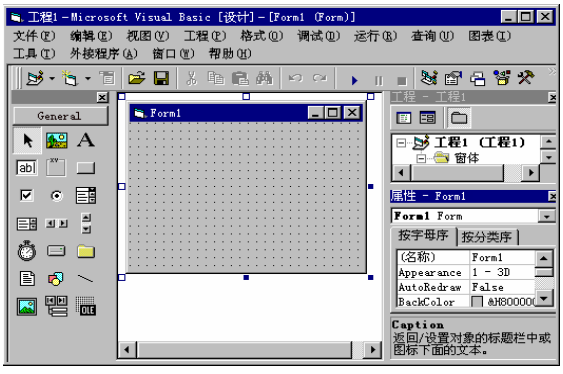

图 2-1 新建窗体

该窗体的标题栏中有“Form1”的字样，在 Visual Basic 6.0 术语中，Form1 是这个窗体的标题(Caption)属性的内容，就像人的名字一样，无论何时启动 Visual Basic 6.0，Form1 都将是这个窗体的缺省标题。如果一个项目包含了多个窗体，则第一个窗体以后建立的窗体标题的数字进行按序排列，如 Form2、 Form3、 Form4 等，这在以后关于多窗体的章节中我们将会看到。

窗体标题栏中右边的三个按钮分别是最小化按钮  ，最大化按钮  和关闭按钮  。这几个按钮在第一次运行 Visual Basic 6.0 程序时，大家都可以试着去用，其功能很容易掌握，与 Windows98 操作系统下的相应按钮功能相同。

单击  按钮时，窗体便布满整个屏幕，同时最大化按钮变成了恢复按钮  ，单击恢复按钮，窗体又会恢复原来的正常尺寸。

单击  按钮时，窗体就会缩小保存在屏幕下方的任务栏中，在任务栏中单击相应窗体又可以将它激活为正常尺寸。

单击  按钮时，就会把窗体关闭。

细心的读者，一定已经注意到窗体的内部由矩阵排列的点组成，它们组成了一个网格。在学习使用控件的时间里，我们会向大家描述如何使用这些点来对齐置于窗体上的控件，以及如何来调整这些点的间距。如果有谁觉得这些点实在让人不舒服时，我们还可以轻易地将它们去掉，但同时仍能保证控件在窗体上排列得整整齐齐。

2.1.3 窗体位置的调整

在 Visual Basic 6.0 中，窗体的位置和大小是很容易调整的。由于控件的数目不同或个人习惯等因素，我们往往希望自己设计的窗体具有令人满意的尺寸，同时也希望在程序运行时，

窗体在屏幕上处于一个比较合适的位置，不至于在程序运行时找不到窗体在哪儿。调整窗体的位置，常用的有两种方法:

一、 使用窗体布局窗口

启动 Visual Basic 6.0，建立一个空白窗体，然后选择“视图”菜单中的“窗体布局窗口”选项。这样屏幕右下角就会出现一个小窗口，如图 2-2 所示，显示出当程序运行时，当前窗体在屏幕上所处的位置。

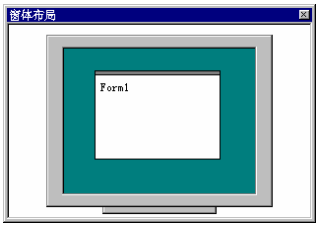

图 2-2 窗体布局窗口

将鼠标移到这个窗口中，放在虚拟显示的空白窗体上，这时鼠标就会改变形状，这表示该窗体可以拖动。按下鼠标左键(不要弹起)，拖动，调整到自己认为满意的地方，然后松开鼠标。

这样，程序运行时，该窗体就会显示在我们设计好的位置上，这样设置出来的位置不够精确。一般情况下，只要要求不高，我们完全可以用这种简单的办法进行设置。

当用户有特殊要求时，要求窗体所处位置比较精确，我们则可以通过设置窗体属性的办法来确定窗体位置。

二、 设置窗体位置属性

任何事物都有它自己的许多特性。例如，一张桌子它的特性包括高度、桌面大小、颜色、材料、新旧等等。在 Visual Basic 6.0 中，我们把类似的一些性质，称为属性，窗体和控件都有自己的一系列属性。它们有许多共同的属性，同时，也有一些只有某个控件才会有的惟一属性，属性的设置和改变是 Visual Basic 6.0 编程中极为重要的一步。

启动 Visual Basic 6.0 程序，建立空白窗体。一般情况下，属性窗口已经自动在屏幕上，若未显示，我们可以选择“视图”菜单中的“属性窗口”选项，这样，属性窗口就会显示在我们刚才启动的“窗体布局窗口”上方，如图 2-3 所示。

图 2-3 打开属性窗口

属性列表分两列，左边一列为窗体属性，右边一列为相应的属性设置值。每列右边的垂直滚动条可以使属性列表上下滚动，滚动属性列表，从中找出 Left 属性和 Top 属性。

Left 属性描述了程序运行时窗体左边到屏幕左边的距离。Visual Basic 6.0 为我们提供了一个缺省值，将光标键入，就可以输入我们认为满意的值。

如果输一个负数，我们会看到当程序运行时，窗体有一部分已处于屏幕之外，试着输入几个值，并且在运行时观察效果，很快，就会掌握这个属性值的范围。

Top 属性与 Left 属性相似，它描述了程序运行时窗体上边距屏幕上边的距离。同样，Visual Basic 6.0 为我们提供了缺省值，试着自己输入几个数并观察运行效果，很快，就会掌握 Top 属性值的范围。

利用这两种方法，对窗体位置进行调整，就可以满足一般用户的需求。

2.1.4 窗体大小的调整

由于种种原因，我们往往要对窗体的大小进行调整，常用调节方法有两种：

一、 拖动控制柄

建立空白窗体后，我们会发现在窗体周围有八个小方块，叫做控制柄。倘若没有，是因为窗体没被选中，在窗体上的任何地方单击，控制柄就会立即出现。

细心的读者一定会发现，这些控制柄中有实心的也有空心的。实心的控制柄，我们可以拖动它来改变窗体的大小，空心的控制柄则不可以拖动。将鼠标移到任一个实心控制柄上，当鼠标变为双箭头时，表示可以被拖动。

按下鼠标左键(不要弹起)，拖动鼠标，窗体大小会随之改变，拖到自己认为满意的位置，放开鼠标，这样就改变了原来窗体的尺寸，如图 2-4 所示。

利用这种方法改变窗体尺寸不够精确，当用户对窗体尺寸要求较高时，我们可以采用设置窗体尺寸属性的办法。

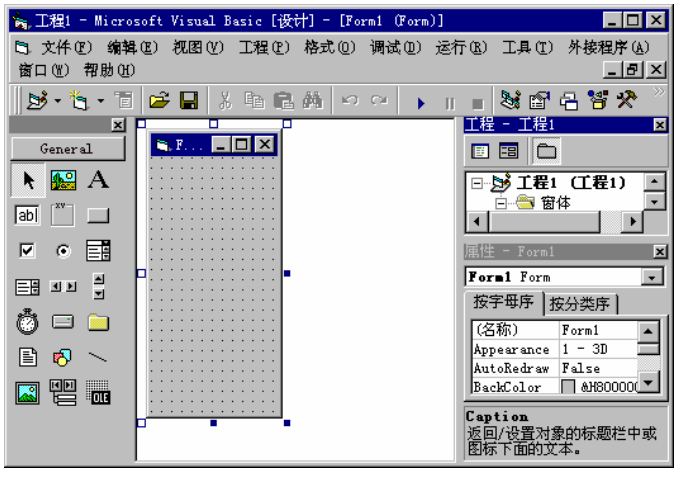

图 2-4 拖动控制柄改变窗体大小

二、 设置窗体的尺寸属性

选择“视图”菜单中的“属性窗口”选项或按下快捷键 F4，**打开属性窗口，滚动属性列表，找到窗体的** Width **属性和** Height **属性**。

Width **属性描述了窗体的宽度，**Height **属性则描述了窗体的高度，**Visual Basic 6.0 **为这两个属性都提供了缺省值，即** Visual Basic 6.0 **为我们自动创建的窗体的大小。在这两个属性中设置自己认为比较满意的值，窗体就会作出相应的改变**。

为了防止窗体被设得太大或太小，这两种属性都有一个数值范围：

Width **属性数值范围：**1620~12000

Height **属性数值范围：**405~9000

现在，我们学会了用两种方法来调节窗体的大小。请记住，上面这些方法，大胆地去试用，很快就会掌握。

2.2 控件的添加

控件是 Visual Basic 6.0 **的用户界面中最重要的组成部分，大概很少有人愿意对一个光秃秃的窗体进行操作，添加了控件的窗体看起来就会舒服些。控件就是用来接收用户输入和显示输出的图形对象**。

2.2.1 工具箱介绍

启动 Visual Basic 6.0，**在屏幕左侧的工具箱中包括了** Visual Basic 6.0 **中最常使用的一些控件，如图** 2-5 **所示**。

图 2-5 Visual Basic 6.0 **工具箱**

现将其名称及大致功能列出，如表 2-1 **所示，使读者对此有一个初步了解**。

表 2-1 工具箱中的控件

对象名称	功能描述
指针	在拖放操作中作为指针显示
图片框(PictureBox)	可以显示来自位图、图标或者源文件的图形

标签(Label)	可以显示用户不能直接改变的文本
文本框(TextBox)	显示设计时用户输入的、或运行时在代码中赋予控件的信息
框架(Frame)	为控件提供可标识的分组
命令按钮(CommandButton)	可以开始、中断或者结束一个进程
复选框(CheckBox)	显示多项选择项，可选择其中的一项或多项
选项按钮(OptionButton)	在选项组中使用，用户只能选择其中的一项
组合框(ComboBox)	将 TextBox 控件和 ListBox 控件的特性结合在一起，既可以在控件的文本框部分输入信息，也可以在控件的列表框部分选择一项
列表框(ListBox)	显示项目列表，从其中可以选择一项或多项
水平滚动条(HscrollBar)	使用滚动条来提供简便的定位，还可以模拟当前所在的位置，滚动条可以作为输入设备，或者速度、数量的指示器来使用(水平显示)
垂直滚动条(VscrollBar)	使用滚动条来提供简便的定位，还可以模拟当前所在的位置，滚动条可以作为输入设备、速度、数量的指示器来使用(垂直显示)
定时器(Timer)	可以有规律地隔一段时间执行一次代码
驱动器列表框(DrivelistBox)	用来显示用户系统中所有有效磁盘驱动器的列表
目录列表框(DirlistBox)	可以显示分层的目录列表
文件列表框(FilelistBox)	用来显示所选择文件类型的文件列表
图形(Shape)	图形控件，显示矩形、正方形、椭圆、圆形、圆角矩形或者圆角正方形
线条(Line)	图形控件，显示水平线、垂直线或者对角线
图像(Image)	如果将 Image 控件绑定到 RemoteData 控件 (RDC) 的保存图像的字段，而该 RDC 使用批处理游标，则 Image 控件将不显示对应的图像
数据(Data)	对存储在数据库中数据的进行访问，允许从一个记录移动到另一个记录，并显示和操纵来自被连接的控件的记录的数据
OLE 控件(OLE)	建立与其它 Microsoft 应用程序的连接

2.2.2 添加控件

在空白的窗体中添加了合适的控件，就形成了 Visual Basic 6.0 中的用户界面。向窗体上添加控件就要用到控件工具箱和窗体编辑器。

控件工具箱

正如前面提到的，工具箱中包括了 Visual Basic 6.0 中最常使用的一些控件，工具箱的名字就是从我们的日常生活中来。因此，使用控件就像平时从工具箱中拿螺丝刀一样简单，只

不过，现在要用鼠标去拿。

窗体编辑器

前面，我们建立的空白窗体就是窗体编辑器的主体，当我们把控件添加到窗体上以后，对控件的布局工作就在这个窗体编辑器中进行。编程过程中，可以选择“视图”菜单中的“对象窗口”命令。

为使用户理解 Visual Basic 6.0，下面这些步骤将创建一个实际的用户界面，可以在相应的图中得到更直观的印象。步骤如下：

1. 在 Windows 中，启动 Visual Basic 6.0，屏幕显示一个名为 Form1 的空白窗体；
2. 用鼠标单击工具箱中的框架(Frame)图标控件；
3. 将鼠标的指针放在窗体中所要放置的地方，按下鼠标左键，拖曳鼠标，使得框架看上去如图 2-6 所示；
4. 在工具箱中单击图片框图标，拖曳鼠标，在框架中增加图片框控件；
5. 在工具箱中单击复选框图标，在框架下方增加一个复选框；
6. 在工具箱中单击标签图标，在窗体右侧添加三个标签控件；
7. 在工具箱中单击文本框图标，在窗体上添加三个文本框与标签对应；
8. 在工具箱中单击命令钮图标，在窗体右下角添加一个命令按钮。

这样，我们就设计好了一个 Visual Basic 6.0 的用户界面，如图 2-7 所示。所有控件都使用了它们的缺省标题，这样的界面看上去未免还有些生硬，当我们通过属性窗口重新设置了其属性后，界面才会变得真正友好起来。

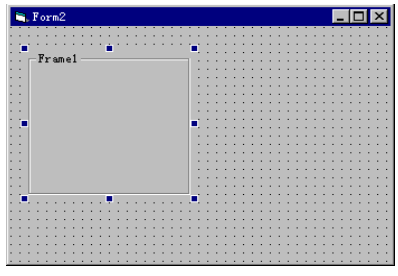

图 2-6　绘制了框架控件

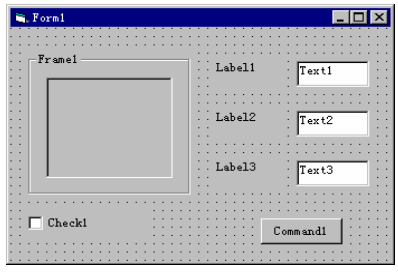

图 2-7　初次绘制的界面

细心的读者可能注意到，在我们刚才绘制控件时，只要操作中稍加注意，那么各个控件

便会排列得整整齐齐，不差分毫。事实上，正是我们前面提到的窗体上的网格帮了大忙。选择“工具”菜单中“选项”命令，然后单击“通用”选项卡，如图 2-8 所示，在此可以进行窗体网格的设置。

2.2.3 窗体网格的设置

依上述方法，便可在“通用”选项卡下找到“窗体网格设置”框架。其中，网格宽度和高度的缺省值都是 120，输入不同的值，窗体网格的间距就会作出相应改变。例如，设置宽度为 400，高度为 50，得到的窗体如图 2-9 所示。在图 2-8 的对话框中，还可以选择是否在窗体上显示网格，以及是否将绘制件对齐到网格。如选择了“对齐控件到网格”复选框，则窗体上的控件将自动向最近的网格对齐。

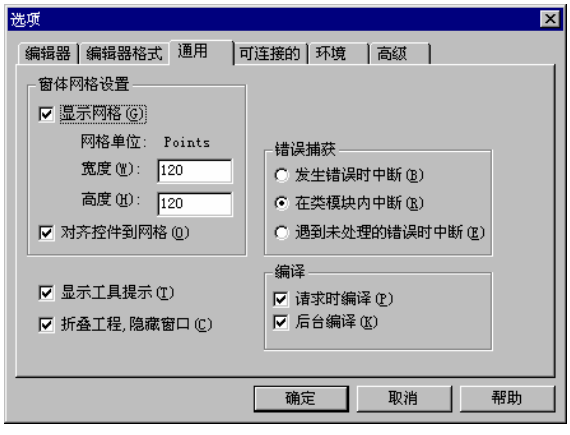

图 2-8 选项对话框中的通用选项卡

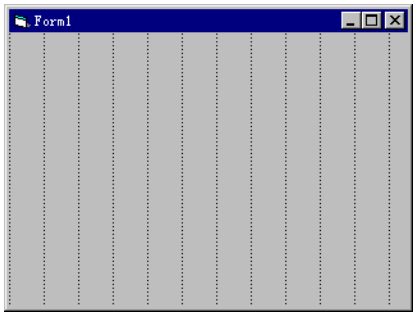

图 2-9 重新设置窗体网格

注意：

“显示网格”和“对齐控件到网格”并不相互依赖，即当我们选择了后者，而不选择前者时，虽然窗体上没有网格显示，但控件仍会向隐藏的网格对齐。相反，当我们选择了前者，而不选择后者时，虽然窗体上有网格显示，但控件却不会自动对齐到网格点，但这时的网格

点，可作为放置控件的位置参考。

一般情况，这两个选项都会被选择，这样，在窗体上绘制控件才会显得更容易些。

2.2.4 多个控件的选择

我们已经知道，只要在任一控件上单击，就可以选中该控件，选中的控件周围会出现八个实心的控制柄。

有时候，为了同时对多个控件进行操作，我们需要同时选择多个控件，这可以使用两种方法：

1. 拖动鼠标指针，在窗体上画一个虚框，则包围在框内的所有控件，将同时被选中。

2. 先选中单个控件，然后按下 Ctrl 键或 Shift 键(不弹起)，再单击其他要选择的控件，这样就可以同时选中多个控件。

这两种方法都很方便，适用场合略有不同。当使用第一种方法时，若虚框内包含了不需要选择的控件，则只要按下 Ctrl 键或 Shift 键，单击该控件，就可以撤消对该控件的选择。要撤消对所有控件的选择，只要在窗体上单击一下即可。

同时选择的多个控件中，只有一个控件被实心控制柄包围，其余控件均由空心控制柄包围，如图 2-10 所示。 被实心控制柄包围的控件我们称之为主控件，单击已选中的多个控件中的任一个，便可以把该控件设置为主控件。

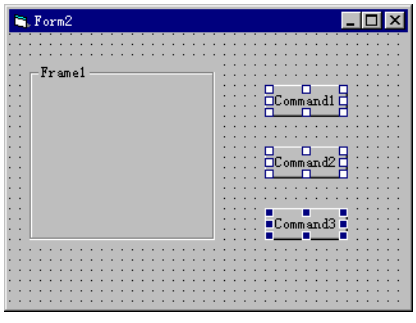

图 2-10 同时选择多个控件

2.2.5 控件大小的调整

对控件的大小不满意时，可以很容易对此进行调整。先选中要调整的控件，然后将鼠标移到该控件的控制柄上，当鼠标变为双箭头时，按下鼠标左键并拖动，就可以改变控件的大小，同调节窗体大小的方法一样。 对于不能改变大小的控件来说，例如，定时器，该操作将毫无意义。

同前面讲到的窗体一样，为了精确设置控件大小，可以使用控件的 Height 属性和 Width 属性，通过设置不同的属性值，就可以改变控件大小。

对于窗体的一组类似控件，当希望它们的大小相同时，可以先选中这些控件，然后选择

“格式”菜单下的“按相同大小制作”选项中的命令。

多个控件调整尺寸时，是以主控件尺寸为基准，即主控件尺寸不变，其余控件根据主控件尺寸来调整尺寸。因此，通过改变主控件，可以设置控件尺寸的基准。

2.2.6 控件位置的改变

控件可以出现在窗体的任何位置，根据需要，我们可以改变控件在窗体中的位置。

最简单的办法是使用鼠标。单击要移动的控件，按下鼠标左键，将鼠标指针移到需要控件出现的地方，释放鼠标键，这就改变了控件的位置。无论什么时候，如要移动控件，使用鼠标便很快地实现，但是，这种方法得到的控件位置不够精确。

为了精确测定控件的位置，可以使用属性窗口，找到控件的 Left 属性和 Top 属性，通过输入不同的属性值就可以改变控件在窗体中的位置。

注意：

对窗体而言，Left 属性是指屏幕左边框到窗体左边框的距离；Top 属性是指屏幕顶部到窗体顶部的距离。

对控件而言，Left 属性是指窗体左边框到控件左边框的距离；Top 属性是指窗体顶部到控件顶部的距离。

当选中多个控件后，对其中任一控件的移动，将会引起所有被选择控件一起移动，并保持各控件相对位置不变。

表 2-2 “格式”菜单中的部分命令

子菜单	命令	功能
对齐	左对齐	以主控件为基准各个控件左边框对齐
	居中对齐	以主控件为基准各个控件水平居中对齐
	右对齐	以主控件为基准各个控件右边框对齐
	顶端对齐	以主控件为基准各个控件顶端对齐
	中间对齐	以主控件为基准各个控件垂直居中对齐
	底端对齐	以主控件为基准各个控件底端对齐
	对齐到网格	所选控件向最近网格点对齐
水平间距	相同间距	以左右两控件为基准调整各个控件水平间距相同
	递增	相邻控件的水平间距增加一个网格间距
	递减	相邻控件的水平间距减少一个网格间距
	删除	删除相邻控件的水平间距
垂直间距	相同间距	以上下两控件为基准调整各个控件垂直间距相同
	递增	相邻控件的垂直间距增加一个网格间距
	递减	相邻控件的垂直间距增加一个网格间距
	删除	删除相邻控件的垂直间距

先在窗体上添加框架控件，然后在框架内绘制另一控件，此时，该控件相当于附属在框

架上，对框架进行移动、删除等操作，都会影响到该控件。但是，若把已存在控件移动到框架上，则二者相互独立，对框架的移动、删除操作不会影响到该控件。

为调整多个控件的相对位置，可先选中要调整的多个控件，然后使用“格式”菜单中的“对齐”、“水平间距”、“垂直间距”三个选项中的命令，现已将这些命令及功能列出如表 2-2 所示。

上面这些命令规律性很强，只要各位读者耐心试用，很快便可以熟练应用。这些命令在绘制用户界面时，使用频率很高，应熟练掌握。

当两个控件重叠时，可以使用“格式”菜单中的“顺序”子菜单来设置哪个控件在上。

2.2.7 控件的删除与拷贝

有时也许画了一个控件，定义好之后，又决定不需要它了，删除控件可谓易如反掌，步骤如下：

1. 选中一个或多个要删除的控件；
2. 按下 Del 键。

注意：

要想删除一个窗体，可不像删除控件那么简单。先选中要删除的窗体，然后选择“工程”菜单中的“删除窗体”命令，这样就可以把一个窗体删除。

在已经绘制好一个控件之后，用户可以对这个控件进行拷贝。为拷贝一个控件，可按下列步骤执行：

1. 单击要拷贝的控件；
2. 按下组合键 Ctrl+C 或者从“编辑”菜单中单击“复制”命令；
3. 按下组合键 Ctrl+V 或者从“编辑”菜单中单击“粘贴”命令。

这时 Visual Basic 6.0 会显示一个对话框，询问是否要创建一个控件数组，如图 2-11 所示。无论选择“是”，还是选择“否”，复制好的控件都会出现在窗件的左上角，并且复制控件与原控件标题相同，可以将控件的拷贝移动到窗体的任一个地方。复制控件与原

图 2-11 复制命令钮时弹出的对话框

控件真正的区别在于：如果选择了“是”，则复制控件与原控件名称相同，而以其索引值（index 属性）来区别；若选择了“否”，则二者虽然标题相同，但名称不同，这一点在代码段中对控件进行调用时，将会体现出差别。以后在用到控件数组时，还会作更详细的说明。

2.2.8 控件的锁定

当我们对窗体上的所有控件都很满意时，这时可以使用“锁定控件”命令，有两种方法：

1. 选择“格式”菜单中的“锁定控件”命令；
2. 单击鼠标右键，在弹出菜单中选择“锁定控件”命令，如图 2-12 所示。

图 2-12 弹出菜单

使用了“锁定控件”命令后，窗体上的所有控件的位置和大小都不可再改变，被选中的控件周围有八个空心的控制柄，表示该控件不可移动。

对于锁定后的控件，要想撤消锁定，也是件很容易的事，同样可以使用上述两种方法。由于该命令具有复选框性质，再选一次便可撤消“锁定控件”命令。

注意：

使用了“锁定控件”命令后，在窗体上添加新的控件，则该控件的位置和大小也不可改变。

2.2.9 添加控件

在窗体上添加控件，就像我们堆积木一样简单容易，但是，控件布局不合理的界面，看起来就会觉得乱七八糟，使用起来也极不方便。因此，在绘制控件前要作一个小小的计划，使程序界面设计得美观、易用。

大多数人习惯于从上到下，从左到右进行阅读，因此，设计界面也应使用户操作时能够按照这个顺序来进行。一般说来，重要的控件最好放在窗体的左上角，使用户一眼看见，倘若在一个选择对话框中，把“确定”按钮放到左上角，就会让人觉得有点莫名其妙。

控件的大小也会直接影响到界面的美观。一般说来，文本框、图片框可以做得大一些，命令按钮则没有必要做得太大，标签控件根据显示的内容可大可小，框架控件则要与它所包含的控件保持一定的间距等等。另外，窗体上类似的控件最好有相同的尺寸，并且要对齐，一个窗体不宜包含太多的控件，这样会导致窗体显得杂乱无章，要适当地留出一些空白，以使界面保持简单明了，容易使用。在设计界面时，还可以采用颜色、图标、字体等，来改善窗体的美观和可用性。

最后还要记住一点，即不要背离 Windows 的界面总原则。对于使用 Windows 的用户来说，他们已经非常习惯 Windows 的界面，倘若我们设计的界面与 Windows 界面相差很大，这就会使用户很不方便。例如，将“保存”命令设置到“帮助”菜单下，恐怕用户就很难找到。

2.3 属性的设置

在编写 Visual Basic 6.0 程序时，我们必须先画用户界面，然后对用户界面中的对象设置属性。

所谓对象的属性，是指对象的性质，是对象固有的本性，例如，“这张桌子高 1 米”则“1 米”就是这张桌子的高度属性。

每个 Visual Basic 6.0 的对象都有其自己的属性，它们的设置，可以控制该对象外观和行为。

2.3.1 属性窗口

每个对象有几十个属性可以设置，为了改变对象的属性，必须先打开属性窗口，步骤如下：

1. 选择需要改变属性的对象；
2. 按 F4 键或者选择“视图”菜单下“属性窗口”命令，为对象打开属性窗口。参见图 2-3 所示。

属性窗口一旦打开，当改变选择对象时，属性窗口自动更新为当前控件的属性。例如，在打开窗体的属性窗口后，用鼠标单击选择窗体上的命令按钮，则属性窗口更新为命令钮的属性窗口。

2.3.2 属性窗口中设置属性

Visual Basic 6.0 自动地对用户界面中的所有对象设置缺省属性值，因此，在实际编程时，只需根据需要改变其中的几个属性即可。

要改变对象的属性，步骤如下：

1. 选择需要改变属性的对象；
2. 在属性窗口选择想要改变的属性；
3. 对那个属性设置新的属性值。

很容易吧？有的属性限于取几个或十几个值，当选择该属性时，右边出现向下箭头，提示激活下拉列表，从中选择属性值。例如：Mousepointer 属性，如图 2-13 所示；有的属性限制性很小，如标题(caption)名称等，选择该属性，用户可自己输入设置值。

注意：

当选择 Font 属性时，右侧出现省略号，单击该按钮将激活一个对话框，如图 2-14 所示，在该对话框中可对字体、字号等进行多种设置。

本质上讲，属性决定了屏幕上对象的外表和行为，对象外表包括它的大小、位置、颜色等；对象的行为指它是否可见，是否可以响应外部事件等。

图 2-13 打开下拉列表

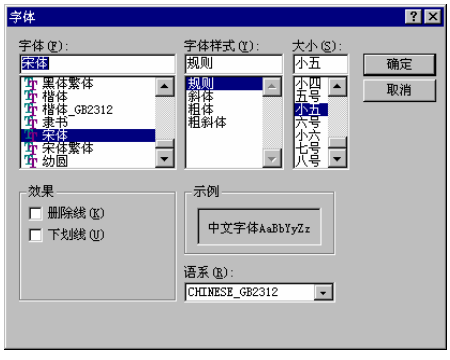

图 2-14 设置 Font 属性对话框

很重要的一点是属性中设置了对象的名称。当我们编写代码时，名称将帮助用户区别不同的对象，例如：

```
Sub Commandl_Click( )
End Sub
```

可以看出，当用户单击(Click)**一个名称为** Command1 **的按钮时这一程序段才会被执行，若没有名称就无法确定某一程序段是针对哪个对象编写的。**

2.3.3 代码窗口中设置属性

对象属性可以在两个阶段进行设置：设计阶段和运行阶段。绝大多数属性可以在设计阶段进行设置；一部分属性也可以在运行阶段改变。但是，并非所有属性都可以在运行时改变。例如，窗体的 Borderstyle **属性、名称就不可以在运行时改变。**

所谓运行阶段改变对象属性法，是将用户对属性的设置写入代码，当代码执行时完成对象属性的改变。例如：

```
Sub Command1_Click( )
    Forml.Caption = "隐藏文本框"
    Textl. Visible = False
End Sub
```

当用户单击一个名称为 Commandl 的按钮时，窗体的标题(Caption)被设置为“隐藏文本框”，Textl 的 Visible 属性被设为 False，不可见，程序运行前后如图 2-15 所示。

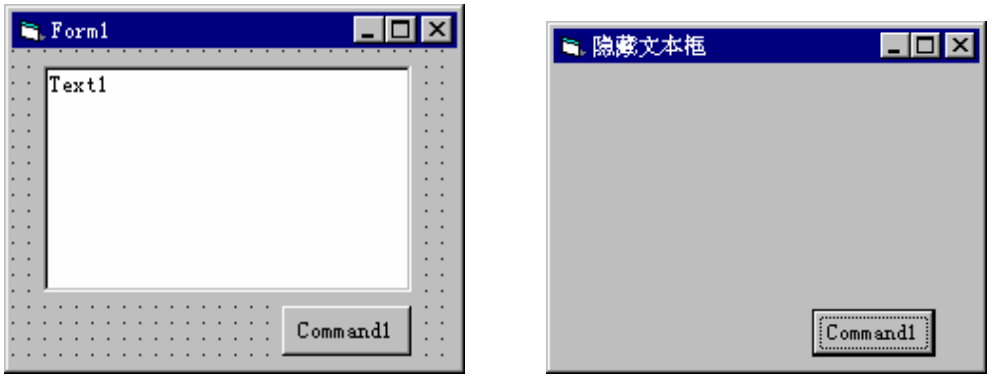

图 2-15　程序运行前后

2.3.4　简单程序举例

下面，我们来进一步改进上面的程序段，编写一个较为完整的 Visual Basic 6.0 程序，来说明如何在运行时改变对象属性绘制界面，在窗体上绘制三个命令钮和一个文本框。

单击命令钮 1　使窗体的文本框隐藏起来，且文本框标题栏显示“隐藏文本框”；

单击命令钮 2　显示窗体上的文本框，且文本框标题栏显示“显示文本框”；

单击命令钮 3　结束程序运行。

一、　绘制界面

绘制界面的步骤如下：

1. 启动 Visual Basic 6.0 程序，建立空白窗体；
2. 在 Visual Basic 6.0 工具箱中单击文本框图标；
3. 将鼠标指针移至需要画文本框的地方，按下鼠标左键，画好文本框；
4. 在 Visual Basic 6.0 工具箱中单击命令按钮图标，在窗体上画好一个命令按钮；
5. 重复第 4 步，再添加另外两个命令钮。

绘制好的界面如图 2-16，这就完成了 Visual Basic 6.0 程序设计的第一步——绘制界面。

注意：

1. 向窗体上添加控件的另一种办法是在 Visual Basic 6.0 工具箱的图标上双击。这样，该控件就会出现在窗体中央，当添加多个控件时，各个控件会相互重叠，最后添加的控件位于最上面，用鼠标移动就会将它们分开，放置在相应位置。

2. 对添加到窗体上的控件，可用“格式”菜单下命令进行调整，各条命令的作用参见表 2-2，建议采用该方法。特别当窗体上控件较多时，使用“格式”菜单下命令会显得快捷、准确。

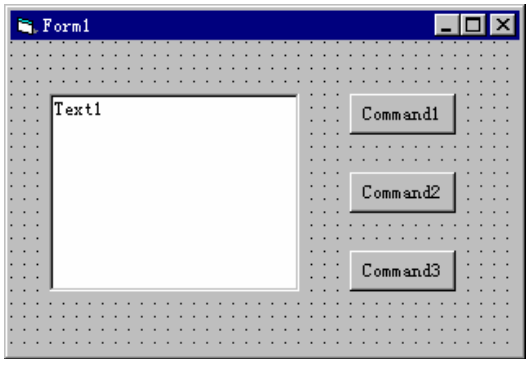

图 2-16 绘制好的界面

向窗体上添加了必要的控件元素之后，就可以进入 Visual Basic 6.0 程序设计的第二个阶段——属性设置。

二、 属性设置

对于窗体及控件的大部分属性，都可以使用其缺省值，只改变对象的几个属性即可。现将需重新设置的属性列表，如表 2-3 所示。

表 2-3 属性设置

对象	属性	设置值
Forml	Caption Borderstyle	演示文本框的 Visible 属性 1-Fixed Single
Commandl	(名称) Caption	Cmdhide 隐藏
Command2	(名称) Caption	Cmddisplay 显示
Command3	(名称) Caption	Cmdend 结束

设置好属性的界面如图 2-17 所示，这样的界面看起来友好了许多，用户对程序的大体功能也有了更进一步了解。

图 2-17 设置好属性的窗体

三、编写代码

设置好属性的界面还是不能对用户的操作作出任何响应，要想实现预期的功能，还要附加代码段。

为 Visual Basic 6.0 程序编写代码，首先要打开代码窗口，方法有三种：

(一) 双击窗体或窗体上的任一控件；

(二) 选择“资源管理窗口”中的  图标；

(三) 选择“视图”菜单中的“代码窗口”命令。

通过上述方法打开的代码窗口如图 2-18 所示。

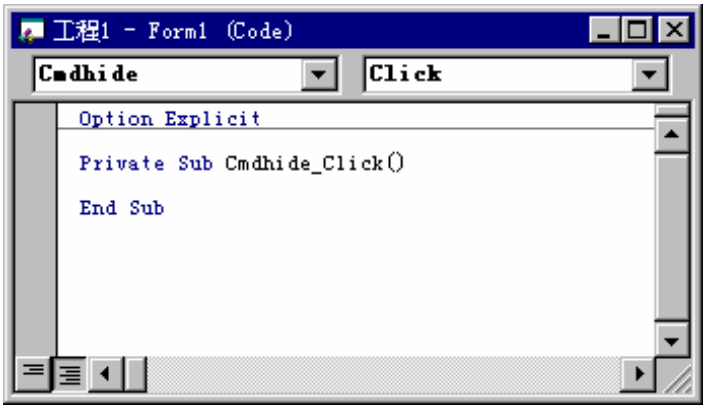

图 2-18 打开代码窗口

左边的下拉列表中包含了窗体及窗体上的所有控件，从中选择任一控件，右边的下拉列表中包含了该控件所有的事件。由于 Visual Basic 6.0 的事件驱动程序设计思想，因此，任意一段程序都是针对某一对象的特定事件编写的，称为事件过程。

注意：

在代码窗口中编写的通用过程和函数过程可以供不同的事件过程调用。

打开代码窗口，在对象列表中选择 Cmdhide，其默认事件是 Click(单击)，需要使用控件的其它事件时，可以在事件列表中进行选择。

注意：

在图 2-18 中的 Option Explicit 是强制变量声明语句，用于窗口和模块的通用声明段。使用它以后，所有变量必须在使用前用 Dim 、Private、Global 或 static 进行声明。否则，系统在编译时会发出出错提示。

在“工具”菜单中选择“选项”，然后选择“编辑器”选项卡，如图 2-19 所示，选择“要求变量声明”复选框。这样，每当打开代码窗口，就会出现 Option Explicit 语句。建议使用这条语句。

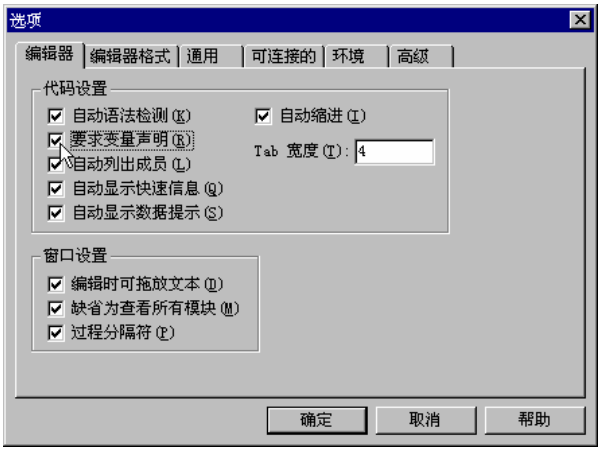

图 2-19 “编辑器”选项卡

在图 2-18 的代码窗口中，已经显示了 Cmdhide_Click **事件过程的首末两条语句，其完整代码如下：**

```
Private Sub Cmdhide_click(     )
    Text1.Visible=False
    Form1.caption=“隐藏文本框”
End Sub
```

在 Text1.Visible=False **这条语句中，**Text1 **为文本框的名称，**Visible **为** Text1 **的属性，**Text1.Visible **表示引用** Text1 **的** Visible **属性。这段程序表示，当** Cmdhide **按钮被单击时，设置** Text1 **的** Visible **属性为** False**，即文本框不可见；同时，在** Form1 **的标题栏中显示“隐藏文本框”字样。**

在图 2-18 的代码窗口中，选择 Cmddisplay **按钮的** Click **事件，编写代码如下：**

```
Private Sub Cmddisplay_Click(    )
    Text1.Visible=True
    Form1.Caption=“显示文本框”
End Sub
```

当 Cmddisplay **按钮被单击时，设置** Text1 **的** Visible **属性为** True**，即显示文本框，同时，在** Form1 **的标题栏中显示“显示文本框”字样。**

在图 2-18 的代码窗口中，选择 Cmdend **按钮的** Click **事件，编写代码如下：**

```
Private Sub Cmdend_Click(    )
    End
End Sub
```

End **为** Visual Basic 6.0 **中的一条语句，当** Cmdend **被单击时，表示结束程序运行。**

四、 运行程序

现在，程序已经可以响应用户事件了。可以对程序进行试运行，方法有两种：

(一) 单击工具栏中的  **按钮；**

(二) 选择“运行”菜单中的“开始”命令。

现在来运行上面编写的程序。单击工具栏中的 按钮，进入试运行状态。单击“隐藏”按钮，结果如图 2-20 左所示；单击“显示”按钮，结果如图 2-20 右所示；单击“结束”按钮，结束程序运行，回到程序设计状态。

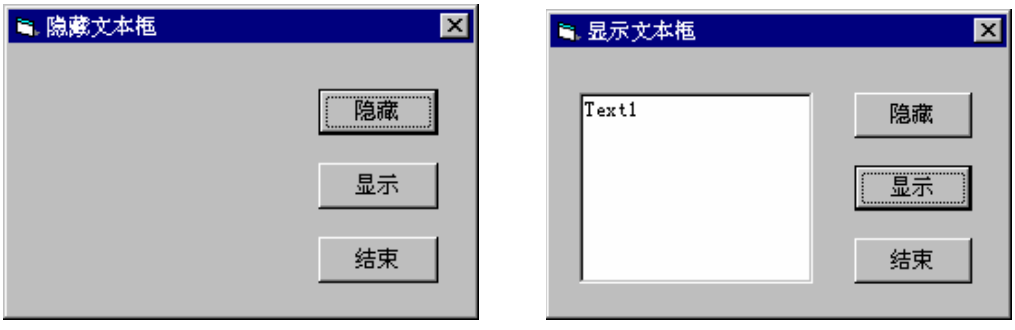

图 2-20 运行程序

注意：

结束程序运行还有另外两种方法：（1）单击工具栏中的 按钮；（2）选择“运行”菜单中的“结束”命令。

程序运行结束，实现了我们所设计的功能，现在可以把它生成一个.exe 文件，作为一个实应用程序保存起来。这样，以后脱离 Visual Basic 6.0 环境，也可以执行该程序。

五、 生成可执行文件

将编好的程序生成可执行文件，步骤：

1. 选择“文件”菜单下的“生成工程 1*.exe ”选项，将弹出一个对话框；

2. 在对话框中找到要保存可执行文件*.exe 的目录，然后输入文件名称，扩展名为.exe，单击确定。

这样，Visual Basic 6.0 就将编好的程序生成了可执行文件，并保存到用户设好的目录下。在该目录下找到刚才生成的可执行文件，双击该文件，就可以脱离 Visual Basic 6.0 环境执行。

在这个例子中，包含了 Visual Basic 6.0 程序设计的基本步骤，试着编写这个程序，会对 Visual Basic 6.0 编程及对象的属性有更深刻的了解。这一节中不懂的内容，在以后的章节中，大家会慢慢得明白，请不必着急。

2.3.5 属性介绍

通过前面的学习，相信各位读者对窗体及控件的属性已有了一些了解。为了不使这部分看起来，繁复累赘，令人生厌，编者并不打算将所有对象的属性一一列出，只作一些简要的介绍，使读者对一些常用的属性留有印象。在以后的章节中，我们将用实例教会大家如何使用这些属性。

一、 窗体属性

窗体是 Visual Basic 6.0 用户界面的基体，现仅将其部分属性列表如下：

表 2-4 窗体的部分属性

属性	作 用
(名称)	定义代码中窗体名字
Backcolor	设置窗体窗口的背景颜色
Borderstyle	设置窗体窗口的边框样式
Caption	确定标题栏中显示的内容
Enable	是否可以响应键盘或鼠标事件
Font	设置窗体正文字体、字体样式和大小
Height	决定窗体窗口高度
Width	决定窗体窗口宽度
Icon	设置最小化时显示图标
Left	设置窗体左边水平坐标
Top	设置窗体顶部垂直坐标
Visible	设置窗体是否可见

二、 名称与标题的区别

一个对象，既有名称也有标题(Caption)，二者有何区别？举个例子，若把人看做一个对象，则人的衣服可以看做是“标题”，人的名字可看做是“名称”。“标题”是直接显示在对象上的，从用户界面上可以直接看到；而“名称”在代码中才会使用，用来区别不同的对象，“名称”在用户界面上是看不到的。

在前一部分的例子中，“隐藏”、“显示”、“结束”分别是各个按钮的“标题”，而“Cmdhide”、“Cmddisplay”、“Cmdend”才是各个按钮的“名称”。当然，对象的标题和名称是可以相同，在 Visual Basic 6.0 的缺省状态下，二者就是相同的。

三、 控件属性

大部分 Visual Basic 6.0 控件都有几十个属性，通过设置它们，可以改变控件的外观和行为。这里，只列出几个最常用控件的重要属性，以使读者有个大体了解，与窗体重复的属性这里不再列出。

(一) 命令钮(CommandButton)

Cancel 指定命令钮是否是 Cancel 按钮。若设置 Cancel 值为 True，则只要按 ESC 键，该按钮就被激活。在 Form 中只有一个按钮可以被设定为 Cancel 按钮，其它按钮的 Cancel 值系统自动设为 False。

Default 指定命令钮是否为缺省按钮。若设置 Default 为 True，则只要按 Enter 键即可激活它，效果同单击该按钮一样。

Value 命令钮的 Value 属性表明该按钮是否被选中。当其值为 True 意味着选中，当其值为 False 意味着未选中。在程序代码中设置为 True，则该命令按钮的 Click 事件被激活。

Index 创建控件数组时，系统自动设置其值，作为控件数组下标，用以区别数组

中的不同对象。控件数组的第一个元素 Index 值为 0，第二个元素 Index 值为 1，……以此类推。

(二) 标签(Label)

Alignment 用于设置标签中正文位置。当它为 0 表示左对齐，当它为 1 表示右对齐，当它为 2 表示居中。

Borderstyle 设置标签的边框类型。当它为 0 表示无边框，当它为 1 时表示有边框。

Backstyle设置 Label 与背景是否透明。当它为 0 时表示透明，为 1 时表示不透明。

Autosize 设置标签是否可以根据显示内容自动调节大小。

(三) 文本框(TextBox)

Maxlength 指定文本框中文本的最大宽度。当它为 0 时，表示没有限制，可以输入系统允许的任意多字符，当它为非 0 数字时，表示可输入的字符数。

Multiline 设置文本框是否可以多行显示。

ScrollBars 指定文本框是否有水平或垂直的滚动条，只有当 Multiline 属性设为 true 时，该属性才有效 。

Passwordchar 用于建立密码输入区。设置为某一字符后，输入文本均显示为该字符，但文本内容并没有改变。

(四) 复选框(CheckBox)

Alignment 同标签一样，设置标题显示位置。

Value 指定复选框状态。当它为 0 时表示复选框被取消，当它为 1 时，表示复选框被选中，当它为 2 时，表示该复选框当前不可选。

(五) 选项按钮(OptionButton)

Value 指定选项按钮状态，与复选框不同，选项按钮属性只有两个值：True 和 False。当设置为 True 时，表示被选中，为 False 时，表示未选中。同一组选项按钮中，只能有一个被选中。

(六) 组合框(ComboBox)

ItemData 它是一个长整型的数组，存放列表中要显示的文本，设计阶段不可用。

List 它是一字符串数组，用于存放显示在列表框中的项目，设计阶段不可用。

ListCount 存放列表中项目数，设计阶段不可用。

ListIndex存放当前选定条目的索引号。

Sorted 指定列表中元素的排列顺序。当 Sorted 为 True 时，以字母顺序排列条目，当 Sorted 为 False 时，条目以 AddItem 的增加顺序排列。

Style 设置组合框的类型和行为。当 Style 为 0 时，组合框是下拉组合框，可以从列表中选择值，也可在编辑区录入值；当 Style 为 1 时，组合框为简单组合框，可选择值，也可在编辑区录入；当 Style 为 2 时，列表可下拉，编辑区不能录入。

NewIndex 存放最近追加进来的项目序号，设计时不可用。

SelCount 存放被选中的项目数。

(七) 列表框(List box)

列表框与组合框有很多共有属性，如 ItemData、 List、 ListCount、 Sorted 等。同时，

二者又各有其专用属性，下面列出列表框的四个专用属性。

Columns 确定列表框中要显示的栏数和滚动方式。当其值为 0 时，列表框显示为单列，并垂直滚动；当其值为 1 或更大的数时，表示要显示栏 ，同时可水平滚动。

Selected 指定列表条目是否被选择。当 Selected 为 True 时，该条目被选中，当 Selected 为 False 时，则当前列表条目未被选中，该属性使用数组，设计阶段不可用。

TopIndex指定显示于列表顶部的条目的索引号。

Multiselect 其值确定了多次选择的功能。当它为 0 时，多次选择无效；当它为 1 时，可分别单击待选条目完成多次选择；当它为 2 时，以组的形式选择多个项目。

上面仅列举了常用控件的一些重要属性，以使大家有个基本概念，在这里，要指出两点，提醒大家注意。

不同对象往往具有许多相同或类似的属性。但是，这些属性的设置值不一定相同。例如，(1) 窗体和标签都有 Borderstyle 属性，但是窗体的 Borderstyle 属性有六个可选值，分别代表六种不同的窗体边框样式；而标签的 Borderstyle 属性只有两个可选值，分别代表标签有边框和无边框。再如，复选框的 Value 属性有三个可选值，而单选钮只有两个可选值，即 True 和 False。

在对象的所有属性中，大部分属性既可以在设计阶段进行设置，也可以运行阶段进行设置。如 Caption、Visible、Value、Font，等等。但是，有些属性只能在设计阶段设置，如“(名称)”属性，窗体的 Borderstyle 属性；而另外一些属性则只能在运行阶段使用，如组合框的 ItemData、 ListCount、 ListIndex 等。

四、 如何学习对象的属性

如果要大家把 Visual Basic 6.0 所有对象的属性都背下来，然后上机编程，这样，恐怕会使大多数人望而却步。事实上，也完全没有这个必要，只要大家对对象属性已经有了一定的了解，在上机时学习对象属性，会更方便，更实用。

建立空白窗体，在窗体上画好控件，选择控件，打开属性窗口。在属性窗口中可以看到所选控件的所有属性。任选其中一个属性，在属性窗口下方都会出现一段简短的提示，如图 2-21 所示，说明该属性的作用。试着改变属性设置，运行程序，从实践中体会属性的作用。

对于一些弄不明白的属性，可以借助联机帮助，在“搜索主题”中键入该属性，这样便可以从 Visual Basic 6.0 的帮助文件中学到很多东西。我们还可以选择“视图”中的“对象浏览器”，其中列出了 Visual Basic 6.0 的所有类及其成员，常用控件的属性、方法和事件都可以从中找到。

图 2-21 属性窗口中的提示

由于在 Visual Basic 6.0 的应用程序中，自带了许多提示，同时还有巨大的联机帮助文件，因此，为我们对 Visual Basic 6.0 有了一定的了解后，利用这些条件来加深学习将是一条很好的途径。而读者最缺乏的是通过许多实际应用的例程来进入 Visual Basic 6.0 的殿堂，因此，根据这一想法，在这一部分首先将 Visual Basic 6.0 编程作一个大体的介绍，以后各章将用大量的实例来教会大家如何使用 Visual Basic 6.0 编程。这也是本书的编排思想。

2.4 程序代码的编写

前面的例子中，已使大家对 Visual Basic 6.0 的代码有一个初步的了解，第二部分也向大家介绍了 Visual Basic 6.0 编程的基础，下面再向大家介绍一些 Visual Basic 6.0 编程方面的有关知识。

2.4.1 事件的概念

人类的行为都是事件的集合。例如，闹钟响了就要起床，肚子饿了就要吃东西，生病了就要去看医生……前面闹钟响，肚子饿及生病都是事件，后面的起床、吃东西、看医生都是事件处理子程序。

Visual Basic 6.0 的处理方式也是如此。但是限于有限的输入设备，目前 Visual Basic 6.0 中只包含了类似窗体装载(Load)、对象内容改变(Change)、鼠标单击(Click)、鼠标双击(Dbclick)键盘被用户按下(Keypress)等几十种事件。

2.4.2 事件驱动过程

当一个对象发生某一事件时，如 Click、Change、Load 等等，对象就要对事件进行处理，这就是事件过程。它是针对某一对象的过程，并与该对象的一个事件相联系，事件过程的格式如下：

```
Sub  对象名称_事件( )
     [代码段]
End Sub
```

只有对象的事件发生时，程序代码才会被执行。若对象事件未发生，程序代码将处于停滞状态，这就是事件驱动程序设计，例如：

```
Private Commandl_click ( )
     Commandl.Caption="事件驱动"
End Sub
```

只有当名称为 Commandl 的按钮被单击时，该按钮的标题才会改变为"事件驱动"，否则，其标题内容不变。

2.4.3 面向对象程序设计

面向对象是今天程序设计中的一个热门话题。总的来说，对象将程序化代码和数据组合为一个独立单元。一旦定义，对象就具有了生命，不再需要知道对象是如何产生的和如何进行工作，而只需简单地将信息传递给它们。同时，对象还可以保存在其它的程序中。

在 Visual Basic 6.0 中，主要的对象是窗体和控件。窗体定义屏幕上的显示窗口，它和它上面的控件是应用程序运行时用户实际看到的东西，控件放置于窗体之上，每种控件完成一个特定功能。可以为每一对象的各种属性定义属性值，它们包括大小、颜色和正文标题等，应用程序运行时可以修改这些值。每一个 Visual Basic 6.0 的对象都有一系列相关的事件过程，正如前面的描述，选择每一对象的可识别事件是靠写事件过程代码来完成的，对象可以独立存在。例如，可将对象作为参数传递给各种过程，可以建立同其它对象一样的新对象。

窗体可以被保存或重新用于其它应用程序中，其中包括窗体描述，窗体上的控件，属性值设置和相关事件过程。这种“对象”在术语意义上近似于面向对象程序设计，它与任何特别的应用程序无关，换句话说，即可将一个保存好的窗体装入其它应用程序中使用。

2.4.4 代码窗口

在 Visual Basic 6.0 中有三种类型的模块：窗体模块、程序代码模块和类模块。所有这三种形式的模块都可以包含过程，并在应用程序中有不同层次的作用范围。

绘制好用户界面之后，双击窗体或窗体上的任一控件，就会打开窗体的代码窗口。如图 2-22 所示，在其中可以编写窗体及窗体上所有控件的事件过程，例如：

```
Private Sub Form_Load(     )                ‘窗体的加载事件
    (处理 Load 事件代码)
End Sub

Private Sub Commancll_Click(    )           ‘命令钮的单击事件
    (处理 Click 事件代码)
End Sub
```

从对象列表中选择“(通用)”，可以进行变量声明或编写 Sub 程序与 Function 程序。在这里声明的变量可以在该代码窗口中的任意过程中使用，Sub 程序、Function 程序也可以被该代码窗口中的任意事件过程调用。

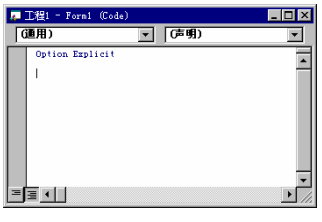

图 2-22 打开窗体代码窗口

图 2-23　程序代码模块窗口

从“工程”菜单中选择“添加模块”命令。如图 2-23 所示，可以为一个工程添加一个模块，模块的代码窗口中可以包括全局变量声明，Sub 程序与 Function 程序，这些在一个工程中，都可以在所有的过程中使用或调用。当一个工程仅包含一个窗体时，建立程序代码模块没有太大意义。

选择“工程”菜单下的“添加类模块”命令，可为一个工程增加一个类模块。 如图 2-24 所示，类模块包含了一个 OLE 自动对象类的正式定义。简单地说，一个类它所体现的就如同是一个模板，在执行阶段会从这里产生一个对象。同时，类定义了对象的属性，以及用类控制对象行为的方法。

2.4.5　属性与方法引用

为了在程序代码中使用对象的属性，就要学会如何引用对象的属性，引用格式如下：

对象名称.属值

例如：

Form1.Caption=“你好！”

Text1.Text=“1+1=2”

Command1.Enabled = False

分别表示：在 Form1 的标题栏中显示“你好！”；在文本框 Text1 中显示“1+1=2”；设置 Command1 不响应事件。

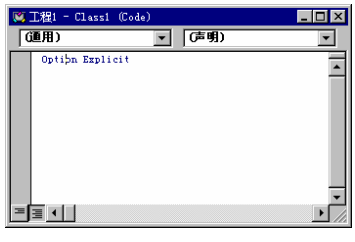

图 2-24　类模块窗口

当对象名称缺省时，表示当前窗体，例如：

```
Private Sub Commamd1_Click( )
    Caption=“你好!”
```

End Sub

其中，“Caption=‘你好！’”等价于“Form1.Caption=‘你好！’”，Form1 为 command1 所在窗体。

当属性缺省时，表示该对象的缺省属性，一个对象只能有一个缺省属性，这是 Visual Basic 6.0 中已经约定好的。例如：

Text1=“1+1=2”

表示在文本框 Text1 中显示“1+1=2”，该语句等价于“Text1.Text=‘1+1=2’”，这表明 Text 属性是文本框的缺省属性。

连续引用同一对象的多个属性时，可使用如下格式：

With 对象名称

.属性=设置值

.属性=设置值

……

End With

例如，

With Label1

.Caption =“Hello！”

.Height =1000

.Width =1000

.Visiable = True

End With

在 Visual Basic 6.0 中，程序语句操纵着对象完成指定的动作，与对象相关的程序语句被称为方法(methods)。要在 Visual Basic 6.0 指令中调用方法，格式如下：

对象名称.方法 [参数]

有些方法要求具有参数，需给出相应参数，有些方法则无参数要求。

例如，Print 语句是过程化程序设计者十分熟悉的一条命令，在 Visual Basic 6.0 中被变换为一个方法，如

Form1.Print“There a method to this madness.”

这里，Print 是作用于 Form1 对象上的一个方法，引号内容是 Print 方法要求的参数，它表示在窗体上显示正文。此时，该语句表示在窗体上显示“There a method to this madness”。

2.4.6 注释的使用

在程序中使用注释是一个很好的习惯，我们可以使用注释来说明某段代码或声明某个变量的目的，以后只要看到注释就会想起当初编程的思路，不至于连自己都看不懂自己编写的程序。同时，在别人阅读程序时，注释也会起到说明作用，大大增强了程序的可读性。

要在代码中添加注释，只要使用“’”作为注释文字的开头即可。“’”会告诉 Visual Basic 6.0，这以后的内容是注释，编译时可忽略，注释可以加在过程的前面，也可以加在程序代码

中。例如：

```
'编写时间：1998 年 10 月
'程序功能：单击命令钮，累加 1-10 个自然数，并在文本框中显示
Private Sub Commandl_Click(  )                    '事件过程开始
    Dim i，Sum as integer                         '定义 i，Sum 为整型
    For I=1   to   10                             '利用 For 循环累加
        Sum=Sum+I
    Next I                                        '结束循环
    Text1.Text=Sum                                '在文本框中显示累加洁果
End Sub                                           '事件过程结束
```

添加了注释固然会增强程序的可读性，但是，太多的注释又会使程序看起来乱七八糟。因此，注意在程序中添加一些重要注释，对于一目了然的语句则没有必要加注释。

2.5 本章小结

这一章的前半部分，介绍了创建用户界面的方法，叙述比较详细。绘制用户界面是 Visual Basic 6.0 编程的基础 ，应熟练掌握。后半部分介绍了属性及代码的有关知识。总的看来，叙述了 Visual Basic 6.0 编程的基本步骤，以后的各章中，将以实例为基础进行编写，在实例中教会大家如何使用 Visual Basic 6.0 来编写应用程序。

第三章　变量

Visual Basic 6.0 程序设计是在可视的环境下，面向对象、由事件驱动的高级程序设计语言。它巧妙地将 Windows 编程的复杂性封装起来，并综合运用了 BASIC 语言和新的可视化程序设计工具，借助 Windows 的优良性能和图形工作环境，让开发人员便捷地构造功能强大的应用程序。

在 Visual Basic 6.0 程序设计，尤其是编写代码过程中，我们要大量使用变量。所谓“变量”就是在程序中可以改变的量。变量用作信息的符号容器，并占有一定的内存空间。

本章将向读者介绍如何在 Visual Basic 6.0 程序中建立和使用变量，以及变量的不同作用范围。

3.1　数据类型介绍

程序从计算机外部获取的信息被称为数据。任何语言都提供标准的数据表示方法，称为数据类型。在 Visual Basic 6.0 语言中使用较多的是数值和文本（字符串）数据类型，此外还有缺省（又叫变体）数据类型。详见表 3-1。

表 3.1　Visual Basic 6.0 标准数据类型

	描述	类型说明符	范围
Integer	2 字节整数	%	-32768~32767
Long	4 字节整数	&	-2147483648~ 2147483647
Single	4 字节浮点数	!	-3.402823E38~ -1.401298E-45
Double	8 字节浮点数	#	±4.94065645841247D324~ ±1.79769313486232D308
Currency	带固定小数点的 8 字节数	@	-922337303685477.5808~ 922337303685477.5807
String	字符串	$	0~大约 65500 个字符
Variant	日期/时间 浮点数或字符串	(无)	上述之一
Type	用户自定义类型 元素为上述之一	（无）	每个元素范围为上述之一

3.2 变量的声明与赋值

变量的基本类型有两种：数值变量和字符串变量。其命名须遵循以下规则：

- 必须以字母开头；
- 后面的字符必须是字母，数字或下划线（_）；
- 变量名的长度不能超过 40 个字符；
- 不能用关键字、属性名、对象名、过程名等作变量名。

例如 007、This*Man、Sub 这些命名都是非法的,China、year_2000、Sub2 则是可以接受的。

此外，Visual Basic 6.0 不区分字母的大小写，变量 APPLE、Apple、 apple 被认为是同一变量。

3.2.1 声明

变量可以显式地声明，也可以隐式地声明。假定变量 Number 在程序中未被定义，代码中有这样的语句：

Number=1234

程序执行这条语句时，变量 Number 就被隐含地创建。在没有另外指定的情况下，变量使用“Variant”类型来创建。因此上例中 Number 是一个在内部表示为 Integer 的 Variant 类型变量。

通常我们要将一个变量明确声明为某个数据类型可以使用 Dim 语句，其格式为：

Dim 变量名 As 数据类型

举例如下：

```
Dim Count As Integer
Dim Yes_or_No As String
Dim Income As Currency
```

也可以直接在变量名后加上类型说明符，声明变量为某种数据类型。如：

```
temp%=999
word$=“Welcome！”
```

3.2.2 赋值

使用赋值语句，格式为：

[Let] 变量名＝表达式

Let 关键字可以省略。

例如：

```
Age=24
Name=“John”
```

```
Sum=A+B
PhoneNumber=“62780654”
TxtPassWord.Height=1200
```

也可将一个变量赋给另一个变量，如：

```
Dim Message As String
Message=“Hello,world !”
Copycat= Message
```

3.3 作用域

在程序中，变量的作用域是指该变量在项目中的可见性（或可访问性），它决定哪些程序可以访问该变量。

变量作用域有以下几种情况：

- **局部变量**
- **窗体级变量**
- **模块级变量**
- **全局变量**

3.3.1 局部变量

只在 Sub 或 Function 过程中使用的变量称为局部变量。当创建它们的过程执行结束时，变量也就消失了。

局部变量除可用 Dim 语句声明外，还可用 static 语句将变量声明为静态变量。只要程序在运行，静态变量就存在；当过程被再次调用时，静态变量的值在上一次执行结果的基础上进行操作。

下面是在一个过程中建立局部变量：

```
Sub Command1_Click（）
    Dim Number As Long, Total As String
    Static Sum As Integer
    …
End Sub
```

3.3.2 窗体级变量

是可被一个窗体的所有过程和函数所共用的变量。该窗体下的所有子程序均可直接使用该变量，且变量值在窗体范围内不会消失。

声明方法：在窗体所对应的代码窗口内，将“对象”置为[通用]，“过程”置为[声明]，然后在代码框内窗体的变量声明部分键入 Dim 语句。如图 3-1。

3.3.3 模块级变量

模块级变量是指仅限于同一个模块文件下的各通用过程和函数可以使用的变量。只要程序在运行，模块级变量就存在，甚至窗体被卸出后，仍可以使用该窗体的模块级变量。

创建模块级变量，只需建立或打开一个模块文件，其余操作与窗体级变量的建立相同。

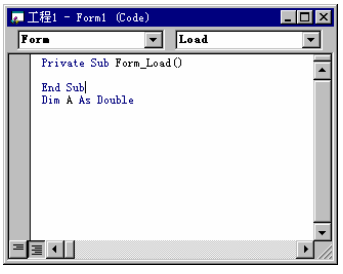

图 3-1 建立窗体级变量

3.3.4 全局变量

全局变量可被程序中任何过程或函数使用。例如，想要在程序的每个窗体的标题中都使用“Happy New year !”，则可以创建全局变量包含该字符串。

在声明中使用 Global 语句，例如：

```
Global Name As String
Global Population
```

注意：

全局变量不能用在窗体对应的代码中，而必须在模块对应的代码窗口中声明。为使用方便，可将所有的全局变量声明放在一个单独的模块中，存成 Global.bas 文件。

3.4 对象变量

所谓对象是指程序员在程序设计中可以访问的元素，它包括控件所代表的图形对象，还包括窗口、屏幕、打印机等环境对象。

在程序设计中，可以将某一类对象声明成一个变量，对该变量进行控制和操作，则表示对项目中所有属于这一类的对象进行相同的控制和操作。窗体对象包括屏幕以及数据库中的表、记录、字段等对象均可在程序中通过变量来引用。

对象变量也有类型。对象类型指控件和对象的名称（Name）、控制集合（Control、Textbox、Commandbutton 等）、窗体集合（Form）、屏幕对象（Screen、Object）等。

可以使用 Dim、Static、Global 等关键字声明对象变量，例如：

```
Dim Controlvar As Control
Dim Formvar As    Form1
```

Global Commandvar As Commandbutton

在对象变量声明以后，所有与对象相关联的属性（大小、位置、状态等）均可通过变量来传递。

如将项目中所有窗体的标题打印出来：

```
Global Forms As Form
I%=0
For I=0 to Forms. count-1
Print Forms (I).caption
Next I
```

又比如希望将项目中所有控件上的标题显示为同种字体设置：

```
Global A As Control
…
A. fontname=“…”
A. fontsize=…
…
```

在对象变量的声明语句中可以使用 New 关键字创建一个原始对象的拷贝，即定义一种新的派生类，如：

```
Dim aForm as New Form1
Dim bForm as New Form1
```

这里的 aForm 和 bForm 继承了 Form1 的所有属性。

对象都具有特殊属性—Count 属性。通过读取变量 Count 属性值，便可了解项目中该类对象的个数，如某项目中窗体的个数。

3.5 自定义变量

用户可以根据需要创建自己的数据类型。格式如下：

```
Type 自定义类型名称
元素名称 As 类型名称
……
End Type
```

例如，为存放一个考生的考试成绩可编写如下代码

```
Type Record
    Name As String
    Grade As Integer
    Subject As String
    Score As Single
End Type
```

定义一个结构

Dim Student As Record '定义存放记录变量

当希望访问变量 Student 的某个成员时，可用变量名加上英文句号，再加上元素名称，例如：

Print Student. Name

3.6 本章小结

了解变量是进行程序设计的第一步。

在 Visual Basic 6.0 程序设计中，变量可归结为两种类型：

- 用户自己建立的变量
- 窗体中作为每个对象属性所定义的变量

一般说来，变量是可以代表任何类型值的名称，而属性是表示影响一个对象外表的特殊变量名。

与变量相对的是常量，声明格式如下：

Const PI！=3.1415926

第四章 VB 程序语句构成要素

程序运行时并非总是顺序执行每一条指令代码，我们常常要根据数学或逻辑的需要设计选择和循环结构，以更好地运行和管理程序。

本章将简要介绍选择和循环在 Visual Basic 程序设计中的实际应用以及如何在程序控制中灵活地使用数组。

4.1 条件选择语句

根据测试条件的类型，Visual Basic 提供两种选择结构：If 结构和 Select Case 结构。前者测试 True 和 False 条件，后者测试指定范围内的值。

4.1.1 If

If 语句是根据条件判断控制程序的分支，条件表达式是由关系或逻辑表达式构成的。书写 If 语句时，要严格遵守语法规则，不能漏写。

一、If-Then 语句

一个典型的 If-Then 语句格式为：

```
If Condition Then Instruction
```

如果判断条件为真，则执行后面的指令。例如：

```
If Number>25 Then txtNote.Text=“Full”
```

二、If-Then-End If 语句

语法规则：

```
If Condition Then
    Instruction1
    Instruction2
End If
```

先检查条件式，如果为真，则执行直到 End If 的所有语句。

三、If-Then-Else 语句

语法规则：

```
If Condition Then
    Instruction1
Else
    Instruction2
End If
```

四、If-Then-Else-If 语句

为了检查多个条件式，可用多重 Else-If 语句，如下面的形式：

```
If Condition1 Then
    Instructions1
Else If Condition2 Then
    Instructions2
Else If Condition3 Then
    …
[Else
    Instructions]
End If
```

其中方括号内的语句可以省略。

注意：

If 与 End If 必须成对出现，表示条件控制结构的开始和结束。Then 和 Else 关键字后面不能跟任何语句，语句块必须另起一行写。

4.1.2 Select\Case

使用 If 结构可以实现几乎所有的选择控制，但在有些情况下，使用 Select 结构效果更好。它是专门适用于多分支结构的程序语句，通过判断变量或表达式的值，转去执行不同结果下的指令。

语法格式为：

```
Select Case Expression
Case A
  Instructions1
Case B
  Instructions2
Case C
  Instructions3
  …
[Case Else
    Instructions]
End Select
```

此句中测试表达式可以是变量、算术表达式或字符串表达式其类型必须和 Case 后面表达式的类型一致。

Case 语句后面的表达式有 3 种表示方法。

一、常量、变量、算术或字符串表达式。

可有多个表达式，其间用逗号分开。只要其中有一个表达式匹配，就执行该 Case 下的语句块。

例：

```
Select Case M－N
Case 0,1,2,3,4
    Msg=Store&“Fill”
Case 5,6,7,8,9
    Msg=Store&“Erase”
Case Else
    M=M+1
End Select
```

二、表达式 To **表达式。**

表示取一定范围的值，指定范围时，表达式值必须从小到大。

例：

```
Select Case Q
Case 150 To 200
    Label.Caption=“超重”
Case 60 To 149
    Label.Caption=“中等”
Case 0 To 59
    Label.Caption=“轻型”
End Select
```

三、Is **关系表达式。**

Is **用于表示测试表达式与关系表达式在指定范围内进行比较，当指定范围为真，则执行该** Case **下面的语句块。**

例：

```
Select Case Val(Text1.Text)
Case Is>0
    A=A＋1
Case Is<0
    A=A－1
End Select
```

以上三种形式可以混合使用，但表达式类型必须一致，每个 Select Case **语句必须与** End Select **语句成对出现。**

4.2 循环控制语句

使用循环可以快速解决复杂的数学运算，可以在屏幕上打印出按一定规律排列的字符，可以反复执行所需操作。

4.2.1 Do

一、Do While **循环**

在 Do While **循环中**，While **引导的条件式放在循环的顶部或底部均可**。

例 1：

```
I=10
Do While I>0
    I＝I－1
    Print I
Loop
```

循环每运行一次，变量 I **的值减少** 1；**当** I **值为零时，循环结束**。

例 2：

```
N＝0
Do
    N＝N＋2
Loop While N<20
```

循环每运行一次，变量 N **的值增加** 2；**当** N **值增加到** 20，**循环结束**。

二、Do Until **循环**

Do Until **循环也有两种形式：**

格式 1：

```
Do Until Condition
    Instructions
Loop
```

格式 2：

```
Do
    Instructions
Loop Until Condition
```

注意：

Do Until **和** Do While **循环语句刚好相反**，Do Wile **语句只有在条件为真时保持重复执行**，**而** Do Until **语句只有在条件为假时保持循环**。

4.2.2 For/Next

For **循环比** Do **循环要简单、清晰得多**。

其语法格式为：

```
For 变量＝初值 To 终值 [步长]
    语句块
Next 变量
```

一个 For 循环必须建立它自己的变量来进行计数。所用步长可以是整数或小数。

例：

```
For XYZ=1 To 50 Step 10
    Tuv＝M*N－XYZ
Next XYZ
```

注意：

在使用循环的时候，一定要注意是否有使循环结束的条件，以免出现无限循环。此外使用 Exit For、Exit Do 或 Goto 语句可跳出循环体，执行其它语句。

4.3 应用举例

关于程序流程与判别更详细的介绍，读者可查阅有关 Basic 编程的书籍。本节我们将以一个具体实例的制作，帮助读者领会如何在 Visual Basic 程序的具体事件中使用选择与循环。

该程序启动后，由用户在一个组合框中输入几门课的成绩，程序将显示几门课的平均成绩；如果平均分大于 85，显示“Excellent！”，大于 60 显示“Pass！”，否则显示不通过。程序运行结果如图 4-1 所示。

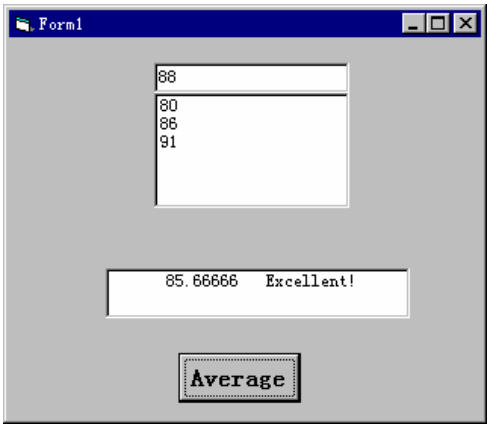

图 4-1 Score 程序运行结果

4.3.1 对象窗口设计

Score 界面需要一个组合框（保存所有成绩）、一个命令按钮（计算平均值），以及一个标签（显示平均成绩）。在 Visual Basic 中创建一个新项目，设计一个包含这三种对象的窗体，属性设置如表 4-1。用户可以在设计窗体界面时加入具有个人风格的设置，使界面更友好。

表 4-1 Score 应用程序的属性设置

对象	属性	设置
Form	Name	Form1
	Caption	Score
	Height	4920
	Width	5730
ComboBox	Name	cboInput
	Enabled	True
	Height	1980
	Visible	True
	Width	1935
Label	Name	Label
	Enabled	True
	Visible	True
CommandButton	Name	btnAvg
	Caption	Average

设计好的窗体如图 4-2。

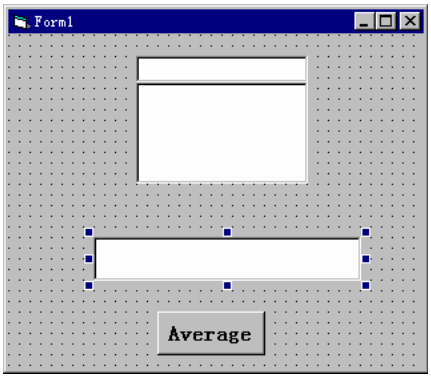

图 4-2 **设计好的窗体**

4.3.2 代码窗口设计

在各对象上双击鼠标，并在代码窗口的过程框中选择指定事件（keypress **或** click）**,为程序编写代码。完整的代码清单如下:**

```
Private Sub cboInput_KeyPress(KeyAscii As Integer)
    If KeyAscii =13 Then
        cboInput.AddItem cboInput.Text
        cboInput.Text =" "
        KeyAscii =0
```

```
    End If
End Sub
'"输入成绩"所用的 Keypress 事件代码
Private Sub btnAvg_Click()
    Sum! = 0
    I% = 0
    Do While I < cboInput.ListCount
        Sum = Sum + Val(cboInput.List(I))
        I = I + 1
    Loop
    Average = Sum / cboInput.ListCount
    Ave$ = Str$(Average)
    If Average > 84 Then
        J = 0
    ElseIf Average > 59 Then
        J = 1
    Else
        J = 2
    End If
'使用 If 结构，将成绩分为不同级别
    Select Case J
    Case 0
        Label.Caption = Ave + "   Excellent!"
    Case 1
        Label.Caption = Ave + "   Pass!"
    Case 2
        Label.Caption = Ave + "   Not Pass!"
    End Select
'使用 Select-Case 结构,根据不同成绩，显示对应等级
End Sub
'求平均成绩所用的 Click 事件代码
```

启动 Score **程序，在组合框的正文部分键入一系列分数，每个值后按** Enter **键。输入完后，按** Average **钮，就可得到程序运行结果。**

4.4 数组

在上例中，我们使用了变量 CboInputList，**它是一个数组。数组就像是超级变量，可以用来存放一系列同一类型的相关数据。例如：可以将一年** 12 **个月份的名字存放在一个一维**

数组中，还可将一个矩阵的所有元素存放在相应的二维数组中。

4.4.1 数组的声明

数组的声明与变量的声明类似，其语法形式为：

Dim 数组名（数组长度）As 类型

在表 4－2 中出了几种声明数组的实例。

表 4－2 声明数组

定义	描述
Dim Month(11) As String	含有 12 个元素的字符串数组： Month(0)到 Month(11)
Dim Profit!(-40 to 15)	含有 56 个元素的单精度数组
Dim A%(40),B%(32),C$(21)	定义三个一维数组
Dim Title$(10 to 25,30 to 64)	二维字符串数组， 第一维的下标范围是 10 到 25 第二维的下标范围是 30 到 64
Global Age(30) As Integer	声明一个全局数组
Static MyArray(55) As Double	声明一个静态数组

给数组元素赋初值，如：

MonthName$（7）=“July”

MyArray（8，7）=3.14

4.4.2 动态数组的创建

动态数组的创建：

如果为公用数组，用 Public 语句声明数组，如下：

```
Public DynArray()
```

如果为模块级数组，在模块级用 Dim 语句声明如下：

```
Dim DynArray()
```

如果为局部数组，在过程中用 Static 或 Dim 语句声明如下：

```
Static DynArray()
Dim DynArray()
```

程序运行时若要改变数组的大小，可以使用 ReDim 指令修改数组界限。

下面是 Redim 的典型使用：

```
Dim Salary（）As Single  '建立一个动态数组
…
ReDim Salary（15，8）  'ReDim 分配动态数组
```

```
…
ReDim Salary（20，11） ‘ReDim 重新分配动态数组
…
Erase Salary          ‘Erase 指令删除数组
```

4.5 本章小结

以上介绍了如何在 Visual Basic 程序设计中使用选择与循环，以及有关数组的简要知识。在以后章节的程序实例中，读者将更多地了解它们的使用。也可以将它们混合使用（即嵌套），如在 For…Next 循环内加上 If…Then 结构，或在 Do 循环内使用数组元素等等。

第五章　过程与事件的基本概念

读者可能已经明白过程就是一组 Visual Basic 语句，有一个惟一的名字，如 Command1_Click、FileOpenProc、Main 等。程序执行过程中的语句来完成指定动作。

过程可以分为事件过程和一般过程。后者又包括通用过程和函数过程。

本章将对 Visual Basic 6.0 中常用的事件响应和过程编写作概要介绍。

5.1　子程序的建立与使用

一个通用过程（又叫子程序）并不和用户界面中的对象相联系，通常一个通用过程不做任何事情，直到一个事件过程详细地告诉它工作时才起作用。使用通用过程的目的是为了在一个地方存放共同使用的指令，如进行一些数学运算，或对某些变量进行共同的操作等。

5.1.1　建立

通用过程名称由用户自己规定，一般遵循变量名的命名规则。

通用过程的声明格式为：

Sub 过程名称（参数 1，参数 2，…）

…程序语句

End Sub

建立窗体级的通用过程，方法是在代码窗口中，将“对象”中的内容置为“通用”，右栏中的内容置为“声明”状态，然后直接在代码框内键入 Sub 过程名（），按 Enter 键，并在其中编写程序代码，如图 5-1 所示。

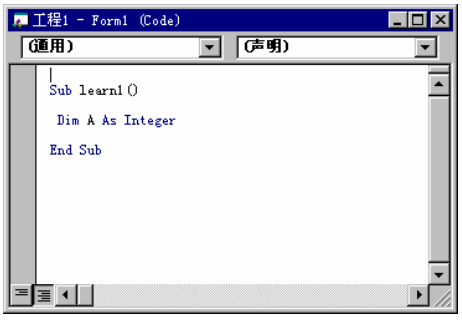

图 5-1　建立通用过程

模块级通用过程，是可被多个窗体中的事件过程所共用的程序模块。其创建方法是：

一、在“工程”菜单中选择“添加模块”，Visual Basic 打开一个模块窗口 Module1，如图 5-2 所示。

二、在其中编写过程。

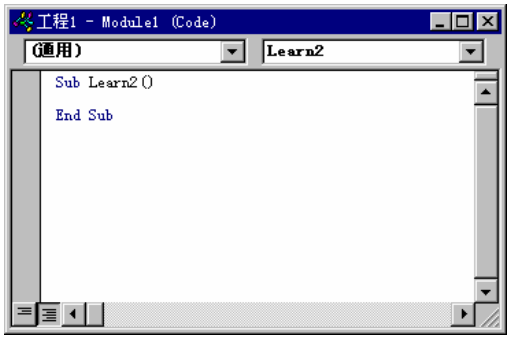

图 5-2　打开模块窗口

5.1.2　使用

当另一个过程想要使用存放在通用过程中的指令时，可通过调用这个过程的名称来实现。例如有如下通用过程：

```
Sub Error（）
    txtDisplay.Text="Error：check your password ! "
End Sub
```

调用时可用下面的形式：

```
Sub cmdstd_Click（）
    Error    '或使用 Call Error
End Sub
```

当要退出一个通用过程时，可使用 Exit Sub **语句来实现。**

5.2　函数过程的定义与调用

函数过程简称函数，是带有返回值的特殊过程。

Visual Basic **包含两大类函数：一类是** VB **本身提供的已封装好的通用函数，包括数学函数、字符串函数等等，需用时可直接调用；另一类是用户自定义函数，其建立方法类似于通用过程的创建，只是将** Sub **改为** Function。

5.2.1　定义

由于函数代表一个值，所以用户可以按一定的数据类型定义函数过程，例如：

```
Function Income（Salary）As Single
const Year=12
    Income=Salary×Year
```

End Function

上面这个函数表明 Income **的值为一个单精度实数**。

一个函数可以被定义为下面任何一种类型：

- **整数**
- **长整数型**
- **单精度实数**
- **双精度实数**
- **货币**
- **字符串**

5.2.2 调用

调用函数的语句格式为：

变量名＝**函数名**（**参数** 1， **参数** 2，…）

例如：

```
Name$=Tom(txtMec.text)
L=Convert(m,n)
```

注意：

变量类型要与函数返回值类型一致，**函数中的参数一定要用括号括起来**。

5.3 参数传递机制

在过程和函数的调用中，**常会遇到参数传递的问题**。Visual Basic **提供两种传递参数的机制**：

- **传址参数**（Passed By Reference）
- **传值参数**（Passed By Value）

5.3.1 传址参数方式

传址调用是子程序的参数与主程序的参数共用内存内的“地址”，所以修改子程序内的参数，会影响主程序的参数 。使用地址传递可提高程序调用及运算效率。传址参数在创建过程或函数时不作特别声明。

下面举例说明：

```
Sub AB（a As Integer, b As Integer）
                    ‘a,b 为传址参数
    a=20
    b=10
End Sub
```

```
Sub Command1_Click（）
    m=200：n=100
    AB m, n                    '调用子程序 AB 后，m 传回 20，n 传回 10
End Sub
```

5.3.2 传值参数方式

这种参数又称形式参数。在传值调用中，只是将主程序参数的内容拷贝一份给子程序的参数，此后子程序参数的改变与主程序无关。

传值参数要用关键字 ByVal **进行声明。**

例如

```
Function Demo（ByVal Index_1 As Integer）As Single
End Function
```

或在调用时给主程序的参数加括号，如将上例中“AB m, n**”改为：**

```
AB （m）,（n）
Print “m =”;  m
Print “n = ”;  n
```

运行后，屏幕上将显示：m = 200　n = 100

m , n **为传值调用，并没有因为调用子程序而改变。**

注意：

数组作为参数进行传递时，只使用数组名和一对空括号，如

```
Sub Pass（My Array（ ）As String）
```

用户自定义类型和对象变量只能通过地址传递。

5.4 事件类型

事件过程与一个特定的控件和事件相关联。当该控件上发生特定的事件时，该过程执行，称为事件响应。如单击 Command1 **按钮时，名称为** Comand1_Click **的过程执行。**

事件类型大致有三类：键盘事件，鼠标事件和程序事件。

一个对象可以对一个或多个事件做出响应，例如，一个命令钮可以对用户单击鼠标钮或按下一个键作出响应。

事件过程名由对象名、下划线和事件名组成，如 CmdExit_Click。Visual Basic **允许用户在不同的窗体中存放同一名称的事件过程。**

Visual Basic **程序设计中常用事见表** 5-1。

表 5-1　常用事件与其作用表

事件	作用
Activate	使一个窗口成为活动窗口

Change	改变一个组合框、列表框、标签、滚动条等
Click	单击鼠标钮
DblClick	双击鼠标钮
Deactivate	使活动窗体变为不活动
DragDrop	单击鼠标钮，移动鼠标指针，再释放鼠标钮
DropDown	显示组合框的表项
GotFocus	按下 Tab 键或单击对象时一个对象被增亮
Keydown	按下一个键
KeyPress	按下并释放一个 ANSI 键
KeyUp	释放一个键
LostFocus	按下 Tab 键或单击另一个对象时该对象不再被增亮
MouseDown	按下鼠标钮
MouseMove	移动鼠标
MouseUp	释放鼠标钮

5.4.1 鼠标响应

在 Visual Basic 编程中，鼠标事件是最常用的事件。主要有以下几种鼠标事件。

一、单击鼠标事件

当用户将鼠标指针置于某对象上，同时按下鼠标按钮并立即释放，便在该对象上引发 Click 事件。

语法格式为：

Sub Form_Click（）

Sub 控件名称_Click（[Index As Integer]）

二、双击鼠标事件

当用户将鼠标指针置于某控件上，同时双击鼠标左键，便在相应控件上引发 DblClick 事件。语法格式为：

Sub Form _DblClick（）

Sub 控件名称_DblClick（[Index As Integer]）

在双击鼠标事件引发的同时，也引发相应对象的单击鼠标事件。

三、按下或释放鼠标事件。

当用户将鼠标指针置于某对象上，同时按下鼠标任一键，便会在该对象上引发 Mouse Down 事件，松开被按下的鼠标键，会引发相应对象的 MouseUp 事件。语法格式为：

Sub Form_ MouseDown （Button As Integer ， Shift As Integer, X As Single, Y As Single）

Sub 控件名称_MouseDown（[Index As Integer,] Button A s Integer, Shift As Integer, X As Single , Y As Single）

在该事件触发时，VB 会自动将鼠标按钮的状态及键盘组合状态和鼠标指针的位置坐标，一起作为参数传递给相应的事件过程。

注意：

在 Click **事件引发的同时，也会触发相应对象的** MouseDown **和** MouseUp **事件**。

四、鼠标移动事件

用户将鼠标指针置于某对象上，同时移动鼠标器，使鼠标指针在该对象区域中移动，便在该对象上触发 MouseMove **事件。语法格式为：**

Sub Form_MouseMove（Button As Integer, Shift As Integer, X As single, Y As Single）

Sub **控制名称**_Mouse Move（[Index As Integer,] Button As Integer, Shift As Integer, X As Single, Y As Single）

只要鼠标指针在某具有控制焦点的对象上移动，便会引发该对象的 MouseMove **事件**。

五、鼠标拖放事件

当用户用鼠标将某一控件拖动到另一控件上，并松开鼠标按钮，便在控件上引发 DragDrop **事件。语法格式为：**

Sub Form_DragDrop（Source As Control, X As Single ,Y As Single）

Sub **控件名称**_DragDrop（[Index As Integer,] Source As Control, X As Single, Y As Single）

该事件过程一般用于将源对象移至目标对象的指定位置。

六、鼠标拖动事件

当用户用鼠标拖曳源对象在目标对象上拖动时，无论鼠标按钮是否松开，都会在目标对象上引发 DragOver **事件。语法格式为：**

Sub Form_DragOver (Source As Control, X As Single ,Y As Single，State As Integer)

Sub **控件名称**_Drag Over （[Index As Integer,] Source As Control, X As Single, Y As Single, State As Integer）

该事件过程一般用于处理当源对象被拖动通过某些特殊区域时，进行一些特殊操作。

5.4.2 键盘响应

有三个键盘事件：

- keyPress **用户按下并松开某个** ASCII **键**
- keyDown **用户按下某个键**
- keyUp **用户释放某个键**

以标签为例，三个键盘事件过程如下：

```
Sub Label1_keyPress（KeyAscii As Integer）
End Sub
Sub Label1_keyDown（KeyCode As Integer，Shift As Integer）
End Sub
Sub Label1_keyUp(KeyCode As Integer，Shift As Integer)
```

其中 keyCode **是被按下或释放的键码，** Shift **指定当前换档键的状态**。

5.4.3 程序响应

程序事件是在 Visual Basic **程序装入**（Load）、**打开**（Open）、**或关闭**（Close）**一个窗体时产生的**。

程序事件分为：

- **窗体事件**
- **改变控制内容事件**
- **控制焦点事件**
- **动态数据交换事件**
- **定时器事件**

限于篇幅，其它内容不能一一介绍。

5.5 本章小结

过程和事件是 Visual Basic **程序最核心的组成部分**。**在以后的编程实战中，读者将近一步详细了解它们的建立与使用**。

每一个过程所必需的、最简单、最重要的事件过程是一个使程序停止运行的事件过程：

```
Subcmd Exit_Click（）
    End
End Sub
```

第三部分　标准控件使用技巧

第一章　MousePointer 属性的应用实例

既然已经有了使用 Visual Basic 6.0 程序设计环境的经验，从这一章开始，就创建一些基于 Windows 的应用程序，在这些实例中，使大家对 Visual Basic 6.0 的了解逐步深入。

这一章，将创建两个 Visual Basic 6.0 的应用程序，用来演示标签 (Label) 的 Mousepointer 属性。第一个例子是通过鼠标单击，进行手动演示；第二个例子是使用 Timer 控件的 Timer 事件，从而实现自动演示。

1.1　实例一

对于 Visual Basic 6.0 中的大部分控件，都有 Mousepointer 属性。如果对窗体中的一个对象设置了 Mousepointer 属性，在用户将鼠标指针移到这个对象上时，鼠标指针会变成这种指定的形状。如果为窗体本身设置了 Mousepointer 属性，除非鼠标指针移到某个已有预定义鼠标指针形状的对象上时，它才会变成这个对象所定义的形状，其它情况下鼠标指针都会变成窗体所指定的形状。

MousePointer 属性的缺省设置为“0-Default”，当选取这一设置时，鼠标指针将根据所处对象不同，自行改变形状。一般情况下，鼠标指针为箭头，而当鼠标指针位于文体框内时，鼠标指针将会自动改变为“I”。

Mousepointer 属性具有 17 种可选值，不同的设置及描述如表 1-1 所示。

这一节，将编一个 Visual Basic 6.0 应用程序，利用标签(Label)的 Click 事件来演示其 Mousepointer 属性。

1.1.1　用户界面的创建

为了实现程序功能，需要在空白窗体上添加一个标签(Label)控件；为了在程序结束时，正确退出，在窗体中添加一个按钮(CommandButton)。界面就这么简单。

一、 添加标签 (Label) 控件

1. 在“文件”菜单中，单击“新建工程”命令或者单击工具栏中的“新建工程”，产生如图 1-1 所示对话框。

2. 单击“确定”，以建立一个标准的 32 位 Visual Basic 6.0 应用程序。

Visual Basic 6.0 为新的程序刷新版面，在屏幕中心显示空白窗体以便创建用户界面，如果在启动 Visual Basic 6.0 后，空白窗体已自动产生，则上面两步可以免去。

3. **单击工具箱中的** Label **控件，将鼠标放到窗体上。**

此时，鼠标指针变成＋字形。＋字形鼠标指针表示可以拖拽鼠标来绘制控件对象。

表 1-1　关于 Mousepointer 属性

设置值	描述
0-Default	缺省的图表光标
1-Arrow	箭头光标
2-Cross	十字光标
3-I-Beam	Ibeam 光标
4-Icon	其中有一个小正方形的正方形光标
5-Size	可调大小的箭头
6-Size NE SW	东北-西南方向的双箭头
7-Size NS	南北向的双箭头
8-Size NW SE	西北-东南方向的双箭头
9-Size W E	东西向的双箭头
10-Up Arrow	向上箭头
11-Hourglass	沙漏光标
12-No Drop	禁止放下的光标
13-Arrow and Hourglass	箭头和沙漏
14-Arrow and Qusetion	箭头和问号
15-Size All	可任意方向调节大小的箭头
99-Custom	利用 MouseIcon 属性自定义鼠标形状

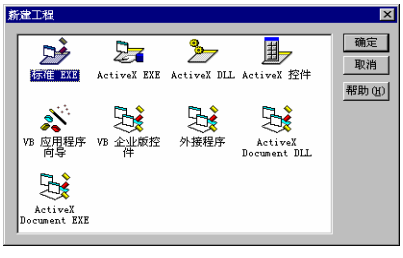

图 1-1　**新建工程对话框**

4. **移动鼠标指针到窗体的左上角，按住鼠标左键，然后向下向右拖拽，产生如图** 1-2 **所示形状时，停止拖拽，松开左键。**

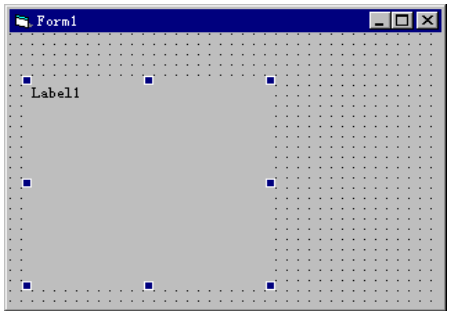

图 1-2　**向窗体添加了标签控件**

这是向窗体添加控件的一般方法，当然，也可以在工具箱中双击控件，把该控件添加到窗体中。

带有控制柄的标签出现在窗体上，这个标签的名字是 Label1。在 Visual Basic 6.0 的设计模式下，可以用鼠标移动标签控件或是用控制柄缩放它们。然而，当程序在运行时，用户不能移动界面部件，除非在程序中改变属性允许这样做。

二、　添加命令按钮(CommandButton)

现在向窗体上添加退出该应用程序的按钮，当用户单击按钮时，退出程序运行。

1. 单击工具箱中的 CommandButton 控件，将鼠标指针移到窗体上，绘制命令按钮；

2. 调节命令钮的大小和位置，得到最终的用户界面如图 1-3 所示。

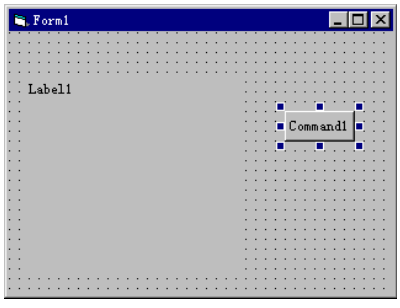

图 1-3　绘制好的用户界面

到现在为止，已经绘制好了用户界面，由于程序功能简单，只用到了标签和命令按钮两个控件。

下面通过设置一些属性来定义界面。

1.1.2　属性的设置

要设置对象属性就会用到该对象的属性窗口，现在我们分别对窗体、标签和按钮进行属性设置。

一、　设置窗体属性

1. 单击窗体，在屏幕右侧出现窗体的“属性”窗口，如图 1-4 所示。

若属性窗口未找到，可按 F4 或选“视图”菜单下的“属性窗口”便可得到。

注意：

“属性窗口”与上下的“工程资源窗口”、“窗体布局窗口”及左边的“窗体编辑窗口”均处于可链接状态，可链接窗口是很有用的，因为它总是可见的，即它们不会完全隐藏在其它窗口后面，要显示连接窗口的更多内容，可以简单地用鼠标拖拽它的边界。

在进行属性设置时，可以双击属性窗口的标题栏，使它漂浮在最前面(不连接)，这时将看到如图 1-5 所示窗口。设置好属性后，可以再次双击属性窗口的标题栏，使“属性窗口”返回到可连接位置。

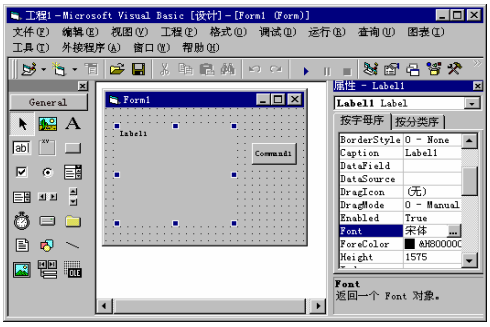

图 1-4　打开窗体的属性窗口

2. **单击属性列表中的** Caption **属性，在其右侧输入“演示** Mousepointer **属性”，回车。**

窗体的 Caption **属性是指在窗体标题栏中显示的内容，往往说明了该窗体的功能，使程序看起来更友好。**

3. **单击属性列表中的** BorderStyle **属性，单击右侧的下拉箭头，从中选择“**1-Fixed Single”，**如图** 1-6 **所示。**

图 1-5　处于不连接状态的属性窗口

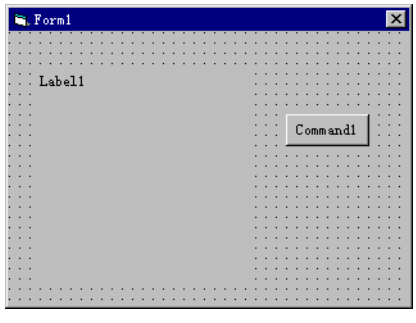

图 1-6　设置了窗体的 BorderStyle **属性**

窗体的 BorderStyle 属性共有六种可选值，分别代表了六种不同的窗体边框类型，改变其设置，可以观察到六种具有不同边框样式的窗体。

二、　设置标签属性

1. 单击标签控件，或者从原属性窗口的对象列表中选择 Label1，得到标签的属性窗口。

2. 在属性列表中选择 BorderStyle 属性，单击右侧向下箭头，选择“1-Fixed Singal”。

标签的 BorderStyle 属性有两种设置值。当它为“0-None”时，表示标签无立体边框，与窗体在同一平面；当它为“1-Fixed Singal”时，使标签产生立体边框。本例中，为使标签区域显得更醒目，选择“1- Fixed Singal”。

3. 在属性列表中选择 Caption 属性，将右侧的“Label1”删除掉，设置为空。

三、　设置按钮属性

1. 在窗体上单击命令按钮或从原属性窗口的对象列表中选择 Command1，得到命令按钮的属性窗口。

2. 在属性列表中选择 Caption 属性，设置为“结束”。

到此为止，已经设置好了窗体、标签及命令按钮的属性，设置好属性的界面如图 1-7 所示。

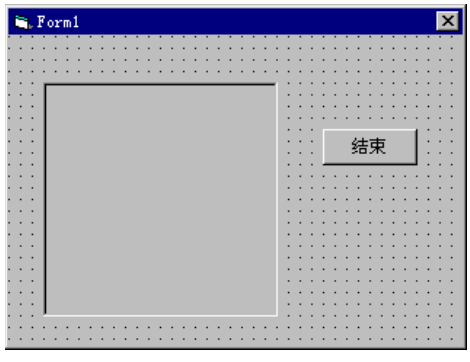

图 1-7　设置好属性的窗体

各对象的属性列表如表 1-2。

表 1-2　对象属性设置

对象	属性	设置值
Form1	BorderStyle Caption	1- Fixed Singal 演示 Mousepointer 属性
Label1	BorderStyle Caption	1-Fixed Singal (Empty)
Command1	Caption	结束

注意：

这里，并没有列出各对象的 Height 属性、Width 属性、Left 属性和 Top 属性的设置，这四个属性主要根据界面的美观和实用进行调节。除非特殊需要，以后也将不再列出。

1.1.3 程序代码的编写

现在开始为程序编写代码，因为当程序运行时，建立的大多数对象已知道怎样工作，所以它们能自动地接受用户和过程的输入。对象固有的功能是 Visual Basic 6.0 最大的优势之一(一旦对象被安放在窗体上且它们的属性已设置好，没有任何附加的程序它们都能运行)。本例的主要部分就是当鼠标单击标签时，鼠标指针形状发生改变。

一、 为标签编写代码

1. 在窗体上双击标签控件或从“工程资源管理器”中单击  图标，打开代码窗口，如图 1-8 所示。

代码窗口的对象列表中包含当前所有对象，过程列表中包含了各对象所包含的事件。

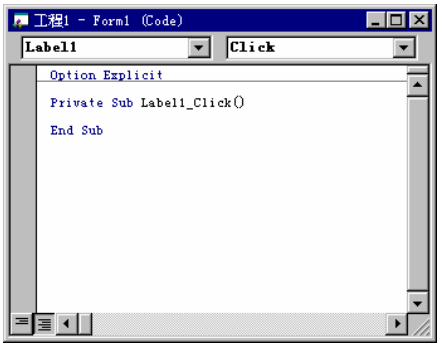

图 1-8 窗体的代码窗口

2. 从对象列表中选择 Label1，从过程列表中选择 Click 事件，这样就得到了 Label1 的 Click 事件过程，即当鼠标单击标签时，该事件发生。

为了告诉 Visual Basic 6.0 当事件过程发生时，Visual Basic 6.0 对象作出何种反应，就要编写事件过程。

3. 编写代码如下：

```
Private Sub Label1_Click()
Static a As Integer
    If a < 15 Then
        a = a + 1
        Label1.MousePointer = a
    Else
        a = a - 14
        Label1.MousePointer = a
    End If
End Sub
```

在该事件过程中，用 Static 语句定义 a 为整型静态变量，这样定义的变量在程序运行期

间一直保持其值不变。本例中，若改用 Dim 语句定义，即

Static a as Integer　改为　Dim a as Integer

则每次鼠标单击标签时，Visual Basic 6.0 会自动给 a 赋值为 0，则标签的 Mousepointer 属性始终设置为 1，无法实现演示功能。当采用 Static 语句定义时，则用户第一次单击标签时，系统会自动给 a 赋值为 1，此时 a=1,且保留下来；第二次单击标签时，由于 a 值已为 1，则通过判断语句，设置标签的 Mousepointer 属性为 2，并且 a=2，且保留下来；这样就可以实现演示功能，使标签的 Mousepointer 属性设置值在 1 到 15 之间循环。

注意：

如果在过程中使用 Static 语句，则应和其它的声明语句(如 Dim)一样将其放在过程开始。

程序中使用了判断语句，当用户有多次单击标签时，使其 Mousepointer 属性在 1~15 间循环。

二、　为命令按钮编写代码

1. 从代码窗口的对象列表中选择 Command1，命令按钮的事件过程变会自动出现在代码窗口。

2. 对命令钮编写代码如下：

```
Private Sub Command1_Click()
    End
End Sub
```

End 为 Visual Basic 6.0 的一条语句，在程序运行时，单击命令按钮会退出程序运行。

现在已经编好了该程序的两个事件过程，写好代码的代码窗口如图 1-9 所示。

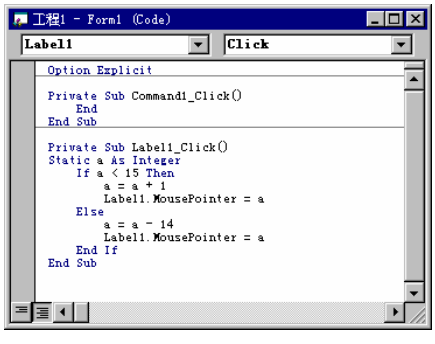

图 1-9　程序代码窗口

注意：

当输入程序代码时，Visual Basic 6.0 自动格式化文本，用不同颜色显示程序的不同部分，以帮助区分不同的元素。当开始键入一个属性时，Visual Basic 6.0 将在一个列表框中显示在使用的对象的所有可获得属性，这样可以双击所需要的属性名或自己键入它。如果 Visual Basic 6.0 显示错误信息，很可能是已经误输入了一条语句，按照上面的程序语句检查输入的第一行，改正它们，然后继续输入，也可删除错误行，再重新开始输入这一行。在 Visual Basic 6.0 中使用的菜单命令和编辑键与在其它 Windows 应用程序中的一样，要了解更多的信息，请参阅 Visual Basic 6.0 联机帮助。

1.1.4 程序的运行

现在，我们已经设计好了应用程序，为了防止误操作或其它原因使窗体丢失，在设计过程中，要养成随时保存窗体的好习惯，特别是在编写大程序时，更要如此。

一、 保存程序

为了便于管理和查找，应专门建一个目录来保存自己编写的应用程序。本例中将窗体保存在“D:/vbfrm/”下，工程保存在“D:/vbpro/”下，可执行文件保存在“D:/vbexe/”下。

1. 从“文件”菜单中选择“演示 Mousepointer 属性.frm 另存为”，弹出“另存为”对话框。

图 1-10 窗体另存为对话框

图 1-11 工程另存为对话框

2. 设定好路径及文件名“D:/vbfrm/演示 Mousepointer 属性”，对话框如图 1-10 所示。

3. 单击确定。

保存窗体时，Visual Basic 6.0 会自动为文件名添加后缀.Frm。

4. 从“文件”菜单中选择“工程另存为”弹出“另存为”对话框。

5. 设定好路径及文件名“D:/vbpro/演示 Mousepointer 属性”，对话框如图 1-11 所示。

6. 单击确定。

保存工程时，Visual Basic 6.0 会自动为文件名添加后缀.vbp。

本例中之所以将窗体和工程分别保存是想说明工程中的窗体是独立的，保存以后，还可以在其它的工程中使用，这在设计多窗体程序时会显得更加明显。

二、 运行程序

程序保存好以后，就可以运行该程序了。

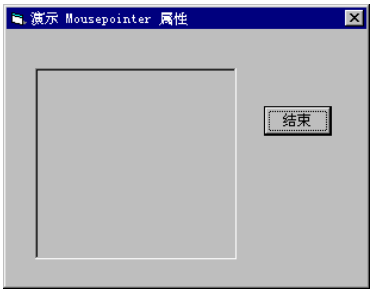

图 1-12　运行时的窗体

1. 单击工具栏中的 ▶ 或者选择“运行”菜单下的“开始”命令，如图 1-12 所示。

2. 将鼠标置于标签框内单击，观察鼠标指针形状变化。

多次单击后，可以看到鼠标指针在 15 种形状之间循环，程序运行达到了预期功能。

祝贺你！你已经成功地编好了一个“演示 Mousepointer 属性”的 Visual Basic 6.0 应用程序，现在，可以结束程序运行了。

3. 单击工具栏中的 ■ ，或者单击窗体上的“结束”按钮，或者从“运行”菜单下选择“结束”命令。

程序试运行符合设计要求，就可以将它生成可执行文件，以便在脱离 Visual Basic 6.0 环境后仍然可以运行。

三、　生成可执行文件

1. 从“文件”菜单中选择“生成工程 1.exe”弹出生成工程对话框;

2. 设定好路径及文件名“D:/vbexe/演示 Mousepointer 属性”，得到如图 1-13 所示对话框;

3. 单击确定。

Visual Basic 6.0 将该程序生成了可执行文件，并保存到指定目录。

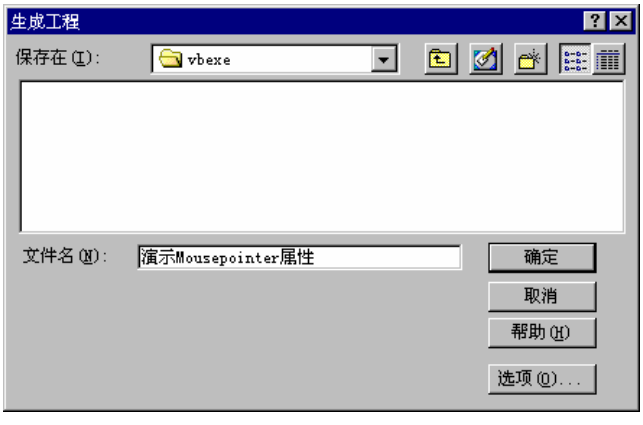

图 1-13　生成工程对话框

1.1.5 说明性标签的添加

退出 Visual Basic 6.0，在“D:/vbexe/”下双击可执行文件“演示 Mousepointer 属性”，可以在 Windows 环境下运行该程序。对于程序设计者来说，他知道用鼠标单击标签框，鼠标指针形状就会改变；但对于其它用户，面对程序界面，他们却不知道如何来“演示 Mousepointer 属性”。因此，程序员需要把程序的界面做得更友好，使所有的用户知道他们该干什么。

现在，我们来改善程序界面，方法就是增加一个说明性标签，告诉用户如何做就可以实现演示功能。

一、 增加说明性标签

1. 启动 Visual Basic 6.0，打开刚才保存好的工程。

本例工程中只包含一个窗体，打开工程和打开窗体区别不大，当工程中包含了多个窗体或模块时，打开工程就打开了工程中的所有窗体和模块，而打开窗体只能打开某一窗体。

2. 在窗体上添加一个说明性标签 Label2，如图 1-14 设置属性。

将该说明性标签的 Font 属性设置为“隶书、规则、小四”。由于说明性文字较多，没有通过设置说明标签的 Caption 属性来实现，而是放在窗体的加载(Load)过程中实现。

二、 添加 Form 的 Load 事件过程

1. 双击窗体的自由区，在代码窗口中得到窗体加载（form_load）事件过程。

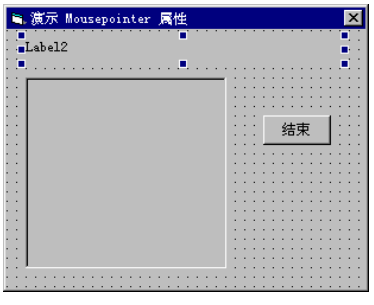

图 1-14 添加了说明性标签

2. 编写代码如下：

```
Private Sub Form_Load()
    Label2.Caption = "用鼠标单击标签，鼠标指针形状改变"
End Sub
```

每次应用程序启动时，窗体的 Load 事件都会自动发生，即该事件过程表示当程序启动时，在标签内显示说明性文字。

三、 运行程序

在运行程序前，先保存窗体文件和工程文件。

1. 单击工具栏中的  或选择“运行”菜单下的“开始”命令，如图 1-15 所示；

图 1-15　运行程序

2. 用鼠标单击标签框，观察鼠标指针形状变化；

3. 结束程序运行。

添加了说明性标签的程序，看起来就友好了许多，使其它用户能够明白如何实现程序功能。

这里对窗体文件的修改并不会影响到刚才已经生成的可执行文件，为了应用修改后的程序，必须将程序重新生成可执行文件，将原来的文件覆盖掉。

1.2　实例二

上一节，我们已经编写了一个程序，利用鼠标单击标签框来演示 Mousepointer 属性。这一节，我们改换另一种方法，当鼠标置于标签框内时，实现自动演示 Mousepointer 属性的功能。

1.2.1　用户界面的创建

由于这个程序与上节中的程序功能大致相同，所以用到的控件也基本一致，只是在这个例程中要增加一个 Timer 控件。创建用户界面的具体操作步骤如下：

1. 启动 Visual Basic 6.0，在“文件”菜单中单击“新建工程”命令，或者单击工具栏中的“新建工程”，建立一个标准的 32 位 Visual Basic 6.0 应用程序；

2. 单击工具箱中的 Label 控件，在窗体中绘制两个标签控件；

3. 单击工具箱中的 CommandButton 控件，在窗体中绘制一个命令按钮；

4. 单击工具箱中的 Timer 控件，在窗体中绘制一个计时器控件。

Timer 控件在程序运行时不可见，可以放置在窗体的任何位置，且 Timer 控件的大小固定，不可增大或缩小，绘制好的用户界面如图 1-16 所示。

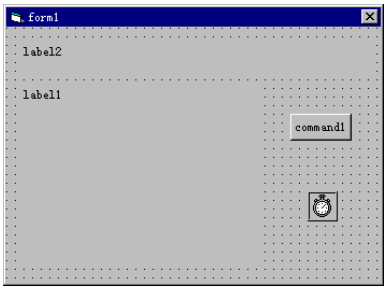

图 1-16　绘制好的用户界面

1.2.2　属性的设置

现在来设置界面对象的属性，以改变对象的外观和行为。

一、　设置计时器属性

计时器的标准属性有：Left、Name **和** Top。**计时器还有其他一些属性，其中一个最重要的属性是** Interval。

此属性用于设置计时器的时间间隔，它的计时单位为 ms(**即千分之一秒**)，**其取值范围是：**0 **到** 65535。**一般说来，计时器事件发生的频率可以从每秒产生** 1000 **个计时器事件到每分钟少于一个计时器事件，当此属性值为** 0 **时，则屏蔽计时器。计时器时间间隔的公式如下：**

T=1000/n

其中，T **是计时器间隔时间，即** Interval **的设置值，**n **是希望每秒发生计时器事件的次数。**

1. **在窗体上单击** Timer **控件，得到其属性窗口，如图** 1-17 **所示；**

图 1-17　Timer 控件的属性窗口

2. **选择** Interval **属性,设置为** 1000。

将 Interval **属性设置为** 1000，**即每隔** 1 **秒钟发生一次计时器事件**(Timer)。

二、　设置其它对象属性

本例中除 Timer **外，其它对象属性(不包括** Height, Width, Left, Top)**与上例中大致相同，因此，属性设置步骤也不在这里赘述，仅将其属性设置列表如下：**

表 1-3　对象属性设置

对象	属性	设置值
Form1	BorderStyle Caption	1-Fixed Single 自动演示 Mousepointer 属性
Label1	BorderStyle Caption	2-Fixed Single (Empty)
Label2	Caption Font	(Empty) 隶书、规则、小四
Command1	Caption	结束
Timer1	Interval	1000

1.2.3　程序代码的编写

本例的主要部分是利用 Timer 控件的 Timer 事件来实现自动演示 MousePointer 属性，即计时器每发生一次计时器事件就改变标签的 Mouse Pointer 属性设置，从而改变鼠标指针形状。

一、　编写计时器事件代码

1. 双击窗体上的计时器控件，代码窗口中自动出现 Timer-Timer 事件过程。

Timer 事件是 Timer 控件最重要的事件，它根据 Interval 属性的设置值而有规律的被激发。使用 Timer 事件时，可用此事件在每次 Timer 控件时间间隔过去之后通知 Visual Basic 6.0 应该做什么。

无论何时，只要 Timer 控件的 Enabled 属性被设置为 True 而且 Interval 属性大于 0，则 Timer 事件以 Interval 属性指定的时间间隔发生。

2. 编写事件过程代码如下：

```
Private Sub Timer1_Timer()
    If a < 15 Then
        a = a + 1
        Label1.MousePointer = a
    Else
        a = a-14
        Label1.MousePointer = a
    End If
End Sub
```

该事件过程的主体与上例相同，也使用判断语句，实现了 MousePointer 属性设置值在 1~15 之间循环。

注意：

与上例不同，这一事件过程中，没有定义变量 a。而是把 a 的定义语句放到“通用”中去，且使用 Dim 定义而非 Static。 如图 1-18，这样定义可以达到与上例相同的效果。

图 1-18　**代码窗口**

二、　编写命令按钮代码

从对象列表中选择 Command1，**编写其** Click **事件过程如下：**

```
Private Sub Command1_Click()
    End
End Sub
```

三、　编写窗体加载代码

从对象列表中选择 Forml，**编写其** Load **事件过程如下：**

```
Private Sub Form_Load()
    Label2.Caption = "将鼠标置于标签框内，鼠标指针形状自动改变"
End Sub
```

该程序段表示，在窗体加载时，在说明标签(Label2)**内显示“将鼠标置于标签框内，鼠标指针形状自动改变”**。

图 1-19　**程序运行时窗体**

1.2.4　程序的运行

1. **在运行程序前，要先保存窗体文件和工程文件**。

2. 单击工具栏中的  按钮，运行程序，如图 1-19 所示。

3. 鼠标置于标签框内，观察鼠标指针形状的变化。

为使鼠标指针形状的变化速度更快，可以将 Timer 控件的 Interval 属性设置得更小。例如 200,这样，每隔 0.2 秒就会激发一次 Timer 事件。在程序运行时，可以看到鼠标指针在标签框内连续迅速地改变形状；

4. 单击工具栏中  按钮或单击窗体上的“结束”按钮结束程序运行。

5. 将程序生成.exe 可执行文件。

1.3　本章小结

本章中编写了两个例子，都是用来演示标签的 Mousepointer 属性。第一例子通过鼠标单击标签框进行手动演示；第二个例子利用计时器的计时器事件实现自动演示。

第二章　各种运算器实例

通过前一章的例程，大家对 Visual Basic 的应用程序已经比较感兴趣了，上一章的两个例程都是演示程序，这一章中我们将编写三个用于数值计算的程序。三个例程的侧重点不同，难度逐渐加深，其中用到了 Visual Basic 的许多内部函数。本章主要知识点：

- 文本框使用
- 控件数组应用
- Visual Basic 的数值计算函数
- SetFocus 方法
- Change 事件
- KeyPress 事件

2.1 实例一：平方运算器

首先，我们编写了一个用于求输入数平方的计算器——平方计算器。

要实现这一功能，就必须解决好以下三个问题：

- 接受输入数，本例中采用文本框(Text Box)接受数据输入；
- 将文本框内的数据转化为数值，进行计算，这要用到 Visual Basic 中的数值转换函数 Val 函数；
- 显示结果。本例中利用文本框显示结果，为了保证计算结果不被随意修改，需要设置显示文本框的 Locked 属性。

2.1.1 用户界面的创建

现在，首先来绘制该程序的用户界面。具体操作步骤：

1. 启动 Visual Basic，在“文件”菜单中单击“新建工程”命令，或者单击工具栏中的“新建工程”建立一个标准的 Visual Basic 应用程序；
2. 单击工具箱中的 Label 控件，在窗体内绘制一个说明性标签；
3. 单击工具箱中的 Label 控件，在窗体内绘制第二个说明性标签；
4. 单击工具箱中的 Text Box 控件，在窗体上添加输入文本框；
5. 单击工具箱中的 Text Box 控件，在窗体上添加显示结果文本框，如图 2-1 所示；
6. 单击工具箱中的 CommandButton 控件，在窗体上添加一个命令按钮；
7. 重复第 6 步，再为窗体添加两个命令按钮；
8. 调节各控件的位置和大小，得到如图 2-2 所示界面。

在调节控件大小及布局时，最好熟练掌握“格式”菜单下的各条命令，当界面复杂，控件元素较多时，为了得到整齐美观的用户界面，使用这些命令会显得更方便。

现在，我们已经绘制好了用户界面，两个标签用于显示说明性文字，两个文本框分别用于接收输入和显示运算结果，三个按命令按钮分别用来复零、转换和结束程序运行。

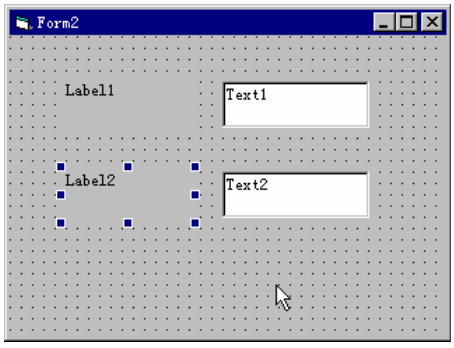

图 2-1　绘制中的界面

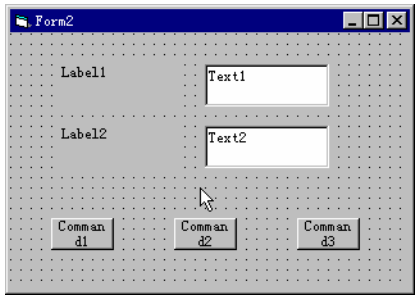

图 2-2　绘制好的界面

2.1.2　属性的设置

设对象属性以改变对象的外观和行为。

一、　设置窗体属性

窗体属性设置如表 2-1 所示：

表 2-1　窗体属性设置

对象	属性	设置值
Form1	（名称）	Form1
	Caption	平方
	MaxButton	False
	Icon	C:\Pr…

Icon 属性用于设置窗体窗口的“控制菜单盒”图标，并在此窗口最小化时显示此图标。在程序设计阶段，就可以通过属性设置窗口指定图标文件，方法如下：

1. 单击窗体的自由区；
2. 选择属性列表中 Icon 属性；

3. **单击属性设置栏右边的省略号按钮，出现“加载图标”对话框，如图** 2-3 **所示**；

图 2-3 **“加载图标”对话框**

4. **在**“Load Icon”**对话框中选定想要显示的图标**；

本例中选取的图标为“C：\Program Files\DevStudio\Vb\Graphics\Icons\Office\Crdfle13”**如图** 2-4 **所示**。

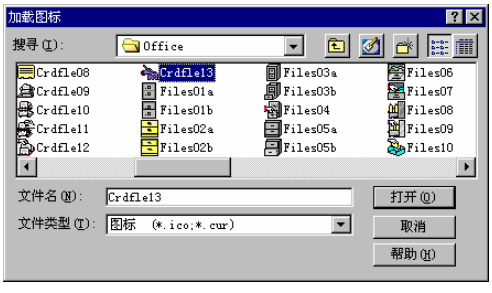

图 2-4 **为窗体添加图标**

5. **单击“打开”按钮，返回编辑界面**。

若想在程序运行时设置 Icon **属性，则要在程序中使用** LoadPicture **函数装入一个图标文件，或将另一个窗体的** Icon **属性赋给该窗体**。

二、 设置标签属性。

两个说明性标签属性设置如表 2-2 **所示**。

表 2-2 标签属性设置

对象	**属性**	**设置值**
Label1	**(名称)**	Label1
	Alignment	2-Center
	Caption	**输入数**
	Font	**宋体、粗体、小三**
Label2	**(名称)**	Label2
	Alignment	2-Center
	Caption	**平方**
	Font	**宋体、粗体、小三**

Alignment 属性设置或返回一个值，决定文本对齐方式。它可以取 0-Left Justify、1-Right Justify 和 2-Center 三个值，分别表示左对齐、右对齐和居中对齐。它还可以应用于 CheckBox 、OptionButton 或 DBGrid 等控件中。对 CheckBox 、 OptionButton 和 TextBox 控件在运行时为只读。为保证 Alignment 属性能够准确工作，Textbox 控件中的 MultiLine 属性必须设置为 True。如果 Textbox 控件中的 MultiLine 设置为 False，则忽略 Alignment 属性。

三、　设置文本框属性

输入输出文本框属性设置如表 2-3 所示。

表 2-3　文本框属性设置

对象	属性	设置值
Text1	(名称) Alignment Font Text	TxtInput 2-center 宋体、粗体、小三 (Empty)
Text2	(名称) Alignment Font Locked Text	TxtOutput 2-center 宋体、粗体、小三 True (Empty)

Text 属性用于存放文本框中的内容。TextBox 控件的 Text 设置值最多可以有 2048 个字符，但是如果 MultiLine 属性设置为 True，此时最大限制大约是 32K。设计时使 Text 属性为空字符串时，则可使正文框空白。

Locked 属性返回或设置一个值，以指定控件是否可被编辑。当它为 True 时，可以滚动和加亮 TextBox 控件中的文本，但不能编辑。程序可以通过改变 Text 属性来改变文本；当它为 False 时，可以编辑 TextBox 控件中的文本。

四、　设置命令按钮属性

三个命令按钮属性设置如表 2-4 所示。

表 2-4　命令按钮属性设置

对象	属性	设置值
Command1	(名称) Caption	Cmdtozero 复零
Command2	(名称) Caption	Cmdchange 转换
Command3	(名称) Caption	Cmdend 结束

现在，已经设置好了所有界面对象的属性，如图 2-5 所示。

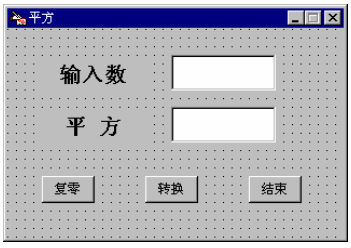

图 2-5　设置好属性的窗体

2.1.3　程序代码的编写

设置好界面属性后，就该编写事件过程来实现程序功能。打开代码窗口，在“通用”中定义：

```
Option Explicit
Dim Inputnumber, Result As Single
```

这样，Inputnumber **和** Result **就可以在该窗体的所有事件过程中使用。**

一、　编写输入文本框事件过程

这一段程序中用到了 TextBox **控制的** Change **事件，该事件当用户向正文框输入新的信息或程序将** Text **属性设置为新值时发生，用户每输入一个字符就发生一次** change **事件。例如，当输入“**1998**”时，就发生四次** change **事件。程序代码如下：**

```
Private Sub TxtInput_Change()
    Inputnumber = Val(TxtInput.Text)
End Sub
```

该程序段将输入文本框的内容转换为数值，赋给变量 Inputnumber**。**

二、　编写“转换”按钮

代码如下：

```
Private Sub Cmdchange_Click()
    Result = Inputnumber ^ 2
    TxtOutput.Text = Result
End Sub
```

语句 Result= Inputnumber^ 2 **表示将** Inputnumber **的二次方赋给** Result；Text=Result，**将运算结果赋给显示文本框的** Text **属性即在显示文本框内显示结果。**

三、　编写“复零”按钮

代码如下：

```
Private Sub Cmdtozero_Click()
    TxtInput.Text = "0"
    TxtOutput.Text = Empty
End Sub
```

其中,TxtOutput= Empty **是设置显示文本框的** Text **属性为空即文本框空白。**

四、编写“结束”按钮

代码如下：

```
Private Sub Cmdend_Click()
    End
End Sub
```

2.1.4　程序的运行

程序设计已经完成后，在运行程序前先保存窗体文件和工程文件。运行程序步骤：

图 2-6　**运行时窗体**

1. **单击工具栏内的**  **按钮，如图** 2-6 **所示；**
2. **在第一个文本框中输入**“14”；
3. **单击“转换”按钮，结果如图** 2-7 **所示；**

图 2-7　**运行结果**

4. **单击“复零”按钮，如图** 2-8 **所示；**

图 2-8　**单击复零按钮**

5. **在第一个文本框中输入其它数字，单击“转换”按钮，观察程序运行结果；**

大家可能会发现，当输入数太大时，显示结果不正确。如图 2-9 所示，这是由于用于显示结果的文本框太小，无法将运算结果全部显示出来。要改善这一点，只要在设计阶段将文本框拉大即可。

6. 单击工具栏中的  按钮，或单击窗体上的“结束”按钮，结束程序运行；

7. 生成可执行文件，保存到指定目录下。

现在，我们已经作好一个平方计算器的应用程序，在计算平方数时就可以实际地使用它，程序中用到的 Val 函数在进行数值转换时会经常用到。

图 2-9　输入数太大时运行结果

2.2　实例二：四则运算器

上节中的平方计算器只能接受单个数据输入，且只能实现单一功能。这一节将要编写的四则运算器，要求输入两个数，并且在加减乘除四种运算中作出选择，然后进行计算，显示计算结果。因为两个输入数之间只能同时具有一种运算关系，所以要使用选项按钮来接受选择，最后程序的界面如图 2-10 所示。

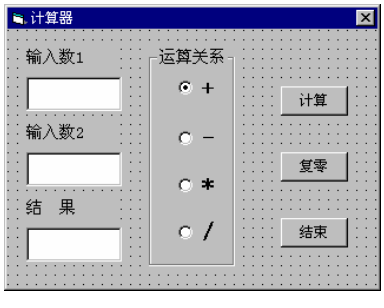

图 2-10　程序最后界面

2.2.1　用户界面的创建

首先会制四则运算器的界面，具体操作步骤如下：

1. 启动 Visual Basic，在“文件”菜单中单击“新建工程”命令，或者单击工具栏中的

“新建工程”，建立一个标准的 Visual Basic 应用程序；

2. 拖动控制柄，增加窗体使其可以放得下所有控件；
3. 单击工具箱中的 Label 控件，在窗体上添加一个说明性标签；
4. 重复第 3 步，再添加两个说明性标签；
5. 单击工具箱中的 Text Box 控件，在窗体上添加一个文本框；
6. 重复第 5 步，再添加两个文本框分别与前面的标签相对应；
7. 单击工具箱中的 CommandButton 控件，在窗体上添加一个命令按钮；
8. 重复第 7 步，再添加两个命令按钮；
9. 单击工具箱中的 Frame 控件，在窗体上添加一个框架；
10. 单击工具箱中的 Option Button 控件，在框架内添加四个选项按钮。

注意：

为了使选项按钮可以同框架一起移动，必须先绘制框架，然后再绘制选项按钮。

11. 调节对象(包括窗体和控件)的大小及布局，得到最终界面如图 2-11 所示。

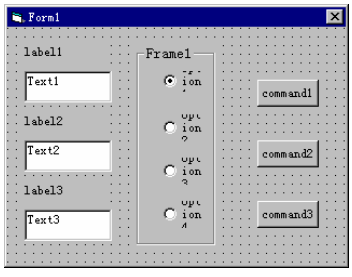

图 2-11　绘制好的界面

2.2.2　属性的设置

本例中用到的对象属性，在前几个例子中都已接触到，现只将其设置列表如下：

表 2-5 对象属性设置

对象	属性	设置值
Form1	(名称) Caption BorderStyle	Form1 计算器 3-Fixed Dialog
Label1	Caption Font	输入数 1 宋体、五号
Label2	Caption Font	输入数 2 宋体、五号
Label3	Caption Font	结果 宋体、五号

Text1	(名称) Alignment Font Text	TxtInput1 2-Center 隶书、粗体、三号 (Empty)
Text2	(名称) Alignment Font Text	TxtInput2 2-Center 隶书、粗体、三号 (Empty)
Text3	(名称) Alignment Font Locked Text	TxtResult 2-Center 隶书、粗体、三号 True (Empty)
Frame1	Caption Font	运算关系 宋体、五号
Option1	（名称） Caption Font Value	Optadd + 宋体、粗体、五号 True
Option2	（名称） Caption Font	Optsub - 宋体、粗体、五号
Option3	（名称） Caption Font	Optmul * 宋体、粗体、五号
Option4	（名称） Caption Font	Optdiv / 宋体、粗体、五号
Command1	(名称) Caption	Cmdcalculate 计算
Command2	(名称) Caption	Cmdtozero 复零
Command3	(名称) Caption	Cmdend 结束

设置好属性的窗体如图 2-10 所示。

在上表中，我们将 Option1 的 Value 属性设置为 True，这样在程序运行时，Option1 将缺省地呈选中状态。同时，其它单选按钮呈未选中状态。

为了突出结果显示，我们可以把“结果”标签的字体设置为粗体，这一步工作我们将在

代码中通过一个叫做对象变量的东西来实现。

2.2.3　程序代码的编写

打开代码窗口，首先在“通用”中声明：

```
Option Explicit
Dim first, second, result As Double
Dim strA，strB As String
Dim lbl As Label
```

第一条语句，定义了三个双精度变量，下面一条语句定义了两个字符串变量，这五个变量可以在该窗体代码窗口的所有事件过程中使用。最后一条语句将 lbl **定义为对象变量，代表窗体中的所有标签（**Label**）控件。**

在程序设计中，可以将某一类对象声明成一个变量，这就称为对象变量。如

```
Dim TextPointer As TextBox          '定义 TextPointer 为对象变量，代表文本框
Set   TextPointer = Text1           '将 TextPointer 指向 Text1
TextPointer.Text =  “对象变量实例”
```

这样，文本框 Text1 **中就会显示为引号之间的内容。对象变量的** Count **属性可以表示对象变量中包含对象的个数。**

一、编写窗体加载事件过程

为了突出结果显示，我们把“结果”标签的字体设置为粗体。代码如下：

```
Private Sub Form_Load()
    Set lbl = Label3
    lbl.FontBold = True
End Sub
```

当然，应用对象变量来引用同一类型对象，还可以在很多场合下发挥其作用。这里只是一个简单的使用对象变量的例子，使大家对此有所了解。

二、为输入文本框编写代码

两个输出文本框 TxtInput1、TxtInput2 **的事件过程如下：**

```
Private Sub TxtInput1_Change()
    strA = TxtInput1.Text
    first = Val(strA)
End Sub

Private Sub TxtInput2_Change()
    strB = TxtInput2.Text
    second = Val(strB)
End Sub
```

这两段代码用到了 TextBox **控件的** Change **事件，程序从其** Text **属性中获取输入数据，**

并转换为数值，以备计算。

三、为计算按钮编写代码

计算按钮的事件过程如下：

```
Private Sub Cmdcalculate_Click()
    If strB = Empty Or strA = Empty Then
        TxtResult.Text = Empty
    Else
        If Optadd.Value = True Then result = first + second
        If Optsub.Value = True Then result = first - second
        If Optmul.Value = True Then result = first * second
        If Optdiv.Value = True Then result = first / second
        TxtResult.Text = result
    End If
End Sub
```

程序中首先判断输入文本框是否为空，如果输入文本框为空，则不进行计算；否则，程序根据用户选取的运算关系，对两个输入数进行操作，并将运算结果输出。

四、为复零按钮编写代码

```
Private Sub Cmdtozero_Click()
    TxtInput1.Text = "0"
    TxtInput2.Text = "0"
    TxtResult.Text = Empty
End Sub
```

五、 为结束按钮编写代码

```
Private Sub Cmdend_Click()
    End
End Sub
```

2.2.4 程序的运行

四则运算器的设计已经完成，现在需要测试其功能。如果存在问题，还要返回设计窗口进行改进。

1. **保存窗体文件和工程文件到指定目录下。**

2. **单击工具栏上的**  **按钮，运行程序，如图** 2-12 **所示。**

大家已经注意到“结果”标签的 FontBold **属性已经重新设置为** True**，这是在窗体的加载过程中完成的。现在，程序默认的运算关系是加法，这是因为绘制界面时这个选项按钮为第一个选项按钮。**

注意：

同一窗体上的所有选项按钮中，总有一个是被默认选中的，这个选项按钮就是创建用户

界面时，程序员在窗体上绘制的第一个选项按钮。

一般情况下，同一个窗体上的所有选项按钮在使用时，只能有一个被选中。但是，如果使用框架控件对其分组，则可以有多个选项按钮同时被选中，但每一组中能有一个被选中，这一点我们在以后的例程中会用到。

图 2-12　运行时的窗体

3. 分别输入两个数“256”和“333”。
4. 单击计算，结果如图 2-13 所示。

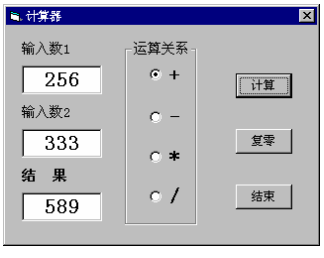

图 2-13　加法计算的结果

5. 保持两个输入数不变，选择运算关系为“*”，单击“计算”按钮，结果如图 2-14 所示。
6. 单击“复零”按钮，观察运行结果。

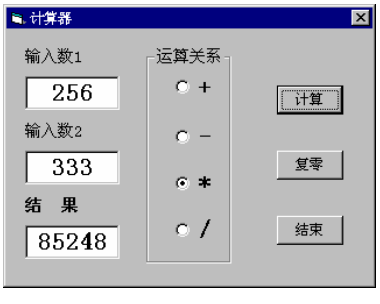

图 2-14　乘法计算结果

7. 改变输入数，重新选择运算关系，观察运行结果。

如果发现程序运行过程中存在问题，还需返回设计阶段进行修改。

8. 单击“结束”按钮，或单击工具栏上的  按钮，结束程序运行。

9. 生成可执行文件.exe，保存到指定目录下。

这一节的四则运算器，需要输入两个数，并选择运算关系后，进行计算，与前一节的平方计算器相比，灵活性增强，功能上也略强大。但是，如果在实际使用中，大家就会发现，这两个计算器都很简单，并且功能太少，界面也不够友好。因此，我们要对其进行改进，下面就实际编写一个函数计算器，来模拟我们平时使用的计算器。二者在界面及功能上都很相似。

2.3 实例三：函数运算器

在这个例子中，我们使用文本框作为计算器屏幕，用来接受用户输入以及显示运算结果，至于其它功能都可使用 CommandButton 控件来实现，程序最终界面如图 2-15 所示。由于界面对象很多，下面我们逐步来绘制程序界面。

2.3.1 添加数字按钮控件数组

这一部分我们又要引入一个新的概念，叫控件数组。控件数组是一组具有共同名称和类型的控件。控件数组类似于我们一般所使用的数组，只不过控件数组表示一组控件，它的元素为控件，而一般数组的元素为数据。在 Visual Basic 中，我们常常把一组功能类似的控件定义为一个控件数组，控件数组中的所有元素都具有相同的名称。读者不禁要问控件数组的不同元素在代码中是如何区别的呢?

事实上，Visual Basic 中的所有对象都有一个属性，叫 Index 属性。这一属性就是专门用来区别同一控件数组中的不同元素，控制数组中的第一个元素，其 Index 属性为 0；控制数组中的第二个元素，其 Index 属性为 1；控制数组中的第三个元素，其 Index 属性为 2；……以此类推。在引用控件数组时，就要以 Index 属性来标识它们，在创建控件数组时，Visual Basic 会自动设置数组元素的 Index 属性。在控件数组中可用到的最大索引值为 32767。

例如

```
Commandl(0).Caption=“o”
Text1(5). Enabled=False
```

第一条语句表示存在一个命令按钮数组 Commandl 将该数组第一元素的 Caption 属性设置为 o。第二条语句表示存在一个文本框数组 Text1 将该数组第六个元素的 Enabled 属性设置为 False。

在设计时，使用控件数组添加控件所消耗的资源比直接向窗体添加多个相同类型的控件消耗的资源要少。当希望若干控件共享代码时，控件数组也很有用。例如，如果创建了一个包含三个选项按钮的控件数组，则无论单击哪个按钮时都将执行相同的代码。使用控件数组时，每个新成员继承数组的公共事件过程。例如，如果窗体上有若干命令按钮，而且每个按钮完成相似的任务，则可创建一个控件数组，使所有命令按钮共享同一个事件过程的代码。

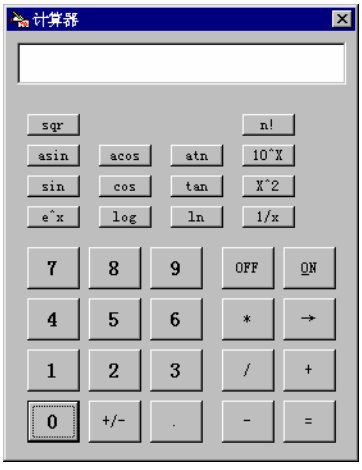

图 2-15　函数计算器

由于控件数组引用方便，且共享同一个事件过程，因此，控件数组的使用范围很广。本例中的 10 个数字按钮就是一个控件数组。下面，我们开始绘制函数计算器界面。具体操作步骤如下：

1. 启动 Visual Basic，在“文件”菜单中单击“新建工程”命令或者单击工具栏中的“新建工程”，建立一个标准的 Visual Basic 应用程序。

2. 参考图 2-15，调整窗体形状。

3. 在工具箱中单击 CommandButton 控件，在窗体上添加一个命令按钮。

4. 调整命令钮大小，如图 2-16 所示。

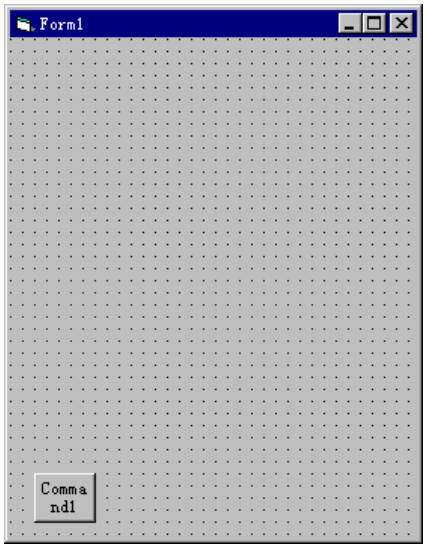

图 2-16　添加了一个命令按钮

5. 单击命令钮，选择“编辑菜单下的“复制”命令或按快捷键 Ctrl+c，复制命令按钮。

6. 选择“编辑”菜单下的“粘贴”命令或按下快捷键 Ctrl+v，弹击如图 2-17 所示对话框。

该对话框询问是否要创建一个控件数组。

图 2-17　创建控件数组

7. 单击“是”，就在窗体左上角得到一个命令按钮，如图 2-18 所示。

这样，我们就成功地创建了一个控制数组，现在数组中只有两个元素，其 Index 属性分别为 0 和 1。

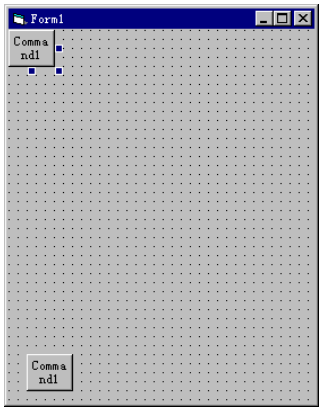

图 2-18　创建了一个控件数组

8. 选择“编辑”菜单下的“粘贴”命令或按下快捷键 Ctrl+v ，继续复制 8 个命令按钮，使按钮总数为 10 个。

注意：

这一步中，Visual Basic 将不再弹出对话框，而是默认用户需要创建控件数组，并自动设置新按钮的 Index 属性。

9. 调节按钮位置，得到如图 2-19 所示的结果。

在摆放按钮时，注意查看各按钮的 Index 属性，使得当按钮如图 2-19 排布时，其 Index 属性从左到右，从上到下依次为 7、8、9、4、5、6、1、2、3、0。这里的 Index 属性设置值将与按钮所代表的数字相对应。

2.3.2　添加运算按钮控件数组

这一组按钮包括四个元素，它也是一个控件数组，操作步骤如下：

1. 单击工具箱中的 CommandButton 控件，在窗体上添加一个命令按钮；
2. 调整按钮大小，使它和前面绘制的 10 个按钮大小一致；

3. **采用绘制数字钮的方法，将刚才添加的按钮复制三个**；
4. **调整按钮位置，得到如图** 2-20 **所示界面**。

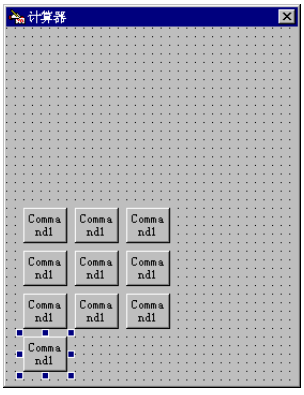

图 2-19　**创建好了数字钮控件数组**

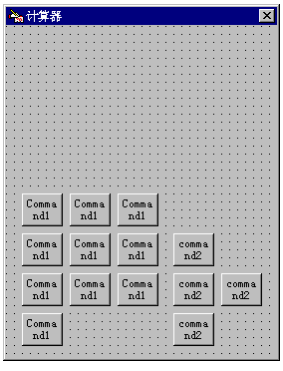

图 2-20　**创建好两个控件数组**

注意第二个控件数组的 Index **置性，图中已注出**。

现在，我们已经绘制好了程序中用到的两个控件数组。主要目的是使大家加深对控制数组的了解，其余控件的添加方法，大家在前面都已经接触到，下面只列出简单添加步骤。

2.3.3　用户界面规划完毕

现在向窗体中添加剩余控件，操作步骤如下：

1. **单击工具箱中的** Text **控件，为窗体添加一个文本框**；
2. **单击工具箱中的** CommandButton **控件，在窗体中添加一个按钮**；
3. **重复第** 15 **步，将剩余按钮全部添加到窗体上**；
4. **调整控件大小及布局，得到如图** 2-21 **所示界面**。

鉴于窗体上按钮整齐一致，因此，在进行该步操作时，建议使用“格式”菜单。

绘制好用户界面，看来我们的函数计算器已经有些眉目了。不过，现在的界面并不能对

我们的操作作出任何响应，实际的计算功能要通过编写代码才能实现。编写代码是开发 Visual Basic 程序的主体，在编写代码前，我们要首先设置对象的属性，特别是当窗体上控件较多时，要尽量使各对象的名称具有明确的意义，便于查找和阅读。

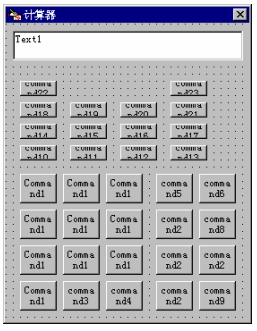

图 2-21　绘制好的计算器界面

2.3.4　属性的设置

现在分别对所有对象的属性进行设置。

一、　设置数字钮属性

表 2-6　数字钮属性设置

对象	属性	设置值
Command1 （数组）	(名称) Font Index	Cmdnumber 宋体、粗体、小四 0 ~ 9

同一控件数组中数组元素的名称相同，Index 属性在创建控件数组时 Visual Basic 自动设置。我们所要做的是核查 Index 属性，与图 2-17 对照，确保各按钮 Index 属性与其 Caption 属性相一致。

大家注意到，在表 2-6 中并没有对 Caption 属性进行设置。这一步设置我们将在窗体加载过程中实现，目的是使大家进一步熟悉控件数组的用法。

表 2-7　对象属性设置（一）

对象	属性	设置值
Form1	(名称) BorderStyle Caption Icon	Frmcalculate 3-Fixed Dialog 计算器 c:\Pr……
Text1	(名称) Alignment Font	Txtscreen 1- Right Justity 宋体、粗体、四号

	Locked MaxLength	True 21
Command2 (**数组**)	（**名称**） Caption Index	Cmdoperater +、一、*、/ 0、1、2、3
Command3	(**名称**) Caption	Cmdsignchange +/-
Command4	(**名称**) Caption	Cmdpoint .
Command5	(**名称**) Caption	Cmdoff OFF
Command6	(**名称**) Caption	Cmdstart ON
Command8	(**名称**) Caption	Cmdback →
Command9	(**名称**) Caption	Cmdequ =

二、 完成属性设置

Maxlength **属性返回或设置一个值，它指出在** TextBox **控件中能够输入的字符是否有一个最大数量。其设置值为一个整数，用来指定在** TextBox **控件中能够输入的最大字符数。**MaxLength **属性的缺省值为** 0，**指出对于用户系统上单行** TextBox **控件来说，最大值不能超过被内存强制建立的值，并且对于多行** TextBox **控件而言，最大值大约为** 32K。**任何大于** 0 **的数表示字符数的最大值。例如，**Text1.Maxlength=7，**表示在文本框** Text1 **中最多只能输入** 7 **个字符。**

表 2-8　函数钮属性设置

对象	**属性**	**设置值**
Command10	(**名称**) Caption	CmdNA e^x
Command11	(**名称**) Caption	CmdCL Log
Command12	(**名称**) Caption	CmdNL Ln
Command13	(**名称**) Caption	Cmdreciprocal 1/x
Command14	(**名称**) Caption	Cmdsin Sin
Command15	(**名称**)	Cmdcos

	Caption	Cos
Command16	(名称) Caption	Cmdtan Tan
Command17	(名称) Caption	Cmdsqv X^2
Command18	(名称) Caption	Cmdarcsin Asin
Command19	(名称) Caption	Cmdarccos Acos
Command20	(名称) Caption	Cmdatn Atn
Command21	(名称) Caption	CmdCA 10^x
Command22	(名称) Caption	Cmdsqr Sqr
Command23	(名称) Caption	Cmdfactorial n!

在表 2-6、**表** 2-7、**表** 2-8 **中列出了该程序所有对象属性的设置，现在程序窗体如图** 7-22 **所示。下面，要分别为各对象编写代码**。

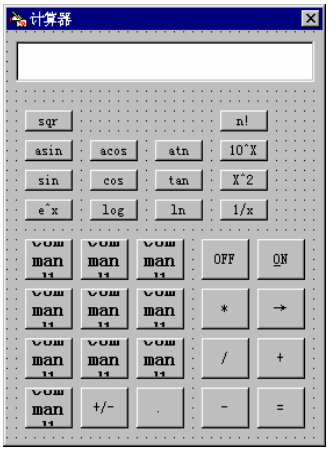

图 2-22 **设置好属性的窗体**

大家可以注意到，数字钮的 Caption **属性仍然为缺省值，下面，我们通过编写代码来设置数字钮属性**。

2.3.5 程序代码的编写

属性设置时，我们没有设置数字钮的 Caption **属性，现在我们把这一步工作拿到代码中**

来做。

1. 双击窗体，打开代码窗口；

2. 首先在“通用”中作如下声明：

```
Dim First, Second, Result, Data As Single
Dim Sign As Integer
```

这两条语句声明的窗体级变量，在下面编写的事件过程中，可以作为中间变量。适当地使用中间变量，会使程序看起来清晰明了。

3. 编写窗体加载事件过程如下：

```
Private Sub Form_Load()
    For Index = 0 To 9
        Cmdnumber(Index).Caption = Cmdnumber(Index).Index
    Next Index
End Sub
```

这段代码中，设置了数字钮的 Caption 属性，请大家注意代码中如何引用控件数组元素的属性。在代码中，将各元素的 Index 属性赋给了 Caption 属性，因此，在前面绘制界面时特别强调了摆放数字钮时要查看各元素的 Index 属性。

4. 单击工具栏中的  按钮，如图 2-23 所示。

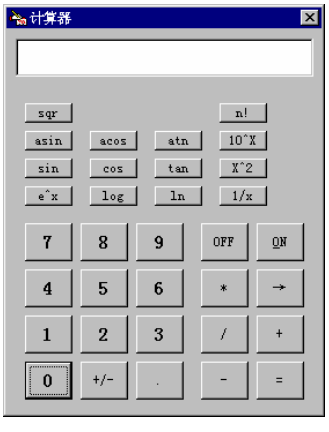

图 2-23　通过代码设置了数字钮的 Caption 属性

可见，这段代码成功地设置了数字钮的 Caption 属性。

2.3.6　数字输入的实现

类似于我们实际生活中用到的计算器，使用时要用按钮进行输入，下面要通过编写代码来实现鼠标输入，即用鼠标单击计算器上的数字按钮，实现输入功能。

大家知道，我们实际使用的计算器，在每次使用时，都要先按一个键，使计算器屏幕复位到“0”，本例中也专门设置一个按钮来实现这一功能。

一、 复位钮代码

```
Private Sub Cmdstart_Click()
    Txtscreen.Text = "0"
End Sub
```

代码很简单，只要实现 Cmdstart **按钮被单击时，使计算器屏幕显示为**“0”，**接下来，对** 10 **个数字钮编写代码，实现数字输入**。

二、 数字钮代码

```
Private Sub Cmdnumber_Click(Index As Integer)
    If Txtscreen.Text = "0" Then
        Txtscreen.Text = CStr(Index)
    Else
        Txtscreen.Text = Txtscreen.Text + CStr(Index)
    End If
End Sub
```

请注意，控件数组的不同元素共享一个事件过程，代码中仅以 Index **属性区分到底哪个数字钮被按下**。

语句中，Cstr **是一个** Visual Basic **函数，它将给定表达式转换为字符型。其中，**Txtscreen.Text = Txtscreen.Text + CStr(Index)，**这条语句中将两个字符串相加，字符串相加只是将两个字符串简单地连接到一起，例如**

Stringsum="Aa"+"134"，**则**

Stringsum="Aa134"

Index **为该事件过程的整型参数**。

三、 输入小数点

```
Private Sub Cmdpoint_Click()
    Txtscreen.Text = Txtscreen.Text + Cmdpoint.Caption
End Sub
```

四、 正负数转变

```
Private Sub Cmdsignchange_Click()
    Dim length As Integer
    Data = Val(Txtscreen.Text)
    length = Len(Txtscreen.Text)
    If Data > 0 Then
        Txtscreen.Text = "-" + Txtscreen
    Else
        Txtscreen.Text = Right(Txtscreen.Text, length - 1)
    End If
End Sub
```

在这个代码中用到了两个字符串函数—Len **函数和** Right **函数**。

(一) Len 函数

Len 函数返回 Long，其中包含字符串内字符的数目，或是存储一变量所需的字节数。

其语法为：

Len (String | Varname)

Len 函数的语法有下面这些部分：

String 为任何有效的字符串表达式。如果 String 包含 Null，会返回 Null。

Varname 为任何有效的变量名称。如果 Varname 包含 Null，会返回 Null。如果 Varname 是 Variant，Len 会视其为 String，并且总是返回其包含的字符数。

(二) Right 函数

Right 函数返回 Variant(String)，其中包含字符串中从右边算起指定数量的字符。

其语法为：

Right (String, Length)

Right 函数的语法有下面的命名参数：

String 为 必要参数。字符串表达式其中最右边的那些字符将被返回。如果 string 包含 Null，将返回 Null。

Length 为必要参数，为 Variant(Long)的数值表达式，指出将返回多少个字符。如果为 0，返回零长度字符串("")。如果大于或等于 String 的字符数，则返回整个字符串。

例如，Name＝“jack”，则，

Len(Name)=4

Right(Name,3)=“ack”

五、 检验输入功能

1. 单击工具栏上的  按钮，运行程序；

2. 单击计算器的 ON 按钮，计算器屏幕显示为“0”；

3. 利用鼠标单击计算器按钮，输入“-456.73014582654”，如图 2-24 所示；

注意：

这是我们允许输入的最大的字符长度为 21 个。

4. 单击工具栏上的  按钮，结束程度运行。

现在，我们已经通过代码实现了计算器的鼠标输入功能，这只是程序的第一步，下面我们通过代码来实现计算功能。

2.3.7 四则运算的实现

要实现四则运算的功能，就会涉及到“+”、“-”、“*”、“/”以及“＝”等五个按钮，其中“+”、“-”、“*”和“/”四个按钮是一个控件数组，只需编写一个事件过程。

其事件过程如下 ：

Private Sub Cmdoperater_Click(Index As Integer)

```
    First = Val(Txtscreen.Text)
    Txtscreen.Text = "0"
    Sign = Index
End Sub
```

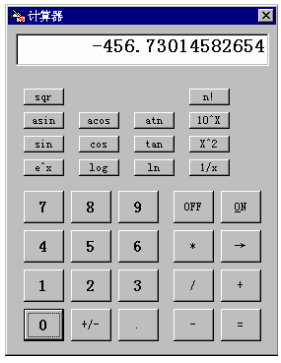

图 2-24　利用数字钮输入

将屏幕内容作为第一个数保存到变量 First 中，并将所击按钮的 Index 值赋给 Sign 变量，这个按钮用来保存用户所选的运算关系，在计算时会用到。

表 2-9　Sign、Index 与运算关系对照表

Sign 变量	Index 属性	运算关系
0	0	+
1	1	-
2	2	*
3	3	/

执行这一段代码后，计算器等待用户输入第二个数，用户将第二个数输入以后，单击“=”按钮，则计算器屏幕显示计算结果。

```
Private Sub Cmdequ_Click()
    Second = Val(Txtscreen.Text)
    Select Case Sign
    Case 0
        Result = First + Second
    Case 1
        Result = First - Second
    Case 2
        Result = First * Second
    Case 3
        Result = First / Second
    End Select
```

```
    Txtscreen.Text = CStr(Result)
End Sub
```

这段代码中，首先将第二个输入数保存到变量 Second **中。以** Sing **的值来区分用户所选的运算关系。代码中用到了** Select-Case **条件分支结构，其语法形式如下：**

```
Select Case VariableName
    Case  X
        Instructions 1
    Case  Y
        Instructions 2
    Case  Z
        Instructions 3
End  Select
```

Visual Basic **根据变量** VarableName **的值来分情况选择执行，如果它的值等于** X**，则执行第一串指令；如果它的值等于** Y**，则的执行第二串指令；如果它的值等于** Z**，则执行第三串指令，根据用户需要检查多少值，决定在分情况选择语句中，使用多少个** Case **行。**

请大家注意理解本段代码中的 Select-Case **条件分支结构的使用方法，在以后的代码中，我们还会用到这一结构。**

添加了这两段代码，已经可以使计算器进行四则运算了，请大家自行测试。

在使用过程中，大家可能会碰到这样的问题：当我们输入一个很大的数时，往往由于只输错一位，就必须全部清除，重新输入。这时候，我们就希望能够将输错的数清除掉，然后继续输入，为了解决这一问题，我们在计算器上设置了退格按钮。

注意：

为了使用户不能直接将光标键入 Txtscreen **中修改其显示内容，我们将它的** locked **属性设置为** True**。因此需要添加退格按钮，用来清除错误字符，而不能直接使用文本框的编辑特征。**

2.3.8　实现退格按钮功能

同我们实际使用的计算器一样，这一按钮只能清除掉输入数的最后一位。

一、　该按钮的代码如下：

```
Private Sub Cmdback_Click()
    Dim length As Integer
    length = Len(Txtscreen.Text)
    If length = 1 Then
        If Txtscreen.Text <> "0" Then
            Txtscreen.Text = "0"
        End If
    Else
```

```
            Txtscreen.Text = Left(Txtscreen.Text, length - 1)
        End If
    End Sub
```

在这个代码中用到了字符串函数 Left **函数**。Left **函数类似于前面用到的** Right **函数**。

二、 Left 函数

Left **函数返回** Variant(String)，**其中包含字符串中从左边算起指定数量的字符**。

其语法为：

Left(String, Length)

Left **函数的语法有下面的命名参数**：

String **为 必要参数**。**字符串表达式其中最左边的那些字符将被返回**。**如果** string **包含** Null，**将返回** Null。

Length **为必要参数，为** Variant(Long)**的数值表达式，指出将返回多少个字符**。**如果为** 0，**返回零长度字符串**("")。**如果大于或等于** String **的字符数，则返回整个字符串**。

三、 检验退格功能

1．**单击工具栏上的**  **按钮，运行程序**；

2．**单击计算器的**“ON”**按钮，使计算器屏幕显示为**“0”；

3．**通过数字钮输入**“123456789093457”，**如图** 2-25 **所示**；

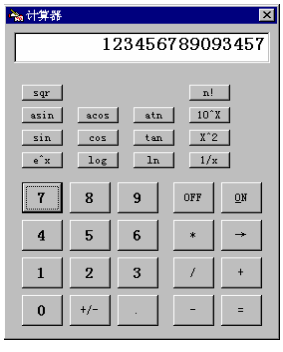

图 2-25 **输入数字**

4．**连续单击五次**“→”**按钮，如图** 2-26 **所示，大家还可以输入负数及小数进行测试，熟悉**“→”**按钮特性**；

5．**单击工具栏的**  **按钮，结束程序进行**。

2.3.9 实现函数计算功能

函数计算器中一个最主要的功能，是可以直接计算一些常用函数的值。本例中，我们设计了 14 **个常用函数，它们分别为** e^x、logx、lnx、1/x、sinx、cosx、tanx、x^2、arcsinx、,arccosx、

atnx、10^x、$\sqrt{x}$ 和 n!，这些函数中，有的是 Visual Basic 自带的函数，可以直接引用；有的则需要自己编写，在这个例子中，我们还用到标准模块，在模块中编写一些函数过程，然后在窗体的代码窗口中引用，现在我们来分别编写各个函数按钮的事件过程。

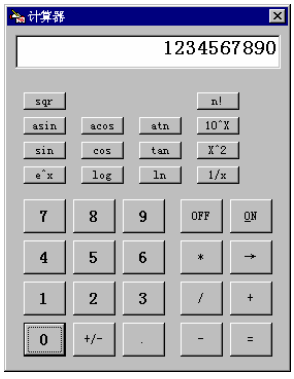

图 2-26　使用退格键

的则需要自己编写，在这个例子中，我们还用到标准模块，在模块中编写一些函数过程，然后在窗体的代码窗口中引用，现在我们来分别编写各个函数按钮的事件过程。

一、　“e^x”按钮代码

```
Private Sub CmdNA_Click()
    Data = Val(Txtscreen.Text)
    Txtscreen.Text = CStr(Exp(Data))
End Sub
```

代码中用到的 Exp 函数为 Visual Basic 的内部函数，用来计算以 e 为底的指数函数。详见附录。

二、　“ln”按钮代码

```
Private Sub CmdNL_Click()
    Data = Val(Txtscreen.Text)
    If Data > 0 Then
        Txtscreen.Text = CStr(Log(Data))
    Else
        Txtscreen.Text = "-E"
    End If
End Sub
```

代码中的 log 函数的 Visual Basic 内部函数，它表示以 e 为底的自然对数，它要求自变量大于零。

三、　“log”按钮代码

```
Private Sub CmdCL_Click()
    Data = Val(Txtscreen.Text)
```

```
        If Data > 0 Then
            Txtscreen.Text = CStr(Log(Data) / Log(10))
        Else
            Txtscreen.Text = "-E"
        End If
    End Sub
```

由于 Visual Basic **没有自带以** 10 **为底的对数函数，因此，我们采用换底公式将自然对数转换成常用对数**。

注意：

代码书写的 log **函数表示自然对数，而计算器按钮上的** log **表示常用对数，** ln **按钮表示自然对数，请读者不要混淆**。

四、 "1/x**" 按钮代码**

```
Private Sub Cmdreciprocal_Click()
    Data = Val(Txtscreen.Text)
    Txtscreen.Text = CStr(1 / Data)
End Sub
```

五、 "x^2**" 按钮代码**

```
Private Sub Cmdsqv_Click()
    Data = Val(Txtscreen.Text)
    Txtscreen.Text = Data ^ 2
End Sub
```

六、 "sin**" 按钮代码**

```
Private Sub Cmdsin_Click()
    Data = Val(Txtscreen.Text)
    Txtscreen.Text = CStr(Sin(Data))
End Sub
```

代码中用到的 Sin **函数为** Visual Basic **的内部函数，用来求正弦值。详见附录**。

七、 "cos**" 按钮代码**

```
 Private Sub Cmdcos_Click()
     Data = Val(Txtscreen.Text)
     Txtscreen.Text = CStr(Cos(Data))
  End Sub
```

代码中用到的 Cos **函数为** Visual Basic **的内部函数，用来求余弦值。详见附录**。

八、 "tan**" 按钮代码**

```
Private Sub Cmdtan_Click()
    Data = Val(Txtscreen.Text)
    Txtscreen.Text = CStr(Tan(Data))
End Sub
```

代码中用到的 Tan **函数为** Visual Basic **的内部函数，用来求正切值。详见附录。**

三个三角函数中，角是以弧度为单位。

九、　“10^x”按钮代码

```
Private Sub CmdCA_Click()
    Data = Val(Txtscreen.Text)
    Txtscreen.Text = CStr(10 ^ Data)
End Sub
```

十、　“atn”按钮代码

```
Private Sub Cmdatn_Click()
    Data = Val(Txtscreen.Text)
    Txtscreen.Text = CStr(Atn(Data))
End Sub
```

代码中用到的 Atn **函数为** Visual Basic **的内部函数，用来求反正切值。详见附录。**

十一、　“sqr”按钮代码

```
Private Sub Cmdsqr_Click()
    Data = Val(Txtscreen.Text)
    Txtscreen.Text = Sqr(Data)
End Sub
```

十二、　“n!”按钮代码

```
Private Sub Cmdfactorial_Click()
    Dim mul, n, i As Integer
    mul = 1
    n = CInt(Txtscreen.Text)
    If n = 0 Then
        Txtscreen.Text = mul
    Else
        For i = 1 To n
            mul = mul * i
        Next i
        Txtscreen.Text = mul
    End If
End Sub
```

请大家注意本段代码中求阶乘的算法。CInt **函数按四舍五入的方法返回一个整型数。详见附录。**

现在，我们已经写好了 12 **个函数按钮的事件过程，剩余的两个函数** arcsinx **和** arccosx**，我们要通过自己编写代码来计算，我们把这两个函数过程放到标准模块中，然后在窗体代码窗口中进行引用。**

2.3.10 标准模块的添加

为了在程序中使用标准模块，必须在工程中添加一个模块，具体操作步骤如下：

1. **选择“工程”菜单中的“添加模块”，得到如图** 2-27 **所示的对话框。**

图 2-27　**添加模块对话框**

2. **单击“打开”，得到标准模块的代码窗口如图** 2-28 **所示。注意“工程资源管理器”中的变化。**

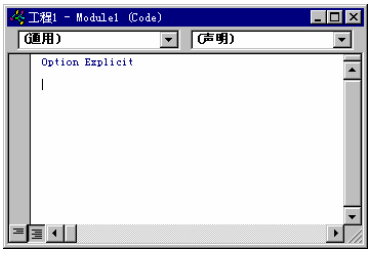

图 2-28　**标准模块代码窗口**

3. **选择“工具”菜单中的“添加过程”，出现如图** 2-29 **所示的对话框。**

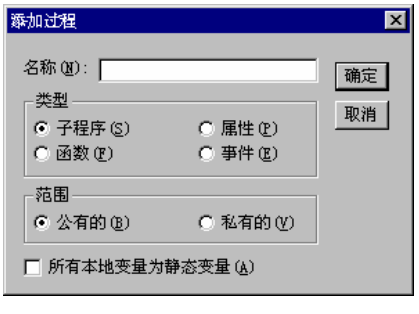

图 2-29　**添加过程对话框**

4. **从类型中选择“函数”，从范围中选择“私有的”，然后输入函数名称“**Arcsin**”，单击“确定”。 标准模块的代码窗口如图** 2-30 **所示。**

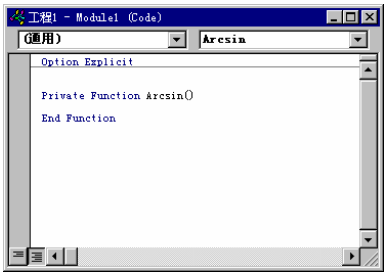

图 2-30　添加了函数过程的标准模块的代码窗口

5. **编写** Arcsin **函数过程如下：**

```
Function Arcsin(x)
    Arcsin = Atn(x / Sqr((-x )*x + 1))
End Function
```

6. **重复第** 3 **到第** 5 **步，编写** Arccos **函数过程如下：**

```
Function Arccos(x)
    Arccos = Atn((-x )/ Sqr((-x )* X + 1)) + 2 * Atn(1)
End Function
```

- **专题讲座：函数过程**

当我们要实现某个特定计算功能时，可以使用函数过程。一般情况，函数过程返回单个值。

(1)　**语法说明**

[Public | Private] Function Functionname [(Argumentlist)] [As type]
[statements]
[Functionname = expression]
End Function

(2)　**描述**

Public　**可选的。表示所有模块中的所有过程都可以访问这个** Function **过程。如果是在包含** Option Private **的模块中使用，则这个过程在该工程外是不可使用的。**

Private　**可选的。表示只有包含其声明的模块中的其它过程可以访问该** Function **过程。**

Functionname　**必需的。函数的名称，遵循标准的变量命名约定。**

Argumentlist　**可选的。代表在调用时要传递给函数过程的参数变量列表。多个变量应用逗号隔开。**

type　**可选的。函数过程的返回值的数据类型，可以是** Byte、Boolean、Integer、Long、Currency、Single、Double、String（**除定长**）、Object、Variant **或任何用户定义类型。任何类型的数组都不能作为返回值，但包含数组的** Variant **可以作为返回值。**

statements　**可选的。在** Function **过程中执行的任何语句组。**

expression　**可选的。**Function **的返回值。**

(3)　**提前退出函数过程**

要提前退出一个函数过程，可以使用 Exit Function **语句。程序接着从调用该函数过程的**

语句之后的语句执行。在函数过程的任何位置都可以有 Exit Function 语句。

(4) 调用函数过程

在表达式中，可以通过使用函数名，并在其后用圆括号给出相应的参数列表来调用一个函数过程。使用下面三种方法都可以正确的调用一个函数过程：

Functionname Argumentlist

Functionname (Argumentlist)

Call Functionname (Argumentlist)

Function 函数过程是一个可以获取参数，执行一系列语句，以及改变其参数值的独立过程。当要使用该函数的返回值时，可以在表达式的右边使用 Function 过程，这与内部函数，诸如 Sqr、Cos 或 Chr 的使用方式一样。例如

Var= Functionname (Argumentlist)

写好函数过程的模块代码窗如图 2-31 所示。在标准模块中写好的函数过程就可以在窗体代码窗口中引用。

7．双击“工程资源管理器”，打开窗体代码窗口分别编写“asin”按钮和“acos” 按钮事件过程。

- “asin”按钮代码

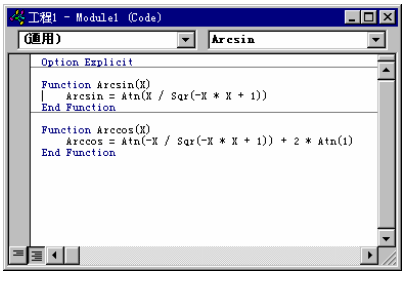

图 2-31 写好函数过程的模块代码窗口

注意：

Arcsinx 函数和 Arccosx 函数不是 Visual Basic 的内部函数，要自己编写代码实现。从 Visual Basic 联机帮助的导出函数中可以查出其计算方法。

由于 arcsinx 函数要求自变量在-1 到+1 之间，并且由 arcsin 的函数过程可以看出，当自变量取±1 时，无法使用标准模块中的函数过程来计算。因此，事件过程中要根据自变量取值范围分情况处理。

代码如下：

```
Private Sub Cmdarcsin_Click()
    Data = Val(Txtscreen.Text)
     If Data < 1 And Data > -1 Then
          Txtscreen.Text = CStr(Arcsin(Data))
     ElseIf Data = 1 Then
```

```
            Txtscreen.Text = 1.5707963
        ElseIf Data = -1 Then
        Txtscreen.Text = -1.5707963
        Else
        Txtscreen.Text = "-E"
      End If
End Sub
```

请注意代码中如何使用函数的返回值。

- **“acos”按钮代码**

```
Private Sub Cmdarccos_Click()
    If Txtscreen.Text <> "-E" Then
        Data = Val(Txtscreen.Text)
        If Data < 1 And Data > -1 Then
            Txtscreen.Text = CStr(Arccos(Data))
        ElseIf Data = 1 Then
            Txtscreen.Text = 0
        ElseIf Data = -1 Then
            Txtscreen.Text = 3.1415926
        Else
            Txtscreen.Text = "-E"
        End If
    End If
End Sub
```

到此为止，我们已经写好了所有的函数计算过程，大家现在可以运行程序，对各个函数计算进行检测。

例如，检测“n!”按钮，具体操作步骤如下：

1．**单击工具栏中**  **按钮，运行程序；**

2．**单击计算器上的**“ON”**按钮；**

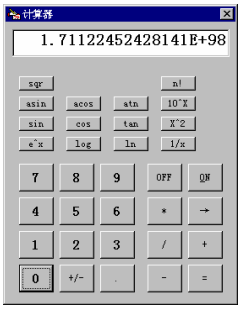

图 2-32　**计算** 69 **的阶乘**

3．用数字钮输入“69”；

4．单击“n!”按钮，结果如图 2-32 所示；

5．输入其他数，检测其他函数按钮的功能；

6．单击工具栏中的  按钮，结束程序运行。

要对所有的函数过程进行检验，确保其正确性。

2.3.11 实现结束按钮功能

“OFF”按钮代码：

```
Private Sub Cmdoff_Click()
    End
End Sub
```

单击该按钮，结束程序运行。

到现在为止，可以说我们的函数计算器，已经编写完成。可实际使用时，若要输入的数很多时，大家就会发现，单靠鼠标单击数字钮进行输入，其速度很慢，而且很多人已经习惯于用键盘进行输入，因此，我们打算对程序作进一步改进，以实现部分键盘操作。

2.3.12 实现键盘输入功能

当用户按下一个 ANSI 键时，就会发生 Keypress 事件，本例中我们就使用 Keypress 事件来接受键盘输入，该事件返回所按键的 ASCII 码。我们可以使用 KeyDown 和 KeyUp 事件过程来处理任何不被 KeyPress 识别的击键，诸如：功能键、编辑键、定位键以及任何这些键和键盘换档键的组合等。与 KeyDown 和 KeyUp 事件不同的是，KeyPress 不显示键盘的物理状态，而只是传递一个字符。

代码如下：

```
Private Sub Cmdnumber_KeyPress(Index As Integer, KeyAscii As Integer)
    Select Case KeyAscii
    Case 48 To 57                                   '数字键被按下
        Char = Chr(KeyAscii)
        Cmdnumber(Char).Value = True
    Case 61                                         '“=”被按下
        Cmdequ.Value = True
    Case 43                                         '“+” 被按下
        Cmdoperater(0).Value = True
    Case 45                                         '“-” 被按下
        Cmdoperater(1).Value = True
    Case 42                                         '“*” 被按下
```

```
        Cmdoperater(2).Value = True
    Case 47                                          ‘“/” 被按下
        Cmdoperater(3).Value = True
    Case 46                                          ‘“.” 被按下
        Cmdpoint.Value = True
    Case 79 Or 111                                   ‘“o”或“O” 被按下
        Cmdstart.Value = True
    End Select
End Sub
```

设置一个按钮的 Value **属性为** True，**就表示该按钮被按下时，引发该按钮的** Click **事件。例如，**Cmdequ.Value=True，**就表示“=”按钮被按下，从而引发** Cmdequ_Click **事件过程。**

2.3.13　SetFocus 方法的使用

只有当某一对象获得焦点时，按下一个键才会引发其 KeyPress **事件。因此，为了能够实现键盘输入，我们必须始终保证焦点在** Cmdnumber **上（因为我们只对** Cmdnumber **按钮编写了** KeyPress **事件）。在属性窗口中，设置** Cmdnumber(0)**的** TabIndex **属性为“0”，这可以保证程序启动时，焦点位于** Cmdnumber(0)。**在程序运行中，当用户用鼠标单击了其它按钮后，我们必须使焦点重新回到** Cmdnumber **上，这样才能使用键盘继续操作。**

注意：

（**一**）TabIndex **属性用来设定当用户按下** Tab **键时，焦点移动的顺序。**

缺省情况下，在窗体上画控件时 Visual Basic **会分配一个** Tab **键顺序，但** Menu、Timer、Data、Image、Line **和** Shape **控件除外，这些控件不包括在** Tab **键顺序中。运行时，不可见或无效的控件以及不能接收焦点的控件（**Frame **和** Label **控件）仍保持在** Tab **键顺序中，但在切换时要跳过这些控件。**

每个新控件都放在 Tab **键顺序的最后。如果用改变控件的** TabIndex **属性值来调整缺省** Tab **键顺序，**Visual Basic **会自动对其它控件的** TabIndex **属性重新编号，以反映出插入和删除操作。可以在设计时用属性窗口或在运行时用代码来作改变。**

（**二**）Click **事件会改变焦点位置，例如，当单击了**“ON”**按钮后，焦点就会移到该按钮上；**KeyPress **事件不改变焦点位置，例如，当焦点位于** Cmdnumber(0)**上时，我们按下**“+”**键，虽然会引发**“+”**按钮的** Click **事件，但焦点仍位于** Cmdnumber(0)**上。**

为了使某一对象获得焦点，就要用到 SetFocus **方法。**

SetFocus **方法将焦点移至指定的控件或窗体。语法：**

Object.SetFocus

Object **所在处代表任一拥有** SetFocus **方法的** Visual Basic **对象。对象必须是** Form **对象、**MDIForm **对象或者能够接收焦点的控件。调用** SetFocus **方法以后，任何的用户输入将指向指定的窗体或控件。**

焦点只能移到可视的窗体或控件。因为在窗体的 Load **事件完成前窗体或窗体上的控件是不可视的，所以如果不是在** Form_Load **事件过程完成之前首先使用** Show **方法显示窗体的话，不能使用** SetFocus **方法将焦点移至正在加载的窗体上。**

也不能把焦点移到 Enabled **属性被设置为** False **的窗体或控件。如果已在设计时将** Enabled **属性设置为** False**，必须在使用** SetFocus **方法使其接收焦点前将** Enabled **属性设置为** True。

由于 Click **事件会改变焦点位置，因此，我们要在所有对象（**Cmdnumber **除外）的** Click **事件过程中添加一条语句，使得相应的按钮被单击后，焦点重新回到** Cmdnumber(0)。

注意：

焦点只要处于 Cmdnumber **数组的任一元素上，都可以引发** Cmdnumber **的** Keypress **事件，这里我们统一使其焦点位于** Cmdnumber(0)**上。**

我们将修改后的代码编写如下，供大家参考，不作详细说明。

```
Private Sub Cmdsignchange_Click()                    ‘“+/-”按钮代码
    Dim length As Integer
    Data = Val(Txtscreen.Text)
    length = Len(Txtscreen.Text)
    If Data > 0 Then
        Txtscreen.Text = "-" + Txtscreen
    Else
        Txtscreen.Text = Right(Txtscreen.Text, length - 1)
    End If
    Cmdnumber(0).SetFocus
End Sub
Private Sub Cmdpoint_Click()                         ‘“.”按钮代码
    Txtscreen.Text = Txtscreen.Text + Cmdpoint.Caption
    Cmdnumber(0).SetFocus
End Sub
Private Sub Cmdoperater_Click(Index As Integer)      ‘运算关系按钮代码
    First = Val(Txtscreen.Text)
    Txtscreen.Text = "0"
    Sign = Index
    Cmdnumber(0).SetFocus
End Sub
Private Sub Cmdequ_Click()                           ‘“＝”按钮代码
    Second = Val(Txtscreen.Text)
    Select Case Sign
    Case 0
        Result = First + Second
    Case 1
```

```
End Sub
Private Sub CmdNL_Click()                          '“ln”按钮代码
    Data = Val(Txtscreen.Text)
    If Data > 0 Then
        Txtscreen.Text = CStr(Log(Data))
    Else
        Txtscreen.Text = "-E"
    End If
    Cmdnumber(0).SetFocus
End Sub
Private Sub Cmdreciprocal_Click()                  '“1/x”按钮代码
    Data = Val(Txtscreen.Text)
    Txtscreen.Text = CStr(1 / Data)
    Cmdnumber(0).SetFocus
End Sub
Private Sub Cmdsin_Click()                         '“sin”按钮代码
    Data = Val(Txtscreen.Text)
    Txtscreen.Text = CStr(Sin(Data))
    Cmdnumber(0).SetFocus
End Sub
Private Sub Cmdcos_Click()                         '“cos”按钮代码
    Data = Val(Txtscreen.Text)
    Txtscreen.Text = CStr(Cos(Data))
    Cmdnumber(0).SetFocus
End Sub
Private Sub Cmdtan_Click()                         '“tan”按钮代码
    Data = Val(Txtscreen.Text)
    Txtscreen.Text = CStr(Tan(Data))
    Cmdnumber(0).SetFocus
End Sub
Private Sub Cmdsqv_Click()                         '“x^2”按钮代码
    Data = Val(Txtscreen.Text)
    Txtscreen.Text = Data ^ 2
    Cmdnumber(0).SetFocus
End Sub
Private Sub Cmdarcsin_Click()                      '“asin”按钮代码
    Data = Val(Txtscreen.Text)
    If Data < 1 And Data > -1 Then
```

```
            Txtscreen.Text = CStr(Arcsin(Data))
        ElseIf Data = 1 Then
            Txtscreen.Text = 1.5707963
        ElseIf Data = -1 Then
            Txtscreen.Text = -1.5707963
        Else
            Txtscreen.Text = "-E"
        End If
          Cmdnumber(0).SetFocus
End Sub
Private Sub Cmdarccos_Click()                          ‘“acos”按钮代码
        If Txtscreen.Text <> "-E" Then
            Data = Val(Txtscreen.Text)
            If Data < 1 And Data > -1 Then
                Txtscreen.Text = CStr(Arccos(Data))
            ElseIf Data = 1 Then
                Txtscreen.Text = 0
            ElseIf Data = -1 Then
                Txtscreen.Text = 3.1415926
            Else
                Txtscreen.Text = "-E"
            End If
      End If
      Cmdnumber(0).SetFocus
End Sub
Private Sub Cmdatn_Click()                             ‘“atn”按钮代码
        Data = Val(Txtscreen.Text)
        Txtscreen.Text = CStr(Atn(Data))
        Cmdnumber(0).SetFocus
End Sub
Private Sub CmdCA_Click()                              ‘“10^x”按钮代码
        Data = Val(Txtscreen.Text)
        Txtscreen.Text = CStr(10 ^ Data)
        Cmdnumber(0).SetFocus
End Sub
Private Sub Cmdsqr_Click()                             ‘“sqr”按钮代码
        Data = Val(Txtscreen.Text)
        Txtscreen.Text = Sqr(Data)
```

```
    Cmdnumber(0).SetFocus
End Sub
Private Sub Cmdfactorial_Click()                    '“n!”按钮代码
    Dim mul, n, i As Integer
    mul = 1
    n = CInt(Txtscreen.Text)
    If n = 0 Then
        Txtscreen.Text = mul
    Else
        For i = 1 To n
            mul = mul * i
        Next i
        Txtscreen.Text = mul
    End If
    Cmdnumber(0).SetFocus
End Sub
```

2.3.14 程序编写完毕

为了使我们的程序更完备一些，请注意以下几个细节。

在属性窗口，将“ON”按钮的 Caption **属性修改为“&ON”。“&”表示在其后的字母下加一条下划线，得到的按钮为** 

。

这样可以提示用户，通过按下“o”或“O”键可以代替用鼠标单击该按钮。这一点我们在 Cmdnumberd **的** KeyPress **事件过程中已经考虑到。**

有时候，用户可能会不经意地将焦点置于 Txtscreen **上，这样就无法实现键盘输入（因为焦点不在** Cmdnumber **上，按下一个键无法引发它的** KeyPress **事件），为防止这一点，增加如下事件过程：**

```
Private Sub Txtscreen_GotFocus()
    Cmdnumber(0).SetFocus
End Sub
```

当用户将光标置于计算器屏幕内时，就会引发 GotFocus(**获得焦点**)**事件。该段代码使焦点回到** Cmdnumber(0)**上。**

这里只列出这三点，事实上，要想使该计算器功能更趋完善，并且有足够高的稳定性，还要在许多方面作出改进。例如，有些按钮没有相应的键盘操作，某些函数当输入数太大时会造成溢出等，读者可以对该程序作进一步的完善，限于篇幅，书中没有写出。

2.3.15　程序的运行

对程序作了上述一些改进后，现在，我们来运行程序，主要检验新加的键盘操作功能。

1. **单击工具栏中**  **按钮，运行程序。**
2. **按“o”或“O”键（大小写），如图** 2-33 **所示。**

注意焦点的位置应在数字钮“0”上。

3. **用键盘输入**“3758.126548”，**如图** 2-34 **所示。**
4. **按“*”键。**
5. **用键盘输入**“6262.14”。
6. **按“=”键，结果如图** 2-35 **所示。**

在这次计算中，我们完全使用了键盘操作，其运行结果同鼠标操作是一样的。但是，

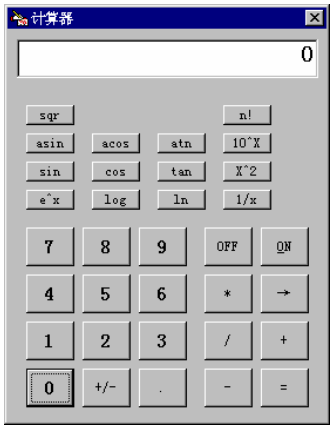

图 2-33　**使用键盘复位**

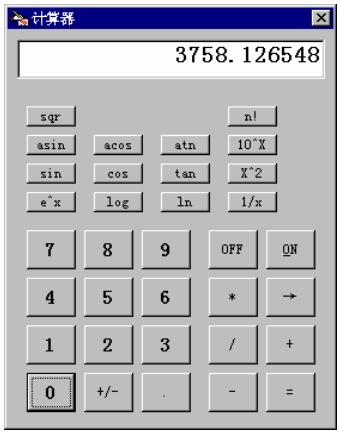

图 2-34　**使用键盘输入数字**

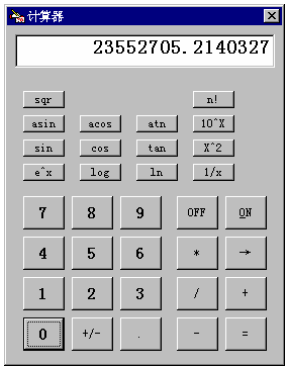

图 2-35　按“=”得到计算结果

如果要用到函数按钮、“+/-”按钮及“→”按钮时，仍然要使用鼠标操作。由于我们保证焦点始终在 Cmdnumber(0)**上，鼠标操作后，可以直接进行键盘输入，当然，可以使用键盘操作实现的功能，都可以使用鼠标进行操作。**

7. **重复多次检验，确保程序运行正确。**

8. **单击工具栏中的**  **按钮，或计算器上的“OFF”按钮结束程序运行。**

现在，我们的函数计算器已经大功告成，记得一定要保存工程文件，因为工程中包含了一个窗体，一个模块，作为一个整体保存起来，便于以后使用和修改。窗体文件和模块文件都可以分别单独保存，需要时，可以在其他工程中引用。

最后，将该程序生成一个可执行的文件.exe**，并保存在指定目录下，这样，需要的时候就可以把它调出来使用。**

2.4　本章小结

这一章，我们编写了三个例子，前面两个例子比较小，只作为一个铺垫，使大家逐步入门；最后的函数计算器，较复杂，用到的知识点也比较多，对于新的知识点，使用时都作了大体介绍，要想了解得更详细，可以参看联机帮助。

第三章　框控件与按钮控件的使用

一般情况下，大多数的用户需要将选的东西清晰地排列在他们的眼前，这样他们可作出各种猜测。事实上，友好地用户界面都采用了这种方法。

Visual Basic 为用户提供了多种处理选择的对象：复选框(CheckBox)、选项按钮(OptionButton)、列表框(ListBox)和组合框(ComboBox)。复选框让用户作出一个或多个选择，选项按钮让用户只能选择一个，列表框和组合框让用户可以作出多重选择。

3.1　复选框控件介绍

复选框很少单独出现，大多数情况下，用户可以看到两个或多个复选框一起出现。当用户选定复选框时，复选框将显示选定标记，使用框架对复选框进行分组，可以直观地给用户一种分组效果。

Value 属性是复选框控件最重要的一个属性，当其值为 0-Unchecked 时，表示没有检查（缺省值）；当其值为 1-Checked 时，表示已检查；当其值为 2-Grayed 时，复选框变灰（变暗）。

用户可以单击复选框控件来指定它为已检查或未检查状态，然后通过检测复选框状态来编写代码以实现某些操作。

在缺省状态下，复选框处于未选中状态，要想预先设定某些复选框为选定状态，可以在 Form _Load 或 Form_ Initialize 过程中，将 Value 属性设置为 1-Checked。

当复选框处于可选状态时，用户单击复选框，将触发其 Click 事件。可以在 Click 事件过程中编写代码，根据复选框的状态执行某些操作。

注意：

复选框不支持双击事件(DblClick)。当用户双击复选框时，将看作两次单击事件 (Click)来处理。

在设置复选框的 Caption 属性时，可以使用“与(&)”字符，在“&”后的字母下加一条下划线，这样就可以使用快捷方式在选定和未选定之间切换。如：Check1.Caption =“&Black”，则控件如图 3-1 所示。

图 3-1　在 Caption 属性中使用“&”字符

这时可以使用 Alt+B 混合键来切换复选框的选择。

3.2 选项按钮控件介绍

选项按钮一般成组出现，显示一系列的选项供用户选择。选项按钮有时也称单选钮，这是因为选项按钮同复选框有一个关键的区别：当用户选择某一个选项按钮时，同组中其它选项按钮会自动失效，即一组选项按钮中只有一个能够被选中，而同组复选框中可以有多个同时被选中。

我们可以用三种方法对选项按钮进行分组：

1. 将一系列选项按钮放在框架(Frame)内；
2. 将一系列选项按钮放在图片框(PictureBox)内；
3. 窗体内的选项按钮也将组成一个系列。

使用上述方法，就可以创建多组选项按钮，运行时，用户在每组选项按钮中只能选定一个选项按钮，在下面的例子中大家会看到这一点。

注意：

所有直接添加到窗体的选项按钮成为一组选项按钮，要添加附加按钮组，应把按钮放置在框架或图片框中，如图 3-2 所示。

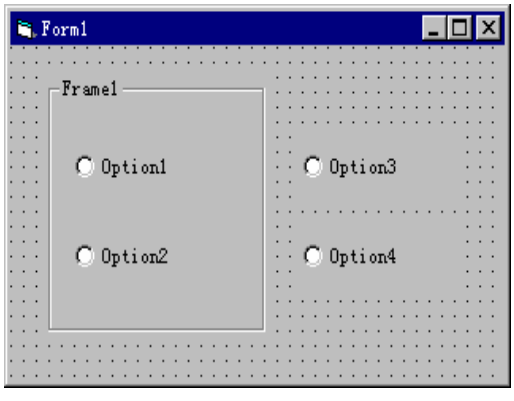

图 3-2 将选项按钮分为两组

要用框架或图片框对选项按钮进行分组，必须先绘制框架或图片框，然后在其内部绘制选项按钮。这样，框架或图片框及其内部控件可作为一个整体移动。

选项按钮的 Value 属性指出是否选定了此按钮。当其值为 True 时，表示选定，当其值为 False 时，表示未选定。Value 属性在设计阶段和运行阶段都可以进行设置。

选定选项按钮时将触发其 Click 事件，编写相应代码,可以根据程序功能实现某些操作。

3.3 程序实例

本例使用选项按钮和复选框来演示字体、字号、颜色及是否加下划线与删除线。程序主要体现了利用选项按钮和复选框进行选择的不同特点，以及使用框架对选项按钮进行分组。

绘制界面如图 3-3 **所示**。

使用框架将选项按钮分为三组，绘制界面时需先画框架，再添加选项按钮，每组选项按钮都是一个控件数组，设置对象属性如表 3-1。

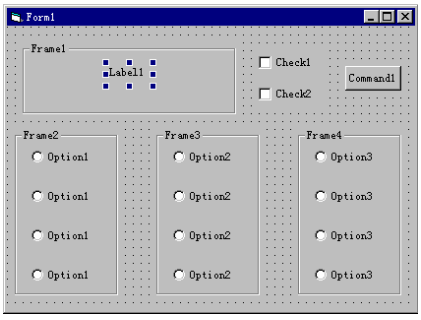

图 3-3　**绘制好的界面**

表 3-1　属性设置表

对象	**属性**	**设置值**
Form1	Caption Icon MaxButton	Font **设置** （Icon） False
Frame1	Caption	**文本示例**
Frame2	Caption	**字体样式**
Frame3	Caption	**字体大小**
Frame4	Caption	**字体颜色**
Label1	Alignment Autosize Backcolor Borderstyle	2—Center True **白色** 1-Fixed Single
Check1	Caption	**删除线**
Check2	Caption	**下划线**
Command1	Caption	**确定**
Option1 (**数组**)	Index-Caption	0-**楷体** 1-**隶书** 2-**黑体** 3-**宋体**
Option2 (**数组**)	Index-Caption	0-9 1-12 2-16 3-20

Option3 (数组)	Index-Caption	0-红色 1-蓝色 2-绿色 3-黄色

这里用到了标签控件的 Autosize 属性，在缺省情况下，当标签的 Caption 属性文本超过标签宽度时，文本会自动换行，而且在超过控件高度时，超出部分将被剪掉。为使标签能够自动调整以适应内容多少，可将 Autosize 属性设置为 True。这样标签可水平扩充以适应 Caption 属性内容的大小。

为使 Caption 属性的内容自动换行并垂直扩充，应将 Wordwrap 属性设置为 True；设置好属性的界面如图 3-4 所示。

图 3-4　设置好属性的窗体

打开代码窗口，将如下代码添加到 Check1_Click 事件过程中。

```
Private Sub Check1_Click()
    If Check1.Value = 1 Then
        Label1.FontStrikethru = True
    Else
        Label1.FontStrikethru = False
    End If
End Sub
```

打开代码窗体，将如下代码添加到 Check2_Click 事件过程中。

```
Private Sub Check5_Click()
    If Check6.Value = 1 Then
        Label1.FontUnderline = True
    Else
        Label1.FontUnderline = False
    End If
End Sub
```

打开代码窗体，将如下代码添加到 Option1_Click 事件过程中。

```
Private Sub Option1_Click(Index As Integer)
```

```
    Select Case Index
        Case 0
        Label1.FontName = "楷体"
        Case 1
        Label1.FontName = "隶书"
        Case 2
        Label1.FontName = "黑体"
        Case 3
        Label1.FontName = "宋体"
    End Select
End Sub
```

打开代码窗体，将如下代码添加到 Option2_Click **事件过程中**。

```
Private Sub Option2_Click(Index As Integer)
    Label1.FontSize = Val(Option1(Index).Caption)
End Sub
```

打开代码窗体，将如下代码添加到 Option3_Click **事件过程中**。

```
Private Sub Option3_Click(Index As Integer)
    Select Case Index
    Case 0
        Label1.ForeColor = RGB(255, 0, 0)          '设置为红色
    Case 1
        Label1.ForeColor = RGB(0, 0, 255)          '设置为蓝色
    Case 2
        Label1.ForeColor = RGB(0, 255, 0)          '设置为绿色
    Case 3
        Label1.ForeColor = RGB(255, 255, 0)            '设置为黄色
    End Select
End Sub
```

打开代码窗体，将如下代码添加到 Command1_Click **事件过程中**。

```
Private Sub Command1_Click()
    End
End Sub
```

打开代码窗体，将如下代码添加到 Form_Load **事件过程中**。

```
Private Sub Form_Load()
    Option1(0).Value = True
    Option2(3).Value = True
    Label1.FontSize = 9
    Label1.FontName = "宋体"
```

```
    Label1.Caption = "文本示例 AaBbCc"
End Sub
```

Forecolor 属性设定标签的前景颜色，即字体颜色。运行程序如图 3-5 所示。

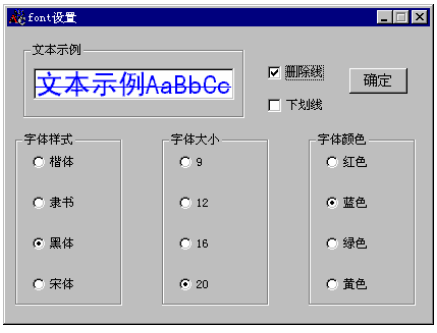

图 3-5　运行时的窗体

这个应用程序中，我们使用复选框和选项按钮来处理用户的选择，使用这种方法时，可以提供给用户的选项是相当有限的。如果需要使用户从大量的项目中进行选择，那么就必须使用列表框和组合框。

3.4　列表框与组合框的使用

当只有少数几个选择时，只要使用复选框和选项按钮就可以工作了。但是，如果有很多可选项时，用户不得不使用整个屏幕的复选框和选项按钮，这样就会令人眼花缭乱，难于作出选择，为此，Visual Basic 还提供了另外两种方法：列表框和组合框。

使用这两个控件用户可以列出很多可选项，但占用的空间却不大，因为 Visual Basic 会自动添加滚动条。列表框为用户提供了选项列表，缺省时，列表为单列垂直显示，也可以设置列表为多列。若列表数目超过了列表框可显示的数目，则列表框自动添加滚动条。

3.4.1　列表框

下面，我们来创建一个列表框的例子，用来说明列表框的常用属性和方法。列表框的绘制方法同其它标准控件相同，创建用户界面如图 3-6 所示。

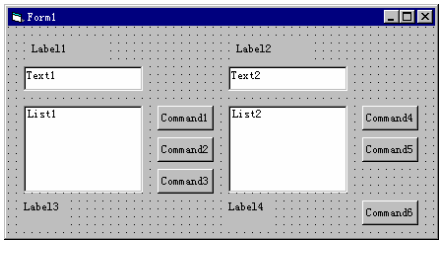

图 3-6　绘制好的界面

本例说明了如何使用 AddItem、RemoveItem 和 Clear 方法，还用到了列表框的 Sorted、Selected、ListIndex、Listcount 、Multiselect 和 List 属性，这些都是列表框最常用最主要的方法和属性。

本例中，用户可把项目输入左边文本框，单击一个按钮将输入项添加到左边列表中。列表下方的标签可用来显示列表中的项目数，用户通过单击另一按钮，可将左边列表中选中的项目传送到右边列表，并自动排序，右边列表上方的文本框中显示最后传入的一个项目，用户还可以删除单个列表项或将列表框清空。

本例中各对象的属性设置值如表 3-2 所示。

可以指定要按字母顺序添加到列表中的项目，为此将 Sorted 属性设置为 True 并省略索引。排序时不区分大小写，因此单词 "japan" 和 "Japan" 将被同等对待。Sorted 属性设置为 True 后，使用带有 index 参数的 AddItem 方法可能会导致不可预料的非排序结果。

MultiSelect 属性用来选定列表框中的一组值。如果单击第一个列表项目，然后按 SHIFT 键并单击最后一个项目（或用 SHIFT+DOWN 方向键），则将选定此范围内的所有项目。

设好属性的界面如图 3-7 所示。

打开代码窗口，将下列代码添加到 Cmdadd 事件过程中：

表 3-2　对象属性设置

对象	属性	设置值
Form1	Caption MaxButton	使用列表框 False
Label1	Caption	添加列表
Label2	Caption	新加内容
Label3	(名称) Borderstyle Caption	Lbldisplay1 1- Fixed Single (Empty)
Label4	(名称) Borderstyle Caption	Lbldisplay2 1-Fixed Single (Empty)
Text1	(名称) Text	Txtname (Empty)
Text2	(名称) Text	Txtnew (Empty)
List1	(名称) Multiselect	Lstsource 2-Extended
List2	(名称) Sorted	Lstobject True
Command1	(名称) Caption	Cmdadd 增加

Command2	(名称) Caption	Cmdtransfer 传送
Command3	(名称) Caption	Cmdclearl 清空
Command4	(名称) Caption	Cmdremove 删除
Command5	(名称) Caption	Cmdclearr 清空
Command6	(名称) Caption	Cmdclose 关闭

```
Private Sub cmdadd_Click()
    Lstsource.AddItem txtname.Text
    txtname.Text = ""
    txtname.SetFocus
    lbldisplay1.Caption = Lstsource.ListCount
End Sub
```

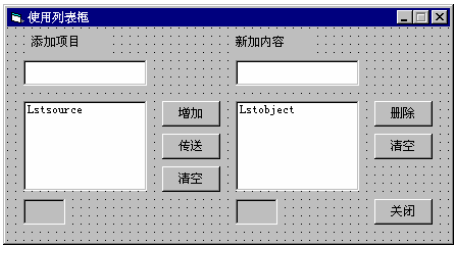

图 3-7　设置好属性的窗体

设计时添加项目

通过设置 ListBox 控件“属性”窗口的 List 属性还可在设计时向列表添加项目。在选定了 List 属性选项并单击向下箭头时，可输入列表项目并按 CTRL+ENTER 组合键换行。

在左边文本框中输入项目，单击增加，如图 3-8 所示。 当添加项目超过列表框显示范围时，列表框自动增加滚动条。

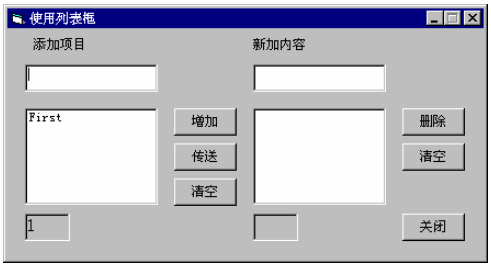

图 3-8　添加了一个列表项

注意：

列表框中的项目顺序就是按项目添加顺序排列，因为其 Sorted **属性为** False。**列表框下面的标签中列出了列表项数目。**

将下列代码添加到 cmdtransfer_Click **事件过程中。**

```
Private Sub cmdtransfer_Click()
    Dim n As Integer
        For n = 0 To (Lstsource.ListCount - 1)
            If Lstsource.Selected(n) = True Then
                Lstobject.AddItem Lstsource.List(n)
                txtnew.Text = Lstsource.List(n)
            End If
        Next n
    lbldisplay2.Caption = Lstobject.ListCount
End Sub
```

可用 List **属性访问列表的全部项目。此属性包含一个数组，列表的每个项目都是数组的元素。每个项目以字符串形式表示。引用列表的项目时应使用如下语法：**

```
box.List(index)
```

box **参数是列表框的引用，**index **是项目的位置。顶端项目的索引为** 0，**接下来的项目索引为** 1，**依此类推。例如，下列语句在一个文本框中显示列表的第三个项目（**index = 2）：

```
Text1.Text = List1.List(2)
```

Selected **属性是一个包含列表框选定状态的布尔数组——判断选定了哪些项目。数组中的每个元素对应一个列表项，选定项目的数组元素值为** True，**未选定项目的数组元素值为** False。

在左边文本框中选中前四项，单击传送，如图 3-9 **所示。**

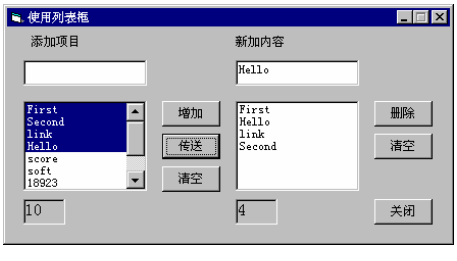

图 3-9　**使用传送按钮**

左边列表框的 Multiselect **设为** True，**可以多选；右边列表框的** Sorted **设为** True，**自动排序。右上方文本框中显示了最近添加到右边列表框中的项目。**

下面，我们将双击列表框中的项目，与选中列表项然后单击传送按钮等价起来，在代码窗口中增加如下事件过程：

```
Private Sub Lstsource_DblClick()
    cmdtransfer.Value = True
```

```
End Sub
```

在代码窗口中加入如下代码：

```
Private Sub cmdremove_Click()
    Dim ind As Integer
        ind = Lstobject.ListIndex
        If ind >= 0 Then
            Lstobject.RemoveItem ind
            lbldisplay2.Caption = Lstobject.ListCount
        Else
            Beep
            MsgBox "无可删除项", 48, "提示"
        End If
End Sub
```

用 ListIndex **属性判断位置，如果要了解列表中已选定项目的位置，则用** ListIndex **属性。此属性只在运行时可用，它设置或返回控件中当前选定项目的索引。设置列表框的** ListIndex **属性也将触发控件的** Click **事件。**

可用 RemoveItem **方法从列表框中删除项目。**RemoveItem **有一参数** index**，它指定删除的项目：**

```
box.RemoveItem index
```

box **和** index **参数与** AddItem **中的参数相同。例如要删除列表中的第一个项目，可添加下行代码：**

```
List1.RemoveItem 0
```

MsgBox **函数用来显示一个提示对话框。**

在右侧列表框中选中某一项，单击“删除”按钮就可以将其删除。不选任何列表项，单击删除就会弹出提示对话框，单击确定就可将提示对话框关闭。

在代码窗口中添加如下代码：

```
Private Sub cmdclearr_Click()
    Lstobject.Clear
    lbldisplay2.Caption = Lstobject.ListCount
End Sub


Private Sub cmdclearl_Click()
    Lstsource.Clear
    lbldisplay1.Caption = Lstsource.ListCount
End Sub
```

Clear **分别表示将左、右列表框中内容全部删除。**

将下列代码添加到 cmdclose_Click **事件过程中。**

```
Private Sub cmdclose_Click()
```

```
    Unload Me
End Sub
```

为了使用户不能随意改动右侧文本框中的内容，在代码窗口中添加如下事件过程：

```
Private Sub txtnew_GotFocus()
    txtname.SetFocus
End Sub
```

这样，右边文本框就无法获得焦点，不能修改其内容。这个例子中，用到了列表框最主要的方法和属性。另外，还有一个常用属性是 Text 属性。通常，使用 Text 属性来获取当前选定的项目值。Text 属性总是对应用户在运行时选定的列表项目。

3.4.2　组合框

列表框只允许用户选取列表框中存在的项目，而组合框则不同，组合框结合了文本框和列表框的功能，用户可以从组合框列表中选定项目，也可以在组合框中输入文本框，选定项目。组合框向用户提供了供选择的列表，如果项目数超过组合框能显示的项目数，则组合框会自动添加滚动条。

组合框有三种样式，可以通过 Style 属性来设置。当 Style 值为 0 时，表示下拉组合框；当 Style 值为 1 时，表示简单组合框；当 Style 值为 2 时，表示下拉列表框。

通常，组合框适用于建议性的选项列表，而当希望将输入限制在列表之内时，应使用列表框。组合框包含编辑区域，因此可将不在列表中的选项输入列表区域中。此外，组合框节省了窗体的空间。只有单击组合框的向下箭头时（样式 1 的组合框除外，它总是处于下拉状态）才显示全部列表，所以无法容纳列表框的地方可以很容易地容纳组合框。

在组合框中添加项目、删除项目，以及排序、访问列表项等方法和属性与列表框相同，具体使用请参见列表框。

3.5　本章小结

本章中，我们使用了四种方式来处理用户的选择：复选框与选项按钮，列表框与组合框。前两种适合于少量选择时使用，后两种适合于大量选择时使用。本章中分别举例说明了四种控件的使用方法和基本特点。

第四章　综合应用举例：记分器

随着对 Visual Basic6.0 的深入了解，大家会逐渐发现，使用 Visual Basic 编写 Windows 下的应用程序是很方便的。由于 Visual Basic 易学易用的特点，Visual Basic 逐渐成为一种广受欢迎的可视化编程工具。而且，一旦你已经掌握了 Visual Basic，再去学习其它的可视化语言，就会觉得非常容易。

这章中，我们继续使用 Visual Basic 标准控件来编写一个实用性很强的应用程序——竞赛记分器。根据现场竞赛的特点，这个程序主要有两个基本功能。

- 记录各参赛队分数，并可作加减操作
- 对选手答题进行定时

程序界面也是根据这两项功能大体分为两块。为了使程序更完善，我们又增加了一些辅助功能添加菜单，最终界面如图 4-1 所示。

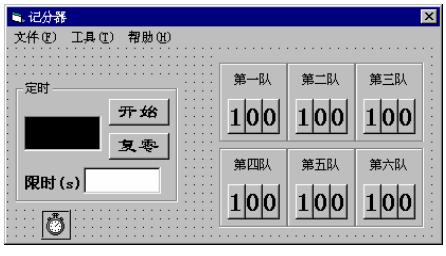

图 4-1　记分器界面

4.1　用户界面的创建

建立用户界面不只意味着只将一些命令钮简单地一起排列在屏幕上，并且希望用户能够算出程序是如何工作的，而是要求设计的用户界面必须使程序容易使用。

一个好的用户界面必须面向用户，以便用户能知道他们该如何使用程序，为此我们要在程序中添加一些说明或提示。界面的设计还常常与我们打算如何编写代码有关。如果我们对很多对象只想编写一个事件过程，那么我们就要使用控件数组。

为了实现记分器的基本功能，我们暂时将控件分为两部分，即定时部分和记分部分。

4.1.1　添加定时器控件

将一个完整的程序按其功能分为几块来处理，往往会减小程序编写的难度。现在，我们先来绘制用于实现定时功能的控件。

基本思路：程序允许用户输入一个限定时间，比如 60s，当用户单击一个按钮后，开始倒记时，直到定时结束。

绘制界面的基本操作步骤如下：

1. 启动 Visual Basic，在"文件"菜单中单击"新建工程"命令，或者单击工具栏中的"新建工程"，建立一个标准的 Visual Basic 应用程序。

2. 参照图 4-1 调整窗体大小。

3. 单击工具箱中的 Frame 控件，在窗体上添加一个框架。

框架控件可以对界面对象分组，这样做的好处是使整个界面看上去整齐美观，有条理，使用框架控件还可以使框架内的对象同框架作为一个整体进行移动和删除。当框架的 Enabled 属性被设置为 False 时，则框架上的所有对象都被屏蔽掉。框架控件还有一个常用的功能是对选项按钮进行分组，使得每组内只一个选项按钮被选中，在以后的例程中我们可以看到。

4. 单击工具箱中的 TextBox 控件，在框架内添加一个文本框。

5. 单击工具箱中的 CommandButton 控件，在框架内添加一个命令按钮。

6. 重复第 5 步，再向框架内添加一个命令按钮。

7. 单击工具箱中的 Label 控件，在框架内添加一个标签。

8. 单击工具箱中的 TextBox 控件，在框架内添加一个文本框。

9. 单击工具箱中的 Timer 控件，在窗体上添加一个定时器，用于记时；界面如图 4-2 所示。

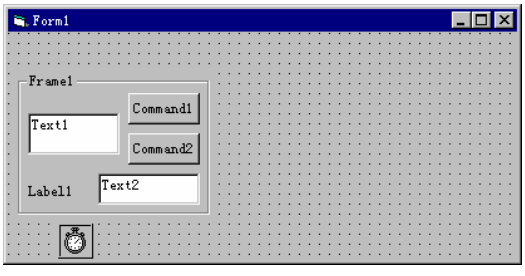

图 4-2　添加好定时控件

由于 Timer 控件在运行时不可见，事实上，可以把它放在窗体的任何位置。

利用这些控件，我们就可以实现定时功能，等一会儿再来编写代码，下面我们先来绘制用于实现记分功能的控件。

4.1.2　记分牌的设计

本例中，我们设计的参赛队有六支，所以要显示六支队伍各自的队名，并设计六个记分牌，用于记录并修改各队的得分情况。一般情况下，各队得分均为三位数，且竞赛开始总会设有一定的基础分，为实现分别对得分的个位、十位和百位进行操作，因此，每队记分牌均由三个命令组成，分别代表得分的个位、十位和百位。

容易想到，我们对各队记分牌个位的操作都是类似的，个位满十向十位进一，十位满十向百位进一，十位、百位也是如此。因此，为了使代码简化，我们使用控件数组来设计记

分牌。即各队得分的个位，使用一个控件数组；各队得分的十位，使用一个控件数组；各队得分的百位，使用一个控件数组。

操作步骤如下：

1. 单击工具箱中的 Frame 控件，向窗体上添加一个框架控件。

2. 重复第上一步，再向窗体添加五个框架控件，如图 4-3 所示，六个框架分别用来显示六支参赛队的名称及得分。

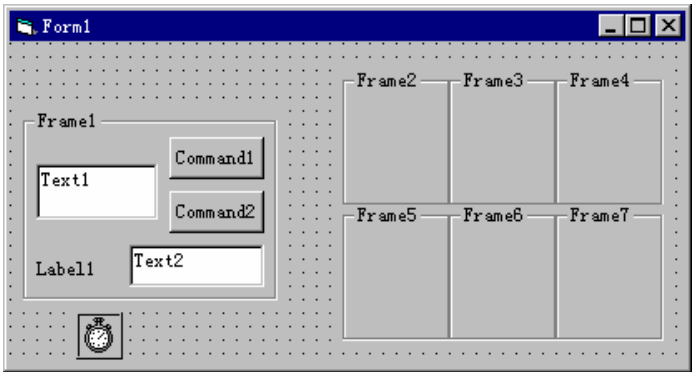

图 4-3　在窗体上添加六个框架

使用框架对界面进行分组，可以使界面看起来整齐，不零乱。

3. 单击工具箱中的 Label 控件，在第一个框架内添加一个标签。

4. 复制刚才添加的标签，分别向其余五个框架内各添加一个标签，如图 4-4 所示。

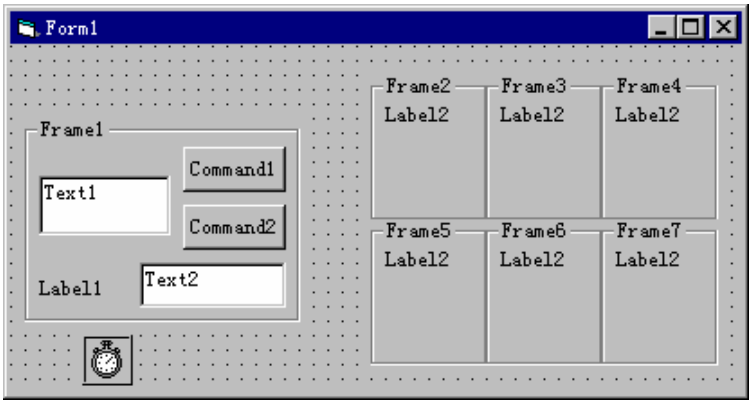

图 4-4　创建标签控件

标签用来显示各队队名，下面，我们来绘制记分牌，创建三个命令钮控件数组。

5. 单击工具箱中的 CommandButton 控件，向第一个框架框内添加一个命令按钮，用来显示第一支参赛队的得分的个位。

6. 调整按钮大小，如图 4-5 所示。

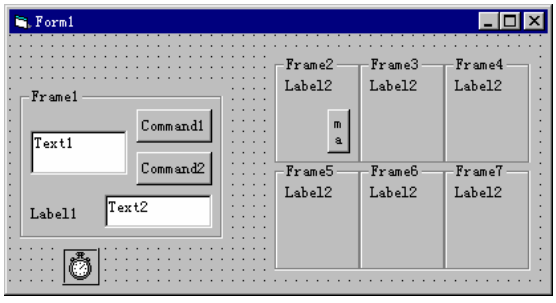

图 4-5　添加第一个按钮

7. **复制五个按钮，分别添加到其余五个框架的相应位置上，最好使该控件数组的** Index **属性从左到右，从上到下分别为** 0,1,2,3,4,5，**如图** 4-6 **所示**。

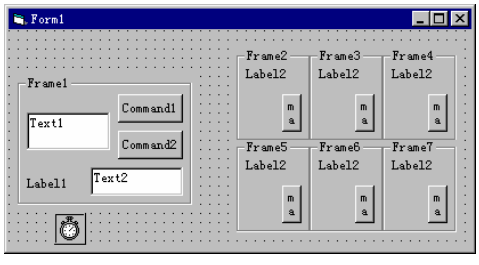

图 4-6　创建好按钮数组

8. **重复上面两步，分别绘制出十位和百位；如图** 4-7 **所示**。

同组的三个按钮其 Index **属性一定要保持一致。这样，会使按钮的代码大为简化，且不易出错。现在，我们已经绘制好了记分器的界面**。

4.2　属性的设置

通过属性窗口，对所有元素的属性进行设置。各对象属性设置值见下表。

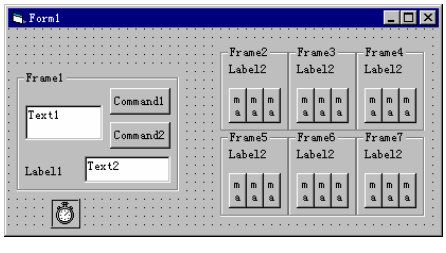

图 4-7　绘制好的界面

表 4-1　定时控件属性设置

对象	属性	设置值
Form1	（名称）	Frmmain

	Caption MaxButton	**记分器** False
Frame1	Caption	**定时**
Text1	（**名称**） Alignment Backcolor Forecolor Locked Text	Txtoutput 2-Center **黑色** **红色** True (Empty)
Text2	（**名称**） Alignment Text	Txtiuput 2-Center (Empty)
Command1	（**名称**） Caption Font	Cmdstart **开始** **隶书、规则、四号**
Command2	（**名称**） Caption Font	Cmdtozero **复零** **隶书、规则、四号**
Label1	（**名称**） Caption Font	Lblrestraintime **限时**(s) **宋体、粗体、五号**
Timer1	Enabled Interval	False 1000

MaxButton **属性设置窗体是否具有“最大化”按钮。当其值为** True（**缺省值**）**时，窗体具有最大化按钮；当其值为** False **时，窗体没有最大化按钮。**

窗口最大化后，最大化按钮自动地变成恢复按钮，将窗口最小化或恢复窗口把恢复按钮变回最大化按钮。

MinButton **属性设置窗体是否具有“最小化”按钮。当其值为** True（**缺省值**）**时，窗体具有最小化按钮；当其值为** False **时，窗体没有最小化按钮。**

最小化按钮能够将窗体窗口最小化为图标。要显示最大化和最小化按钮，必须将 BorderStyle **属性设置为** 1（**固定单边框**）、2（**可变尺寸**）**或** 3（**固定双边框**）。

为 MaxButton、MinButton **和** BorderStyle **属性指定的设置值直到运行时才能在窗体的外观上反映出来。**

BackColor **和** ForeColor **属性分别用来设置文本框的背景色和前景色。**Visual Basic **用** Microsoft Windows **运行环境的红-绿-蓝** (RGB) **颜色方案来设定颜色。**

表 4-2 记分控件属性设置

对象	**属性**	**设置值**
Frame2—Frame7	Caption	(Empty)

Label2 (队名数组)	(名称) Alignment Index－Caption	Lblteam 2-Center 0－第一队 1－第二队 2－第三队 3－第四队 4－第五队 5－第六队
Command3 (个位数组)	(名称) Caption Font Index	Cmdone 0 宋体、粗体、小二 0－5
Command4 (十位数组)	(名称) Caption Font Index	Cmdten 0 宋体、粗体、小二 0－5
Command5 (百位数组)	(名称) Caption Font Index	Cmdhundred 1 宋体、粗体、小二 0－5

4.3　定时功能的实现

利用 Timer 事件实现定时功能，我们在属性窗口中设置 Timer 控件的 Interval 属性为 1000，即每一秒钟发生一次 Timer 事件。在 Timer 事件过程中我们使限定时间按秒减少，直到为零。

4.3.1　定义通用变量

打开代码窗口，首先在“通用”中作如下定义。

```
Dim a, e, f As Integer
Dim b, c, d As String
```

这些变量在下面的代码段中会用到。

4.3.2　定时代码

在输入文本框前面的提示时，我们已经告诉用户，输入时间是以秒为单位(s)的，若某一道题允许选手在 30s 内回答完备，便可以在“限时”文本框中输入“30”。

代码如下：

```
Private Sub Txtinput_Change()
    b = Txtinput.Text
End Sub
```

利用 Txtinput 的 Change 事件，将用户输入的限定时间记录下来，保存到变量 b 中。

4.3.3 记时代码

输入限定时间后，在选手开始答题时，开始记时，代码如下：

```
Private Sub Cmdstart_Click()
    Txtoutput.Text = b
    Timer1.Enabled = True
End Sub
```

首先，将用户输入的限定时间赋给记时文本框，然后设置 Timer 控件有效，使记时文本框开始倒记时。

4.3.4 Timer 事件代码

在 Cmdstart_Click 事件过程中，将限定时间赋给了记时文本框，并且使 Timer 控件有效，这样就可以使用 Timer-Timer 事件过程来实现倒记时功能。如果 Timer 的 Enabled 属性为 False，则下面的代码段是无效的。

Timer 事件代码如下：

```
Private Sub Timer1_Timer()
    Dim i As Integer
    a = Val(Txtoutput.Text) - 1
    If a >= 0 Then
        Txtoutput.Text = a
    Else
        Txtoutput.Text = "0"
    End If
End Sub
```

代码使限定时间每秒减小 1，直到减小到零为止。

4.3.5 复零按钮代码

在竞赛中常常会遇到这样的情况：很多选手都可以在限定时间以内完成回答，即不等定时文本框显示为“0”，就可以作出完整的回答。比如，我们限定选手在 30 秒内回答完备，而选手在 15 秒内就完成回答，这时，我们就没有必要睁大眼睛，等着限定时间一秒一秒地减小到 0。为此，我们在定时部分设置了“复零”按钮。

添加了“复零”按钮，就可以使定时文本框显示为“0”，而不必在每次开始记时后，都得等待它逐秒减少到 0。

代码如下：

```
Private Sub Cmdtozero_Click()
    Txtoutput.Text = "0"
    Timer1.Enabled = False
End Sub
```

4.4　记分功能的实现

根据平时的经验，我们为各队设置了 100 分的基础分，在对得分进行加减操作时，个位允许每次加减 5 分，十位允许每次加减 1，即加减 10 分。一般情况，我们并不直接操作百位，因此，没有对百位编写事件过程。

操作中，用户单击左键表示加分，用户单击右键表示减分，因此，我们要求程序能够识别用户到底按下了左键还是右键，以便作出相应的加减分操作。

4.4.1　Mouseup 事件的使用

大家很容易想到使用 Click 事件来识别左右键。但是很遗憾 Click 事件并不能区分左、右键，它只能识别左键，而当用户按下右键时，程序不会作出任何反应。因此，我们必须找到另一个事件过程来代替 Click 事件，要求这一事件可以识别用户到底按下了左键还是右键。

本例中，使用了另一鼠标事件 Mouseup 事件。

- **专题讲座**：Mouseup 事件

MouseUp 这一事件是当释放鼠标按钮时发生。与此联系，当按下鼠标按钮时会发生 MouseDown 事件。这两个事件在很多情况下是等效的。

MouseUp 事件的语法如下：

```
Private Sub object _MouseUp([index As Integer,]button As Integer, shift As Integer, x As Single, y As Single)
```

语法部分说明：

object　表示发生 MouseUp 事件的对象。

Index　为一个整数，用来标识惟一地一个在控件数组中的控件。

Button　为一个整数，用来标识该事件的产生是按下（MouseDown）或者释放（MouseUp）按钮引起的。当 Button 为 1 时，表示左按钮被按下；当 Button 为 2 时，表示右按钮被按下；当 Button 为 4 时，表示中间按钮被按下。

Shift　返回一个整数，在 button 参数指定的按钮被按下或者被释放的情况下，该整数相应于 Shift、Ctrl 和 Alt 键的状态。这些位中可能有一些，全部，或者一个也没有被设置，指示这些键中的一些、全部，或者一个也没有被按下。例如，Ctrl 和 Alt 键都被按下，则 Shift 的值就是 6。

表 4-3　Shift 键、Ctrl 键、Alt 键的不同组合

Shift 参数	Shift 键是否按下	Ctrl 键是否按下	Alt 键是否按下
0	N	N	N
1	Y	N	N
2	N	Y	N
3	Y	Y	N
4	N	N	Y
5	Y	N	Y
6	N	Y	Y
7	Y	Y	Y

x, y　　返回一个指定鼠标指针当前位置的值。x 和 y 的值所表示的总是通过该对象 ScaleHeight, ScaleWidth, ScaleLeft, 和 ScaleTop 属性所建立的坐标系统的方式。

为了在给定的一个鼠标按钮按下或释放时指定将引起的一些操作，应当使用 MouseDown 或者 MouseUp 事件过程。不同于 Click 和 DblClick 事件的是，MouseDown 和 MouseUp 事件能够区分出鼠标的左、右、和中间按钮。也可以为使用 Shift、Ctrl 和 Alt 等键盘换挡键编写用于鼠标～键盘组合操作的代码。

使用 Mouseup 事件分别对计分器的个位和十位按钮编写代码，实现正确的加减分操作。

4.4.2　个位按钮代码

个位按钮代码：

```
Private Sub Cmdone_MouseUp(Index As Integer, Button As Integer, Shift As Integer,
    _ X As Single, Y As Single)
    f = Val(Cmdone(Index).Caption)
    If Button = 2 Then                          '是否按下右键
        If f < 5 Then                                   '个位减分
            Cmdone(Index).Caption = f + 10 - 5
            Cmdten(Index).Caption = Val(Cmdten(Index).Caption) - 1
        Else
            Cmdone(Index).Caption = f - 5
        End If
        e = Val(Cmdten(Index).Caption)
        If e < 0 Then                               '是否向百位借位
            Cmdten(Index).Caption = e + 10
            Cmdhundred(Index).Caption = Val(Cmdhundred(Index).Caption) - 1
        Else
        End If
    Else                                            '按下左键
```

```
        If f >= 5 Then
            Cmdone(Index).Caption = f + 5 – 10          '个位加分
            Cmdten(Index).Caption = Val(Cmdten(Index).Caption) + 1
        Else
            Cmdone(Index).Caption = f + 5
        End If
        e = Val(Cmdten(Index).Caption)
        If e > 9 Then                                   '是否向百位进位
            Cmdten(Index).Caption = e - 10
            Cmdhundred(Index).Caption = Val(Cmdhundred(Index).Caption) + 1
        Else
        End If
    End If
End Sub
```

个位的操作会影响到十位，甚至是百位，例如，原来得分为 195 **分，若加** 5 **分，则变为** 200 **分。因此，相应的改动，在这段代码中大家要充分体会，我们为什么要使同一记分牌的三个按钮的** Index **要属性一致。**

4.4.3　十位按钮代码

这段代码与上段代码大同小异，只是最小变化尺度为 1，**且十位的改变不会影响到个位数，只需要根据条件相应的百位数进行修改。代码如下：**

```
Private Sub Cmdten_MouseUp(Index As Integer, Button As Integer, Shift As Integer,
        _ X As Single, Y As Single)
    d = Val(Cmdten(Index).Caption)
        If Button = 2 Then
            If d = 0 Then
                Cmdten(Index).Caption = d + 10 - 1
                Cmdhundred(Index).Caption = Val(Cmdhundred(Index).Caption) - 1
            Else
                Cmdten(Index).Caption = d - 1
            End If
        Else
            If d >= 9 Then
                Cmdten(Index).Caption = d + 1 - 10
                Cmdhundred(Index).Caption = Val(Cmdhundred(Index).Caption) + 1
            Else
                Cmdten(Index).Caption = d + 1
```

```
            End If
        End If
    End Sub
```

到此为止，我们以上编写的代码，已经可以实现记分器的两个基本功能：定时和记分。下面，我来进行测试，看看程序能否实现我们的设想的功能。

4.4.4 程序检测

编写程序过程中，一边编写一边检测，很容易及时发现错误并改正。当程序一步一步地按着我们的设想运行时，大家就会体会到用 Visual Basic **编程带来的乐趣。**

运行程序，操作步骤如下：

1. **单击工具栏中的**  **按钮，运行程序；**

2. **在“限时”文本框中输入“**30**”，然后单击“开始”按钮，观察定时文本框的变化，**5 **秒后如图** 4-8 **所示；**

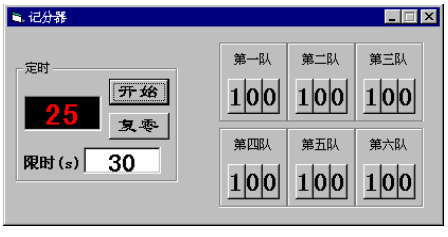

图 4-8 **运行时的窗体**

4. **利用鼠标左、右键操作各队的得分，调整各队得分，如图** 4-9 **所示；**

5. **在“限时”文本框中输入一个数，单击“开始”，然后单击“复零”，观察定时文本框变化；**

图 4-9 **运行时的窗体**

6. **单击工具栏中的**  **按钮，结束程序运行，回到编辑状态。**

看来，使用 Visual Basic **编程确实不难，随着对** Visual Basic **的熟悉，大家对** Visual Basic **的兴趣也会越来越浓。**

一个好的应用程序，往往会给用户许多提示，告诉他们程序运行到了哪一步，或者用户该如何来操作等等，下面，我给程序增加一些提示信息。

4.5　提示信息的显示

在竞赛中，若限定时间到，则要求选手停止回答，但若操作者心不在焉，定时文本框早已显示为零，而操作者却没有发现，这就是说容易出错，为此，我们在限定时间到时，增加一些提示信息，来提醒操作者。

这里，我们打算使用声音（Beep）和提示对话框两重提示来提醒操作者。

Beep 语句是通过计算机喇叭发出一个声调，呼叫的频率与时间长短取决于硬件和系统软件，并随电脑不同而不同。我们使用 MsgBox 函数来显示一个对话框。

4.5.1　MsgBox 函数的使用

MsgBox 函数在对话框中显示消息，等待用户单击按钮，并返回一个 Integer 告诉用户单击哪一个按钮。对话框包含用户可以修改的四个部分：标题条、信息、可见图标和显示命令钮的数及类型。这些特征如图 4-10 所示。

标题表明对话框的目的，如“About This Program”（有关这个程序）等；信息包含对话框中显示的正文，如“真的要退出吗？”等；图标提供有关对话框的重要信息。

一、　增加对话框的图标

图标有助于将用户的注意力吸引到对话框上，在 Visual Basic 中有四个有用的图标。

(一) Stop 图标对极其重要的问题提醒用户，如"这个操作可能会导致程序完全丢失，确定要这样做吗？"。

(二) 问号用于没有威胁性的问题，只作为一般的提示，如：“确定要退出吗？”。

(三) 感叹号用于强调警告用户必须知道的事件，如：“程序尚未存储，退出后将全部丢失。”

(四) 信息号可以使得单调乏味和令人厌烦的对话框显得有些生气。

图 4-10　用 MsgBox 显示的对话框

正常情况下，Visual Basic 允许用户在对话框中显示一个图标。如果用户对这个限制感到厌烦，则可以伪造一个对话框，如创建一个独立的窗体画几个命令钮和一个图形框，在画好图形框之后，可以装入需要的任何类型的图标。要使用独立窗体创建一个对话框需要有命令钮、标签、图形框和代码，以保证窗体能够工作。

如果我们只想创建一个快速简便的对话框，则可以用 MsgBox 函数来代替。

二、 确定命令钮数组和类型

对话框中最多可以包含 3 **个命令钮，每种命令钮组合可以用一个数组来代替，命令钮数和类型见表** 4-4。

表 4-4 在 Visual Basic 中有效的命令钮组合

显示	值
OK 命令钮	0
OK 和 Cancel 命令钮	1
Abort、Retry 和 Ignore 命令钮	2
Yes、No 和 Cancel 命令钮	3
Yes 和 No 命令钮	4
Retry 和 Cancel 命令钮	5

三、 返回用户所选的命令

用两个或多个命令的对话框给用户作一个选择。当然，在用户作出选择之后，程序必须确定用户选择了什么，表 4-5 **中显示用数字表示可供用户选择的七种可能。**

为了使程序确定用户选择了哪一个命令钮，我们必须用一个变量来接受 MsgBox **函数的返回值。如，**

Reply = MsgBox(“**返回被选按钮的数值**”，48，“**提示**”)

我们用变量 Reply **来接受从对话框返回的选择命令钮的数值。当我们用一个变量来接受从对话框返回的选择命令钮的数值时，我们必须用括号将对话框参数括起来。如选择了** OK, **则** Reply **为** 1。

表 4-5 用户可以选择的命令钮

被选择的命令钮	数组
OK 命令钮	1
Cancel 命令钮	2
Abort 命令钮	3
Retry 命令钮	4
Ignore 命令钮	5
Yes 命令钮	6
No 命令钮	7

4.5.2 Timer 事件代码的改写

为了能够在时间到时，显示提示对话框，并发出声音。我需要在 Timer_Timer **事件过程代码中添加** Beep **语句和** MsgBox **函数。改写后代码如下：**

```
Private Sub Timer1_Timer()
    Dim i As Integer
    a = Val(Txtoutput.Text) - 1
```

```
        If a >= 0 Then Txtoutput.Text = a
        If Txtoutput.Text = "0" Then
            Beep
            MsgBox "回答时间到，请停止回答", 48, "提示"
            Timer1.Enabled = false
        End If
    End Sub
```

由于提示信息添加在 Timer_Timer 事件过程内，因此当我们按“复零”按钮使定时文本框显示为“0”时，并不出现提示信息，这也符合实际情况的。而且，在显示提示对话框后，我们将 Timer 的 Enabled 属性设置为 false，这可以保证提示对话框只显示一次。

4.5.3　程序检测

现在，我们来检测一下刚才添加的提示信息是否能够按要求正确显示。

1. 单击工具栏中的 ▶ 按钮，运行程序。
2. 在“限时”文本框内输入“10”。
3. 单击“开始”按钮，等待定时文本框减小到 0。

当定时文本框减小到“0”时，发出声音，同时显示提示对话框，如图 4-11 所示。

4. 单击确定，关闭对话框。
5. 在“限时”文本框内输入“10”，单击“开始”。

然后单击“复零”，观察是否出现提示对话框。

可见，使用“复零”按钮使定时文本框显示为“0”时，并不显示提示信息，程序运行同我们设想的结果完全一致。

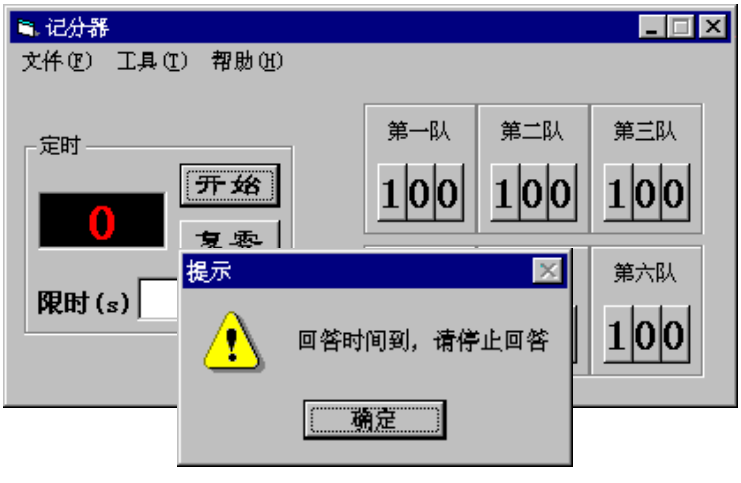

图 4-11　弹出提示对话框

4.6 菜单的添加

对 Windows 的用户来说，对菜单的依赖已经达到了无可复加的程度，离开了菜单，简直就无法进行操作。典型的菜单条一般在屏幕的顶端显示菜单标题，如果选择其中一个标题，则显示一个下拉菜单，如图 4-12 所示，是一个典型的菜单的例子。

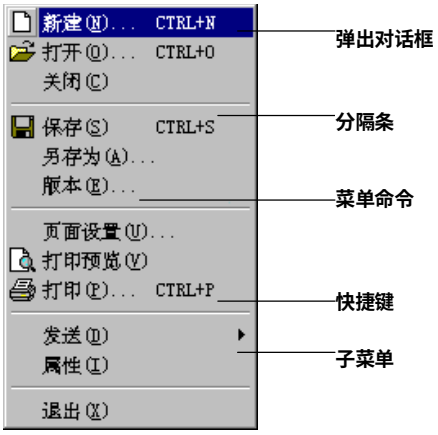

图 4-12 打开一个下拉菜单

一般，菜单条中包含了“文件”、“编辑”、“工具”、“窗口”和“帮助”等菜单标题，并且，他们的顺序也是大致不变的。每一个菜单由菜单标题和菜单命令组成。菜单标题出现在屏幕顶部的菜单条中，菜单命令出现在下拉式菜单中。对于大多数 Windows 下的应用程序，它们拥有一些相同的菜单标题，并且每个菜单标题下的命令也是大致相同的。

为了在一个菜单条中包含更多的命令，往往在下拉式菜单中隐藏了子菜单。Visual Basic 让用户最多创建四级子菜单，但是大多数程序只使用一级子菜单，这可以避免因隐藏大深而寻找困难。当一个菜单标题有一个箭头符号显示，它表示在该标题中存在子菜单。

根据实用的需要，本例中我们需要创建“文件”、“工具”和“帮助”三个菜单标题，菜单标题下包含了菜单命令及子菜单。

4.6.1 菜单设计窗口

为了创建菜单，用户必须打开菜单设计窗口。

1. 从“工具”菜单下选择“菜单编辑器”命令，或在工具栏中单击  图标，打开菜单设计窗口。若工程中包含多个窗体，则可以对每个窗体使用完全不同的菜单。打开菜单设计窗口前用户选择了哪个窗体，那么编辑好的菜单就会出现在那个窗体上。

2. 在“标题”文本框中输入“文件（&F）”，在“名称”文本框中输入 Mnufile，如图 4-13 所示。

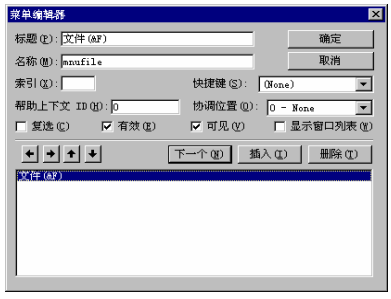

图 4-13　菜单编辑器

每一个菜单标题和菜单命令都有一个“标题（Caption）”和“名称（Name）”，同我们前面使用的控件一样，Caption 是屏幕上实际显示的东西，如"文件"、"编辑"、"帮助"等等，而 Name 从不在屏幕中显示，它只是用于代码中表明用户选取了哪一个菜单命令。

由于 Caption 在屏幕上是可见的，所以我们可以在标题中使用“与（&）”字符，如&F，这样可以在该字母下显示一个下划线，对于那些偏爱键盘的用户来说，当菜单中一个字母有下划线时，用户可以同时按下 Alt 键和该字母键进入菜单。

如果在菜单命令中使用“与（&）”字符，则用户可以不按 Alt 键，只要输入具有下划线的字母就可以选择该标题。

由于 Name 从不在屏幕上显示，因此，我们可以使它长一些，以便在代码段中明确地表示其含义。对 Name，Visual Basic 推荐使用以 Mnu 开头的 Name ，如 Mnufile。当给一个菜单命令或子菜单命名时，往往可以在后面添加 Item ，如 MnufilenewItem。

3. 单击菜单设计窗口中的“下一个”按钮。
4. 在“标题”文本框输入“设置基础分”，在“名称”文本框中输入 Mnusetscoreitem。
5. 重复 3 到 5 步，分别向菜单设计窗口中添加其他内容，如图 4-14 所示。

图 4-14　添加好菜单标题

现在我们添加的内容都将作为菜单标题显示于窗体的菜单条中。事实上，我们只要求其中几个作为菜单标题显示，而其它内容作为菜单命令显示于下拉菜单中。菜单标题和菜单名称对照表如图 4-6 所示。

4.6.2　菜单命令的建立

要在菜单标题下创建菜单命令，其操作是很简单的，本例中，我们在“文件”菜单下创

建“设置基础分”，“设置队名”及“关闭”三个命令；在“工具”菜单下创建“统计得分”命令；在“帮助”菜单下，创建“帮助信息”命令。

要创建菜单命令，首先要在菜单控制列表框中输入其名称，这一步，我在上面已经做过了，现在只要把它们变为菜单命令即可。

表 4-6　菜单标题和菜单名称

菜单标题	菜单名称
文件(&F)	Mnufile
设置基础分	Mnusetscoreitem
设置队名	Mnusetteam
关闭(&E)	Mnuexititem
工具(&T)	Mnutool
统计得分	Mnuscore
帮助(&H)	Mnuhelp
帮助信息	Mnuhelpitem

创建菜单命令的方法：从菜单控制列表框中选中想要作为菜单命令的选项，单击  按钮，使其缩进一次，这样，它就由菜单标题变为了菜单命令，继续缩进，可以使它在第一级子菜单中显示。

具体操作步骤：

1. 从“工具”菜单下选择“菜单编辑器”命令，打开菜单设计窗口；
2. 在菜单控制列表中，选择“设置基础分”选项；
3. 单击  按钮，使它成为“文件”菜单下的一个命令；
4. 重复 2 到 3 步，建立其它菜单命令，结果如图 4-15 所示。

图 4-15　建立菜单命令

可见，要使一个菜单标题变为菜单命令，其操作非常简单。同样，要想将一个菜单命令提升为菜单标题也很容易，只需先选中要想要改变的菜单命令，然后单击  按钮。

附带说明，使用菜单设计窗口中的 按钮和 按钮，可以上下移动在菜单控制列表中增亮的菜单标题或命令。因此，当我们最初创建下拉菜单时，可以不必将它做得很完美，随时都可以调整它们之间的相对位置和级别。

4.6.3　分离条的设置

分隔条是指在菜单中将命令项分组的横线。一般情况下，分隔条将几个相关命令分成一组，这样用户可以很快找出他们需要的命令。Windows 中的应用程序，都遵循了 Windows 的原则，具有类似分组。

本例“文件”菜单中包含了三个命令，我们将其分为两组，具体操作步骤如下：

1. 打开窗体编辑器。

2. 从菜单控制列表中选择“关闭”命令项，因为“关闭”命令项要直接在分离条下面显示。

3. 单击“插入”按钮，Visual Basic 将增亮一个空行。

使用“插入”按钮，可以随时在菜单控制列表的任何位置增添一个菜单项。同样，使用“删除”按钮可以删除菜单控制列表中的任何菜单项，这样做的前提是，将要删除的菜单项增亮。

要在一个下拉菜单中设置分离条，必须保证分隔条与其上下菜单命令在同级水平上。

图 4-16　添加了分隔条

4. 在 Caption 文本框中输入连字符“-”，在 Name 文本框中输入 mnuseparate，如图 4-16 所示。

5. 单击“确定”，关闭菜单设计窗口。

现在，我们的记分器窗体上已经显示了三个菜单标题，单击“文件”菜单，如图 4-17 所示。

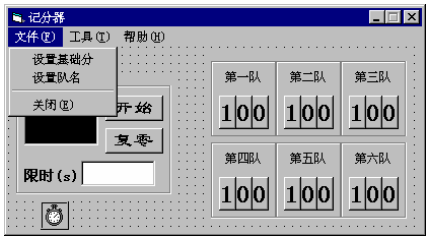

图 4-17　打开文件菜单

4.6.4　子菜单的增加

要想在菜单条或多级下拉菜单中出现尽可能多的命令，那么就要创建子菜单，子菜单经常将命令隐藏在一系列下拉式菜单的深处。

定义子菜单仍然要在菜单设计窗口中进行，在菜单控制列表中，任何一个未缩进的项都被作为菜单标题。缩进一次的项表示在下拉式菜单中显示，缩进二次的项表在第一级子菜单中显示，缩进三次的项表示在第二级子菜单中显示，……直到最后一级子菜单（第四级子菜单）。

本例中，我们为“设置队名”添加子菜单，分别用来设置各支参赛队的队名，为创建一个子菜单，可按下面步骤执行：

1. 从“工具”菜单中选择“菜单编辑器”命令，或单击工具栏中的  图标，打开菜单设计窗口；
2. 从菜单控制列表中选择分隔条，单击“插入”按钮；
3. 在 Caption 文本框中，输入“第一队”，在 Name 文本框中输入 mnufirstitem；
4. 单击  按钮，使它在“设置队名”的子菜单中显示；
5. 重复第 2 到第 4 步，增加其它子菜单命令，如图 4-18 所示；
6. 单击确定，关闭菜单设计窗口。

在记分器窗体中显示子菜单，如图 4-19 所示。

图 4-18　增加子菜单

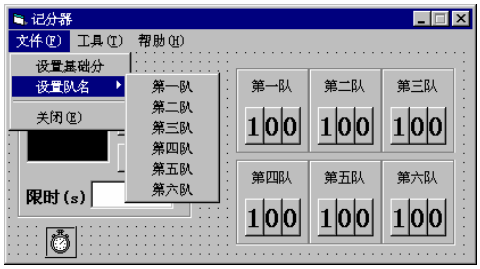

图 4-19　打开子菜单

4.6.5　快捷键的设置

每次都要通过下拉菜单做一些事情，不久就会发现这很麻烦。对于一些常用的命令，为它们设置快捷键将会大大提高操作速度，例如，Ctrl+c 代表复制，Crtl+v 代表粘贴。

注意：

前面使用"与（&）"，可以给菜单标题或命令创建访问键，而快捷键只能对菜单命令进行设置，菜单标题无快捷键。

快捷键在菜单中总是跟在它们所代表的命令后面。为了给一个菜单命令设置热键，必须再次使用菜单设计窗口，Visual Basic 允许用户从有限的热键中为菜单命令选择热捷，记住，一个热键只能赋给一个菜单命令。

本例中，我们为“第一队”等六个子菜单命令及“帮助信息”菜单命令设置快捷键，具体操作步骤如下：

1. 从“工具”菜单中选择“菜单编辑器”命令，或单击工具栏中的  图标，打开菜单设计窗口；

2. 在菜单控制列表中选择“第一队”，从“快捷键”右边的下拉列表中选择“Ctrl+F1”；

3. 重复第 2 步，分别为其他菜单命令设置快捷键，设置好的窗口菜单编辑器如图 4-20 所示；

图 4-20　添加了快捷键

4. 单击确定，关闭菜单设计窗口。

在记分器窗体中打开子菜单如图 4-21 所示。

现在，还没有编写任何代码告诉程序这些命令做什么，所以目前按下任何快捷都不起作用，一旦编写好菜单命令的代码，那么按下快捷键就相当于用鼠标在下拉菜单中进行选择。

另外，使用复选、有效、可见、显示窗口列表四个复选框，还可以为菜单增加确认标志，使菜单命令模糊，消失及使一个菜单成为窗口菜单。

- 增加确认标记

在菜单中，靠近命令的确认标志表示该命令项已经被选择，我们需要在下拉菜单中使用缺省选择时，则可以设置确认标志，在窗体编辑器中设置了确认标志，我们可以使用代码将其删除，如：

```
MnuTools.checked = False
```

代码也可以为菜单命令增加一个确认标志，如：

```
MnuFontsise.checked = True
```

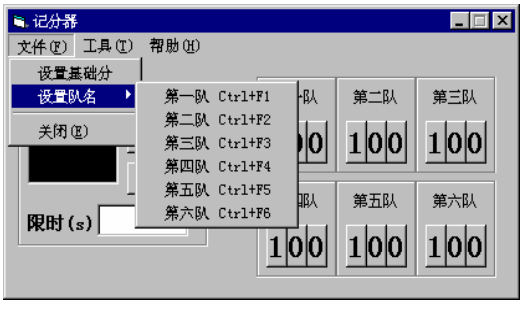

图 4-21　添加了快捷键

- 使菜单命令模糊

有时候，某些菜单命令毫无意义，如没有进行剪切或复制时，粘贴就毫无意义，这时，我们可以将它模糊，表示菜单命令不可选。

在程序运行时模糊一个命令，可以使用：

```
MnuEditPasteItem.Enabled = False
```

若要使一个命令不再模糊，可以使用：

```
MnuEtlitPasteItem.Enabled= True
```

- 使菜单命令消失

我们还可以使一个菜单命令完全消失，在菜单编辑器中，选中要消失的项，单击 Visible 复选框。

在代码中使一个菜单命令可见，可以使用：

```
Mnufileopen.Visible =True
```

在代码中使一个菜单命令消失，可以使用：

```
MnuEditcut.Visible = False
```

要记住，所有这些方法都是为了在窗体中添加菜单，并使菜单整齐美观，便于使用。

通过以上这些步骤，我们已经为我们的记分器程序创建好了菜单系统，只是由于我们还没有对菜单编写代码，现在的菜单命令还没有任何实际作用。

4.7　建立文件菜单

文件菜单中包含了“设置基础分”、“关闭”以及“第一队”等六个子菜单命令，为了使这些菜单命令可以正常工作，我们必须为其编写代码。

4.7.1　基础分代码

竞赛中往往会为各支参赛队设置一个相同的基础分，在本例中，我们使用属性窗口将这一基础分设置为“100”。为了适应用户的不同需求，我们允许用户重新设置基础分，这就是说，当用户选择该命令后，我们必须允许他们输入一个分值作为基础分，并根据输入，修改记分牌分值。在接受用户输入时，我们用到一个重要的 Visual Basic 函数——InputBox 函数。

- **专题讲座**：InputBox 函数。

使用 InputBox 函数可以建立一个输入框，一个输入框由用户可以改变的三个部分组成：标题条、用户提示字符串及正文输入框。输入框还包括用户不能改变的确定和取消命令钮。

标题条表明对话框的目的，用户提示字符告诉用户需要输入什么样的信息，正文输入框是允许用户输入信息的地方。

因为输入框要求用户输入信息，所以必须要用一个变量来存放用户输入文本框中的内容。

正常情况下，正文输入框是空的，但是用户可以在正文输入框中显示缺省值。一般一个正文框在屏幕的中间显示，但是用户也可以通过设置X、Y坐标值来指定输入框位置。

如下面的例子：

```
Reply= InputBox(“请输入您的选择”，“投票”，“刘明”，500，650)
```

弹出的输入框如图 4-22 所示。

图 4-22　使用 InputBox 函数弹出输入框

记分器例程中，我们使用 InputBox 函数来接受用户重新设置的基础分，“设置基础分”命令的代码如下：

```
Private Sub mnusetscoreitem_Click()
    basescore = InputBox("设定各参赛队的基础分(不超过 3 位数)：", "设置对话框")
```

```
    scorevalue = Val(basescore)
    Call setbasescore((scorevalue), (basescore))
End Sub
```

菜单命令只有一个事件，即 Click **事件，当用户单击选择某一菜单命令时，引发相应菜单命令的** Click **事件**。

最后一条语句：

```
Call setbasescore((scorevalue), (basescore))
```

表示在事件过程中调用一个通用过程，其中 scorevalue **和** basescore **为通用过程中要用到的参数**。

- **专题讲座：使用通用过程**

如果在程序代码的多个事件过程中包含了几乎相同的结构，这时，我们就可以在一个地方存放共同使用的指令，当任一事件过程用到这些指令时，只需进行调用即可，这就是通用过程。特别地，当我们需要改动那些共同使用的指令时，只需要在一个地方进行改动即可。

本例中，我们使用通用过程，一方面是介绍过程的用法，另一方面，使用通用过程，也可以使程序结构模块化，看起来清晰易读。本例的通用过程，就是针对用户的输入，重新设置各队的基础分。

Visual Basic **不能为用户创建一个通用过程，用户必须自己创建它，我们可以将通用过程创建和保存到两种类型的文件中：窗体文件**.Frm **和模块文件**.Bas。 **当通用过程保存在一个窗体文件时，这个通用过程可以被存储在相同窗体中的所有其它事件过程或通用过程使用，当一个通用过程存在一个模块文件时，这个通用过程可以被模块所属工程中的所有其他过程使用**。

本例中，我们只需要在记分器主窗体内调用通用过程，因此我们将通用过程保存在窗体文件中即可。

为了创建和保存一个通用过程到记分器的主窗体文件中，可以按下面的步骤执行：

1. **打开记分器主窗体的代码窗口**；
2. **在对象列表中选择“通用“**；
3. **从“工具”菜单中选择“添加过程”命令，弹出一个对话框**；
4. **在对话框中输入过程名** Setbasescore，**并选择“子过程”，“私有的”，如图** 4-23 **所示，单击确定**。

图 4-23　**添加过程对话框**

这样，我们就在记分器主窗体的代码窗口中得到了一个名为 Setbasescore 的通用过程，如图 4-24 所示。

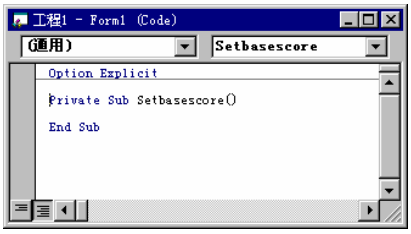

图 4-24　代码窗口中添加了通用过程

注意：

在每一个通用过程名后面都会有一对括号，这对括号称为参数表。最简单的过程有一个空参数表，用一对空括号表示。如果在调用通用过程时，需要传入参数，那么就可以把这些参数存放在括号内。

当一个程序调用一个通用过程时，它是根据名称来调用这个通用过程的，大多数情况下，我们不仅要使用通用过程的结构，还需要给通用过程一些数据信息，这就是参数。使用参数，用户可以把几个结构类似的过程合并起来，编写一个不明确的过程代替多个特殊的过程。

这样，我们就可以在调用通用过程时，同时传给通用过程一些参数，用来实现一个特定的功能，这样做的前提是我们必须先定义过程的参数表。

在一过程的参数表中可以包含一个或多个参数，同时，我们还可以在参数表中定义参数的数据类型，例如：

```
Privnte Sub Sum (first as Integer, Second as Integer , Sign as String).
```

这个例子定义了三个参数，First、Second 为整型，Sign 为字符型，指明参数的数据类型的目的只为了防止用户输入错误的数据，当然，我们也可以不说明一个参数的类型。

现在，大家对通用过程已经有了一个比较清楚的了解，下面我们来为记分器主窗体编写一个通用过程，其作用是根据用户输入，重新设置各队的基础分。

前面的步骤中，我们已经为记分器主窗体建创了一个名为 Setbasescore 的通用过程，现编写其代码如下：

```
Private Sub Setbasescore(var1 As Integer, var2 As String)
    Length = Len(var2)
    Select Case Length
        Case 0
        Case 1
            For i = 0 To 5
                Cmdone(i).Caption = Mid(var2, Length, 1)
                Cmdten(i).Caption = "0"
                Cmdhundred(i).Caption = "0"
            Next i
```

```
        Case 2
            For i = 0 To 5
                Cmdone(i).Caption = Mid(var2, Length, 1)
               Cmdten(i).Caption = Mid(var2, Length - 1, 1)
                Cmdhundred(i).Caption = "0"
            Next i
        Case 3
            For i = 0 To 5
                Cmdone(i).Caption = Mid(var2, Length, 1)
                Cmdten(i).Caption = Mid(var2, Length - 1, 1)
                Cmdhundred(i).Caption = Mid(var2, Length - 2, 1)
            Next i
        Case Else
            MsgBox "基础分设置无效，请重新设置", 48, "警告"
    End Select
End Sub
```

因为在 mnusetscoreitem **的** Click **事件过程中，我们已经提示用户输入不超过三位数，所以在这个通用过程中，我们分别根据用户无输入，输入一位、二位及三位数进行相应操作，若用户输入数太大，则显示提示信息，要求重新输入。**

通用过程参数表中定义了两个参数，代码中使用的 Mid **函数是一个字符串函数。**

一、 Mid 函数

Mid **函数取出字符串中指定数量的字符。**

语法：

Mid(String, Start [, length])

String **为字符串表达式，从中返回字符。**Start **指出字符串中被取出部分的字符位置。如果** Start **超过字符串的字符数，**Mid **返回零长度字符串。**Length **为可选参数，指出要返回的字符数。如果省略或** Length **超过文本的字符数（包括** Start **处的字符），将返回字符串中从** Start **到尾端的所有字符。**

二、 调用通用过程

为了调用一个通用过程并将参数传送给它，用户可以有三种方法：

ProcedureName Argument

ProcedureName (Argument)

Call ProcedureName (Argument)

其中，ProcedureName **表示过程名，**Argument **为参数。**

大家可以看到，在前面的 mnusetscoreitem_Click **事件过程中，我们使用了第三种方法调用通用过程。**

注意：

在 mnusetscoreitem_Click **事件过程中，调用通用过程时，我将参数** scorevalue, basescore **用括号括起来，倘若不这样做，程序编辑时容易产生错误。**

如果产生类型不匹配的错误，用括号将参数括起来，则情况就会大大改善。

当调用带有参数表的通用过程时，就会存在参数传递的问题，在传递参数时要注意两个问题：一是传递的参数与过程定义的参数数目要匹配；二是传递的参数与过程定义的参数类型要匹配。

当用户定义了一个具有参数表的过程时，参数表确定了它需要的参数，因此，在调用该过程时，我们必须确保传入的参数与过程定义的参数数目及类型要匹配。例如：

```
Private Sub openfile (Path as String)
```

调用 Call openfile (72，“Hello！”)

这个调用中犯了两个错误：参数的数目及类型不匹配。

三、　值传递与地址传递

一般情况下，用户在调用过程传递参数时，是按地址传递的，这是 Visual Basic 缺省默认的，这样，在过程中对参数的改变在返回时将保留下来，即调用过程后，改变了原来参数的值。

如果我们不想将被调用过程对参数的改变带回到调用过程中，则我们可以采用值传递方式，这时用户给过程一个参数，过程可以改变它所需要的值。然而，任何对这个变量的改变都只会限定到那个特定的过程中，对于调用它的过程，参数的值是不会变的。

为定义一个参数作为值传递，用户可以在参数表中用 Byval 关键字进行定义，如

```
Private Sub Found (Byval X As Integer, Y As Integer)
  X=5
  Y=5
End Sub
```

这表示，X 是值传递参数，Y 则是地址传递参数。

在另一主事件过程中调用上面的子过程，若主过程中对 X、Y 赋值均为 3，则调用后结果见表 4-7。

表 4-7　值传递与地址传递的区别

变量	调用前	子过程	返回主过程
X	3	5	3
Y	3	5	5

这里，大家注意体会值传递与地址传递的区别。

整个这一部分，我们编写好了“设置基础分”菜单命令的代码，同时介绍有关 InputBox 函数的内容。

下面，我们来编写设置队名的代码。

4.7.2　队名代码

“设置队名”是一个子菜单，在它下面包含了六个子菜单命令分别用来给六支参赛队设置各自的队名，要使用户能够设置队名，必须允许用户进行输入，并根据用户输入修改相应的队名显示。为此，我们仍然会用到 InputBox 函数。

“第一队”子菜单命令代码：

```
Private Sub mnufirstitem_Click()
    teamnam = InputBox("第一队队名：", "设置参赛队队名")
    If teamnam <> Empty Then Lblteam(0).Caption = teamnam
End Sub
```

“第二队”子菜单命令代码：

```
Private Sub mnuseconditem_Click()
    teamnam = InputBox("第二队队名：", "设置参赛队队名")
    If teamnam <> Empty Then Lblteam(1).Caption = teamnam
End Sub
```

“第三队”子菜单命令代码：

```
Private Sub mnuthreeitem_Click()
    teamnam = InputBox("第三队队名：", "设置参赛队队名")
    If teamnam <> Empty Then Lblteam(2).Caption = teamnam
End Sub
```

“第四队”子菜单命令代码：

```
Private Sub mnufouritem_Click()
    teamnam = InputBox("第四队队名：", "设置参赛队队名")
    If teamnam <> Empty Then Lblteam(3).Caption = teamnam
End Sub
```

“第五队”子菜单命令代码：

```
Private Sub mnufiveitem_Click()
    teamnam = InputBox("第五队队名：", "设置参赛队队名")
    If teamnam <> Empty Then Lblteam(4).Caption = teamnam
End Sub
```

“第六队”子菜单命令代码：

```
Private Sub mnusixitem_Click()
    teamnam = InputBox("第六队队名：", "设置参赛队队名")
    If teamnam <> Empty Then Lblteam(5).Caption = teamnam
End Sub
```

编好了这些子菜单命令代码，我们就可以使用它来设置各支参赛队的队名。记住，可以从菜单中选择设置，也可以按下快捷键弹出输入框进行设置。

4.7.3 关闭菜单代码

一般，在 Windows 的应用程序中，“文件”菜单下的最后一条命令都是“关闭”或“退出”命令，这已经成了约定俗成的作法。如果谁把“关闭”命令放到“工具”菜单中，恐怕这样的菜单就会使用户感到束手无策。

“关闭”菜单命令代码如下：

```
Private Sub mnuexititem_Click()
    End
End Sub
```

这段代码很简单，只要选择该命令，就会退出程序运行。

4.7.4　程序检测

现在，我们已经辛辛苦苦地编好了“文件”菜单的命令，我们马上来试试它与我们设想的效果有何偏差，具体操作步骤如下：

1. **单击工具栏中的**  **按钮，运行程序。**

2. **打开“文件”菜单，选择“文件”菜单，选择“设置基础分”命令，弹击如图** 4-25 **所示的输入框。**

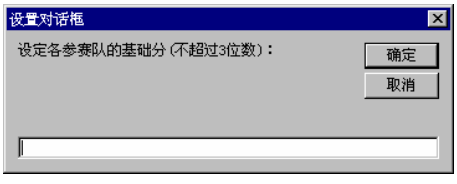

图 4-25　**设置基础分输入框**

3. **在输入框中输入**“150”，**单击确定。**

这里，最大只能输入一个三位数，否则将弹出警告对话框，要求重新输入。

4. **打开“文件”菜单，从“设置队名”中选择“第三队”，弹出一个输入框。**

5. **输入“梦之队”队名，单击确定。**

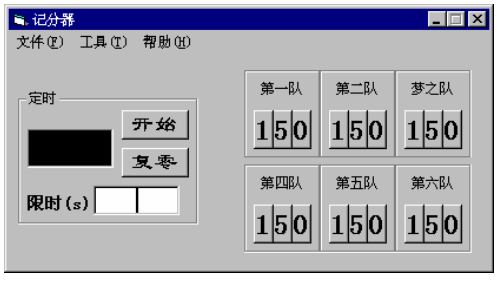

图 4-26　**重新设置了基础分及队名**

程序根据用户输入信息，重新设置了第三队的队名。这时得到的记分器窗体如图 4-26 **所示。**

6. **选择“文件”菜单下的“关闭”命令，退出程序运行。**

可见程序运行的结果完全符合我们的设计要求，我们可以松一口气了，因为我们已经编

好了程序中最复杂的一个菜单，下面我们再来慢慢地编写剩下的两个菜单。

4.8 建立工具菜单

根据实用的需要，本例的“工具”菜单中只包含了一条命令，即“统计得分”，这条命令，在竞赛中或竞赛结束时，用来统计出各队得分，并排列名次，显示出来，若得分相同，则以先后排序。

4.8.1 窗体的添加

以往编写的例子中，我们都是使用了一个窗体，事实上，一般的应用程序中，仅仅使用一个窗体是远远不够的，一个程序中往往包含许多窗体，它们的地位是平等的，用户可以在不同的窗体间进行切换。

注意：

这里所说的多个窗体，并不是指 MDI 多文档界面，多文档界面 MDI 与单文档界面 SDI 相对应，多文档应用程序允许同时显示多个文档，每一个文档都显示在自己的窗口中，例如：Word 就是一个多文档应用程序。

到底采用多文档界面，还是采用单文档界面，就要从应用程序的目的出发来确定，一般情况，我们编写程序时都是采用单文档界面。

多文档界面应用程序的创建

1. 创建 MDI 窗体

从“工程”菜单中选择“添加 MDI 窗体”，则在工程中加入了一个多文档界面的窗体，一个工程中只能有一个 MDI 窗体。

2. 创建子窗体

先在工程中添加一个新窗体，然后把它的 MDIchild 属性设置为 True，这样就可以创建一个 MDI 子窗体。

明白了这一点，现在我们只是在程序中添加一个新的窗体，而并非要创建多文档界面。向工程中添加一个窗体的方法很简单，只要选择“工程”菜单中的“添加窗体”命令即可这样就可以使工程中增加一个新窗体。一个工程的多个窗体，其地位都是平等的，而不像 MDI 多文档界面中有父窗体和子窗体之分。

新添加的窗体，名称为 frmscore，我们打算用它来统计得分的结果，为此，我在这个窗体上添加一个图形框（ PictureBox）和一个命令按钮（CommandButton）。

图形框专门用来显示图片，同时也可以显示文本。本例中我们将统计结果打印到图形框内，命令按钮用来关闭该窗体，返回主窗体。

4.8.2　统计代码

要想将竞赛结果显示出来，首先必须统计得分并排定名次，为此，我们必须在记分器主窗体内增加一些代码。

1. 双击“工程资源管理器”中的记分器主窗体 frmmain，以回到窗体；

2. 打开 frmmain 的代码窗口；

现在，我们要做的是把各队得分统计出来并排好名次，为了使程序结构看起来清楚一点，我们使用一个通用过程来排名次，然后在主事件过程中进行调用。

因为我们要在 frmmain 窗体中排好名次，并送到 frmscore 窗体中显示，所以需要把各队队员及其得分保存到一些变量中，而且要求这些变量在不同的窗体中都能够使用，因此，我们必须在一个标准模块中将这些变量定义为全局变量，以便在工程中的所有过程中都可以使用。

注意：

不要轻易使用全局变量，除非我们真的需要，因为使用太多的全局变量，往往会导致程序编译错误。

要在记分器程序中添加一个标准模块，可以按下面步骤来操作。

1. 选择“工程菜单”中的“添加模块”命令，这样就会在工程中增加一个模块；

2. 在模块中作如下定义

```
Option Explicit
Public score(5) As Integer
Public tm(5) As String
```

使用 Public 关键字把变量定义为全局变量，其用法同 Dim 关键字相同。在模块中，我们定义了两个一维数组，score 用来存放得分，tm 用来存放队名，两个数组都包含六个元素，索引号从 0~5，且相应的数组元素一一对应。如 Score 的第一个元素 Score(0)存放第一队得分，相应的 tm（0）中存放第一队队名。

定义了全局变量后，我们就可以回到记分器主窗体 frmmain 来编写代码了。

在 frmmain 的代码窗口中添加了一个用于排序的通用过程。排序的方法，在高级语言 Pascal、C 语言中，有好几种。这里采用替换排序，由于我们已将通用过程内用到的变量定义为全局变量，因此该通用过程不需要定义参数表，过程内可以直接使用变量。过程代码如下：

```
Private Sub order()
    Dim i, j, p As Integer
        For j = 0 To 4
            p = j
            For i = j + 1 To 5
                If score(i) > score(p) Then
                    p = i
                End If
            Next i
```

```
                midvar1 = score(j)
                score(j) = score(p)
                score(p) = midvar1
                midvar2 = tm(j)
                tm(j) = tm(p)
                tm(p) = midvar2
            Next j
    End Sub
```

过程中我们将得分进行了排序，高分在前，低分在后，同时，也将队名作了相应的调换，保证 Score(I)**与** tm(I)**都属于同一个队。**

写好了通用过程，我们就可以在 mnuscore（**统计得分**）**菜单的** Click **过程中进行调用。代码如下：**

```
Private Sub mnuscore_Click()
    Dim scorevalue As Integer
    Dim basescore As String
    For i = 0 To 5
        tm(i) = Lblteam(i).Caption
        score(i) = Val(Cmdhundred(i).Caption + Cmdten(i).Caption + Cmdone(i).Caption)
    Next i
    Call order
    frmscore.Show
End Sub
```

语句 frmscore.Show **表示显示** frmscore **窗体，** Show **是一个方法。**Show **方法表示显示一个窗体，与之对应，**Hide **表示隐藏一个窗体。值得注意，在显示一个窗体前首先要进行加载，这是程序自动完成的，而隐藏一个窗体后，并不将其卸载掉。**

现在我们已经在 frmmain **窗体将排好名次的得分及相应队名保存到了** Score **和** tm **两个数组中，要将结果显示出来，还必须转到** frmscore **窗体中去。**

4.8.3 打印代码

frmmain **窗体中排好的结果，还必须在** frmscore **窗体的图片框中进行显示，这样显示的一般格式为：**

第***名　　**** **队　　*****（**分**）**

因此，我们希望有一种数据类型，使得它的一部分代表名次，一部分代表队名，另一部分代表得分，这样的数据类型，Visual Basic **并没有把它作为一个基本的数据类型来定义，但是我们可以通过** Visual Basic **的基本数据类型来自己定义一种新的数据类型，以满足我们的要求。为此，我们要使用** Type **语句。**

Type **语句只能在模块级使用。使用** Type **语句声明了一个用户自定义类型后，就可以**

在该声明范围内的任何位置声明该类型的变量。可以使用 Dim、Private、Public、ReDim 或 Static 来声明用户自定义类型的变量。

在标准模块中，用户自定义类型按缺省设置是公用的。可以使用 Private 关键字来改变其可见性。而在类模块中，用户自定义类型只能是私有的，且使用 Public 关键字也不能改变其可见性。

有了这些知识，我们就可以来定义上面提到的数据类型。要自定义数据类型，需要在标准模块中进行，为此，从“工程资源管理器”中打开我们刚才定义全局变量使用过的标准模块 Modulel ，在模块中作如下定义：

```
Type printrecord
    order As String
    teamname As String
    score As Integer
End Type
```

这里定义的数据类型我们可以在工程中的所有事件过程中使用。

现在，回到 frmscore 窗体，打开其代码窗口，在显示一个窗体前，首先要将该窗体加载，因此，我们只要将打印统计结束的代码放到窗体加载事件过程中，那么就可以使统计值在 frmscore 窗体中显示出来。

代码如下：

```
Private Sub Form_Load()
    Dim i As Integer
    Dim record(5) As printrecord
    record(0).order = "第一名"
    record(1).order = "第二名"
    record(2).order = "第三名"
    record(3).order = "第四名"
    record(4).order = "第五名"
    record(5).order = "第六名"
        Picture1.Print
        Picture1.Print "名次", "队名", "得分"
        Picture1.Print
        For i = 0 To 5
            record(i).teamname = tm(i)
            record(i).score = score(i)
             Picture1.Print record(i).order, record(i).teamname, record(i).score
        Next i
End Sub
```

注意：

代码中如何引用自定义数据类型中的各项。

Print 为 Visual Basic 的方法，表示在相应的对象上打印出引号中的内容。

关闭按钮代码：

```
Private Sub cmdback_Click()
    Unload frmscore
End Sub
```

Unload 为一条 Visual Basic 语句，表示把窗体卸载。

到此为止，我已经成功地编完了“工具”菜单，下面就来检测一下我们的工作是否得到认可。

4.8.4 设置启动对象

现在，我们的工程中包含了两个窗体，frmmain 和 frmscore。这样，当程序运行时，Visual Basic 就无法确定到底该先从哪个窗体起动，因此，在运行程序前，我们必须设置好起动窗体，设置方法如下：

1. 从“工程”菜单中选择“工程属性”命令，弹出一个对话框，如图 4-27 所示。
2. 在“工程”菜单的下拉列表中选择 frmmain。

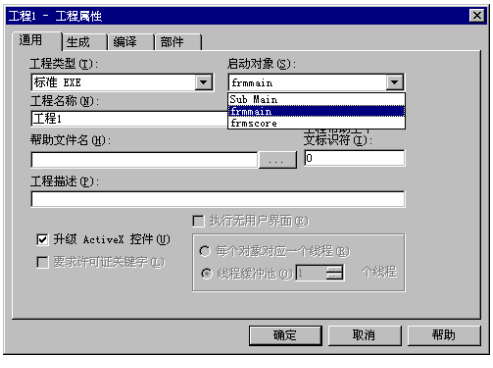

图 4-27　工程属性对话框

这样我们就把 frmmain 设置为起动窗体。

3. 单击确定。

我们就设置好了一个多窗体工程的启动窗体，程序运行就从这个窗体开始。如果以后用户再向工程中添加了新的窗体，则 Visual Basic 仍然会从以前设定的起动窗体中起动，除非用户更新设定。

下面，我们来测试一下“工具”菜单，可按下面步骤操作：

1. 单击工具栏中的  图标，运行程序，注意程序从 frmmain 窗体运行；
2. 使用鼠标左、右键，调整各队得分；
3. 选择“工具”菜单中“统计得分”，显示如图 4-28 所示的窗体，其中显示了统计排名结果；

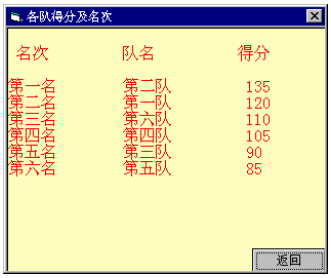

图 4-28　统计得分结果

4. 单击“关闭”按钮，关闭 frmscore 窗体。

程序到现在，我们已经做完了绝大部分工作，但是还不能松懈，“帮助”菜单还没有编写。

4.9　建立帮助菜单

一般应用程序的帮助菜单中，往往会列出主题索引帮助，产品使用说明及版本等一系列内容。本例中，在帮助菜单中主要列出记分器的使用说明。为此，我们需要在工程中，再添加一个新的窗体，当用户选择“帮助”菜单下的“帮助信息”时，用该窗体来显示帮助信息。

首先，在记分器主窗体 frmmain 中添加如下事件过程：

```
Private Sub mnuhelpitem_Click()
    frmhelp.Show
End Sub
```

当用户选择“帮助信息”时，显示一个名为 frmhelp 的窗体，用来显示提示信息。

添加 frmhelp 窗体

从“工程”菜单中选择“添加窗体”命令，在工程中添加一个新的窗体，在该窗体上添加一个文本框，用来显示帮助信息，增加一个按钮用来返回主程序。

这里我们设置文本框的 MultiLine 属性为 True，设置 ScrollBars 属性为 2-Vertical。

MultiLine 属性用来指示 TextBox 控件是否能够接受和显示多行文本。当其值为 True 时，允许多行文本；当其值为 False 时，忽略回车符并将数据限制在一行内。

ScrollBars 属性用来指示一个对象是有水平滚动条还是有垂直滚动条。对于 TextBox 控件，ScrollBars 属性的设置值为：当其值为 0 时，表示无滚动条；当其值为 1 时，表示显示水平滚动条；当其值为 2 时，表示显示垂直滚动条；当其值为 3 时，表示两者都显示。对于 ScrollBars 的属性设置值为 1（水平）、2（垂直）、或 3（两种）的 TextBox 控件，必须将 MultiLine 属性设置为 True。

滚动条只在对象的内容超过对象的边框时才被显示在对象上。例如，当 TextBox 控件不能显示它所有的文本行时，一个垂直滚动条（VScrollBar 控件）将被显示在 TextBox 控件上。如果 ScrollBars 被设为 False，那么对象将没有滚动条，而不管其内容如何。

Frmhelp 窗体加载事件过程代码如下：

```
Private Sub Form_Load()
    txthelp.Text = Chr(13) + Chr(10) + Chr(13) + "该程序具有两个功能:定时功能和记分_
    功能" + Chr(13) + Chr(10) + Chr(13) + Chr(13) + Chr(10) + Chr(13) + "(一) 定时功_能
    说明: " + Chr(13) + Chr(10) + Chr(13) + "1、在限时文本框中输入限定时间, _
    以秒为单位; " + Chr(13) + Chr(10) + Chr(13) + "2、单击开始按钮, 开始计时; " +
    _Chr(13) + Chr(10) + Chr(13) + "3、时间每秒减少 1, 直到减到 0 为止; " + Chr(13) _+
    Chr(10) + Chr(13) + "4、时间到, 显示提示对话框, 并发出声音; " + Chr(13) + _Chr(10)
    + Chr(13) + "5、如果提前答题完毕, 可按复零按钮, 时间减少到 0, 但 _不显
    示提示对话框。" + Chr(13) + Chr(10) + Chr(13) + Chr(13) + Chr(10) + Chr(13) _+ "
    (二) 记分功能说明: " + Chr(13) + Chr(10) + Chr(13) + "1、单击鼠标左键加_分,
    单击鼠标右键减分; " + Chr(13) + Chr(10) + Chr(13) + "2、得分的个位数每 _
    次单击可加减 5 分; " + Chr(13) + Chr(10) + Chr(13) + "3、得分的十位数每次单 _
    击可加减 1, 即 10 分; " + Chr(13) + Chr(10) + Chr(13) + "4、不可以通过鼠标单 _
    击直接改变百位数。"
End Sub
```

这里，当 frmhelp 加载时，在文本框内显示帮助信息，添加了这段代码后，运行程序，按下 F1（帮助信息的快捷键），如图 4-29 所示。

关闭按钮代码：

```
Private Sub Cmdexit_Click()
    Unload frmhelp
End Sub
```

到此为止，我们已经编好了所有的菜单代码。大家不妨进一步地对记分器例程进行全面测试，能使这么多的菜单为我们服务，一定会感到一种极大的成就感。

现在工程中，包含了三个窗体：frmmain， frmscore 和 frmhelp ，还有一个模块 module 。这从“工程资源管理器”中可以看出。

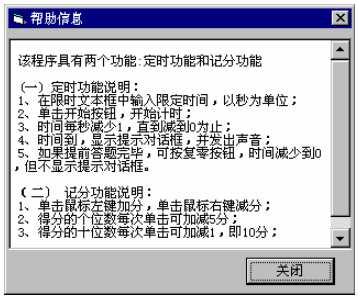

图 4-29 打开帮助信息窗口

4.10 建立弹出菜单

对于 Windows 的用户，看到弹出菜单一点也不会觉得奇怪，因为我们几乎每天都在使用

它。弹出菜单经常可以快速地在屏幕上列出常用命令表，任何菜单或子菜单都可以作为弹出菜单出现。

为了创建一个弹出菜单，我们必须使用 PopupMenu 命令，例如：

```
PopupMenu mnuEdit
```

这条语句就是用来显示一个名称为 mnuEdit 的弹出菜单。

注意：

这样做的前提是我们必须已经为窗体设计好了一个 mnu Edit 菜单。

在 PopupMenu 命令中，我们还可以指定弹出菜单出现的位置，如下面的例子：

```
PopupMenu  mnuTools ，450，320
```

这个例子表示在 X 坐标为 450 和 Y 坐标为 320 处显示一个弹出菜单，当我们改变了度量单位后，那么弹出菜单在屏幕中出现的位置将发生变化，必要时，我们可以更改坐标值。

当然，上面的坐标参数是可以省略的，在缺省情况下，弹出菜单出现在鼠标所指的地方。

一般情况下，弹出菜单是当我们单击鼠标右键时显示的，因此，要显示弹出菜单就要用到 Mouseup 事件，因为这一事件是区别鼠标左右键的。

既然建立弹出菜单如此容易，那么我们下面就在记分器程序建立一个弹出菜单。在记分器主窗体 frmmain 中增加如下代码：

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 2 Then
        PopupMenu mnufile
    End If
End Sub
```

就这样，在运行中，我们在窗体自由区单击鼠标右键时，就会出现一个弹出菜单。如图 4-30 所示。

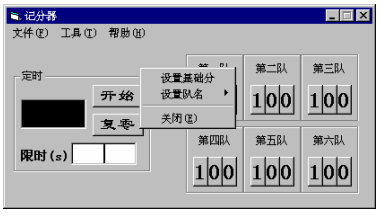

图 4-30　建立弹出菜单

4.11　本章小结

程序已经完成，本例中，除程序基本功能外，我们主要为窗体添加了菜单。在实现菜单功能时用到 InputBox 函数来接受输入，MsgBox 函数来显示提示信息，同时还教会大家如何在一个工程中使用多个窗体。

第四部分　文本处理技巧

第一章　字体

Visual Basic 6.0 有着强大而复杂的处理文本的功能。作为一个文本，其大小，形状和颜色都可以被我们所看到，那么我们就可以把它当成可视化的元素来处理，这可以大大增强文本所能提供的信息。试想如果我们的报纸，杂志如果没有了丰富多彩的字体，颜色，取而代之的是千篇一律的字体，大小，和颜色，那又怎么能吸引我们阅读它呢？

在实际编程过程中，我们会接触到各式各样的文本，不论一句话，还是一篇文章，它们都属于文本，但这些文本都是通过字体来显示的，通过控制字体我们可以改变字符的大小，风格，粗细，使程序看起来更丰富多彩，更具吸引力。请看下列字体的对比如图 1-1。

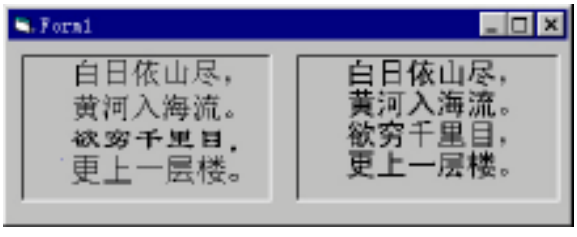

图 1-1　字体对比

你可以让你编的程序使用各种各样的字体，但是切记，编程的最终目的。是让用户使用方便，这就要求我们从用户角度思考问题。如果你为应用程序选择了一种用户操作系统上并没有的字体，那么用户在使用程序时就会带来一些麻烦。例如，你的应用程序开头标题使用“魏碑”而用户操作系统并没有安装这种字体，那么此时，windows 将会从用户的字库选择一种最接近的字体。一般为仿宋体来显示，这还算好，如果 windows 选择的字体放大了文本，那么你的应用程序有一些文本就会产生重叠，覆盖现象，这样就大大降低了你的应用程序的可移值性。所以，在选择字体上一定要慎重。

避免该类问题的一种方法是，使用大部分用户操作系统都有的字体。

1.1　程序实例：检查可用字体

我们可以利用 Printer 和 Screen 对象的 Fonts 属性检测用户系统和打印机中是否有匹配的字体。Fonts 属性的值在这里为一数组，是打印机或屏幕可用的所有字体的列表。我们可以通过检查该数组，查出所要的名称字符串。我们现在设计一个程序来检查用户系统中是否有“魏碑”这种字体。

1.1.1　用户界面的创建

在这里我们用到了三个控件：

- Button
- Lable
- Textbox

其对象属性的具体设置值如下表所示：

表 1-1　属性设置值

控件	属性	设置值
Lablel	Caption	空
Lable2	Caption	输入要查询字体
Form	Caption	检查字体程序
TextBox	Name Text	Text1 魏碑
Button	Name Caption	Button1 检查

设计好的界面如图 1-2 所示：

图 1-2　界面设计。

1.1.2　函数设计代码

我们设计一个如上所述的取得字体的函数 GetFont ( )。

```
Function GetFont(FontName As String) As Boolean
Dim i As Integer
Dim com As Integer
com = 0
For i = 0 To Screen.FontCount - 1
        Com = StrComp(FontName, Screen.Fonts(i))
        If com = 0 Then
```

```
                GetFont = True
                Exit Function
            End If
        Next i
        GetFont = False
    End Function
```

1.1.3 事件过程代码

我们在 Button1_click ( )**事件过程中添加如下代码**。

```
Dim s As String
 s = Text1.Text
If GetFont(s) Then
 Label1.Caption = "您的系统中，存在这种字体！"
End If
If Not GetFont(s) Then
 Label1.Caption = "对不起！您的系统中没有这种字体。"
End If
```

它所起的作用是当用户单击“开始检查”按钮时，程序取出 Textl、text **的值(即** Textbox **中的内容)送入** GetFont() **函数中，返回得到一个** Boolean **型的值，根据此值来到断用户是否有此种字体，并显示不同的文本**。

运行程序。

此程序的界面如图。输入“魏碑”后，按开始检查。结果如下。如图 1-3 **所示**。

图 1-3　**运行结果**

1.2 设置 Font 属性

我们可以通过控制显示文本的窗体或控件中的 Font **属性来设置文本的可视化特征**。

- **字体名**

- 字体大小
- 特别特征(斜体……)

在程序设计段，我们可以双击属性对话框中“字体”项，弹出字体对话框，通过它来更改字体的属性，例如字体，大小，下划线等等，如图 1-4 所示：

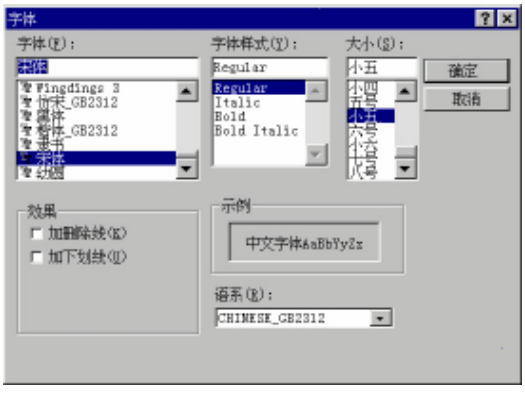

图 1-4　字体对话框

在程序运行阶段，我们同样可以控制窗体、控件中的字体特征。只不过要利用到 Font 对象的一些属性。见下表。

表 1-2　字体对话框中的一些属性设置

属性	类型	说明
Name	String	指定字体的名字，如”True Type”，“楷体”。
Size	Single	用磅为单位来指定字体的大小(打印时，每英寸为 T2 不旁)。
Bold	Boolean	如果为 True 则文本为黑体。
Italic	Boolean	如果为 True 则文本为斜体。
Strike	Boolean	如果为 True，则在文本中画一条删除线.
Underline　Boolean	Boolean	如果为 True，则在文本中添加下划线.
Weight	Integer	返回或设置字体笔画的粗细。

例如：下列代码是以一个程序的主 Form 设置字体属性:

```
With    Form1.Font
        Name=“宋体”          ’ 把字体改为宋体
        Size=12              ’ 把字体大小改为 12
        Bold=True            ’ 把字体改成黑体
End with
```

我们也可以在 Font 对象中创建一组字体属性，存储在 StdFont 类当中，用以声明 Font 对象例如：

```
Dim MyFont As New StdFont.
With MyFont
Name=“楷体”
```

Size=12

Bold=True

Weight=10

然后我们就可以用我们定义的 MyFont **对象定义别的对象**。

Set Forml.Font=MyFont.

在这里读者要注意一条，我们创建新的 Font **对象并把它定义为** StdFont **类的，要使用“引用”对话框(在“工程”菜单中的“引用”项)。用来对标准** OLE **类型的设置。如图** 1-5 **所示：**

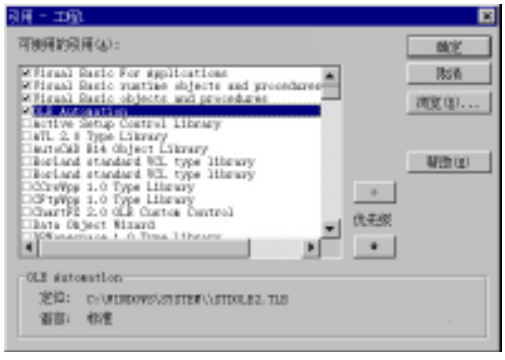

图 1-5 **“引用”对话框**

1.3 本章小结

本章介绍了如何在程序设计期间，以及程序运行期间调整字体的可视化特征字体，大小，粗斜体等等。读者现在对字体控制应该有一个比较粗浅的认识，下一章中将具体介绍如何控制一个字的可视化特征。

第二章　文本的显示

上一章中简要地叙述了控制窗体和控件上的字体，本章将介绍如何在窗体中输出文本，以及控制文本的可视化特征。

2.1　print 方法的使用

在窗体和图片框中显示文本。要用到 print 方法，其具体语法为：

[Object.] print [OutPutList]

其中 object 为想要在哪个控件上显示文本，如果省略，则默认为当前窗体。

OutPutList 参数为想要输出的文本，想要输出多个项可以用逗号或分号隔开，但用逗号和分号又有所不同。当用分号隔开的两个输出项在一行上显示，如果用的是逗号，在打印一项后，VisualBasic 将跳到下一个制表列打印下一项。例如下面两条语句将分别显示这种情况。

print“千山鸟飞绝”；“万径人踪灭。”

print “千山鸟飞绝，”，“万径人踪灭。”如图 2-1 所示。

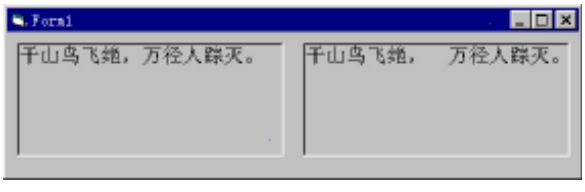

图 2-1　比较结果。

在这里要注意如果显示的文本超出了窗体或图片框的宽度与长度。文本将会被截断，而且这种截断方式不能通过滚动窗体或者是图片框来得到被截断的文本。在什么地方截断文本，取决于文本打印开始点的位置坐标。

那么这里就涉及到如何控制在窗体和图片框中显示文本的问题，让我们先介绍一下与窗体以及图文框有关的基本知识。

- Currentx 和 Currenty 属性，通过它们，可以获得或者设置当前输出焦点。
- Cls 方法，清除窗体和图文框中的一切文字与图片，并使当前输出焦点回到 10,07 点，即窗体或图片框的左上角。
- TextHeight 和 Text Width 方法，这两个方法分别用来获得和设置窗体与图片框中输出文本行的写度和宽度、具体用法是：

[object.] TextHeight (string)

Iobject,J Text width (string)

如果 String 参数包含输入的回车符(chr$(13))，则文本对应于多行，TextHeight 属性将返回字符串中所有行的文本的高度。如果有回车符，TextHeight 将只返回一行的尺度。

下面看一个如何控制在图片框中显示文件的例子。

先看看这个程序所要实现的功能，用户可以在文本编辑框中输入想要显示的文本，然后按“显示”按钮，文本便会在图片框中显示出来。第一个字母放大显示，且能够自动换行，当遇到单词被截断时，能够在词尾显示“-”连接符。

2.1.1 用户界面的创建

在这个程序中我们用到了常控件中的三个控件，它们分别是

- PictureBox
- TextBox
- Button

它们的属性设计值如下：

表 2-1 属性设置

对象	属性	设置值
PictureBox	Name	Picture1
	Height	200
	Width	300
Button	Caption	显示
	Name	Buttonl
TextBox	Name	Textl
	Text	空

在这里把 TextBox 的 TabIndex 属性设为 0，就可以控制当程序运行时，输入焦点就位于 TextBox 中，以此提示用户输入文本。

设计好的界面如图 2-2 所示：

图 2-2 界面设计

2.1.2　程序代码

首先要控制当用户单击 Buttonl 时，文本就可以显示出来。所以显示文本代码部分就应写在 Buttoul 的 click 事件中，为了使文本能重复显示，即当输入一个新文本，单击显示按钮时，原来的文本会被当前文本覆盖掉。实现这一功能，想必读者早已猜到了，是用 Picturebox 的 cls 清屏方法，即：

Picturel.Cls

再看显示文本部分，当程序从 Textbox 中获得 Text 部分后，便开始处理文本。它应该先在文本开头输出两个空格，然后取出它的第一个字符并放大显示，然后按顺序输出文本。当遇到文本框边界时，要判断被截断的是否为词，如果不是，就应换行，否则要输出一个连接符号“-”，请看代码。

用 If Then 来判断当(Picture1.width-Picture1.CurrentX)的值（即当前输入焦点距图文框的边界的值）大于 2 倍的一个字符的宽度时，便要判断此时被截断的是否为单词，用取词函数 Mid$（）取出当前字符和下一字符并判断是否属于 A~Z 或 a~z。当都是时，则输出连接符“-”和一个回车符 Chr$(13) 。如果不是，则只输入出一个回车符 Chr$(13) 。

完整的代码如下：

```
Dim m, s, t, As String
 Dim i, j As Integer
    s = Text1.text
  Picture1.Cls
  For i = 1 To Len(s)
    m = Mid$(s, i, 1)
    If i = 1 Then
       Picture1.Print "    ";
       Picture1.FontSize = 20
       Picture1.Print m;
       Picture1.FontSize = 10
       Picture1.CurrentY = TextHeight(m)
    Else
      Picture1.Print m;
    End If
    If (Picture1.Width - Picture1.CurrentX) <= 2 * TextWidth(Mid$(s, i + 1, 2)) Then
       t = Mid$(s, i + 1, 1)
        If (t > "A" And t < "Z") Or (t > "a" And t < "z") Then
           Picture1.Print "-";
           Picture1.Print Chr$(13); " ";
        Else
           Picture1.Print Chr$(13); " ";
```

```
        End If
    End If
    Next i
```

进行调试程序

当界面和代码设计好后运行程序，在文本中输入下列文本：

One of the most important reasons for forming the lifelong habit of reading is to have a means of plunging into the midst of an adventure at any moment that you wish.

单击显示按钮后，文本输出如图 2-3 **所示：**

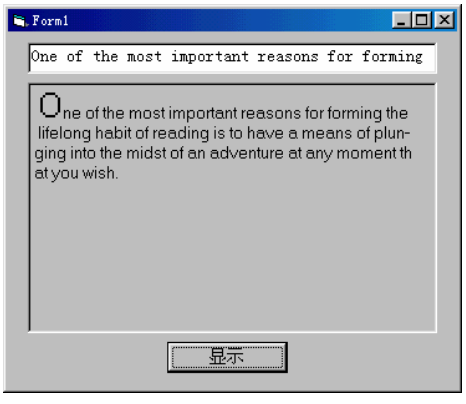

图 2-3　**运行结果**

这个程序我们只用输入 chr(13)**控制换行，还有一种方式可以控制换行，即通过控制当前输入焦点来做到。用以下语句可以将** cuttenty **设置到第** n **行：**

CurrentY=[object.JTextlteight (string)*m-1]

Text Height **和** Text width **方法所返回的字符串宽度与高度，都考虑了该对象的字体大小与风格，因此对于许多具有均衡宽度的字符，本方法将会很有用处。**

2.2　Format **函数的使用**

在显示数字、时间、日期的格式上，Visual Basic **提供了巨大的灵活性，我们可以很容易地用国际格式来显示。**

2.2.1　数字输出格式的控制

Format **函数把数字值能换为文本字符串，从而能够对该字符串的外观进行控制。例如，可以指定小数的位数，前导和尾部零，以及货币格式。它具体的语句如：**

Format (Expression, Format, Firstday of week, Firstweek of year)

Expression **参数指定了转换的数值，**format **参数是字符串，该串是由一些符号组成的，这些符号用于说明如何确定该数字的格式。表** 2-2 **列出了一些单用的符号：**

表 2-2　Format 函数的常用格式符号

符号	描述
0	数字保留区，如果需要，在本位置打印尾部前导零
#	数字保留区，不打印尾部或前导零
.	小数保留区
,	4 位分隔符
—+$( ) 空格%	字母字节，各种字符，都要按格式字符串中打入的原样，明确显示出来

我们举一些使用 Format 函数的例子，并用前面学过的图文框把这些例子显示出来。

一 界面设计

我们使用图文框和按钮这两个控件

读者要把图文框大小调整适当，因为我们在这里并不判断所输出的文本是否被截断，按钮的 Caption 仍为“显示”。

在 Buttonl_Click 事件中写入如下代码：

```
Dim FormatTemp As String
    Picture1.Cls
    Picture1.FontSize = 10
    FormatTemp = Format(9728.56, "00000.000")
    Picture1.Print FormatTemp
    FormatTemp = Format(9728.56, "#####.###")
    Picture1.Print FormatTemp
    FormatTemp = Format(19728.56, "##,##0.000")
    Picture1.Print FormatTemp
    FormatTemp = Format(1, "0.00%")
    Picture1.Print FormatTemp
    FormatTemp = Format(975.5, "$000.00")
    Picture1.Print FormatTemp
```

二 运行程序

单击“显示”按钮，可以看到结果如图 2-4 所示。数分隔符是句号，千位分隔符为豆号，但是，分隔符的正确显示得依靠 windows 控制面板中的“区域”项的设置的输出格式

2.2.2　日期与时间输出格式的控制

要想控制日期和时间的输出格式，就应在 Format 函数中正确使用代表日期和时间的符号。下面我们使用 Format 函数和 Now 参数来标识和格式化当前日期与时间。我们要注意，对于日期分隔中与（/）AM/PM 和时间分隔号（：）等等这些符号来说，其显示格式会因 Windows 中的“区域设置”的设置项改变而不同。那么我们在开发程序时，日期与时间是以短期的格式并配合代码的国标标准来显示。而在程序运行时，系统的国际标准和代码的国际标准可能并不相同，而程序显示的格式则以系统的国际标准为准，这样就有可能出现程序输

出和你的设想结果不一致的情况。我们下面的例子默认读者的系统的 Windows，并且“控制面板”中的“区域”项设置为“英语（美国）”。

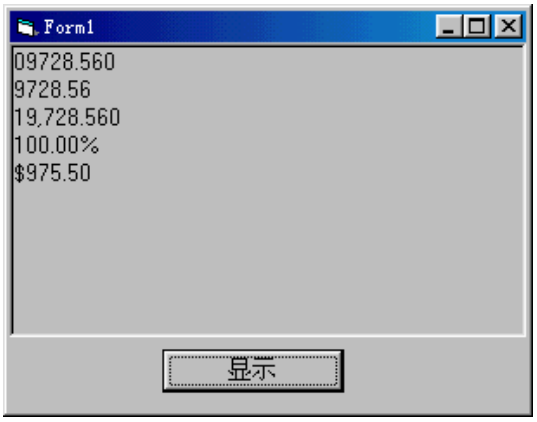

图 2-4　运行结果

我们这个例子界面设计和上次一样，这里就不在重复了。下面代码部分，每一句后面都以注释的形式写出了每句的结果：

```
Format(Now, “m/d/yy”)                    ’返回值为：1/27/93
Format(Now, “dddd, mmmm dd, yyyy”)       ’返回值为：Thursday, October 8, 1998
Format(Now, “d-mmm”)                     ’返回值为：8-Oct
Format(Now, “mmmm-yy”)                   ’返回值为：October-98
Format(Now, “hh:mm AM/PM”)               ’返回值为：09:23 AM
Format(Now, “h:mm:ss a/p”)               ’返回值为：9:23:00 a
Format(Now, “d-mmmm h:mm”)               ’返回值为：8-October 9:23
```

2.2.3　文本输出格式的控制

我们通过 Format 函数同样可以控制文本的输出格式，例如下面几条语句可以改变文本的大小写。

```
Format Temp=Format(“TITANIC,"<"”)返回值“titanic”
Format Temp=Format (“My Heart will Go on”,“>”)
返回值“MY HEART WILL GO ON”
```

Visual Basic6.0 提供了几种与 Format 函数一起使用的标准格式。在 Format 函数的 format 参数中，可以使用名字来指定这些格式，而不需要用那些指定的符号。格式名义须用双引号“”括起来，下面的例子都用了命名格式：

```
FormatTemp = Format (None ,"long Time ")      ‘返回值“12：36：12”
FormatTemp = Format (37586 “Currency”)        ‘返回值“￥37，586、00”
FormatTemp = Format (123， “Yes/No”)          ‘返回值“Yes”
```

表 2-3　Format 函数的格式名

命名的格式	描述
General Number	显示设有千位分隔符的数字。
Carreney	显示带千位分隔符的数字；在小数点的右边显示两位数字。而输出则依据用户的系统设置。
Fixed	在小数点的右边至少显示一位数字；在小数点的右边显示两位数字
Standord	显示带有千位分隔符的数字，在分隔符的左边至少显示一位数字而在分隔符的右边至少显示两位
Percent	该值乘以（00）在后面加一个百分号
Scientific	如果 expression 同时包含日期和时间，则显示它们。如果 expression 只包含日期
GeneralDate	或包含日期，则缺少的信息不显式。日期的显示取决于用户的系统设置。
LongDate	使用用户的系统设置所指定的 logDate 格式。
MedtumDate	使用 dd-mm-yy 格式（例如，03-Apr-98）。日期的显式取决于用户的系统设置。
ShortDate	使用用户的系统设置所指定的 shortDate 格式。
ShortDate	使用用户系统的长时间格式显示时间，包括时、分、秒。
LongTrue	使用 Hh:mmAm/Pm 格式，显示小时，分钟和 AM 或 PM
Medium Time	使用 hh:mm 格式，显示小时和分钟
Short Time	任何非零数字（通常为－1）为 YES，零 NO。
Yes/No	任何非零数字值（通常为－1）为 TRUE ，零 FALSE
on/off	任何非零数字值（通常为－1）为 ON，零为 OFF。

Format 函数还支持其它许多特殊字符，如百分点保留区和指数等等。

如果读者想知道更多详细的信息，请参考 Visual Basic6.0 的联机帮助中的 Format 函数的内容。

2.3　文本的选择

我们上面介绍了控制文本的一些属性，下面我们来介绍使用剪贴板，文本框和组合框来选择文本，以及一些有用的属性。例如，我们可以引用控件内选定的文本块（突出显示），能够为用户创建剪切和粘贴功能。下面我们看一个小例子。

2.3.1　用户界面的创建

我们在这个程序中只用了二个控件，TextBox 和 CommandButton 属性如表 2-4

表 2-4　属性设置

对象	属性	设置值
TextBox	Name	Text1

	Text	空
CommandButton	Name Caption	Command1 确定

设计好的界面如图 2-5：

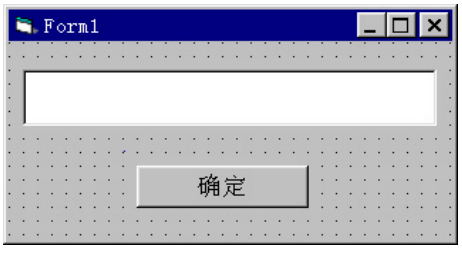

图 2-5　界面设计

2.3.2　程序代码

我们在 Command1_Click ( )事件中写入如下代码：

```
Text1.Text = "I am the king of the world!"
Text1.SetFocus
'从第一个字符之前开始突出显示。
Text1.SelStart = 0
'突出显示，一直到文本尾。
Text1.SelLength = Len(Text1.Text)
```

我们运行程序并单击“确定”按钮，得到（图 2-6）如示结果：

图 2-6　运行结果

如果给 SelText 赋值新的字符串，该字符串将替换选定的文本，并且把插入点放在新插入文本尾之后。例如，下面这条语句将用字符串“Wo,wooooooooooooo!”替换选定的文本：

```
Text1.SelText = "Wo,wooooooooooooo!"
```

如果没有被选定的文本，则该字符串就粘贴到文本框中的插入点处。

有关 SelStart , SelText , SelLength 的具体属性如下表：

表 2-5 SelStart, SelText, SelLength 的属性

属性	描述
SelStart	Long 整数，用来指定选定文本块的起始位置。如果没有选定的文本，则该属性指定插入点的位置。若设置值为 0，所指示的位置是在文本框或组合框中第一个字符之前。若设置值等于文本框或组合框中文本的长度，所指示的位置，是在控件中最后一个字符之后。
SelLength	Long 整数，指定所选的字符个数。
SelText	String，包含选定的字符（如果没有字符被选定的话，就是空字符串）。

2.4　Clipboard 的使用

剪贴板，即 Clipboard 对象没有属性或事件，但是它有几个与环境剪贴板之间往返传送数据的方法。它们是：

- GetText 和 SetText 方法，用来传送文本。
- GetData 和 SetData 方法，用来传送图形。
- GetFormat 和 Clear 方法，可以处理文本和图形两种格式。

我们先来看怎样使用剪贴板剪切、复制和粘贴文本。

在这里我们使用 Clipboard 的 SetText 和 GetText 方法向剪贴板和从 Clipboard 传送字符串数据。其具体使用情况如图 2-7：

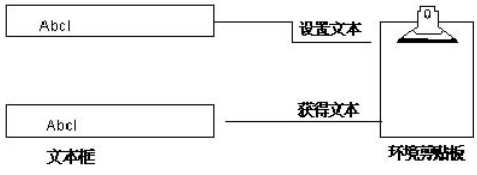

图 2-7　用 SetText 和 GetText 方法向剪贴板和从剪贴板传送数据

我们使用 SetText 方法将文本复制到 Clipboard 上，替换掉原先的文本，其语法如下：

Clipboard.SetText Data[, Format]

使用 GetText 方法可以返回储存在剪切板上的文本，语法如下：

S = Clipboard.GetText ( )

于是我们可以使用 SetText 和 GetText 方法与上面介绍的选择属性结合起来使用，可容易地编写文本框的“复制”，“剪切”和“粘贴”命令。

下面我们编写一个可以实现“复制”，“剪切”和“粘贴”功能的小程序。

2.4.1　用户界面的创建

在这个程序中使用 TextBox, CommandButton 两种控件；属性设置如下表。

表 2-6 属性设置

对象	属性	设置值
Form	Name	Form1
TextBox	Name Text	Text1 空
Command1	Caption	复制
Command2	Caption	剪切
Command3	Caption	粘贴

2.4.2 程序代码

在三个 CommandButton **的** Click **事件中加入如下代码：**

```
Private Sub Command1_Click()
   Clipboard.Clear
   Clipboard.SetText Text1.SelText
End Sub

Private Sub Command2_Click()
   Clipboard.Clear
   Clipboard.SetText Text1.SelText
   Text1.SelText = ""
End Sub

Private Sub Command3_Click()
   Text1.SelText = Clipboard.GetText()
End Sub
```

需要注意的是在“复制” 和“剪切” 这两个过程，都要先用 Clear **方法将** Clipboard **清空。然后“复制” 和“剪切” 这两个过程，都用下面的语句将** Text1 **中所选择的文本复制到** Clipboard **上：**

```
Clipboard.SetText Text1.SelText
```

在“粘贴”命令中，GetText **方法将返回** Clipboard **上当前的文本字符串。然后用一条赋值语句将该字符串复制到文本框的指定位置** (Text1.SelText)。**如果当前没有被选定的文本，则** Visual Basic **将该文本放置在文本框中插入点处：**Text1.SelText = Clipboard.GetText()**该代码假定全部文本被传送到或传送出文本框** Text1，**而用户可在** Text1 **和其它窗体上的控件之间进行复制、剪切和粘贴。由于** Clipboard **是被整个环境所共享的，所以在** Text1 **和任何正在使用剪贴板的应用程序之间，也能传送文本。**

运行程序后，在文本框中输入字符串“你好！”，然后就可以实现“复制” 和“剪切”和“粘贴”功能。如图 2-8：

图 2-8　**运行结果**

2.5　本章小结

本章我们学会了怎样控制文本的具体可视化特征，小到一个字符，大到一篇文章，我们都可以通过上面所介绍的方法去控制它的输出格式以及它的字体，大小等等的可视化信息。对一些重要的属性及方法读者应该牢记，这样在以后处理文本的程序当中，可以信手拈来，会使编程难度下降不少，同样，也可以使你的程序更加丰富多彩。