第一篇　FoxPro 2.0 程序设计基础篇

本篇将您在 FoxPro 2.0 程序设计中常用到的命令概念，进行了有系统的介绍，如果仔细阅读本篇，您将能很快地掌握 FoxPro 2.0 程序设计的技巧。

数据库软件的演进

Apple（8 位）计算机时代，国外流行 dBASE II 及 PFS (Personal File System)两套软件包，因 dBASE II 的用法很像英文，并且处理的数据较多，所以很快的在商用上广泛流行。

随着硬件进步到 16 位，dBASE 也翻新版本到 dBASE III，此时号称速度较快的 FOXBASE 及 CLIPPER 兼容软件应运而生，FOXBASE 走完全兼容的路线，CLIPPER 号称能将 dBASE III程序转成可执行文件(.EXE)，但全屏幕的命令不接受。

FOXBASE 及 CLIPPER 除了兼容 dBASE III外，又新增许多好用的命令，例如，数组的命令及制作选项菜单的命令..等。随着 dBASE III PLUS 的产生，这两套软件版本也随之翻新，逐渐地反客为主将数据库的市场瓜分殆尽。

开发 dBASE 的公司于是参考 FOXBASE 及 CLIPPER 新增的命令，推出比 dBASE III PLUS 新增 270 多个命令函数的 dBASE IV想力挽狂澜，不料因功能太强、占存储器太大而叫好不叫座，开发 CLIPPER 的公司乘胜追击自立门户，于是翻新版本时根本不管 dBASE IV新增的命令，甚至在 5.01 版时推出一系列与 dBASE IV无关而类似 C 语言的命令。想想 dBASE 忠实用户为了想产生执行文件又必须分辨哪些命令 CLIPPER 接受，哪些不接受岂不命苦。

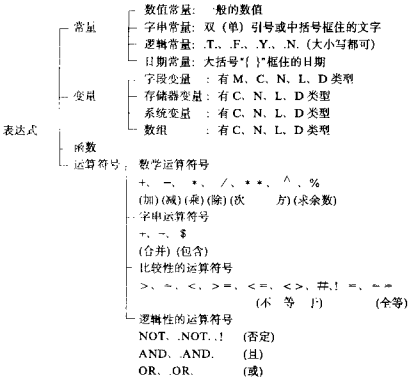
此时开发 FOXBASE 的公司，推出与 dBASE IV完全兼容的 FoxPro 软件，在前年(1991)更推出比 dBASE IV新增 260 多条命令函数的 FoxPro 2.0 版，不但能产生执行文件、速度极快，而且 CLIPPER 5.x 版之前版本的命令，十分之九以上都接受，立刻席卷整个 PC 的数据库市场。

由此历史渊源可知，学习数据库软件可先学 dBASE III PLUS 再升级到 FoxPro 2.0，如此才能缩短学习的时间。当然，您也可直接从 FoxPro 2.0 版开始学起。

本书假设您已具备 dBASE III PLUS 的基础，在此基础上往 FoxPro 2.0 程序设计的领域迈进。

操作数及表达式

使用 FoxPro 2.0 时，要注意类型一致才能运算，而表达式的组成成分如下：



例: 日期常量

```
use friend
list
```

Record #	NAME	ADDRESS	TEL	BIRTHDAY	SEX
1	卓一夫	基隆市东信路 180 巷 22 号	032251222	03 / 06 / 28	男
2	欧阳无敌	台北市忠孝东路 3 段 276 巷 6 弄 25 号	025462376	09 / 10 / 53	男
3	黄蓉	台北市吴兴街 281 巷 11 弄 6 号 3 楼	027081503	12 / 14 / 56	女
4	李小龙	台北市县兴店中下路 234 号	029114444	11 / 10 / 48	男
5	潘金莲	高雄市前镇区平昌里 2 号 113-7 号	078117585	04 / 14 / 53	女
6	马盖先	花莲县吉安乡吉安路盖先巷 20 号 3 楼	038123456	08 / 13 / 55	男

```
1
? birthday = {03 / 06 / 28}
.T.

list for birthday > = {8 / 5 / 1948} and birthday < = {10 / 01 / 54}
```

Record #	NAME	ADDRESS	TEL	BIRTHDAY	SEX
2	欧阳无敌	台北市忠孝东路 3 段 276 巷 6 弄 25 号	025462376	09 / 10 / 53	男
4	李小龙	台北市县兴店中下路 234 号	029114444	11 / 10 / 48	男
5	潘金莲	高雄市前镇区平昌里 2 号 113-7 号	078117585	04 / 14 / 53	女

说明:

一、{ }内以 / 号分隔月 / 日 / 年，年为公元年份。

二、birthday 为日期类型，第一笔记录的birthday字段内容为1928年3月6日，所以为真(.T.)。

三、显示1948年8月5日到1954年10月1日生日的人，之间要用AND(或.AND.)将条件串起来。

例：数值运算

```
? 2 ^ 3          (2 的 3 次方)
    8.00
? 2 * * 3        (2 的 3 次方)
    8.00
? 15 / 6
   2.50
? 15 % 6         (15 除 6 余 3)
    3
```

说明：

一、? 15 % 6 相当于? MOD(15, 6)

二、MOD( )语法为 MOD(被除数，除数)得出的结果为余数。

例： 相不相等

```
? 2 # 6              (不等于成立)
.T.
? 2 < > 3            (不等于成立)
.T.
? 2 ! = 3            (不等于成立)
.T.
SET  EXACT  ON
? "ABC" = "AB"
.F.
? "ABC" = = "AB"  (不全等)
.F.
SET  EXACT  OFF
? "ABC" = "AB"
.T.
? "ABC" = = "AB"  (不全等)
.F.
```

说明：

一、全等(= =)不受 SET EXACT 开关影响。

二、等号(=)受SET EXACT开关影响，当开关为OFF时，等号右方的字串到左方逐一对比，当右方字串结束前相等视为相等。

三、当SET EXACT开关为ON时，? "ABC" = "AB"的结果与? "ABC" = = "AB"相同。

变量

变量有字段变量、存储器变量、数组，处理变量要知道哪些命令可产生变量，而该变

量的类型是什么？

1. 字段变量

在用CREATE命令建立文件结构时，字段变量的类型便已确定，而字段变量的内容是随记录指针而改变。

2. 存储器变量

存储器变量的类型，根据您建立时所用的命令有关。

① ACCEPT，WAIT命令一定产生字串类型的存储器变量。

② SUM，AVERAGE，COUNT命令一定产生数值类型的存储器变量。

③ STORE <表达式> TO <存储器变量>
<存储器变量> = <表达式>

这两道命令所产生存储器变量的类型，随表达式的结果的类型而定。

④ INPUT <"提示字串"> TO <存储器变量>

此命令所产生的存储器变量的类型，随用户输入值的类型而定，一般都用来产生数值类型的存储器变量。

⑤ MEMU TO <变量>
READ MENU TO <变量>
READ MENU BAR TO <变量>，<变量>

这三个命令所产生的变量为数值类型，目的是记录功能菜单出现时，用户的选项是什么。

⑥ SAVE SCREEN TO <变量>

此命令所产生的变量为S类型，一般不用来作运算。

①到④为dBASE III PLUS就有的命令，而⑤与⑥为FOXBASE新增的命令，当然FoxPro 2.0也接受。

3. 系统变量

FOXPRO系统变量名称为底线"_"打头，便于您直接设定值控制，可用DISPLAY MEMORY命令查看其现值。而CLEAR MEMORY及CLEAR ALL命令无法清除，必须用设定新值的方式将其覆盖。

例：

```
_PSPACING = 2        (每行之间空一行)
_PSPACING = 1        (每行之间不空行)
(系统变量的应用请参考报表篇的介绍)
```

4. 数组

数组的产生有以下几种

① 用DIMENSION或DECLARE命令声明

② COPY TO ARRAY命令产生

③ SCATTER TO命令产生

详细内容在后面介绍。

SAVE SCREEN 命令

1. 语法

   SAVE SCREEN [TO <memvar>]

2. 功能

   将目前的屏幕存储到 BUFFER 或存储器变址

3. SAVE SCREEN后，目前的屏幕存储到BUFFER，可用RESTORE SCREEN命令调用存储在 BUFFER 内的屏幕。

   例:

```
SAVE SCREEN          (目前的屏幕存储到 BUFFER)
CLEAR                (目前的屏幕清除)
RESTORE SCREEN       (调出存储在 BUFFER 内的屏幕)
```

4. SAVE SCREEN TO <变量>后，目前的屏幕存储到<变址>，可用RESTORE SCREEN FROM <变址>命令调出存储在<变量>内的屏幕。

   例:

```
SAVE SCREEN TO TEST      (目前的屏幕存储到变量 TEST 内)
CLEAR                    (目前的屏幕清除)
RESTORE SCREEN FROM TEST (调出存储在变量 TEST 内的屏幕)
```

DIMENSION 及 DECLARE 命令声明数组

1. 语法

   DIMENSION 数组名称 (数值 1[,数值 2]) [,数组名称(数值 1[,数值 2])]...
   DECLARE 数组名称 (数值 1[,数值 2]) [,数组名称(数值 1[,数值 2])]...

   例:

```
DIMENSION AA (5)          (一维数组)
DIMENSION AA[5]           (用括号、中括号都可)
DECLARE AA(5)
DECLARE AA[5]
DIMENSION AA(5,4)         (二维数组)
DIMENSION AA[5,4]         (用括号、中括号都可)
DECLARE AA(5,4)
DECLARE AA[5,4]
DIMENSION AA(5,4), A(5), B(6,6)        (一次设定三个数组)
                  ↑
              逗号隔开
```

2. 用途

   DECLARE、DIMENSION 命令用来建立一个或数个数组

3. DISPLAY MEMORY 命令可显示数组内容，CLEAR MEMORY、RELEASE ALL、CLEAR ALL、RELEASE 数组名称 ←— 都能清除数组。

例

```
DIMENSION  AA(5,4), A(5), B(6,6)        (一次设定三个数组)
RELEASE  AA, B        (清除数组 AA 及 B)
```

下标区分数组内元素

数组就是一组使用同一变量名称的所有元素所成的集合，数组内的某一元素用下标区分。

1. 一维数组

例:

DIME ABC(5)

相当产生了 5 个元素（变量）如下（从 1 开始编号）:

```
ABC(1)
ABC(2)
ABC(3)
ABC(4)
ABC(5)
   ↑
  下标
```

2. 二维数组

例:

DIME ABC(3,2)

相当产生了 3 * 2 个元素(变量)如下（行、列都从 1 开始编号）:

```
ABC(1,1)    ←第一个元素
ABC(1,2)    ←第二个元素
ABC(2,1)    ←第三个元素
ABC(2,2)    ←第四个元素
ABC(3,1)    ←第五个元素
ABC(3,2)    ←第六个元素
   ↑
  下标
```

3. FoxPro 2.0标准版（主机286版）最多可拥有的数组个数为3600，而每一数组内最多可拥有 3600 个元素个数。
4. FoxPro 2.0升级版（主机386以上版）最多可拥有的数组个数为65000，而每一数组内最多可拥有 65000 个元素个数。

数组内容

数组用 DECLARE 或 DIMENSION 声明时，内定其内容为逻辑类型的.F.。可用 = 命令设定全部或某一元素的内容。

例:

DIME A(3),B(2,3)　　(设定两个数组)
DISPLAY MEMORY (内定所有元素内容为逻辑类型的.F.)

```
A            Priv   A
        (  1)      L  .F.
        (  2)      L  .F.
        (  3)      L  .F.
B            Priv   A
   (  1,   1)      L  .F.
   (  1,   2)      L  .F.
   (  1,   3)      L  .F.
   (  2,   1)      L  .F.
   (  2,   2)      L  .F.
   (  2,   3)      L  .F.
     2 variables defined,     0 bytes used
   254 variables available

Print System Memory Variables

_ALIGNMENT  Pub    C  "LEFT"
```

A=5　　(设定 A 数组所有元素的内容为 5)
B(2,2)="ABC"　　(设定 B 数组足标 2, 2 元素的内容为 ABC 字串)
DISPLAY MEMORY

```
A            Priv   A
        (  1)      N            5  (          5.00000000)
        (  2)      N            5  (          5.00000000)
        (  3)      N            5  (          5.00000000)
B            Priv   A
   (  1,   1)      L  .F.
   (  1,   2)      L  .F.
   (  1,   3)      L  .F.
   (  2,   1)      L  .F.
   (  2,   2)      C  "ABC"
```

```
(   2,   3)     L  .F.
  2 variables defined,    10 bytes used
 254 variables available

Print System Memory Variables

 ALIGNMENT  Pub    C  "LEFT"
 BOX        Pub    L  .T.
            .
            .
            .
```

RELEASE A (清除数组 A)
DISPLAY MEMORY

```
B           Priv   A
  (   1,   1)     L  .F.
  (   1,   2)     L  .F.
  (   1,   3)     L  .F.
  (   2,   1)     L  .F.
  (   2,   2)     C  "ABC"
  (   2,   3)     L  .F.
    1 variable defined,    10 bytes used
 255 variables available

Print System Memory Variables

_ALIGNMENT  Pub    C  "LEFT"
            .
            .
            .
```

CLEAR MEMORY (清除存储器)

同名数组重复设定

DELCLARE、DIMENSION 设定的数组名称相同，维数以最后设定的有效。而先前同名数组内的元素已设定的内容除非元素减少，否则内容尚"健在"。数组的元素个数为行数乘列数。

例:

DIMENSION AA(6) 有 6 个元素
DIMENSION BB(6,3) 有 6 * 3 共 18 个元素

例:

```
CLEAR  MEMORY
DIME  AA(5)              (设定 AA 为一维数组)
AA=7                     (AA 数组的所有元素内容都为 7)
AA(3)="FOXPRO 2.0  GOOD"
DIME  AA(3,2)            (数组同名,以后设定的维数为主)
AA(3,2)=DATE( )
DISPLAY  MEMORY
```

```
A           Priv   A
(   1,   1)     N            7  (          7.00000000)
(   1,   2)     N            7  (          7.00000000)
(   2,   1)     C  "FOXPRO2.0  GOOD"
(   2,   2)     N            7  (          7.00000000)
(   3,   1)     N            7  (          7.00000000)
(   3,   2)     D  03/06/92
  1 variable defined,    22 bytes used
255 variables available
          :
```

```
DIME  AA(3,1)            (元素减少,其以后的内容不见)
DISPLAY  MEMORY  (数组 AA 变为 3 行 1 列)
```

```
AA          Priv   A
(   1,   1)     N            7  (          7.00000000)
(   2,   1)     N            7  (          7.00000000)
(   3,   1)     C  "FOXPRO2.0  GOOD"
  1 variable defined,    22 bytes used
255 variables available
```

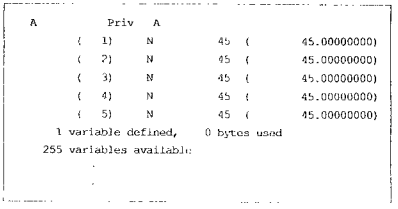
数组与存储器变量同名

数组名称与已有的存储器变量同名，已有的同名存储器变量被其覆盖且内容也不见，若已声明数组后，才用 AVERAGE、SUM、COUNT、STORE 命令产生同名的存储器变量，则是将运算的结果放入同名数组内的所有元素，并不另外产生同名的存储器变量。

例:

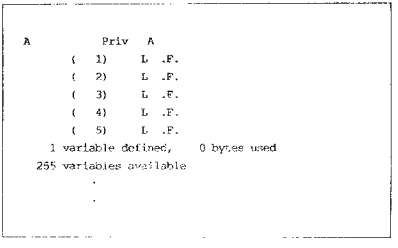
```
CLEAR  MEMORY
A=DATE( )                (先设定了存储器变量 A)
```

```
DIME A(5)                (后设定同名的数组)
USE FRIEND
COUNT TO A               (统计出的笔数"本例是 45" 放入数组 A)
DISPLAY MEMORY
```



例:

```
CLEAR MEMORY
USE FRIEND
COUNT TO A               (统计出的笔数"本例是 45"放入存储器变量 A)
DIME A (5)               (后设定同名的数组 A，存储器变量 A 不见)
DISPLAY MEMORY (数组 A 内的所有元素内容为.F.)
```



例:

```
CLEAR MEMORY
DIME AA(3,2)
USE FRIEND
COUNT TO AA(3,1)         (统计出的笔数放入数组内编号 3,1 的元素内)
DISP MEMORY LIKE AA*     (只显示 aa 打头的变量)
```

```
AA          Priv   A
  (   1,    1)     L  .F.
  (   1,    2)     L  .F.
  (   2,    1)     L  .F.
  (   2,    2)     L  .F.
  (   3,    1)     N          45  (        45.00000000)
  (   3,    2)     L  .F.
```

ALEN( ) 函数得知数组的数据

1. 语法

   ALEN(<数组名称>[,<expN>])

2. 用途

   可获知数组的行数、列数及元素总个数。

3. [,<expN>]

   当[,<expN>]不设定或值为 0，得知数组总个数。

   当[,<expN>]值为 1，得知数组的行数。

   当[,<expN>]值为 2，得知数组的列数。

例:

```
CLEAR  MEMORY
DIME  AA(5,4)
?ALEN(AA,1)          (数组的行数)
  5
?ALEN(AA,2)          (数组的列数)
  4
?ALEN(AA,0)          (数组的个数)
  20
?ALEN(AA)            (数组的个数)
  20
```

COPY TO ARRAY 命令

1. 语法

   COPY TO ARRAY <数组名称> [FIELDS< field list >]

   [<scope>]

   [FOR <expL>]

   [WHILE <expL>]

   [NOOPTIMIZE]

2. 用途

可将数据库的数据复制到数组。

3. 数据库(.DBF)本身便有如数组，行为记录(Record)，列为字段(Field)

4. 数组未声明

① COPY TO ARRAY＜数组名称＞，存储器内没有同名的数组时，便产生指定的数组。

② 数组有几行由[＜scope＞]、[FOR ＜expL＞]、[WHILE ＜expL＞]子句决定复制多少记录。

③ 数组有几列由[FIELDS＜field list＞]子句决定复制多少字段。

④ [＜scope＞]、[FOR ＜expL＞]、[WHILE ＜expL＞]、[FIELDS ＜field list＞]都未指定时，复制整个数据库。

⑤ 数组各元素的内容即对应数据库内行、列的内容。(不包括M型类型字段)

例:

```
CLEAR  MEMORY
USE  TEST
LIST  STRU          (有 2 笔记录，5 个字段，MM 为 M 型类型)
```

```
Structure for database: D:\TEST.DBF
Number of data records:      2
Date of last update   : 03/07/92
Memo file block size  :     64
Field  Field Name  Type       Width    Dec    Index
    1  CC          Character     10
    2  NN          Numeric       10
    3  FF          Float         10
    4  DD          Date           8
    5  MM          Memo          10
** Total **                      49
```

```
LIST
```

```
Record#  CC                NN          FF DD       MM
      1  aaa               11          99 05/07/87 Memo
      2  bb                 2       22222 05/07/88 Memo
```

```
COPY  TO  ARRAY  A1     (A1 数组有 2 行 5 列)
COPY  TO  ARRAY  A2  FIELD  NN, CC
   (指定第 2 笔记录的两个字段，A2 数组有 2 行 2 列)
DISP  MEMORY  LIKE A *
```

```
A1          Priv   A
  (   1,    1)     C  "aaa        "
  (   1,    2)     N          11  (         11.00000000)
```

```
(   1,   3)     N            99  (         99.00000000)
(   1,   4)     D  05/07/87
(   1,   5)     L  .F.
(   2,   1)     C  "bb         "
(   2,   2)     N             2  (          2.00000000)
(   2,   3)     N         22222  (      22222.00000000)
(   2,   4)     D  05/07/88
(   2,   5)     L  .F.
A2          Priv   A
(   1,   1)     N             2  (          2.00000000)
(   1,   2)     C  "bb         "
```

M 类型的内容其数组对应的元素 A1(1,5)及 A1(2,5)为.F.。

5. 数组已声明

① COPY TO ARRAY <数组名称>，存储器内已有同名的数组时，此数组的行数及列数为接受数据库数据的极限。

② 存储器极限由数组的行数决定，字段个数的极限由数组的列数决定。

例:

```
CLEAR MEMORY
DIME A1(1,2)
DIME A2(1,3)
USE TEST
LIST STRU
LIST
```

```
Record#  CC              NN        FF DD       MM
      1  aaa             11        99 05/07/87 Memo
      2  bb               2     22222 05/07/88 Memo
```

```
COPY TO ARRAY A1          (数组 A1 最多接受 1 行 2 列的数据)
COPY TO ARRAY A2 RECORD 2 FIELD NN, CC      (A(1,3) 仍为.F.)
DISP MEMORY LIKE A*
```

```
A1          Priv   A
(   1,   1)     C  "aaa        "
(   1,   2)     N            11  (         11.00000000)
A2          Priv   A
(   1,   1)     N             2  (          2.00000000)
(   1,   2)     C  "bb         "
(   1,   3)     L  .F.
```

6. 可用APPEND FROM ARRAY <数组名称>命令将数组的内容，添加到数据库的尾端。

7. FoxPro 2.0的Rushmore技术，能加速（最优化）[FOR]子句，若不想用Rushmore

技术最优化 COPY TO ARRAY 命令，只需加上[NOOPTIMIZE]子句即可。

SCATTER 命令

1. 语法

SCATTER [MEMO][FIELDS <field list>][子句] <数组名称> | TO <数组名称> BLANK | MEMVAR | MEMO[ALL]

2. 用途

可将数据库目前记录指针的记录复制到数组

3. 数组未声明

① SCATTER TO <数组名称>，存储器内没有同名的数组时，便产生指定的数组。

② 数组为一维，数组元素的总个数由[FIELDS <field list>]子句决定，未指定[FIELDS <fields list>]子句时，为字段总个数。

③ 数组名元素的内容即目前记录指针对应字段的内容。（内定不包括M类型的字段，除非指定[MEMO]子句）

例:

```
CLEAR MEMORY
USE TEST
LIST
```

Record#	CC	NN	FF DD	MM
1	aaa	11	99 05/07/87	Memo
2	bb	2	22222 05/07/88	Memo

```
1       (记录指针到第一笔)
SCATTER TO A1
SCATTER FIELD NN, CC TO A2      (指定两个字段)
2       (记录指针到第二笔)
SCATTER TO A3 MEMO              (指定要 M 类型)
DISP MEMORY LIKE A*
```

```
A1          Priv   A
        (  1)      C  "aaa        "
        (  2)      N             11  (          11.00000000)
        (  3)      N             99  (          99.00000000)
        (  4)      D  05/07/87
A2          Priv   A
        (  1)      N             11  (          11.00000000)
        (  2)      C  "aaa        "
A3          Priv   A
        (  1)      C  "bb         "
        (  2)      N              2  (           2.00000000)
```

```
(  3)   N        22222  (     22222.00000000)
(  4)   D  05/07/88
(  5)   C  "RECORE IS 2"
```

A3(5)的内容为记录指针第二笔 MM (M 类型)字段的内容。

4. 数组已声明

SCATTER TO <数组名称>，存储器内已有同名的数组时

① 若其数组元素声明的比要复制出的字段少，自动将已声明的数组加大。

② 若其数组元素声明的比要复制出的字段多，数组尺寸不变。

例:

```
CLEAR MEMORY
DIME A1(2)        (只有两个元素)
DIME A2(4)
USE TEST
LIST
```

```
Record#  CC          NN          FF DD       MM
      1  aaa         11          99 05/07/87 Memo
      2  bb           2       22222 05/07/88 Memo
```

```
1
SCATTER TO A1
   (有 5 个字段，M 类型内定不复制，A1 扩成 A1(4)
SCATTER FIELD NN, CC TO A2
   (A2(4)比复制出的字段个数多)
2
SCATTER TO A3 MEMO
DISP MEMORY LIKE A*
```

```
A1          Priv    A
        (  1)   C  "aaa"
        (  2)   N                  11.00000000)
        (  3)   N                  99.00000000)
        (  4)   D  05/07/
A2          Priv    A
        (  1)   N                  11.00000000)
        (  2)   C  "aaa"
        (  3)   L  .F.
        (  4)   L  .F.
A3          Priv    A
        (  1)   C  "bb"
        (  2)   N           2  (       2.00000000)
```

```
(  3)     N        22222  (     22222.00000000)
(  4)     D  05/07/88
(  5)     C  "RECORE IS 2"
```

A1(2)扩大为 A1(4)

5. 可用GATHER FROM <数组名称> | MEMVAR [FIELDS <field list>] [MEMO]命令，将数组内容复制到目前记录指针所在的对应字段。

变量的存储

1. 若要将目前的变量（不包括系统变量），保留到一个文件（变量文件）而存储在磁盘，可用 SAVE TO <变量文件>命令。下达完该命令，磁盘内便多了一个扩展名为.MEM 的变量文件。
2. 可用CLEAR MEMORY来清除所有的变量。也可用RELEASE命令清除部分或指定的变量（不包括系统变量）。
3. 清除所有的变量前，若已用SAVE命令存储，便可用RESTORE FROM <变量文件>命令将当时存储的变量再调回．而使用 RESTORE 命令时，在先做 CLEAR MEMORY 的动作后，才将当时存储的变量装入．若希望装入时不做 CLEAR MEMORY 的动作（即保留目前的变量），可在 RESTORE 命令后多加 [ADDITIVE]这个子句。

例:

```
CLEAR MEMORY
DECLARE A(2)
B=1
SAVE SCREEN TO C
SAVE TO MMM
CLEAR MEMO
D=DATE( )
RESTORE FROM MMM
DISPLAY MEMORY
```

```
A            Priv   A
       (  1)     L  .F.
       (  2)     L  .F.
B            Priv   N            1  (          1.00000000)
C            Priv   S  80 by 25
     3 variables defined,    0 bytes used
   253 variables available

Print System Memory Variables
```

```
_ALIGNMENT  Pub    C  "LEFT"
_BOX        Pub    L  .T.
```

目前存储器只有A、B及C两个变量，因RESTORRE命令先将目前存储器变量D清除后，才将保留在 MMM.MEM 的 A、B 及 C 变量装入。

例:

```
CLEAR MEMORY
DECLARE A(2)
B=1
SAVE SCREEN TO C
SAVE TO YYY
CLEAR MEMO
D=DATE( )
RESTORE FROM YYY ADDITIVE
DISPLAY MEMORY
```

```
D           Priv   D  03/07/92
A           Priv   A
       (  1)       L  .F.
       (  2)       L  .F
B           Priv   N          1  (          1.00000000)
C           Priv   S  80 by 25
    4 variables defined,     0 bytes used
  252 variables available

Print System Memory Variables

_ALIGNMENT  Pub    C  "LEFT"
_BOX        Pub    L  .T.
```

目前存储器有A、B、C及D四个变量。因多加ADDITIVE子句，则D变量不会被清除。