

FOXPRO 2.5

命令与函数范例解析

(台)章立民 编著
科 培 改编



清华大学出版社

北京科海培训中心

FoxPro 2.5

命令与函数范例解析

(松岗) 章立民 编著
科 培 改编

清华大学出版社

FoxPro 2.5 for Windows

命令与函数范例解析

章立民 编著

本书由台湾松岗电脑图书资料股份有限公司出版,1993。本书中文简体字版经松岗公司授权北京科海培训中心改编并与清华大学出版社合作出版,1993。未经出版者书面允许,不得用任何手段复制或抄袭本书内容。

(京)新登字 158 号

FoxPro 2.5

命令与函数范例解析

(松岗)章立民 编著

科 培 改编

☆

清华大学出版社出版

北京 清华园

门头沟胶印厂印刷

新华书店总店科技发行所发行

☆

开本:787×1092 1/16 印张:34.5 字数:839 千字

1993 年 12 月第 1 版 1993 年 12 月第 1 次印刷

印数:0001~5000 册

ISBN 7-302-01462-0/TP·577

定价:48.00 元

JS425/09

序

FoxPro 2.5 是目前世界上最快速的 PC 机数据库,其速度虽不敢说是“绝后”,但确实是非常快。FoxPro 2.5 对索引式数据的特定查询之所以会比同级的数据库快百倍甚至千倍,主要是利用名为 Rushmore 的快速数据查询技术。而且数据记录的数目愈多,就愈能感受到 FoxPro 2.5 速度快的优点。

除了在速度上的表现令人惊叹之外,在用户接口的设计上,FoxPro 2.5 for MS-DOS 可以说是文本模式(Text Mode)中的典范。而 FoxPro 2.5 for Windows 更能充分发挥图形界面的特色。用户会发现它不仅美观且颇具智慧!正如您所期望的,OLE(对象连接与嵌入)及 DDE(动态数据交换)在 FoxPro for Windows 中都可发挥得淋漓尽致。此外,如众多程序设计人员所期盼的,一个可产生可执行文件. EXE 的编译与连接程序也包含在 FoxPro 2.5 for MS-DOS/Windows 中,而您所熟悉的用户自定义窗口及用户自定义菜单更作了强化的工作。凡此种种在本书中都将有详细的介绍。

开发 FoxPro 的 Microsoft 公司已表示,未来将陆续推出 FoxPro for Unix 及 FoxPro for Mac,且已着手设计开发,因此可以预见,未来 FoxPro 将普及并得到广泛应用。

本书的习惯用法

为了使读者在阅读本书时能迅速掌握以达事半功倍之效,特将书中各个命令与函数语法的惯用表示式列示如下:

1. 所有冠以中括号([])的命令或函数表示可选择的内容(参数或关键字)。
2. 在一命令中以符号“|”分隔的两个选项(例如 ON|OFF),表示一次只能使用两者之一,而不能同时使用。

3.

<expC>	:表示字符表达式
<expN>	:表示数值表达式
<expL>	:表示逻辑表达式
<expD>	:表示日期表达式
<expr>与<expression>	:表示不限定类型的表达式

4. 所有命令皆以大写列示,但在使用时大、小写皆可。
5. 每当一个函数允许或需要一个 <alias> 的参数时,可以使用别名(alias name)、工作区英文字母代号(A~J)、工作区的数字编号(1~225)。
6. 若是在一个参数后面加上三个句点“...”,表示此参数可以无限重复使用,直到用完计算机所有的内存或是达到单一程序行最多只能有 2048 个字符的限制。

目 录

第一章 命令篇

#DEFINE... #UNDEF	(1)	APPEND MEMO	(85)
#IF... #ENDIF	(3)	AVERAGE	(85)
RUN !	(4)	BROWSE	(87)
\$	(7)	BUILD APP	(101)
%	(7)	BUILD PROJECT	(101)
&	(8)	CALCULATE	(101)
=	(9)	CALL	(103)
\ \ \	(10)	CANCEL	(103)
? ??	(11)	CHANGE	(103)
???	(13)	CLEAR	(105)
@... SAY/GET	(14)	CLEAR ALL	(105)
@... GET-Check Boxes	(26)	CLEAR FIELDS	(106)
@... GET-Invisible Buttons	(32)	CLEAR GETS	(106)
@... GET-Lists	(37)	CLEAR MACROS	(106)
@... GET-POPUPS	(41)	CLEAR MEMORY	(106)
@... GET-Push Buttons	(46)	CLEAR MENUS	(106)
@... GET-Radio Buttons	(54)	CLEAR POPUPS	(106)
@... GET-Spinners	(60)	CLEAR PROGRAM	(106)
@... EDIT	(65)	CLEAR PROMPTS	(107)
@... BOX	(71)	CLEAR READ	(107)
@... CLEAR TO	(71)	CLEAR TYPEAHEAD	(107)
@... FILL TO	(71)	CLEAR WINDOWS	(107)
@... MENU	(71)	CLOSE [ALL]	(107)
@... PROMPT	(72)	CLOSE MEMO	(107)
@... TO	(74)	COMPILE	(108)
ACCEPT	(76)	CONTINUE	(109)
ACTIVATE MENU	(76)	COPY FILE	(109)
ACTIVATE POPUP	(77)	COPY INDEXES	(109)
ACTIVATE SCREEN	(78)	COPY MEMO	(110)
ACTIVATE WINDOW	(78)	COPY STRUCTURE	(110)
APPEND	(79)	COPY STRUCTURE EXTENDED	(110)
APPEND FROM	(79)	COPY TAG	(111)
APPEND FROM ARRAY	(83)	COPY TO	(112)
APPEND GENERAL	(84)		

COPY TO ARRAY	(113)	DO WHILE... ENDDO	(169)
COUNT	(114)	EDIT	(170)
CREATE	(115)	EJECT	(171)
CREATE COLOR SET	(115)	EJECT PAGE	(172)
CREATE CURSOR	(115)	ERASE	(172)
CREATE FROM	(116)	EXIT	(173)
CREATE LABEL	(116)	EXPORT	(174)
CREATE MENU	(117)	EXTERNAL	(174)
CREATE PROJECT	(118)	FILER	(175)
CREATE QUERY	(119)	FIND	(177)
CREATE REPORT	(120)	FLUSH	(177)
CREATE REPORT - Quick Report	(121)	FOR... ENDFOR	(178)
CREATE SCREEN	(121)	FUNCTION	(179)
CREATE SCREEN - Quick Screen	(124)	GATHER FROM	(180)
CREATE TABLE	(125)	GETEXPR	(183)
CREATE VIEW	(126)	GO/GOTO	(184)
DEACTIVATE MENU	(126)	HELP	(184)
DEACTIVATE POPUP	(127)	HIDE MENU	(185)
DEACTIVATE WINDOW	(128)	HIDE POPUP	(186)
DECLARE	(130)	HIDE WINDOW	(187)
DEFINE BAR	(131)	IF... ENDIF	(187)
DEFINE BOX	(137)	IMPORT	(189)
DEFINE MENU	(138)	INDEX	(190)
DEFINE PAD	(141)	INPUT	(194)
DEFINE POPUP	(146)	INSERT	(195)
DEFINE WINDOW	(153)	INSERT INTO	(195)
DELETE	(161)	JOIN	(196)
DELETE FILE	(161)	KEYBOARD	(199)
DELETE TAG	(161)	LABEL	(201)
DIMENSION	(162)	LIST	(202)
DIR DIRECTORY	(163)	LIST FILES	(203)
DISPLAY	(164)	LIST MEMORY	(203)
DISPLAY FILES	(164)	LIST STATUS	(204)
DISPLAY MEMORY	(165)	LIST STRUCTURE	(205)
DISPLAY STATUS	(165)	LOAD	(205)
DISPLAY STRUCTURE	(165)	LOCATE	(206)
DO	(166)	MENU	(206)
DO CASE	(167)	MENU TO	(210)
		MODIFY COMMAND/MODIFY FILE	(211)

MODIFY GENERAL	(213)	QUIT	(256)
MODIFY LABEL	(215)	READ	(256)
MODIFY MEMO	(216)	READ MENU	(258)
MODIFY MENU[<file> ?]	(218)	RECALL	(258)
MODIFY PROJECT	(218)	REGIONAL	(259)
MODIFY QUERY	(219)	REINDEX	(261)
MODIFY REPORT	(219)	RELEASE	(261)
MODIFY SCREEN	(220)	RELEASE MODULE	(262)
MODIFY STRUCTURE	(220)	RENAME	(262)
MODIFY WINDOW	(220)	REPLACE	(263)
MOVE POPUP	(222)	REPORT	(264)
MOVE WINDOW	(223)	RESTORE FROM	(265)
NOTE * &&	(224)	RESTORE MACROS	(266)
ON BAR	(224)	RESTORE SCREEN	(266)
ON ERROR	(226)	RESTORE WINDOW	(267)
ON ESCAPE	(228)	RESUME	(268)
ON KEY	(229)	RETRY	(268)
ON KEY =	(230)	RETURN	(269)
ON KEY LABEL	(231)	RUN !	(270)
ON PAD	(236)	SAVE MACROS	(273)
ON PAGE	(238)	SAVE SCREEN	(273)
ON READERROR	(238)	SAVE TO	(274)
ON SELECTION BAR	(239)	SAVE WINDOWS	(274)
ON SELECTION MENU	(240)	SCAN...ENDSCAN	(275)
ON SELECTION PAD	(241)	SCATTER	(276)
ON SELECTION POPUP	(242)	SCROLL	(279)
ON SHUTDOWN	(244)	SEEK	(279)
PACK	(245)	SELECT	(280)
PARAMETERS	(245)	SET	(281)
PLAY MACRO	(246)	SET ALTERNATE	(281)
POP KEY	(246)	SET ANSI	(281)
POP MENU	(247)	SET AUTOSAVE	(282)
POP POPUP	(247)	SET BELL	(282)
PRINTJOB...ENDPRINTJOB ...	(247)	SET BELL TO	(282)
PRIVATE	(248)	SET BLINK	(282)
PROCEDURE	(249)	SET BLOCKSIZE	(283)
PUBLIC	(250)	SET BORDER	(283)
PUSH KEY	(251)	SET BRSTATUS	(283)
PUSH MENU	(255)	SET CARRY	(284)
PUSH POPUP	(256)	SET CENTURY	(284)

SET CLEAR	(284)	SET HOURS	(297)
SET CLOCK	(284)	SET INDEX	(297)
SET CLOCK TO	(285)	SET INTENSITY	(299)
SET COLOR OF	(285)	SET KEYCOMP	(299)
SET COLOR OF SCHEME	(285)	SET LIBRARY	(300)
SET COLOR OF SCHEME TO ...	(286)	SET LOCK	(301)
SET COLOR ON OFF	(287)	SET LOGERRORS	(301)
SET COLOR SET TO	(287)	SET MACKEY	(301)
SET COLOR TO	(287)	SET MARGIN	(302)
SET COMPATIBLE	(288)	SET MARK OF MENU	(302)
SET CONFIRM	(288)	SET MARK OF PAD	(305)
SET CONSOLE	(288)	SET MARK OF POPUP	(307)
SET CURRENCY	(288)	SET MARK OF BAR	(309)
SET CURRENCY LEFT RIGHT	(289)	SET MARK TO	(311)
SET CURSOR	(289)	SET MEMOWIDTH	(312)
SET DATE	(289)	SET MESSAGE	(312)
SET DEBUG	(289)	SET MOUSE	(313)
SET DECIMALS	(290)	SET MULTILOCKS	(313)
SET DEFAULT	(290)	SET NEAR	(313)
SET DELETED	(290)	SET NOTIFY	(313)
SET DELIMITERS	(291)	SET ODOMETER	(314)
SET DEVELOPMENT	(292)	SET OPTIMIZE	(314)
SET DEVICE	(292)	SET ORDER	(315)
SET DISPLAY TO	(293)	SET PALETTE	(316)
SET DOHISTORY	(293)	SET PATH	(316)
SET ECHO	(293)	SET POINT	(317)
SET ESCAPE	(294)	SET PRINTER	(317)
SET EXACT	(294)	SET PRINTER TO	(318)
SET EXCLUSIVE	(294)	SET PROCEDURE	(318)
SET FIELDS	(294)	SET REFRESH	(318)
SET FIELDS TO	(295)	SET RELATION	(319)
SET FILTER	(295)	SET RELATION OFF	(320)
SET FIXED	(295)	SET REPROCESS	(321)
SET FORMAT	(295)	SET RESOURCE	(322)
SET FULLPATH	(296)	SET RESOURCE TO	(322)
SET FUNCTION	(296)	SET SAFETY	(322)
SET HEADING	(296)	SET SCOREBOARD	(323)
SET HELP	(296)	SET SEPARATOR	(323)
SET HELPFILTER	(297)	SET SHADOWS	(323)
		SET SKIP	(323)

SET SKIP OF	(324)	SHOW GETS	(336)
SET SPACE	(325)	SHOW MENU	(337)
SET STATUS	(325)	SHOW OBJECT	(338)
SET STATUS BAR	(325)	SHOW POPUP	(339)
SET STEP	(325)	SHOW WINDOW	(339)
SET STICKY	(326)	SIZE POPUP	(342)
SET SYSMENU	(326)	SKIP	(343)
SET TALK	(327)	SORT	(344)
SET TEXTMERGE	(327)	STORE	(345)
SET TEXTMERGE DELIMITERS	(329)	SUM	(345)
SET TOPIC	(330)	SUSPEND	(346)
SET TRBETWEEN	(330)	TEXT...ENDTEXT	(346)
SET TYPEAHEAD	(330)	TOTAL	(347)
SET UDFPARMS	(330)	TYPE	(347)
SET UNIQUE	(332)	UNLOCK	(348)
SET VIEW	(332)	UPDATE	(348)
SET VIEW TO	(332)	USE	(348)
SET WINDOW OF MEMO	(332)	WAIT	(352)
SHOW GET	(333)	ZAP	(353)
		ZOOM WINDOW	(354)

第二章 函数篇

ABS()	(355)	AT()	(370)
ACOPY()	(355)	ATAN()	(371)
ACOS()	(356)	ATC()	(371)
ADEL()	(356)	ATCLINE()	(372)
ADIR()	(358)	ATLINE()	(372)
AELEMENT()	(360)	ATN2()	(373)
AFIELDS()	(361)	BAR()	(373)
AFONT()	(362)	BETWEEN()	(374)
AINS()	(364)	BOF()	(374)
ALEN()	(365)	CAPSLOCK()	(378)
ALIAS()	(366)	CDOW()	(379)
ALLTRIM()	(366)	CDX()	(379)
ASC()	(366)	CEILING()	(380)
ASCAN()	(367)	CHR()	(380)
ASIN()	(367)	CHRS AW()	(380)
ASORT()	(368)	CHRTRAN()	(381)
ASUBSCRIPT()	(370)	CMONTH()	(382)

CNTBAR().....	(382)	FILE()	(420)
CNTPAD().....	(384)	FILTER()	(421)
COL()	(385)	FKLABEL()	(421)
COS()	(385)	FKMAX()	(421)
CTOD()	(385)	FLOCK()	(421)
CURDIR()	(386)	FLOOR()	(423)
DATE()	(386)	FONTMETRIC()	(423)
DAY()	(387)	FOPEN()	(424)
DBF()	(387)	FOUND()	(426)
DDEAbortTrans()	(387)	FPUTS()	(427)
DDEAdvise()	(388)	FREAD()	(429)
DDEEnabled()	(393)	FSEEK()	(429)
DDEExecute()	(394)	FSIZE()	(430)
DDEInitiate()	(395)	FULLPATH()	(431)
DDELastError()	(396)	FV()	(431)
DDEPoke()	(398)	FWRITE()	(431)
DDERequest()	(402)	GETBAR()	(432)
DDESetOption()	(403)	GETDIR()	(434)
DELETED()	(403)	GETENV()	(434)
DIFFERENCE()	(404)	GETFILE()	(435)
DISKSPACE()	(404)	GETFONT()	(437)
DMY()	(405)	GETPAD()	(437)
DOW()	(405)	GOMONTH()	(438)
DTOC()	(406)	HEADER()	(439)
DTOR()	(406)	IIF()	(439)
DTOS()	(407)	INKEY()	(440)
EMPTY()	(407)	INLIST()	(442)
EOF()	(408)	INSMODE()	(442)
ERROR()	(409)	INT()	(442)
EVALUATE()	(410)	ISALPHA()	(443)
EXP()	(410)	ISCOLOR()	(443)
FCHSIZE()	(411)	ISDIGIT()	(444)
ECLOSE()	(411)	ISLOWER()	(444)
FCOUNT()	(413)	ISUPPER()	(445)
FCREATE()	(414)	KEY()	(445)
FEOF()	(415)	LASTKEY()	(446)
FERROR()	(416)	LEFT()	(447)
FFLUSH()	(418)	LEN()	(447)
FGETS()	(419)	LIKE()	(448)
FIELD()	(419)	LINENO()	(449)

LOCFILE()	(449)	POPUP()	(478)
LOCK()	(454)	PRINTSTATUS()	(479)
LOG()	(457)	PRMBAR()	(479)
LOG10()	(458)	PRMPAD()	(480)
LOOKUP()	(458)	PROGRAM()	(481)
LOWER()	(459)	PROMRT()	(482)
LTRIM()	(459)	PROPER()	(483)
LUPDATE()	(459)	PROW()	(483)
MAX()	(460)	PUTFILE()	(483)
MCOL()	(460)	PV()	(485)
MDOWN()	(461)	RAND()	(485)
MDX()	(462)	RAT()	(486)
MDY()	(462)	RATLINE()	(486)
MEMLINES()	(463)	RDLEVEL()	(487)
MEMORY()	(463)	READKEY()	(487)
MENU()	(464)	RECCOUNT()	(488)
MESSAGE()	(464)	RECNO()	(488)
MIN()	(465)	RECSIZE()	(489)
MLINE()	(465)	RELATION()	(489)
MOD()	(465)	REPLICATE()	(490)
MONTH()	(466)	RIGHT()	(490)
MRKBAR()	(466)	RLOCK()	(490)
MRKPAD()	(468)	ROUND()	(490)
MROW()	(470)	ROW()	(491)
MWINDOW()	(471)	RTOD()	(491)
NDX()	(472)	RTRIM()	(491)
NETWORK()	(473)	SCHEME()	(491)
NUMLOCK()	(473)	SCOLS()	(492)
OBJNUM()	(473)	SECONDS()	(492)
OCCURS()	(473)	SEEK()	(492)
OEMTOANSI()	(474)	SET()	(493)
ON()	(474)	SIGN()	(495)
ORDER()	(475)	SIN()	(495)
OS()	(476)	SKPBAR()	(495)
PAD()	(476)	SKPPAD()	(495)
PADC,PADL,PADR	(476)	SOUNDEX()	(496)
PARAMETERS()	(477)	SPACE()	(496)
PAYMENT()	(478)	SQRT()	(496)
PCOL()	(478)	SROWS()	(497)
PI()	(478)	STR()	(497)

STRTRAN()	(497)	WBORDER()	(516)
STUFF()	(497)	WCHILD()	(517)
SUBSTR()	(498)	WCOLS()	(518)
SYSMETRIC()	(498)	WEXIST()	(519)
SYS()	(499)	WFONT()	(520)
TAG()	(509)	WLAST()	(521)
TAN()	(510)	WLCOL()	(521)
TARGET()	(510)	WLROW()	(521)
TIME()	(511)	WMAXIMUM()	(521)
TRANSFORM()	(511)	WMINIMUM()	(522)
TRIM()	(512)	WONTOP()	(522)
TXTWIDTH()	(512)	WOUTPUT()	(523)
TYPE()	(513)	WPARENT()	(524)
UPDATED()	(513)	WREAD()	(524)
UPPER	(514)	WROWS()	(524)
USED()	(514)	WTITLE()	(525)
VAL()	(515)	WVISIBLE()	(525)
VARREAD()	(515)	YEAR()	(526)
VERSION()	(516)		

附录一 Windows 的 FoxPro 2.5 各系统菜单名称	(527)
附录二 FoxPro 2.5 可使用的颜色与颜色代码	(531)
附录三 INKEY()及 LASTKEY()函数的返回值	(532)
附录四 READKEY()函数的返回值	(534)
附录五 “ON KEY=”所用的按键码	(535)
附录六 ASCII 码列表	(536)

第一章 命令篇

#DEFINE... #UNDEF

语法: #DEFINE <constant name> <expr>
#UNDEF <constant name>

如果您熟悉 C 语言,相信对预处理命令(Preprocessor Directives)必定不会陌生。FoxPro 2.5 提供的预处理命令 #DEFINE 与 #UNDEF 能在程序中建立编译常量。利用 #DEFINE 建立常量而不使用内存变量,可少占内存,提高程序执行的效率,简化程序并增加程序的可读性。

要使用 #DEFINE 建立一个常量,只要在<constant name>中指定此常量名称,并在<expr>中指定此常量所代表的值。当程序被编译时,FoxPro 便进行正文替换,且每当<constant name>出现在程序中时,FoxPro 便将它视为是<expr>。一旦想终止常量的替换,只要在 #UNDEF 之后指定此常量的名称(即<constant name>)即可。

替换仅仅发生在 #DEFINE 语句与 #UNDEF 语句之间的那些程序行中,而且此常量仅适用于建立此常量的程序中。

<constant name>

欲使用 #DEFINE 建立一个编译时的常量,请在<constant name>中指定此常量的名称,<constant name>必须是一个合法的 FoxPro 名字。这意味着<constant name>必须以一个英文字母或是下划线开头,而且最长不可超过 10 个字符。为了提高程序的可读性并简化调试过程,建议在对编译的常量命名时,采用一个标准的命名约定。

注意: 不要使用 FoxPro 的关键字(Keyword)作为常量的名称。

<expr>

当使用 #DEFINE 建立一个编译时的常量时,请在<expr>中指定此常量的值。<expr>可以是一个字符、数值、日期或逻辑型的表达式。

程序范例

正如前面所提到的,利用预处理命令 #DEFINE 建立一个编译常量而不使用内存变量,可少占内存、提高程序执行的效率及简化程序。此外,最直接让程序员感受到的好处是增加程序的易读性。举例来说,我们经常在程序中根据 INKEY()或 LASTKEY()函数的返回值来判断用户究竟按下哪一个键,以决定后面的程序走向。然而各个按键的 INKEY()及 LASTKEY()的返回值皆不相同,因此当我们见到类似如下程序代码时,很难了解究竟哪一个数值代表哪一个键:

```
...  
...  
choice = INKEY(0,"HM")  
CLEAR TYPEAHEAD  
DO CASE
```

```

CASE choice = 27
    EXIT
CASE choice = 1
    GO TOP
CASE choice = 6
    GO BOTTOM
...
...
ENDCASE
...
...

```

但是,如果我们能够利用有明确含义的常数来置换每个按键 INKEY() 及 LASTKEY() 的返回值,将大大增加程序的易读性。如下所示:

```

* 程序文件名称: DEMO1. PRG
#DEFINE Home 1
#DEFINE End 6
#DEFINE PgDn 3
#DEFINE PgUp 18
DOW alex FROM 5,01 TO 24,79:
    FONT "Hlv", 12 ;
    STYLE "B";
    TITLE "数据浏览"
mROW = WROWS ("alex") - 1
mCOL = WCOLS ("alex")
ACTIVATE WINDOW alex
@ 01,06 TO 08,55 PEN 2
DO WHILE .NOT. JUDGE
    @ 2,10 SAY "客户编号:" + cust_id
    @ 3,10 SAY "公司名称:" + company
    @ 4,10 SAY "接洽人:" + contact
    @ 5,10 SAY "住址:" + address1
    @ 6,10 SAY "税率:" + STR (taxrate, 4, 2)

    @mROW - 1, 01 SAY "<Home 键 第一项> <End 键 最后一项> ";
                        "<PgUp 键 下一项> <PgDn 键 上一项> ";
                        COLOR RGB (225,0,0,0,0,0)
    @mROW, 26 SAY "<Esc 键 退出>";
                        COLOR RGB (225,0,0,0,0,0)

    CHOICE = INKEY(0,"HM")
    CLEAR TYPEAHEAD
    DO CASE
        CASE CHOICE = Home
            GO TOP
        CASE CHOICE = End
            GO BOTTOM
        CASE CHOICE = PgDn
            SKIP
            IF EOF( )
                WAIT "已到达最后一项数据" WINDOW NOWAIT TIMEOUT 5
                SKIP -1
            ENDIF
        CASE CHOICE = PgUp
            SKIP -1

```

```

        IF BOF( )
            WAIT "已到达第一项数据" WINDOW NOWAIT TIMEOUT 5
            GO TOP
        ENDIF
CASE CHOICE = ESC
    DEACTIVATE WINDOW alex
    RELEASE WINDOWS alex
    # UNDEF    Home
    # UNDEF    End
    # UNDEF    PgDn
    # UNDEF    PgUp
    # UNDEF    Esc
    STORE .T. TO JUDGE
ENDCASE
ENDDO

```

IF... # ENDIF

语法: # IF<expN1>|<expL1>
 <statements>
 [# ELIF <expN2>|<expL2>
 <statements>
 ...
 ...
 # ELIF <expNn>|<expLn>
 <statements>]
 [# ELSE
 <statements>]
 # ENDIF

预处理命令 # IF... # ENDIF 可进行有条件地编译,即有条件地将原始程序代码嵌入一个已编译的程序中。# IF... # ENDIF 能够提高程序的易读性,使已编译的程序文件减小,在某些情况下甚至能提高程序的执行效率。

当 # IF... # ENDIF 结构被编译时,此结构中连续的逻辑或数值表达式便会被计算。而计算结果决定了哪一部分的 ForPro 内容(即程序码)将会被编译并放入编译后的程序文件中。

```

# IF <expN1>|<expL1>
<statements>

```

FoxPro 会计算紧跟在 # IF 之后的数值表达式<expN1> 或逻辑表达式<expL1>。如果此表达式是数值表达式且为非零值,或者此表达式为逻辑表达式且为逻辑真值. T. ,FoxPro 将会编译 #IF 后下一行开始的程序码(<statements>),即将它们嵌入已编译的程序中。接着便跳离 # IF... #ENDIF 结构,而且 #ENDIF 后的程序行将跟着被编译。

反之,如果此表达式是数值表达式且为零值,或者此表达式为逻辑表达式且为逻辑假值. F. , FoxPro 将不会编译 # IF 后的程序代码(<statements>)。但是,所有位于 # ELIF 之后的表达式将会继续被计算。


```
#ELIF <expN2>|<expL2>
    <statements>
```

...

...

```
#ELIF <expNn>|<expLn>
    <statements>
```

如果 #IF 中 <expN1> 为零或 <expL1> 为逻辑假值 .F. , 则 #ELIF 后的表达式将会接着被计算。如果存在的话, 第一个 #ELIF 表达式 <expN2> 或 <expL2> 会被计算。假如 <expN2> 为非零值或 <expL2> 为逻辑真值 .T. , 紧跟在 #ELIF 后的程序代码 (<statements>) 将会被编译而加入到已编译的程序中。接着便跳离 #IF... #ENDIF 结构, #ENDIF 后的程序代码将跟着被编译。

假如 <expN2> 为零或者 <expL2> 为逻辑假值 .F. , 则位于 #ELIF 后的程序代码 (<statements>) 将不会被编译——不会被嵌入已编译的程序代码中。而如果存在下一个 #ELIF, 则此 #ELIF 后的表达式将继续被计算。其余依此类推。

#ELSE

```
<statements>
```

如果并未指定任何 #ELIF 命令, 或者所有的 #ELIF 命令都被计算为零或逻辑假值 .F. , 那么是否存在 #ELSE 命令将决定其他程序代码 (<statements>) 是否会被编译并嵌入已编译的程序中:

- 假如指定了 #ELSE 命令, 则 #ELSE 后的程序码 (<statements>) 将会被编译并被嵌入已编译的程序中。
- 假如并未指定 #ELSE 命令, 则所有位于 #IF 与 #ENDIF 之间的程序代码 (<statements>) 将不会被编译——不会被嵌入已编译的程序中。此时 FoxPro 会跳离 #IF... #ENDIF 结构, 并继续从 #ENDIF 后的第一行程序代码开始编译。

程序范例

```
* 程序文件名称, DEMO2. PRG
#IF 'WINDOWS' $ UPPER(VERSION())
    ? 'This is FoxPro for Windows'
#ELIF 'MAC' $ UPPER(VERSION())
    ? 'This is FoxPro for Mac'
#ELIF 'UNIX' $ UPPER(VERSION())
    ? 'This is FoxPro for UNIX'
#ELSE
    ? 'This is FoxPro for DOS'
#ENDIF
```

RUN|!

用法: RUN[/N[K]]<MS-DOS 命令|外部程序名>
或