MUMPS程序设计语言

郭荣江　孙传林　王海鸥　　编写

人民卫生出版社

MUMPS程序设计语言

郭荣江　孙传林　王海鸥　编写

人民卫生出版社出版
（北京市崇文区天坛西里10号）
长春科技印刷厂印刷
新华书店北京发行所发行

850×1168毫米32开本　7 $\frac{3}{4}$印张　211千字
1988年3月第1版　1988年3月第1版第1次印刷
印数：00,001—1,500
ISBN 7-117-00353-7/R·354　定价：2.20元

〔科技新书目152—63〕

目　录

第一篇 MUMPS语言

第一章 MUMPS语言的特征

第一节 MUMPS的产生和发展

MUMPS是1967年由美国麻省总医院(Massachusetts General Hospital, MGH) 计算机科学研究室开发的面向医疗信息处理的系统。MUMPS表示MGH Utility Multi-Programming System。当我们使用MUMPS这个名字时，有时指语言本身，有时指包括操作系统、语言和数据库管理系统在内的整个系统。这个系统的设计目标是：编写程序容易，建立数据库简单、高可靠性的分时系统（TSS）。MUMPS实现了这个目标，特别适用于多用户的医疗信息的收集和利用。

一个医务工作者为解决自己工作中的问题，需要具有自己编写程序的能力。MUMPS的文法简单，易于学习。它的文件处理功能很强，数据库的操作简便。MUMPS是一个在中、小型机上使用的，具有高的性能价格比，容易编程及实现的新一代的计算机语言。

在MUMPS产生后，很多厂家提供了能支持MUMPS运行的系统。因之，派生出许多MUMPS方言。为了使这种新的语言能统一起来，以便软件的积累与交换，1973年成立了MUMPS开发委员会（MDC），在标准化方面做了许多工作。1977年，美国国家标准局（ANSI）在继FORTRAN、COBOL之后，批准MUMPS为第三个标准计算机语言。与其他计算机语言相比，例如，作为面向科学计算的FORTRAN语言及面向事务处理的COBOL语言，后者在使用穿孔卡片或磁带作为输入的批处理方面有很大优势。而MUMPS是以磁盘为基础，因此在程序运行过程中实时的人-机对

话的方面占有优势。标准MUMPS在用于科学计算的各种函数方面稍微逊色，但以磁盘作为外部记忆装置的数据库功能方面却很强。因此，除了医学领域之外，历来为COBOL所占有的银行事务处理等领域，已成为MUMPS开始渗透的目标。

第二节　MUMPS简介

为什么医院需要产生MUMPS这种类型的计算机语言呢？这是因为医院病员的流动性很大，必须有一个编写程序简单、容易修改数据的计算机语言。编制MUMPS程序简单到怎样的程度呢？用下面一个例子加以说明。在该程序中，用了如下三个命令：READ命令是向计算机输入数据；WRITE命令是由计算机输出数据；SET命令是数据的赋值、合成或置换。也就是说，用这三个命令来对数据进行处理。

这个程序如下：

```
001  READ  “ 长度=?”*, L
002  READ  “ 宽度=？”, B
003  SET   A=L*B
004  WRITE “ 面积=”, A
```

即便对于不懂得MUMPS语言的人，这个程序也是一目了然的。它是用来计算面积的。计算机执行这个程序时，在终端的屏幕上进行如下形式的对话：

长度=？12　宽度=？25　面积=300

每行语句前头的号码001，002，003，004，称为行标号，是为了标识该行使用。

为了让系统记住一个字母，字符串或数值，设置了一些“变量”分别记忆。例如，在上例中，用L这个变量代表长度，B变量代表宽度，S变量代表面积。而各个命令，只需用它的英文的头一个字母即可。例如，READ命令，可简写为R。于是程序可简化为

*引号内的提示使用中文，是为了便于理解。在配置了中文的MUMPS系统中，这一点是可以实现的。

```
001  R  "  长度=？ ",L
002  R  "  宽度=?",B
003  S  A=L*B
004  W  "面积=", A
```

使用简写，可以减轻程序书写过程中的劳动。一般情况，一行中可以包括数个命令语句，因此，可以把一个小的处理单位放在同一行中，这样，程序就显得十分紧凑。

MUMPS系统可以对一个个命令直接执行。也就是说，每向系统输入一行命令，它就立刻执行，并给出结果。这称为直接模式（direct mode）。因为在直接模式下，程序的局部已被执行并给出结果，因此，容易发现程序的错误，可随时改正。

MUMPS是一种解释性的语言。也就是说，机器每执行一个命令时，首先对该命令进行解释，翻译成机器语言，然后立刻执行。如果系统运行整个程序，而且运行中发现某个命令的书写错误，则在出错处停机。这时，用户可以对错误进行修改。修改程序的方法依机器的不同厂家而异，一般使用编辑程序。

使用变量带来很大的灵活性。MUMPS 在引用新的变量，特别是引用带下标的数组变量时，不必经过事先的定义（这是本语言的特征之一）。这样，在书写新的程序或修改已作成的变量时，可以自由地引用变量。

MUMPS的运算速度，比起用编译程序先编译成机器语言的FORTRAN、COBOL等，是较慢的。因为MUMPS的解释是在执行中进行的。因此，对一个同样内容却反复运行的程序，MUMPS是不利的。

第三节　MUMPS数据库简介

MUMPS的数据，有两种存储的形式：局部变量（Local variable）在内存中存放，全局变量（globle variable）在磁盘上存放。局部变量在程序的执行过程中是有效的。程序执行完毕并退出内存时，局部变量随之消失。而全局变量在程序执行完毕并退出内存时仍然存在。存放全局变量，或者说对全局变量的数据

进行收集和检索的“基地”，就是数据库。

MUMPS的数据库可为不同的用户所共享。因此，如果某一个程序向数据库输入新的数据，或对数据库进行更新，之后，另一个程序可以向数据库检索这些数据。数据的收集或更新，一般是通过终端进行，也可以从其他计算机或其他数据收集设备（如医院生化自动分析仪）直接输入。MUMPS的数据库是MUMPS的最大特色。

假定从医院的生化自动分析仪送来了如下一批数据，这批数据用一个^LAB的名字称呼：

```
〔^LAB〕
    453
    —    1000201    1.68^817^17.88
    —    1000301    1.66^832^18.73
    —    1000302    1.98^822^12.49
    —    1000401    1.54^846^21.09
    —    1000402    1.05^824^21.31
    —    1000404    1.44^807^12.12
    —    1001201    1.99^837^15.05
    —    1001202    1.51^842^18.09
    —    1001203    1.77^801^13.02
```

在上面的数据中，453是某一化验项目的代码，而数据的左列1000201，1000301等，是病人的病历号，右列1.68^817^17.88等，是此项化验所包括的三个数据，每个数据间用“^”隔开。现在，如果把这组数据命名为K，则可写一个如下的建库命令：

```
SET^LAB（CODE，ID）=K
```

其中CODE是化验项目的代码，ID是病人的病历号，它们都作为下标使用。^LAB就是一个全局变量。只须不断地使用这个命令，填入CODE、ID及K的各个实际值，就可建立数据库。

MUMPS的数据库是使用树型结构的，如图1—1。

现在如果要从数据库中读出某个数据，只需给出CODE及ID的具体数值，再用如下命令

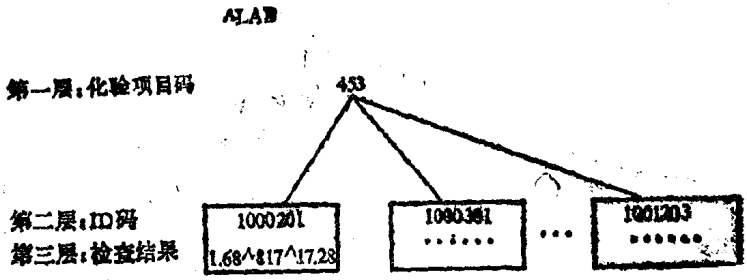

图1—1 MUMPS的数据库结构

SET DATA=^LAB（CODE，ID）

即可。

CODE与ID不必按顺序建立，可以随机的形式给出。

可见，MUMPS数据库的建立、增加、更新、删除和读出，都是非常方便的。

与数据库的功能有关的一个函数是$NEXT。$NEXT的意义是，给出紧跟着指定的某一个全局变量的最低一层下标的下一个下标值。例如，对上列的^LAB来说，$NEXT（^LAB(453, 1000201)）=1000301。下面，是一个使用$NEXT函数进行数据库检索的例子。

```
MEAN, GET MEAN VALUE OF LAB  RESULT,  FOR  LAB
        CODE 453
        ;
1     S  ID=-1, N=0, SUM=0
2     S  ID=$N(^LAB(453, ID))  I  ID<0  G  OUT
      S  DATA=^LAB(453, ID)
      S  DATA1=$P(DATA,"^", 1)
      S  SUM=SUM+DATA1, N=N+1
      G  2
        ;
OUT   S  MEAN=SUM/N
      W  !!,"SAMPLE MUMBER:",N
      W  !!,"MEAN VAULE:",MEAN
      W  !!!!
      Q
```

这个程序的执行及其结果是：

```
>D^MEAN
SAMPLE  NUMBER：10
MEAN  VALUE    ：1.627
```

这个被命名为MEAN的程序，是对^LAB的所有数据中由"^"符号加以分隔的3种检查结果的第一种数据的全部求平均值。在程序中，$NEXT是用来对第一个下标是453的全部数据的第二个下标（ID）进行追踪。

逐条分析这个程序。

标号为1的行：

为了$NEXT的追踪，给下标ID赋初值－1。并给检查个数N及检查结果值的总和SUM赋初值0。

标号为2的行：

在第1个下标固定为453的情况下，用$NEXT（简写为$N）对第二个下标进行追踪。如果第二个下标在全部范围内已追踪完毕，则转移到OUT行。

标号为2的行＋1行：

取出一个全局变量^LAB（453，ID），然后将其赋给局部变量DATA。

标号为2的行＋2行：

使用$PIECE函数（简写为$P，详见本篇第二章）将DATA中用"^"分隔的第一个数据取出，并放入DATA1中。

行标号为2的行＋3行：

把DATA1的值加到SUM中，并把N加1。

行标号为2的行＋4行：

转移到第2行，以便追踪下标ID的下一个值。

行标号为OUT的行：

计算平均值。

以下4行：

打印处理结果并停机。

在行2中，$NEXT的追踪过程如下：

```
$N(^LAB(453, -1))      → 1000201
$N( ^LAB(453, 1000201))→ 1000301
$N( ^LAB(453, 1000301))→ 1000302
$N( ^LAB(453, 1000302))→ 1000401
$N( ^LAB(453, 1000401))→ 1000402
$N( ^LAB(453, 1000402))→ 1000403
$N( ^LAB(453, 1000403))→ 1000404
$N( ^LAB(453, 1000404))→ 1001201
$N( ^LAB(453, 1001201))→ 1001202
$N( ^LAB(453, 1001202))→ 1001203
$N( ^LAB(453, 1001203))→ -1
```

用这个函数，可以追踪数据库中一部份或全部数据。在这个例子中，三个数据用“^”分隔后，放在一个字符串中。MUMPS允许的串长度是255。

MUMPS的数据库，对数据个数不能确定，数据形式不能确定，数据发生的顺序不能确定，或是字母或是数字不能确定的自由行文等形式的数据的处理是十分有力的。

此外，下标不必顺序选取，而允许自由选择这一点，是MUMPS的又一特征。下标不但可以是数值，也可以是字母的组合。例如，下面的全局变量是存在的：

^FEEL（“GOOD”）=“5 PATIENTS”

这在数据用编码分类有困难或根本不可能用编码分类的情况下是很有利的。例如，在构成一个以关键字为分类码的文献数据库时。

当采用字母组合作下标时，$NEXT的跟踪是按字母顺序来进行的。

第二章　标准MUMPS文法

第一节　数据类型与值

标准MUMPS操作的数据，是由128个ASCII字符形成的可变长字符串表示。数据的最大长度为255个字符。共分三种类型的

数据：

1. 数值

由特殊的字符串组成，这些字符是0，1到9共10个数字字符，正负符号（+、-）及表示以10为底的指数的大写英文字母E。例如

通常表示法	指数表示法
125	12.5E1
1389.468	13.89468E2
+1000	+10.E2
-172.4	-17.24E1
0.0156	15.6E-3
0.00000762	76.2E-8

下面，为数值数据的几种错误写法：

正确写法	错误写法
35	thirty-five
29000	29,000
10E5	10×10(5)
0.34	34/100
13.0E4	13.0 E4（插入一个空格）
10	E+1（E前无数值）

标准MUMPS的数的最大长度为9位。如123456789或1.23456789为有效。而99.87654321则被机器理解为99.87654320，即最后一位被4舍5入。ANSI标准MUMPS的数值精度在10^{-25}与10^{+25}之间。

2. 字符串

由ASCII码中除控制字符外的可印刷字符系列组成。字符串用双引号括起，字符串中双引号本身用一对双引号表示。例如：

定义	实际字符串
" "	空字符串

本章根据ANSI X11.1-1977"MUMPS Language Standerd"写成

“xyz”	xyz
“John Doe”	John Doe
“a”“super”“party”	a“super”party

字符串的长度不能超过255个字符。

字符串作为数值来处理时的取值规则将在第三节提到。

3. 逻辑值

只有1，0两个字符。逻辑演算结果为真时表为1，逻辑演算结果为伪时表为0。

第二节 变 量

变量是指其值可以变化的数据。MUMPS 中有三种类型的变量。第一种称为局部变量，是由特定用户所定义和管理的。它只存放在内存中，不在磁盘或数据库中。第二种称为全局变量，为所有用户定义和管理，存放在磁盘或数据库中。第三种称为特殊变量（special variable），它反映 MUMPS 的系统状态，只能由MUMPS系统加以改变，用户只能使用它，不能更改它。

变量由它的变量名加以识别。局部变量由%或大写英文字母起首，续以大写英文字母或数字，有效长度为8个字符。例如以下变量名是正确的：

A，X，%，MUG，X12，%MUMPS，A5B2，%SYSI。

以下的变量名是错误的：

12A，A%，$XGZ，GXabc，%%ABC。

全局变量的变量名称除了在局部变量名前加^（或↑）外，其余规定与局部变量名相同。

全局变量名举例如下：

^A，^X，^%，^MUG，^X12，^%MUMPS，^A5B2，^%SYSI。

以下形式的全局变量名是不允许的：

^12A，^A%，^$XGZ，^GXabc，^%%ABC，G^H，Z^。

特殊变量的第一个字符一定是$，每个特殊变量有它的书写方法（参看本章第七节）。

特殊变量的例子：

$TEST，$X，$Y，$HOROLOG。

一连串在某种意义上相关的局部变量与全局变量的集合，形成数组。给数组的每一个元素加以一个下标，例如，可以给GAME，SET,MATCH代以A（1），A（2），A（3），这样便利得多。在必要的情况下,下标可以是多个的,这时数组就成为多维的。在这种情况下，例如，A（1）与A（1，4，13）都是A的下位节点，而A（1，4，13）又是A（1）的下位节点。反之，A与A（1）是A（1，4，13）的上位节点，A是A（1）的上位节点。所以MUMPS的数据结构是树型结构。如图1—2示。

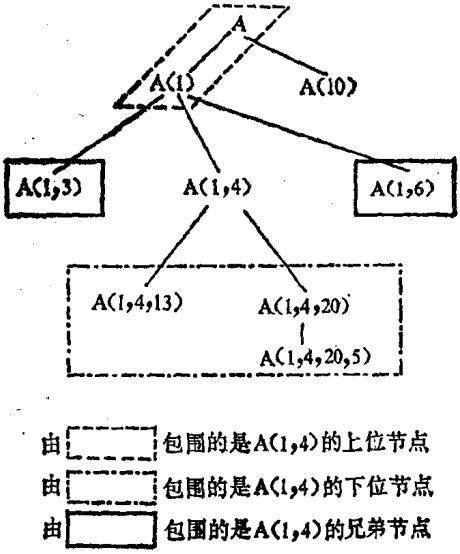

图1—2 MUMPS的树型数据结构

数组下标的取值范围为0到999999999的任意非负整数。

全局变量可以用一种省略的形式来表示。省略形式的全局变量的第一个下标与最后执行的语句中的该全局变量的最后一个下标是同一层的。

例如，有一个赋值语句为：

SET ^MUG（1，5）=3,^MUG（1,6,100）=4，

^MUG(1，6，200) =5

则可用简式表示为：

SET ^MUG (1, 5) =3, ^(6, 100) =4, ^)200) = 5

其中

^MUG (1, 5) →^MUG(1, 5)

^(6, 100) →^MUG (1, 6, 100)

^(200) →^MUG (1, 6, 200)

第三节 算 子

标准MUMPS使用如下几种算子：

- 算子
  - 一项算子
    - 算术算子：+，−
    - 逻辑算子：，
  - 二项算子
    - 算术算子
      - 算术算子：+，−，*，／，＼，#
      - 算术关系算子：＞，＜
    - 字符串算子
      - 字符串关系算子：=，⊐，⊂
      - 模式匹配算子：？
      - 连接算子：—
    - 逻辑算子：&，!
  - 间接命令算子：@

在解释上列各种算子的意义之前，必须对字符串作为数来处理时的取值规定加以说明。如果在某一个字符串左端有一个用指数表示的数（如86E5）、小数（如－182.45）或整数（如1984），则这个字符串作为数来处理时，就取这个数的值；这个字符串作为整数来处理时，就取左端的这个数取整后的值；这个字符串作为真伪值来处理时，若该数为0，则真伪值取0，否则取1。如字符串的左端冠以空格，则取值为0。

例如：

字符串	取值	取整	真伪值
“810”	810	810	1
“98KLOGRAM”	98	98	1

“ ” （空字符串）	0	0	0
“ 35”	0	0	0
“86 + 9”	86	86	1
“PAGE 10”	0	0	0
“ - 8.4”	- 8.4	- 8	1
“86E - 1”	8.6	8	1
“ - - - 9”	- 9	- 9	1
“ - 0”	0	0	0
“0.010”	0.01	0	1
“1.5.7”	1.5	1	1
“1E + 2E + 5”	100	100	1

（最后一个字符串取值为1E + 2，即100）

MUMPS对各种算子的演算对象的解释及演算结果列表如下：

算子		演算对象解释	结果
+	一项算子	数值解释	数值
-	一项算子	数值解释	数值
,	一项算子	真伪值解释	真伪值
+		数值解释	数值
-		数值解释	数值
*		数值解释	数值
/		数值解释	数值
\		数值解释	数值
#		数值解释	数值
>		数值解释	真伪值
<		数值解释	真伪值
=		字符串解释	真伪值
[		字符串解释	真伪值
]		字符串解释	真伪值
_		字符串解释	字符串
?		左边，字符串解释 右边，模式解释	真伪值
&		真伪值解释	真伪值

!	真伪值解释	真伪值

以下，用列表方法，并用具体的例子对各个算子进行说明。间接命令在形式上是一个算子，由于使用方法特殊，在第八节中详细说明。

一、算术一项算子

算子	意义	例	结果与说明
+	进行数值解释	+2	2
		+"34A"	34
-	进行数值解释并变号	-A	如果A的值为7则结果为-7
		-"34A"	-34

二、算术二项算子

算子	意义	例	结果与说明
+	和	2+7	9
		A+3	在A的值上加上3
		"3DOGS"+"5CATS"	8
-	差	2-7	-5
		A-B	从A的值中减去B的值
*	积	2*7	14
/	商	7/4	1.75
\	整除	11\3	3
		7\4	1
		1\3	0
		4\-3	-1
#	取模*	11#3	2
		11#-3	-1
		-11#3	1
		-11#-3	-2

* 取模的计算式如下：N#D=N-(D*floor(N/D))，式中floor(x)为不大于X的最大整数。