



初识MFC

周艳

西南交通大学电气工程学院

西南交通大学

什么是MFC

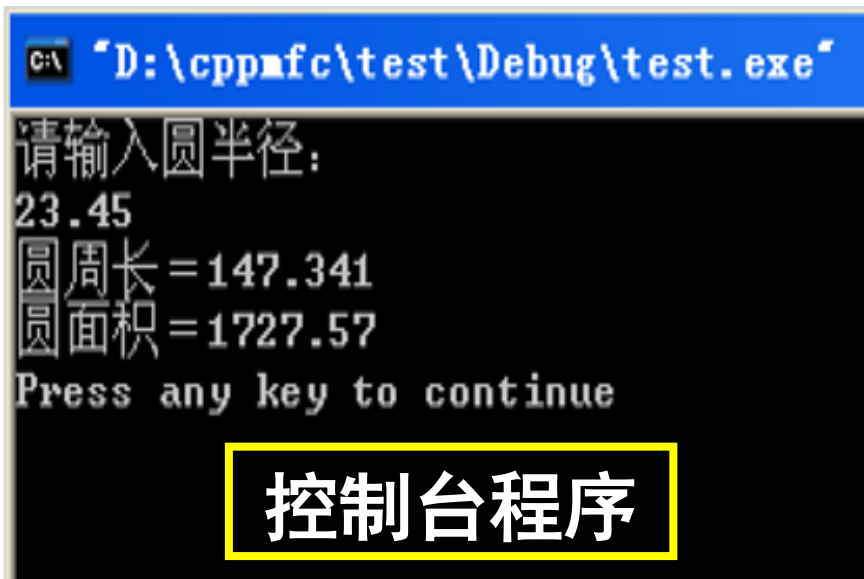


先了解什么是可视化编程

□ 可视化 (Visual) 编程有三个基本特点:

- 可视化的图形用户界面设计
- 使用面向对象方法编程
- 采用事件驱动的程序运行方式

从DOS到Windows的飞跃



```
C:\ "D:\cppmfc\test\Debug\test.exe"
请输入圆半径:
23.45
圆周长=147.341
圆面积=1727.57
Press any key to continue
```

控制台程序



图形用户界面GUI
(Graphical User Interface)

一、Windows编程的特点

□ Windows是基于**图形界面**的**多任务**操作系统。

■ **输入方式多样化**：除接收键盘输入，还可接收鼠标左键、右键、单击和双击等各类输入事件。

■ **基于事件的消息 (Message) 驱动机制**

区别于过程驱动

○ “事件→消息→处理” 非顺序机制

1、某**事件**发生时，Windows根据具体事件产生对应**消息**，并发送到指定应用程序的消息队列



2、应用程序从消息队列中取出**消息**，并根据不同的消息进行不同的**处理**



事件发出消息，消息激活对象

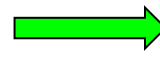
□事件(event): 对于Windows的某种操作。

- 无论系统产生的动作还是运行程序时用户产生的动作都称为事件。最常见的用户事件是鼠标事件和键盘事件。

□消息(message): 一种特殊的数据结构，描述某个“事件”发生的相关信息，例如：消息接收者的句柄、事件的类型和来源、消息参数等。

- Windows系统消息传递无时无刻不在发生，电脑闲置时还会有系统时间的消息传递。

什么对象可以产生或接收什么事件①



Windows
规定

- 命令按钮有鼠标单击 (BN_CLICKED) 和双击 (BN_DOUBLECLICKED) 事件;
- 文本编辑框有改变文本 (EN_CHANGE)

□ 当对象上发生了某事件并发出相应的消息后, 接收者对象就要响应并处理该消息。

□ 每个需要响应的消息要求对应一个处理该消息的程序——

消息处理函数。

消息和程序如何对应①

□ 通过消息与消息处理函数——对应的消息映射表



二、VC++的两种可视化编程方法

1. 基于Windows API函数的编程方法

- **Windows API**(Application Programming Interface) 是Windows 操作系统与Windows应用程序之间的标准接口，提供了上千个标准函数、宏和数据结构的定义。
- **利用Windows API开发程序**需要对Windows编程原理有很深刻的认识，需要手工编写冗长的代码。当程序长度逐渐膨胀时，调试程序会变得越来越困难。

2. 基于Microsoft MFC的编程方法

- MFC是利用面向对象的思想，将**Windows大部分API函数封装**起来的一个浩瀚的**类库**，利用MFC编写程序**本质是选择该类库中合适的类**，并调用其下相应成员函数来完成某个功能。
- 传统的API编程步骤不见了，不知道编写的程序何时建立、何时消亡。因为**MFC的应用程序架构**隐藏了类似于API编程所要求的步骤，提供了一个**标准化的程序结构**，使开发人员不必从头设计一个Windows应用程序。

(1) MFC部分数据类型

数据类型	对应的基本数据类型	意义
BOOL	bool	布尔值
BSTR	unsigned short *	32位字符指针
BYTE	unsigned char	8位无符号数
WORD	unsigned short	16位无符号数
DWORD	unsigned long	32位无符号数
LONG	long	32位带符号数
UINT	unsigned int	32位无符号数
COLORREF	unsigned long	用作颜色值的32位值
LPSTR	char *	指向字符串的32位指针
LPCSTR	const char *	指向字符串常量的32位指针
LPARAM	long	作为参数传递给窗口过程或回调函数
LPARAM	unsigned int	作为参数传递给窗口过程或回调函数



(2) 学习MFC的要求

- 刚开始要“不求甚解”，懂得代码的放置以及原理就可以了。不要一开始就试图了解整个MFC类库。从理解和使用两个方面学习MFC，理解MFC应用程序的框架结构。
- 随着学习的深入，在查看MFC源代码后，就能逐渐了解到该类型应用程序的执行流程，从而最终掌握MFC应用程序的开发。

三、MFC的基本概念

□ Microsoft Foundation Class (微软基础类库)

- 从**物理**角度看是一个庞大的类库，对应Windows系统目录下的一系列mfc*.dll文件；
- 从**逻辑**角度看是一个面向对象的应用程序框架，程序员可使用这一框架创建Windows应用程序。
- MFC的**组织**以C++类的层次形式组织在一起，高层类提供一般功能，而低层类实现更具体的功能，每一个低层类都是从高层类派生而来，因此继承了高层类的功能。



四、MFC体系结构

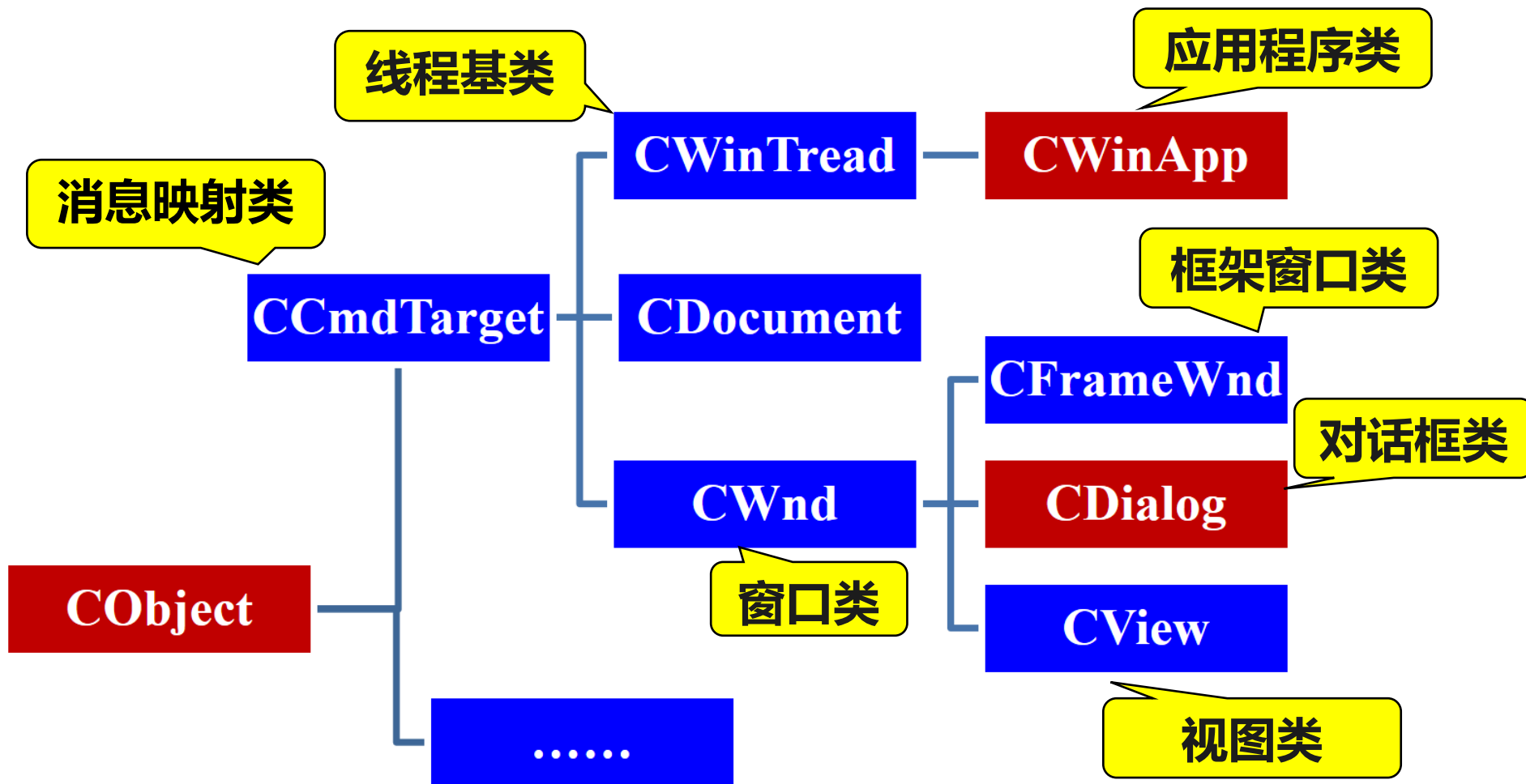
□ MFC主要组成部分：**类、宏和全局函数。**

- **MFC宏**主要功能：**消息映射**、运行时对象类型服务、诊断服务、异常处理。
- MFC约定：全局函数以 “**Afx**” 为前缀，全局变量以 “**afx**” 为前缀。
- **类是MFC中最主要的内容。** MFC类以层次结构方式组织。



MFC类的主要层次结构

- **CObject**是MFC的**抽象基类**，是MFC中多数类的根类。
提供了许多编程所需的公共操作。



(1)消息映射类：CCmdTarget类

- 所有具有消息映射属性的基类。
- **消息映射**规定了当一对象接收到消息命令时，应调用哪个函数对该消息进行处理。

(2)线程基类：CWinThread类

- 所有线程的基类，可直接使用。
- CWinApp类就是从CWinThread类中派生出来的。



(3)窗口应用程序类：CWinApp类

- 每个应用程序有且仅有一个从CWinApp类中派生的应用程序对象。在程序运行期间该对象与其它对象相互协调；
- CWinApp类封装了初始化、运行、终止应用程序的代码。

(4)窗口类：CWnd类

- 提供所有窗口类的基本功能；
- 所有从CWnd类派生的类都有m_hWnd句柄。
- 单文档框架窗口类CFrameWnd也是CWnd的派生类



(5) 框架窗口类：CFrameWnd类

- 往往用于创建应用程序的主窗口
- 在编写文档/视图结构的应用程序时，CFrameWnd作为主窗口管理视图和文档对象。
- CFrameWnd支持单文档界面（SDI）

(6) 文档类：Cdocument类

- 文档对象由文档模板对象创建，管理应用程序的数据

(7) 视类：CView类

- 表示框架窗口的用户区，显示文档数据并允许用户与之交互

五、MFC开发设计应用程序

- 尽管每个应用程序具体实现的功能不同，但同一类程序的基本结构是相同的。因此，通常采用**MFC AppWizard**创建一个**MFC应用程序框架**。
- 使用MFC应用程序向导，没有编写一句代码，就创建出一个应用程序，这就是MFC给开发人员带来的便利。

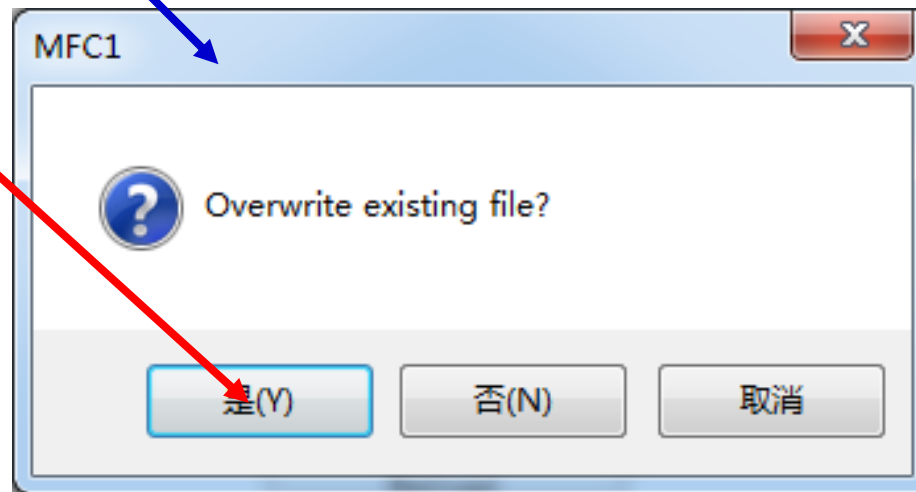


□ MFC AppWizard可选择的应用程序类型

- **Single document**: **单文档界面(SDI)应用程序**。应用程序运行时只能打开一个文档。如记事本, 当选择File菜单中的Open菜单项打开新文档时, 当前显示的文件在新文件打开前自动关闭。
- **Multiple document**: **多文档界面(MDI)应用程序**。应用程序可同时打开多个文档, 例如 Word。
- **Dialog based**: **基于对话框的应用程序**。应用程序将显示一个简单的对话框来处理用户的输入。例如计算器Calculator。

对话框的概念

- ❑ 对话框(Dialog)是收集信息或提供反馈的窗口，它通过控件与用户交互。
- ❑ 控件(Controls)是程序与用户之间交互的接口，完成用户的输入和程序运行过程中的输出功能。



提示消息对话框



对话框程序的类型

□ 模式对话框

- 大部分窗口采用的形式。当对话框弹出时，用户必须在对话框中操作，在退出对话框之前，对话框所在的应用程序不能继续执行。
- 最典型的应用是应用程序的“关于对话框”。

□ 非模式对话框

- 不用关闭非模态窗口就可以操作其他窗口。
- 最典型的应用就是QQ的聊天窗口。

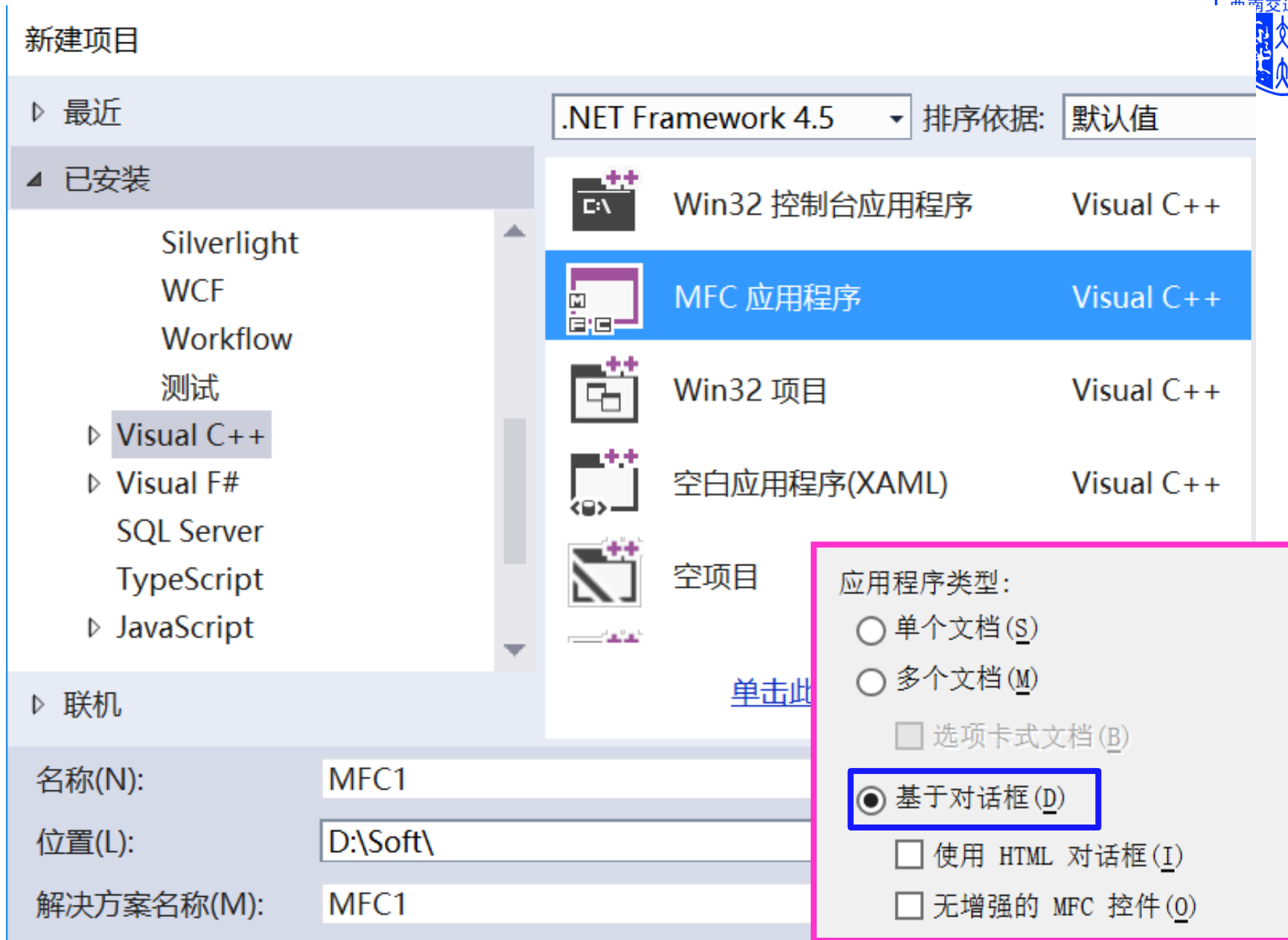
对话框实际上是一个窗口，封装在CDialog中，是所有对话框的基类

控件

□ 程序与用户之间交互的接口，完成输入输出功能

控件类型	功能	控件类
静态控件	用来提供标题或说明性信息	CStatic
按钮控件	提供一种单击输入机制。有三类：自动复位按钮、单选按钮和复选框	CButton
滚动条	用来水平或垂直滚动另一控件内的文本或图像	CScrollBar
列表框	提供一个选项列表，有效的可以是一项或多项	CListBox
编辑控件	允许文本输入或对显示的文本进行编辑	CEdit
组合框	可以从中选择的选项列表，还允许用户直接输入文本	CComboBox

示例：利用向导生成一个MFC模式对话框应用程序MFC1



MFC 应用程序向导 - MFC1



生成的类

概述

应用程序类型

复合文档支持

文档模板属性

数据库支持

用户界面功能

高级功能

生成的类

生成的类(G):

CMFC1App
CMFC1Dlg

类名(L):

CMFC1App

基类(A):

CWinApp

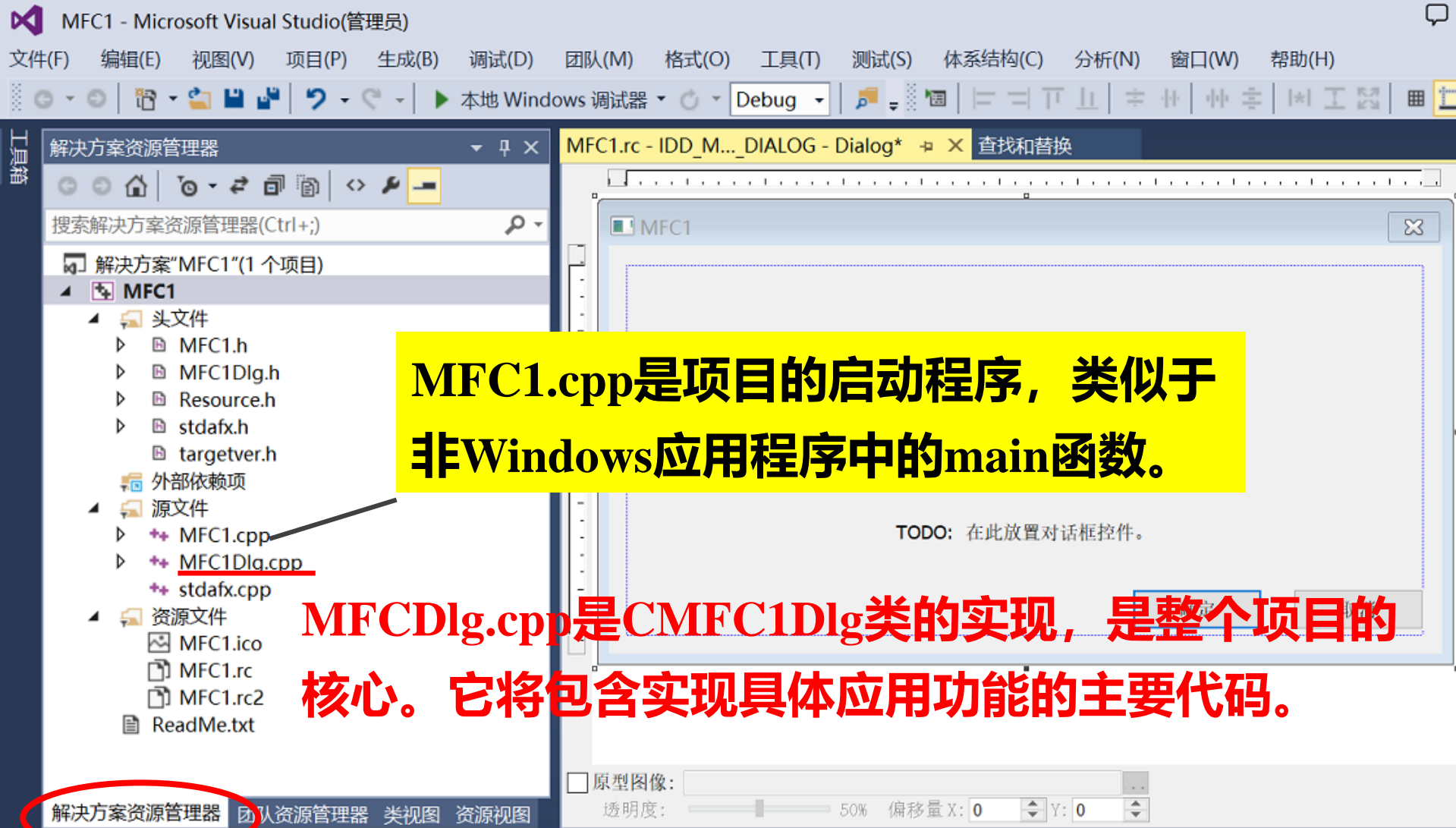
.h 文件(E):

MFC1.h

.cpp 文件(P):

MFC1.cpp

main函数去哪儿了?



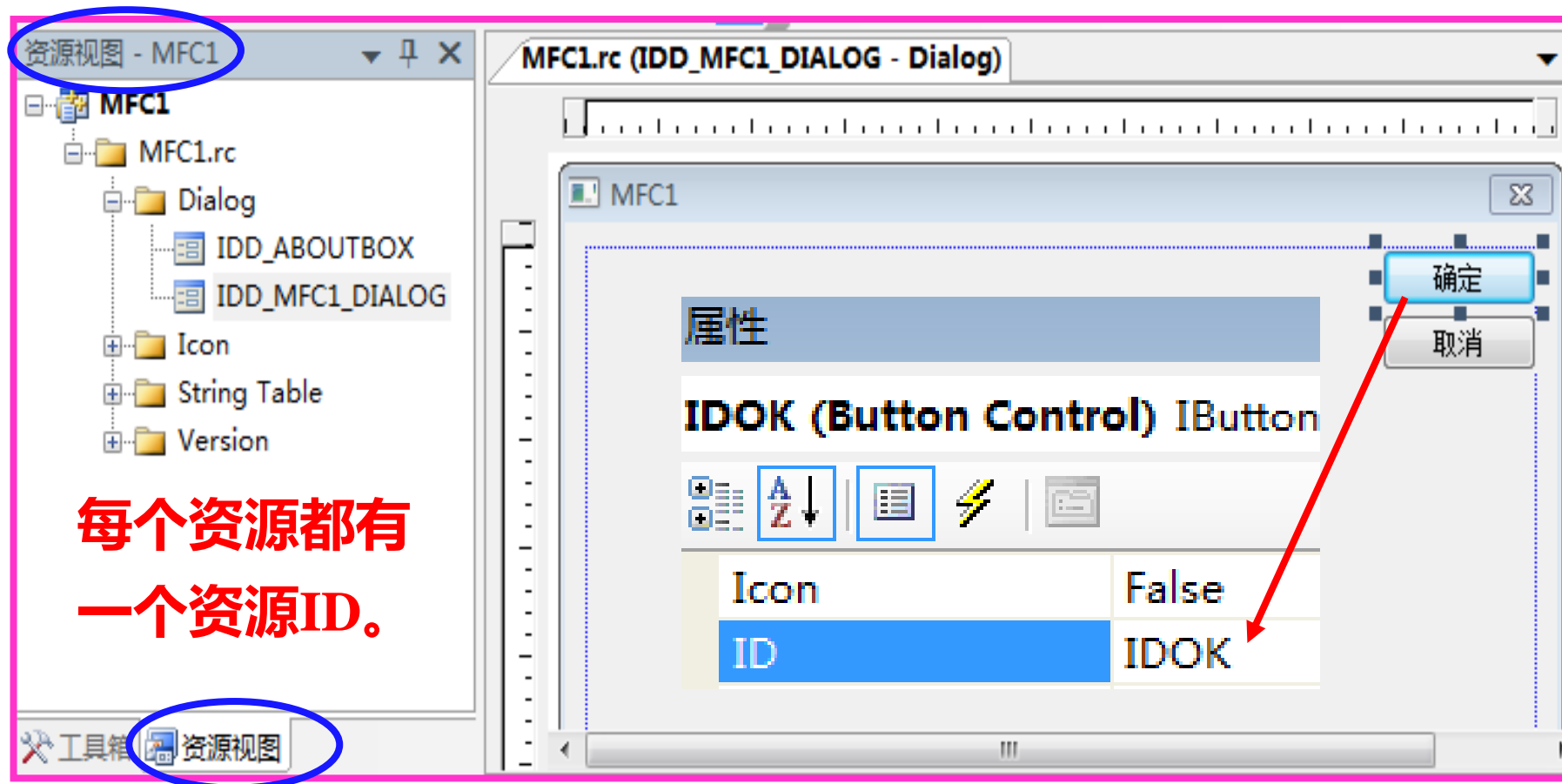
The screenshot shows the Microsoft Visual Studio interface. On the left, the 'Solution Explorer' (解决方案资源管理器) is open, displaying the project structure for 'MFC1'. The 'Source Files' (源文件) folder is expanded, showing 'MFC1.cpp', 'MFC1Dlg.cpp', and 'stdafx.cpp'. 'MFC1Dlg.cpp' is highlighted with a red box and a line pointing to a yellow text box. The main window shows a dialog box titled 'MFC1' with a 'TODO: 在此放置对话框控件。' (TODO: Place dialog box controls here) message. At the bottom, the 'Solution Explorer' tab is circled in red.

MFC1.cpp是项目的启动程序，类似于非Windows应用程序中的main函数。

MFC1Dlg.cpp是CMFC1Dlg类的实现，是整个项目的核心。它将包含实现具体应用功能的主要代码。

- 在Windows中，对话框、菜单、图片、字符串统称为**资源** (resource)。资源描述文件(*.rc)是用于描述资源的一种文本文件

每个资源类别下都有一个或多个相关资源



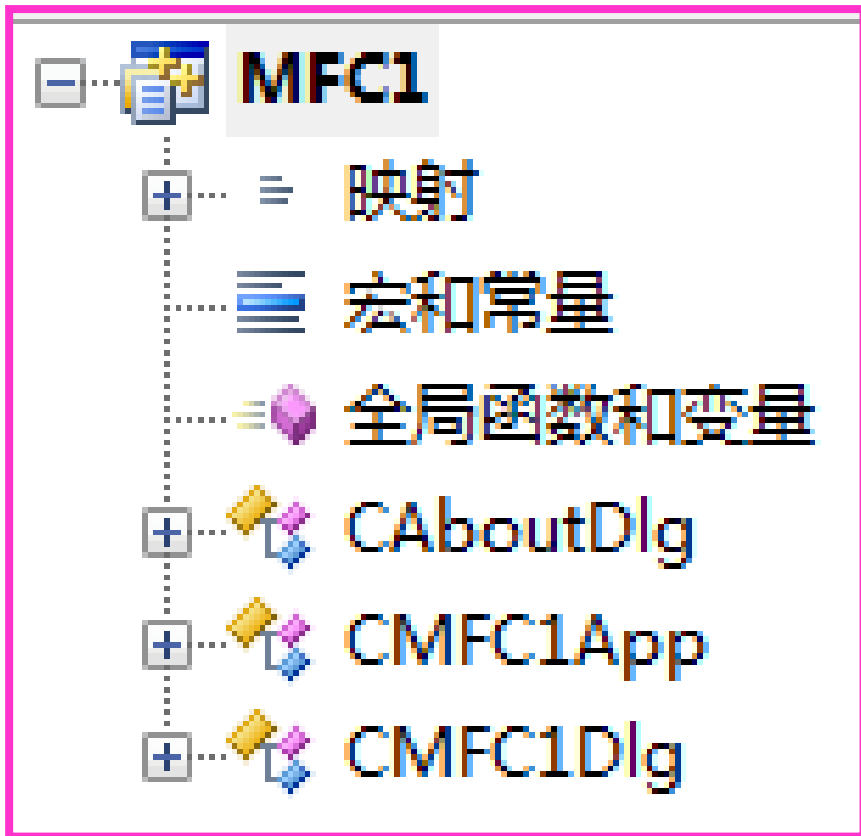


常用资源ID（标识符）的前缀

标识符前缀	含义
IDR_	表示快捷键或菜单相关资源
IDD_	表示对话框资源
IDC_	表示光标资源或控件
IDI_	表示图标资源
IDB_	表示位图资源
IDM_	表示菜单项
ID_	表示命令项
IDS_	表示字符表中的字符串
IDP_	表示消息框中使用的字符串

MFC对话框程序的类结构

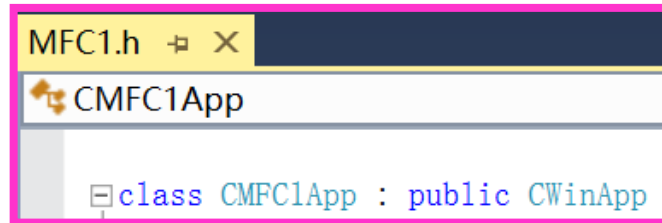
□ 向导生成的对话框项目由三个类构成



- CAboutDlg: 关于对话框类
- CMFC1App: 对话框应用程序类，用于创建应用程序
- CMFC1Dlg: 对话框类，从 CDialog 派生出来。

CMFC1App应用程序类

□ 应用程序类：定义



```
MFC1.h  X
CMFC1App

class CMFC1App : public CWinApp
```

□ 应用程序类的对象：基于框架生成的应用程序有且仅有一个CWinApp派生的类的对象。用于应用程序的初始化、运行和终止，创建窗口前先构造该对象。

■ 默认定义一个全局对象theApp。



```
MFC1.cpp  X
(全局范围)

// 唯一的一个 CMFC1App 对象
CMFC1App theApp;
```

□ main函数：启动应用程序时，Windows调用应用程序框架内置的WinMain函数，WinMain寻找由CWinApp派生出的全局构造的应用程序对象theApp。

□ **函数InitInstance**：发现应用程序对象theApp后自动调用虚函数InitInstance初始化应用程序。

```
MFC1.cpp  X
→ CMFC1App

// CMFC1App 初始化

BOOL CMFC1App::InitInstance()
{
```

可看作MFC程序的入口点。

应用程序的初始代码写在此处

- InitInstance函数可动态(在程序运行时)创建主窗口对象、视图对象和文档对象，以及主框架窗口和视图窗口，并显示应用程序的主框架窗口和视图窗口。

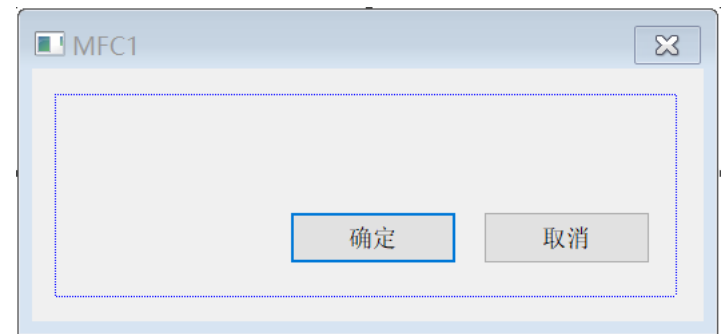
对话框类如何和对话框资源对应❓

❑ 类中的IDD对话框编号与对话框资源中的某对话框ID对应

```
MFC1Dlg.h  X
CMFC1Dlg

// CMFC1Dlg 对话框
class CMFC1Dlg : public CDialogEx
{
// 构造
public:
    CMFC1Dlg(CWnd* pParent = NULL); // 标准构造函数

// 对话框数据
    enum { IDD = IDD_MFC1_DIALOG };
};
```



属性	
IDD_MFC1_DIALOG (Dialog) IDlgEditor	
Context Help	False
Control	False
Control Parent	False
ID	IDD_MFC1_DIALOG



CDialog对话框类的四个基本函数

□ (1)OnInitDialog函数

- 虚函数，响应WM_INITDIALOG消息（在DoModal函数调用期间系统发送此消息）。

□ (2)DoModal()函数

- 激活模式对话框，完成后返回对话框结果。

□ (3)OnOK()函数

- 虚函数，用户单击OK按钮时调用。

□ (4)OnCancel()函数

- 虚函数，用户单击Cancel按钮或ESC键时调用。



模式对话框的创建和初始化

- 创建模式对话框是调用CDialog::DoModal()

```
BOOL CMFC1App::InitInstance()
```

```
{
```

```
    CWinApp::InitInstance();
```

```
    // 标准初始化
```

```
    CMFC1Dlg dlg;
```

```
    m_pMainWnd = &dlg;
```

```
    INT_PTR nResponse = dlg.DoModal();
```

应用程序的初始代码

- DoModal()的返回值为IDOK和IDCANCEL。表明用户在对话框上选择“确认”或“取消”。
- 对话框被生成时会自动调用CDialog::OnInitDialog()。如果要在对话框显示前对其中的控件进行初始化，需要重载此函数，并写入相关的初始化代码。

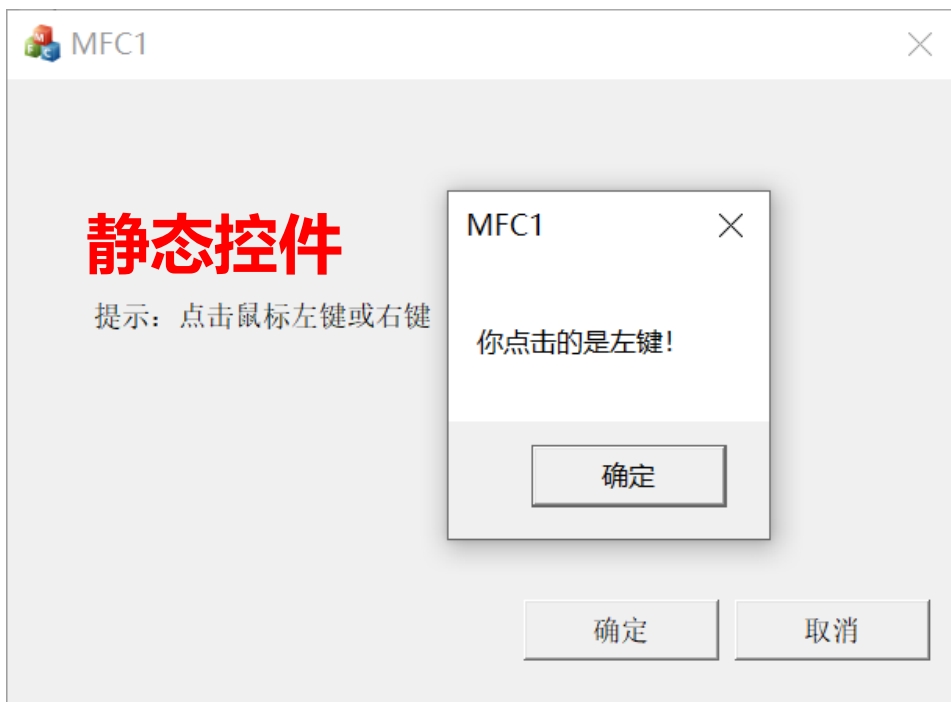
□ 控件对象的创建

- **静态创建**：对话框模板上把控件画好，当CDialog创建对话框时，自动把控件创建好。
- **动态创建**：运行时动态创建，可灵活设计，较复杂。

□ 本例使用的控件

- **静态控件(CStatic)**：显示文本、图标、位图等，通常不进行输入输出。默认ID标识为IDC_STATIC，若要区分和操作不同的静态控件，须重新为它指定一个唯一的ID标识。

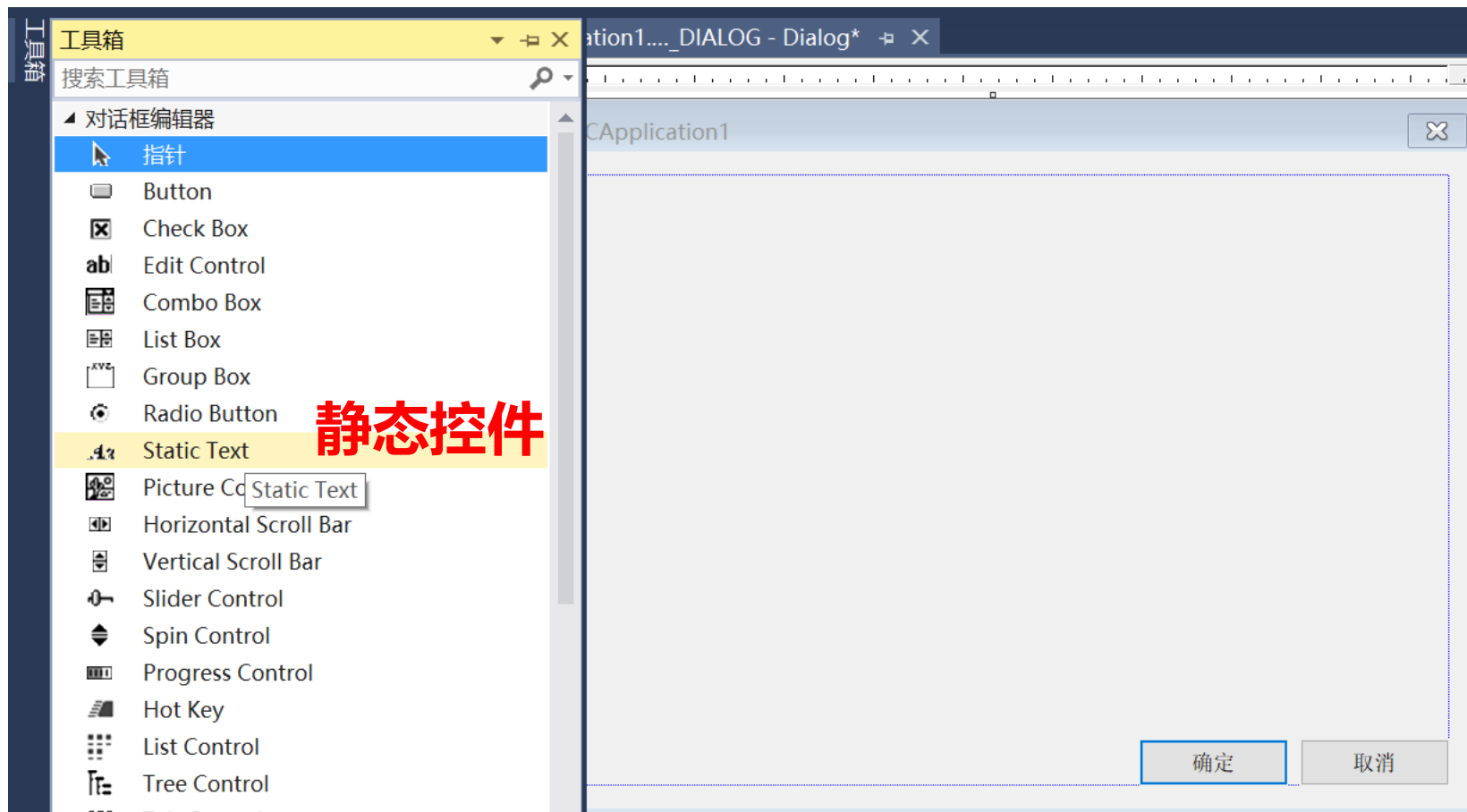
六、对话框应用程序基础示例



使用工具箱在对话框
界面上添加静态控件

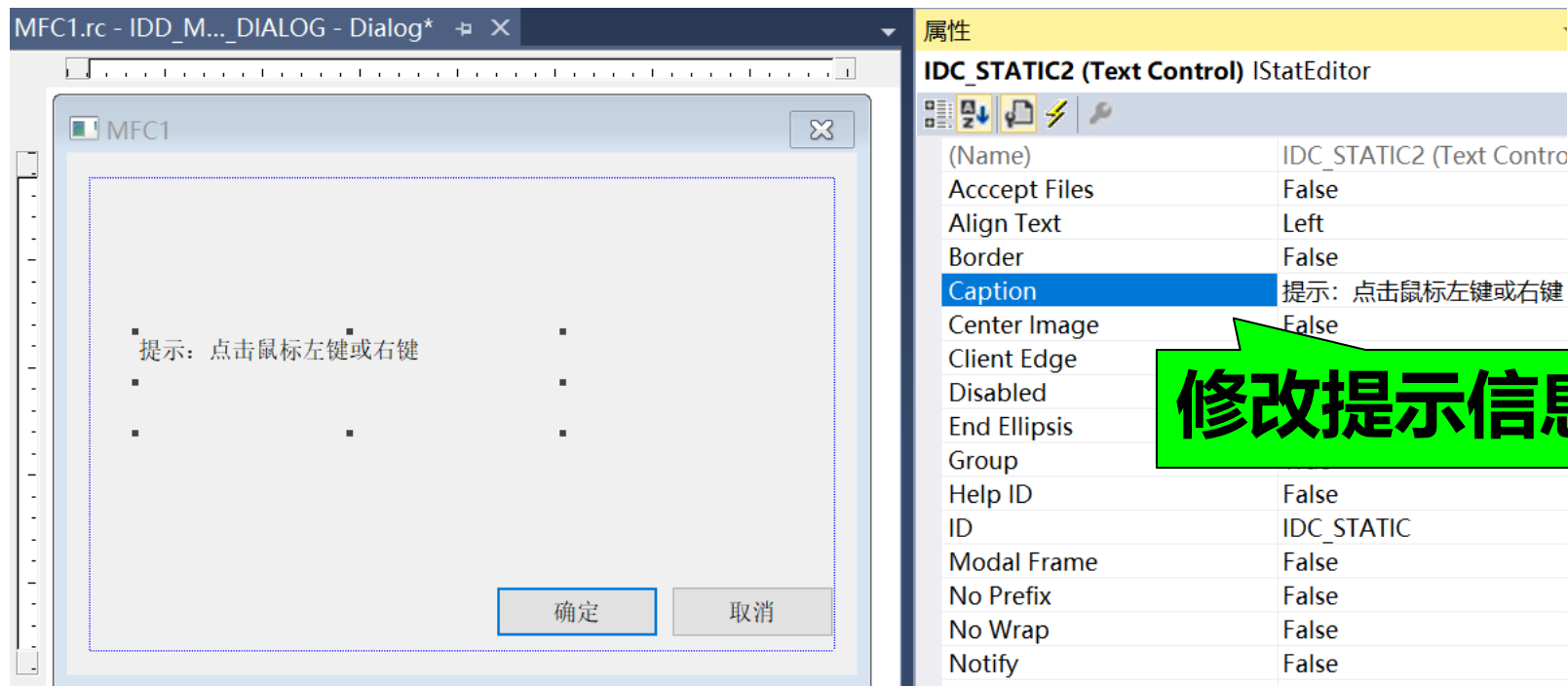


1、添加静态控件



菜单：视图--->工具箱

2、修改静态控件属性



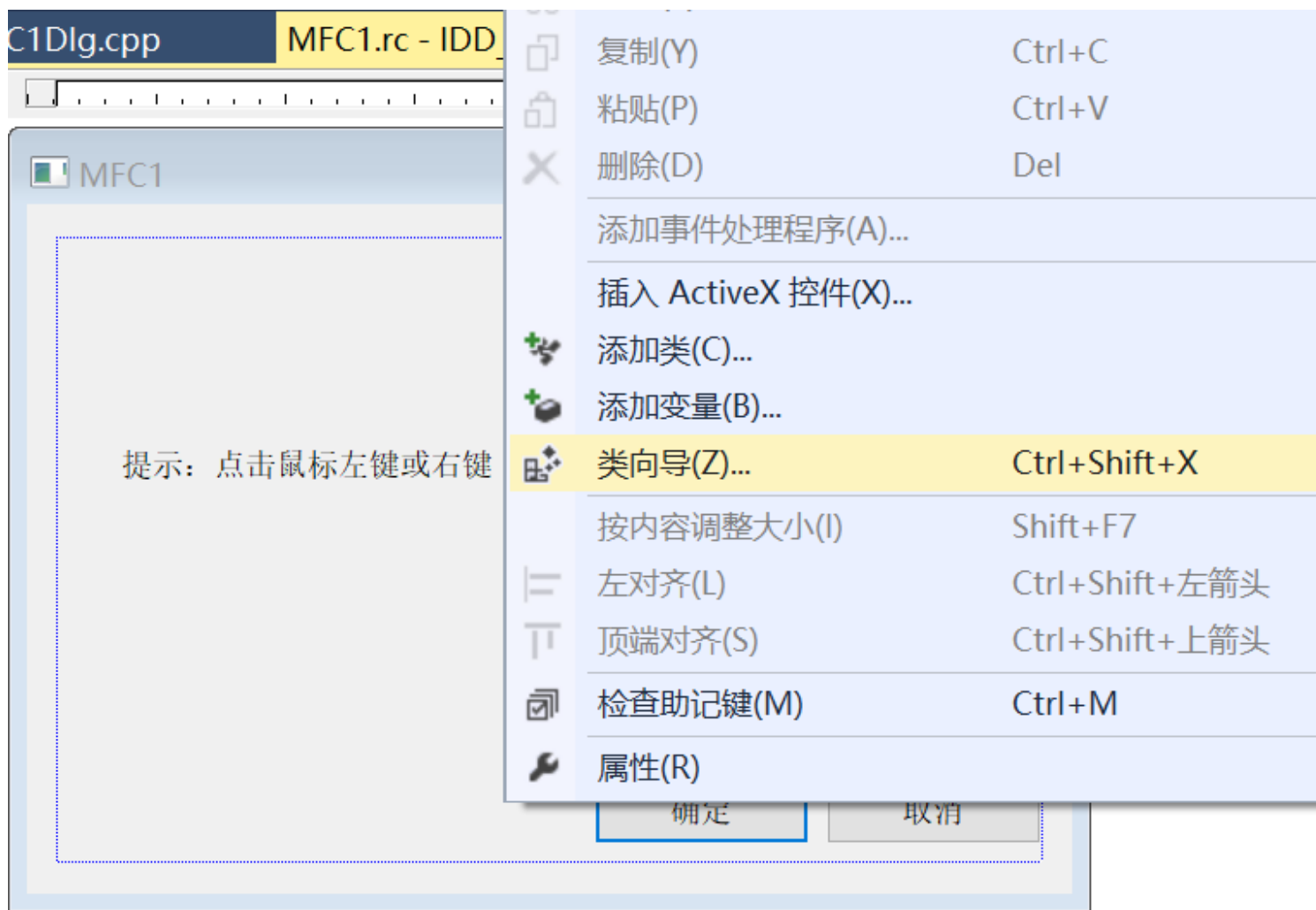
属性

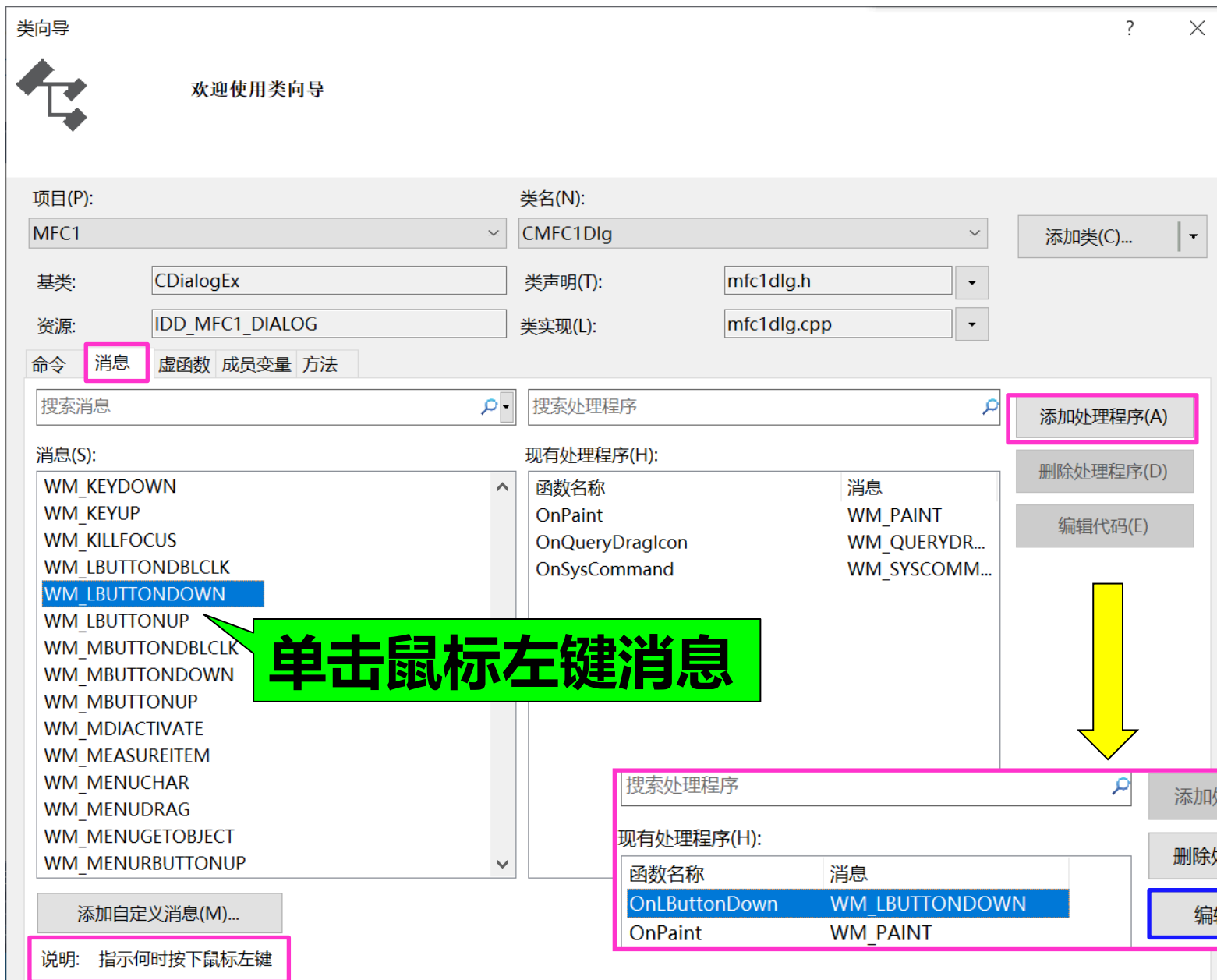
IDC_STATIC2 (Text Control) IStatEditor

(Name)	IDC_STATIC2 (Text Control)
Accept Files	False
Align Text	Left
Border	False
Caption	提示: 点击鼠标左键或右键
Center Image	False
Client Edge	
Disabled	
End Ellipsis	
Group	
Help ID	False
ID	IDC_STATIC
Modal Frame	False
No Prefix	False
No Wrap	False
Notify	False

修改提示信息

3、使用类向导添加消息处理函数







当用户单击鼠标左键时

WM_LBUTTONDOWN

□ MFC通过“**添加消息处理程序**”为WM_LBUTTONDOWN消息映射作了三方面的安排：

■ **MFC1Dlg.h文件：**自动声明消息处理函数OnLButtonDown ()

```
public:  
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
```

■ **MFC1Dlg.cpp文件：**自动在前面的消息映射入口处添加相应的消息映射宏

```
BEGIN_MESSAGE_MAP(CMFC1Dlg, CDialogEx)  
    ON_WM_SYSCOMMAND()  
    ON_WM_PAINT()  
    ON_WM_QUERYDRAGICON()  
    ON_WM_LBUTTONDOWN()  
END_MESSAGE_MAP()
```

■ MFC1Dlg.cpp文件：自动写入一个空的消息处理函数的模板，以便用户填入具体代码



```
MFC1Dlg.cpp*  × MFC1.rc - IDD_M..._DIALOG - Dialog*
→ CMFC1Dlg  OnLButtonDown(UINT nFlags, CPoint point)

void CMFC1Dlg::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: 在此添加消息处理程序代码和/或调用默认值

    CDialogEx::OnLButtonDown(nFlags, point);
}
```

4、编辑消息处理函数

```
MFC1Dlg.cpp  MFC1.rc - IDD_M... _DIALOG - Dialog
(全局范围)

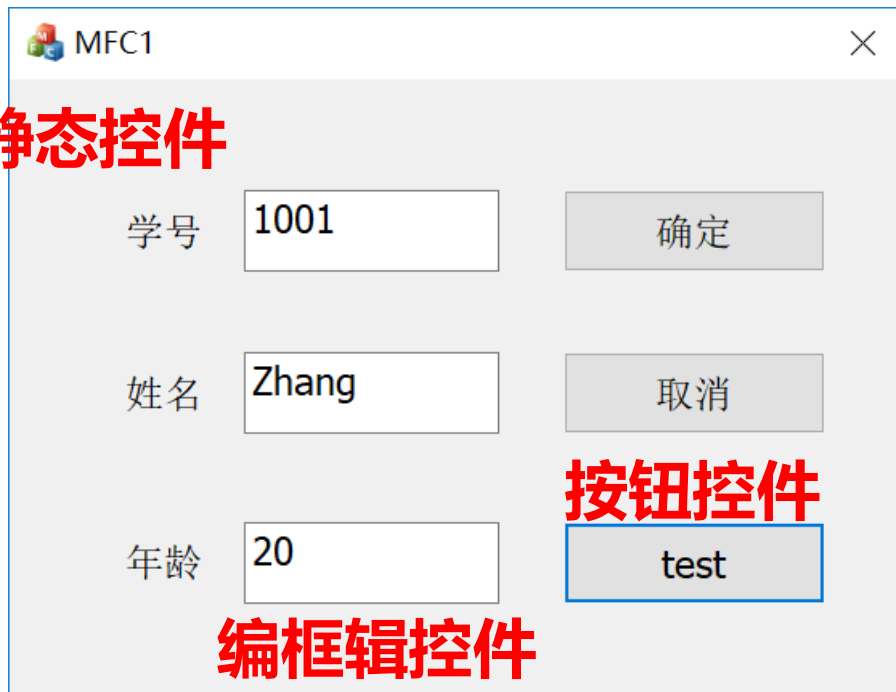
void CMFC1Dlg::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: 在此添加消息处理程序代码和
    MessageBox(_T("你点击的是左键!"));
    CDialogEx::OnLButtonDown(nFlags, point);
}
```

添加提示消息

_T是一个宏，作用是让程序支持Unicode编码。

请大家参考此方式添加鼠标右键消息处理函数

对话框应用程序示例2



MFC1

学号 1001 确定

姓名 Zhang 取消

年龄 20 test

静态控件

按钮控件

编辑控件

按钮控件(CButton): 通过单击或双击执行某种操作。

编辑控件(CEdit): 输入文本信息



MFC1

学号 1001 确定

姓名 Zhang 取消

年龄 20 test

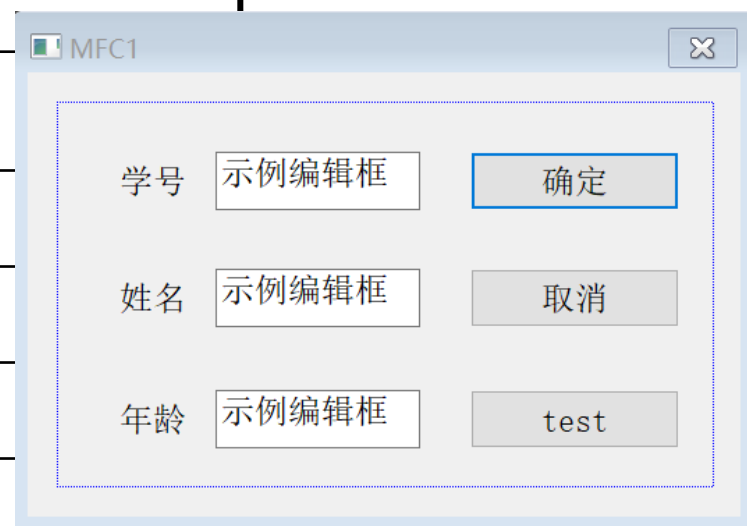
MFC1

学号:1001
姓名:Zhang
年龄:20

确定

1、设计对话框控件

控件	控件标识符	显示的文本
Static Text	IDC_STATIC	学号
Static Text	IDC_STATIC	姓名
Static Text	IDC_STATIC	年龄
Edit Box	IDC_ID	
Edit Box	IDC_NAME	
Edit Box	IDC_AGE	
Button	IDC_TEST	test



2、设计对象的消息处理函数

- 作为对话框工程的主界面，对话框对象需要响应并处理所有的控件消息。设计消息处理函数的步骤：
 - **确定**各个控件要传递的**数据**；
 - 确定对象的**消息映射**
 - 在消息处理函数的函数体内**添加代码**。

对话框和控件之间如何实现数据传递 

- MFC提供标准方法读取或更新控件上的数据
 - **DDX**(Data Exchange, 数据交换)技术
 - **DDV**(Data Validation, 数据校验)技术。

□ DDX通过**成员变量**(member variable) 实现对话框与控件之间的数据传递。

①定义控件的**成员变量**。

②在对应的消息处理函数中通过成员变量访问控件，调用MFC函数传递数据。最简单的MFC函数有：

`UpdateData(TRUE);`//更新成员变量，将控件上的数据传给成员变量

`UpdateData(FALSE);`//更新控件，将成员变量的值传给控件

□ 两类成员变量

- **Control型变量**：可以获得控件的实例，通过这个变量可以操纵控件。**其值实际上就是控件的句柄。**
- **Value型变量**：只用于传递数据，不能对控件进行其它的操作。**该变量的值为控件的数据。**

(1)为本例的编辑控件添加对应的Value型变量

控件	控件标识	成员变量	变量类型
Edit Box	IDC_ID	m_ID	CString
Edit Box	IDC_NAME	m_Name	CString
Edit Box	IDC_AGE	m_Age	int

添加成员变量向导 - MFC1

欢迎使用添加成员变量向导

访问(A): public

变量类型(Y): CString

变量名(N): m_ID

☒ 控件变量(O)

控件 ID(I): IDC_ID

控件类型(Y): EDIT

类别(T): Value

最大字符数(Y):

注意：成员变量名的前缀“m_”是MFC的风格

□ 添加成员变量后，ClassWizard作了三方面的修改

■ MFC1Dlg.h：自动添加了成员变量的声明

```
public:  
    CString m_ID;  
    CString m_Name;  
    int m_Age;
```

字符串类CString：字符串格式化函数是CString::Format，根据格式控制字符串和变量来格式化一个串。

■ MFC1Dlg.cpp文件中的CMFC1Dlg构造函数实现中，自动添加了成员变量的初始代码：

```
CMFC1Dlg::CMFC1Dlg(CWnd* pParent /*=NULL*/)  
    : CDialog(CMFC1Dlg::IDD, pParent)  
    , m_Name(_T(""))  
    , m_ID(_T(""))  
    , m_Age(0)
```

成员变量属于哪个控件?

□ 在MFC1Dlg.cpp文件中的DoDataExchange函数体内，自动添加了控件的DDX/DDV代码。

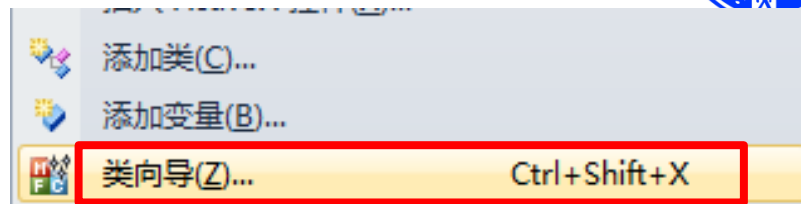
```
void CMFC1Dlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Text(pDX, IDC_NAME, m_Name);
    DDX_Text(pDX, IDC_ID, m_ID);
    DDX_Text(pDX, IDC_AGE, m_Age);
}
```

(2) 在CMFC1Dlg::OnInitDialog()中手动添加如下代码

```
m_ID="1001";
m_Name="Zhang";
m_Age=20;
UpdateData(false);//FALSE表示数据从变量传给控件
```

(3)为控件添加消息处理函数

□多种方法:

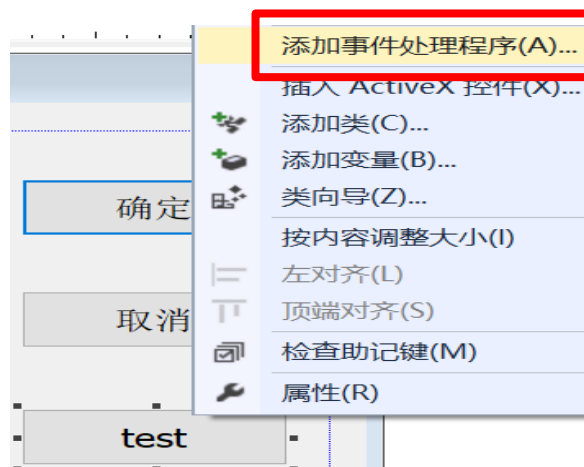


1)使用Class Wizard添加消息处理函数

2)通过“添加事件处理程序”添加消息处理函数

3)在按钮的属性视图中添加消息处理函数

4)双击按钮添加消息处理函数



当用户单击按钮控件时 BN_CLICKED

□ MFC通过给按钮控件“**添加事件处理程序**”为BN_CLICKED消息映射作了三方面的安排:

■ **MFC1Dlg.h文件:** 自动声明消息处理函数OnBnClickedTest()

```
public:  
    afx_msg void OnBnClickedTest();
```

■ **MFC1Dlg.cpp文件:** 自动在前面的消息映射入口处添加相应的消息映射宏

```
ON_BN_CLICKED(IDC_TEST, &CMFC1Dlg::OnBnClickedTest)
```

表示按下IDC_TEST时, MFC会调用OnBnClickedTest函数处理

- **MFC1Dlg.cpp文件：自动**写入一个空的消息处理函数的模板，以使用户填入具体代码

```
void CMFC1Dlg::OnBnClickedTest ()  
{  
    // TODO: 在此添加控件通知处理程序代码  
}
```

(4) 在CMFC1Dlg::OnBnClickedTest ()中手工添加代码

```
CString str;  
str.Format(_T("年龄:%d"),m_Age); //Format是格式化输入。  
str=_T("学号:") + m_ID + _T("\n姓名:") + m_Name + _T("\n") + str;  
MessageBox(str);
```

```

CMFC1Dlg
BOOL CMFC1Dlg ::OnInitDialog( )
{
    ...
    UpdateData(false);
}
void CMFC1Dlg ::OnBnClickedTest( )
{
    ...
    MessageBox(str);
}
    
```

①

②

③



MFC1

学号 1001 确定

姓名 Zhang 取消

年龄 20 test



MFC1

学号 1001 确定

姓名 Zhang 取消

年龄 20 test

MFC1

学号:1001
姓名:Zhang
年龄:20

确定

如何实时反映用户输入 



两个问题:

1、在何处添加代码?

CMFC1Dlg::OnBnClickedTest ()

2、添加什么代码?

UpdateData(true);

```
void CMFC2Dlg::OnBnClickedtest()
{
    // TODO: 在此添加控件通知处理程序代码
    CString str;
    UpdateData(true);
    str.Format(_T("年龄:%d"), m_age); //Format是格式化输入。
    str = _T("学号:") + m_id + _T("\n姓名:") + m_name + _T("\n") + str;
    MessageBox(str);
}
```

示例：简易计算器

- 1、建立对话框项目，设计界面
- 2、为编辑框添加成员变量



成员变量(V):

控件 ID	类型	成员
<自定义变量>	HICON	m_hIcon
IDC_BUTTON_ADD		
IDC_FIRSTNUM	float	m_FirstNum
IDC_RESULTNUM	float	m_ResultNum
IDC_SECNUM	float	m_SecondNum
IDCANCEL		
IDOK		

- 3、添加按钮相应函数，代码示例：

```
void CSimpleDlg::OnBnClickedButtonAdd()
{
    // TODO: 在此添加控件通知处理程序代码
    UpdateData(true);
    m_ResultNum = m_FirstNum + m_SecondNum;
    UpdateData(false);
}
```