第 6 章　Windows 高级界面设计

6.1　高级窗体应用

任务 1　透明窗体

1．运行界面

运行界面如图 6-1 所示。

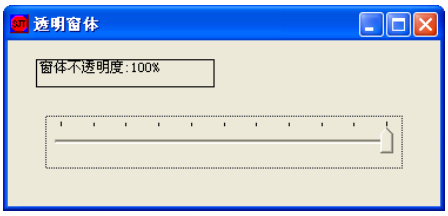

图 6-1　运行界面

2．功能

（1）通过调整 TrackBar 控件的滚动块以改变窗体的透明度。

（2）程序启动时窗体的透明度发生渐变。

知识点 1　Control 类

定义控件的基类，控件是带有可视化表示形式的组件。

Control 类实现向用户显示信息的类所需的最基本功能，它处理用户通过键盘和指针设备所进行的输入，还处理消息路由和安全。虽然它并不实现绘制，但定义控件的边界（其位置和大小），同时还提供窗口句柄（hWnd）。

1．主要属性（如表 6-1 所示）

表 6-1　Control 类主要属性说明

名称	说明
AllowDrop	获取或设置一个值，该值指示控件是否可以接受用户拖放到它上面的数据
BackColor	获取或设置控件的背景色
BackgroundImage	获取或设置在控件中显示的背景图像
Bottom	获取控件下边缘与其容器的工作区上边缘之间的距离（以像素为单位）
Bounds	获取或设置控件（包括其非工作区元素）相对于其父控件的大小和位置（以像素为单位）
CanFocus	获取一个值，该值指示控件是否可以接收焦点
CanSelect	获取一个值，该值指示是否可以选中控件
Capture	获取或设置一个值，该值指示控件是否已捕获鼠标
ClientRectangle	获取表示控件的工作区的矩形

续表

名称	说明
ClientSize	获取或设置控件的工作区的高度和宽度
ContextMenu	获取或设置与控件关联的快捷菜单
ContextMenuStrip	获取或设置与此控件关联的ContextMenuStrip
Cursor	获取或设置当鼠标指针位于控件上时显示的光标
DefaultBackColor	获取控件的默认背景色
DefaultFont	获取控件的默认字体
DefaultForeColor	获取控件的默认前景色
DisplayRectangle	获取表示控件的显示区域的矩形
Dock	获取或设置哪些控件边框停靠到其父控件并确定控件如何随其父级一起调整大小
Enabled	获取或设置一个值，该值指示控件是否可以对用户交互作出响应
Focused	获取一个值，该值指示控件是否有输入焦点
Font	获取或设置控件显示的文字的字体
ForeColor	获取或设置控件的前景色
Handle	获取控件绑定到的窗口句柄
HasChildren	获取一个值，该值指示控件是否包含一个或多个子控件
Height	获取或设置控件的高度
Left	获取或设置控件左边缘与其容器的工作区左边缘之间的距离（以像素为单位）
Location	获取或设置该控件的左上角相对于其容器的左上角的坐标
Margin	获取或设置控件之间的空间
MouseButtons	获取一个值，该值指示哪一个鼠标按钮处于按下的状态
MousePosition	获取鼠标光标的位置（以屏幕坐标表示）
Name	获取或设置控件的名称
Right	获取控件右边缘与其容器的工作区左边缘之间的距离（以像素为单位）
Size	获取或设置控件的高度和宽度
TabIndex	获取或设置在控件的容器的控件的 Tab 键顺序
TabStop	获取或设置一个值，该值指示用户能否使用 Tab 键将焦点放到该控件上
Text	获取或设置与此控件关联的文本
Top	获取或设置控件上边缘与其容器的工作区上边缘之间的距离（以像素为单位）
Visible	获取或设置一个值，该值指示是否显示该控件
Width	获取或设置控件的宽度

2．主要方法（如表 6-2 所示）

表 6-2　Control 类主要方法说明

名称	说明
CreateGraphics	为控件创建 Graphics
Dispose	已重载。释放由 Control 使用的所有资源
Focus	为控件设置输入焦点

续表

名称	说明
Hide	对用户隐藏控件
Invalidate	已重载。使控件的特定区域无效并向控件发送绘制消息
RectangleToClient	计算指定屏幕矩形的大小和位置（以工作区坐标表示）
RectangleToScreen	计算指定工作区矩形的大小和位置（以屏幕坐标表示）
Refresh	强制控件使其工作区无效并立即重绘自己和任何子控件
Show	向用户显示控件
Update	使控件重绘其工作区内的无效区域

3．主要事件（如表 6-3 所示）

表 6-3　Control 类主要事件说明

名称	说明
Click	在单击控件时发生
DoubleClick	在双击控件时发生
DragDrop	在完成拖放操作时发生
DragEnter	在将对象拖入控件的边界时发生
DragLeave	在将对象拖出控件的边界时发生
DragOver	在将对象拖到控件的边界上发生
Enter	进入控件时发生
GiveFeedback	在执行拖动操作期间发生
GotFocus	在控件接收焦点时发生
Invalidated	在控件的显示需要重绘时发生
KeyDown	在控件有焦点的情况下按下键时发生
KeyPress	在控件有焦点的情况下按下键时发生
KeyUp	在控件有焦点的情况下释放键时发生
Leave	在输入焦点离开控件时发生
LostFocus	当控件失去焦点时发生
MouseClick	在鼠标单击该控件时发生
MouseDoubleClick	当用鼠标双击控件时发生
MouseDown	当鼠标指针位于控件上并按下鼠标键时发生
MouseEnter	在鼠标指针进入控件时发生
MouseHover	在鼠标指针停放在控件上时发生
MouseLeave	在鼠标指针离开控件时发生
MouseMove	在鼠标指针移到控件上时发生
MouseUp	在鼠标指针在控件上并释放鼠标键时发生
MouseWheel	在移动鼠标轮并且控件有焦点时发生
Paint	在重绘控件时发生
PreviewKeyDown	在焦点位于此控件上的情况下，当有按键动作时发生（在 KeyDown 事件之前发生）
Resize	在调整控件大小时发生

知识点 2　Form 类

Form 类表示组成应用程序的用户界面的窗口或对话框。

Form 是应用程序中所显示的任何窗口的表示形式。Form 类可用于创建标准窗口、工具窗口、无边框窗口和浮动窗口，还可用于创建模式窗口，如对话框。一种特殊类型的窗体，即多文档界面（MDI）窗体可包含其他称为 MDI 子窗体的窗体。通过将 IsMdiContainer 属性设置为 true 来创建 MDI 窗体，将 MdiParent 属性设置为将包含 MDI 子窗体的 MDI 父窗体来创建 MDI 子窗体。

使用 Form 类中可用的属性，可以确定所创建窗口或对话框的外观、大小、颜色和窗口管理功能。Text 属性允许在标题栏中指定窗口的标题。Size 和 DesktopLocation 属性允许定义窗口在显示时的大小和位置。可以使用 ForeColor 颜色属性更改窗体上放置的所有控件的默认前景色。FormBorderStyle、MinimizeBox 和 MaximizeBox 属性允许控制运行时窗体是否可以最小化、最大化或调整窗体大小。

除了属性之外，还可以使用此类的方法来操作窗体。例如，可以使用 ShowDialog 方法将窗体显示为模式对话框，可以使用 SetDesktopLocation 方法在桌面上定位窗体。

1．Form.Opacity 属性

获取或设置窗体的不透明度级别。

语法如下：

Visual Basic（声明）	Visual Basic（用法）
Public Property Opacity As Double	Dim instance As Form Dim value As Double value = instance.Opacity instance.Opacity = value

该属性可以指定窗体及其控件的透明度级别。将此属性设置为小于 100%（1.00）的值时，会使整个窗体（包括边框）更透明。将此属性设置为值 0%（0.00）时，会使窗体完全不可见。可以使用此属性提供不同级别的透明度，或者提供如窗体逐渐进入或退出视野这样的效果。例如，可以通过将 Opacity 属性设置为值 0%（0.00），并逐渐增加该值直到它达到 100%（1.00），使一个窗体逐渐进入视野。

2．Form.WindowState 属性

获取或设置窗体的窗口状态。

属性值：FormWindowState，表示窗体的窗口状态。默认为 FormWindowState.Normal。

FormWindowState 枚举：指定窗体窗口如何显示。此枚举由 Form 类使用，它表示窗体的不同状态。默认状态为 Normal。

成员名称	说明
Maximized	最大化的窗口
Minimized	最小化的窗口
Normal	默认大小的窗口

3．Form.TopMost 属性

获取或设置一个值，指示该窗体是否应显示为最顶层窗体。最顶层窗体是重叠所有其他窗体（非最顶层窗体）的窗体，即使该窗体不是活动窗体或前台窗体。最顶层窗体始终显示在桌面上

Z 顺序窗口的最高点。可以使用此属性创建在应用程序中始终显示的窗体，如“查找和替换”工具窗口。

属性值：如果将窗体显示为最顶层窗体，则为 true，否则为 false。默认为 false。

4．Form.StartPosition 属性

获取或设置运行时窗体的起始位置。

属性值：FormStartPosition，表示窗体的起始位置。

FormStartPosition 枚举：指定窗体的初始位置。

成员名称	说明
CenterParent	窗体在其父窗体中居中
CenterScreen	窗体在当前显示窗口中居中，其尺寸在窗体大小中指定
Manual	窗体的位置由 Location 属性确定
WindowsDefaultBounds	窗体定位在 Windows 默认位置，其边界也由 Windows 默认决定
WindowsDefaultLocation	窗体定位在 Windows 默认位置，其尺寸在窗体大小中指定

5．Form.Size 属性

获取或设置窗体的大小。该属性允许同时设置窗体的高度和宽度（以像素为单位），而不是分别设置 Height 和 Width 属性。

属性值：Size，表示窗体的大小。

6．实例

创建 Form 实例，并调用 ShowDialog 方法以将该窗体显示为对话框。

该示例设置 FormBorderStyle、AcceptButton、CancelButton、MinimizeBox、MaximizeBox 和 StartPosition 属性，将窗体的外观和功能更改为对话框形式。该示例还使用窗体的 Controls 集合的 Add 方法添加 2 个 Button 控件。该示例使用 HelpButton 属性在对话框的标题栏中显示“帮助”按钮。

```
Public Sub CreateMyForm()
    Dim form1 As New Form()
    Dim button1 As New Button()
    Dim button2 As New Button()

    button1.Text = "确定"
    button1.Location = New Point(10, 10)
    button2.Text = "取消"
    button2.Location = _
        New Point(button1.Left, button1.Height + button1.Top + 10)
    form1.Text = "自定义对话框"
    form1.HelpButton = True

    form1.FormBorderStyle = FormBorderStyle.FixedDialog
    form1.MaximizeBox = False
    form1.MinimizeBox = False
    form1.AcceptButton = button1
    form1.CancelButton = button2
    form1.StartPosition = FormStartPosition.CenterScreen

    form1.Controls.Add(button1)
    form1.Controls.Add(button2)

    form1.ShowDialog()
```

```
End Sub
```

知识点 3　TrackBar 控件

Windows 窗体 TrackBar 控件用于在大量信息中进行浏览，或用于以可视的形式调整数字设置。TrackBar 控件有两部分：滚动块（又称为滑块）和刻度线。滚动块是可以调整的部分，其位置与 Value 属性相对应。刻度线是按规则间隔分隔的可视化指示符。跟踪条按指定的增量移动并且可以水平或垂直排列。使用跟踪条的一个示例是设置光标闪烁频率或鼠标速度。使用该控件，用户可以通过直观地调节数字设置来浏览信息。

TrackBar 控件的关键属性为：Value、TickFrequency、Minimum 以及 Maximum。TickFrequency 为刻度间隔，Minimum 和 Maximum 为跟踪条上能表示的最大值和最小值，如表 6-4 所示。

表 6-4　TrackBar 控件关键属性

名称	说明
Value	获取或设置表示跟踪条上滚动框的当前位置的数值。 属性值：处于 Minimum 和 Maximum 范围内的数值。默认值为 0
TickFrequency	获取或设置一个值，该值指定控件上绘制的刻度之间的增量。 属性值：表示刻度之间的增量的数值。默认值为 1
Minimum	获取或设置此 TrackBar 使用的范围的下限。 属性值：TrackBar 的最小值。默认值为 0
Maximum	获取或设置此 TrackBar 使用的范围的上限。 属性值：TrackBar 的最大值。默认值为 10
SmallChange	获取或设置当滚动框短距离移动时对 Value 属性进行增减的值。 属性值：一个数值。默认值为 1
LargeChange	获取或设置一个值，当滚动框长距离移动时向 Value 属性加上该值或从中减去该值。 属性值：一个数值。默认值为 5

其他两个重要的属性是 SmallChange 和 LargeChange。SmallChange 属性值是滚动块响应按下向左键或向右键时移动的位置数，LargeChange 属性值是滚动块响应按下 PageUp 或 PageDown 键，或者响应鼠标在跟踪条上的滚动块任一边单击时所移动的位置数。

实例：下面的代码示例显示了一个包含 TrackBar 控件和 TextBox 控件的窗体。该示例演示如何设置 Maximum、TickFrequency、LargeChange 和 SmallChange 属性以及如何处理 Scroll 事件。Scroll 事件发生时，TextBox 内容被更新为 Value 属性值。

```
Imports System
Imports System.Drawing
Imports System.Windows.Forms

Public Class Form1
    Inherits System.Windows.Forms.Form

    Private WithEvents trackBar1 As System.Windows.Forms.TrackBar
    Private textBox1 As System.Windows.Forms.TextBox

    <System.STAThread()> _
    Public Shared Sub Main()
        System.Windows.Forms.Application.Run(New Form1)
    End Sub
```

```
    Public Sub New()
        Me.textBox1 = New System.Windows.Forms.TextBox
        Me.trackBar1 = New System.Windows.Forms.TrackBar
        Me.textBox1.Location = New System.Drawing.Point(240, 16)
        Me.textBox1.Size = New System.Drawing.Size(48, 20)
        Me.ClientSize = New System.Drawing.Size(296, 62)
        Me.Controls.AddRange(New System.Windows.Forms.Control() {Me.textBox1, Me.trackBar1})
        Me.Text = "TrackBar 实例"
        Me.trackBar1.Location = New System.Drawing.Point(8, 8)
        Me.trackBar1.Size = New System.Drawing.Size(224, 45)
        trackBar1.Maximum = 30
        trackBar1.TickFrequency = 5
        trackBar1.LargeChange = 3
        trackBar1.SmallChange = 2
    End Sub

    Private Sub trackBar1_Scroll(ByVal sender As Object, _
                    ByVal e As System.EventArgs) Handles trackBar1.Scroll
        '在文本框中显示 TrackBar 的值
        textBox1.Text = trackBar1.Value
    End Sub

End Class
```

知识点 4　Timer 控件

Windows 窗体 Timer 是定期引发事件的控件，该控件是为 Windows 窗体环境设计的。

Timer 控件具有一个 Interval 属性，该属性指定一个计时器事件与下一个计时器事件之间间隔的毫秒数。除非该控件被禁用，否则计时器会以大致相等的时间间隔继续接收 Tick 事件。

当编写 Timer 控件时，需要考虑 Interval 属性的几点限制：

- 如果应用程序或另一个应用程序对系统需求很大（如长循环、大量的计算或驱动程序、网络或端口访问），那么应用程序可能无法以 Interval 属性指定的频率来获取计时器事件。
- 间隔可以在 1～64767 之间（包括 1 和 64767），这意味着即使最长的间隔（大约 64.8 秒）也不会超过一分钟很多。
- 不能保证间隔所精确经过的时间。若要确保精确，计时器应根据需要检查系统时钟，而不是尝试在内部跟踪所积累的时间。
- 系统每秒生成 18 个时钟刻度，因此即使 Interval 属性以毫秒为单位，间隔的实际精度也不会超过 1/18 秒。

使用计时器组件以设置的间隔运行过程的步骤如下：

（1）在窗体中添加 Timer。

（2）为计时器设置 Interval 属性（以毫秒为单位）。该属性决定在再次运行该过程之前所经过的时间。（注意：计时器事件发生越频繁，用于响应该事件的处理器时间就越长。这会降低整体性能。请勿将间隔设置得比所需值小）。

（3）在 Tick 事件处理程序内编写合适的代码。在该事件中编写的代码将以 Interval 属性中所指定的间隔运行。

（4）将 Enabled 属性设置为 true，以启动计时器。Tick 事件将开始发生，从而以设置的间隔运行过程。

（5）需要的时候，可将 Enabled 属性设置为 false，以使过程停止再次运行（注意：将间隔

设置为 0，并不会导致计时器停止）。

任务 1 解析

1．控件的属性（如表 6-5 所示）

表 6-5　控件属性说明

控件	Name 属性	Text 属性	其他属性
Form	Form1	透明窗体	
Timer	Timer1		Invertal=20
Label	Label1	窗体不透明度 100%	
TrackBar	TrackBar1		

2．源代码

```
Public Class Form1
    Inherits System.Windows.Forms.Form
    Dim temp As Integer
    Dim flag As Boolean

    Private Sub TrackBar1_Scroll(ByVal sender As Object, ByVal e As System.EventArgs) Handles TrackBar1.Scroll
        Me.Opacity = TrackBar1.Value / 100
        Label1.Text = "窗体不透明度：" & CStr(Me.Opacity * 100) & "%"
    End Sub

    Private Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        If flag = False Then
            temp = temp + 1
            Me.Opacity = temp / 100
            If Me.Opacity >= 1 Then
                Timer1.Enabled = False
                flag = True
            End If
        Else
            temp = temp - 1
            Me.Opacity = temp / 100
            If Me.Opacity <= 0 Then
                Timer1.Enabled = False
                flag = False
            End If
        End If
    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Timer1.Enabled = True
    End Sub

    Private Sub Form1_Closing(ByVal sender As Object, ByVal e As System.ComponentModel.CancelEventArgs) Handles
MyBase.Closing
        Timer1.Enabled = True
        If MsgBox("真的关闭窗体吗？", MsgBoxStyle.OkCancel) = MsgBoxResult.Ok Then
            e.Cancel = False
        Else
            Timer1.Enabled = False
            Me.Opacity = 1
            temp = 100
            flag = True
```

```
            e.Cancel = True
        End If
End Sub

End Class
```

6.2 对话框应用

任务 2 通用对话框

1．运行界面

运行界面如图 6-2 所示。

图 6-2 运行界面

2．功能

（1）通过 OpenFileDialog 组件打开文本文件。
（2）通过 ColorDialog 组件设置文本颜色。
（3）通过 FontDialog 组件设置文本字体。
（4）通过 SaveFileDialog 组件另存文本。

知识点 1 OpenFileDialog 控件

Windows 窗体 OpenFileDialog 控件是一个预先配置的对话框。它与 Windows 操作系统所公开的“打开文件”对话框相同，该控件从 CommonDialog 类继承。

在基于 Windows 的应用程序中可将该组件用作简单的文件选择解决方案，而不用配置自己的对话框。利用标准的 Windows 对话框，可以创建其基本功能可立即为用户所熟悉的应用程序。但是应注意，使用 OpenFileDialog 控件时，必须编写自己的文件打开逻辑。

可使用 ShowDialog 方法在运行时显示该对话框。使用 Multiselect 属性可使用户选择多个要打开的文件。另外，可使用 ShowReadOnly 属性确定在对话框中是否出现只读复选框；ReadOnlyChecked 属性指示是否选中只读复选框；Filter 属性设置当前文件名筛选字符串，该字符串确定出现在对话框的“文件类型”框中的选择。

将 OpenFileDialog 控件添加到窗体后，它出现在 Windows 窗体设计器底部的栏中 OpenFileDialog 控件有如下主要属性：

（1）FileName 属性：获取或设置一个包含在“文件”对话框中选定的文件名的字符串。文件名既包含文件路径也包含扩展名。如果未选定文件，该属性将返回空字符串（""）。

（2）CheckFileExists 属性：获取或设置一个值，该值指示如果用户指定不存在的文件名，

对话框是否显示警告。

（3）CheckPathExists 属性：获取或设置一个值，该值指示如果用户指定不存在的路径，对话框是否显示警告。

（4）DefaultExt 属性：获取或设置默认文件扩展名。

（5）Filter 属性：获取或设置当前文件名筛选器字符串，该字符串决定对话框的“另存为文件类型”或“文件类型”框中出现的选择内容。

（6）FilterIndex 属性：获取或设置“文件”对话框中当前选定筛选器的索引。

（7）InitialDirectory 属性：获取或设置“文件”对话框显示的初始目录。

（8）Title 属性：获取或设置“文件”对话框标题。

实例：使用 OpenFileDialog 控件以文件方式打开文件。

使用 ShowDialog 方法显示对话框，并使用 OpenFile 方法打开文件。

在下面的实例中，将实例化一个具有 cursor 筛选器的 OpenFileDialog 控件，使用户只能选择具有.cur 文件扩展名的文件。如果选择了一个.cur 文件，该窗体的光标将设置为选定的光标。

```
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
    '创建选择光标文件的 OpenFileDialog
    Dim openFileDialog1 As New OpenFileDialog()
    openFileDialog1.Filter = "光标文件|*.cur"
    openFileDialog1.Title = "选择光标文件"

    '显示选择光标文件的 OpenFileDialog
    If openFileDialog1.ShowDialog() = DialogResult.OK Then
       '将窗体的光标设置为选定的光标
       Me.Cursor = New Cursor(openFileDialog1.OpenFile())
    End If
End Sub
```

知识点 2　SaveFileDialog 控件

Windows 窗体 SaveFileDialog 控件是一个预先配置的对话框。它与 Windows 使用的标准“保存文件”对话框相同，该组件继承自 CommonDialog 类。

使用该控件作为一个简单的解决方案，使用户能够保存文件，而不用配置您自己的对话框。利用标准的 Windows 对话框，创建基本功能可立即为用户所熟悉的应用程序。但应注意，使用 SaveFileDialog 控件时，必须编写自己的文件保存逻辑。

可使用 ShowDialog 方法在运行时显示该对话框。使用 OpenFile 方法可以读写方式打开文件。

将 SaveFileDialog 控件添加到窗体后，它出现在 Windows 窗体设计器底部的栏中。

知识点 3　ColorDialog 控件

Windows 窗体 ColorDialog 控件是一个预先配置的对话框，它允许用户从调色板选择颜色以及将自定义颜色添加到该调色板。此对话框与在其他基于 Windows 的应用程序中看到的用于选择颜色的对话框相同。可在基于 Windows 的应用程序中使用它作为简单的解决方案，而不用配置自己的对话框。

此对话框中选择的颜色在 Color 属性中返回。如果 AllowFullOpen 属性设置为 false，则将禁用“定义自定义颜色”按钮，并且用户只能使用调色板中的预定义颜色。如果 SolidColorOnly 属性设置为 true，则用户无法选择抖色。若要显示此对话框，必须调用它的 ShowDialog 方法。

ColorDialog 控件有如下主要属性：

（1）AllowFullOpen 属性：获取或设置一个值，该值指示用户是否可以使用该对话框定义自定义颜色。

（2）AnyColor 属性：获取或设置一个值，该值指示对话框是否显示基本颜色集中可用的所有颜色。

（3）Color 属性：获取或设置用户选定的颜色。

（4）CustomColors 属性：获取或设置对话框中显示的自定义颜色集。

（5）FullOpen 属性：获取或设置一个值，该值指示用于创建自定义颜色的控件在对话框打开时是否可见。

（6）SolidColorOnly 属性：获取或设置一个值，该值指示对话框是否限制用户只选择纯色。

实例：使用 ColorDialog 控件选择颜色。

（1）使用 ShowDialog 方法显示对话框。

（2）使用 DialogResult 属性确定如何关闭对话框。

（3）使用 ColorDialog 控件的 Color 属性设置选定的颜色。

在下面的实例中，Button 控件的 Click 事件处理程序打开一个 ColorDialog 控件。当用户选定颜色并单击“确定”按钮后，Button 控件的背景色将设置为选定的颜色。

```
Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button1.Click
    If ColorDialog1.ShowDialog() = DialogResult.OK Then
        Button1.BackColor = ColorDialog1.Color
    End If
End Sub
```

知识点 4　FontDialog 控件

Windows 窗体 FontDialog 控件是一个预先配置的对话框，该对话框是标准的 Windows“字体”对话框，用于公开系统上当前安装的字体。可在基于 Windows 的应用程序中将其用作简单的字体选择解决方案，而不是配置您自己的对话框。

默认情况下，该对话框显示字体、字体样式和字体大小的列表框；删除线和下划线等效果的复选框；脚本的下拉列表以及字体外观的示例（脚本是指给定字体可用的不同字符脚本，如希伯来语或日语）。若要显示“字体”对话框，请调用 ShowDialog 方法。

用户可以使用 FontDialog 控件选择字体，并可以更改字体显示方式，例如粗细和大小。该对话框中选定的字体在 Font 属性中返回。

FontDialog 控件有如下主要属性：

（1）AllowScriptChange属性：获取或设置一个值，该值指示用户能否更改“脚本”组合框中指定的字符集，以显示除了当前所显示字符集以外的字符集。

（2）Color属性：获取或设置选定字体的颜色。

（3）Font属性：获取或设置选定的字体。

（4）MaxSize属性：获取或设置用户可选择的最大磅值。

（5）MinSize属性：获取或设置用户可选择的最小磅值。

（6）ShowColor属性：获取或设置一个值，该值指示对话框是否显示颜色选择。

（7）ShowEffects属性：获取或设置一个值，该值指示对话框是否包含允许用户指定删除线、下划线和文本颜色选项的控件。

实例：使用 FontDialog 控件选择字体属性.

（1）使用 ShowDialog 方法显示对话框。

（2）使用 DialogResult 属性确定如何关闭对话框。

（3）使用 Font 属性设置所需的字体。

在下面的实例中，Button 控件的 Click 事件处理程序打开一个 FontDialog 控件。当用户选定字体并单击“确定”按钮时，窗体上的 TextBox 控件的 Font 属性被设置为选定的字体。本实例假定窗体上有 1 个 Button 控件、1 个 TextBox 控件和 1 个 FontDialog 控件。

```
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
    If FontDialog1.ShowDialog() = DialogResult.OK Then
        TextBox1.Font = FontDialog1.Font
    End If
End Sub
```

任务 2 解析

1．控件的属性（如表 6-6 所示）

表 6-6　控件属性说明

控件	Name 属性	Text 属性	其他属性
Form	Form1	通用对话框	
OpenFileDialog	OpenFileDialog1		
TextBox	TxtNote		Multiline=True
Button	BtnOpen	打开	
	BtnSav	保存	
	BtnClr	颜色	
	BtnFnt	字体	
	BtnExt	结束	

2．源代码

```
Imports System.IO
Imports System.Drawing.Printing

Public Class Form1
    Inherits System.Windows.Forms.Form

    Private Sub BtnOpen_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnOpen.Click
        Dim opdg As New OpenFileDialog()
        opdg.Filter = "文本文件(*.txt)|*.txt"
        Dim sr As StreamReader
        TxtNote.Text = ""
        If opdg.ShowDialog = DialogResult.OK Then
            Try
                sr = File.OpenText(opdg.FileName)
                Dim x As String
                While sr.Peek <> -1
                    x = sr.ReadLine()
                    TxtNote.Text += x & vbCrLf
                End While
                sr.Close()
            Catch ex As FileNotFoundException
                MsgBox(opdg.FileName & " 没发现！")
            End Try
```

```
        End If
    End Sub

    Private Sub BtnSav_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnSav.Click
        Dim svdg As New SaveFileDialog()
        svdg.Filter = "文本文件(*.txt)|*.txt"
        Dim sr As StreamWriter
        If svdg.ShowDialog = DialogResult.OK Then
            Try
                sr = File.CreateText(svdg.FileName)
                Dim ch As Char
                For Each ch In TxtNote.Text
                    sr.Write(ch)
                Next
                sr.Close()
            Catch ex As Exception
                MsgBox(ex.Message)
            End Try
        End If
    End Sub

    Private Sub BtnClr_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnClr.Click
        Dim clrdg As New ColorDialog()
        If clrdg.ShowDialog = DialogResult.OK Then
            TxtNote.ForeColor = clrdg.Color
        End If
    End Sub

    Private Sub BtnFnt_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnFnt.Click
        Dim fntdg As New FontDialog()
        If fntdg.ShowDialog = DialogResult.OK Then
            TxtNote.Font = fntdg.Font
        End If
    End Sub

    Private Sub BtnExt_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnExt.Click
        Close()
    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        TxtNote.Text = "在此处输入文字....."
    End Sub
End Class
```

6.3 菜单应用

任务 3 写字板

1．运行界面

运行界面如图 6-3 所示。

2．功能

（1）实现新建、打开、保存、另存为功能。

（2）实现剪切、复制、粘贴、全选操作功能。

（3）实现快捷菜单功能。

（4）设置背景颜色和字体。

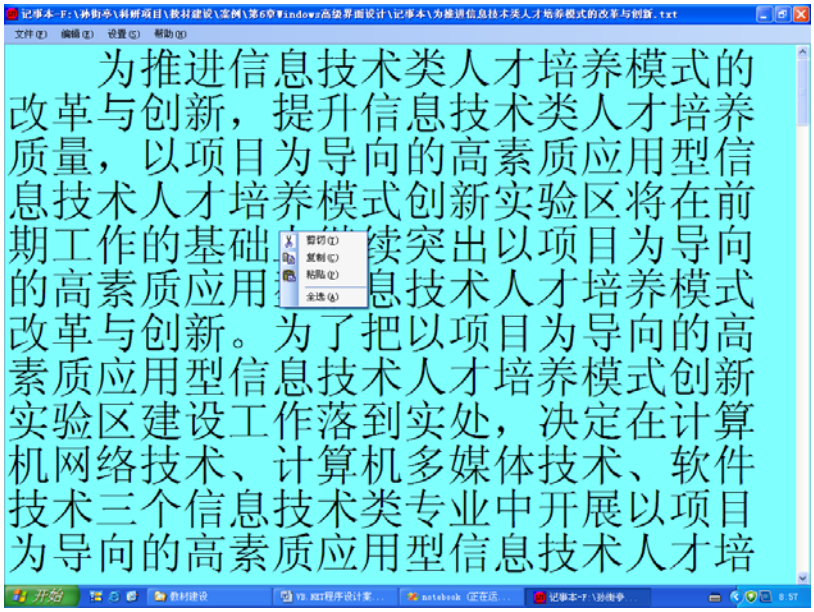

图 6-3　运行界面

知识点 1　MainMenu 控件

Windows 窗体 MainMenu 控件在运行时显示一个菜单。主菜单的所有子菜单和单个项均为 MenuItem 对象。

通过将 DefaultItem 属性设置为 true，可以将某菜单项指定为默认项。单击菜单时，默认项以粗体文本显示。菜单项的 Checked 属性为 true 或 false，它指示是否选定了该菜单项。菜单项的 RadioCheck 属性会自定义选定项的外观：如果 RadioCheck 设置为 true，则该项旁边出现一个单选按钮；如果 RadioCheck 设置为 false，则该项旁边出现一个选中标记。

MenuItem 类：表示在 MainMenu 或 ContextMenu 内显示的单个项。为了显示 MenuItem，必须将其添加到 MainMenu 或 ContextMenu。若要创建子菜单，可以将 MenuItem 对象添加到父 MenuItem 的 MenuItems 属性。

MenuItem 类提供使您得以配置菜单项的外观和功能的属性。若要显示菜单项旁边的选中标记，可使用 Checked 属性，使用该功能来标识在互斥的菜单项列表中选择的菜单项。例如，如果有一组用于在 TextBox 控件中设置文本颜色的菜单项，则可以使用 Checked 属性来标识当前选定的颜色。Shortcut 属性可用于定义键盘组合（可按下该键盘组合来选择菜单项）。

对于在多文档界面（MDI）应用程序中显示的 MenuItem 对象，可使用 MergeMenu 方法将 MDI 父级菜单与其子窗体菜单合并以创建合并的菜单结构。因为无法同时在多个位置重用 MenuItem（如在 MainMenu 和 ContextMenu 中），所以可以使用 CloneMenu 方法创建可用于其他位置的 MenuItem 的副本。

Popup 事件使您得以在显示菜单前执行任务。例如，可以基于代码状态为该事件创建一个事件处理程序以显示或隐藏菜单项。通过 Select 事件可以执行任务，如当用户将鼠标指针放在菜单项上时，为应用程序的菜单项提供详细的帮助。

1．主要属性（如表 6-7 所示）

表 6-7　MainMenu 组件主要属性说明

名称	说明
Checked	获取或设置一个值，通过该值指示选中标记是否出现在菜单项文本的旁边
DefaultItem	获取或设置一个值，通过该值指示菜单项是否为默认菜单项
Enabled	获取或设置一个值，通过该值指示菜单项是否启用
Index	获取或设置一个值，通过该值指示菜单项在其父菜单中的位置
IsParent	已重写。获取一个值，通过该值指示菜单项是否包含子菜单项
MdiList	获取或设置一个值，通过该值指示是否使用在关联窗体内显示的多文档界面（MDI）子窗口列表来填充菜单项
MdiListItem	获取一个值，通过该值指示用于显示多文档界面（MDI）子窗体列表的 MenuItem（从Menu继承。
MenuItems	获取一个值，通过该值指示与菜单关联的 MenuItem 对象的集合（从Menu 继承）
Name	获取或设置Menu的名称（从Menu 继承）
RadioCheck	获取或设置一个值，通过该值指示 MenuItem（如果已选中）是否显示单选按钮而不是选中标记
Shortcut	获取或设置一个值，通过该值指示与菜单项关联的快捷键
ShowShortcut	获取或设置一个值，通过该值指示与菜单项关联的快捷键是否在菜单项标题的旁边显示
Text	获取或设置一个值，通过该值指示菜单项标题
Visible	获取或设置一个值，通过该值指示菜单项是否可见

2．主要事件（如表 6-8 所示）

表 6-8　MainMenu 组件主要事件说明

名称	说明
Click	当单击菜单项或使用为该菜单项定义的快捷键或访问键选择菜单项时发生。如果用户使用键盘并按 Enter 键选择菜单项，此事件也会发生
Popup	在显示菜单项的菜单项列表之前发生。该事件仅在菜单项有要显示的子菜单项时发生。在显示各菜单项之前，可基于应用程序的状态使用该事件处理程序添加、移除、启用、禁用、选中或取消选中菜单项
Select	当用户将指针放在菜单项上时发生。当用户将鼠标指针放在菜单项上时，通常会引发此事件。当用户使用键盘上的箭头键滚动到菜单项上来突出显示该菜单项时，也可引发该事件。可使用该事件在应用程序的状态栏中显示有关该菜单项的详细帮助字符串

实例 1：Click 事件的使用。

下面的实例演示了如何使用 Click 事件在单击 MenuItem 时执行任务。此实例将创建一个名为 mainMenu1 的 MainMenu，并添加两个 MenuItem 对象：topMenuItem（File）和 menuItem1（Open），然后它将 Click 事件连接到 menuItem1_Click 事件处理程序。当用户单击 Open 菜单项时，就会初始化并显示一个 OpenFileDialog。此实例要求已创建了一个名为 Form1 的Form。

```
Public Sub CreateMyMenu()
    '创建主菜单对象
    Dim mainMenu1 As New MainMenu()
    '创建菜单项对象
    Dim topMenuItem As New MenuItem()
```

```
    Dim menuItem1 As New MenuItem()
    '设置菜单项显示文字
    topMenuItem.Text = "&File"
    menuItem1.Text = "&Open"
    '把菜单项加到主菜单中
    topMenuItem.MenuItems.Add(menuItem1)
    mainMenu1.MenuItems.Add(topMenuItem)
    '把 menuItem1.Click 和 Me.menuItem1_Click 关联
    AddHandler menuItem1.Click, AddressOf Me.menuItem1_Click
    '把 mainMenu1 和主窗体关联
    Me.Menu = mainMenu1
End Sub

Private Sub menuItem1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    '创建并显示 OpenFileDialog
    Dim fd As New OpenFileDialog()
    fd.DefaultExt = "*.*"
    fd.ShowDialog()
End Sub
```

实例 2：Popup 事件的使用。

下面的实例演示如何使用 Popup 事件，来确定是否在那些为剪切、复制和删除操作提供支持的 MenuItem 对象所在的菜单显示之前就启用了这些对象。此实例确定在启用 MenuItem 对象之前，textBox1（窗体上的 TextBox 控件）是否已被启用、是否具有输入焦点以及是否选定了文本。此实例要求已创建 3 个名称分别为 menuCut、menuCopy 和 menuDelete 的 MenuItem 对象。

```
Private Sub PopupMyMenu(ByVal sender As Object, ByVal e As System.EventArgs) Handles menuEdit.Popup
    If textBox1.Enabled = False OrElse textBox1.Focused = False OrElse textBox1.SelectedText.Length = 0 Then
        menuCut.Enabled = False
        menuCopy.Enabled = False
        menuDelete.Enabled = False
    Else
        menuCut.Enabled = True
        menuCopy.Enabled = True
        menuDelete.Enabled = True
    End If
End Sub
```

实例 3：MenuItem 类的 Select 事件的使用。

下面的实例演示了如何使用 MenuItem 类的 Select 事件向 StatusBar 控件的 StatusBarPanel 分配帮助文本。此实例要求名为 menuOpen、menuSave 和 menuExit 的 MenuItem 对象已添加到窗体的 MainMenu 控件中。此实例还要求已将名为 statusBar1 的 StatusBar 控件添加到窗体中。StatusBar 控件应包含 StatusBarPanel。

```
Private Sub MenuSelected(ByVal sender As Object, ByVal e As System.EventArgs) Handles menuOpen.Select,
menuExit.Select, menuSave.Select
    If sender Is menuOpen Then
        StatusBar1.Panels(0).Text = "打开文件"
    Else
        If sender Is menuSave Then
            StatusBar1.Panels(0).Text = "保存文件"
        Else
            If sender Is menuExit Then
                StatusBar1.Panels(0).Text = "退出应用程序"
            Else
                StatusBar1.Panels(0).Text = "就绪"
            End If
        End If
    End If
End Sub
```

知识点 2　MenuStrip 控件

MenuStrip 控件支持多文档界面（MDI）和菜单合并、工具提示和溢出。可以通过添加访问键、快捷键、选中标记、图像和分隔条，来增强菜单的可用性和可读性。MenuStrip 控件取代了 MainMenu 控件并向其中添加了功能；但也可以选择保留 MainMenu 控件以备向后兼容和将来使用。MenuStrip 类在.NET Framework 2.0 版中是新增的。

MenuStrip 控件表示窗体菜单结构的容器。可以将 ToolStripMenuItem 对象添加到表示菜单结构中各命令的 MenuStrip 中。每个 ToolStripMenuItem 可以成为应用程序的命令或其他子菜单项的父菜单。MenuStrip 是 ToolStripMenuItem、ToolStripComboBox、ToolStripSeparator 和 ToolStripTextBox 对象的容器。

使用 MenuStrip 控件可以：

（1）创建支持高级用户界面和布局功能的易自定义的常用菜单，例如文本和图像排序和对齐、拖放操作、MDI、溢出和访问命令的其他模式。

（2）支持操作系统的典型外观和行为。

（3）对所有容器和包含的项进行事件的一致性处理，处理方式与其他控件的事件相同。

主要属性如表 6-9 所示。

表 6-9　MenuStrip 控件主要属性说明

名称	说明
MdiWindowListItem	获取或设置用于显示 MDI 子窗体列表的 ToolStripMenuItem
System.Windows.Forms.ToolStripItem.Merge Action	获取或设置 MDI 应用程序中子菜单与父菜单合并的方式
System.Windows.Forms.ToolStripItem.Merge Index	获取或设置 MDI 应用程序的菜单中合并项的位置
System.Windows.Forms.Form.IsMdiContainer	获取或设置一个值，该值指示窗体是否为 MDI 子窗体的容器
ShowItemToolTips	获取或设置一个值，该值指示是否为 MenuStrip 显示工具提示
CanOverflow	获取或设置一个值，该值指示 MenuStrip 是否支持溢出功能
ShortcutKeys	获取或设置与 ToolStripMenuItem 关联的快捷键
ShowShortcutKeys	获取或设置一个值，该值指示与 ToolStripMenuItem 关联的快捷键是否显示在 ToolStripMenuItem 旁边

知识点 3　ContextMenuStrip 控件

ContextMenuStrip 控件提供了与某个控件关联的快捷菜单。快捷菜单（也称为上下文菜单）在用户右击时会出现在鼠标位置。快捷菜单在鼠标指针位置提供了工作区或控件的选项。ContextMenuStrip 控件旨在无缝地与新的 ToolStrip 和相关控件结合使用，但是也可以很容易地将 ContextMenuStrip 与其他控件关联。ContextMenuStrip 类在.NET Framework 2.0 版中是新增的。

ContextMenuStrip 类表示快捷菜单，这些快捷菜单在用户在窗体中的控件或特定区域上单击鼠标右键时显示。快捷菜单通常用于组合来自窗体的一个 MenuStrip的不同菜单项，便于用户在给定应用程序上下文中使用。例如，可以使用分配给 TextBox控件的快捷菜单提供菜单项，以便更改文本字体，在控件中查找文本或实现复制和粘贴文本的剪贴版功能。还可以在快捷菜单中显示不位于 MenuStrip 中的新的 ToolStripMenuItem 对象，从而提供与特定情况有关且不适合在 MenuStrip 中显示的命令。

当用户在控件或窗体本身上右击时，通常会显示快捷菜单。许多可视控件（以及 Form 本

身）都有一个 Control.ContextMenuStrip 属性，该属性将 ContextMenuStrip 类绑定到显示快捷菜单的控件。多个控件可使用一个 ContextMenuStrip。

将 ToolStripDropDownMenu.ShowCheckMargin 属性设置为 true 可向 ToolStripMenuItem 的左侧添加用以容纳选中标记的空间，选中标记显示是否启用或选择了该菜单项。ToolStripDropDownMenu.ShowImageMargin 属性默认被设置为 true。使用 ToolStripMenuItem 左侧的此空间可以为菜单项显示一个图像。

ContextMenuStrip 是 ToolStripMenuItem、ToolStripComboBox、ToolStripSeparator 和 ToolStrip TextBox 对象的容器。

知识点 4　ToolStripMenuItem 类

表示 MenuStrip 或 ContextMenuStrip 上显示的可选选项。为了显示 ToolStripMenuItem，必须将其添加到 MenuStrip 或 ContextMenuStrip。

ToolStripMenuItem 类提供使您得以配置菜单项的外观和功能的属性。若要显示菜单项旁边的选中标记，使用 Checked 属性。使用此功能可以标识在互斥的菜单项列表中选定的菜单项。例如，如果有一组用于在 TextBox 控件中设置文本颜色的菜单项，则可以使用 Checked 属性来标识当前选定的颜色。使用 ShortcutKeys 属性可以定义组合键（可以按该组合键来选择菜单项）。

主要属性如表 6-10 所示。

表 6-10　ToolStripMenuItem 类主要属性说明

名称	说明
Checked	获取或设置一个值，该值指示是否选中 ToolStripMenuItem
CheckOnClick	获取或设置一个值，该值指示 ToolStripMenuItem 是否应在被单击时自动显示为选中或未选中
CheckState	获取或设置一个值，该值指示 ToolStripMenuItem 处于选中、未选中还是不确定状态
ShortcutKeyDisplayString	获取或设置快捷键文本
ShortcutKeys	获取或设置与 ToolStripMenuItem 关联的快捷键
ShowShortcutKeys	获取或设置一个值，该值指示与 ToolStripMenuItem 关联的快捷键是否显示在 ToolStripMenuItem 的旁边

任务 3 解析

1．控件的属性（如表 6-11 所示）

表 6-11　控件属性说明

控件	Name 属性	Text 属性	其他属性
Form	FormNote	写字板	
MenuStrip	MenuStripNote		
ContextMenuStrip	ContextMenuStripNote		
TextBox	TextBoxNote		ContextMenuStrip=ContextMenuStripNote Multiline= true

续表

控件	Name 属性	Text 属性	其他属性
ToolStripMenuItem (MenuStrip)	ToolStripMenuItemFile	文件(&F)	
	ToolStripMenuItemFileNew	新建(&N)	
	ToolStripMenuItemFileOpen	打开(&O)...	
	ToolStripMenuItemFileSave	保存(&S)	
	ToolStripMenuItemFileSaveAs	另存为(&A)...	
	ToolStripMenuItemFileExit	退出(&X)	
	ToolStripMenuItemEdit	编辑(&E)	
	ToolStripMenuItemEditUndo	撤消(&U)	
	ToolStripMenuItemEditCut	剪切(&T)	
	ToolStripMenuItemEditCopy	复制(&C)	
	ToolStripMenuItemEditPaste	粘贴(&P)	
	ToolStripMenuItemEditSelectAll	全选(&A)	
	ToolStripMenuItemSetup	设置(&S)	
	ToolStripMenuItemSetupColor	颜色(&C)...	
	ToolStripMenuItemSetupFont	字体(&F)...	
	ToolStripMenuItemHelp	帮助(&H)	
	ToolStripMenuItemHelpAbout	关于写字板(&A)...	
ToolStripMenuItem (ContextMenuStrip)	ToolStripMenuItemCut	剪切(&T)	
	ToolStripMenuItemCopy	复制(&C)	
	ToolStripMenuItemPaste	粘贴(&P)	
	ToolStripMenuItemSelectAll	全选(&A)	

2．源代码

```
Imports System.IO

Public Class FormNote

    Private strFileName As String

    Private Sub FormNote_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Me.TextBoxNote.Width = Me.Width - 8
        Me.TextBoxNote.Height = Me.Height - Me.MenuStripNote.Height - 32
        Me.ToolStripMenuItemEditUndo.Enabled = False
        Me.ToolStripMenuItemEditCut.Enabled = False
        Me.ToolStripMenuItemEditCopy.Enabled = False
        Me.ToolStripMenuItemCut.Enabled = False
        Me.ToolStripMenuItemCopy.Enabled = False
        strFileName = "新文件.txt"
        Me.Text = "写字板-" + strFileName
    End Sub

    Private Sub ToolStripMenuItemFileExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ToolStripMenuItemFileExit.Click
        Me.Close()
```

```
        End Sub

        Private Sub ToolStripMenuItemEditCopy_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ToolStripMenuItemEditCopy.Click, ToolStripMenuItemCopy.Click
            If Me.TextBoxNote.SelectionLength > 0 Then
                '将文本框中的当前选定内容复制到“剪贴板”
                Me.TextBoxNote.Copy()
            End If
        End Sub

        Private Sub ToolStripMenuItemEditCut_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ToolStripMenuItemEditCut.Click, ToolStripMenuItemCut.Click
            If Me.TextBoxNote.SelectedText <> "" Then
                '将文本框中的当前选定内容移动到“剪贴板”
                Me.TextBoxNote.Cut()
            End If
        End Sub

        Private Sub ToolStripMenuItemEditPaste_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ToolStripMenuItemEditPaste.Click, ToolStripMenuItemPaste.Click
            If Clipboard.GetDataObject().GetDataPresent(DataFormats.Text) = True Then
                If Me.TextBoxNote.SelectionLength > 0 Then
                    If MessageBox.Show("是否覆盖当前选择的内容?", "提示", MessageBoxButtons.YesNo) =
System.Windows.Forms.DialogResult.No Then
                        '重新设置文本框中选定的文本起始点
                        Me.TextBoxNote.SelectionStart = Me.TextBoxNote.SelectionStart + Me.TextBoxNote.SelectionLength
                        Me.TextBoxNote.SelectionLength = 0
                    End If
                End If
                '用剪贴板的内容替换文本框中的当前选定内容
                Me.TextBoxNote.Paste()
            End If
        End Sub

        Private Sub ToolStripMenuItemEditSelectAll_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ToolStripMenuItemEditSelectAll.Click, ToolStripMenuItemSelectAll.Click
            If Me.TextBoxNote.SelectionLength = 0 Then
                '选定文本框中的所有文本
                Me.TextBoxNote.SelectAll()
            End If
            '将文本框中的当前选定内容复制到“剪贴板”
            Me.TextBoxNote.Copy()

            Me.ToolStripMenuItemEditCut.Enabled = True
            Me.ToolStripMenuItemEditCopy.Enabled = True
            Me.ToolStripMenuItemCut.Enabled = True
            Me.ToolStripMenuItemCopy.Enabled = True
        End Sub

        Private Sub ToolStripMenuItemFileNew_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ToolStripMenuItemFileNew.Click
            If Me.TextBoxNote.Modified Then
                Dim messageBoxResult As DialogResult
                messageBoxResult = MessageBox.Show("当前文档已被修改，是否保存？", "保存",
MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1)
                If messageBoxResult = Windows.Forms.DialogResult.Yes Then
                    Me.ToolStripMenuItemFileSaveAs.PerformClick()
                ElseIf messageBoxResult = Windows.Forms.DialogResult.Cancel Then
                    Exit Sub
                End If
```

```
        End If

        Me.TextBoxNote.Clear()
        strFileName = "新文件.txt"
        Me.Text = "写字板-" + strFileName
    End Sub

    Private Sub ToolStripMenuItemFileOpen_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ToolStripMenuItemFileOpen.Click
        If Me.TextBoxNote.Modified Then
            Dim messageBoxResult As DialogResult
            messageBoxResult = MessageBox.Show("当前文档已被修改，是否保存？", "保存",
MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1)
            If messageBoxResult = Windows.Forms.DialogResult.Yes Then
                Me.ToolStripMenuItemFileSaveAs.PerformClick()
            ElseIf messageBoxResult = Windows.Forms.DialogResult.Cancel Then
                Exit Sub
            End If
        End If

        Dim opdg As New OpenFileDialog()
        opdg.Filter = "文本文件(*.txt)|*.txt"

        Dim sr As StreamReader
        If opdg.ShowDialog = System.Windows.Forms.DialogResult.OK Then
            Try
                Me.TextBoxNote.Text = ""
                strFileName = opdg.FileName
                Me.Text = "写字板-" + strFileName
                sr = File.OpenText(opdg.FileName)
                Dim x As String
                While sr.Peek <> -1
                    x = sr.ReadLine()
                    Me.TextBoxNote.Text += x & vbCrLf
                End While
                sr.Close()
            Catch ex As FileNotFoundException
                MsgBox(opdg.FileName & " 没发现！")
            End Try
        End If
    End Sub

    Private Sub ToolStripMenuItemFileSaveAs_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ToolStripMenuItemFileSaveAs.Click
        Dim svdg As New SaveFileDialog()
        svdg.Filter = "文本文件(*.txt)|*.txt"
        svdg.FileName = strFileName

        Dim sr As StreamWriter
        If svdg.ShowDialog = System.Windows.Forms.DialogResult.OK Then
            Try
                strFileName = svdg.FileName
                Me.Text = "写字板-" + strFileName
                sr = File.CreateText(svdg.FileName)
                Dim ch As Char
                For Each ch In Me.TextBoxNote.Text
                    sr.Write(ch)
                Next
                sr.Close()
            Catch ex As Exception
```

```
                MsgBox(ex.Message)
            End Try
        End If
    End Sub

    Private Sub ToolStripMenuItemFileSave_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ToolStripMenuItemFileSave.Click
        If strFileName = "新文件.txt" Then
            Me.ToolStripMenuItemFileSaveAs.PerformClick()
        Else
            Dim sr As StreamWriter
            Try
                sr = File.CreateText(strFileName)
                Dim ch As Char
                For Each ch In Me.TextBoxNote.Text
                    sr.Write(ch)
                Next
                sr.Close()
            Catch ex As Exception
                MsgBox(ex.Message)
            End Try
        End If
    End Sub

    Private Sub FormNote_Resize(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Resize
        Me.TextBoxNote.Width = Me.Width - 8
        Me.TextBoxNote.Height = Me.Height - Me.MenuStripNote.Height - 32
    End Sub

    Private Sub ToolStripMenuItemHelpAbout_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ToolStripMenuItemHelpAbout.Click
        Dim about As FormAbout
        about = New FormAbout()
        about.ShowDialog()
    End Sub

    Private Sub ToolStripMenuItemSetupColor_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ToolStripMenuItemSetupColor.Click
        Dim clrdg As New ColorDialog()
        clrdg.Color = Me.TextBoxNote.BackColor
        If clrdg.ShowDialog = System.Windows.Forms.DialogResult.OK Then
            Me.TextBoxNote.BackColor = clrdg.Color
        End If
    End Sub

    Private Sub ToolStripMenuItemSetupFont_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ToolStripMenuItemSetupFont.Click
        Dim fntdg As New FontDialog()
        fntdg.ShowColor = True
        fntdg.Color = Me.TextBoxNote.ForeColor
        fntdg.Font = Me.TextBoxNote.Font
        If fntdg.ShowDialog = System.Windows.Forms.DialogResult.OK Then
            Me.TextBoxNote.Font = fntdg.Font
            Me.TextBoxNote.ForeColor = fntdg.Color
        End If
    End Sub

    Private Sub FormNote_FormClosing(ByVal sender As System.Object, ByVal e As System.Windows.Forms.FormClosingEventArgs)
```

```
Handles MyBase.FormClosing
        Dim messageBoxResult As DialogResult
        If Me.TextBoxNote.Modified Then
            messageBoxResult = MessageBox.Show("当前文档已被修改，是否保存？", "保存",
MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1)
            Select Case messageBoxResult
                Case Windows.Forms.DialogResult.Yes
                    Me.ToolStripMenuItemFileSave.PerformClick()
                    e.Cancel = False
                Case Windows.Forms.DialogResult.No
                    e.Cancel = False
                Case Windows.Forms.DialogResult.Cancel
                    e.Cancel = True
            End Select
        End If
    End Sub

    Private Sub ToolStripMenuItemEditUndo_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ToolStripMenuItemEditUndo.Click
        If Me.TextBoxNote.CanUndo = True Then
            '撤消文本框中的上一个编辑操作
            Me.TextBoxNote.Undo()
            '从该文本框的撤消缓冲区中清除关于最近操作的信息
            Me.TextBoxNote.ClearUndo()
        End If
    End Sub

    Private Sub TextBoxNote_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
TextBoxNote.TextChanged
        Me.ToolStripMenuItemEditUndo.Enabled = True
    End Sub

    Private Sub TextBoxNote_MouseUp(ByVal sender As System.Object, ByVal e As System.Windows .Forms.MouseEventArgs)
Handles TextBoxNote.MouseUp
        If Me.TextBoxNote.SelectedText() <> "" Then
            Me.ToolStripMenuItemEditCut.Enabled = True
            Me.ToolStripMenuItemEditCopy.Enabled = True
            Me.ToolStripMenuItemCut.Enabled = True
            Me.ToolStripMenuItemCopy.Enabled = True
        Else
            Me.ToolStripMenuItemEditCut.Enabled = False
            Me.ToolStripMenuItemEditCopy.Enabled = False
            Me.ToolStripMenuItemCut.Enabled = False
            Me.ToolStripMenuItemCopy.Enabled = False
        End If
    End Sub
End Class
```

6.4 ListView 控件应用

任务 4 文件浏览器

1．运行界面

运行界面如图 6-4 所示。

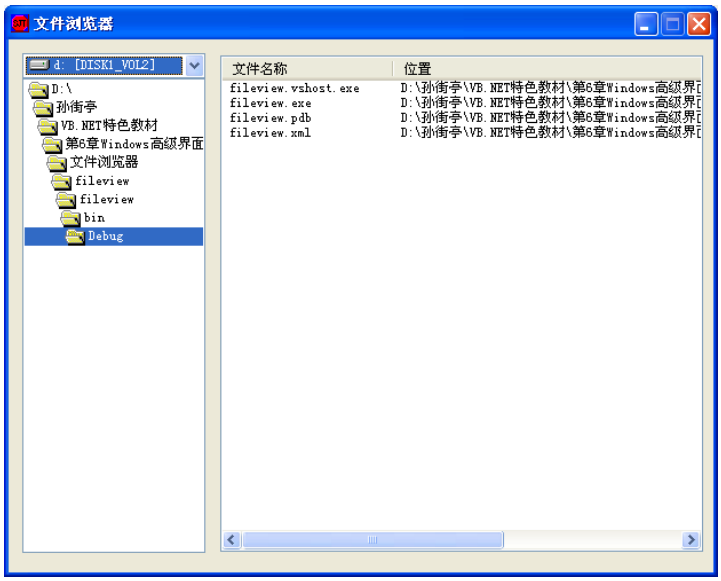

图 6-4　运行界面

2．功能

（1）根据选择的驱动器浏览当前驱动器内的目录。

（2）根据选择的目录浏览当前目录内的文件。

知识点 1　ListView 控件

Windows 窗体 ListView 控件显示了带图标的项的列表。可使用列表视图创建类似于 Windows 资源管理器右窗格的用户界面。该控件具有 4 种视图模式：LargeIcon、SmallIcon、List 和 Details。

大图标视图模式在项文本旁显示大图标；如果控件足够大，则项显示在多列中。小图标视图模式除显示小图标外，其他方面与大图标视图模式相同。列表视图模式显示小图标，但总是显示在单列中。Details 视图模式在多列中显示项。

ListView 控件的主要属性是 Items，该属性包含该控件显示的项。SelectedItems 属性包含控件中当前选定项的集合。如果将 MultiSelect 属性设置为 true，则用户可选择多项，例如，同时将若干项拖放到另一个控件中。如果将 CheckBoxes 属性设置为 true，ListView 控件可以显示这些项旁的复选框。

Activation 属性可以确定用户激活列表中的某项时必须执行的操作类型：选项有 Standard、OneClick 和 TwoClick。执行 OneClick 激活时，需要通过一次单击激活该项。执行 TwoClick 激活时，要求用户通过双击激活该项。

知识点 2　DirectoryInfo 类

公开用于创建、移动和枚举目录和子目录的实例方法。无法继承此类。

1．DirectoryInfo 构造函数

在指定的路径中初始化 DirectoryInfo 类的新实例。

下面的实例使用此构造函数创建指定的目录及子目录，并演示包含子目录的目录不能被删除。

```
Imports System
Imports System.IO
```

```
Public Class Test
    Public Shared Sub Main()
        Dim di1 As DirectoryInfo = New DirectoryInfo("c:\MyDir")
        Dim di2 As DirectoryInfo = New DirectoryInfo("c:\MyDir\temp")
        Try
            di1.Create()
            di2.Create()
            Console.WriteLine("I am about to attempt to delete {0}.", di1.Name)
            di1.Delete()
            Console.WriteLine("The Delete operation was successful, which was unexpected.")
        Catch e As Exception
            Console.WriteLine("The Delete operation failed as expected.")
        End Try
    End Sub
End Class
```

2．DirectoryInfo.Exists 属性

获取指示目录是否存在的值。

属性值：如果目录存在，则为 true，否则为 false。

下面的示例演示了 Exists 属性在将源目录复制到目标目录时的用法。

```
Imports System
Imports System.IO

Module Module1
    Public Sub CopyDirectory(ByVal SourceDirectory As String, ByVal TargetDirectory As String)
        Dim source As DirectoryInfo = New DirectoryInfo(SourceDirectory)
        Dim target As DirectoryInfo = New DirectoryInfo(TargetDirectory)

        If (source.Exists = False) Then
            Return
        End If
        If (target.Exists = False) Then
            target.Create()
        End If

        Dim sourceFiles As FileInfo() = source.GetFiles()
        Dim i, j As Integer
        For i = 0 To sourceFiles.Length - 1
            File.Copy(sourceFiles(i).FullName, target.FullName + "\\" + sourceFiles(i).Name, True)
        Next i

        Dim sourceDirectories As DirectoryInfo() = source.GetDirectories()
        For j = 0 To sourceDirectories.Length - 1
            CopyDirectory(sourceDirectories(j).FullName, target.FullName + "\\" + sourceDirectories(j).Name)
        Next j
        source = Nothing
        target = Nothing
    End Sub

    Sub Main()
        CopyDirectory("D:\\Tools", "D:\\NewTools")
    End Sub
End Module
```

3．DirectoryInfo.Name 属性

获取此 DirectoryInfo 实例的名称。

属性值：目录名称。

下面的实例仅显示当前 DirectoryInfo 实例的名称。

```
Imports System
Imports System.IO

Class GetAName
    Public Shared Sub Main()
        Dim dir As New DirectoryInfo(".")
        Dim dirName As String = dir.Name
        Console.WriteLine("DirectoryInfo name is {0}.", dirName)
    End Sub
End Class
```

4．DirectoryInfo.Parent 属性

获取指定子目录的父目录。

属性值：父目录，或者如果路径为空或如果文件路径表示根（如“\”、“C:”或*“\\server\share”），则为空引用（在 Visual Basic 中为 Nothing）。

下面的实例演示了引用指定目录的父目录的方法。

```
Imports System
Imports System.IO

Public Class MoveToTest

    Public Shared Sub Main()
        Dim di As New DirectoryInfo("TempDir")
        If di.Exists = False Then
            di.Create()
        End If

        Dim dis As DirectoryInfo = di.CreateSubdirectory("SubDir")
        Dim parentDir As DirectoryInfo = dis.Parent
        Console.WriteLine("The parent directory of '{0}' is '{1}'", dis.Name, parentDir.Name)
        di.Delete(True)
    End Sub
End Class
```

5．DirectoryInfo.Root 属性

获取路径的根部分。

属性值：代表路径的根的 DirectoryInfo 对象。

下面的实例演示了如何确定指定目录的根目录位置。

```
Imports System
Imports System.IO

Public Class MoveToTest

    Public Shared Sub Main()
        Dim di As New DirectoryInfo("TempDir")
        If di.Exists = False Then
            di.Create()
        End If

        Dim dis As DirectoryInfo = di.CreateSubdirectory("SubDir")
        Console.WriteLine("The root path of '{0}' is '{1}'", dis.Name, dis.Root)
        di.Delete(True)
    End Sub
End Class
```

6．DirectoryInfo.Create 方法

创建目录。如果目录已经存在，则此方法不执行任何操作。

下面的实例检查指定目录是否存在；如果该目录不存在，则创建此目录，然后删除该目录。

```
Imports System
Imports System.IO

Public Class Test

    Public Shared Sub Main()
        Dim di As DirectoryInfo = New DirectoryInfo("c:\MyDir")
        Try
            If di.Exists Then
                Console.WriteLine("That path exists already.")
                Return
            End If

            di.Create()
            Console.WriteLine("The directory was created successfully.")

            di.Delete()
            Console.WriteLine("The directory was deleted successfully.")

        Catch e As Exception
            Console.WriteLine("The process failed: {0}", e.ToString())
        End Try
    End Sub
End Class
```

7．DirectoryInfo.Delete 方法

如果此 DirectoryInfo 为空，则删除它。

下面的实例在试图删除一个非空目录时引发异常。

```
Imports System
Imports System.IO

Public Class Test
    Public Shared Sub Main()
        Dim di1 As DirectoryInfo = New DirectoryInfo("c:\MyDir")
        Try
            di1.Create()
            di1.CreateSubdirectory("temp")
            Console.WriteLine("I am about to attempt to delete {0}", di1.Name)
            di1.Delete()
            Console.WriteLine("The Delete operation was successful, which was unexpected.")
        Catch
            Console.WriteLine("The Delete operation was unsuccessful, as expected.")
        End Try
    End Sub
End Class
```

8．DirectoryInfo.GetDirectories 方法

返回当前目录的子目录。如果没有子目录，则此方法只返回根目录。

返回值：DirectoryInfo 对象的数组。

下面的实例检索根目录下的所有目录并显示目录的名称。

```
Imports System
Imports System.IO

Public Class GetDirectoriesTest
    Public Shared Sub Main()
        Dim di As New DirectoryInfo("c:\")
        Dim diArr As DirectoryInfo() = di.GetDirectories()
        Dim dri As DirectoryInfo
        For Each dri In diArr
```

```
            Console.WriteLine(dri.Name)
        Next dri
    End Sub
End Class
```

9．DirectoryInfo.GetFiles 方法

返回当前目录的文件列表。如果 DirectoryInfo 中没有文件，则此方法返回一个空数组。

返回值：FileInfo 类型数组。

下面的实例从指定的目录检索文件。

```
Imports System
Imports System.IO
Public Class GetFilesTest
    Public Shared Sub Main()
        Dim di As New DirectoryInfo("c:\")
        Dim fiArr As FileInfo() = di.GetFiles()
        Dim fri As FileInfo
        For Each fri In fiArr
            Console.WriteLine(fri.Name)
        Next fri
    End Sub
End Class
```

知识点 3　FileInfo 类

提供创建、复制、删除、移动和打开文件的实例方法，并且帮助创建 FileStream 对象。无法继承此类。

1．FileInfo 构造函数

初始化 FileInfo 类的新实例，它作为文件路径的包装。

参数：fileName，新文件的完全限定名或相对文件名。

下面的实例使用此构造函数创建两个文件，并接着对其进行写入、读取、复制和删除操作。

```
Imports System
Imports System.IO

Class Test
    Public Shared Sub Main()
        Dim path As String = "c:\temp\MyTest.txt"
        Dim fi1 As FileInfo = New FileInfo(path)

        If fi1.Exists = False Then
            Dim sw As StreamWriter = fi1.CreateText()
            sw.WriteLine("Hello")
            sw.WriteLine("And")
            sw.WriteLine("Welcome")
            sw.Flush()
            sw.Close()
        End If

        Dim sr As StreamReader = fi1.OpenText()
        Do While sr.Peek() >= 0
            Console.WriteLine(sr.ReadLine())
        Loop

        Try
            Dim path2 As String = path + "temp"
            Dim fi2 As FileInfo = New FileInfo(path2)

            fi2.Delete()
```

```
            fi1.CopyTo(path2)
            Console.WriteLine("{0} was copied to {1}.", path, path2)
            fi2.Delete()
            Console.WriteLine("{0} was successfully deleted.", path2)
        Catch e As Exception
            Console.WriteLine("The process failed: {0}", e.ToString())
        End Try
    End Sub
End Class
```

2．FileInfo.Directory 属性

获取父目录的实例。

属性值：表示此文件父目录的 DirectoryInfo 对象。

下面的实例打开或创建一个文件，确定其完整路径，确定并显示目录的所有内容。

```
Imports System
Imports System.IO

Public Class DirectoryTest
    Public Shared Sub Main()
        Dim fi As New FileInfo("temp.txt")
        Dim di As DirectoryInfo = fi.Directory
        Dim fsi As FileSystemInfo() = di.GetFileSystemInfos()
        Console.WriteLine("The directory '{0}' contains the following files and directories:", di.FullName)
        Dim info As FileSystemInfo
        For Each info In fsi
            Console.WriteLine(info.Name)
        Next info
    End Sub
End Class
```

3．FileInfo.DirectoryName 属性

获取表示目录的完整路径的字符串。

属性值：表示目录的完整路径的字符串。

下面的实例检索指定文件的完整路径。

```
Dim fileName As String = "C:\autoexec.bat"
Dim fileInfo As New FileInfo(fileName)
If Not fileInfo.Exists Then
    Return
End If
Console.WriteLine("{0} has a directoryName of {1}", fileName, fileInfo.DirectoryName)
```

4．FileInfo.Exists 属性

获取指示文件是否存在的值。

属性值：如果该文件存在，则为 true；如果该文件不存在或如果该文件是目录，则为 false。

下面的实例使用 Exists 属性确保文件在打开之前已经存在。

```
Function OpenDataFile(ByVal FileName As String) As Byte()
    If FileName Is Nothing OrElse FileName.Length = 0 Then
        Throw New ArgumentNullException("FileName")
    End If
    Dim fInfo As New FileInfo(FileName)
    If Not fInfo.Exists Then
        Throw New FileNotFoundException("The file was not found.", FileName)
    End If

    Dim fStream As New FileStream(FileName, FileMode.Open)
    Dim buffer(fStream.Length) As Byte
    fStream.Read(buffer, 0, Fix(fStream.Length))
    Return buffer
```

```
End Function
```

5．FileInfo.Length 属性

获取当前文件的大小。如果包含该文件的文件系统不支持此信息，则此属性值为空引用（在 Visual Basic 中为 Nothing）。

属性值：当前文件的大小。

下面的实例显示指定文件的大小。

```
Imports System
Imports System.IO

Public Class FileLength
    Public Shared Sub Main()
        Dim di As New DirectoryInfo("c:\")
        Dim fiArr As FileInfo() = di.GetFiles()
        Dim f As FileInfo
        Console.WriteLine("The directory {0} contains the following files:", di.Name)
        For Each f In fiArr
            Console.WriteLine("The size of {0} is {1} bytes.", f.Name, f.Length)
        Next f
    End Sub
End Class
```

6．FileInfo.Name 属性

获取文件名。

属性值：文件名。

下面的实例使用 Name 属性显示当前目录中的文件名。

```
Imports System
Imports System.IO

Public Class NameTest
    Public Shared Sub Main()
        Dim di As New DirectoryInfo(Environment.CurrentDirectory)
        Dim fi As FileInfo() = di.GetFiles()
        Console.WriteLine("The following files exist in the current directory:")
        Dim fiTemp As FileInfo
        For Each fiTemp In fi
            Console.WriteLine(fiTemp.Name)
        Next fiTemp
    End Sub
End Class
```

7．FileInfo.Create 方法

创建文件。默认情况下，将向所有用户授予对新文件的完全读/写访问权限。

返回值：新文件。

下面的示例创建对文件的引用，然后使用 FileInfo.Create()在磁盘上创建此文件。

```
Imports System
Imports System.IO

Public Class DeleteTest
    Public Shared Sub Main()
        Dim fi As New FileInfo("temp.txt")
        Dim fs As FileStream = fi.Create()
        fs.Close()
        fi.Delete()
    End Sub
End Class
```

8．FileInfo.CopyTo 方法

将现有文件复制到新文件。

格式 1：

```
FileInfo.CopyTo(String)                    '将现有文件复制到新文件，不允许改写现有文件
```

Visual Basic（声明）：

```
Public Function CopyTo ( _
   destFileName As String _
) As FileInfo
```

参数：destFileName 为要复制到的新文件的名称。

返回值：带有完全限定路径的新文件。

格式 2：

```
FileInfo.CopyTo(String, Boolean)           '将现有文件复制到新文件，允许改写现有文件
```

Visual Basic（声明）：

```
Public Function CopyTo ( _
   destFileName As String, _
   overwrite As Boolean _
) As FileInfo
```

参数：destFileName 为要复制到的新文件的名称；overwrite 若为 true，则允许改写现有文件；否则为 false。

返回值：新文件，或者如果 overwrite 为 true，则改写现有文件。如果文件存在，且 overwrite 为 false，则会发生 IOException。

下面的实例演示了 CopyTo 方法的两个重载。

```
Imports System
Imports System.IO

Public Class Test
    Public Shared Sub Main()
        Dim path As String = "c:\temp\MyTest.txt"
        Dim path2 As String = path + "temp"
        Dim fi As FileInfo = New FileInfo(path)
        Dim fi2 As FileInfo = New FileInfo(path2)
        Try
            Dim fs As FileStream = fi.Create()
            fs.Close()
            fi2.Delete()
            fi.CopyTo(path2)
            Console.WriteLine("{0} was copied to {1}.", path, path2)
            fi.CopyTo(path2, True)
            Console.WriteLine("The second Copy operation succeeded, which is expected.")
        Catch
            Console.WriteLine("Double copying was not allowed, which is not expected.")
        End Try
    End Sub
End Class
```

9．FileInfo.Delete 方法

永久删除文件。如果文件不存在，则此方法不执行任何操作。

下面的实例说明 Delete 方法。

```
Imports System
Imports System.IO
Imports System.Text

Public Class Test
    Public Shared Sub Main()
```

```
        Dim path As String = "c:\temp\MyTest.txt"
        Dim fi As FileInfo = New FileInfo(path)
        Try
            Dim sw As StreamWriter = fi.CreateText()
            sw.Close()
            Dim path2 As String = path + "temp"
            Dim fi2 As FileInfo = New FileInfo(path2)
            fi2.Delete()
            fi.CopyTo(path2)
            Console.WriteLine("{0} was copied to {1}.", path, path2)
            fi2.Delete()
            Console.WriteLine("{0} was successfully deleted.", path2)
        Catch e As Exception
            Console.WriteLine("The process failed: {0}", e.ToString())
        End Try
    End Sub
End Class
```

知识点 4　DirListBox、DriveListBox 和 FileListBox 控件

建议使用 OpenFileDialog 和 SaveFileDialog 组件提供对文件系统的访问；但是，如果创建自己的文件对话框，可以使用 Visual Basic 2005 在 Microsoft Visual Basic 兼容性运行库中提供的 DirListBox、DriveListBox 和 FileListBox 控件。

可以使用下列过程将 DirListBox、DriveListBox 或 FileListBox 添加到 Windows 应用程序项目。

添加对兼容性库的引用：

（1）选择“项目”→“添加引用”命令。

（2）在弹出的“添加引用”对话框中，单击“.NET”选项卡。在“组件名称”列表中选择 Microsoft.VisualBasic.Compatibility，然后单击“确定”按钮，一个引用随即会添加到项目中。

将控件添加到工具箱的方法如下：

（1）选择“视图”→“工具箱”命令。

（2）展开“所有 Windows 窗体”部分。右击其标题栏，然后单击“选择项”。

（3）在“选择工具箱项”对话框中选中 DirListBox、DriveListBox 和 FileListBox 组件，然后单击“确定”按钮，这些控件随即会添加到“工具箱”中，可以像任何其他 Windows 窗体控件一样使用它们。

任务 4 解析

1．控件的属性（如表 6-12 所示）

表 6-12　控件属性说明

控件	Name 属性	Text 属性	其他属性
Form	FormView	文件浏览器	
DriveListBox	DriveListBoxView		
DirListBox	DirListBoxView		
ListView	ListViewFile		

2．源代码

```
Public Class FormView
```

```
        Private Sub DriveListBoxView_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles DriveListBoxView.SelectedIndexChanged
            Me.DirListBoxView.Path = Me.DriveListBoxView.Drive
        End Sub

        Private Sub PopulateListView()
            Me.ListViewFile.View = View.Details
            Dim header1, header2 As ColumnHeader
            header1 = New ColumnHeader
            header2 = New ColumnHeader
            header1.Text = "文件名称"
            header1.TextAlign = HorizontalAlignment.Left
            header1.Width = 150
            header2.Text = "位置"
            header2.TextAlign = HorizontalAlignment.Left
            header2.Width = 550
            Me.ListViewFile.Columns.Add(header1)
            Me.ListViewFile.Columns.Add(header2)
            Dim dirinfo As New System.IO.DirectoryInfo(Me.DirListBoxView.Path)
            Dim file As System.IO.FileInfo
            Dim files() As System.IO.FileInfo = dirinfo.GetFiles()
            Me.ListViewFile.Items.Clear()
            If Not (files Is Nothing) Then
                For Each file In files
                    Dim item As New ListViewItem(file.Name)
                    item.SubItems.Add(file.FullName)
                    Me.ListViewFile.Items.Add(item)
                Next
            End If
        End Sub

        Private Sub DirListBoxView_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles DirListBoxView.SelectedIndexChanged
            Me.PopulateListView()
        End Sub
    End Class
```

1．Visual Basic.NET 中，程序员修改了主窗体的某个属性后，发现无法启动程序，原因可能是（　　）。

A．修改了主窗体的 Caption 属性　　B．修改了主窗体的 isMainForm 属性

C．修改了主窗体的 Name 属性　　D．修改了 Main 函数

2．在窗体的成员方法 dosomthing 中，将窗体位置居中显示，应调用（　　）窗体方法。

A．Center()　　B．CenterToScreen()

C．MoveToCenter()　　D．Show()

3．（　　）窗体在关闭之前不允许用户与程序中其他窗体进行交互。

A．主窗体　　B．对话框　　C．模态窗体　　D．非模态窗体

4．Visual Basic.NET 窗体中提供的是 Hide 方法的作用是（　　）。

A．销毁窗体对象　　B．关闭窗体

C．将窗体极小化　　D．隐藏窗体

5．Visual Basic.NET 窗体对象的 Close 方法的作用是（　　）。

A．极小化窗体　　B．隐藏窗体　　C．关闭窗体　　D．销毁窗体对象

6．在使用 OpenFileDialog 对话框控件时，希望通过对话框只查找 C++文件的程序文件（*.CPP）文件和头文件（*.H），则 OpenFileDialog 对象的 Filter 属性应（　　）设置。

A．“C++程序文件:*.CPP;C++头文件:*.H”

B．“C++程序文件|*.CPP|C++头文件|*.H”

C．“C++程序文件*.CPP\C++头文件*.H”

D．“C++程序文件->*.CPP;C++头文件->*.H”

7．Win 窗体设计时，应该为工具栏的（　　）事件编写事件处理程序，以响应用户单击工具栏的按钮。

A．Click　　B．ButtonClick　　C．KeyDown　　D．MouseDown

8．Win 窗体的工具栏对象为 ToolBar1。为了在工具栏中添加新的按钮，应该在对象 ToolBar1 的“属性”视图下编辑 ToolBar1 的（　　）属性。

A．Appearance　　B．ImageList　　C．Buttons　　D．ShowToolTips

9．Win 窗体的工具栏对象为 ToolBar1。为了将工具栏中的所有按钮用图形显示，则应该在对象 ToolBar1 的“属性”视图下编辑 ToolBar1 的（　　）属性。

A．Appearance　　B．ImageList　　C．Buttons　　D．ShowToolTips

10．若不准备使用状态栏的窗格显示信息，则应该将（　　）属性设置为 False。

A．ShowPanels　　B．Panels　　C．Enabled　　D．Visible

11．如要实现菜单功能，应向菜单项的（　　）事件添加代码。

A．Command　　B．Click　　C．Popup　　D．Select

12．Visual Basic.NET 窗体常见属性中，_____属性用于显示控件文本，_____属性用于显示控件文本的字体，_____属性用于设置窗体在项目中的名称。

13．Visual Basic.NET 中，用于显示某个窗体的方法是_____，用于关闭窗体，把窗体从内存中清除的方法是_____，用于隐藏窗体，但窗体仍然在内存中的方法是_____。

14．在列表框添加选项有哪几种方法？

15．如果要一个定时器每半分钟产生一个 Tick 控件，则 InterVal 属性应设置为多少？

16．代码中如何判断用户在通用对话框中选择了“取消”按钮？

17．创建弹出菜单需要什么控件？通过什么属性才能将控件与弹出菜单建立关联？

18．什么是模式对话框？什么是非模式对话框？

19．窗体之间数据互访如何才能实现？