第 5 章　结合实例详解 Windows 标准控件

读者平常玩游戏时都会看到很炫的界面，当用户需要与软件进行交互时就会有很多好看的控件出现。有了这些控件，就像人类间的语言，软件知道用户下一步要干什么，用户也会大致了解到软件最终会实现什么功能。

本章主要涉及到的知识点有：

- Windows 标准控件分为哪些，被用来干嘛。
- 熟悉常用各种控件的创建和应用。
- Active 控件的应用场合。

5.1　简单介绍 Windows 标准控件

本章介绍的 Windows 标准控件应用场合一般是 MFC 的应用程序。它就是用户与软件进行交流的一种接口，读者通过控件输入各种命令，随后程序作出相应的响应或者在界面上关联的控件上输出结果。

因为控件通常出现在对话框上，所以此章所建的项目都是基于对话框的 MFC 应用程序。首先新建名为 Fif_Solu1 的项目，单击项目的工作区中的“资源视图”功能按钮，会出现如图 5.1 所示的视图区。

在这个项目中，控件的建立是需要对话框作为载体的，因此双击虚拟文件夹“Fif_Solu1.rc|Dialog”下的 IDD_FIF_SOLU1_DIALOG 选项，在其右边看到本项目执行的默认载体对话框，如图 5.2 所示。

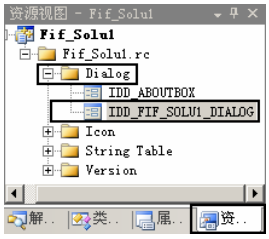

图 5.1　项目资源视图区

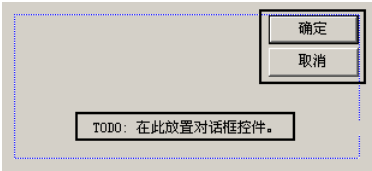

图 5.2　默认载体对话框

项目一经建立，在对话框的工作区中默认有两种控件：按钮（确定、取消）、静态文本框。如果读者要设计的项目不需要这些控件，可以将其删除，重新添加合适的对象。而 VS 的资源控件位于项目工作区的最右边，只要单击“工具箱”按钮，就会出现如图 5.3 所示各种控件。

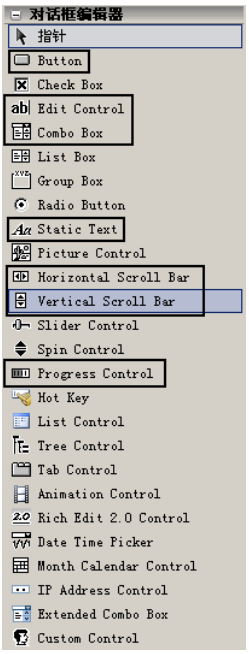

图 5.3　控件工具箱

对于工具箱中所列的各种控件，这里一一给出介绍，下面都是一些编写界面时会用到的控件。

提示：图中的“指针”不是控件，只是功能按钮。

- 按钮（Button）：界面上用于被点击的各种按钮编辑。
- 复选框（Check Box）：用于编辑多个选项同时被选中的状态。
- 编辑框（Edit Control）：用于获取和输出文字、数字等信息。
- 组合框（Combo Box）：一系列选项组合在一起位于下拉列表中，只能看到默认项的内容。
- 列表框（List Box）：与组合框实现的功能一样，但表现形式不同，所有选项的内容都可以看到。
- 集合框（Group Box）：它的作用是将同功能或同系列的控件围起来，使看起来美观、整洁。
- 单选按钮（Radio Button）：与复选框相反，用于只能选其中一项的场合。
- 静态文本框（Static Text）：用于标注内容。
- 图像控件（Picture Control）：用于显示图像。
- 滚动条控件（Horizontal Scroll Bar、Vertical Scroll Bar）：如果图像太大，在一定区域中显示不全，必须用滑块进行拖动，以看到全景。
- 滑块控件（Slider Control）：用于音频编辑中音量的调大调小。
- 旋转控件（Spin Control）：常用于数值的增大或减小。
- 进度条控件（Progress Control）：显示当前系统资源被加载了百分之多少。
- 热键（Hot Key）：让软件的使用者自定义设置热键。

- 列表控件（List Control）：以列表的方式控制其所覆盖的对象。
- 树控件（Tree Control）：以树形结构管理对象。
- 选项卡控件（Tab Control）：有点像属性页的样式，分为不同页，用于管理具有不同功能的对象。
- 动画控件（Animation Control）：用于播放一段动画。
- 复文本控件（Rich Edit 2.0 Control）：支持多行的文本输入/出。
- 日期时间选择器（Data Time Picker）：在一系列日期时间中进行选择。
- 月历控件（Month Calendar Control）：用于选择年月日。
- IP 地址控件（IP Address Control）：输入/出 IP 地址。
- 扩展的组合框（Extended Combo Box）：在组合框的基础上，增加了与图像的关联。

下面内容就对常用控件的创建、使用进行详细介绍。

5.2　学习按钮控件

按钮是最常用的一种简单控件，而它最常用的通告消息就是来自鼠标单击。当用户单击某个按钮，系统就响应其对应的功能。为了使界面变得很美观，可以对按钮进行各种贴图操作。下面就从创建一个原始状态的按钮开始对它进行编辑。

5.2.1　创建一个按钮

在“控件工具箱”中找到“按钮”控件，拖动它到资源视图中的对话框界面上。这样一个原始的按钮就创建好了，它的默认名字是 Button1，如图 5.4 所示。

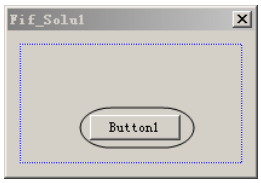

图 5.4　被创建的原始按钮

上面提到的是其中一种方法（纯手工）创建按钮，还有另一种方法——动态创建。动态创建按钮需要在代码中进行，按钮所关联的类是 CButton（可在 MSDN 中查看它的成员函数），本章以基于对话框的 MFC 应用程序 Fif_Solu1 为例。代码 5-1 是动态创建按钮的其中一个添加处，用于对所建按钮的声明。

代码 5-1　动态创建按钮示例：Fif_Solu1Dlg.h

```
01  class CFif_Solu1Dlg : public CDialog
02  {
03  // 构造函数
04  public:
05      CFif_Solu1Dlg(CWnd* pParent = NULL);    //标准构造函数
06  …
```

```
07  public:
08      CButton btn;                                    //创建一个按钮对象
09  …
10  };
```

第 08 行是在创建一个 CButton 对象，在 MFC 中关联按钮的类为 CButton。对象被定义好之后，就要对其进行创建，该类中一个 Create()函数用于编辑按钮的各种状态，请看下面关于 Create()函数的声明。

```
01  virtual BOOL Create(        //创建按钮函数
02     LPCTSTR lpszCaption,     //按钮名字
03     DWORD dwStyle,           //按钮风格
04     const RECT& rect,        //按钮大小
05     CWnd* pParentWnd,        //按钮属于哪个窗口
06     UINT nID                 //按钮的 ID 号
07  );
```

代码 5-2 就是本节所创建的按钮的具体命令。

注意：按钮的属性编辑代码位于 CFif_Solu1Dlg::OnInitDialog()函数中。

代码 5-2　动态创建按钮示例：Fif_Solu1Dlg.cpp

```
01  // Fif_Solu1Dlg.cpp : 实现文件
02
03  #include "stdafx.h"
04  #include "Fif_Solu1.h"
05  #include "Fif_Solu1Dlg.h"
06
07  …
08
09  BOOL CFif_Solu1Dlg::OnInitDialog()
10  {
11      CDialog::OnInitDialog();
12  …
13      // TODO: 在此添加额外的初始化代码
14      btn.Create(L"C 语言", WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
15          CRect(10, 10, 100, 40), this, 0);  //按钮创建
16
17      return TRUE;                          //除非将焦点设置到控件，否则返回 TRUE
18  }
```

第 14～15 行是动态创建按钮的命令，对应上面所列的 Create()函数。可以看出动态创建的按钮名字叫 Button2，它的样式属性被设置了 3 种：WS_CHILD（这个属性一定要有）、WS_VISIBLE（可见）、BS_PUSHBUTTON（可以接收命令消息）。对于其格式的属性，读者可以查看 MSDN 相关文档。第 3 个参数是指按钮的大小，即按钮的左上角点位于对话框的(10,10)处，而其右下角点位于对话框的(100,40)处。

提示：MFC 的坐标原点位于界面的左上角。它的 x 轴正方向是从屏幕左边指向右边，y 轴正方向是从上边指向下边。

第 4 个参数表明此按钮的父窗口是谁，这里给出的是 this，也就是对话框本身。最后一个参数就是按钮的 ID 号，与用手工创建按钮系统给的那个 ID 意思一样。这种方法创建

的按钮如图 5.5 所示。

图 5.5　动态创建按钮

图中 Button1 是手工创建的，按钮“C 语言”是动态创建的。

5.2.2　编辑按钮的属性与消息类型

动态创建按钮已经给“C 语言”按钮添加了对象，但是手工按钮并没有在程序中加入对象。因此还得进行这一步，意思是手工创建可以在资源视图的对话框上直接看到按钮实体。但动态创建的弊端是看不到实体，所以它不直观，编辑起来有些不顺手。因此一般还是采用手工创建。

1．CFif_Solu1Dlg对话框关联按钮变量

（1）给 CFif_Solu1Dlg 对话框添加按钮变量，右击按钮 Button1，在弹出的菜单中选择如图 5.6 所示的“添加变量…”命令。

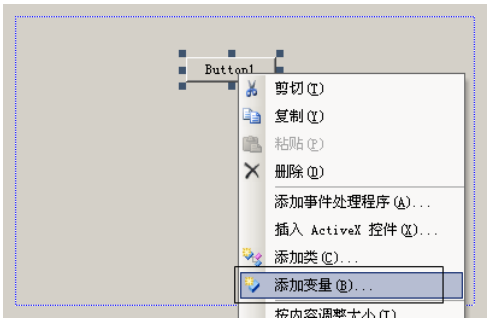

图 5.6　为对话框添加按钮变量

（2）出现如图 5.7 所示的“添加成员变量向导”对话框。

图中的“访问”选项是将此变量设为 public、private、protected 访问方式。因为是为按钮申请变量，所以“变量类型”是 CButton，“变量名”这里设置为 btn1。右面的“控件 ID”是指此按钮的 ID 号，“类型”默认为 Control，那“控件类型”很明显就应该是 BUTTON。下方的“注释”是指添加完此变量后代码中出现的注释内容，同时这个信息也可以不写。如果一切信息确定好后可以单击“完成”按钮。之后会在 Fif_SoluDlg.h 头文件中自动出现下面的代码。

```
01  public:
02      // 按钮变量
03      CButton btn1;
```

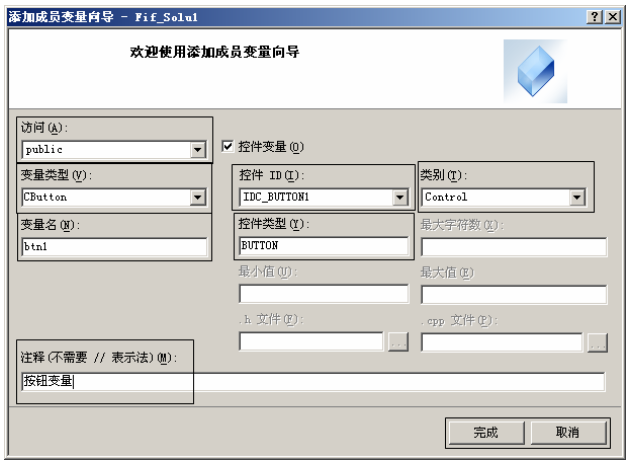

图 5.7　添加成员变量向导

2．编辑按钮的属性

关联变量建好后，就要根据具体的要求编辑按钮的属性，以及要它响应什么样的消息类型。本节对按钮的操作目的是对按钮进行贴图，并且当单击其中一个按钮另一个就会消失，再单击一次又会出现。

用动态创建按钮也有一定的好处，那就是一行代码就可以实现有关按钮的主要属性（名字、ID 号、样式、大小）。但是用手工创建就必须要手工修改（如有需要），前 3 种属性可以参看如图 5.8 所示的属性表修改。而按钮的大小就必须直接作用于按钮。

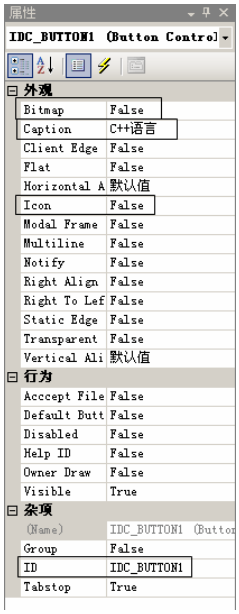

图 5.8　按钮属性列表

按钮属性有 3 大块，外观、行为、杂项。此处将手工创建的 Button1 改名为“C++语言”，下面一一列举这些属性的意义。

提示：它的 ID 号可以保持默认值。

（1）外观

- 位图（Bitmap）：可以为按钮关联位图。
- 标题（Caption）：可修改按钮的标题。
- 边框（Client Edge）：设置按钮是否要带凹陷边缘的边框。
- 平面（Flat）：指定控件的外观是否是二维的。
- 水平对齐（Horizontal Alignment）：设置按钮题目的文本是向左对齐，还是向右对齐或居中。
- 图标（Icon）：设置按钮是否要关联图标。
- 双边框（Modal Frame）：指定按钮是否有双边框。
- 多行（Multiline）：当文本太多，是否以多行显示。
- 通知（Notify）：指定控件是否向父级发送通知。
- 文本右对齐（Right Align Text）：按钮中文本是否要右对齐。
- 阅读顺序（Right To Left Reading Order）：指定是否按照从右到左的阅读顺序。
- 三维边框（Static Edge）：按钮是否具有三维边框。
- 透明背景（Transparent）：按钮的背景是否透明。
- 垂直对齐（Vertical Alignment）：设置按钮题目文本是向上对齐、向下对齐、居中。

（2）行为

- 接受文件（Accept Files）：按钮是否接受拖放文件。
- 默认按钮（Default Button）：设置按钮是否为本对话框的默认命令按钮。
- 禁用（Disabled）：此按钮是否被禁用。
- 帮助 ID（Help ID）：是否给按钮分配基于资源 ID 的帮助 ID。
- 所有者描述（Owner Draw）：指定此按钮是否为所有者描述的按钮。
- 可见（Visible）：设置按钮是否可见。

（3）杂项

- 组（Group）：在一组控件中，设置它是否为第一个控件。
- ID：按钮的 ID 号。
- Tabstop：在程序运行过程中按 Tab 键时，是否可以跳到此按钮。

首先给两个按钮贴图、设置图标，如给标题为“C 语言”的按钮贴图，给标题为“C++语言”的按钮关联图标。

3．给按钮关联资源

（1）给“C++语言”按钮关联自定义的图标。

可以将“C++语言”按钮的 Icon 属性设置为 True，然后加载自定义的图标资源。在资源视图中右击 Fif_Solu1.rc 文件夹，在弹出的菜单中选择“添加资源”命令，会出现如图 5.9 所示的“添加资源”对话框。

图 5.9 “添加资源”对话框

在选项卡的左边选择 Icon，然后单击“导入”按钮，出现如图 5.10 所示的“导入”对话框。

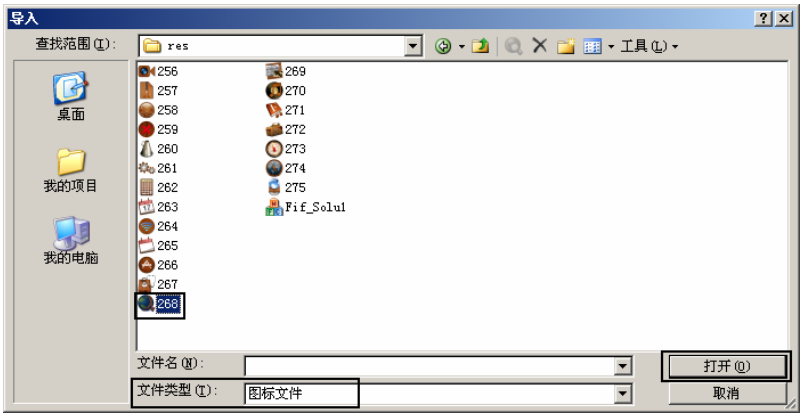

图 5.10 “导入”对话框

首先将“文件类型”文本框设置为“图标文件”，找到图标文件所在的文件夹，选择其中一个图标，然后单击“打开”按钮。这样就在资源视图的 Icon 资源中多了一个名为 IDI_ICON1 的图标资源。

将图标资源加载到项目后，就要在 CFif_Solu1Dlg::OnInitDialog()函数中添加相应的代码，如下：

```
01  HICON hBtnIcon;                                    //按钮对应图标
02  hBtnIcon = AfxGetApp()->LoadIconW(IDI_ICON1);      //加载图标资源
03  btn1.SetIcon(hBtnIcon);                            //按钮关联图标
```

在第 01 行中定义一个 HICON 图标资源，第 02 行是加载图标 IDI_ICON1 到变量 hBtnIcon 上，第 03 行是实现将图标变量的值关联到按钮上。

上面代码是其中一种方法，另外还有两种方法（主要不同在第 02 行），下面一一列出。

方法 1：

```
hBtnIcon = LoadIconW(AfxGetApp()->m_hInstance, (LPCWSTR)IDI_ICON1);
```

方法 2：

```
hBtnIcon = LoadIconW(AfxGetApp()->m_hInstance, MAKEINTRESOURCE(IDI_ICON1));
```

（2）给“C 语言”按钮关联自定义的位图资源。

同与图标的关联原理一样，首先要把按钮的样式加上 BS_BITMAP 属性。因为是为动态创建的按钮关联位图，所以需在代码中设置。

```
btn.Create(L"C 语言", WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON | BS_BITMAP,
    CRect(10, 10, 100, 40), this, 0);//按钮创建
```

如果是手工创建的按钮，只需将选项 Bitmap 设置为 True 即可。

然后在图 5.9 中的左侧选项卡中选择 Bitmap，之后单击“导入”按钮，选择目的位图即可。这个过程跟关联图标类似，所以不再赘述。这样就在资源视图中多了个 Bitmap 的文件夹，如图 5.11 所示。

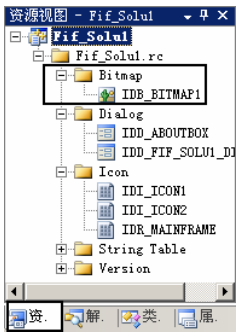

图 5.11　加载的位图资源

按钮关联位图资源 IDB_BITMAP1 的代码如下，它也是位于对话框的 OnInitDialog() 函数中。

```
01  HBITMAP hBitmap;                    //位图对象
02  hBitmap = LoadBitmapW(AfxGetApp()->m_hInstance, (LPCWSTR)IDB_BITMAP1);
                                         //赋值位图
03  btn.SetBitmap(hBitmap);              //与按钮关联
```

接下来讲按钮的消息类型。当单击对话框中的按钮时就会在项目工作区右边出现如图 5.12 所示的“按钮消息类型”属性。

图 5.12　按钮消息类型

注意：要单击图中的第 4 个按钮才可以出现以上画面。

其实这些知识在第 4 章已略有介绍，读者如果忘了可以查找图中所列消息的含义。而

在项目应用中经常会用第 2 个（单击按钮）和第 3 个（双击按钮）消息类型。

5.2.3　响应按钮的消息

本节响应按钮的消息类型选择的是“单击”，具体添加步骤在第 4 章中也有介绍，读者可前去查看。这里要达到的效果是单击其中一个按钮，另一个按钮消失，再单击一次，又出现。下面是响应标题为“C++语言”的按钮（手工创建）的代码。

```
01  // Fif_Solu1Dlg.h : 头文件
02  class CFif_Solu1Dlg : public CDialog
03  {
04  …
05  public:
06      // 按钮变量
07      CButton btn1;
08      int count;                          //单击手工按钮次数
09  public:
10      afx_msg void OnBnClickedButton1();
11  };
12  // Fif_Solu1Dlg.cpp : 源文件
13  CFif_Solu1Dlg::CFif_Solu1Dlg(CWnd* pParent /*=NULL*/)
14      : CDialog(CFif_Solu1Dlg::IDD, pParent)
15  {
16      …
17      count = 0;                          //单击手工按钮次数初始化
18  }
19  void CFif_Solu1Dlg::OnBnClickedButton1()
20  {
21      // TODO: 在此添加控件通知处理程序代码
22      count ++;                           //计算单击次数
23      if((count % 2) == 1)                //如果是单数次单击
24          btn.ShowWindow(false);          //隐藏另一个按钮
25      else                                //如果是双数次单击
26          btn.ShowWindow(true);           //重新显示
27  }
```

第 08 行是程序运行过程中单击按钮的次数的定义，第 17 行是成员变量 count 在构造函数中的初始化。第 19～27 行是具体实现消息响应的函数体。首先要计算单击了多少次此按钮，如第 22 行；如果是单数次单击的按钮就隐藏另一个按钮；如果是双数次单击按钮就重新显示另一个按钮。

上面是对手工创建的按钮消息响应的步骤，但对于动态创建的按钮，处理又会不一样。首先要手工建立消息映射，而消息映射要用到按钮的 ID 号，因此要在头文件 Resource.h 中定义动态创建按钮的 ID 号，如下：

```
#define IDC_BUTTON_CREATE              1017 //动态创建按钮的 ID 号
```

创建好后，还要把 Create()函数中的 ID 号改过来，同时手动写出消息映射的代码。在头文件中进行如下修改：

```
01  // Fif_Solu1Dlg.h : 头文件
02  class CFif_Solu1Dlg : public CDialog
03  {
```

```
04  …
05  public:
06  …
07      int count;                                  //单击手工按钮次数
08      int Active_count;                           //单击动态按钮次数
09  public:
10      afx_msg void OnBnClickedButton1();          //手工按钮消息响应函数
11      afx_msg void OnBnClickedC();                //动态按钮消息响应函数
12  };
```

第 08 行的变量是动态按钮的单击次数，第 11 行是它的消息响应函数。在源文件中添加如下代码：

```
01  …
02  CFif_Solu1Dlg::CFif_Solu1Dlg(CWnd* pParent /*=NULL*/)
03      : CDialog(CFif_Solu1Dlg::IDD, pParent)
04  {
05  …
06      Active_count = 0;                   //单击动态按钮次数初始化
07  }
08  …
09  BEGIN_MESSAGE_MAP(CFif_Solu1Dlg, CDialog)
10      ON_WM_SYSCOMMAND()
11      ON_WM_PAINT()
12      ON_WM_QUERYDRAGICON()
13      //}}AFX_MSG_MAP
14      ON_BN_CLICKED(IDC_BUTTON1, &CFif_Solu1Dlg::OnBnClickedButton1)
15      ON_BN_CLICKED(IDC_BUTTON_CREATE, &CFif_Solu1Dlg::OnBnClickedC)
16  END_MESSAGE_MAP()
17  …
18  BOOL CFif_Solu1Dlg::OnInitDialog()
19  {
20  …
21      btn.Create(L"C 语言", WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON |
BS_BITMAP,
22          CRect(10, 10, 100, 40), this, IDC_BUTTON_CREATE);//按钮创建
23  …
24      return TRUE;                        //除非将焦点设置到控件，否则返回 TRUE
25  }
26  …
27  //动态创建的按钮
28  void CFif_Solu1Dlg::OnBnClickedC()
29  {
30      Active_count ++;                    //计算单击次数
31      if((Active_count % 2) == 1)         //如果是单数次单击
32          btn1.ShowWindow(false);         //隐藏另一个按钮
33      else                                //如果是双数次单击
34          btn1.ShowWindow(true);          //重新显示
35  }
```

第 06 行是对单击次数的初始化，第 15 行是对动态按钮的一种消息映射。同时记得把第 22 行的 ID 号换成在 Resource.h 头文件中为动态按钮定义好的。

注意：第 28 ~ 35 行的消息响应函数与手工按钮的类似，就不再介绍了。

5.2.4　调试及效果图

编辑好后，运行代码，其结果如图 5.13 所示。

单击其中一个按钮，另一个消失；再单击一次，另一个按钮重新显示。但从图中可以看出，位图和图标都没有被完全显示，而是被截了一部分。究其原因是位图和图标没有按控件的比例缩放。这里介绍的方法是手动增大控件的大小，让其适应位图和图标的大小，经调试得到如图 5.14 所示效果。

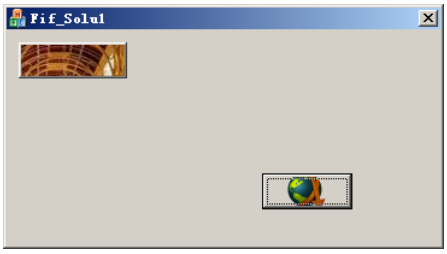

图 5.13　编辑按钮运行结果

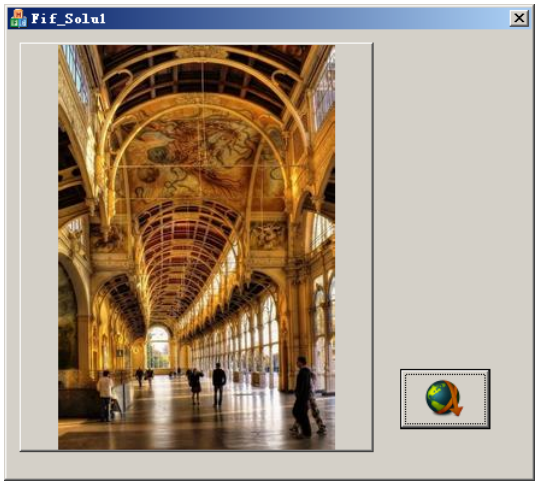

图 5.14　调整效果

5.3　学习静态控件

静态控件是用来标注某块区域的功能作用的。比如说一些功能性的控件（编辑框控件、图像控件、滚动条控件等），没有明显的文字信息来指出它们的用途，所以必须有一个标识信息，而静态控件正好可以实现这个功能。除了标识作用，它还可以接收通告消息，作出响应。下面就对静态控件的这两个功能作详细介绍。

5.3.1　创建一个静态文本框

控件都是可以动态创建的，但是它有一个缺陷就是不够直观。如果想对很多控件排版，用动态创建的控件在资源视图中不能看到。如果想看具体效果必须要运行程序，比较麻烦，所以从本节到下面介绍的控件都是手工拖放形成的。

提示：读者可以通过查看 MSDN 中的 CStatic（静态控件）类了解它的成员函数。

如图 5.15 所示就是新添加的基于对话框的 MFC 应用程序的最初效果图。

图中有一个标题为“TODO：在此放置对话框控件。”的静态文本框。这是所有基于对话框的 MFC 应用程序新建后的控件布局图。在这里可以它为例应用静态控件，读者也

可以从控件工具箱中选择控件，然后拖放到目的对话框处。

图 5.15　最初效果

5.3.2　编辑静态框的属性与消息类型

静态文本框的标题也是可以改变的，请看如图 5.16 所示的静态控件属性。

本节所讲的静态文本框控件是基于这样的一个功能：单击静态控件，就在另一个静态控件中显示一张位图。因此把静态控件的标题改为“显示图片”，因为静态控件默认是不接收通告消息的，所以 Notify 改为 True。另外所有创建的静态控件 ID 号是一样的，都为 IDC_STATIC。如果静态控件不准备进行消息响应，可以不用改；但如果响应消息，这个 ID 号必须要改，以区别其他要响应消息的静态控件。

只有这个 ID 号被改变，在项目工作区的右边“控件事件”选项中才可以出现关联静态控件的消息类型，如图 5.17 所示。

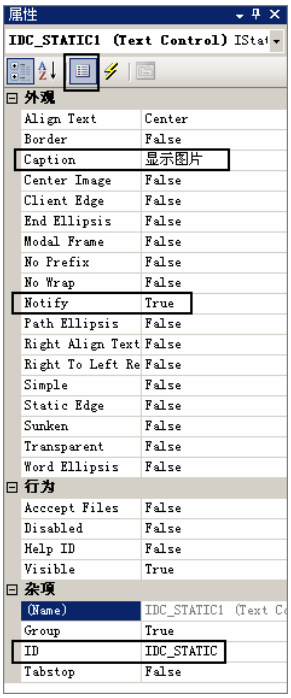

图 5.16　静态控件属性

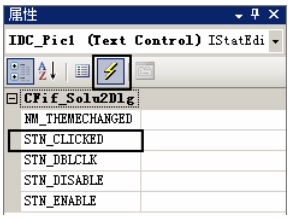

图 5.17　静态控件的消息类型

从图 5.17 中可以看出静态控件有 5 种消息：指示主题更改、单击、双击、被禁用、被

启用。针对于平常项目中多用到单击消息，所以本节就以它为例为静态控件添加消息响应。

注意：如果找不到如图 5.17 所示的消息，可以单击图 5.17 中的第 4 个“控件事件”按钮。

最后再返回静态控件的属性，关于它的其他性质设置与按钮的大同小异，读者可以前去查看。除本节重点介绍的这些属性外，那些都是无技术性的，只关系着静态控件的外观是否好看，因此不多做讲解。

5.3.3　响应控件的消息

1．控件消息响应类型

当静态控件被设置为可以接收通告时，关联控件的消息响应就可以实现了。与按钮相同，可以为静态控件自动添加消息映射宏和消息响应函数。首先在图 5.17 的基础上单击消息 STN_CLICKED 右边的下三角按钮，如图 5.18 所示。

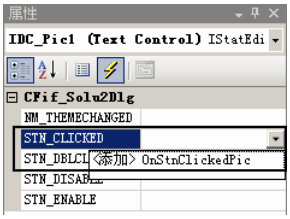

图 5.18　为静态控件添加消息响应

选中“<添加>OnStnClickedPic”选项，就会在代码中的相应位置形成对应功能的代码。此方式添加消息响应函数虽然简单，但它有个缺点，就是名字是既定的，不可改变。如果在创建消息响应函数时想用自定义的函数名，可以右击“显示图片”静态控件，在弹出的菜单中选择“添加事件处理程序”命令，出现如图 5.19 所示的“事件处理程序向导”对话框。

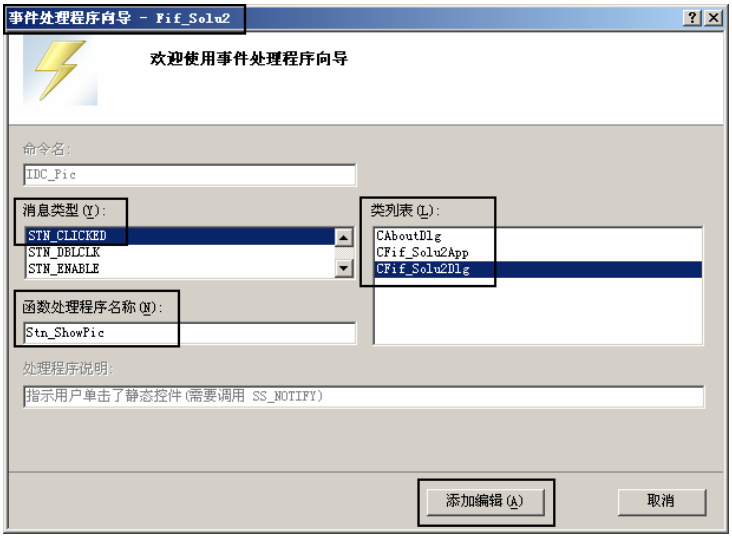

图 5.19　为静态控件添加事件向导

在“消息类型”列表框中选择 STN_CLICKED（单击），“类列表”中选择 CFif_Solu2Dlg

（静态控件关联对话框），“函数处理程序名称”自定义为 Stn_ShowPic，最后单击“添加编辑”按钮。根据显示结果的具体要求，在相应位置处添加功能代码。

提示：因为位图显示在另一个静态控件上，所以要在对话框上再创建一个。

2．控件关联变量类型

同时还要为显示位图的静态控件关联一个对象，它的创建过程跟按钮的一样，读者可参考。按钮的变量类型只有一种 Control，而静态控件关联的变量有两种类型：Control、Value。重点是它们的区别，先看如图 5.20 所示的为静态控件添加变量的向导。

注意：它的 ID 号必须要改为其他名字。

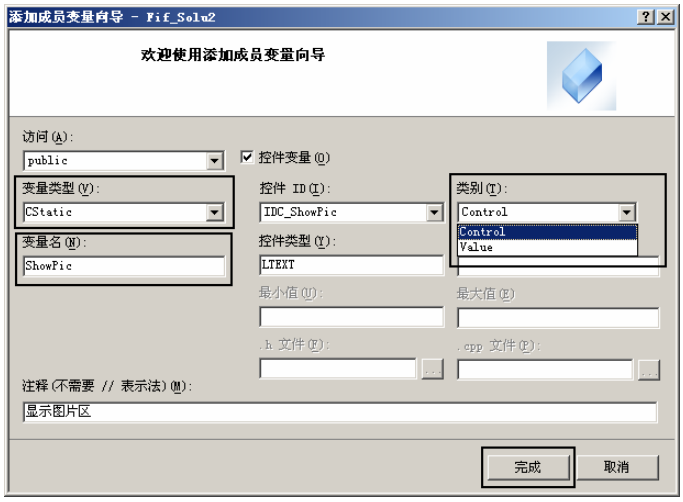

图 5.20　为静态控件关联变量

（1）Control

如果要给静态控件关联 Control 型的变量，只有一种类型：CStatic。这种类型是用来响应 CStatic 类的各种成员函数的。变量名设为 ShowPic，单击“完成”按钮即可。

（2）Value

静态控件还可以关联另一种类型的变量，即 Value。而 Value 这种类型又有很多种类型数据，如图 5.21 所示。

图 5.21　静态控件关联变量为 Value 型

图中最左边的“变量类型”就是 Value 类别下的静态控件可被定义的不同数据类型。

读者可根据项目要求自行选择，这里只是改变文本显示内容，所以选择 CString 用来改变静态控件的显示内容。

上面步骤完成好后，编写具体内容。代码 5-3 为各种所用变量和函数的声明。

代码 5-3　静态控件应用示例：Fif_Solu2Dlg.h

```
01  class CFif_Solu2Dlg : public CDialog
02  {
03  …
04  public:
05      afx_msg void Stn_ShowPic();     //响应静态控件函数
06  public:
07      //显示图片区
08      CStatic ShowPic;                //静态控件类对象
09      HBITMAP hBitmap;                //位图对象
10  public:
11      CString brand;                  //是否显示图片标志量
12      int count;                      //按了几下
13  };
```

第 08 行是关联显示位图的静态控件 CStatic 变量，第 09 行是位图对象的定义，第 11 行是关联文本的静态控件 CString 变量，第 12 行用来计算到底单击了多少次静态控件。代码 5-4 是成员变量的应用和成员函数的具体定义。

代码 5-4　静态控件应用示例：Fif_Solu2Dlg.cpp

```
01  …
02  CFif_Solu2Dlg::CFif_Solu2Dlg(CWnd* pParent /*=NULL*/)
03      : CDialog(CFif_Solu2Dlg::IDD, pParent)
04      , brand(_T(""))
05  {
06      m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
07      brand = _T("显示图片");                  //静态控件 CString 变量初始化
08      count = 0;                               //标志量初始化
09  }
10
11  void CFif_Solu2Dlg::DoDataExchange(CDataExchange* pDX)
12  {
13      CDialog::DoDataExchange(pDX);
14      DDX_Control(pDX, IDC_ShowPic, ShowPic);    //静态变量与控件数据关联
15      DDX_Text(pDX, IDC_Pic, brand);             //标志量与控件关联
16  }
17
18  BEGIN_MESSAGE_MAP(CFif_Solu2Dlg, CDialog)
19      ON_WM_SYSCOMMAND()
20      ON_WM_PAINT()
21      ON_WM_QUERYDRAGICON()
22      //}}AFX_MSG_MAP
23      ON_STN_CLICKED(IDC_Pic, &CFif_Solu2Dlg::Stn_ShowPic)//消息映射宏
24  END_MESSAGE_MAP()
25
26  BOOL CFif_Solu2Dlg::OnInitDialog()
27  {
28  …
29      // TODO: 在此添加额外的初始化代码
30      hBitmap = LoadBitmapW(AfxGetApp()->m_hInstance,
```

```
(LPCWSTR)IDB_BITMAP1);//赋 31    值位图资源
32      ShowPic.ModifyStyle(0, SS_BITMAP | SS_CENTERIMAGE);//这句非常重要,
33  让静态控件可以关联图像
34      return TRUE;  // 除非将焦点设置到控件，否则返回 TRUE
35  }
36  …
37  void CFif_Solu2Dlg::Stn_ShowPic()
38  {
39      // TODO: 在此添加控件通知处理程序代码
40      count ++;                                   //计数
41      if((count % 2) == 0)                        //如果是双数次单击静态控件
42      {
43          brand = _T("显示图片");
44          ShowPic.ShowWindow(false);              //隐藏控件
45      }
46      else                                        //单数次
47      {
48          brand = _T("隐藏图片");
49          ShowPic.SetBitmap(hBitmap);             //与静态文本框关联
50          ShowPic.ShowWindow(true);               //显示控件
51      }
52      GetDlgItem(IDC_Pic)->SetWindowText(brand);
                                                    //将当前控件的变量值实时显示
53  }
```

第 07～08 行是对变量的初始化，第 14～15 行是进行数据实时交互，将静态控件与关联的变量捆绑起来。第 23 行是控件响应函数的消息映射宏，第 52 行是用 SetWindowText() 函数改变静态控件的内容。

提示：最重要的是第 32 行，没有这一行代码，控件上是不会显示出位图的。它的作用是向控件的样式中添加显示位图这一项功能并且使位图居中。

5.3.4　调试并展示效果图

编辑完代码，经调试得如图 5.22 所示的程序刚被运行时的效果。“显示图片”是一个静态控件，但它的作用相当于按钮，因为可以响应消息。当被单击时，位图会被显示在“位图显示区”中，如图 5.23 所示，同时它自己的文本也会改变为“隐藏图片”。

图 5.22　静态控件被双数次单击

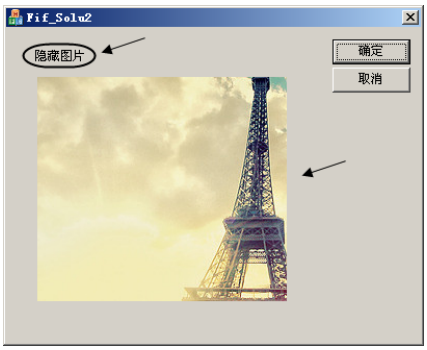

图 5.23　静态控件被单数次单击

再一次单击“隐藏图片”控件时，位图消失，控件显示内容变为“显示图片”。这样循环往复，单击“确定”按钮退出系统。

5.4　学习编辑框控件

编辑框控件提供了接收用户输入和编辑文字的功能。它所关联的类是 CEdit，CEdit 类有很多成员函数，本节对它的学习只针对其中一部分函数，其余的读者可查找 C++字典——MSDN。在这里编辑框要实现的功能是完成数学中的四则运算。

5.4.1　创建一个编辑框

首先对实现四则运算的程序界面进行排版，效果如图 5.24 所示。

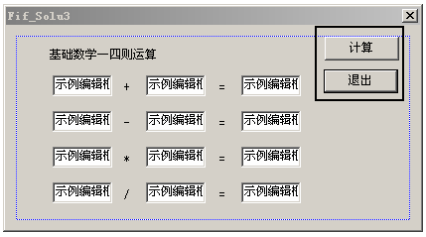

图 5.24　编辑框应用排版图

图中的标题“基础数学—四则运算”是用静态文本框做的，剩下的白色框为编辑框控件。如果编辑框中填好数字后，可以单击“计算”按钮显示结果。在控件的排版方面，工具栏中有一系列功能按钮，如图 5.25 所示。

图 5.25　排版控件的功能按钮

下面对这些按钮功能从左到右一一作介绍：

- 测试对话框：在不运行程序的前提下，可以方便地看到对话框在运行时的样子，也可以提前得知排版整齐与否。
- 左对齐：所选中的两个或两个以上的控件向左对齐。
- 右对齐：所选中的两个或两个以上的控件向右对齐。
- 顶部对齐：所选中的两个或两个以上的控件顶部对齐。
- 底部对齐：所选中的两个或两个以上的控件底部对齐。

注意：这几对齐方式的参考标准都是最后一个被选中的控件。

- 垂直：被选中的控件离对话框的上边框和下边框的距离相等（如果只有一个控件，参考对象就是它；如果是两个或两个以上，参考对象就是整体的外边缘）。
- 水平：被选中的控件离对话框的左边框和右边框的距离相等（如果只有一个控件，参考对象就是它；如果是两个或两个以上，参考对象就是整体的外边缘）。

- 横向：被选中控件的间隔在横向上相等。
- 纵向：被选中控件的间隔在纵向上相等。
- 使宽度相同：被选中控件的尺寸在宽度上相等。
- 使高度相同：被选中控件的尺寸在高度上相等。
- 使大小相同：被选中控件的尺寸在宽度和高度上都相等。

注意：尺寸相同的参考标准都是最后一个被选中的控件。

5.4.2　编辑控件的属性与消息类型

1．编辑控件的属性

对于编辑框控件的属性也分为 3 种：外观、行为、杂项。本节先介绍“行为”和“杂项”两种属性（“外观”属于一看就懂级别）。“杂项”中最重要的就是控件的 ID 号，并且不同编辑框控件的 ID 号不一样，就像按钮控件，都是默认可以接收通告消息的。“行为”中有 3 个最重要——Password，Read Only，Want Return。

- Password。当它被设为 True，输入到编辑框中的字符都会以星号（*）形式显示出来。

提示：这个属性用于编辑密码的场合，此时需要用星号代替。

- Read Only。当它为 True 时，编辑框中字符只能被读取，不可以写入。也就是只有初始值，不能被修改，同时编辑框会变灰。
- Want Return。当它为 True 时，编辑框控件可以接收 Enter 键命令。因为这个程序只是数学的四则运算，所以编辑框中只允许输入数字。正好“外观”中有一个 Number 属性，它若为 True，只允许在编辑框中输入数字。

2．编辑控件的消息类型

编辑框的消息也有很多种，如图 5.26 所示。它的具体含义在第 4 章中也有介绍，读者可前去查看，另外这些消息有些是需要跟其他控件配合使用（如 EN_HSCROLL 等）。

控件的这些消息的意义可以这样理解，比如 EN_SETFOCUS 消息，当关联此消息的编辑框控件已获得输入焦点时，会作出怎样的响应。而这个具体的响应就是由函数体中的代码来完成。

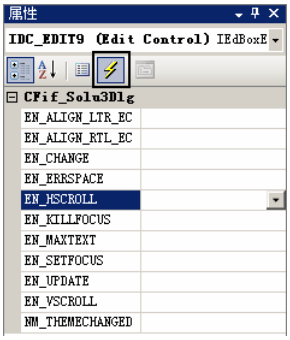

图 5.26　编辑框消息类型

5.4.3　响应控件的消息

本节中对于编辑框的应用是数学的四则运算，需要往编辑框中输入数字，当单击“计算”按钮后，会在另一些编辑框中输出结果。另外在程序开始运行时，光标会定位到其中一个编辑框中，按 Enter 键，光标会跳到下一个目标编辑框中。请看代码 5-5 是在头文件 Fif_Solu3Dlg.h 中所添加的变量和函数。

代码 5-5　编辑框控件应用示例：Fif_Solu3Dlg.h

```
01  class CFif_Solu3Dlg : public CDialog
02  {
03  …
04  public:
05      double num[12];                                  //存放从编辑框中获取的数据
06      afx_msg void OnBnClickedCal();                   //响应“计算”按钮
07  …
08      virtual BOOL PreTranslateMessage(MSG* pMsg);    //提前筛选消息
09  …
10  };
```

第 05 行用于存储从编辑框获取的数据，第 06 行是响应“计算”按钮的消息函数，第 08 行是一个虚函数，是从基类继承来，可以被重写。在本节用于实现响应 Enter 键。代码 5-6 是函数的具体定义内容。

代码 5-6　编辑框控件应用示例：Fif_Solu3Dlg.cpp

```
01  CFif_Solu3Dlg::CFif_Solu3Dlg(CWnd* pParent /*=NULL*/)
02      : CDialog(CFif_Solu3Dlg::IDD, pParent)
03  {
04      m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
05      for(int i = 0; i< 12; i++)
06      {
07          num[i] = 1.0;                                //编辑框变量初始化
08      }
09  }
10  …
11  BOOL CFif_Solu3Dlg::OnInitDialog()
12  {
13  …
14      // TODO: 在此添加额外的初始化代码
15      SetDlgItemText(IDC_EDIT1, _T("1"));              //让编辑框显示初始内容
16      SetDlgItemText(IDC_EDIT2, _T("1"));
17      SetDlgItemText(IDC_EDIT3, _T("1"));
18      SetDlgItemText(IDC_EDIT4, _T("1"));
19      SetDlgItemText(IDC_EDIT5, _T("1"));
20      SetDlgItemText(IDC_EDIT6, _T("1"));
21      SetDlgItemText(IDC_EDIT7, _T("1"));
22      SetDlgItemText(IDC_EDIT8, _T("1"));
23      SetDlgItemText(IDC_EDIT9, _T("1"));
24      SetDlgItemText(IDC_EDIT10, _T("1"));
25      SetDlgItemText(IDC_EDIT11, _T("1"));
26      SetDlgItemText(IDC_EDIT12, _T("1"));
```

```
27
28      GetNextDlgTabItem(GetFocus())->SetFocus(); //初次光标位置,按 Tab 顺序
29      return TRUE;  // 除非将焦点设置到控件，否则返回 TRUE
30  }
31  …
32  void CFif_Solu3Dlg::OnBnClickedCal()
33  {
34      // TODO: 在此添加控件通知处理程序代码
35      CString str1;                                       //临时变量 1
36      CString str2;                                       //临时变量 2
37      CString str3;                                       //临时变量 3
38  //-------------------------加法-----------------------------------
39      GetDlgItem(IDC_EDIT1)->GetWindowTextW(str1);   //获取编辑框中的文本
40      GetDlgItem(IDC_EDIT2)->GetWindowTextW(str2);   //同上
41
42      num[0] = _wtoi(str1);           //将 CString 型转换为相应的 double 型
43      num[1] = _wtoi(str2);           //同上
44      num[2] = num[0]+num[1];         //相加
45
46      str3.Format(_T("%.7f"), num[2]);//将 double 型转换为相应的 CString 型
47
48      GetDlgItem(IDC_EDIT3)->SetWindowTextW(str3);   //显示计算结果
49  //-------------------------减法-----------------------------------
50      GetDlgItem(IDC_EDIT4)->GetWindowTextW(str1);
51      GetDlgItem(IDC_EDIT7)->GetWindowTextW(str2);
52
53      num[3] = _wtoi(str1);           //将 CString 型转换为相应的 double 型
54      num[4] = _wtoi(str2);
55      num[5] = num[3]-num[4];
56
57      str3.Format(_T("%.7f"), num[5]);//将 double 型转换为相应的 CString 型
58
59      GetDlgItem(IDC_EDIT10)->SetWindowTextW(str3);
60  //-------------------------乘法-----------------------------------
61      GetDlgItem(IDC_EDIT5)->GetWindowTextW(str1);
62      GetDlgItem(IDC_EDIT8)->GetWindowTextW(str2);
63
64      num[6] = _wtoi(str1);           //将 CString 型转换为相应的 double 型
65      num[7] = _wtoi(str2);
66      num[8] = num[6]*num[7];
67
68      str3.Format(_T("%.7f"), num[8]);//将 double 型转换为相应的 CString 型
69
70      GetDlgItem(IDC_EDIT11)->SetWindowTextW(str3);
71  //-------------------------除法-----------------------------------
72      GetDlgItem(IDC_EDIT6)->GetWindowTextW(str1);
73      GetDlgItem(IDC_EDIT9)->GetWindowTextW(str2);
74
75      num[9] = _wtoi(str1);           //将 CString 型转换为相应的 double 型
76      num[10] = _wtoi(str2);
77      if(num[10] == 0.0)              //判断除数是否为 0
78      {
79          MessageBox(_T("除数不可为零,请重新输入!"), _T("警告"), 0);
80      }
81      num[11] = num[9]/num[10];
82
83      str3.Format(_T("%.7f"), num[11]);//将 double 型转换为相应的 CString 型
```

```
84
85      SetDlgItemText(IDC_EDIT12, str3);  //在编辑框上显示计算结果
86  }
87  …
88  BOOL CFif_Solu3Dlg::PreTranslateMessage(MSG* pMsg)
89  {
90      // TODO: 在此添加专用代码和/或调用基类
91      if(pMsg->message == WM_KEYDOWN && pMsg->wParam == VK_RETURN)
                                              //当按下键
92  盘上 Enter 键
93          pMsg->wParam = VK_TAB;            //响应 Enter 键(按 Tab 顺序)
94      return CDialog::PreTranslateMessage(pMsg);
95  }
```

第 07 行是对存放数据的数组进行初始化，第 15～26 行是程序刚运行时编辑框上显示的内容。第 28 行是用来设置光标的位置，它按照 Tab 顺序依次高亮显示编辑框内容。第 39 行是获取编辑框中的内容，然后存在 CString 变量中。第 42 行是将 CString 型数据变换为 double 型数据，第 46 行是 double 型数据转换为 CString 数据。第 48 行是将变换好的结果 CString 数据显示在编辑框中。

因为除法运算中除数不可以为 0，所以在第 77 行做出了判断。如果除数为 0，则弹出一个警告消息框。第 85 行是以另一种形式将文本内容显示在编辑框中。

提示：MFC 中默认按 Tab 键可以移动当前输入光标，现在是把 Tab 键可实现的功能赋给 Enter 键，让 Enter 键也可有此功能。

5.4.4　调试并展示效果图

上面总是说到 Tab 顺序，先来看下这个顺序到底指的是什么。如图 5.27 所示就是在对话框中所有控件的 Tab 编号。

当程序运行后，光标先定位在编号为 1 的编辑框处。按一下 Enter 键，光标进入编号为 2 的编辑框处，这样一直到编号为 8 的编辑框。再按 Enter，程序会响应“计算”按钮。如果一直按 Enter 键，会一直往后推，直到响应到最后一个编号的控件上。然后再循环到编号为 1 的控件上，运行效果如图 5.28 所示。

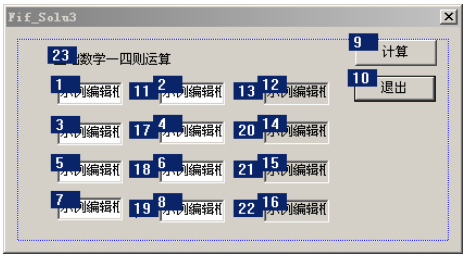

图 5.27　控件的 Tab 顺序图

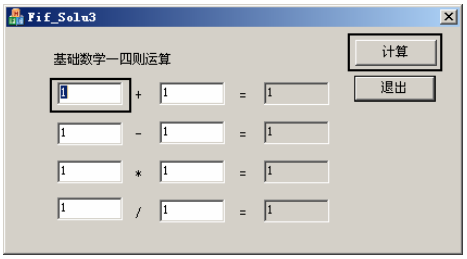

图 5.28　程序运行结果图

5.5　学习组合框控件

组合框控件是一个具有很多可选项的集合，比如某人选择星期几去上班，那它的可选

范围就是星期一到星期天。组合框就是用来管理这些可选项的，应用它的好处就是不必将所有选项都罗列到界面，因此界面也就显得很整洁。

5.5.1　创建一个组合框

本节还是以一个例子为主线介绍组合框的应用流程。要建立两个组合框，每个组合框中都有 3 个备选项。当第一个组合框中的选项为 A，则第二个组合框中的选项为甲；如果为 B，另一个就必须为对应的乙；如果为 C，则另一个为丙。

在资源视图中的对话框编辑区创建两个组合框之后，默认状态下它们的 ID 号各不相同，分别为：IDC_COMBO1 和 IDC_COMBO2。众所周知，当新变量被定义后就应该被初始化；同理，控件也是这样。因此要在基于对话框的 MFC 应用程序中的 OnInitlDialog()虚函数中对这两个组合框控件进行初始化，即分别赋值 3 个状态，如图 5.29 所示。

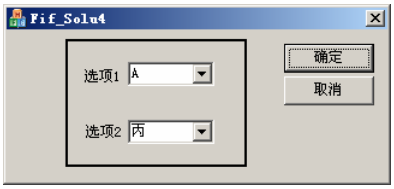

图 5.29　组合框控件

图中的“选项 1”和“选项 2”是静态文本控件，位于它们右边的就是组合框控件。代码 5-7 是对这两个组合框控件的初始设置。

代码 5-7　组合框控件初始化：Fif_Solu4Dlg.cpp

```
01  BOOL CFif_Solu4Dlg::OnInitDialog()
02  {
03      CDialog::OnInitDialog();
04  …
05      // TODO: 在此添加额外的初始化代码
06      ((CComboBox*)GetDlgItem(IDC_COMBO1))->AddString(_T("A"));
                                                          //选项 1
07      ((CComboBox*)GetDlgItem(IDC_COMBO1))->AddString(_T("B"));
                                                          //选项 2
08      ((CComboBox*)GetDlgItem(IDC_COMBO1))->AddString(_T("C"));
                                                          //选项 3
09
10      ((CComboBox*)GetDlgItem(IDC_COMBO1))->SetCurSel(0);    //光标定位
11
12      ((CComboBox*)GetDlgItem(IDC_COMBO2))->AddString(_T("甲")); //同上
13      ((CComboBox*)GetDlgItem(IDC_COMBO2))->AddString(_T("乙"));
14      ((CComboBox*)GetDlgItem(IDC_COMBO2))->AddString(_T("丙"));
15
16      ((CComboBox*)GetDlgItem(IDC_COMBO2))->SetCurSel(0);        //同上
17
18      return TRUE;  // 除非将焦点设置到控件，否则返回 TRUE
19  }
```

代码第 06～10 行是对第一个组合框的设置；第 12～16 行是对第二个组合框的设置。

第 06～08 行是给组合框中设置 3 个备选项——A、B、C；第 10 行设置的效果是：当程序运行刚开始时，光标会定位在哪个备选项上。

提示：关联组合框控件的元素中有个索引值（int 型）变量。比如说这里的 IDC_COMBO1 组合框有 3 个选项，第一个加入的选项 A，它的索引值就是 0；第二个加入的选项 B，它的索引值是 1；第三个加入的选项 C，它的索引值是 2。按照这个规律，其他组合框控件中选项的索引值具体数据都是这样给定的。

这里 IDC_COMBO1 控件初始会定位在 A 选项上。此外组合框控件也可以动态创建，与组合框关联的类是 CComboBox。但是这里有个疑问，ID 号为 IDC_COMBO2 的组合框控件的选项输入顺序是“甲”、“乙”、“丙”，同时设置的光标初始位置是“甲”。可是从图 5.29 中可以看出它的初始值是“丙”。所以这里还需要对组合框的属性进行设置，读者请看 5.5.2 节。

5.5.2　编辑控件的属性与消息类型

如图 5.30 所示是组合框控件的所有属性，这里只列出了组合框的其中一部分属性。另外也只重点介绍两个属性，因为其余的都跟先前控件大同小异，读者可自由想象并前往查看。

在“行为”属性系列中有个 Sort 性质。当 Sort 设置为 True 时，系统会对输入组合框中的字符串自动排序（字符串第一个字符按 A—Z 顺序排列）；反之，如果为 False，就会按照输入顺序来显示。

接下来介绍另一个属性——Type，顾名思义就是描述组合框的外表。它有 3 个选择——Simple，Dropdown，Drop List。设置为 Dropdown，用户可以在组合框中输入字符串，但是如果设置为 Drop List，是不可以输入外部字符串的。

组合框控件的消息有很多，如图 5.31 所示。

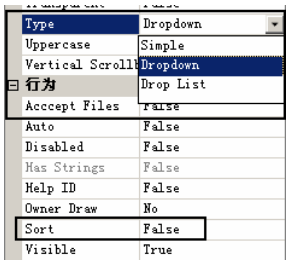

图 5.30　组合框的属性

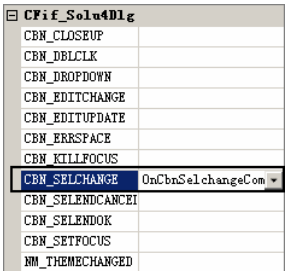

图 5.31　组合框消息

图中显示此项目为组合框控件添加了 CBS_SELCHANGE 的消息，也就是说当此控件中的内容被改变，就会执行什么指令。而对于其他消息，读者可根据自己所做项目的要求来添加不同的消息响应，这里不多做说明。

注意：在项目工作区中控件消息的正下方会有对应消息的文字说明。

5.5.3　响应控件的消息

接 5.5.2 节的内容，当第一个组合框的内容发生改变，第二个组合框会按照选择规律显示相应内容，代码 5-8 是控件的消息响应命令。

代码 5-8　组合框控件消息响应：Fif_Solu4Dlg.cpp

```
01  void CFif_Solu4Dlg::OnCbnSelchangeCombo1()
02  {
03      // TODO: 在此添加控件通知处理程序代码
04      int index;                              //组合框索引号
05      CString str;                            //临时字符串变量
06
07      index = ((CComboBox*)GetDlgItem(IDC_COMBO1))->GetCurSel();
08                                              //获取当前状态的索引值
09      if(index == 0)                          //如果选第 1 个
10          ((CComboBox*)GetDlgItem(IDC_COMBO2))->SetCurSel(0);
11                                              //第二个组合框的光标自动跳到第 1 个
12      else if(index == 1)                     //第 2 个
13          ((CComboBox*)GetDlgItem(IDC_COMBO2))->SetCurSel(1);
                                                //第 2 个
14      else if(index == 2)                     //第 3 个
15          ((CComboBox*)GetDlgItem(IDC_COMBO2))->SetCurSel(2);
                                                //第 3 个
16      else                                    //如果是其他
17      {}                                      //不做任何动作
18
19      ((CComboBox*)GetDlgItem(IDC_COMBO1))->GetLBText(index, str);
                                                //获取内容
20      MessageBox(str, _T("说明"), 0);         //当前第一个组合框被选择的内容
21  }
```

第 07 行所示的 GetCurSel()也是组合框关联类 CComboBox 的其中一个成员函数，它的作用是获取当前选择的索引值。第 19 行是用另一个成员函数 GetLBText()获取组合框的内容。这个函数有两个参数，从参数的输入输出类型讲，第一个参数是输入，第二个是输出参数。获取的字符串变量被存放到临时变量 str 中，在第 20 行中利用 MessageBox()将其以说明的方式显示。而第 09～17 行也就是这个消息响应起最主要的作用的代码，它的具体意思就不赘述了，在学习完第 2 章后，读者应该有能力看懂（同时也检验下自己的学习效率）。

5.5.4　调试并展示效果图

上面代码经调试后，得到其效果如图 5.32 所示。

可以看出，选项 1 的初始选择是 C。当用户选择 B 时，选项 2 的光标会自动跳到其对应选项“乙”上，同时会弹出一个“说明”对话框，以显示当前选项 1 选择的是 B。当单击“说明”对话框上的“确定”按钮后，就会出现如图 5.33 所示的最终效果图。

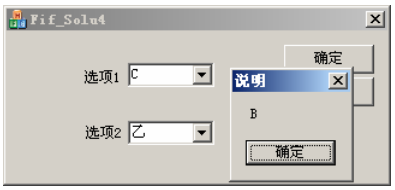

图 5.32　组合框应用效果 1

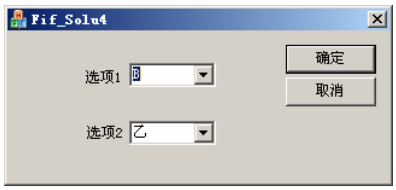

图 5.33　组合框应用效果 2

5.6　学习进度条、滑块控件

这一节的例子结合进度条和滑块两种控件来介绍。进度条一般用于显示文件的下载状态、软件的安装过程（百分比形式）等这些需要一段过程的事件。滑块一般用于调节所播放视频、音频的音量大小。

5.6.1　创建进度条

添加一个新的基于对话框 MFC 应用程序，名为 Fif_Solu5，把它设为启动项。从工具箱中选择进度条控件拖放到对话框资源上，此外还可以动态创建进度条控件，它所关联的类为 CProgressCtrl。

对于控件，可以将其想象为最基本的数据——变量。变量被定义后，就要先初始化；同理，进度条控件也需要进行初始化（在 OnInitlDialog()函数中添加代码）。

首先为进度条添加变量 progress，让此变量关联到被创建的进度条上。然后添加代码设置它的范围（以百分比来表示进度）为 0～100，进度条控制点的初始所在位置为 1，背景颜色设为绿，具体代码如下：

```
01  progress.SetRange(0, 100);                    //设置进度条的范围(0～100)
02  progress.SetPos(1);                           //初始位置
03  progress.SetBkColor(RGB(255, 255, 255));      //为进度条设置背景颜色
```

运行结果如图 5.34 所示。

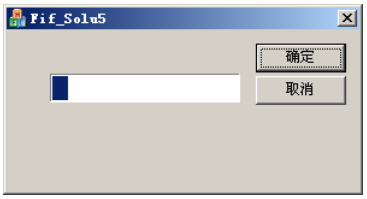

图 5.34　进度条初始效果

5.6.2　使用进度条

进度条的属性中有一项为 Smooth，即是否要平滑填充进度控件。如果为 True，就是

平滑填充，它的效果如图 5.35 所示；如果为 False，它的效果如图 5.36 所示，系统默认为 False。

图 5.35　平滑填充进度条

图 5.36　不是平滑填充进度条

此外进度条的消息也有很多，如图 5.37 所示就是关联进度条的 4 种事件。

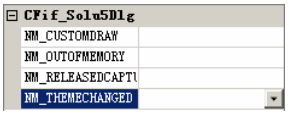

图 5.37　进度条消息

- NM_CUSTOMDRAW：将自定义绘图操作通知给父级。
- NM_OUTOFMEMORY：指示控件由于内存不足而未能完成某项操作。
- NM_RELEASEDCAPTURE：将鼠标捕获释放通知父级。
- NM_THEMECHANGED：指示主题已更改。

而这些消息在一般场合中不会用到，这里为了让进度条可以动起来，给对话框类添加了一个消息函数 OnTimer()。在类视图中，单击项目 Fif_Solu5 下的 CFif_Solu5Dlg 类，可以在右侧看到这个对话框类的属性，单击上面的“消息”按钮，在它的列表中找到 WM_TIMER 消息添加即可。代码 5-9 是 OnTimer()函数的具体内容。

代码 5-9　进度条的定时器消息：Fif_Solu5Dlg.cpp

```
01  void CFif_Solu5Dlg::OnTimer(UINT_PTR nIDEvent)
02  {
03      // TODO: 在此添加消息处理程序代码和/或调用默认值
04      int pos;                                    //位置变量
05
06      progress.SetStep((upper - lower)/10);  //设置进度条运行步长
07      progress.StepIt();                          //更新进度条当前的位置
08      pos = progress.GetPos();                    //获取进度条当前的位置
09      if(pos == upper)                            //如果到达终点处
10          KillTimer(1);                           //关闭时钟
11
12      CDialog::OnTimer(nIDEvent);
13  }
```

第 06 行是用来设置进度条的运行步长，所用到的 upper、lower 变量是在 OnInitlDialog()函数中按如下代码的第 01 行获取的：

```
01  progress.GetRange(lower, upper);        //获取进度条的最大、最小值
02  SetTimer(1, 1000, NULL);                //设置进度条更新时钟
```

第 02 行是让进度条可以更新显示，其中的第 2 个参数表示每隔 1 秒更新一次进度条的状态。代码 5-9 的第 07 行是用来更新进度条的当前位置，第 08 行用进度条类的成员函数 GetPos()来得到进度条的当前位置。

5.6.3　创建滑块控件

同样在这个项目的对话框上手动新建一个滑块控件，它所关联的类为 CSliderCtrl。它同进度条一样，也需要在 OnInitlDialog()函数中进行初始化。首先在对话框类中添加一个 CSliderCtrl 型的变量，名为 slider，具体代码如下：

```
01  slider.SetRange(0, 100);          //设置滑块的范围(0～100)
02  slider.SetPos(1);                 //初始位置
```

在进度条的基础上，应该不难理解这两行代码的意义。第 01 行是滑块控件的表示范围，第 02 行是它的初始显示位置。

注意：滑块控件除了可以关联 Control 型的 CSliderCtrl 变量，还可以关联 Value 型的变量，而 Value 型的变量只有 int 型。

为了让滑块显示刻度线并且有文字提示，就要涉及到滑块的属性设置，如图 5.38 所示就是滑块控件的“外观”属性。图中的 Auto Ticks 当它设为 True 时，可以让指定滑块在其值范围内对于每增量都有一个刻度线。

注意：但它起作用必须在 Tick Marks（指定滑块显示刻度线）为 True 的前提下。

Tooltips 是指定滑块控件支持工具提示。经此设定后，可以得到如图 5.39 所示的效果。

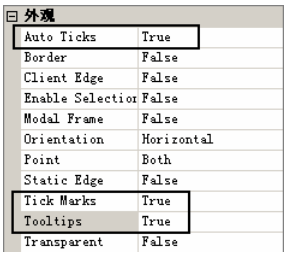

图 5.38　滑块控件的“外观”属性

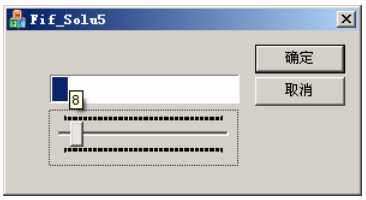

图 5.39　滑块显示刻度效果

可以看出，图中显示的数字 8 就是滑块当前位置的刻度，但是滑块的刻度相对于当前滑块的范围显得有点拥挤。在 OnInitlDialog()函数中写入如下代码可以改善此效果，如图 5.40 所示。

```
slider.SetTicFreq(5);//每隔 5 个单位设 1 个刻度线
```

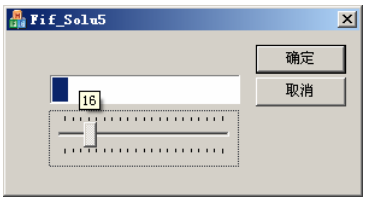

图 5.40　滑块刻度改善效果

5.6.4　使用滑块控件

滑块控件的事件类型同进度条的一样，也是如图 5.37 所示的 4 种消息响应。这里响应消息 NM_CUSTOMDRAW，下面代码是此事件函数的命令。

```
01  void CFif_Solu5Dlg::OnNMCustomdrawSlider2(NMHDR *pNMHDR, LRESULT
    *pResult)
02  {
03      LPNMCUSTOMDRAW pNMCD = reinterpret_cast<LPNMCUSTOMDRAW>(pNMHDR);
04      // TODO: 在此添加控件通知处理程序代码
05      int pos = slider.GetPos();              //获取滑块的当前位置
06
07      CString strText = _T("");               //初始化 CString 变量
08      strText.Format(_T("%d"), pos);          //int 型转为 CString 型
09      SetDlgItemText(IDC_EDIT1, strText);     //在编辑框控件中显示
10      *pResult = 0;
11  }
```

它实现的效果是将当前滑块的位置在编辑框中显示，第 05 行是要获取滑块的当前位置值，第 09 行就是将滑块刻度值实时显示在编辑框 IDC_EDIT1 中，效果如图 5.41 所示。

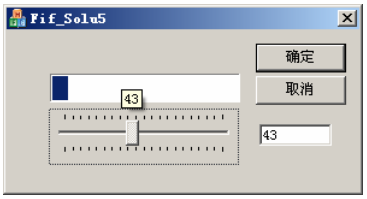

图 5.41　滑块事件响应效果

5.6.5　进度条、滑块控件编程实例

结合上面的例子，当滑块来回滑动时，进度条上的高亮块显示也随之变化，代码如下：

```
01  void CFif_Solu5Dlg::OnTimer(UINT_PTR nIDEvent)
02  {
03      // TODO: 在此添加消息处理程序代码和/或调用默认值
04      //int pos;//位置变量
05      …
06      slider.SetPos(pos);                  //滑块位置
07
08      CDialog::OnTimer(nIDEvent);
09  }
10  void  CFif_Solu5Dlg::OnNMCustomdrawSlider2(NMHDR  *pNMHDR,  LRESULT
*pResult)
11  {
12      LPNMCUSTOMDRAW pNMCD = reinterpret_cast<LPNMCUSTOMDRAW>(pNMHDR);
13      // TODO: 在此添加控件通知处理程序代码
14      //int pos = slider.GetPos();         //获取滑块的当前位置
15      pos = slider.GetPos();               //获取滑块的当前位置
16  …
```

```
17      progress.SetPos(pos);                    //进度条位置
18      *pResult = 0;
19  }
```

可以看出第 04～14 行都是在定义进度条与滑块的位置变量，因为它们类型相同，表达的意义也相同，所以可以将位置变量 pos 声明为对话框类的成员变量。另外这两种控件的位置是有关联的，此项目中它们的位置是同步发生变化的。因此可以用同一个 int 型变量，如代码第 06～17 行，项目最后调试效果如图 5.42 所示。

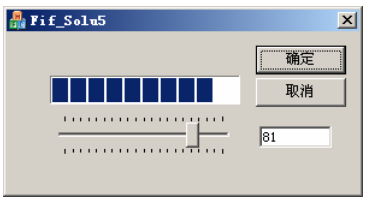

图 5.42　进度条与滑块关联效果

5.7　简单应用 ActiveX 控件

ActiveX 控件与前面讲的控件在实质上是一样的，只不过前面的按钮、组合框、静态文本框这些控件是 VC++开发软件提供的。而 ActiveX 控件来自外部资源，它是由不同的语言编写封装而成。

类似于 VS 提供的按钮控件，在它的基础上可以继续编辑代码以形成新的 ActiveX 控件，ActiveX 控件也有自己的属性与消息类型。VS 有专门的项目可以编辑 ActiveX 控件，VS 下的 MFC 有 3 种项目类型，如图 5.43 所示。

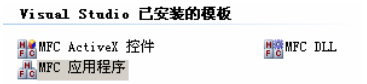

图 5.43　MFC 项目

提示：本书只用到 MFC DLL 与“MFC 应用程序”两种项目，而“MFC ActiveX 控件”项目就可以生成以 OCX 结尾的文件，也就是 ActiveX 控件。

本节不去介绍如何生成一个 ActiveX 控件，实际上它同后面要讲的动态链接库 dll 的生成原理类似。下面只简单介绍如何应用 ActiveX 控件，它的优点是由不同类型语言编写的 ActiveX 控件可以被 VS 程序调用，同时 VS 程序生成的 ActiveX 控件也可以应用于其他场合。下面即添加一个名为 Fif_Solu6 的基于对话框的 MFC 应用程序，分以下两种情况介绍：注册表中已有的 ActiveX 控件、现从网上下载的 ActiveX 控件。

5.7.1　应用注册表中已有的 ActiveX 控件

在“资源视图”的对话框面板上右击，弹出如图 5.44 所示的快捷菜单。选择“插入

ActiveX 控件”命令后，弹出如图 5.45 所示的“插入 ActiveX 控件”的对话框。

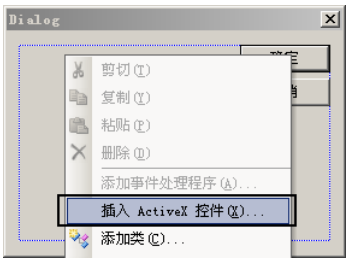

图 5.44　添加 ActiveX 控件菜单

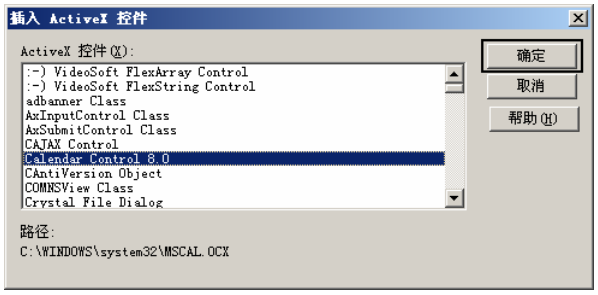

图 5.45　选择插入的 ActiveX 控件

这里选择 Calendar Control 8.0，然后单击“确定”按钮，就可以在对话框上看到这个控件的样子。但它的初始尺寸太小，经过调整得到如图 5.46 所示的完整画面。

图 5.46　调整大小后的所选控件

这只是添加了 ActiveX 控件资源，但如果没有关联控件的类成员变量与函数接口，就不知道如何应用控件。所以还得再添加与此控件相关的类，可以右击“类视图”下的 Fif_Solu6 项目，在弹出的菜单中选择“添加→类”命令，弹出如图 5.47 所示的对话框。

在“类别”中选择 MFC，对应右边的“Visual Studio 已安装的模板”中选择“ActiveX 控件中的 MFC 类”，单击“添加”按钮后，弹出如图 5.48 所示的“添加类向导”对话框。

因为这类的 ActiveX 控件是系统中已有的，所以“从以下来源添加类”选中“注册表”单选按钮。右边的“可用的 ActiveX 控件”组合框中选择对应插入控件名称的 Calendar Control 8.0<1.0>。接着在下面的“接口”框中出现 ICalendar，之后单击中间的一系列按钮

完成生成类文件工作。如果“从以下来源添加类”选中“文件”单选按钮，则需要在“位置”中加载控件的路径。

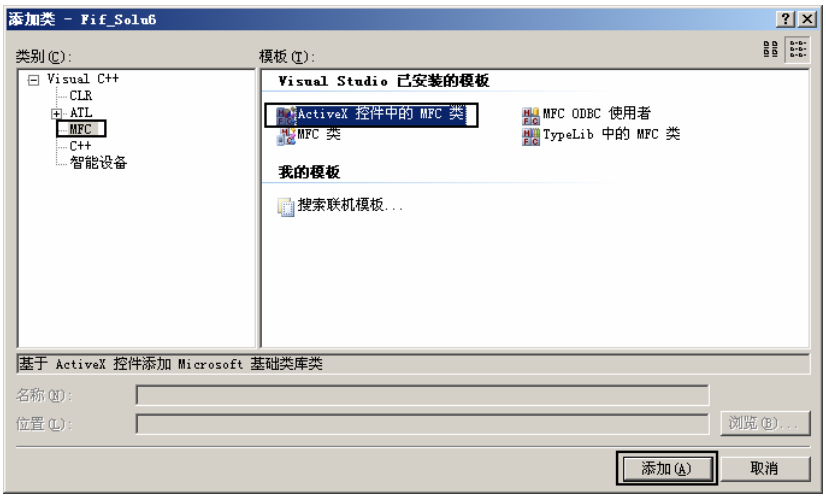

图 5.47　为 ActiveX 控件添加类

图 5.48　从 ActiveX 控件添加类

提示：从“文件”加载的接口个数会比从“注册表”中要全。

在单击中间的第 1 个按钮前，要选中左边的接口名称，它表示生成此接口的类，如“生成的类”这项所示。如果选中的接口名称有多个，就单击第 2 个“添加所有接口”按钮。第 3 个按钮是“移除接口”，第 4 个是“移除所有接口”按钮。下面一栏是接口生成的类名、头文件、源文件名称，另外还可以单击旁边的“浏览”按钮去选择文件，最后单击“完成”按钮，在“类视图”中看到确实有 CCanlendar 这个类，如图 5.49 所示。

提示：不同的 ActiveX 控件的接口个数不一样。

图 5.49　插入的 ActiveX 控件类

接下来就是像对普通控件编辑一样，应用 CCalendar 控件。这就需要仔细明确类中成员函数的实现意义，在日历中选择完日期后，单击对话框中的“确定”按钮后，弹出提示所选日期的消息框。代码 5-10 是“确定”按钮的响应函数。

代码 5-10　应用 ActiveX 控件示例：Fif_Solu6Dlg.cpp

```
01  void CFif_Solu6Dlg::OnBnClickedOk()
02  {
03      // TODO: 在此添加控件通知处理程序代码
04      CString str_day, str_month, str_year;       //存放年月日变量的字符串
05      CString str;                                //结果字符串
06      short day, month, year;                     //年月日 short 型变量
07      day = cal.get_Day();                        //获取日值
08      month = cal.get_Month();                    //获得月值
09      year = cal.get_Year();                      //获得年值
10
11      str_day.Format(_T("%d"), day);              //类型变换
12      str_month.Format(_T("%d"), month);
13      str_year.Format(_T("%d"), year);
14
15      str = str_year + CString("年") + str_month + CString("月") + str_day +
CString("日");//结果
16
17      AfxMessageBox(str);                         //消息盒子显示最后结果
18  //  OnOK();
19  }
```

在写这个函数前，记得在对话框头文件中加入下面代码：

```
01  #include "CCalendar.h"
02  …
03  CCalendar cal;//日历控件对象
```

第 01 行位于类声明体外，第 03 行对象的定义位于类体内。为了获取日历的年、月、日值，如上段代码的第 07～09 行运用该类的成员函数。第 11 行是将获得的 short 型变量转换为 CString 型，最后用消息框显示结果（第 17 行）。经这样编写后的程序编译过程中没有错误提示，且在运行的开始阶段也无异常，但当单击“确定”按钮时，会弹出如图 5.50 所示的 bug 对话框。

编译时无错误，说明代码语法没有 error。那就从另一种角度考虑，之前操作的“按钮”控件，不管在哪些地方都要与控件的 ID 号发生关联。但这个“日历”控件似乎没有这样的动作，因此不妨大胆地假设下是否在与“日历”控件的 ID 号关联后，就会消除 bug。经验证，当在对话框类的 DoDataExchange()函数中写下如代码后，果然运行正常。

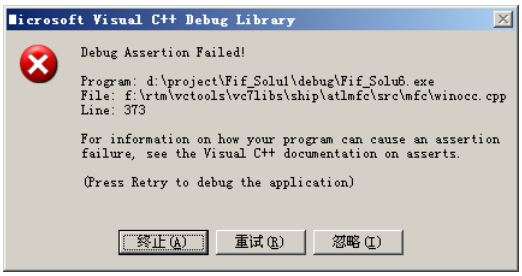

图 5.50　运行过程中的 bug 提示

```
01  void CFif_Solu6Dlg::DoDataExchange(CDataExchange* pDX)
02  {
03      CDialog::DoDataExchange(pDX);
04      DDX_Control(pDX, IDC_CALENDAR2, cal);
05  }
```

第 04 是 cal 对象与 IDC_CALENDAR2 控件发生数据交换，运行效果如图 5.51 所示。

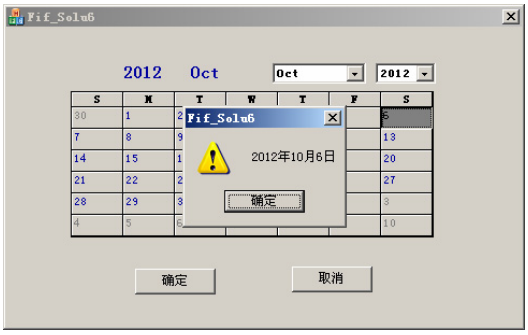

图 5.51　正常运行效果

上面的添加类或与控件关联的变量操作都是手动进行的。同理也可以自动生成这些东西，添加名为 Test_ActiveX 的项目（基于对话框的 MFC 应用程序）。依照上面方法插入控件后，右击控件，单击“添加变量”菜单，弹出如图 5.52 所示的向导。

图 5.52　为 ActiveX 控件添加变量

“变量类型”为 CCalendar1，给变量取名为 cal，“控件类型”是指该控件文件的后缀名（此为 OCX）。单击“完成”按钮会自动生成标志该控件的类，并且有了数据交互代码，接下来就只需编辑响应对话框上的“确定”按钮的函数，运行效果同上。

5.7.2　应用新下载的 ActiveX 控件

前面是对 VS 开发环境自带的 ActiveX 控件的添加，而对于新下载的 ActiveX 资源，在使用前还得有个前奏——注册资源。例如上网下了个描述“二维坐标曲线”的 OCX 控件 PlotLineControl，如果它自带有注册程序，可以运行这个.exe 文件；但如果没有，就是选择“开始”→“运行”命令，输入下面命令：

```
regsvr32
```

接着输入一个空格，然后将 ActiveX 控件拖放到组合框中，如图 5.53 所示。

图 5.53　注册 ActiveX 控件

单击“确定”按钮后，会弹出注册成功信息框，如图 5.54 所示。

图 5.54　注册成功提示信息

之后在插入 ActiveX 控件列表中确实找到了 PlotLineControl 控件。下面对该控件的编辑可参照（1）中介绍的方法。如果想卸载这个控件，还是在“运行”菜单中输入下面指令：

```
regsvr32 /u
```

后把要反注册的控件拖放进去，单击“确定”弹出反注册成功信息，如图 5.55 所示。

图 5.55　反注册成功提示信息

5.8　本 章 总 结

本章内容是关于控件的介绍，前 6 节是对 Windows 标准控件的讲解，5.7 节简单介绍

了 ActiveX 控件的基本使用：如何注册、怎样卸载、响应控件的效果。

读者可以在 5.1 节大致了解下标准控件都有哪些，都用来干什么。接着中间的 5 节的内容介绍了常用控件（按钮、静态文本框、编辑框、组合框、进度条与滑块）的基本信息：如何创建、控件的属性与消息类型、怎样响应控件，最后给出了运行效果图。

总体来说，本章从实用的角度出发，避免了过多繁琐的理论介绍。本着先模仿后知根知底的策略，让读者可以有一个不同的学习方法。

5.9　课 后 练 习

思考题：

1. Windows 标准控件与 ActiveX 控件的不同之处是什么?

2. 是否可以给按钮贴图，它关联的类名是什么?

3. 静态文本框控件可否接收通告消息?

4. 用哪个函数可以获取编辑框中的文本信息?

上机练习：

1. 设计一个包含按钮、编辑框、静态文本框、组合框这 4 种控件的 MFC 程序。

2. 试编个程序，可以播放音频文件，用进度条显示它的播放进度，用滑块可以调整声音大小。

3. 下载一个新的 ActiveX 控件，并应用于 MFC 程序中。