Visual FoxPro 7.0 实例解析

孙立明　张含智　等编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

Visual FoxPro 7.0 实例解析

孙立明　张含智　等 编著

清 华 大 学 出 版 社

(京)新登字 158 号

内 容 简 介

Visual FoxPro 7.0 是 Microsoft 公司推出的最新版本的数据库应用系统。本书以实例的形式详细介绍了 Visual FoxPro 7.0 数据库应用程序的开发过程、使用方法和操作技巧。全书共 14 个实例,分别介绍了 Visual FoxPro 7.0 的新增功能、数据库的使用、各种控件的使用、应用程序的系统开发等内容,每一个实例均包括"实例结果与技术要点"和"实现步骤与技术分析"两个部分。

本书内容丰富、实用性强,适合从事 Visual FoxPro 应用和开发的技术人员及大中专院校师生参考。

书　　名:Visual FoxPro 7.0 实例解析
作　　者:孙立明　张含智　等
出 版 者:清华大学出版社(北京清华大学学研大厦,邮编 100084)
　　http://www.tup.tsinghua.edu.cn
责任编辑:胡先福
印 刷 者:清华大学印刷厂
发 行 者:新华书店总店北京发行所
开　　本:787×1092 1/16　印张:19　字数:448 千字
版　　次:2002 年 1 月第 1 版　2002 年 1 月第 1 次印刷
书　　号:ISBN 7-302-05088-0/TP·2977
印　　数:0001 ~ 6000
定　　价:28.00 元

前　言

Visual FoxPro 7.0 是 Microsoft 公司推出的 Visual FoxPro 系列软件的最新版本。Visual FoxPro 7.0 继承了以前版本功能强大、界面友好、简单易学的优点，同时又新增和增强了许多先进的功能，为数据库系统的开发提供了更快的速度、更强的能力和更大的灵活性。本书以实例的形式，详细介绍了 Visual FoxPro 7.0 数据库应用程序的开发过程、使用方法和操作技巧。

全书包括预备知识和 14 个实例，从内容上看，可以分为如下 4 个部分。

第 1 部分：预备知识和实例 1，介绍了 Visual FoxPro 的基本语法和 Visual FoxPro 7.0 的新增功能。

第 2 部分：实例 2～6，介绍了 Visual FoxPro 7.0 数据库操作中的自由表、数据库表、查询、视图、SQL 语句以及报表的创建与使用，为后面应用程序开发奠定了数据库操作的基础。

第 3 部分：实例 7～9，这部分共包括 12 个小实例，系统地介绍了 Visual FoxPro 7.0 表单控件的使用、OLE 控件的使用和 API 函数的使用，为后面应用程序开发奠定了程序设计的基础。

第 4 部分：实例 10～14，在第 2、3 部分的基础上，系统地介绍了一个 Visual FoxPro 7.0 应用程序的开发过程，包括应用程序的规划设计、主体功能与辅助功能的设计、程序界面设计以及程序的调试与编译，从而开发出适合用户需求的应用程序。

本书中有关数据库操作的实例均具有相关性，每一个实例总是在前面实例的基础上进行操作；有关程序设计的实例均具有独立性，每一个实例均可以单独运行；而最后一部分，则将这两部分内容有机地结合，从而最终实现 Visual FoxPro 7.0 应用程序的开发。

本书在实例讲解的过程中，不但讲述了各个实例的设计开发过程，而且讲述了在 Visual FoxPro 7.0 的程序开发中应该注意的问题，以及各种命令、函数的使用方法和技巧，使读者能够在阅读中触类旁通、举一反三。

除封面署名外，黄劲杉、巨泽建、何苏浩、李和杰、杨烨、孙立伟、宋烨阳等参与了本书的程序调试、文字录入和校对等工作，在此表示感谢。由于水平所限，书中不足和纰漏之处在所难免，恳求广大读者批评指正。

作　者

2001 年 12 月

预 备 知 识

VFP 提供了为数据库结构和应用程序开发而设计的面向对象环境。它功能强大且易于使用，是开发数据库管理系统的优秀语言。VFP 支持交互方式和程序方式两种使用方式，下面先主要介绍程序设计中的核心语法，交互方式将在以后的实例中介绍。

☞ 数据类型与字段类型

1. 数据类型

VFP 程序设计中的数据类型如表 0.1 所示。

表 0.1　VFP 的数据类型

数据类型	说明
Character （字符型）	由字母、数字、空格、符号、标点和汉字等组成，在定义时要用引号（在 VFP 中使用单引号和双引号均可）
Currency （货币型）	保存有关货币计量的数据。货币型数据可以自动控制小数点的位数，当所赋予的数值的小数点的位数超过 4 位时，VFP 将在运算之前将其自动作舍入处理。货币型数据的格式可以使用 SET CURRENCY 进行设置
Date （日期型）	存储有关年、月、日的信息，通常用大括号括起来，如｛12/31/00｝，在内存中的存储格式为“YYYYMMDD”，年份占 4 个字节，月份和日各占 2 个字节，即｛12/31/00｝与｛12/31/2000｝所表达的信息完全相同。日期型数据的格式可以使用 SET DATE、SET MARK、SET CENTURY 等系统命令来设置
DataTime （日期时间型）	同时包括日期和时间，也可以只包含两者之一，如｛12/31/00 12：00AM｝，日期时间型数据占有 8 个字节，前 4 个字节用整数型来存储日期，后 4 个字节用整数型来存储时间。对日期部分的格式设定和日期型数据相同，而对时间部分的格式可以使用 SET HOURS、SET SECONDS 来设置
Logical （逻辑型）	具有两个布尔值 .T. 和 .F.。用户定义的变量、数组在默认的情况下都是逻辑型，值为 .F.
Numeric （数值型）	表示一个整数或者实数，在内存中占有 8 个字节，取值范围为： -0.9999999999E+19 ~ 0.9999999999E+20
Object （对象型）	作为一个对象的名称，惟一地表示该对象。可以使用一些特殊的命令来创建，如：poMyform = CREATEOBJECT（"FORM"）
Varient （不定型）	能存储任意类型的数据，一旦把数据赋值给一个不定型的变量，该变量的类型即变为该数据的类型

2. 字段类型

VFP 程序设计中的字段类型如表 0.2 所示。

表 0.2 VFP 的字段类型

数据类型	字段简写	说明
Character（字符型）	C	字符是大多数数据表中常用的字段，由字母、数字、空格、符号、标点和汉字等组成，某些特定的字符（如回车键）不能出现在普通的字符型字段中。字符型字段可以存储 1 到 256 个字符，如果需要更多的字符，则需要定义备用字段
Character（binary）（二进制字符型）	C	将数据存储为二进制格式
Currency（货币型）	Y	当需要保存有关货币计量的数据时，使用货币型字段。在数据表中需要 8 个字节的存储空间，默认保留 4 位小数
Date（日期型）	D	与数据类型中的日期型数据一致
DataTime（日期时间型）	T	与数据类型中的日期时间型数据一致
Double（双精度型）	B	需要 8 个字节的存储空间，采用压缩格式最多 18 位数字，用户只能决定小数点后位数。取值范围为：-4.94065648541247E+324 ~ -1.79769313486232E+308
Integer（整型）	I	在表中以二进制存储，需要 4 个字节的存储空间，取值范围为：-2147483647 ~ 2147483646
Float（浮点型）	F	支持最多 19 位小数的 20 位数字，每个数字需要一个字节的存储空间，用户可以指定字段所需字节的数目，如果字段的长度相对于需要存储的值太小了，VFP 将显示“*”。取值范围为：-0.9999999999E+19 ~ 0.9999999999E+20
General（常规型）	G	该字段是专门的 Memo 字段，VFP 把 Memo 字段和 General 字段存储在与数据表同名的 .FPT 文件中，但是两者的使用方法不同。General 型字段最常见的用途是用来存储图形
Logical（逻辑型）	L	该字段以 .T. 或者 .F. 的格式存储二进制信息
Numeric（数值型）	N	支持最多 19 位小数的 20 位数字，每个数字需要一个字节的存储空间，用户可以指定字段所需字节的数目，如果字段的长度相对于需要存储的值太小了，VFP 将显示“*”。取值范围为：-0.9999999999E+19 ~ 0.9999999999E+20
Memo（备注型）	M	存储超过 256 个字符的大型字符串。该字段实际上是一个 4 字节的引用，指向实际的备注内容。备注内容存储在与数据表同名的 .FPT 文件中，其存储长度只受磁盘空间大小的限制
Memo（binary）（二进制备注型）	M	将数据存储为二进制格式

变量

在命令操作和程序运行过程中其值允许变化的量称为变量。变量包括内存变量、字段变量和系统内存变量3种。

1. 内存变量

内存变量可以用来存储数据，定义内存变量时需要为它取名并赋值，内存变量建立后存储于内存中。

(1) 命名规则

以字母(汉字也可以)或者下划线开头，由字母、数字和下划线组成，至多128个字符，不可与系统保留字同名。

(2) 赋值语句

语法1：

```
VarName = Expression
```

功能:将表达式的值赋给变量。

例如：

```
lcTemp = 'VFP7' && 字符串 VFP7 赋给变量 lcTemp,lcTemp 成为字符型变量
```

语法2：

```
STORE Expression TO VarNameList | ArrayNameList
```

功能:将表达式的值赋给变量列表或者数组列表。其中"|"表示"或者"。

例如：

```
STORE 6 * 6 TO lnNum1,lnNum2,lnNum3
*!计算6*6得36并赋给这3个变量,使它们成为数值型变量,值为36
```

"&&"为注释符号，表示符号后面的字符均为注释语句。"*!"(其实一个"*"即可，一般为了与运算符号"*"和"**"区分，用户可以再添加特殊的字符)也为注释符号，但是只能另起一行注释，不能跟在命令语句的后面。

如果某一行的后面有";"，表示下一行的内容为该行内容的后续。

(3) 显示变量的值

语法：

```
? | ?? Expression1 [PICTURE cFormatCodes] |
 [FUNCTION cFormatCodes] | [VnWidth] [AT nColumn]
```

```
[FONT cFontName [, nFontSize]
 [STYLE cFontStyle | Expression2]][, Expression3] ...
```

功能:该命令首先计算变量或者表达式的值,然后按照指定的格式将结果显示在屏幕上。其中"[]"中的内容为可选择内容。

例如:

```
? lcTemp    && 在 VFP 主窗口显示 lcTemp 的值:VFP7
?? '实用 VFP7 案例分析'
*!紧接在上一命令的显示结果"VFP7"的后面显示:实用 VFP7 案例分析
```

2. 数组

数组是按一定顺序排列的一组内存变量,数组中的各个变量称为数组元素,数组必须先定义后使用。

(1) 数组的定义

语法:

```
DIMENSION | DECLARE ArrayName1 (nRows1 [, nColumns1])
[, ArrayName2 (nRows2 [, nColumns2])] ...
```

功能:定义一维或者多维数组,并定义其下标的上界。

例如:

```
DIMENSION   laName(6)   && 定义一个有 6 个元素的一维数组
DECLARE   laAge(3,6)   && 定义一个 3 行 6 列的二维数组。
```

(2) 数组的赋值

在一般的高级语言中,同一数组各元素的类型必须相同。但是 VFP 允许同一数组的元素取不同的类型,而且同一个元素的前、后类型也允许改变。在定义数组时,系统将数组元素的初值设置为.F.。用赋值命令可以为数组元素进行单个赋值,也可以整体赋相同的值。

3. 内存变量的作用域

VFP 的变量遵循传统的作用域规定方式,变量可以由生成它的应用程序使用,也可以由下一级程序使用,可以使用 LOCAL、PRIVATE 和 PUBLIC 关键字指定变量的作用域。

(1) LOCAL 关键字

用于声明内存变量和内存变量数组为局部变量。用 LOCAL 声明的变量或者数组只能在创建它们的程序中使用,不能被更高层或者低层的程序访问。在它们的所属程序停止运行时,局部变量将被释放。如果变量没有使用作用域关键字声明,而直接定义并赋值,系统默认为局部变量。

(2) PRIVATE 关键字

用于声明内存变量和内存变量数组为私有变量。用 PRIVATE 声明的变量或者数组

将会隐藏在高层程序中创建的、与它们同名的变量，在当前程序运行这些私有变量，不会影响被隐藏的变量的值。在它们所在的程序执行完毕后，与它们同名的高层程序中的变量将恢复使用。

（3）PUBLIC 关键字

用于声明内存变量和内存变量数组为全局变量。用 PUBLIC 声明的变量或者数组对整个应用程序都是可以使用的。在命令窗口中创建的任何内存变量和数组都自动设置为全局变量。

4. 内存变量命名的约定规则

为了防止变量命名出现重名，同时增进在多人合作时阅读程序的方便，VFP 内存变量有一定的约定规则：变量由两个小写字母开头，第一个表示作用域，第二个表示数据类型，其后跟一个长度合理的名字。

表 0.3 和表 0.4 列出了作用域和数据类型的前缀字符。

表 0.3　作用域的前缀字符

作用域	前缀	例子
局部	L	lcName
私有	P	pnAge
全局	G	gnNumber

表 0.4　数据类型的前缀字符

数据类型	前缀	例子
数组	A	paArray（6）
双精度型	B	lbGrade
字符型	C	lcName
日期型	D	pdDate
浮点型	F	lfTotalNum
通用型	G	lgPicture
整数型	N	lnNumber
对象型	O	poApp
日期时间型	T	ptDateTime
货币型	Y	pyCurrency

5. 字段变量

表的每一个字段都是一个字段变量。认为字段是变量，主要是由于对于某一个字段，它的值允许因记录的变化而变化。

例如：

```
USE grade
```

```
? 课程名称
*! grade.dbf 文件打开之后,记录指针指向第一记录,显示该记录的字段内容
GO 5    && 将记录指针指向第 5 个记录
? 课程名称    && 显示第 5 个记录的该字段的内容
*! 关于操作表的详细内容,请参阅实例 2
```

如果字段变量与内存变量同名，而且字段变量又没有采用表别名作为前缀时，那么使用这个变量时，系统将先检查当前工作区中的字段变量，若没有这样的字段，再检查内存变量。

但是字段变量的值只能由 REPLACE 和 UPDATE 命令更改，而不能由赋值语句或者 STORE 命令更改。当一个变量名出现在赋值语句的左边或者作为 STORE 命令操作对象时，系统仅检查内存变量。

6. 系统变量

系统变量是 VFP 字段创建并维护的内置内存变量。默认的情况下，这些变量的属性为 PUBLIC，也可以声明为 PRIVATE，用户可以在程序中运用并更改这些变量，但是不能使用内存清除命令如 CLEAR MEMO 或者 CLEAR ALL 清除这些变量。所有的系统变量都以下划线开始。

例如：

```
SCREEN    && 控制系统屏幕显示属性的变量
DIARYDATE    && 存储当前日期
```

操作符

1. 字符、日期操作符

（1）字符操作符

可以使用“+”、“-”和“&”对字符进行连接和比较等操作。

例如：

```
'my favorite' + 'book'
*! 结果为:'my favorite book',表示两个字符串的简单连接
'my favorite '  'book'
*! 结果为:'my favoritebook ',表示两个字符串连接,但是将前一个字符
*! 串末尾的空格移到了结果字符串的末尾。
'VFP' $ 'VFP7' && 结果为:.T.
*!  判断右边的字符串中是否含有左边的字符串
```

（2）日期操作符

日期操作符适用于日期时间型变量。“+”表示数据相加，“-”表示数据相减。

例如：

```
SET STRICTDATE TO 0
*！使用通常的日期格式，不同的日期格式对应不同的日期输入方式。
{07/31/01} + 31  && 结果为:{08/31/01}
{07/31/01} - 61  && 结果为:{05/31/01}
```

2. 算术、关系、逻辑操作符

这3种操作符如表0.5所示。优先级越高(即数值越大)，表示运算时越优先进行。

表0.5 数值、关系、逻辑操作符

操作	优先级	操作符	说明
算术	8	()	圆括号
	7	^或 * *	乘方
	6	*、/	乘、除
	5	+、-	加、减
关系	4	<、<=	小于、小于等于
		>、>=	大于、大于等于
		=	相等:在SET EXACT OFF(默认)的情况下，比较两个字符串时串首相同就是真，若SET EXACT ON，则与==相同
		==	完全相等
		<>、#或！=	不相等
逻辑	3	NOT或！	非
	2	AND	与
	1	OR	或

程序结构

1. 条件结构

(1) IF条件语句

语法：

```
IF Expression [THEN]
   Commands1
[ELSE
   Commands2]
ENDIF
```

功能：当表达式的值为真时，执行语句1，否则执行语句2。如果不需要处理ELSE的情况，可以省略ELSE部分。

(2) DO CASE语句

语法：

```
DO CASE
  CASE Expression1          Commands
  [CASE Expression2         Commands
   ...
  CASE ExpressionN          Commands]
  [OTHERWISE                Commands]
ENDCASE
```

功能：当程序遇到第一个为.T.的表达式时，执行该表达式后面的语句，然后跳出DO CASE语句，执行其他的语句。如果所有的表达式的值都为.F.，则执行OTHERWISE后面的语句。

2. 循环结构

（1）FOR循环语句

语法：

```
FOR Var = nInitialValue TO nFinalValue [STEP nIncrement]  Commands
[EXIT]
[LOOP]
ENDFOR | NEXT
```

功能：当计数器的值在初值和终值之间时执行循环中的语句。Var指定作为计数器的内存变量，nInitialValue为计数器的初值，nFinalValue为计数器的终值，STEP nIncrement指定步长。EXIT子句表示跳出FOR循环，LOOP子句表示不再执行LOOP与ENDFOR之间的语句，直接进入下一轮循环。

（2）DO…WHILE循环语句

语法：

```
DO WHILE Expression
 Commands
 [LOOP]
 [EXIT]
ENDDO
```

功能：当表达式的值为.T.时，执行循环中的语句。LOOP子句与EXIT子句的功能与FOR语句一致。

（3）SCAN循环语句

语法：

```
SCAN [NOOPTIMIZE]  [Scope]
 [FOR lExpression1] [WHILE lExpression2]
 [Commands]
  [LOOP] [EXIT]
ENDSCAN
```

功能：针对当前表进行循环，找到满足FOR或者WHILE条件的记录，对这些记录执行循环中的语句。NOOPTIMIZE子句表示阻止使用RUSHMORE优化技术，Scope子句

表示记录范围，默认为 ALL。

函数与过程

1. 函数

函数是用来计算并返回一个值的一段程序代码。VFP 提供了 200 多种函数，具有强大的功能。

（1）函数的定义

```
FUNCTION FunctionName
[ParameterList]
Commands
[RETURN [eExpression]]
ENDFUNC
```

其中，FunctionName 为函数名称，ParameterList 为参数列表，RETURN［eExpression］为返回值。

（2）函数的调用

```
DO ProgramName1 | ProcedureName  [IN ProgramName2]
  [WITH ParameterList]
```

其中，ProgramName1 是要执行的程序名称，ProgramName2 是包含要执行程序的程序文件名。如果找不到程序文件，系统会显示“File does not exist”，如果找到了文件，但是找不到要执行的程序，系统会显示“Procedure not found”。

函数也可以采用直接在函数名的后面加上一对小括号，并在括号中输入需要的参数的方式来调用。

（3）VFP7 中的常用函数

- 数据库操作函数：用来处理数据库、表、表中的字段、查询以及视图等的函数。如 EOF( )同函数可以用来测试一个表的记录是否在表尾，ATAGINFO( ) 函数可以返回一个包含表的索引、关键字的名称、数量和类型信息的数组。
- 数值处理函数：用来进行数值运算的函数。如 SIN( )、SQRT( )等函数。
- 字符处理函数：用来处理字符型数据的函数。如 ALLTRIM( )函数，可以删除字符串前导和末尾的函数，GETWORDNUM( )可以返回指定位置的字符。
- 数据转换函数：把一种类型的数据转换成为另外一种类型的数据。如 STR( )函数可以把一个数值型的数据转换成字符型。
- 环境设置函数：用来管理 VFP7 的系统和环境参数。如 HOME( )函数可以返回启动 VFP7 的目录名，SYS(2003)可以返回当前工作路径，SYS(2800)可以管理 ActiveX 控件的可操作程度并设置必要的选项来跟踪在 VFP7 表单中所选控件的键盘焦点。

2. 过程

过程是作为一个相对独立的单位而工作的一组指令或者子程序。

（1）过程的定义

```
PROCEDURE ProcedureName
[ParameterList ]
 Commands
 [RETURN [eExpression ]]
ENDPROC
```

过程的定义与函数的定义基本相同。一般情况下，过程只是完成一定的操作，而不需要返回值。

（2）过程的调用

过程的调用如同函数调用一样，采用 DO 语句来调用过程。但是当过程作为一个独立的文件存放在磁盘上时，其他的程序要调用该过程，需要在调用这个过程之前，执行 SET PROCEDURE TO 命令将过程所在的文件装入内存当中。

实例 1　VFP7 新增功能

实例结果与技术要点

实例结果

VFP7 继承了以往版本功能强大、界面友好、简单易学的优点，同时又比以往的版本具有了更多更先进的功能。本实例将主要介绍 VFP7 的新增功能。

VFP7 的主窗口界面如图 1.1 所示。

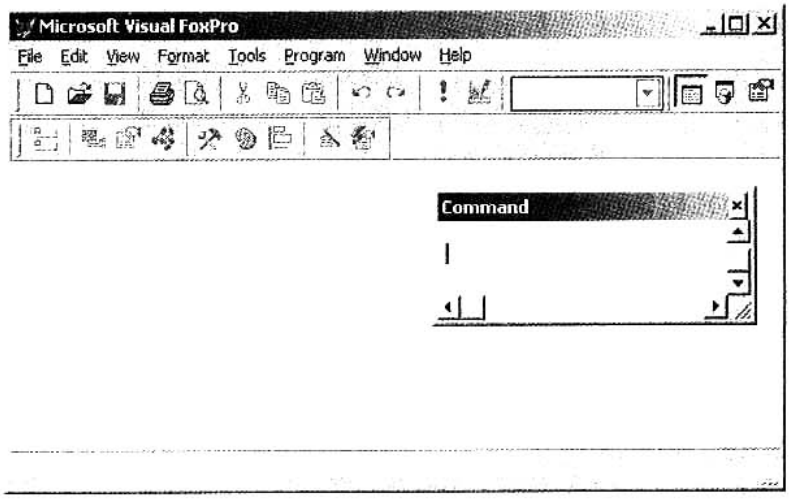

图 1.1　VFP7 的主窗口

技术要点

本实例将介绍 VFP7 提供的几个主要新增功能：

- 智能感知技术
- IDE 功能的增强
- 语言功能的增强
- DBC 事件

实现步骤与技术分析

智能感知功能

VFP7 提供了新的智能感知功能，包括：

- 自动列出成员
- 自动显示快速信息
- 自动显示变量值列表
- 自动列出表的字段名称
- 自动显示 MRU(Most Recently Used，最近使用的)文件列表

这些功能能够自动填写声明、属性、参数，从而使得编写代码变得更容易，减少了程序员必须输入的代码数量，而且程序员不再需要去参考文档里查找所需函数的参数。

1. 自动列出成员

该功能可以为一个专门的对象提供一个显示有效成员(包括工具、方法、事件、对象)的参考列表。对 COM 对象，信息是从数据类型库中读取的。该功能可由 Edit(编辑)菜单中 List Members(列出成员)菜单项激活，或者按快捷键 Ctrl + J。当输入一个对象、属性、有效的 VFP7 命令或者函数时，如果输入触发键就会显示一个下拉列表，列出当前对象的有效成员。当继续输入或者通过列表选择时，该功能还会显示所选项目的提示性描述窗口。

如果用户进行了下述操作：

- 输入所有需要的参数
- 按下一个终止键(Esc、End 或者无效键)
- 改变当前活动窗口

该功能显示的下拉列表会关闭，否则会一直打开。无论是在 VFP7 的编辑器、命令窗口还是在所有的局部编辑器中，只要可以输入 VFP7 的代码，该功能就有效。在输入类方法或者对象时，触发键一般是"."键；当输入命令或函数时，触发键是空格键或者是左括号。如图 1.2 所示。

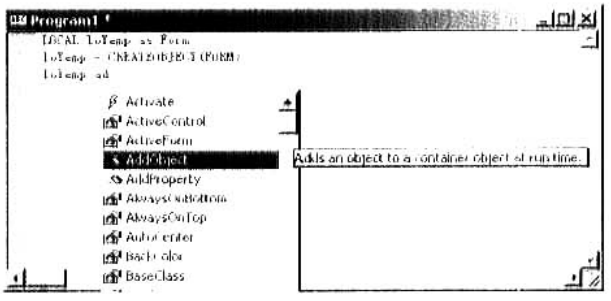

图 1.2 自动列出成员

要在代码编写窗口(如图 1.2 所示)或者 COM 服务器中使用智能感知功能，首先必须严格地定义对象实例，如采用图 1.2 中的语句“LOCAL x as Form”，而不能直接使用“Form”(虽然这样已经可以使用智能感知的功能，但是代码却是无法编译通过的)。然后必须创建对象实例，如采用 CREATEOBJECT( )、CREATEOBJECTX( )、NEWOBJECT( )和 GETOBJECT( )等命令。智能感知功能将为该对象实例提供其属性、方法和事件的下拉列表，这些内容将从 VFP7 的下述位置获得：

- VFP7 对象的基类
- 内存中定义的类
- 当前程序定义的类
- 由 SETCLASS 命令打开的 .vcx 文件中定义的类
- 由 SET PROCEDURE 命令打开的过程中定义的类
- 由 VFP 程序执行序列中定义的类
- 系统注册表中的数据库

2. 自动显示快速信息

该功能可以为命令、函数、属性、方法和事件显示一个提示性的数据项窗口。在窗口中的信息包括一个函数参数或者命令参数的列表，还有它们的数据类型。该功能可由 Edit 菜单中 Quick Info(快速信息)菜单项激活，或者按快捷键 Ctrl + I。当输入一个有效的 VFP7 命令或者函数的名称时，它们的语法规则和第一个参数就会立即以黑体字在当前行的下面显示出来。当用户输完一个变量，下一个变量就会以黑体字显示出来。该功能触发键为“(”。如图 1.3 所示，当输入结束并输入右括号时，与之对应的左括号之间的内容背景就会暂时性地变为黑色同时字体反显，提示用户这一对括号间的内容已经输入完毕，如果有多对括号，则每对括号输入结束都会出现上述的提示。反显时间可以通过选择 Tools(工具)菜单中的 Options(选项)菜单项，在弹出的对话框中进行设置，在后面的章节中将进一步介绍。

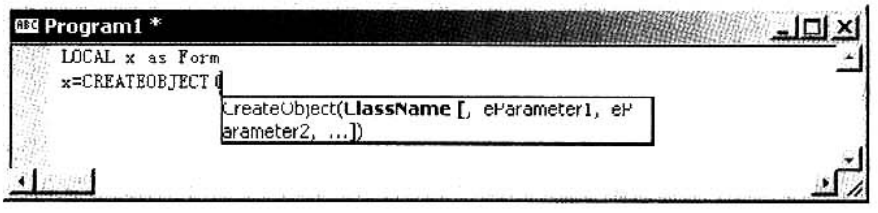

图 1.3　自动显示快速信息