



M

VA M Programming Introduction

Lesson 4

Putting a System Together

VA M PROGRAMMING

Introduction

Table of Contents

Guide to Symbols Used in Some Lessons	4
Optional Reading References	4
References	4
Document Update Note	4
Objectives	5
Routine Structure	5
Checking the size of your routine with \$STORAGE	5
Modularity	6
Modularity using the Do command	7
Making code more compact	9
Routine Efficiency	10
Routine Matenance	10
Input validation	11
Single character READ	11
READ with a READ count	12
Pattern match	12
Control of timing with HANG	17
Intrinsic functions	18
Functions from previous lessons	18
\$FIND() function	18
\$EXTRACT() function	19
\$LENGTH() function	21
\$ASCII() function	22
\$CHAR() function	22
Sample routine using \$ASCII and \$CHAR	23
\$RANDOM() function	25
Sample routine using \$RANDOM	26
Arrays	27
Sample routine using arrays and intrinsic functions	28
Sample systems	34

VA M PROGRAMMING

Introduction

Table of Contents *(Continued)*

Assignment (optional)	41
Assignment specifications	41

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 and reviewing material. Sincere appreciation to:

Debbie Channell
Tuscaloosa OI Field Office, Customer Service



Harris H. Lloyd
Central Arkansas Veterans Health Care System

Bob Lushene
Hines OI Field Office, Technical Service

VA M PROGRAMMING

Introduction

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

Introduction

Objective

The objective of this lesson is to prepare you to put a complete system together using the following commands and techniques:

- Maximum sizes of routines and symbol table and how to check them.
- Creating modular systems.
- Making routines more compact and efficient.
- Maintainability concerns.
- Input validation techniques including single character READ, READ with READ count, pattern match.
- Control of timing with HANG.
- Intrinsic functions including \$FIND, \$EXTRACT, \$LENGTH, \$ASCII, \$CHAR, \$RANDOM.
- Using arrays for data search operations.

Routine Structure

Checking the Size of Your Routine with \$STORAGE

When you logon to the system, you are automatically given a partition (workspace) of a fixed size. Because this space is limited, you must be aware of your limitations. When you create a variable, its name and value go into what is called a symbol table.

Your Partition (Workspace)

```
ROUTINE 10,000 bytes  
SYMBOL TABLE 8000 bytes
```

When you have a routine in your workspace, you can use the \$STORAGE variable to check to see how much space you have left in bytes (characters).

M Code

In immediate mode, type:

```
WRITE $STORAGE or WRITE $$
```

The variable \$STORAGE tells you how much space you have left in bytes or characters. Thus, it becomes smaller as you add variables to your symbol table.

VA M PROGRAMMING

Introduction

You can use the \$STORAGE variable to check to see how much space you have left by typing the command WRITE \$STORAGE or WRITE \$S.

You should be aware that advanced methods of dynamic allocation of memory for both routines and variables have rendered this special variable of very limited use. It rarely is accurate but can be a useful estimate of remaining space.

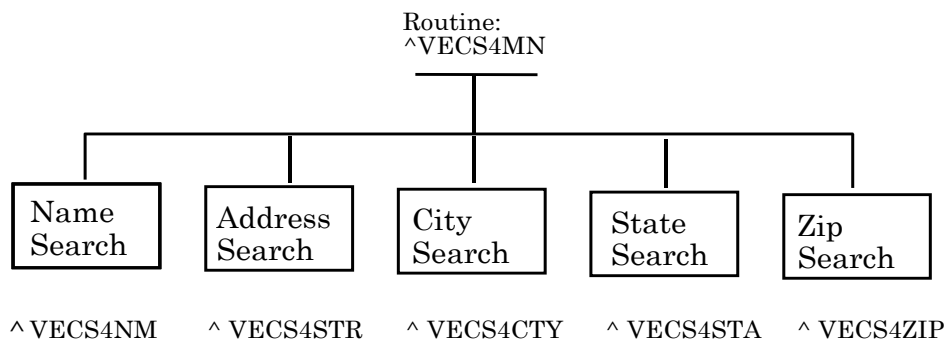


VA Programming Standards and Conventions

The maximum routine size is 10,000 bytes. The combination of a routine and symbol table must run in a partition using standards set forth in the Cookbook.

Modularity

We are going to design a rudimentary inquiry system that will allow us to search for data by name (first letter of last name or entire last name), street address, city, state, or zip. We can conceive of this system as being comprised of modules, where each function is going to be kept in a separate routine. By doing this, we can fit the entire system within the limits we just described.



The main routine, ^VECS4MN, will branch to the appropriate routine based upon the type of search we select.



VA M PROGRAMMING

Introduction

Modularity Using the DO Command

We will use the DO command to link these modules together. There are three ways we can enter a routine using the DO command as shown below.

<i>Sample M Code</i>	Explanation
DO START	Will look for line label START in the current routine
DO ^VECS4MN	Because there is a ^ in front of the name, M will go to the disk and look for the routine named VECS4MN; execution will begin with the first line of that routine
DO LOAD^VECS4MS	Will go to the disk and look for the routine named VECS4MS (^ in front of VECS4MS); execution will begin with the line label LOAD

Notice the importance of the up-arrow in front of the name. It shows that what follows is the name of a routine. Without the up-arrow, it is a label in a routine as we saw in the first example.

The next page shows the complete search routine. Don't worry if you don't recognize all of the commands.

Notice the DO DISMENU following the START label. That label appears later in the routine and obtains the type of search the user wants to perform. It stores the selection in the variable OPT.

After making sure that the user has made a choice, the lines following the START label show a series of commands using the DO command. Each has a post-conditional so it is executed only if OPT has the specified value.

VA M PROGRAMMING

Introduction

Sample M Code

```
VECS4MN      ;ATL/ACE PROGRAMMER-ADDRESS SYSTEM ;3/1/90
      ;;1
      ;      VARIABLE LIST
      ;
      ;      DTIME.....Delay time for READ (not to be KILled)
      ;      OPT.....option number entered by the user
      ;      ITEMS.....number for each menu option
      ;      NAMES.....array of names
INIT ;
      DO LOAD^VECS4MS
START      ;
      DO DISMENU
      IF OPT=""!(OPT["^") GOTO EXIT
      DO ^VECS4NM:OPT=1
      DO ^VECS4STR:OPT=2
      DO ^VECS4CTY:OPT=3
      DO ^VECS4STA:OPT=4
      DO ^VECS4ZIP:OPT=5
      GOTO START
DISMENU      ;
      WRITE !!,?20,"Let's try searching..."
      WRITE !!,"Search by..."
      FOR ITEMS=1:1:5 WRITE !,ITEMS,?5,$P($T(OPTIONS+ITEMS),";",2)
DISMENU1      ;
      READ !!,"Enter option number: ",OPT:DTIME QUIT:OPT=""
      IF '$T!(OPT["^") QUIT
      IF (OPT<1)!(OPT>5) WRITE !!,"Option number must be "
      IF WRITE "between 1 and 5" GOTO DISMENU1
      QUIT
EXIT ;
      KILL OPT,ITEMS,NAMES
      QUIT
OPTIONS      ;
      ;;Name
      ;;Street name
      ;;City
      ;;State
      ;;Zip
```


VA M PROGRAMMING

Introduction

Making Code More Compact

You can abbreviate:

- Commands to one letter (such as D for DO).
- Functions to one letter (but they must start with the \$ and there are a few exceptions where a function is abbreviated to two letters: \$FN for \$FNUMBER).
- Special variables to one letter (but they must start with the \$).

Other ways you can shorten code is to:

- Make variable names one letter. (However, it may be better for variables to mean something – like OPT for option as in the previous sample.)
- Leave out comments. (However, modern systems compile the code and comments really do not take up space in your partition. Therefore, you should not use this as an excuse for a lack of comments.)
- Put multiple commands on a line. There is a maximum of 255 characters in ANS Standard. Almost all programs are written this way and make the program listing much more compact.)
- Use postconditionals, \$SELECT, Boolean comparisons (and, or) to shorten conditional statements. (For example, IF (not) T is the same as IF T equals zero.)

Use some caution. Abbreviating commands and functions is appropriate but making variable names one letter and leaving out comments may not be. You need to strike a balance between efficiency and maintainability.

Regarding multiple commands on one line, some printers may not print more than 80 characters so that "wrap around line" (i.e., those that are greater than 80 characters) may not print.



VA Programming Standards and Conventions

No more than 245 characters per line.

VA M PROGRAMMING

Introduction

Routine efficiency

The efficiency of a routine refers to its processing speed and utilization of resources.

One way to increase efficiency is to make sure you KILL any variables you don't need. Variables take space in the symbol table.



VA Programming Standards and Conventions

All variables used by an option must be KILLed before exiting the option.

The VA has a programming convention that requires that all variables created by an option must be killed before exiting the option. Actually, the requirement is that those variables created by the option must be killed – system variables and certain VA program-wide variables, which are used by the program but not created by the program, such as the DTIME time-out variable must not be killed.

Routine Maintenance

Things that make maintaining a routine or system of routines a nightmare:

- No documentation in the routine or on paper, especially if lots of shortcuts have been taken.



VA Programming Standards and Conventions

All packages (systems of routines) should be considered as a whole, including documentation.

This means that the programs should be documented either by comments in the code or in the documentation.

VA M PROGRAMMING

Introduction

Input Validation

Single Character READ

The purpose of the single character READ is to accept a single keystroke from the user. What is stored in the variable name is the ASCII decimal code of the key.

M Code Sample

```
R !,"Press any key to continue...",&ANS
```

M Code Sample

```
R !,"Enter Y or N",&RESP  
I RESP=89 GOTO AGAIN
```

You know that these are single-character reads because of the asterisk before the variable that is being read. No <ENTER> is required. Hit a single key and the program will continue.

The good news is that this is a way that control and non-printing characters can be read – even <ENTER>. The bad news: you have to figure out the decimal equivalent. (Hint: there is a function that will help you – \$A of Y returns 89.)

Notice that the check of the input recognizes that it is a decimal number. The test is whether RESP equals 89 and not whether it is a Y.



VA Programming Standards and Conventions

You should not use the single character READ to accept users' input. All user input shall be terminated by a carriage return character (<ENTER>).

VA M PROGRAMMING

Introduction

READ with a READ Count

The purpose of the READ with a READ count is to limit the number of characters that the user can enter. If the user enters the maximum, the READ is terminated and processing continues. If the user does not enter the maximum, the user must press <ENTER> to go on.

M Code Sample

```
R !, "Enter name: ", NAME#25
```



VA Programming Standards and Conventions

You should not use the READ with a READ count to accept users' input. All user input shall be terminated by a carriage return character (<ENTER>). However, a #READ and *READ can be used with interfaces, etc., which do not require user input.

There are some exceptions to the prohibition to using the single-character READ and READ with a count. They can be used with non-human inputs such as tape drives or instruments. Also, under certain circumstances a waiver can be obtained such as when the computer is used to administer a psychological test to people who are usually not regular computer users and in which the task is to answer many questions with a T or F for example.



Pattern Match

The purpose of a pattern match is to allow the programmer to create a pattern by which the user's input is checked for validity. The most general description of how to use a pattern match is shown below (the ? is called the pattern match operator):

IF <variable>?<pattern>

If the value of the IF statement is true, that means the data in the variable matches the pattern



VA M PROGRAMMING

Introduction

Pattern Match

A <pattern> is comprised of:

<nr. of occurrences><code><nr. of occurrences><code>...etc

Number of Occurrences

<nr. of occurrences>

Format	Example	Description
n	1	fixed occurrence of the code
n1.n2	1.5	fixed range of occurrences (e.g., from 1 to 5)
.	.	any number of occurrences, including 0
.n	.5	any number of occurrences, from 0 up to a fixed number (e.g., from 0-5)

A pattern is specified by indicating the number of occurrences of a code and there can be any number of the couplets following each other.

This graphic above indicates how the number of occurrences is indicated. There may be a fixed number of occurrences of the code in which case the number of occurrences is used.

For a range of occurrences, the lower value followed by a period followed by the higher value represents the number of occurrences. Since this is the number of times something occurs there should not be any negative numbers.

In the special case where the lower limit is none, a period followed by the upper limit can be used.

VA M PROGRAMMING

Introduction

Pattern Match Codes

<code>

Code	Description
E	Everything (any character)
U	uppercase alphas A-Z
L	lowercase alphas a-z
N	numerics 0-9
A	any alphas, A-Z, a-z
P	punctuation
C	control characters

You can also use strings in place of codes when you are looking for an exact match to a string. For example, the code could be the word YES. This check will be case-sensitive so you should be careful when doing this to consider uppercase and lowercase possibilities.

You can put two or more codes together to indicate that the proper pattern is <code1> OR <code2> (e.g., AN means Alphas *OR* Numerics are okay)

Pattern Match

M Code Sample

	<i>Output</i>
SET SSN="000/45/6789"	
WRITE SSN?3N1P2N1P4N	1
WRITE SSN?3N1"/"2N1"/"4N	1
WRITE SSN?3N1"-"2N1"-"4N	0
TEST ;	
I SSN'?3N1"-"2N1"-"4N W !!,"Wrong Pattern",!	
E W !!,"Right Pattern",!	

The code above shows some truth values resulting from a pattern match.

The first match merely checks whether the SSN is three digits (the 3N), followed by a punctuation character (the 1P), followed by two digits (the 2N), followed by a punctuation character (the 1P), followed by four digits (the 4N). The example matches and hence the value returned by the pattern match is 1 for true. But this match would work for any punctuation character. 000,45,6789 would also match.

VA M PROGRAMMING

Introduction

The second pattern match shown replaces the 1P with a 1"/". This also returns a 1 since the example does use slashes.

The third example checks for the more common way of displaying social security numbers, using a hyphen. Here the example with slashes fails and a 0 is returned.

You might use the WRITE command to test a pattern match but in an actual program you would probably use the IF command as shown in the sample code following the TEST label. These two lines of code would test and indicate whether a SSN is in the proper format.

Pattern Match

M Code Sample

In immediate mode:

```
S SSN=000  
D TEST
```

Wrong Pattern

```
S SSN="000-45-6789"  
D TEST
```

Right Pattern

```
S SSN="000/45/6789"  
D TEST
```

Wrong Pattern

In the sample above, the variable SSN is set to some values and the previous TEST program executed.

When SSN is set to 000, the program indicates that it is a wrong pattern for an SSN. When SSN is set correctly, the program indicates that it is the right pattern.

The final example is a correct SSN – nine digits – but in a less commonly used format. Since the program is looking for hyphens and not slashes, the pattern fails.

VA M PROGRAMMING

Introduction

Pattern match

M Code Sample

<i>Sample M Code</i>	<i>Output</i>
SET NAME="LAST, FIRST"	
WRITE NAME?.AlP.A	1
SET NAME="last, first"	
WRITE NAME?.AlP.A	1
SET NAME="Last,First"	
WRITE NAME?1U.A1", "1U.A	1

Here are some examples using a name. Notice the A, which means as few as 0 to as many as infinite alphabetic characters. The check is for some number of alphabetic characters followed by a punctuation character, followed by some number of alphabetic characters. Both the upper case and lower-case names match this pattern.

But some other strange things would also match the pattern. Would just jones comma work? Yes. Would just a period or a comma, or just a hyphen work? Yes. There need not be any alphabetic characters before or after the punctuation character. There *may* be – but there doesn't *have* to be.

The final example is a little more realistic. It requires a last name beginning with an uppercase letter followed by an unknown number of additional alphabetic characters followed by a comma followed by a first name which begins with an uppercase case letter and again followed by some number of alphabetic characters. Since the A code is being used, this will work for both upper and lower-case names, as here, or all upper-case names.

Pattern Match

M Code Sample

	<i>Output</i>
W 999?1.2N	0
W "M"?5AP	0
W "M"?1.3AP	1
W "M"? .U	1
W "123-444"?1.3N1" - "3N	1

VA M PROGRAMMING

Introduction

Here are some other kinds of tests.

Is the number 999 one or two digits? No – it has three. Notice the 1.2 and recall that means one to two occurrences of the code.

Is the letter M five characters long with the characters being either alphabetic or punctuation? No

Is the letter M one to three characters long and consisting of alpha or punctuation characters? Yes.

Is the letter M some unknown number of uppercase letters? Yes, it's actually one character.

Finally, does the string 123-444 match the pattern shown? Yes. Notice that the hyphen and the final three digits are required. The initial three digits could have been one, two, or three digits long and matched the pattern.

Control of Timing with HANG

The purpose of HANG is to pause the execution of the program for a specified number of seconds. This can be used to give the user enough time to read the display. There also may be cases where one simply wants to pause the program to make sure that some other event occurs.

M Code Sample

```
DO DISPLAY  
HANG 10  
.  
.  
.
```



Intrinsic Functions

Functions from Previous Lessons

\$PIECE \$TEXT \$SELECT \$JUSTIFY \$FNUMBER

In previous lessons, we looked at the M functions above. We shall now study some others.

\$FIND() Function

The purpose of the \$FIND function is to look for one string inside another string and return its location. If it finds the string it's looking for, the result is a number that is the character position immediately following the string it was looking for.

If that sounds like a legal document might read, you're right. While the output of this function may seem strange, it is one way of answering the question.

The function looks for the second string in the first string beginning at the character position indicated by the third optional parameter. If the third parameter is missing, the search begins at the beginning.

Sample M Code

Output of \$FIND

SET TALE="ONCE UPON A TIME"	
WRITE \$FIND(TALE,"N")	3
WRITE \$FIND(TALE,"O")	2
WRITE \$FIND(TALE,"X")	0
WRITE \$FIND(TALE,"O",5)	9

Does the string `Once Upon A Time` contain an N? Yes, it does. Actually, there are two. But the first is in the second character position so the position following the N, as required by the definition, is three. Is there an O? Yes, there is. The first character. So, the returned value is one more or a 2. Notice that the function only identifies the first occurrence and ignores additional occurrences. Does the string have an X? No, it does not. So, the value is 0.

VA M PROGRAMMING

Introduction

Finally, assume we know that the string begins with the word `Once` and want to know if there is a letter `O` after the initial word. The last example begins the search at the fifth character position, the space following `Once`, and finds an `O` in the word `upon`. It is the eighth character of the string and hence a 9 is returned.

Would you like some extra credit? What if the second parameter was `PO` instead of a single character?

`PO` is in `Upon` and is the seventh and eighth characters. So a 9 is returned.



\$EXTRACT() Function

The purpose of the `$EXTRACT` function is to copy a character or characters out of a string.

Sample M Code

```
SET TALE="ONCE UPON A TIME"  
WRITE $EXTRACT(TALE)  
WRITE $EXTRACT(TALE,4)  
WRITE $EXTRACT(TALE,6,9)
```

Output of \$EXTRACT

```
O  
E  
UPON
```

The first parameter is required and is the string of interest.

The second parameter, if it exists, is the beginning position of the string that we want to make a copy of.

In the first example above, there is no second parameter so the first character of the string, the letter `O`, is returned – not a zero – the letter `O` from `Once`.

The second example asks for the fourth character. That's the last character of the word `Once` or an `E`.

The last example has a third parameter, which, as you might guess, is the ending position since the second parameter is the beginning position. What is the value of the sixth through the ninth characters of the string?

It's the second word, `Upon`.

Want more extra credit? What would `$E(TALE,0)` be?

How about `$E(Tale,99)`?

The result of both would be null. Neither exist!

VA M PROGRAMMING

Introduction

M Code Sample

```
SET SSN="000-62-7777"  
SET INDATA1=$E(SSN,1,3)_$E(SSN,5,6)_$E(SSN,8,11)  
  
WRITE INDATA1           Output: 00062777  
  
SET EXTDATA1=$E(INDATA1,1,3)_"_"$E(INDATA1,4,5)_"_"$E(INDATA1,6,9)  
  
WRITE EXTDATA1         Output: 000-62-7777  
  
SET EXTDATA2=$E(INDATA1,6,9)  
  
WRITE EXTDATA2         Output: 7777
```

The code above illustrates some ways that \$E can be used to add or remove characters from a string or to retrieve a sub-string of interest. The examples are of a nine-digit social security number. Since we know where the hyphens are (or should be) we can remove them if need be since that would save two characters in every value we stored. INDATA1 uses \$E to extract just the digits – notice that the fourth and seventh characters, which are the hyphens, are omitted.

In EXTDATA1 we take this value without hyphens and put them back in as we would probably do when displaying the value. Here there are no gaps in the character positions we are extracting since we want all the digits. We are just breaking up the string in order to put the hyphens in the right places.

If we had need of just the last four digits of the SSN, the final example shows how to extract them. The last four digits are actually the sixth through the ninth digits of the string without hyphens as shown in EXTDATA2. The values would have been 8 through 11 for the string with hyphens.



VA M PROGRAMMING

Introduction

\$LENGTH() Function

The purpose of the \$LENGTH function is to count the number of characters in a string or to count the number of pieces given a specified delimiter.

Sample M Code

Output of \$LENGTH

```
SET NAME="ZYXWVUTS,DCBA"
WRITE $LENGTH(NAME)                13

SET NAME="DCBA J. ZYXWVUTS"
WRITE $LENGTH(NAME," ")            3

WRITE $PIECE(NAME," ", $LENGTH(NAME," "))  ZYXWVUTS
```

Sample M Code

Output of \$LENGTH

```
SET SSN="000-62-7777"
WRITE $LENGTH(SSN)                11

SET SSN="000-62-7777"
WRITE $LENGTH(SSN,"-")            3

WRITE $PIECE(SSN,"-", $LENGTH(SSN,"-"))  7777
or
WRITE $PIECE(SSN,"-", 3)
```

If \$LENGTH has only one parameter, the value returned is the number of characters in the string. In the first example, the name has 13 characters.

If there is a second parameter, then \$L returns the number of pieces in the string using the second parameter as a delimiter. The second code sample asks for the number of pieces in a name using a space as a delimiter and properly returns the answer 3.

The final example in these name manipulations shows how two functions can be combined to get the last name from the string. The \$L tells us there are three pieces, the last being the last name, and the \$P function gets that piece. Why make things so complex? This would work for people with no middle initial, two middle names, or just a last name. No matter how many pieces, it gets the last one.

The remaining examples involve a social security number. The one argument \$L tells us there are 11 characters in the SSN and the two argument \$L tells us there are three pieces using a hyphen as a delimiter.

VA M PROGRAMMING

Introduction

The last two examples show two ways of getting the last four digits of the number. The example that is analogous to the name example, except using a hyphen rather than a space as a delimiter, will certainly work. But then, for lazy types, so will the last example taking the third piece. Here we are coding based upon our knowledge that a social security number always has three pieces delimited by hyphens. Names don't always fit such nice patterns.



\$ASCII() Function

The purpose of the \$ASCII function is to convert a character to its ASCII decimal equivalent.

<i>Sample M Code</i>	<i>Output of \$ASCII</i>
WRITE \$ASCII("B")	66
SET TALE="ONCE UPON A TIME"	
WRITE \$ASCII(TALE)	79
WRITE \$ASCII(TALE,2)	78

The one-argument version returns the value for the first character of the string. The two-argument version returns the value for the character in the position indicated by the second argument.

The decimal equivalent of B is 66. For the first letter of Once, the letter O, it is 79. For the second letter, the N, it is 78.

\$CHAR() Function

The purpose of the \$CHAR function is to convert its ASCII decimal value to its equivalent character.

<i>Sample M Code</i>	<i>Output of \$CHAR</i>
WRITE \$CHAR(66)	B
WRITE \$CHAR(77,85,77,80,83)	MUMPS
WRITE \$CHAR(34),"HELLO",\$CHAR(34)	"HELLO"

WRITE \$C(66) is the same as WRITE *66

W \$C(7) is the same as W *7.

VA M PROGRAMMING

Introduction

The first example on the previous page shows that ASCII 66 is a capital B.

If there are multiple arguments, as in the second example, a string is returned composed of the characters corresponding to the numbers. MUMPS in this case.

The third example is interesting. If you write "HELLO" you get just hello since the quote marks are required to delimit the string. But what if you wanted to actually write the quotes also? This is one way to do it.

Not very realistic since a quote is printable and could be directly printed. On the other hand, it is clearer than writing four quotes in a row, which is how you would have to do it to output a single quote.

WRITE \$C(66) is the same as WRITE *66. In both cases, the program sends the character corresponding to decimal 66 to the output device. The preferred method is to use WRITE \$C(66)

Most commonly, you will find a WRITE *7, which rings a bell, in programs. WRITE\$C(7) is the preferred method to use.



Sample Routine Using \$ASCII and \$CHAR

The next page shows a routine that will produce a chart that is almost impossible to find when you need it --- an ASCII to decimal conversion.

It begins with the space character and goes to the end of the character sequence. It prints 95 values. Why 95? Twenty-six upper case letters, 26 lower case letters, 10 digits, and some both common and strange punctuation characters.

The first SET command sets START equal to the decimal value of a space. The next line is a loop that prints 95 values. In CHART, you see that it uses the value of START the first time, and then increments it by one for the following 94 characters.

VA M PROGRAMMING

Introduction

```
VECS422 ;ATL/ACE PROGRAMMER-ASCII CHART ;3/1/90
; ;1
; This routine will print an ASCII chart
;
; VARIABLE LIST
;
; X.....loop counter
; START...starting character (space)
;
SET START=$A(" ")
F X=1:1:95 DO CHART
EXIT ;
KILL X,START
QUIT
CHART ;
WRITE !,$CHAR(START),?5,START
S START=START+1
QUIT
```

Output from the Sample Routine Using \$ASCII and \$CHAR

Here is the partial output from the program above. It begins with a space having a value of 32 and shows some of the other values. The complete output would have 95 lines and go from 32 to 127.

Partial Output:

	32
!	33
"	34
#	35
\$	36
%	37
&	38
.	
.	
0	48
1	49
.	
.	
A	65
B	66
.	
.	

VA M PROGRAMMING

Introduction

\$RANDOM() Function

The purpose of \$RANDOM is to generate random numbers. In parentheses you specify the range of numbers (n) and M will generate numbers from 0 through n-1.

<i>Sample M Code</i>	<i>Range of numbers generated</i>
WRITE \$RANDOM(50)	0 - 49
WRITE \$RANDOM(100)	0 - 99
WRITE \$RANDOM(101)	0 - 100
WRITE \$RANDOM(100)+1	1 - 100
WRITE \$RANDOM(99)+1	1 - 99

Since one often want numbers in the range from 1 to something, the last two examples show how 1 can be added to the result and thus produce a final value, which is from 1 to the argument value.

Each use of \$R produces a single random number.



On the next page, there is a program using \$R in a loop to produce 15 occurrences of a phrase randomly selected from a group of five. Since we don't want a zero value, and have five phrases, we use \$R(5)+1 to get our record number and then use \$TEXT to retrieve the phrase.

This is often used in a Tip of the Day or Daily Thought kind of program. It is also very useful in statistical or sampling programs.

VA M PROGRAMMING

Introduction

Sample Routine Using \$RANDOM

```
VECS424      ;ATL/ACE PROGRAMMER-RANDOM MESSAGES ;3/1/90
              ;;1
              ;    VARIABLE LIST
              ;
              ;    RECNR.....record number of current record
              ;    X.....loop counter
              ;
START         ;
              F X=1:1:15 S RECNR=$RANDOM(5)+1 D MESSAGE
EXIT          ;
              K X,RECNR
              Q
MESSAGE       ;
              W !,"LOOP NUMBER ",X
              W ?20,$P($T(DATA+RECNR),";;",2)
              Q
DATA          ;
              ;;HOLD ON...I'M THINKING
              ;;BE BACK IN A LITTLE WHILE
              ;;BEAM ME UP SCOTTY!
              ;;TOOK A COFFEE BREAK
              ;;SOME THINGS TAKE LONGER THAN OTHERS
```

Partial Output:

```
LOOP NUMBER 1      BE BACK IN A LITTLE WHILE
LOOP NUMBER 2      TOOK A COFFEE BREAK
LOOP NUMBER 3      BE BACK IN A LITTLE WHILE
LOOP NUMBER 4      BEAM ME UP SCOTTY!
LOOP NUMBER 5      TOOK A COFFEE BREAK
LOOP NUMBER 6      HOLD ON...I'M THINKING
LOOP NUMBER 7      TOOK A COFFEE BREAK
.
.
.
LOOP NUMBER 15     BEAM ME UP SCOTTY!
```

VA M PROGRAMMING

Introduction

Arrays

So far, we have worked with variables that have one value at a time. Arrays are named locations that can hold multiple values at one time.

A reference to an array is comprised of:

`<array name>(<subscript>)`

`<array names>` are assigned just like variable names

`<subscripts>` can be numeric or string values

The array name is just like a variable name and follows the same naming rules.

The example below shows array references where NAMES is the name of the array and the numbers 1, 2, and 3 are the subscripts.

Once the subscript is specified, the array can take on values just as a simple variable would. Here, the array NAMES is being set to names and addresses for seven people. The entire name address string is the value of the variable for each of the subscripted elements in the array.

Sample Array References

`<array name>(<subscript>)`

NAMES (1)

NAMES (2)

NAMES (3)

Arrays are assigned values, just as simple variables are, by using the SET or READ commands.

VA M PROGRAMMING

Introduction

M Code Sample

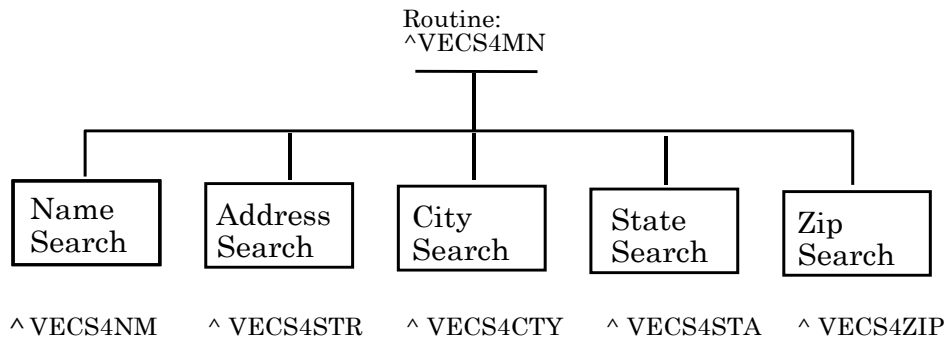
```
SET NAMES(1)="Mommm,Marty^123 Any St.^MyCity,GA 00002"
SET NAMES(2)="Smsss,Bill^530 My St.^AnyCity,CA 00005"
.
.
SET NAMES(7)="Mimmm,Ned^4 My Ave.^MyTown,PA 00002"
SET NAMES(1)="Mommm,Marty^123 Any St.^MyCity,GA 00002"
SET NAMES(2)="Smsss,Bill^530 My St.^AnyCity,CA 00005"
.
.
SET NAMES(7)="Mimmm,Ned^4 My Ave.^MyTown,PA 00002"

WRITE !,$P(NAMES(1),"^",1)      Mommm,Marty
WRITE !,$P(NAMES(2),"^",2)      530 My St.
```

The array reference can be used just as a simple variable is used. Here, we write the values from the first piece of NAMES (1) and the second piece of NAMES (2).

Did you get caught thinking that 530 Maple is the address of Marty Mommm? The address is that of the second person.

Sample Routine and Output Using Arrays and Intrinsic Functions



Remember this graphic? It was one of the first used in this section. It describes a program, which can retrieve a person or persons using a variety of search criteria. The ^VEC names are the names of the routines that perform the indicated function. The names don't necessarily have any meaning. But it's helpful when they do, as they do here, where the final characters indicate what is retrieved: NM for name, STR for street, CTY for city and so forth.

The next few pages show examples of how some interactions and output from this program might work. What the user typed is underlined.

VA M PROGRAMMING

Introduction

We executed ^VECS4MN, which, as you may recall, asks you what kind of search you want to perform. So we can assume that the search by display is from that program as is the READ command that obtains the option number.

>D ^VECS4MN

Hold on...I'm thinking

Let's try searching...

Search by...

- 1 Name
- 2 Street name
- 3 City
- 4 State
- 5 Zip

Enter option number : 1

Enter first letter or last name: M

Mommm,Marty 123 Any St.
MyCity,GA 00002

Mimmm,Joe 4 My Ave.
MyTown,PA 00002

Mommm,Ralph 5 Any St.
MyCity,GA 00002

Mimmm,Maggie 4 My Ave.
MyTown,PA 00002

Mimmm,Ned 4 My Ave.
MyTown,PA 00002

Press Enter to return to the menu

In the example above, where do you think the READ command for the first letter or last name and the following output are coming from?

^VECS4NM– the name of the search routine.

VA M PROGRAMMING

Introduction

The example below is the same program but using a complete last name. The output is more manageable.

Let's try searching...

Search by...

- 1 Name
- 2 Street name
- 3 City
- 4 State
- 5 Zip

Enter option number : 1

Enter first letter or last name: Mommm

Mommm,Marty 123 Any St.
 MyCity,GA 00002

Mommm,Ralph 5 Any St.
 MyCity,GA 00002

Press Enter to return to the menu

In the example below, the search is by street name. Presumably, the initial routine is now calling ^VECS4STR to get the street name and do the search.

Note that even though the search is by street addresses containing the word ANY the output is formatted the same as before and reads like a normal name and address output.

Let's try searching...

Search by...

- 1 Name
- 2 Street name
- 3 City
- 4 State
- 5 Zip

Enter option number : 2

Enter street name: Any

Mommm,Marty 123 Any St.
 MyCity,GA 00002

Mommm,Ralph 5 Any St.
 MyCity,GA 00002

Press Enter to return to the menu

VA M PROGRAMMING

Introduction

Another example, here using the city name. By now you should be able to figure out the routine that is being used to do the search. Again, the output is in normal name-address style.

Let's try searching...

Search by...

- 1 Name
- 2 Street name
- 3 City
- 4 State
- 5 Zip

Enter option number : 3

Enter city: MyTown

Mimmm,Joe	4 My Ave.
	MyTown,PA 00002

Mimmm,Maggie	4 My Ave.
	MyTown,PA 00002

Mimmm,Ned	4 My Ave.
	MyTown,PA 00002

Press Enter to return to the menu

VA M PROGRAMMING

Introduction

Shown below is the fourth option, searching by state.

Let's try searching...

Search by...

- 1 Name
- 2 Street name
- 3 City
- 4 State
- 5 Zip

Enter option number : 4

Enter state: PA

Mimmm,Joe	4 My Ave.
	MyTown,PA 00002

Mimmm,Maggie	4 My Ave.
	MyTown,PA 00002

Mimmm,Ned	4 My Ave.
	MyTown,PA 00002

Press Enter to return to the menu

Finally, here's option number 5. This search is by zip code.

Search by...

- 1 Name
- 2 Street name
- 3 City
- 4 State
- 5 Zip

Enter option number : 5

Enter zip code: 93215

Smsss,Bill	530 My St.
	AnyCity,CA 00005

Smsss,Mary	530 My St.
	AnyCity,CA 00005

Press Enter to return to the menu

VA M PROGRAMMING

Introduction

So. How do we end this and stop our searches? Just hit <ENTER> for the option.

You may recall from one of the earlier sections that there was a check to see if the value entered was null – hitting <ENTER> – or an up-arrow.

Search by...

- 1 Name
- 2 Street name
- 3 City
- 4 State
- 5 Zip

Enter option number : <ENTER>

>

VA M PROGRAMMING

Introduction

Sample Systems

The following pages contain several M routines for you to study. Please review these routines taking note of the routine structure.

```
VECS4MN      ;ATL/ACE PROGRAMMER-ADDRESS SYSTEM ;3/1/90
      ;;1
      ;      VARIABLE LIST
      ;
      ;      DTIME.....Delay time for READ (not to be KILLed)
      ;      OPT.....option number entered by the user
      ;      ITEMS.....number for each menu option
      ;
INIT ;
      D LOAD^VECS4MS
START      ;
      D DISMENU
      I OPT=""(OPT["^") GOTO EXIT
      D ^VECS4NM:OPT=1
      D ^VECS4STR:OPT=2
      D ^VECS4CTY:OPT=3
      D ^VECS4STA:OPT=4
      D ^VECS4ZIP:OPT=5
      G START
DISMENU      ;
      W !!,?20,"Let's try searching..."
      W !!,"Search by..."
      F ITEMS=1:1:5 W !,ITEMS,?5,$P($T(OPTIONS+ITEMS),";;",2)
DISMENU1      ;
      R !!,"Enter option number: ",OPT:DTIME Q:OPT=""
      I '$T!(OPT["^") QUIT
      I (OPT<1)!(OPT>5) W !!,"Option number must be "
      I W "between 1 and 5" G DISMENU1
      Q
EXIT ;
      K OPT,ITEMS,NAMES
      Q
OPTIONS      ;
      ;;Name
      ;;Street name
      ;;City
      ;;State
      ;;Zip
```

VA M PROGRAMMING

Introduction

```
VECS4MS      ;ATL/ACE PROGRAMMER-ADDRESS SYSTEM ;3/1/90
; ;1
;      Miscellaneous routine
;
;      VARIABLE LIST
;
;      RECORD.....Current record
;      RECNR.....Record number of current record
;      NAMES.....The array that holds the name and
;                  address data
;      NAMES(0).....When the array has been loaded, this
;                  variable contains the number of
;                  records loaded into the array
;
LOAD ;
; We put the next two lines in to simulate what you
; might want to do if your routine will be engaging
; in a process that may delay the rest of the processing.
; Since this subroutine doesn't actually take any
; extra time, we've added the HANG.
START ;
W ?20,"...", $P($T(MSG+($R(5)+1)), ";", 2)
H 10
F RECNR=1:1 D INPUT Q:RECORD=""
S NAMES(0)=RECNR-1
K RECORD, RECNR
Q
INPUT ;
S RECORD=$P($T(DATA+RECNR), ";", 2)
Q:RECORD=""
S NAMES(RECNR)=RECORD
Q
DATA ;
; ;Mommm, Marty^123 Any St.^MyCity, GA 00002
; ;Smsss, Bill^530 My St.^AnyCity, CA 00005
; ;Smsss, Mary^530 My St.^AnyCity, CA 00005
; ;Mimmm, Joe^4 My Ave.^MyTown, PA 00002
; ;Mommm, Ralph^5 Any St.^MyCity, GA 00002
; ;Mimmm, Maggie^4 My Ave.^MyTown, PA 00002
; ;Mimmm, Ned^4 My Ave.^MyTown, PA 00002
MSG ;
; ;Hold on...I'm thinking
; ;Beam me up Scotty!
; ;Be back in a little while
; ;Took a coffee break
; ;Some things take longer than others
;
;
WRREC ; Display output from any search
W !, $P(RECORD, "^", 1), ?30, $P(RECORD, "^", 2)
W !, ?30, $P(RECORD, "^", 3)
Q
```

VA M PROGRAMMING

Introduction

```
VECS4NM      ;ATL/ACE PROGRAMMER-ADDRESS SYSTEM ;3/1/90
; ;1
;   VARIABLE LIST
;
;   SNAME.....Name user wants to search for
;   NAME.....Full name of the current record
;   LNAME.....Last name of current record
;   FNAME.....First name of current record
;   RECNR.....record number of current record
;   RECORD.....current record
;   ANS.....user's response to "Press Enter to return
;               to the menu"
;
START        ;
R !!,"Enter first letter or last name: ",SNAME:DTIME
I '$T!(SNAME["^") G EXIT
I SNAME'?1U.A W !,"Enter a letter or a last name" G START
F RECNR=1:1:NAMES(0) D SEARCH
G EXIT
SEARCH      ;
S RECORD=NAMES(RECNR)
S NAME=$P(RECORD,"^",1)
S LNAME=$P(NAME,"",1)
S FNAME=$P(NAME,"",2)
I $L(SNAME)=1 D LETTER
I $L(SNAME)'=1 D WORD
Q
LETTER      ;
I $E(LNAME)=SNAME DO WRREC^VECS4MS
Q
WORD        ;
I SNAME=LNAME DO WRREC^VECS4MS
Q
EXIT        ;
R !!,"Press Enter to return to the menu",ANS:DTIME
KILL SNAME,LNAME,FNAME,RECNR,ANS,NAME,RECORD
Q
```

VA M PROGRAMMING

Introduction

```
VECS4STR ;ATL/ACE PROGRAMMER-ADDRESS SYSTEM ;3/1/90
; ;1
; VARIABLE LIST
;
; STRNAME.....Street name user wants to search for
; RECORD.....Current record
; RECNR.....Record number of current record
; ADDR.....Full address
; ANS.....user's response to "Press Enter to return
;          to the menu"
;
START ;
R !!,"Enter street name: ",STRNAME:DTIME
I '$T!(STRNAME["^") G EXIT
F RECNR=1:1:NAMES(0) D SEARCH
G EXIT
SEARCH ;
S RECORD=NAMES(RECNR)
S ADDR=$P(RECORD,"^",2)
I ADDR[STRNAME D WRREC^VECS4MS
Q
EXIT ;
R !!,"Press Enter to go to the menu",ANS:DTIME
K ADDR,STRNAME,RECNR,RECORD,ANS
Q
```

VA M PROGRAMMING

Introduction

```
VECS4CTY ;ATL/ACE PROGRAMMER-ADDRESS SYSTEM ;3/1/90
; ;1
; VARIABLE LIST
;
; CTYNAME.....City user wants to search for
; RECORD.....Current record
; RECNR.....record number of current record
; ADDR.....Full address of current record
; CITY.....City of current record
; ANS.....user's response to "Press Enter to return
; to the menu"
;
START ;
R !!,"Enter city: ",CTYNAME:DTIME
I '$T!(CTYNAME["^") G EXIT
I CTYNAME'?1U.A W !,"City name must start with an uppercase letter" G START
F RECNR=1:1:NAMES(0) D SEARCH
G EXIT
SEARCH ;
S RECORD=NAMES(RECNR)
S ADDR=$P(RECORD,"^",3)
S CITY=$P(ADDR,"",1)
I CITY[CTYNAME D WRREC^VECS4MS
Q
EXIT ;
R !!,"Press Enter to go to the menu",ANS:DTIME
K ADDR,CTYNAME,RECNR,RECORD,ANS
Q
```

VA M PROGRAMMING

Introduction

```
VECS4STA ;ATL/ACE PROGRAMMER-ADDRESS SYSTEM ;3/1/90
; ;1
; VARIABLE LIST
;
; STNAME.....State user wants to search for
; RECORD.....Current record
; RECNR.....Record number of current record
; ADDR.....Full address of current record
; CITYST.....City and state of current record
; STATE.....State of current record
; ANS.....user's response to "Press Enter to return
; to the menu"
START ;
R !!, "Enter state: ", STNAME: DTIME
I '$T!(STNAME["^") G EXIT
I STNAME'?2U W !, "State must be a 2 letter code" G START
F RECNR=1:1:NAMES(0) D SEARCH
G EXIT
SEARCH ;
S RECORD=NAMES(RECNR)
S ADDR=$P(RECORD, "^", 3)
S CITYST=$P(ADDR, " ", 1)
S STATE=$P(CITYST, " ", 2)
I STATE[STNAME D WRREC^VECS4MS
Q
EXIT ;
R !!, "Press Enter to go to the menu", ANS: DTIME
K ADDR, ZIPCODE, RECNR, RECORD, ANS, CITY
Q
```

VA M PROGRAMMING

Introduction

```
VECS4ZIP ;ATL/ACE PROGRAMMER-ADDRESS SYSTEM ;3/1/90
; ;1
; VARIABLE LIST
;
; RECORD.....Current record
; RECNR.....Record number of current record
; ADDR.....Full address of current record
; ZIPCODE.....Zipcode that user wants to search for
; ZIP.....Zipcode of current record
; ANS.....user's response to "Press Enter to return
; to the menu"
;
START ;
R !!,"Enter zip code: ",ZIPCODE:DTIME
I '$T!(ZIPCODE["^") GOTO EXIT
I ZIPCODE'?5N W !,"Zip code must be 5 digits" G START
F RECNR=1:1:NAMES(0) DO SEARCH
G EXIT
SEARCH ;
S RECORD=NAMES(RECNR)
S ADDR=$P(RECORD,"^",3)
S ZIP=$P(ADDR," ",2)
I ZIP[ZIPCODE DO WRREC^VECS4MS
QUIT
EXIT ;
R !!,"Press Enter to go to the menu",ANS:DTIME
K ADDR,ZIPCODE,RECNR,RECORD,ANS
Q
```


VA M PROGRAMMING

Introduction

Optional Assignment

In general:

- Follow the VA Programming Conventions as illustrated in this lesson. Also, remember that the second line of the routine contains the version number; use 1 if this is the first transmission, use 2 if this is a REDO.
- Use only the techniques, commands, and features described in this lesson or the previous lessons.
- Save the routine(s) under a namespace assigned to you by your local facility. Specifically, your routines should be named as follows:

main menu:	VECB4nnA
miscellaneous:	VECB4nnB
book title search:	VECB4nnC
Optional routines:	
author's name search:	VECB4nnD
topic search:	VECB4nnE
book nr. search:	VECB4nnF
where:	
	4 is the lesson number
	nn is your student number

Assignment Specifications

Design a library book search system patterned after the sample system ^VECS4CTY. Make up your own data (at least eight books).

For each book, include the book title, author, topic, and book number. Allow the user to search by book title. If you have time, add routines to search by (optional):

author's name (either first letter of last name, last name, or full name)
topic
book number

You should include your own random waiting messages when your book array is being loaded.

Don't forget to include variable lists in all your routines.

Create a main menu even if you implement only one option now.