



# M

## VA M Programming Intermediate

### Lesson 8

## Completing Your M System

Veterans Health Administration  
Office of Information  
OI National Training and Education Office

# VA M PROGRAMMING

## Intermediate

---

### Table of Contents

Guide to Symbols Used	1
Optional Reading References	1
References	1
Document Update Note	1
Objective	2
Enhancements to Our Mailing List System	2
M Code Sample	2
The Slightly Modified Menu Routine	3
M Code Sample	4
The New Enter or Edit Entries Routine	5
New Code for the Enter Routine	5
M Code Sample	5
M Code Sample	6
M Code Sample	7
M Code Sample	7
M Code Sample	7
The Enter/Edit Routine: VECS8ENT	8
M Code Sample	8
The Edit Fields Routine	10
M Code Sample	10
The NEW Command	13
M Code Sample	14
M Code Sample	14
Fancy Searching Technique	15
M Code Sample	15
M Code Sample	17
Validating State Codes	20
M Code Sample	20
M Code Sample	20
M Code Sample	21
Validating Dates	21
M Code Sample	21
M Code Sample	22
M Code Sample	23
M System Architecture	24
Overview of M	26

# VA M PROGRAMMING

## Intermediate

---

Who uses M ?	26
On What Systems Does M Run?	26
Operating Systems	26
Computers	27
How Does M Compare with Other Languages?	27
What Are the Strengths of M?	28
What Is the Future of M?	30
Assignment (optional)	31
In General	31
Mandatory Specifications	31
Important	32

# VA M PROGRAMMING

## Intermediate

---

### *Acknowledgements*

Developing a comprehensive training package for the **M** computer programming language as it is used in VA was not an easy task. Much gratitude goes to the members of the M Programming Language Training Development Team who spent countless hours developing this training program. Sincere appreciation to:

Debbie Channell  
Tuscaloosa OI Field Office, Customer Services

Harris H. Lloyd  
Central Arkansas Veterans Health Care System

Bob Lushene  
Hines OI Field Office, Technical Services

# VA M PROGRAMMING

## Intermediate

---

### Guide to Symbols Used

<ENTER> Press the Enter or Return key

<TAB> Press the Tab key

<space> One space



What follows is from part of the VA Programming Standards and Conventions.



This indicates an OPTIONAL reading reference. Additional information on this topic may be found in the references listed below.

### Optional Reading References

Lewkowicz, John. *The Complete M: Introduction and Reference Manual for the M Programming Language*. 1989.

Walters, Richard. *M Programming – A Comprehensive Guide*. 1997.

The optional reading references are available through:

M Technology Association  
1738 Elton Road, Suite 205  
Silver Spring, MD 20903  
301-431-4070

### References

The VA M Programming Standards and Conventions shown were taken from “VA DHCP Programming Standards and Conventions” document, prepared March 19, 1991, modified in 1995 and approved January 22, 1996.

<http://vaww.vista.med.va.gov/Policies/sacc.htm>.

### Document Update Note

**January 2007** – To meet current standards on personal data privacy and accessibility, this document was updated. It was edited to include document metadata and edited to de-identify any personal data. Only the information in this document is affected. This should have no effect on the M routines.

# VA M PROGRAMMING

## Intermediate

---

### Objective

The objective of this lesson is to complete your system using the following commands and techniques:

- Consolidating code into one function called Enter or Edit Entries
- The NEW command
- Fancy searching techniques
- Validating state codes using global variables
- Validating dates
- An overview of M

### Enhancements to Our Mailing List System

In this lesson, we have enhanced our Mailing List system in one additional way. We have once again reduced the redundancy of our code by combining the Create and Lookup routines into a new menu function called Enter or Edit Entries. Below is a sample interaction where the user's entries are shown in double underline.

#### *M Code Sample*

>D ^VECS8MN

MAILING LIST SYSTEM

OPTIONS      FUNCTIONS

1.....ENTER OR EDIT MAILING LIST ENTRY

2.....PRINT MAILING LIST ENTRIES

ENTER OPTIONS NUMBER: 1

Select MAILING LIST NAME: MIMMM,RALPH

ARE YOU ADDING 'MILLER,RALPH' AS A NEW MAILING LIST (THE 15TH?) Y  
Record # 18

NAME: MIMMM,RALPH//<ENTER>

ADDRESS: 1 ANY AVE.

CITY: DOWNTOWN

STATE: TX

ZIP: 00000

PHONE: 000-333-2345

LAST CONTACT DATE: 01/01/1987

## VA M PROGRAMMING

### Intermediate

---

Select MAILING LIST NAME: MIMMM,RALPH  
NAME: MILLER,RALPH//^

Select MAILING LIST NAME: "MILLER,RALPH"  
ARE YOU ADDING 'MIMMM,RALPH' AS A NEW MAILING LIST (THE 16TH?) Y  
Record # 19

NAME: MIMMM,RALPH//<ENTER>  
ADDRESS: 123 ANY ST.  
CITY: ANYPLACE  
STATE: PA  
ZIP: 00000  
PHONE: 000-111-1111  
LAST CONTACT DATE: 02/02/1990

Select MAILING LIST NAME: MIMMM,RALPH  
1) MIMMM,RALPH 000-333-2345  
2) MIMMM,RALPH 000-111-1111  
CHOOSE 1 - 2: <ENTER>

Select MAILING LIST NAME: "MIMMM,RALPH"  
ARE YOU ADDING 'MIMMM,RALPH' AS A NEW MAILING LIST (THE 17TH?) Y

Record # 20

NAME: MIMMM,RALPH//<ENTER>  
ADDRESS: 10 ANY STREET  
CITY: NOPLACE  
STATE: TX  
ZIP: 00000  
PHONE: 000-333-3333  
LAST CONTACT DATE: 12/12/1990

Select MAILING LIST NAME: <ENTER>

### The Slightly Modified Menu Routine

As you see in the M code sample on the next page, we only have to change the OPTIONS paragraph (M code enclosed in the box is what has been changed) to reflect the consolidation of the Enter or Edit Entries function.

# VA M PROGRAMMING

## Intermediate

### *M Code Sample*

```
VECS8MN;ATL/ACE PROGRAMMER--MAILING LIST SYSTEM; 12/1/90
;;1
; VARIABLE LIST
;
; SPECIAL VARIABLES USED IN VA:
;     DTIME.....Delay time (not to be KILLED)
;     IOF.....contains the codes to start output
;             at the top of a page or screen
; APPLICATION VARIABLES:
;     OPT.....Option number chosen by the user
;     ROUTINE...Routine name for option chosen
;     ITEMS.....Number of items in menu
;     II.....Menu loop counter
;     XDESCR....Text for menu option
MAIN ;
D DISMENU
I OPT="!"(OPT["^") G EXIT
S ROUTINE=$P($T(OPTIONS+OPT),"^",2,3)
D @ROUTINE G MAIN
EXIT ;
K OPT,ROUTINE,ITEMS,XDESCR,II
Q
DISMENU;
W @IOF
W !!!,?20,"MAILING LIST SYSTEM"
W !!,"OPTIONS     FUNCTIONS"
F II=1:1 D DISMENU2 Q:XDESCR=""
S ITEMS=II-1
DISMENU1 ;
R !!,"ENTER OPTION NUMBER: ",OPT:DTIME
Q:OPT="!"('$T)!(OPT["^")
I (OPT<1)!(OPT>ITEMS) W !!,"OPTION NUMBER MUST BE "
I W "BETWEEN 1 AND ",ITEMS
I R !! ,?10,"Press RETURN to continue",OPT:DTIME G DISMENU1
Q
DISMENU2 ;
S XDESCR=$P($T(OPTIONS+II),";;",2)
I XDESCR'="" W !!,II,$P(XDESCR,"^",1)
Q
OPTIONS;
;;.....ENTER OR EDIT MAILING LIST ENTRY^^VECS8ENT
;;.....PRINT MAILING LIST ENTRIES^^VECS8PRI
```



# VA M PROGRAMMING

## Intermediate

---

### The New Enter or Edit Entries Routine

The new routine called Enter or Edit Entries is a combination of the Lesson 7 Create and Lookup routines plus some new code. Below is a list of the paragraphs in the new Enter routine and an indication of their origin.

<i>Paragraph</i>	<i>Origin</i>
ASK	Modification: combined code from Create & Lookup
EXIT	Added some new variables
LOOKUP	Taken directly from Lesson 7 Lookup routine
MATCH	Taken directly from Lesson 7 Lookup routine
CHOOSE	Taken directly from Lesson 7 Lookup routine
REPLY	Taken directly from Lesson 7 Lookup routine
CHEXIT	Taken directly from Lesson 7 Lookup routine
CREATE	New code
CREXIT	New code
ADD	Taken directly from Lesson 7 Create routine

### New Code for the Enter Routine

The new code added to the Enter routine will allow the user to enter a name and have the system check to see if it is a duplicate. If it is, the user will be given the option of adding it to the file or not adding it to the file as shown below.

#### *M Code Sample*

```
Select MAILING LIST NAME: MIMMM,RALPH
ARE YOU ADDING 'MIMMM,RALPH' AS A NEW MAILING LIST (THE 15TH?) N

ARE YOU ADDING 'MIMMM,RALPH' AS A NEW MAILING LIST (THE 15TH?) Y
```

Most of the code for this feature is straightforward, except for what looks deceptively easy: displaying an entry number for additions, for example, the 15th.

# VA M PROGRAMMING

## Intermediate

---

Before we could create an algorithm to do this, we had to discover the pattern in English for adding endings of this nature:

1st	2nd	3rd	4th	5th	6th	7th	8th	9th	
10th	11th	12th	13th	14th	15th	16th	17th	18th	19th
20th	21st	22nd	23rd	24th	25th	26th	27th	28th	29th
30th	31st	32nd	33rd	34th	35th	36th	37th	38th	39th

etc.

One of the first things we see is that all numbers in the teens use the ending *th*. Aside from that, here is the remaining pattern.

***Numbers that end in...                      use the ending....***

0	th
1	st
2	nd
3	rd
4-9	th

We can determine if a record number is between 11 and 19 (they all will have the same ending: *th*):

### ***M Code Sample***

```
SET TEEN=NREC#100>10&(NREC#100<20)
```

For example:

12#100=12	(12th)
112#100=12	(112th)
212#100=12	(212th)

If we know that a number is anywhere in the teens (we've included 11 and 12 with 13 - 19), we can simply apply the *th* ending.

## VA M PROGRAMMING

### Intermediate

---

We can determine the ending of other numbers in the following manner.

#### *M Code Sample*

```
SET MOD=NREC#10
```

For example:

```
1#10=1      (1st)
21#10=1     (21st)
31#10=1     (31st)
```

We can use the MOD variable (the result of NREC#10) as the piece number to pick out the proper ending from a list. We must add 1 to MOD because MOD can be 0 and the first piece number is 1.

#### *M Code Sample*

```
$S(TEEN:"TH",1:$P("TH^ST^ND^RD^TH^TH^TH^TH^TH^TH", "^",MOD+1))
```

Below is the new CREATE paragraph in its entirety. It will be shown with the Enter routine in the next section.

There is actually a flaw in this last code that some might object to. Can you find It?

*Here's a hint: how would the answer y-n-o be evaluated?*

#### *M Code Sample*

```
CREATE;Create a new record
      S NREC=$S($D(^VECS8G(0)):$P(^VECS8G(0), "^",4)+1,1:1)
      S MOD=NREC#10
      S TEEN=NREC#100>10&(NREC#100<20)
      S NREC=NREC_$S(TEEN:"TH",1:$P("TH^ST^ND^RD^TH^TH^TH^TH^TH^TH", "^",MOD+1))
ASKCR ;
      W $C(7),!,?4,"ARE YOU ADDING '",NAME,'" AS A NEW MAILING LIST (THE ",NREC,")?"
      R ANS:DTIME I '$T!(ANS="^") G CREXIT
```

<pre>; The line below includes a trick used to check to see ; if the first letter of the variable ANS is a Y, y, N, or n. I ANS="^"!("NnYy"[\$E(ANS)) W \$C(7),"  ?? " G ASKCR I "Yy"[\$E(ANS) D ADD</pre>
--

```
CREXIT ;
      Q
```



# VA M PROGRAMMING

## Intermediate

---

### The Enter/Edit Routine: VECS8ENT

Shown below is the new code for the Enter function.

There are lines of code that follow that must be typed on one statement but are too long to fit on one line in the lesson. The arrow notations  and  are part of this lesson to show you that we couldn't show you the entire statement on one line.

#### *M Code Sample*

```
VECS8ENT ;LCM;04:19 PM  2 Dec 1989
;          VARIABLE LIST
;
;          FORCE.....Flag to indicate whether a record with
;                      a duplicate name should be forced
;                      into the file; FORCE=1 means bypass
;                      the lookup
;          RECNr..... Record number
;          ANS.....User's response to a question
;          NREC.....Record number with proper ending
;          MOD.....Result of modulo division
;          TEEN.....Record number is in the teens
;          COUNT.....Counter of duplicate names
;          CHOICE.....User's choice from list of duplicates
;          CH.....Selection number on list of duplicates
;          NAMES().....Local array which holds duplicate names
;          BUSY.....Flag that indicates file is locked:
;                      BUSY=1 when file is busy
;          FORCE.....Flag that indicates whether a duplicate
;                      record is being forced into the file:
;                      FORCE=0 if record is NOT forced
;                      FORCE=1 if record IS to be forced
START    ;
W @IOF
ASK      ;
S (FORCE,RECNr)=0
R !!, "Select MAILING LIST NAME: ",NAME:DTIME
I '$T!(NAME["^")!(NAME="") G EXIT
I NAME[""] S NAME=$P(NAME,"",2),FORCE=1
I NAME'?1U.U1", "1U.UP!($L(NAME)>30)!(NAME["?")
I W !, "Enter the name of the desired entry."
I W "Enter last name,first name.",! G ASK
I 'FORCE D LOOKUP I CHOICE="" G ASK
I RECNr'>0 D CREATE I RECNr'>0 W $C(7)," ??" G ASK
D ^VECS8FLD G ASK
EXIT    ;
K ANS,NREC,MOD,TEEN,COUNT,CH
Q
```

# VA M PROGRAMMING

## Intermediate

---

```

LOOKUP      ;
            S COUNT=0

            F RECNR=0:0 S RECNR=$O(^VECS8G("B",NAME,RECNR)) Q:RECNR!>0
            I $D(^VECS8G(RECNR,0)), $P(^VECS8G(RECNR,0), "^", 1)=NAME D MATCH
            S CHOICE=""
            I COUNT=0 S CHOICE="None" Q
            I COUNT=1 S CHOICE=1
            I COUNT>1 D CHOOSE
            I $D(NAMES(+CHOICE)) S RECNR=NAMES(CHOICE)
            K NAMES
            Q
MATCH       ;
            S COUNT=COUNT+1
            S NAMES(COUNT)=RECNR, NAMES(COUNT, 0)=^VECS8G(RECNR, 0)
            Q
CHOOSE      ;
            F CH=1:1:COUNT W !,CH,"", $P(NAMES(CH,0), "^", 1), ?40, $P(NAMES(CH,0), "^", 6)
            I CH#5=0!(CH=COUNT) D REPLY Q:CHOICE!=""
            Q
REPLY       ;
            W !, "CHOOSE 1 - ",CH,"": " R CHOICE:DTIME
            I '$T!("^"[CHOICE) G CHEXIT
            I '$D(NAMES(CHOICE)) W !, ?10, "Must be a number "
            I W "from 1 through ",CH,!, $C(7) G REPLY
CHEXIT      ;
            Q
CREATE      ;Create a new record

            S NREC=$S($D(^VECS8G(0)): $P(^VECS8G(0), "^", 4)+1, 1:1), MOD=NREC#10,
            TEEN=NREC#100>10&(NREC#100<20)
            S NREC=NREC_$S(TEEN:"TH", 1:$P("TH^ST^ND^RD^TH^TH^TH^TH^TH", "^", MOD+1))
ASKCR       ;
            W $C(7), !?4, "ARE YOU ADDING '", NAME, "'" AS A NEW MAILING LIST (THE ", NREC, ")? "
            R ANS:DTIME I '$T!(ANS="^") G CREXIT
            I ANS=""! ("NnYy"[$E(ANS)) W $C(7), " ?? " G ASKCR
            I "Yy"[$E(ANS) D ADD
CREXIT      ;
            Q
ADD         ;
            L ^VECS8G(0):5 I '$T W !, "File is busy, try again later." S BUSY=1 Q
            I '$D(^VECS8G(0)) S ^VECS8G(0)="MAILING LIST^1^0^0"
            F RECNR=$P(^VECS8G(0), "^", 3)+1:1 I '$D(^VECS8G(RECNR)) Q
            S $P(^VECS8G(0), "^", 3, 4)=RECNR_"^"_$P(^VECS8G(0), "^", 4)+1
            S ^VECS8G(RECNR, 0)=NAME, ^VECS8G("B", NAME, RECNR)=" " L
            W !, "Record # ", RECNR, !
            Q



```

# VA M PROGRAMMING

## Intermediate

### The Edit Fields Routine

There were no changes made to the Edit fields routine.

There are lines of code that follow that must be typed on one statement but are too long to fit on one line in the lesson. The arrow notations  and  are part of this lesson to show you that we couldn't show you the entire statement on one line.

### *M Code Sample*

```
VECS8FLD ;ATL/ACE PROGRAMMER--MAILING LIST SYSTEM; 12/1/90
; ;1
; VARIABLE LIST
; RECNR.....Record number
; RECORD.....copy of data node
; NAME.....Name as stored in the record
; OLDNAME.....Original name as stored in the record
; ADDR.....Address
; CITY.....City
; STATE.....State
; ZIP.....Zip code
; PHONE.....Phone number
; LCD.....Internal form of last contact date
; OLDDATE.....Display form of date as stored in node
; ANS.....User's answer to delete? questions
; INDATA.....Holds users input before validation
; OK.....Flag to designate if user is deleting
; record (OK=1 says record is to be
; deleted)
;Edit data fields
START ;
;Edit data fields
L ^VECS8G(RECNR,0):5
I '$T W !,"File is busy, try again later" Q
S RECORD=^VECS8G(RECNR,0)
S (OLDNAME,NAME)=$P(RECORD,"^",1),ADDR=$P(RECORD,"^",2)
S CITY=$P(RECORD,"^",3)
S STATE=$P(RECORD,"^",4),ZIP=$P(RECORD,"^",5)
S PHONE=$P(RECORD,"^",6)
S LCD=$P(RECORD,"^",7)
I LCD="" S OLDDATE=$E(LCD,4,5)_"_"$E(LCD,6,7)_"_"($E(LCD,1,3)+1700)
I LCD="" S OLDDATE=""
NAME ;
W !,"NAME: ", $S(NAME="": "",1:NAME_//") R INDATA:DTIME
I '$T!(INDATA["^") G EDEXIT
G: INDATA="" ADDR
I INDATA="@",INDATA'?1U.U1", "1U.UP!($L(INDATA)>30)!(INDATA["?")
I W !,"Enter the name of the desired entry. "
I W "Enter last name,first name.",! G NAME
I INDATA="@ S OK=0 D DELETE G EXIT:OK,NAME
S NAME=INDATA
```

# VA M PROGRAMMING

## Intermediate

---

```
ADDR
;
W !,"ADDRESS: ",$$$(ADDR="":",1:ADDR_//") R INDATA:DTIME
I '$T!(INDATA["^") G EDEXIT
G:INDATA="" CITY
I INDATA'="@",INDATA'?.ANP!($L(INDATA)>25)! (INDATA["?")
I W !,"Address must not exceed 25 characters.",!,$C(7) G ADDR
I INDATA="@ " D DELFLD I INDATA'="" G ADDR
S ADDR=INDATA

CITY
;
W !,"CITY: ",$$$(CITY="":",1:CITY_//") R INDATA:DTIME
I '$T!(INDATA["^") G EDEXIT
G:INDATA="" STATE
I INDATA'="@",INDATA'?.ANP!($L(INDATA)>20)! (INDATA["?")
I W !,"City must not exceed 20 characters.",!,$C(7) G CITY
I INDATA="@ " D DELFLD I INDATA'="" G CITY
S CITY=INDATA

STATE
;
W !,"STATE: ",$$$(STATE="":",1:STATE_//") R INDATA:DTIME
I '$T!(INDATA["^") G EDEXIT
G:INDATA="" ZIP
I INDATA'="@",INDATA'?2U W !,"State must be a 2 character "
I W "abbreviation.",!,$C(7) G STATE
I INDATA="@ " D DELFLD I INDATA'="" G STATE
S STATE=INDATA

ZIP
;
W !,"ZIP: ",$$$(ZIP="":",1:ZIP_//") R INDATA:DTIME
I '$T!(INDATA["^") G EDEXIT
G:INDATA="" PHONE
I INDATA'="@",INDATA'?5N&(INDATA'?9N) W !,"Zip must be five or "
I W "nine digits, no hyphen.",!,$C(7) G ZIP
I INDATA="@ " D DELFLD I INDATA'="" G ZIP
S ZIP=INDATA

PHONE
;
W !,"PHONE: ",$$$(PHONE="":",1:PHONE_//") R INDATA:DTIME
I '$T!(INDATA["^") G EDEXIT
G:INDATA="" LCD
I INDATA'="@",INDATA'?3N1"- "3N1"- "4N&(INDATA'?3N1"- "4N)
I W !,"Phone must be entered with or without area code, using "
I W "hyphens.",!,$C(7) G PHONE
I INDATA="@ " D DELFLD I INDATA'="" G PHONE
S PHONE=INDATA

LCD
;
W !,"LAST CONTACT DATE: ",$$$(OLDDATE="":",1:OLDDATE_//")
R INDATA:DTIME
I '$T!(INDATA["^") G EDEXIT
G:INDATA="" DATE
I INDATA'="@",INDATA'?2N1"/ "2N1"/ "4N W !,"Enter a date in the "
I W "format of MM/DD/YYYY, where MM & DD are each two "
I W "digits and YYYY is 4 digits.",!,$C(7) G LCD
I INDATA="@ " D DELFLD I INDATA'="" G LCD

DATE
;
S:INDATA'="" LCD=($E(INDATA,7,10)-1700)_$E(INDATA,1,2)_$E(INDATA,4,5)
S:INDATA=""&(OLDDATE'="") LCD=($E(OLDDATE,7,10)-1700)_$E(OLDDATE,1,2)_$E(OLDDATE,4,5)
I INDATA=""&(OLDDATE'="") S LCD=""
```

---

# VA M PROGRAMMING

## Intermediate

---

```

EDEXIT      ;
             S $P(^VECS8G(RECNR,0),"^",1,7)=NAME_"^"_ADDR_"^"_CITY_"^"_
             ^_STATE_"^"_ZIP_"^"_PHONE_"^"_LCD
             I OLDNAME'=NAME K ^VECS8G("B",OLDNAME,RECNR)
             I S ^VECS8G("B",NAME,RECNR)=" "
             L
EXIT         ;
             K NAME,ADDR,CITY,STATE,ZIP,PHONE,LCD,OLDDATE,ANS,INDATA
             K OK,OLDNAME,RECNR,RECORD
             Q
DELFLD      ;
             R !,"Sure you wish to delete ? ",ANS:DTIME
             I '$T!(ANS["^")!(ANS=" ") GOTO DELFEXIT
             I (ANS="Y")!(ANS="YES") S INDATA=" "
             I W " ... <FIELD DELETED>"
             E W " ... <FIELD NOT DELETED>"
DELFEXIT    ;
             QUIT
DELETE      ;
             W !,"Sure you want to delete the entire '",NAME,"' MAILING LIST ? "
             R ANS:DTIME I '$T!(ANS["^")!(ANS=" ") G DELEXIT
             I (ANS'="Y"),(ANS'="YES") W !," ... not deleted !",$C(7),!
             I G DELEXIT
             K ^VECS8G(RECNR),^VECS8G("B",NAME,RECNR)
             L ^VECS8G(0):5
             I '$T W !,"File is busy, try again later" Q
             S $P(^VECS8G(0),"^",4)=$P(^VECS8G(0),"^",4)-1 L
             W !," '",NAME,"' DELETED !",! S OK=1
DELEXIT     ;
             Q

```



# VA M PROGRAMMING

## Intermediate

---

### The NEW Command

As a programmer, two of the tasks you may encounter are:

1. To write a routine that will be used by other programmers for a special purpose. Often these routines are called utility programs.
2. To access another programmer's routine without having access to their M code.

In either case, you may not know how the other programmer(s) named their variables. What if they used some of the same variable names as you did?

The purpose of the NEW command is to address the concern of how to avoid overlapping variable names with another programmer. When you use the NEW command, M will save the current values of the variables you've specified and allow you to reassign values to those variables until you encounter the QUIT at the end of the subroutine. When M encounters the end of the subroutine, it will restore the original values of the variables.

There are three forms of the NEW command:

#### **NEW**

This form takes all variables currently specified and stores their values (called the *argumentless NEW*). This form uses a lot of resources and is not recommended.

#### **NEW A,B**

This form takes only the values of variables A and B and stores their values.

#### **NEW (C)**

This form takes all values except C and stores them (called the *exclusive NEW*)



#### **VA Programming Standards and Conventions**

- 1) The argumentless form of the NEW command may not be used.
- 2) The exclusive form of the NEW command may not be used.
- 3) Every NEW command must have an explicit, corresponding QUIT command.

# VA M PROGRAMMING

## Intermediate

---

Shown below are various examples of the use of the NEW command. Study them carefully. It is important to understand how the values of variables can be different in routines or subroutines where the NEW command has been used.

What the user typed is double underlined in the M code samples below.

### *M Code Sample*

```
VECS81      ;
            SET A=15
            SET B=20
            DO SUB1
            QUIT
SUB1      ;
            NEW A
            WRITE !,"THE VALUE OF A IS NOW: ",A
            QUIT

DO VECS81

THE VALUE OF A IS NOW:
<UNDEF>SUB1+2:3 A
```

The value of A is undefined after the NEW A command because we have not yet reassigned it.

### *M Code Sample*

```
VECS82      ;
            SET A=15
            SET B=20
            DO SUB1
            WRITE !,"AFTER SUB1, THE VALUE OF A IS: ",A
            QUIT
SUB1      ;
            NEW A
            SET A="ABCDEF"
            WRITE !,"IN SUB1, THE VALUE OF A IS NOW: ",A
            WRITE !,"IN SUB1, THE VALUE OF B IS: ",B
            QUIT

DO VECS82

IN SUB1, THE VALUE OF A IS NOW: ABCDEF
IN SUB1, THE VALUE OF B IS: 20
AFTER SUB1, THE VALUE OF A IS: 15
```

# VA M PROGRAMMING

## Intermediate

---

### Fancy Searching Technique

One of the types of searches we may be required to do is by the first letter of the individual's last name. Below is a partial global listing from our sample database.

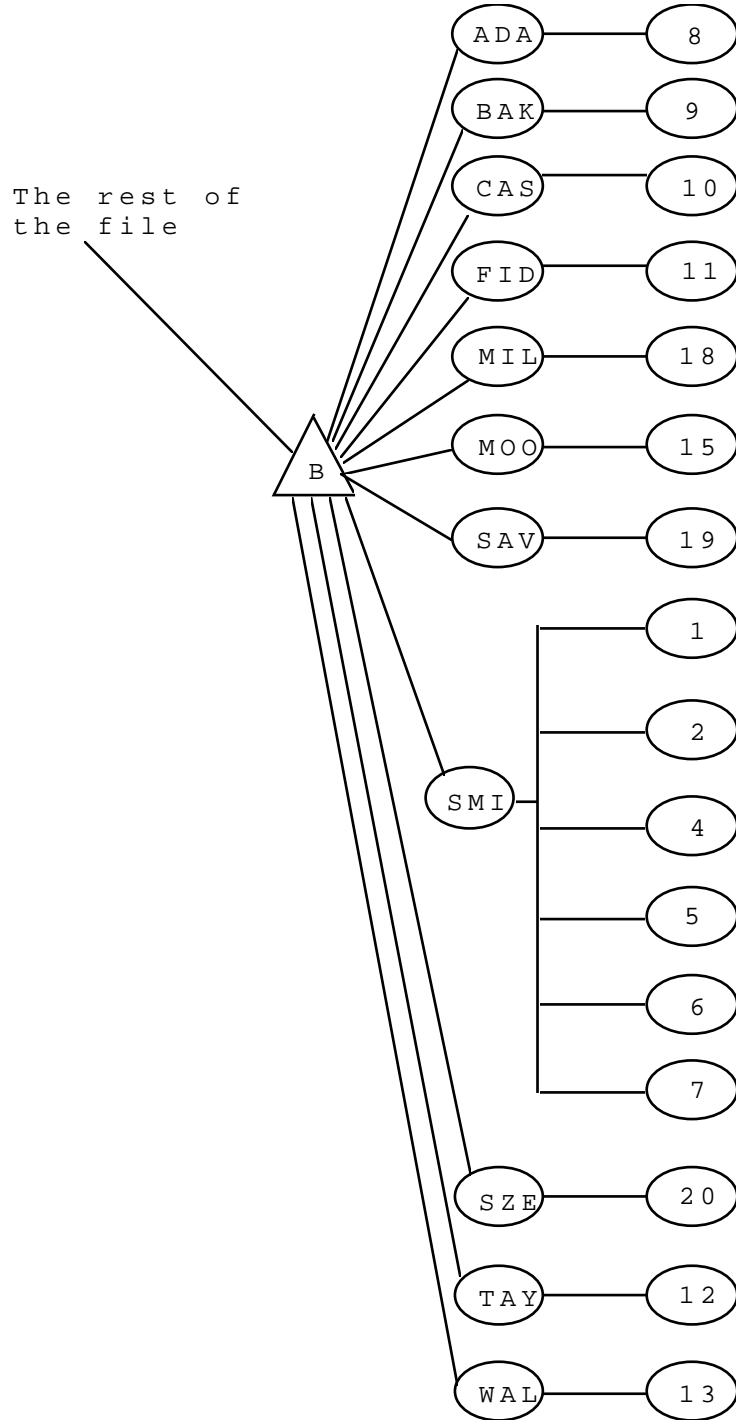
#### *M Code Sample*

```
.... (header and data nodes not shown)
^VECS8G("B","ADAAA,JOHN",8)
^VECS8G("B","BAKKK,RALPH",9)
^VECS8G("B","CASSS,BILL",10)
^VECS8G("B","FIDDD,MICKEY",11)
^VECS8G("B","MILLL,RALPH",18)
^VECS8G("B","MOOOO,LINDA",15)
^VECS8G("B","SAVVV,JESSICA",19)
^VECS8G("B","SMIII,BILL",1)
^VECS8G("B","SMIII,BILL",2)
^VECS8G("B","SMIII,BILL",4)
^VECS8G("B","SMIII,BILL",5)
^VECS8G("B","SMIII,BILL",6)
^VECS8G("B","SMIII,BILL",7)
^VECS8G("B","SZETCZXEG,RALPH",20)
^VECS8G("B","TAYYY,MARY",12)
^VECS8G("B","WALLL,RON",13)
```

# VA M PROGRAMMING

## Intermediate

The diagram below graphically displays the same data that was shown in the partial global listing on the previous page. Here you see the first three letters of the last name and the various record number or numbers that are associated with persons whose last name begins with those three letters.





# VA M PROGRAMMING





## Intermediate

---

If we use the same techniques we have been using for searches, we might use \$ORDER to move through all the index nodes. Below is a sample routine that will isolate the proper range of index nodes so that we do not have to process all the index nodes.

There are lines of code that follow that must be typed on one statement but are too long to fit on one line in the lesson. The arrow notations  and  are part of this lesson to show you that we couldn't show you the entire statement on one line.

### *M Code Sample*

```
VECS83      ;
            ; VARIABLE LIST
            ; FIRST      = Letter to search by
            ;              as entered by user
            ; BEFOREFR   = Letter of the alphabet
            ;              that is before the
            ;              letter entered by the
            ;              user
            ; NAME       = NAME subscript
            ; RECNR      = Record nr. subscript
            ;
            READ !,"ENTER FIRST LETTER OF NAME: ",FIRST
            DO FIND
            QUIT
FIND      ;
          SET BEFOREFR=$A(FIRST)-1
          SET NAME=$C(BEFOREFR)_"zzzz"
          F X=1:1 S NAME=$O(^VECS8G("B",NAME))
          Q:NAME](FIRST_"zzzz")!(NAME="") DO DISPLAY
          QUIT
DISPLAY   ;
          SET RECNR=""
          F X=1:1 S RECNR=$O(^VECS8G("B",NAME,RECNR))
          Q:RECNR="" DO DISPl
          QUIT
DISPl     ;
          WRITE !,"RECORD NUMBER OF ",NAME," IS ",RECNR
          QUIT
```

## VA M PROGRAMMING

### Intermediate

---

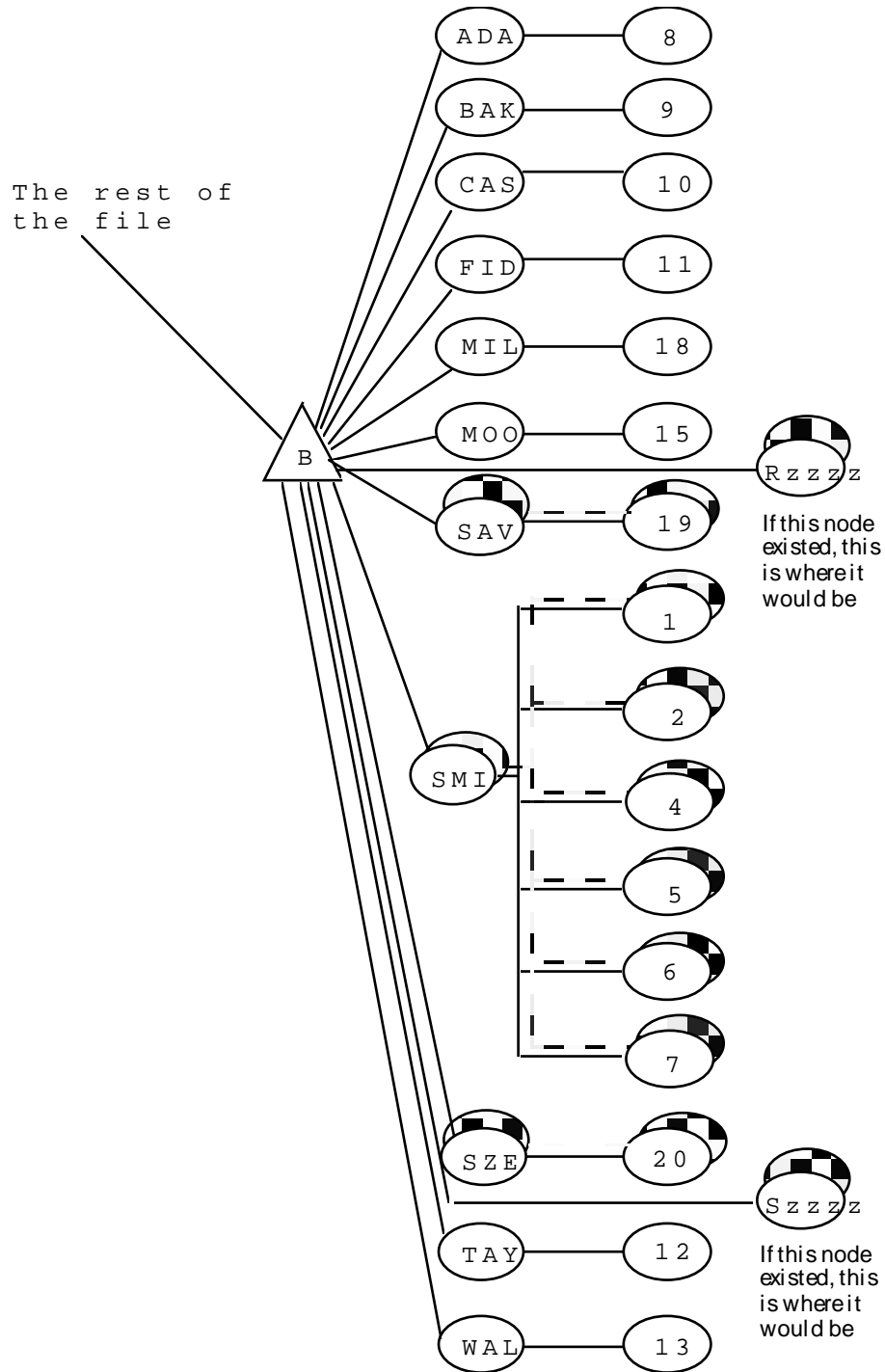
When we run the routine on the previous page, we get only those names that begin with S as shown below:

```
ENTER FIRST LETTER OF NAME: S
RECORD NUMBER OF SAVVV,JESSICA IS: 19
RECORD NUMBER OF SMIII,BILL IS: 1
RECORD NUMBER OF SMIII,BILL IS: 2
RECORD NUMBER OF SMIII,BILL IS: 4
RECORD NUMBER OF SMIII,BILL IS: 5
RECORD NUMBER OF SMIII,BILL IS: 6
RECORD NUMBER OF SMIII,BILL IS: 7
RECORD NUMBER OF SZETCZXEG,RALPH IS: 20
```

# VA M PROGRAMMING

## Intermediate

This page displays where names beginning with Rzzzzz and Szzzzz would fall in the list. These were purposely selected so that a name would not begin with these letters. A letter followed by z's could safely be used as a separator between groups of names beginning with successive letters.



# VA M PROGRAMMING

## Intermediate

---

### Validating State Codes

When validating data, sometimes you will need to compare the user's input to a specific list of codes. State codes might be one example of this. We would start by creating global variables that would contain all the state codes, as shown in the M code sample below. This only has to be done once.

*What the user typed is double underlined in the samples that follow.*

#### *M Code Sample*

```
VECS8SET  ;
    SET ^VECST1="/AL/AK/AZ/CA/CO/CT/"
    SET ^VECST2="/DE/DC/FL/"
    QUIT
```

DO ^VECS8SET

we would have as many global variables as we need to contain all the state codes OR put all the state codes into one global variable.

Next, we can use code such as that shown in the M code sample below to check the state codes. Since the state codes are separated by a slash in the global variables, we must concatenate slash's around the user's input so we can check for an exact match.

#### *M Code Sample*

```
VECS8STA ;
STATE    ;
    READ !,"ENTER STATE CODE: ",STATE
    DO CHECK
    IF ERROR GOTO STATE
    QUIT
CHECK    ;
    SET ERROR=0
    SET STATE1="/ "_STATE_"/"
    IF ^VECST1[STATE1 QUIT
    IF ^VECST2[STATE1 QUIT
    WRITE !,"STATE CODE NOT VALID!! TRY AGAIN."
    SET ERROR=1
    QUIT
```



# VA M PROGRAMMING

## Intermediate

---

The M code sample below shows a typical user interaction when incorrectly entering a state code.

### *M Code Sample*

```
DO ^VECS8STA  
  
ENTER STATE CODE: RZ  
STATE CODE NOT VALID!! TRY AGAIN.  
ENTER STATE CODE: AP  
STATE CODE NOT VALID!! TRY AGAIN.  
ENTER STATE CODE: FL
```

### Validating Dates

In validation of dates, there are three pieces that can be validated: month (1 through 12), years (within a reasonable range), and days of the month. Checking days of the month can be tricky since we have a different number of days in every month and in leap years, February has a different number of days. Below is code that will check a date, including days of the month.

### *M Code Sample*

```
VECS8DT ;  
DATE      ;  
    SET ERROR=0  
    READ !,"ENTER THE DATE: ",XDATE  
    SET MON=$P(XDATE,"/",1)  
    SET XDAY=$P(XDATE,"/",2)  
    SET YR=$P(XDATE,"/",3)  
    IF MON<1!(MON>12) DO ERRMSG G DATE  
DAYS      ;  
    D ERRMSG:$P("31,"_(YR#4=0+28)_,31,30,31,30,31,31,30,31,30,31"," ",MON)<XDAY  
    G DATE:ERROR  
    QUIT  
ERRMSG    ;  
    SET ERROR=1  
    WRITE !,"REENTER DATE"  
    QUIT
```

Line DAYS+1 contains the validation for days of the month. There are three parts to that line of code.

## VA M PROGRAMMING

### Intermediate

---

The previous page indicated that line DAYS+1 contains the validation for days of the month and that there are three parts to that line of code.

1. DO ERRMSG if
2. Actual days for MON
3. Are less than days entered by user

#### *M Code Sample*

```
D ERRMSG:$P("31,"_(YR#4=0+28)_"",31,30,31,30,31,31,30,31,30,31","",MON)<XDAY
| 1 | 2 | 3 |
```

Part 2 is important to understand. The \$PIECE function can have three parameters:

\$P (string to be searched,delimiter,piece number)

In Part 2 of our line, the three parameters specifically are:

\$P("string of days/mon.,"delimiter of comma",MON used as piece number)

**NOTE:** If you carefully followed the year 2000 issues, you may know that year modulo 4 is not correct for some centuries divisible by 4 and that an additional operation would be necessary in this code.

Since the month numbers are consecutive from 1-12, we can list the respective days per month in positions 1-12 of the string of days/mon. Then using "," as the delimiter in the string of days/month and MON as the piece number, we pull out piece number MON. For example, if MON=3, we pull out piece number 3 which is 31, the number of days in March.

# VA M PROGRAMMING

## Intermediate

---

You will notice that piece number 2 of the "string of days/mon" is not just a number but is an expression.

### *M Code Sample*

( YR#4=0+28 )

This expression is evaluated from left to right and can be summarized as:

1. **YR#4:** divide YR by 4 and give the remainder (modulo division)
2. **Compare result of Step 1=0:** compare the remainder to 0
  - a. if the remainder is 0, the YR is evenly divisible by 4 and is a leap year and the truth value is 1
  - b. if the remainder is not 0, the truth value is 0 and it's not a leap year
3. **Add result of Step 2+28:** add the truth value to 28

Shown below are two examples of this. Take time to study the examples closely.

#### **Example 1:**

assume YR=84

Step 1.  $84 \div 4$  gives us a remainder of 0  
therefore

Step 2.  $0=0$  is true (1)

Step 3.  $1+28=29$  days in Feb. in a leap year

#### **Example 2:**

assume YR=86

Step 1.  $86 \div 4 = \text{remainder of } 2$   
therefore

Step 2.  $2=0$  is false (0)

Step 3.  $0+28=28$  days in Feb. when YR is not a leap year

### M System Architecture

The graphic on the next page shows a complete M system.

The top right shows your little part of the world -- the memory where your routines and variables are stored along with some execution control data needed by the system. This section of memory is just one partition. Many partitions usually exist. Most represent users like you but some may be background jobs running unattended or system jobs.

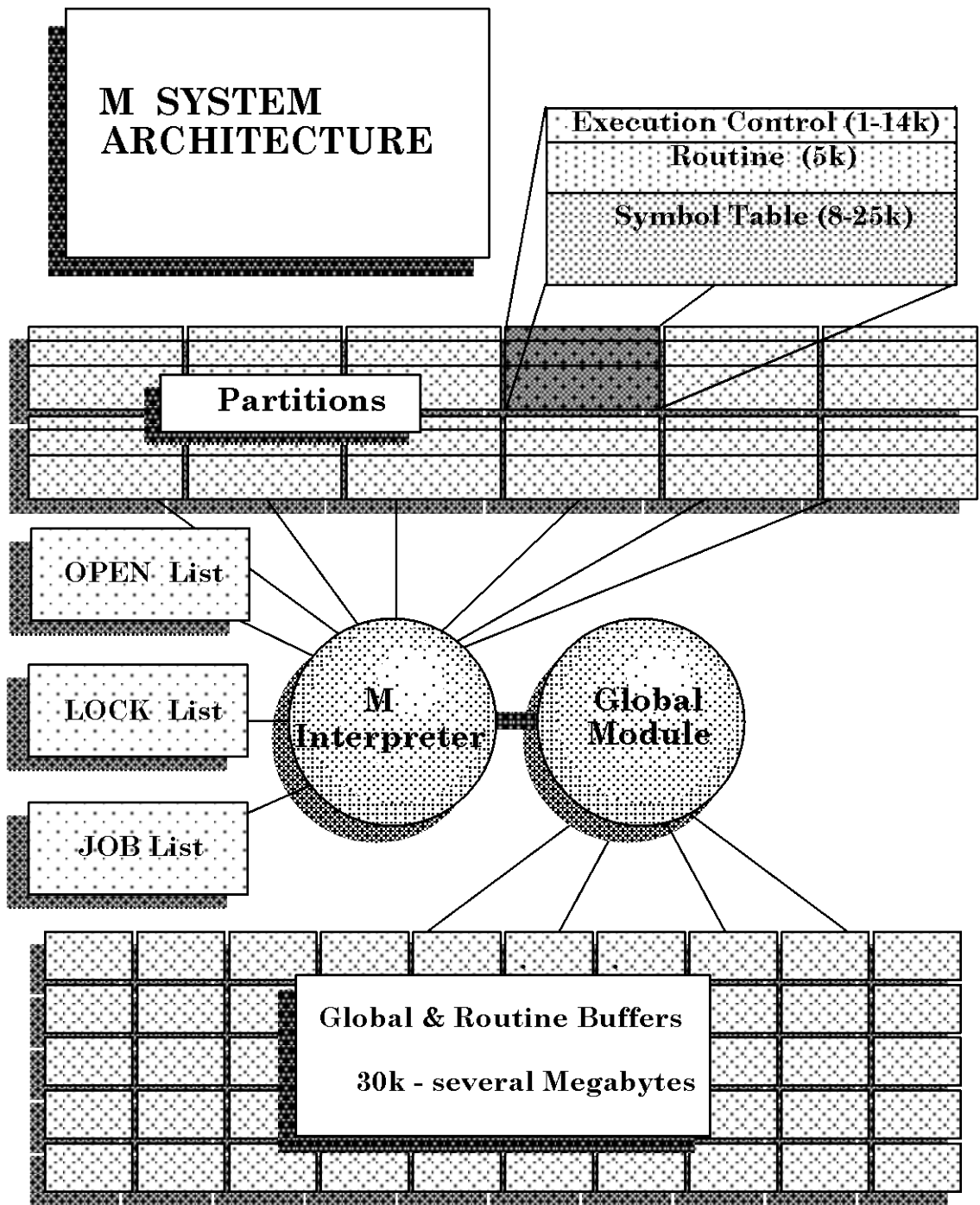
The M Interpreter executes your routine and obtains data from the database using the Global Module. Shown are some of the lists that the M system uses such as the list of jobs, devices in use, and locked nodes.

Finally, the graphic shows an area called Global and Routine buffers, which are copies of routines and data from the database that are kept in memory in order to speed up execution of the system.

# VA M PROGRAMMING

## Intermediate

---



# VA M PROGRAMMING

## Intermediate

---

### Overview of M

#### Who uses M ?

For a list of software vendors, see the MUG Publication *MUMPS Software Sources*.

There are an estimated 15,000 users of M in North America and 23,000 M users in Europe with additional users in the United Kingdom, Japan, Brazil, and SE Asia.

A partial list:

- Veteran's Health Administration, 400 hospitals and clinics
- Indian Health Services, 200 hospitals and clinics
- Navy Occupational Health Information Management System, 168 sites
- Shared Medical System Corp.
- HBO & Co.
- Baxter International
- National Computer Systems, Inc.
- Chase Manhattan Bank
- The May Company
- American Airlines
- The Swiss government
- The Smithsonian Institution
- Registrar Data Group, who used a M system to register delegates to the Democratic National Convention

#### On What Systems Does M Run?

For a complete list, see "MUMPS Marketplace Today", MUG Quarterly, Vol. XXI, No. 5, 1991.

#### *Operating Systems*

CP/M	CTOS	DG AOS	Harris VOS
IBM VM	Macintosh O/S	Modulex	MS-DOS
LISP	OS-9	UNIX	VAX/VMS
VS			

# VA M PROGRAMMING

## Intermediate

---

### *Computers*

Apple	NCR
AT&T	NEC
BBN	Nippon
Burroughs	Nixdorf
CMS	Okitac
Cobra	Phillips
Codata	Pixel
Codex	Perkin Elmer
Convergent Technologies	Plexus
Counterpoint Computers	Pyramid Technology Corp.
CNET	Radio Shack
DEC	Sharp
Data General	Smoke Signal Broadcasting (SSB)
EXXON Office Systems	Sumitomo
Fujitsu	Tandem
Gould	Tandy
Harris	Texas Instruments
Hazelwood	Tokyo Sanyo
Hewlett Packard	Tosmec
Hitachi	UNIX Systems
Honeywell	Wang
IBM microcomputers (compatibles)	8080 Z980 CP/M systems
and mainframes	8086/88CP/M86 Systems
Motorola	

### *How Does M Compare with Other Languages?*

1. Chase Manhattan Bank in New York reported that M allows programmers to write 30-40 line applications with the power of 600-900 lines of COBOL.

*Smith, J. Y. & Harvey, W. J. M is germinating business productivity.  
Journal of Systems Management, April 1988, 26-31.*

2. A benchmark study was done for a department store in Spain in which a database was created with six files. Code was written in M, COBOL, FORTRAN IV to do several queries. The results are shown on the next page.

## VA M PROGRAMMING

### Intermediate

---

	<i><b>M</b></i>	<i><b>COBOL</b></i>	<i><b>FORTRAN</b></i>
Lines of source code	183	1050	2684
Database occupation (Mbytes)	6.073	21.700	23.083

*Alonso, C. A case for M. Computerworld, Jan. 9, 1984, 27-32.*

3. In an article, Tom Munnecke relates the results of translating a COBOL program into M. The COBOL program was part of a payroll system which received a batch of time and attendance records and computer the gross and net pays, various leave balances, etc. In programmers' hours, the M system saved five to eight man months.

ITEM	COBOL	M	PERCENT
Lines of code	3600	300	8
IF statements	460	89	19
GOTO statements	650	43	6
Total program size	120K	9K	8

*Munnecke, T. A linguistic comparison of M and COBOL, Proceedings of the National Computer Conference, 1980, 723-729.*

### ***What Are the Strengths of M?***

1. Standardized:
  - ANSI
  - FIPS (Federal Information Processing Standard)
2. Includes functionality from several categories of information systems software:
  - Programming language: READ, WRITE, SET
  - Database management system: \$ORDER, \$DATA
  - Operating system: OPEN, USE, CLOSE
  - Artificial intelligence language support system: file structure supports semantic networks



# VA M PROGRAMMING

## Intermediate

---

3. M offers unique economic advantages over other software technologies:
  - Decreased development time
  - Less hardware needed
  - Less programmer support needed
  - Less managerial overhead needed
4. Support tools are available:
  - Programming:
    - Editors
    - Cross-reference utilities
  - Database:
    - MQL
    - FileMan (public domain)
5. Supported by:
  - M Development Committee (MDC) which began in 1972
  - M Users Group (MUG) which has been active since 1972
6. Unique features within the language:
  - Line oriented language: multiple commands can be put on one line
  - Only one data type: string; variable elements do not have to be declared before being set to a value
  - File structure:
    - shared, hierarchical, sparse arrays; (no need to declare max. size of file)
    - facilities to define and maintain these file are built into the language
    - records and fields are represented as nodes in a tree (concatenate fields together, \$PIECE apart)
    - no need to open, read, write a file: SET ^NAME=value
    - multiply nested keys with string or numeric subscripts
    - files can be accessed randomly or sequentially
  - "M means never having to say you're sorting" (from a tee-shirt sold by MUG years ago); array subscripts are automatically maintained in sorted order (no need for separate sort utilities)
  - Indirection: M code can be put into variables and executed
  - Pattern match for data validation

## VA M PROGRAMMING

### Intermediate

---

#### *What Is the Future of M?*

1. There will continue to be rapid growth:
  - More implementations
  - Increased numbers and diversity of users
2. M will be used more in expert systems and natural language processing
3. M will be given the ability to handle non-text information: object-oriented data (digitized images, voice recordings, etc.)

*The entire issue of The MUG Quarterly, Vol. XVII, No. 4, Winter 1989.*

4. There will be interfaces with other national and standards bodies:
  - International Standards Organization (ISO)
  - Graphics standards (international graphics standard GKS)
  - Distributed data/network protocols

*The entire issue of The MUG Quarterly, Vol. XIX, No. 4, April 1990.*

- Office automation
- International relational database standard SQL

*Grabscheid, P. The 'relational-ization' of MUMPS, Computers in Healthcare, MUMPS Special Edition, Vol. 9, No. 7, 1988, pp. 16-18.*

*Harvey, W. J. Update on SQL for M users. MUG Quarterly, Vol. XVII, Nr. 2, pages 11-12.*

# VA M PROGRAMMING

## Intermediate

---

### Assignment (optional)

#### In General

- Follow the VA Programming Conventions as illustrated in this lesson.
- Use only the techniques, commands, and features described in this lesson or the previous lessons.
- Save the routine(s) under the namespace assigned to you. For this lesson, use the following suffix pattern:
  - VEC8A for the menu routine
  - VEC8B for the enter or edit function (called from main menu) routine
  - VEC8C for the edit fields routine
  - VEC8E for the print menu routine
  - VEC8F for the print by record number routine
  - VEC8G for the print by name routine
  - VEC8H for the print selected individuals
  - VEC8I for the print selected last names
  - VEC8J for the print selected names that begin with the same letter
- You will need to create another global file. In your routine, use the following suffix pattern:
  - VEC8G *where G stands for global*
- You will need to create one or more global variables to contain the state codes. Use the following suffix pattern:
  - VEC8A for the first global variable
  - VEC8B for the second global variable (if you use one)

#### Mandatory Specifications

You are to redesign and reimplement the personnel system you first developed in Lesson 5 and enhanced in Lessons 6 and 7. You should now follow the Sample System in this lesson.

The fields and their validation are the same as in previous lessons except where noted below.

To summarize, your final system should have two menu options:

1. Enter or Edit entries
2. Print entries

# VA M PROGRAMMING

## Intermediate

---

You must keep your file structure as it was in Lesson 7 (including the index nodes). You must have one routine to handle both the Enter or Edit Entries function. When adding a new entry, this function should show the entry number with the appropriate ending (e.g., 1st, 2nd, etc.).

The fields and their validation are the same as in Lesson 7.

**NOTE:** Please make sure that your system prompts for the fields in the same order as the previous lesson. Also, follow each prompt with a colon(:) and a space. For example, your prompt for the name field might be:

**Enter Name** |\_**:**|\_ *where |\_**:**|\_ indicates a space.*

### ***Important***

Add the following features to your final system (the exact procedures have not been integrated into the sample routine but the features below can be created using techniques and structures that you have used throughout the course):

1. In the Enter or Edit Entries routine, if the user enters a ?? to the Name prompt, list all the names in alpha order that are currently in the file (use the "B" name index nodes for this; even though in FileMan when you type ?? you do not get duplicates shown, for the purpose of this class, please show any duplicate names). Show a group of 10 names at one time. In this lesson, you need only show the user the list of names; they don't have to enter a selection number to choose. You may optionally add the ability for the user to choose from a list of selections (which is much like the procedure for selecting from a list of duplicate names).
2. In the edit routine, use the date validation techniques shown in this lesson to validate the date entry.
3. Use the state code validation technique to validate the state code entry (make sure you put the code to create the global(s) in your menu routine; also, use the namespace for your state code global variables shown at the beginning of the assignment). Make sure you have included at least the following states: GA, PA, TX, NY, FL, DE, OR, CA.
4. In the Print menu routine there will be five options. Each of the print options should use the same output format as Lesson 7 and print the data for all the fields. Display a print menu that will allow the user to:
  - a. Display the output sorted by record number as we did in Lessons 6 and 7.
  - b. Print all records in order by name (including duplicates) as we did in Lesson 7.

# VA M PROGRAMMING

## Intermediate

---

- c. Enter an individual's name and get a report for just that individual; if there are duplicates of the individual's name, list them and allow the user to choose the one they want (just as in the old lookup routine)
- d. Obtain a listing of all individuals whose last names start with a certain letter (e.g., all those whose last names start with M). Use the technique shown in this lesson.
- e. Obtain a listing of all individuals with the same last name (e.g. Moore). Use the technique shown in this lesson.