CHAPTER 第3章 常用控件

控件是在系统内部定义的用于和用户交互的基本单元。在所有的控件中，根据它们的使用及 Visual C++ 对其支持的情况，可以把控件分为 Windows 一般控件（即早期的如编辑框、列表框、组合框和按钮等）、通用控件（如列表视图、树视图等控件）、MFC 扩展控件和 ActiveX 控件。ActiveX 控件可以理解成一个 OLE（Object Linking and Embedding，对象连接与嵌入）组件，它既可用于 Windows 应用程序中，也可用于 Web 页面中。

本章重点介绍在 Windows 应用程序中经常使用的控件，主要有静态控件、按钮、编辑框、列表框、组合框、滚动条、进展条、旋转按钮控件、滑动条、日期时间控件（包括计时器）、列表控件和树控件。

3.1 创建和使用控件

在 MFC 应用程序中使用控件不仅可简化编程，还能完成常用的各种功能。为了更好地发挥控件作用，还必须理解和掌握控件的属性、消息、变量以及创建和使用的方法。

3.1.1 控件的创建方式

控件的创建方式有以下两种。

一种是可视化方式，即在对话框模板中用编辑器指定控件，也就是说，将对话框看作控件的父窗口。这样做的好处是显而易见的，因为当应用程序启动该对话框时，系统就会为对话框创建控件，而当对话框消失时，控件也随之清除。

另一种是编程方式，即调用 MFC 相应控件类的成员函数 Create() 来创建，并在 Create() 函数中指定控件的父窗口指针。例如，下面的示例过程是使用编程方式来创建一个按钮。

【例 Ex_Create】

（1）在“D:\Visual C++ 程序”文件夹中，创建本章应用程序工作文件夹“第 3 章”。

（2）启动 Visual C++，选择“文件”→“新建”→“项目”菜单命令、按快捷键 Ctrl+Shift+N 或单击标准工具栏中的按钮，弹出“新建项目”对话框。在左侧“项目类型”中选中 MFC，在右侧的“模板”栏中选中 MFC 应用程序 类型，检查并将项目工作文件夹定位到“D:\Visual C++ 程序\第 3 章”，在“名称”栏中输入项目名“Ex_Create”，检查并取消勾选“为解决方案创建目录”复选框。

(3) 单击 确定 按钮,出现“MFC 应用程序向导”欢迎页面,单击 下一步 > 按钮,出现“应用程序类型”页面。选中“基于对话框”应用程序类型,此时右侧的“项目类型”自动选定为“MFC 标准”,取消勾选“使用 Unicode 库”复选框。

(4) 保留其他默认选项,单击 完成 按钮,系统开始创建,并又回到了 Visual C++ 主界面,同时还自动打开对话框资源(模板)编辑器以及控件工具栏、控件布局工具栏等。将项目工作区窗口切换到“解决方案管理器”页面,双击头文件结点 stdafx.h ,打开 stdafx.h 文档,滚动到最后代码行,将“#ifdef _UNICODE”和最后一行的“#endif”注释掉。

说明:为叙述方便,以后“创建一个默认的对话框应用程序”就是指上述过程,本书做此约定。

(5) 将工作区窗口切换到“类视图”页面,展开 Ex_Create 所有的类结点,右击 CEx_CreateDlg 类名,弹出如图 3.1 所示的快捷菜单。从弹出的快捷菜单中选择“添加”→“添加变量”命令,弹出“添加成员变量向导”对话框,如图 3.2 所示。在“变量类型”框中输入“CButton”(MFC 按钮类),在“变量名”框中输入要定义的 CButton 类对象名“m_btnWnd”。保留其他默认选项,单击 完成 按钮,向导开始添加。

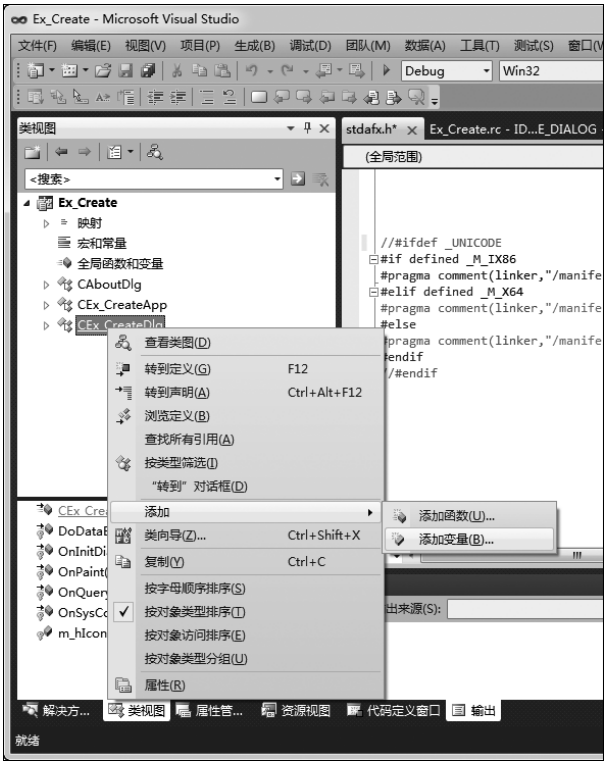

图 3.1 弹出的快捷菜单

注意:对象名通常以“m_”作为开头,表示“成员”(member)的意思。

需要说明的是,在 MFC 中,每一种类型的控件都用相应的类来封装。如编辑框控件的类是 CEdit,按钮控件的类是 CButton,通过这些类创建的对象来访问其成员,从而实现控件的相关操作。

图 3.2　添加控件类对象成员

(6) 在项目工作区窗口的“类视图”页面中，双击 CEx_CreateDlg 类“成员”区域中的 OnInitDialog 函数名结点，在打开的函数定义中添加下列代码(在 return TRUE 语句前添加)。

```
BOOL CEx_CreateDlg::OnInitDialog()
{
    CDialogEx::OnInitDialog();
    …
    m_btnWnd.Create("你好", WS_CHILD | WS_VISIBLE
                            | BS_PUSHBUTTON | WS_TABSTOP,
                            CRect(20, 20, 120, 44), this, 201);     //创建
    CFont * font = this->GetFont();         //获取对话框的字体
    m_btnWnd.SetFont(font);                 //设置控件字体
    return TRUE;                            //除非将焦点设置到控件，否则返回 TRUE
}
```

说明：

① 前面曾说过，由于 OnInitDialog()函数在对话框初始化时被调用，因此将对话框中的一些初始化代码都添加在此函数中。

② 由于 Windows 操作系统使用的是图形界面，因此在 MFC 中，对于每种界面元素的几何大小和位置常使用 CPoint(点)类、CSize(大小)类和 CRect(矩形)类来描述(以后还会讨论)。

③ 代码中，CButton 类成员函数 Create()用来创建按钮控件。该函数第一个参数用来

指定按钮的标题。第二个参数用来指定按钮控件的样式,其中,BS_PUSHBUTTON(以BS_开头的)是按钮类封装的预定义样式,表示创建的是按键按钮。WS_CHILD(子窗口)、WS_VISIBLE(可见)、WS_TABSTOP(可用 Tab 键选择)等都是 CWnd 类封装的预定义窗口样式,它们都可以直接引用。当多个样式指定时,需要使用按位或运算符"|"来连接。第三个参数用来指定按钮在父窗口中的位置和大小。第四个参数用来指定父窗口指针。最后一个参数是指定该控件的标识值。

④ 由于按钮是作为对话框的一个子窗口来创建的,因此 WS_CHILD 样式是必不可少的,且还要使用 WS_VISIBLE 使控件在创建后显示出来。

(7) 编译并运行,结果如图 3.3 所示。

图 3.3 Ex_Create 运行结果

由以上可以看出,控件编程创建方法是使用各自封装的类的 Create 成员来创建的,它最大的优点就是能动态创建,但它涉及的编程代码比较复杂,且不能发挥对话框编辑器可视化的优点,故在一般情况下都采用第一种方法,即在对话框模板中用编辑器指定控件。

3.1.2 控件的消息及消息映射

应用程序创建一般控件或公共控件之后,当控件的状态发生改变(例如,用户利用控件进行输入)时,控件就会向其父窗口发送消息,这个消息称为"通知消息"。对于每条消息,系统都会用一个 MSG 结构来记录,MSG 具有下列原型。

```
typedef struct tagMSG {        //msg
    HWND    hwnd;              //接收到消息的窗口句柄
    UINT    message;           //消息
    WPARAM  wParam;            //消息的附加信息,它的含义取决于 message
    LPARAM  lParam;            //消息的附加信息,它的含义取决于 message
    DWORD   time;              //消息传送时的时间
    POINT   pt;                //消息传送时,光标所在的屏幕坐标
} MSG;
```

对于一般控件来说,其通知消息通常是一条 WM_COMMAND 消息,这条消息的

wParam 参数的低位字中含有控件标识符，wParam 参数的高位字则为通知代码，lParam 参数则是指向控件的句柄。

而对于有些控件，其通知消息通常是一条 WM_NOTIFY 消息，这条消息的 wParam 参数是发送通知消息的控件的标识符，而 lParam 参数则是指向一个特殊结构的指针。

1. 映射控件消息

不管是什么控件消息，一般都可以用“属性”窗口的“事件”页面、“MFC 类向导”以及控件的快捷方式（后面会说明）对它们加以映射。例如，下面的示例过程。

（1）将项目工作区窗口切换到“资源视图”页面，展开所有资源结点，双击 Dialog 资源类别下的标识 IDD_EX_CREATE_DIALOG，打开 Ex_Create 项目的对话框资源（模板）。

（2）选中“TODO：在此放置对话框控件。”控件，按 Delete 键删除。从控件工具箱中拖放添加一个按钮控件，保留其默认属性。

（3）在对话框资源（模板）空白处右击，从弹出的快捷菜单中选择“属性”命令，弹出其“属性”窗口，在窗口上部单击“事件”按钮，将其切换到“事件”页面，找到并展开 IDC_BUTTON1 结点，可以看到该结点下可映射的事件所产生的消息列表。

（4）在 BN_CLICKED 消息右侧栏单击，然后单击右侧的下拉按钮，从弹出的下拉项中选择“添加 OnBnClickedButton1”（默认处理函数名称），如图 3.4 所示。

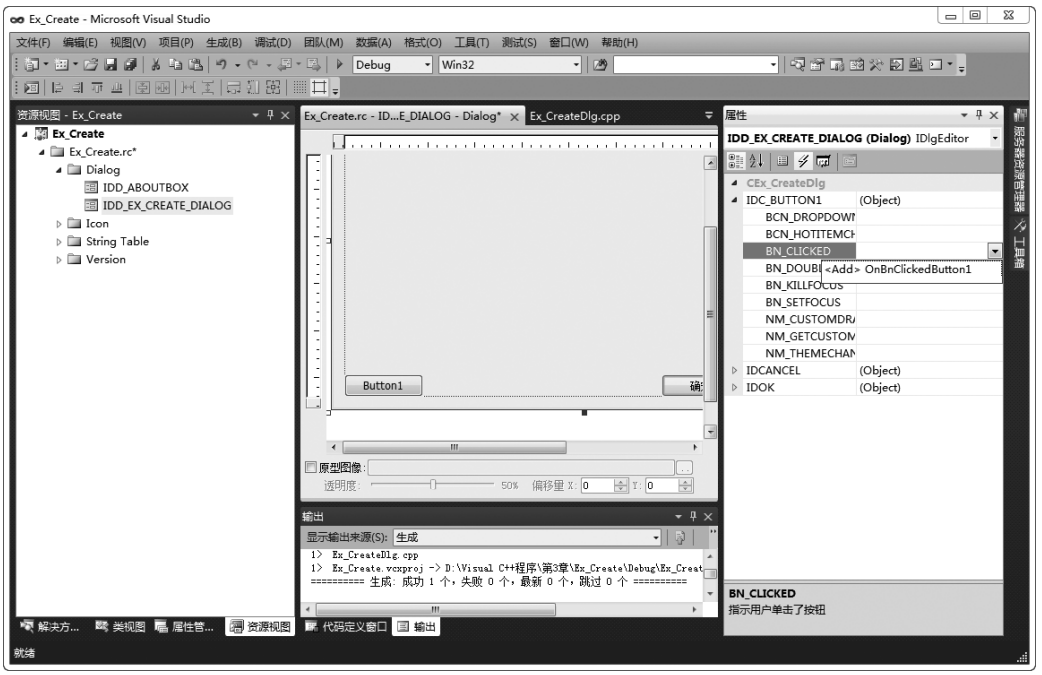

图 3.4　添加按钮消息映射函数

（5）此时自动转向文档窗口，并定位到 CEx_CreateDlg::OnBnClickedButton1()函数实现的源代码处。关闭对话框资源（模板）的“属性”窗口，添加下列代码。

```
void CEx_CreateDlg::OnBnClickedButton1()
{
    MessageBox("你按下了\"Button1\"按钮!");
}
```

(6) 编译并运行,当单击 Button1 按钮时,就会执行 OnBnClickedButton1()函数,弹出一个消息对话框,显示"你按下了"Button1"按钮!"内容。

说明:

① 不同资源对象(控件、菜单项等)所产生的消息是不相同的。例如,按钮控件 IDC_BUTTON1 的通知消息最常用的有两个:BN_DOUBLECLICKED 和 BN_CLICKED,分别表示当用户双击或单击该按钮时事件所产生的通知消息。

② 一般不需要对对话框中的"确定"与"取消"按钮进行消息映射处理,因为系统已自动设置了这两个按钮的动作,当用户单击这两个按钮时都将自动关闭对话框,且"确定"按钮动作还使得对话框数据有效。

③ 控件的消息映射处理也可通过在控件上右击,从弹出的快捷菜单中选择"添加事件处理程序"命令,通过弹出的"事件处理程序向导"对话框进行。

这就是一个按钮的 BN_CLICKED 消息的映射处理过程,其他控件的消息映射处理也可类似进行。

2. 映射控件通用消息

上述过程是映射一个控件的某一条消息,事实上,也可以通过 WM_COMMAND 消息处理虚函数 OnCommand()的重写(重载)来处理一个或多个控件的通用消息,如下面的过程。

(1) 将项目工作区窗口切换到"类视图"页面,右击 CEx_CreateDlg 类结点,从弹出的快捷菜单中选择"属性"命令,弹出其"属性"窗口,在窗口上部单击"重写"按钮,将其切换到"重写"页面,找到 OnCommand,在其右侧栏单击,然后单击右侧的下拉按钮,从弹出的下拉项中选择"添加 OnCommand",如图 3.5 所示,这样 OnCommand()虚函数重写(重载)函数就添加好了。

(2) 此时自动转向文档窗口,并定位到 CEx_CreateDlg::OnCommand()函数实现的源代码处。关闭"属性"窗口,添加下列代码。

```
BOOL CEx_CreateDlg::OnCommand(WPARAM wParam, LPARAM lParam)
{
    WORD nCode = HIWORD(wParam);                    //控件的通知消息
    WORD nID   = LOWORD(wParam);                    //控件的 ID
    if((nID == 201)&&(nCode == BN_CLICKED))
        MessageBox("你按下了\"你好\"按钮!");
    if((nID == IDC_BUTTON1)&&(nCode == BN_CLICKED))
        MessageBox("这是在 OnCommand 处理的结果!");
    return CDialogEx::OnCommand(wParam, lParam);
}
```

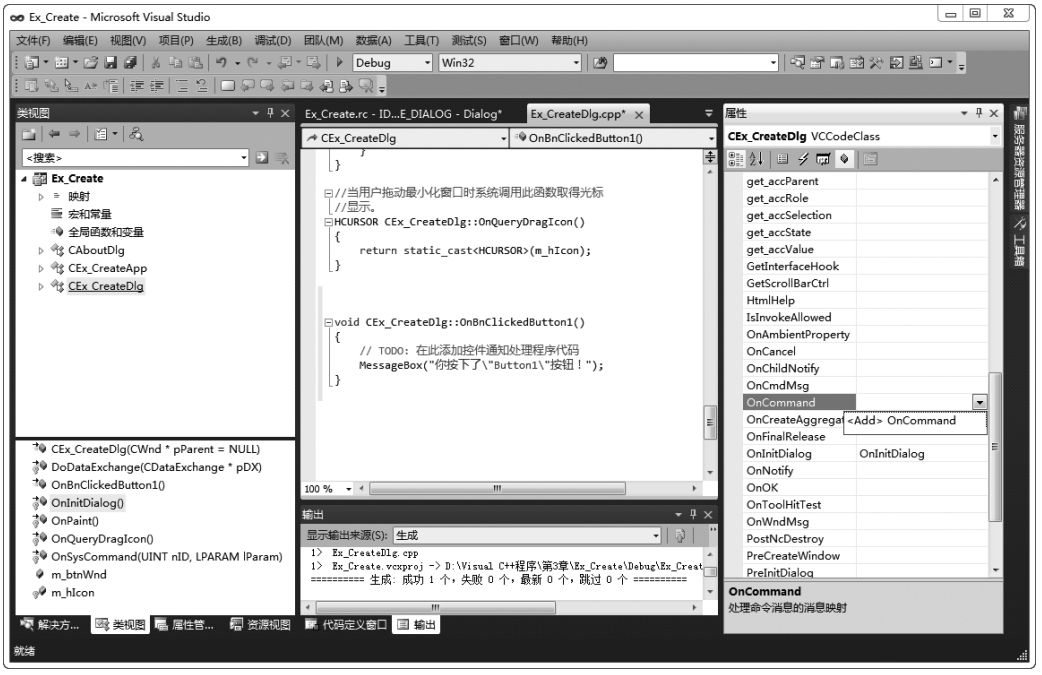

图 3.5　添加 OnCommand()虚函数的重写(重载)

(3) 编译并运行。当单击如图 3.3 所示的对话框中的 你好 按钮时，弹出消息对话框，显示“你按下了"你好"按钮！”内容。

说明：

① 在 MFC 中，资源都是用其 ID 来标识的，而各资源的 ID 本身就是数值，因此上述代码中，201 和 IDC_BUTTON1 都是程序中用来标识按钮控件的 ID，201 是前面创建控件时指定的 ID 值。

② 在上述编写的代码中，Button1 按钮的 BN_CLICKED 消息用不同的方式处理了两次，即同时存在两种函数 OnBnClickedButton1()和 OnCommand()，因此若单击 Button1 按钮，系统会先执行哪一个函数呢？测试的结果表明，系统首先执行 OnCommand()函数，然后执行 OnBnClickedButton1()代码。之所以还能执行 OnBnClickedButton1()()函数代码，是因为 OnCommand()函数的最后一句代码“return CDialogEx:: OnCommand(wParam, lParam);”，它将控件的消息交由对话框其他函数处理。

③ 由于用 Create 创建的控件无法用“属性”窗口的“事件”页面、“MFC 类向导”等直接映射其消息，因此上述 OnCommand()方法弥补了控件事件处理的不足，使用时要特别留意。

3.1.3　控件类和控件对象

一旦创建控件后，有时就需要使用控件进行深入编程。控件在使用时先要获得该控件的类对象指针或引用或绑定的类对象，然后通过该指针或引用或类对象来调用其成员函数进行操作。表 3.1 列出了 MFC 封装的常用控件类。

表 3.1 常用控件类

控件名称(工具箱中的名称)	MFC 类	功 能 描 述
静态控件(多个)	CStatic	用来显示一些几乎固定不变的文字或图形
按钮(Button)	CButton	用来产生某些命令或改变某些选项,包括单选按钮、复选框和组框
编辑框(Edit Control)	CEdit	用于完成文本和数字的输入和编辑
列表框(List Box)	CListBox	显示一个列表,让用户从中选取一个或多个项
组合框(Combo Box)	CComboBox	是一个列表框和编辑框组合的控件
滚动条(2 个)	CScrollBar	通过滚动块在滚动条上的移动和滚动按钮来改变某些量
进展条(Progress Control)	CProgressCtrl	用来表示一个操作的进度
滑动条(Slider Control)	CSliderCtrl	通过滑动块的移动来改变某些量,并带有刻度指示
旋转按钮(Spin Control)	CSpinButtonCtrl	带有一对反向箭头的按钮,单击这对按钮可增加或减少某个值
日期时间选择器(Date Time Picker)	CDateTimeCtrl	用于选择指定的日期和时间
图像列表(无)	CImageList	一个具有相同大小的图标或位图的集合
列表控件(List Control)	CListCtrl	可以用“大图标”“小图标”“列表视图”或“报表视图”等多种列表方式来显示一组信息
树控件(Tree Control)	CTreeCtrl	以根、父、子结点构成的树来显示一组信息
标签控件(Tab Control)	CTabCtrl	类似于一个笔记本的分隔器或一个文件柜上的标签,使用它可以将一个窗口或对话框的相同区域定义为多个页面

在 MFC 中,获取一个控件的类对象指针是通过 CWnd 类的成员函数 GetDlgItem()来实现的,它具有下列原型。

```
CWnd* GetDlgItem(int nID) const;
void GetDlgItem(int nID, HWND* phWnd) const;
```

C++ 允许同一个类中出现同名的成员函数,只是这些同名函数的形参类型或形参个数必定各不相同(为叙述方便,这些同名函数从上到下依次称为第 1 版本、第 2 版本、……)。其中,nID 用来指定控件或子窗口的 ID 值,第 1 版本是直接通过函数来返回 CWnd 类指针,而第 2 版本是通过函数形参 phWnd 来返回其句柄指针。

特别地,由于 CWnd 类是通用的窗口基类,因此想要调用实际的控件类及其基类成员,还必须对其进行类型的强制转换。例如:

```
CButton* pBtn = (CButton*)GetDlgItem(IDC_BUTTON1);
```

由于 GetDlgItem()获取的是类对象指针，因而它可以用到程序的任何地方，且可多次使用，并可对同一个控件定义不同的对象指针，均可对指向的控件操作有效。

与控件关联的变量可通称为控件变量，在 MFC 中，控件变量分为两个类别，一是用于操作的控件类对象，二是用于存取的数据变量。它们都与控件或子窗口进行绑定，但 MFC 只允许每个类别绑定一次。下面就来看一个示例。

【例 Ex_Member】

(1) 创建一个默认的对话框应用程序 Ex_Member。其中要将 stdafx.h 文件最后面内容中的＃ifdef _UNICODE 行和最后一个＃endif 行删除(注释掉)。

(2) 在打开的对话框资源模板中，删除“TODO：在此放置对话框控件。”静态文本控件，将 确定 和 取消 按钮向对话框左边移动一段位置，然后将鼠标移至对话框资源模板右下角的实心深蓝色方块处，拖动鼠标，将对话框资源模板的大小缩小一些(大小调到 230×120px)。

(3) 在对话框资源(模板)的左边添加一个编辑框控件和一个按钮控件，保留其默认属性，并将其布局得整齐一些，如图 3.6 所示。

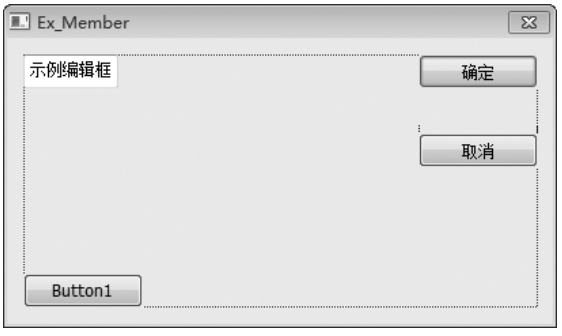

图 3.6　添加编辑框和按钮

(4) 选择“项目”→“类向导”菜单命令或按快捷键 Ctrl＋Shift＋X，弹出“MFC 类向导”对话框。查看“类名”组合框中是否已选择了 CEx_MemberDlg，切换到“成员变量”页面，在“控件变量”列表中双击按钮控件标识符 IDC_BUTTON1，或选定后单击 添加变量(A)... 按钮，弹出“添加成员变量”对话框，如图 3.7 所示。

(5) 在“成员变量名称”框中填好与控件相关联的成员变量“m_btnWnd”，且使“类别”项为 Control，单击 确定 按钮，又回到 MFC 类向导“成员变量”页面中，在“成员变量”列表中出现刚才添加的 CButton 控件对象 m_btnWnd。这样，按钮控件 IDC_BUTTON1 的编程操作就可用与之绑定的对象 m_btnWnd 来操作。

(6) 将 MFC 类向导切换到“命令”页面，为 CEx_MemberDlg 添加 IDC_BUTTON1 的 BN_CLICKED 事件处理程序函数 OnClickedButton1()。单击 确定 按钮，退出“MFC 类向导”对话框，添加下列代码。

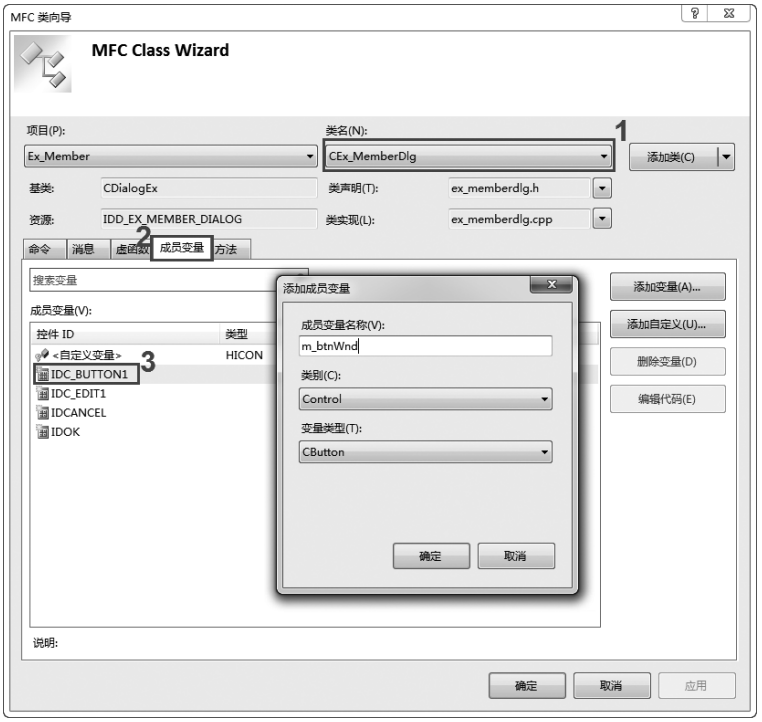

图 3.7 添加成员变量

```
void CEx_MemberDlg::OnClickedButton1()
{
    CString strEdit;                          //定义一个字符串
    CEdit   *pEdit = (CEdit*)GetDlgItem(IDC_EDIT1);
    pEdit->GetWindowText(strEdit);            //获取编辑框中的内容

    strEdit.TrimLeft();
    strEdit.TrimRight();

    if(strEdit.IsEmpty())
        m_btnWnd.SetWindowText("Button1");
    else
        m_btnWnd.SetWindowText(strEdit);
}
```

由于 strEdit 是 CString 类对象,因而可以调用 CString 类的公有成员函数。其中,TrimLeft()和 TrimRight()函数不带参数时分别用来去除字符串最左边或最右边一些白字符(空格符、换行符、Tab 字符等),IsEmpty()用来判断字符串是否为空。

这样,当编辑框内容有除白字符之外的实际字符的字符串时,SetWindowText()便将其内容设定为按钮控件的标题。否则,按钮控件的标题为 Button1。

(7) 编译并运行。当在编辑框中输入“Hello”后,单击 Button1 按钮,按钮的名称就变成了编辑框控件中的内容 Hello。

3.1.4　DDX 和 DDV

对于控件的数据变量，MFC 还提供了独特的 DDX 和 DDV 技术。DDX 将数据成员变量同对话框模板内的控件相连接，这样就使得数据在控件之间很容易传输。而 DDV 用于数据的校验，例如，它能自动校验数据成员变量数值的范围，并发出相应的警告。

一旦某控件与一个数据变量绑定后，就可以使用 CWnd::UpdateData()函数实现控件数据的输入和读取。UpdateData()函数只有一个参数，它为 TRUE 或 FALSE。当在程序中调用 UpdateData()指定参数为 FALSE 时，数据由控件绑定的成员变量向控件传输，而当指定 TRUE 或不带参数时，数据从控件向绑定的成员变量复制。

需要说明的是，数据变量的类型由被绑定的控件类型而定，例如，对于编辑框来说，数值类型可以有 CString、int、UINT、long、DWORD、float、double、BYTE、short、BOOL 等。不过，任何时候传递的数据类型只能是一种。这就是说，一旦指定了数据类型，则在控件与变量传递交换的数据就不能是其他类型，否则无效。

下面来看一个示例，它是在 Ex_Member 项目基础上进行的。

(1) 选择“项目”→“类向导”菜单命令或按快捷键 Ctrl＋Shift＋X，弹出“MFC 类向导”对话框。查看“类名”组合框中是否已选择了 CEx_MemberDlg，切换到“成员变量”页面。

(2) 在“控件变量”列表中双击编辑框控件标识符 IDC_EDIT1，或选定后单击 添加变量(A)... 按钮，弹出“添加成员变量”对话框，将“类别”项选为 Value(值)，再将“变量类型”选为默认的 CString，指定“最大字符数”为 10，在“成员变量名称”框中填好与控件相关联的成员变量 m_strEdit，如图 3.8 所示。

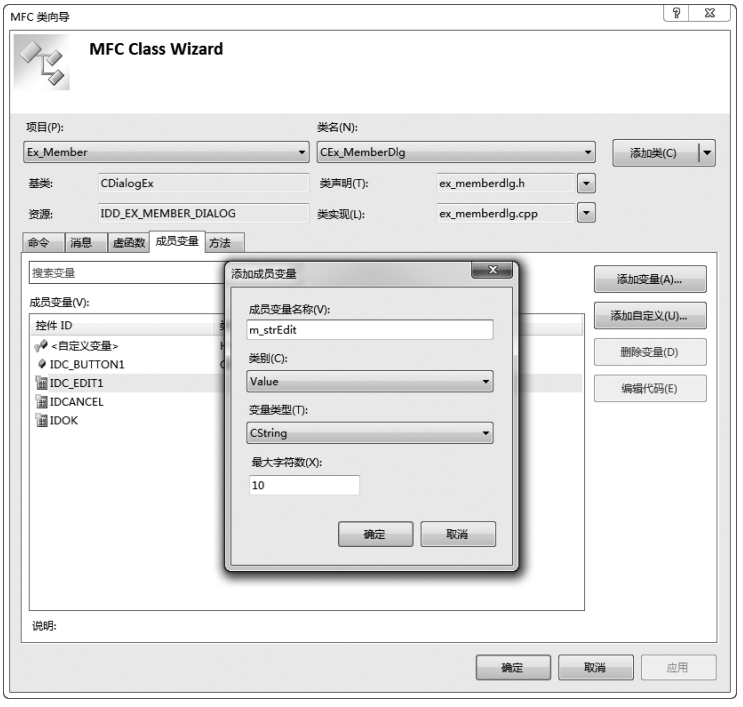

图 3.8　添加控件变量

(3) 单击[确定]按钮,又回到 MFC 类向导“成员变量”页面中。单击[确定]按钮,退出“MFC 类向导”对话框。

(4) 将文档窗口切换到 Ex_MemberDlg.cpp 页面,定位到 CEx_MemberDlg::OnClickedButton1()函数实现代码处,将代码修改如下。

```
void CEx_MemberDlg::OnClickedButton1()
{
    UpdateData();                          //将控件的内容存放到变量中
    //不指定参数,表示使用的是默认参数值 TRUE
    m_strEdit.TrimLeft();
    m_strEdit.TrimRight();
    if(m_strEdit.IsEmpty())
        m_btnWnd.SetWindowText("Button1");
    else
        m_btnWnd.SetWindowText(m_strEdit);
}
```

(5) 编译并运行。当在编辑框中输入“Hello”后,单击[Button1]按钮,OnClickedButton1()函数中的 UpdateData()将编辑框内容保存到 m_strEdit 变量中,从而执行下一条语句后按钮的名称就变成了编辑框控件中的内容 Hello。若输入“Hello DDX/DDV”,则当输入第 10 个字符后,再也输入不进去了,这就是 DDV 的作用。

3.2 静态控件和按钮

静态控件和按钮是 Windows 最基本的控件之一。

3.2.1 静态控件

一个静态控件是用来显示一个字符串、框、矩形、图标、位图或增强的图元文件。它可以被用来作为标签、框或用来分隔其他的控件。一个静态控件一般不接收用户输入,也不产生通知消息。

在工具箱中,属于静态控件的有:静态文本(Aa Static Text)、组框(xyz Group Box)和静态图片(Picture Control)3 种。

3.2.2 按钮

在 Windows 中所用的按钮是用来实现一种开与关的输入,常见的按钮有 3 种类型:按钮、单选按钮、复选框,如图 3.9 所示。

图 3.9 按钮的不同类型

1. 不同按钮的作用

“按钮”通常可以立即产生某个动作，执行某个命令，因此也常称为命令按钮。按钮有两种样式：标准按键按钮(标准按钮)和默认按键按钮(默认按钮)。从外观上来说，默认按键按钮是在标准按键按钮的周围加上一个青色边框(见图 3.9)，这个青色边框表示该按钮已接收到键盘的输入焦点，这样一来，用户只须按 Enter 键就能按下该按钮。一般来说，只把最常用的按键按钮设定为默认按键按钮，具体设定的方法是在按键按钮“属性”窗口中将 Default Button(默认按钮)属性设定为 True。

“单选按钮”(或称“单选框”)的外形是在文本前有一个圆圈，当它被选中时，圆圈中就标上一个青色实心圆点，它可分为一般和自动两种类型。在自动类型中，用户若选中同组按钮中的某个单选按钮，则其余的单选按钮的选中状态就会清除，保证了多个选项始终只有一个被选中。

“复选框”的外形是在文本前有一个空心方框。当它被选中时，方框中就加上一个“✓”标记。通常复选框只有选中和未选中两种状态，若方框中是青色实心方块，则这样的复选框是三态复选框，如图 3.9 所示的 Check2，它表示复选框的选择状态是“不确定”。设定成三态复选框的方法是在复选框“属性”窗口中将 Tri－State(三态)属性设定为 True。

2. 按钮的消息

按钮消息常见的只有两个：BN_CLICKED(单击按钮)和 BN_DOUBLE_CLICKED(双击按钮)。

3. 按钮操作

最常用的按钮操作是设置或获取一个按钮或多个按钮的选中状态。封装按钮的 CButton 类中的成员函数 SetCheck()和 GetCheck()分别用来设置或获取指定按钮的选中状态，其原型如下。

```
void SetCheck(int nCheck);
int GetCheck() const;
```

其中，nCheck()和 GetCheck()函数返回的值可以是：0(不选中)、1(选中)和 2(不确定，仅用于三态按钮)。

若对于同组多个单选按钮的选中状态的设置或获取，则需要使用通用窗口类 CWnd 的成员函数 CheckRadioButton()和 GetCheckedRadioButton()，它们的原型如下。

```
void CheckRadioButton(int nIDFirstButton, int nIDLastButton,
                      int nIDCheckButton);
int GetCheckedRadioButton(int nIDFirstButton, int nIDLastButton);
```

其中，nIDFirstButton 和 nIDLastButton 分别指定同组单选按钮的第一个和最后一个按钮 ID 值，nIDCheckButton 用来指定要设置选中状态的按钮 ID 值，函数 GetCheckedRadioButton()返回被选中的按钮 ID 值。

3.2.3　制作问卷调查对话框示例

问卷调查是日常生活中经常遇到的调查方式。例如，图 3.10 就是一个问卷调查对话

框,它针对“上网”话题提出了三个问题,每个问题都有 4 个选项,除最后一个问题外,其余都是单项选择。本例用到了组框、静态文本、单选按钮、复选框等控件。实现时,需要通过 CheckRadioButton()函数来设置同组单选按钮的最初选中状态,通过 SetCheck()来设置指定复选框的选中状态,然后利用 GetCheckedRadioButton()和 GetCheck()来判断被选中的单选按钮和复选框,并通过 GetDlgItemText()或 GetWindowText()获取选中控件的窗口文本。

图 3.10 “上网问卷调查”对话框

1. 创建并设计对话框

(1) 创建一个默认的基于对话框的应用程序 Ex_Research。系统会自动打开对话框编辑器并显示对话框资源模板。将 stdafx.h 文件最后面内容中的#ifdef _UNICODE 行和最后一个#endif 行删除(注释掉)。

(2) 将文档窗口切换到对话框资源模板页面,单击对话框编辑器工具栏中的“网格切换”按钮,显示对话框资源网格。打开对话框模板的“属性”窗口,将 Caption(标题)属性改为“上网问卷调查”。调整对话框的大小(大小调为 227×179px),删除“TODO:在此放置对话框控件。”静态文本控件,将确定和取消按钮移至正下方。

(3) 从工具箱中选定并向对话框模板中添加组框(Group Box)控件,然后调整其大小和位置(大小调为 216×30px)。右击添加的组框控件,从弹出的快捷菜单中选择“属性”命令,出现其“属性”窗口,将 Caption(标题)属性内容由 Static 改成“你的年龄”(双引号不输入,以下同)。需要说明的是,组框的 Horizontal Alignment(水平排列)属性用来指定 Caption(标题)文本在顶部的对齐方式:默认值、Left(左)、Center(居中)还是 Right(右),当为“默认值”时表示“左”(Left)对齐。

(4) 在组框内添加 4 个单选按钮,默认的 ID 依次为 IDC_RADIO1、IDC_RADIO2、IDC_RADIO3 和 IDC_RADIO4,依次选中这 4 个控件,单击对话框编辑器工具栏的按钮平均它们的水平间距。在其属性窗口中将 ID 属性内容分别改成 IDC_AGE_L18、IDC_AGE_18T27、IDC_AGE_28T38 和 IDC_AGE_M38,然后将其 Caption(标题)属性内容分别改成“<18”“18 — 27”“28 — 38”和“>38”。结果如图 3.11 所示。

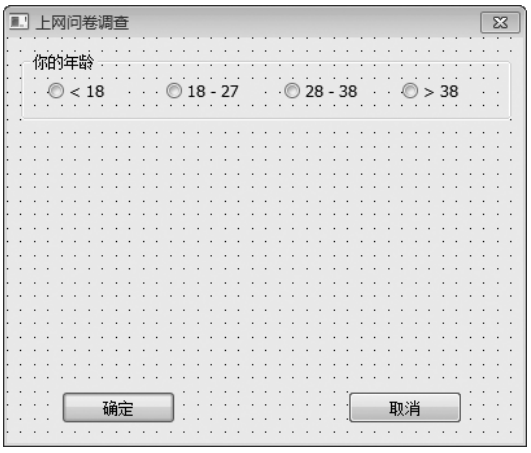

图 3.11　添加的组框和单选按钮

（5）在组框下添加一个静态文本（Aa Static Text），其 Caption（标题）属性设为“你使用的接入方式：”，然后在其下再添加 4 个单选按钮，依次选中这 4 个控件，单击“对话框编辑器”工具栏的 按钮平均它们的水平间距，然后将 Caption（标题）属性分别指定为“FTTL 或 ADSL”“单位 LAN”“拨号 56K”和“其他”，并将相应的 ID 属性依次改成 IDC_CM_FTTL、IDC_CM_LAN、IDC_CM_56K 和 IDC_CM_OTHER。结果如图 3.12 所示。

（6）在对话框模板下方，再添加一个组框控件（大小仍为 216×30px），其 Caption（标题）属性设为“你上网主要是”。然后添加 4 个复选框，依次选中这 4 个控件，单击“对话框编辑器”工具栏的 按钮平均它们的水平间距，然后将其 Caption（标题）属性分别设为“收发邮件”“浏览资料”“聊天游戏”和“其他”，ID 属性依次分别为 IDC_DO_POP、IDC_DO_READ、IDC_DO_GAME 和 IDC_DO_OTHER。结果如图 3.13 所示。

图 3.12　再次添加的单选框

图 3.13　三个问题全部添加后的对话框

（7）单击工具栏中的“测试对话框”按钮。测试后可以发现：顺序添加的这 8 个单选按钮全部变成一组，也就是说，在这组中只有一个单选按钮被选中，这不符合本意。为此，需要分别将上面两个问题中的每一组单选按钮中的第一个单选按钮的 Group（组）属性选定为

True。这样,整个问卷调查的对话框就设计好了,单击工具栏中的按钮测试对话框。如有必要,还可单击对话框编辑器工具栏中的“切换辅助线”按钮,然后将对话框中的控件调整到辅助线以内,并适当对其他控件进行调整。

2. 完善代码

(1) 将项目工作区窗口切换到“类视图”页面,单击 CEx_ResearchDlg 类结点,在“成员”区域中双击 OnInitDialog()函数结点,将会在文档窗口中自动定位到该函数的实现代码处,在此函数中添加下列初始化代码。

```
BOOL CEx_ResearchDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    //…
    CheckRadioButton(IDC_AGE_L18, IDC_AGE_M38, IDC_AGE_18T27);
    CheckRadioButton(IDC_CM_FTTL, IDC_CM_OTHER, IDC_CM_FTTL);
    CButton * pBtn = (CButton * )GetDlgItem(IDC_DO_POP);
    pBtn->SetCheck(1);              //使"收发邮件"复选框选中
    return TRUE;                    //除非将焦点设置到控件,否则返回 TRUE
}
```

(2) 将文档窗口切换到对话框资源模板页面,在对话框模板中,右击 确定 按钮,从弹出的快捷菜单中选择“添加事件处理程序”命令,弹出“事件处理程序向导”对话框,保留默认选项,单击 添加编辑(A) 按钮,在打开的文档窗口中的 CEx_ResearchDlg:: OnBnClickedOk()函数中添加下列代码。

```
void CEx_ResearchDlg::OnBnClickedOk()
{
    //定义两个字符串变量,CString 是操作字符串的 MFC 类
    CString str, strCtrl;
    //获取第一个问题的用户选择
    str = "你的年龄: ";
    UINT nID = GetCheckedRadioButton(IDC_AGE_L18, IDC_AGE_M38);
    GetDlgItemText(nID, strCtrl);          //获取指定控件的标题文本
    str = str + strCtrl;
    //获取第二个问题的用户选择
    str = str + "\n 你使用的接入方式: ";
    nID = GetCheckedRadioButton(IDC_CM_FTTL, IDC_CM_OTHER);
    GetDlgItemText(nID, strCtrl);          //获取指定控件的标题文本
    str = str + strCtrl;
    //获取第三个问题的用户选择
    str = str + "\n 你上网主要是: \n    ";
    UINT nCheckIDs[4]={IDC_DO_POP,IDC_DO_READ,IDC_DO_GAME,IDC_DO_OTHER};
    CButton * pBtn;
```

```
    for (int i=0; i<4; i++)    {
        pBtn = (CButton*)GetDlgItem(nCheckIDs[i]);
        if(pBtn->GetCheck())
        {
            pBtn->GetWindowText(strCtrl);
            str = str + strCtrl;        str = str + " ";
        }
    }
    MessageBox(str);
    CDialogEx::OnOK();
}
```

代码中，GetDlgItemText()是 CWnd 类成员函数，用来获得对话框（或其他窗口）中指定控件的窗口文本。在单选按钮和复选框中，控件的窗口文本就是它们的标题属性内容。该函数有两个参数，第一个参数用来指定控件的标识，第二个参数是返回的窗口文本。后面的函数 GetWindowText()的作用与 GetDlgItemText()相同，也是获取窗口文本。不过，GetWindowText()使用更加广泛，要注意这两个函数在使用上的不同。

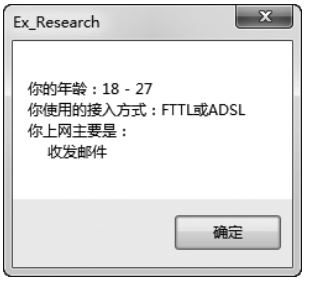

图 3.14　显示选择的内容

（3）编译并运行，出现“上网问卷调查”对话框，当回答问题后，单击 确定 按钮，出现如图 3.14 所示的消息对话框，显示选择的结果内容。

3.3　编辑框和旋转按钮控件

编辑框（ab| Edit Control）是一个让用户从键盘输入和编辑文本的矩形窗口，用户可以通过它很方便地输入各种文本、数字或者口令，也可使用它来编辑和修改简单的文本内容。

当编辑框被激活且具有输入焦点时，就会出现一个闪动的插入符（又可称为文本光标），表明当前插入点的位置。

3.3.1　编辑框的属性和通知消息

在编辑框控件“属性”窗口中，可以方便地设置编辑框的属性。常见的属性含义如表 3.2 所示。

注意：多行编辑框具有简单文本编辑器的常用功能，例如，它可以有滚动条，用户按 Enter 键另起一行以及文本的选定、复制、粘贴等常见操作。而单行编辑框功能较简单，它仅用于单行文本的显示和操作。

当编辑框的文本被修改或者被滚动时，会向其父窗口发送一些消息，如表 3.3 所示。

表 3.2 编辑框常见属性

项 目	说 明
AlignText(排列文本)	文本对齐方式：Left(靠左,默认)、Center(居中)、Right(靠右)
Multiline(多行)	为 True 时,为多行编辑框,否则为单行编辑框
Number(数字)	为 True 时,控件只能输入数字
HorizontalScroll(水平滚动)	水平滚动,仅对多行编辑框有效
Auto HScroll(自动水平滚动)	默认为 True,当用户在行尾输入一个字符时,文本自动向右滚动
VerticalScroll(垂直滚动)	垂直滚动,仅对多行编辑框有效
Auto VScroll(自动垂直滚动)	为 True 时,当在最后一行按 Enter 键时,文本自动向上滚动一行,仅对多行编辑框有效
NoHide Selection(无隐藏选择)	为 True 时,即使编辑框失去焦点,被选择的文本仍然反色显示
OEMConvert(OEM 转换)	为 True 时,实现对特定字符集的字符转换成 OEM 字符集
Password(密码)	为 True 时,输入的字符都将显示为“*”,仅对单行编辑框有效
WantReturn(需要返回)	为 True 时,用户按下 Enter 键,编辑框中就会插入一个回车符
Border(边框)	为 True 时,在控件的周围存在边框
Uppercase(大写)	为 True 时,输入在编辑框中的字符全部转换成大写形式
Lowercase(小写)	为 True 时,输入在编辑框中的字符全部转换成小写形式
Read-Only(只读)	为 True 时,防止用户输入或编辑文本

表 3.3 编辑框通知消息

通 知 消 息	说 明
EN_CHANGE	当编辑框中的文本已被修改,在新的文本显示之后发送此消息
EN_HSCROLL	当编辑框的水平滚动条被使用,在更新显示之前发送此消息
EN_KILLFOCUS	编辑框失去键盘输入焦点时发送此消息
EN_MAXTEXT	文本数目到达了限定值时发送此消息
EN_SETFOCUS	编辑框得到键盘输入焦点时发送此消息
EN_UPDATE	编辑框中的文本已被修改,新的文本显示之前发送此消息
EN_VSCROLL	当编辑框的垂直滚动条被使用,在更新显示之前发送此消息

3.3.2 编辑框的基本操作

由于编辑框的形式多样,用途各异,因此下面针对编辑框的不同用途,分别介绍一些常用操作,以实现一些基本功能。

1. 口令设置

口令设置在编辑框中不同于一般的文本编辑框,用户输入的每个字符都被一个特殊的

字符代替显示，这个特殊的字符称为口令字符。默认的口令字符是“*”，应用程序可以用成员函数 CEdit::SetPasswordChar()来定义自己的口令字符，其函数原型如下。

```
void SetPasswordChar(TCHAR ch);
```

其中，参数 ch 表示设定的口令字符；当 ch=0 时，编辑框内将显示实际字符。

2. 选择文本

当在编辑框中编辑文本时，往往需要选定文本作为整体进行各种编辑操作。除了使用鼠标或键盘来选择文本外，还可通过编程方式来选择文本，这时需要通过调用成员函数 CEdit::SetSel()来实现。与该函数相对应的还有 CEdit::GetSel()和 CEdit::ReplaceSel()，它们分别用来获取编辑框中已选择文本的开始和结束的位置以及替换被选择的文本。

3.设置编辑框的页面边距

设置编辑框的页面边距可以使文本在编辑框中的显示更具满意效果，这在多行编辑框中尤为重要，应用程序可通过调用成员函数 CEdit::SetMargins()来实现，这个函数的原型如下。

```
void SetMargins(UINT nLeft, UINT nRight);
```

其中，参数 nLeft 和 nRight 分别用来指定左、右边距的像素大小。

4. 剪贴板操作

编辑框不仅可以通过 CEdit 类的 Copy()、Paste()和 Cut()成员函数来实现文本的复制、粘贴、剪切等操作，而且还自动支持键盘的快捷操作，其对应的快捷键分别为 Ctrl+C、Ctrl+V 和 Ctrl+X。若调用 CEdit::Undo()函数，则还可撤销最近一次编辑框的操作，再调用一次该函数，则恢复刚才的撤销操作。例如，下面的代码。

```
if(m_Edit.CanUndo())   m_Edit.Undo();
```

5. 获取多行编辑框文本

获取多行编辑框控件的文本可以有以下两种方法。

一种是使用 DDX/DDV，当将编辑框控件所关联的变量类型选定为 CString 后，则不管多行编辑框的文本有多少都可用此变量来保存，从而能简单地解决多行文本的读取。但这种方法不能单独获得多行编辑框中的某一行文本。

另一种方法是使用编辑框 CEdit 类的相关成员函数来获取文本。例如，下面的代码将显示编辑框中第二行的文本内容。

```
char str[100];
if(m_Edit.GetLineCount()>=2)          //判断多行编辑框的文本是否有两行以上
{
    int nChars;
    nChars = m_Edit.LineLength(m_Edit.LineIndex(1));
    //获取第二行文本的字符个数。0 表示第一行，1 表示第二行，以此类推
    //LineIndex 用于将文本行转换成能被 LineLength()识别的索引
```

```
    m_Edit.GetLine(1,str,nChars);          //获取第二行文本
    str[nChars] = '\0';
    MessageBox(str);                       //使用读取的文本
}
```

代码中,由于调用 GetLine()获得某行文本内容时,并不能自动在文本后添加文本的结束符'\0',因此需要首先获得某行文本的字符数,然后设置文本的结束符。

3.3.3 旋转按钮控件

"旋转按钮控件"(Spin Control,也称为上下控件)是一对箭头按钮,可通过单击它们来增加或减小某个值,比如一个滚动位置或显示在相应控件中的一个数字。

一个旋转按钮控件通常是与一个相伴的控件一起使用的,这个控件称为"结伴窗口"。若结伴的控件的"Tab 键顺序"刚好在旋转按钮控件的前面,则这时的旋转按钮控件可以自动定位在它的结伴窗口的旁边,看起来就像一个单一的控件。通常,将一个旋转按钮控件与一个编辑框一起使用,以提示用户进行数字输入。单击向上箭头使当前位置向最大值方向移动,而单击向下箭头使当前位置向最小值的方向移动,如图 3.15 所示。

图 3.15 旋转按钮控件及其结伴窗口

默认时,旋转按钮控件的最小值是 100,最大值是 0。单击向上箭头可减少数值,而单击向下箭头则增加它,这看起来就像颠倒一样,因此还需使用 CSpinButtonCtrl::SetRange()成员函数来改变其最大值和最小值。但在使用时不要忘记在旋转按钮控件"属性"窗口中将 Alignment(排列)属性选定为 Left(靠左)或 Right Align(靠右)(默认值为 Unattached,独立)、同时须将 Auto Buddy(自动结伴)属性设为 True。需要说明的是,若将 Set Buddy Integer(设置结伴整数)属性设为 True,则结伴窗口的数值按整数自动改变。

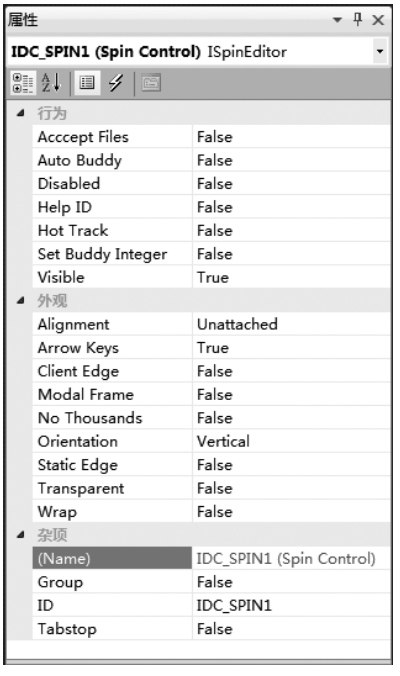

图 3.16 旋转按钮控件"属性"窗口

1. 旋转按钮控件常用的风格

旋转按钮控件有许多属性,它们都可以通过旋转按钮控件"属性"窗口进行设置,如图 3.16 所示,其中常见属性的含义如表 3.4 所示。

2. 旋转按钮控件的基本操作

MFC 的 CSpinButtonCtrl 类提供了旋转按钮控件的各种操作函数,使用它们可以进行基数、范围、位置设置和获取等基本操作。

成员函数 SetBase()是用来设置其基数的,这个基数值决定了伙伴窗口显示的数字是十进制还是十六进制。如果成功,则返回先前的基数值;如果给出的是一个无效的基数,则返回一个非零值。函数的原型如下。