IV. Arrays and Subscripts

IBM

Learning System/23 BASIC

IV. Arrays and Subscripts

IBM

Learning System/23 BASIC

First Edition (January 1981)

IV. Arrays and subscripts

Contents

About this book v

Chapter 1. Numeric arrays 1-1
Introduction 1-1
Array names and subscripts 1-2
The OPTION statement 1-3
The DIM statement 1-4
Using arrays in programs 1-7
Using RD in the OPTION statement 1-10
Chapter summary 1-12
Exercises 1-14
Answers 1-16

Chapter 2. Character arrays 2-1
Introduction 2-1
Array names and subscripts 2-2
The OPTION statement 2-3
The DIM statement 2-4
Chapter summary 2-12
Exercises 2-13
Answers 2-15

Chapter 3. Two-dimensional arrays 3-1
Introduction 3-1
Numeric arrays 3-2
Character arrays 3-4
The DIM statement 3-5
Using two-dimensional arrays in programs 3-7
Chapter summary 3-9
Exercises 3-10
Answers 3-12

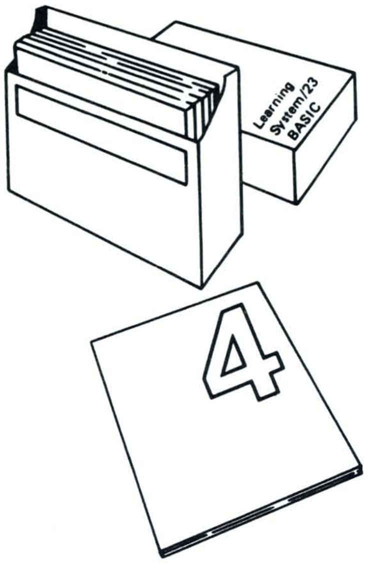
Chapter 4. Matrix operations 4-1
Introduction 4-1
Using PRINT and PRINT #255: 4-2
Using READ/DATA and INPUT 4-4

IV. Arrays and subscripts

Contents (continued)

Using MAT assignments 4-6
Chapter summary 4-10
Exercises 4-11
Answers 4-13

About this book



This is the fourth in your series of seven books on *Learning System/23 BASIC*. In this book, you will learn about *arrays*. An array is a collection of data items, which are all referred to by the same variable name.

We will show you how to specify the maximum number of characters in a character string. We will also show you how to use an array, and how to specify the maximum number of data items in an array.

You will learn how to assign values to the variable items in an array, both individually and to the array as a whole. You will also learn how to print or display the contents of an array.

Chapter 1. Numeric arrays

Introduction

So far in this course, we have discussed numeric and character variables as single items. In this chapter, you will learn how to group several numeric variables together in an *array*.

You will learn how to refer to a numeric array by its array name. You will learn how to specify the number of variables in an array. You will also learn how to use the OPTION statement.

Objectives

Upon completion of this chapter, you should be able to do the following:

- Refer to a collection of numeric variables by using an array.
- Specify the number of numeric variables in an array by using the DIM statement.
- Specify BASE 0 or BASE 1 in a program by using the OPTION statement.
- Specify how your System/23 should round unformatted numbers by using the OPTION statement.

If you are familiar with these tasks, try the exercises located at the end of this chapter. If not, read through the chapter before going on to the exercises.

Numeric arrays

Array names and subscripts

Up until now, when you have assigned numeric variable names in a program, each variable has been referred to separately.

P
I
AGE

These are *simple variables*. Each data item has its own name.

An *array* is a collection of data items. In an array, several data items are referred to by the same name. The entire array is named just like a numeric variable.

The individual variables in an array are called *elements*. Each element is named by a *subscripted variable*. A subscripted variable looks like this:

```
 P(1)
 ↑  ↖
Array Subscript
name
```

The first part of the subscripted variable is the array name. This can be any valid numeric variable name. The second part of the subscripted variable is the *subscript*. The subscript, which must be in parentheses, indicates the position of the element in the array.

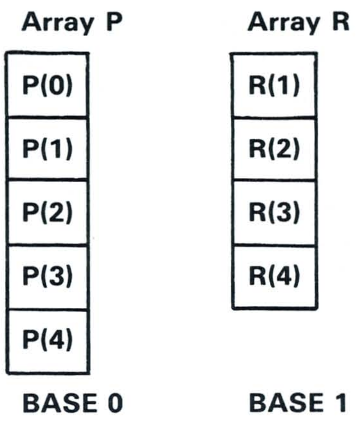
Array P, as shown here, is a *one-dimensional* array. The variables in the P array can be thought of as a list of variables. P(1) can be pronounced P sub 1.

The entire list of variables is an *array*. Each individual variable is an *element*.

The OPTION statement

An array element is referred to by the array name and, in parentheses, its position in the list.

An array begins with either position 0 or 1. For example,



The array on the left begins with P(0) and has five elements. The array on the right begins with R(1) and has four elements.

You use the OPTION statement to indicate which position the arrays in a program start with.

OPTION BASE 0 indicates that the first element of your arrays is in position 0. OPTION BASE 1 indicates that the first element of your arrays is in position 1. With OPTION BASE 0, your array has an additional element (the element 0), and your program may take up more storage space.

We'll use the OPTION statement in a program in just a moment. But first, you need to learn about one more statement.

Numeric arrays

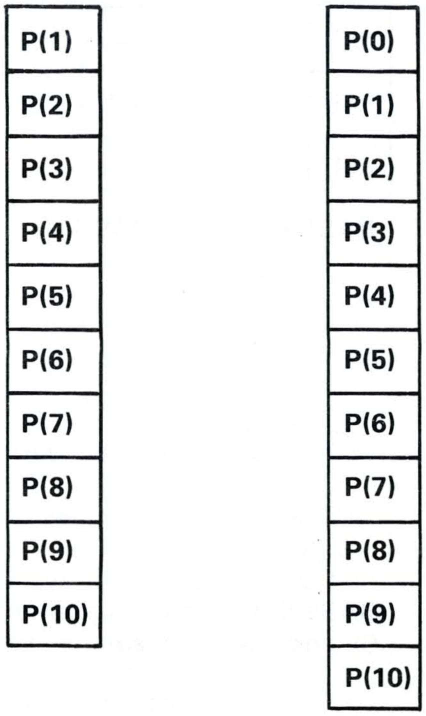
The DIM statement

The DIM statement *dimensions* an array. That is, it specifies the *number* of elements in an array.

```
20 DIM P(10)
```

This statement indicates that the last element in array P is element P(10). The array could have 10 or 11 elements, depending on the OPTION statement.

OPTION BASE 1 **OPTION BASE 0**



Let's look at an example.

```
10 DIM R(10),AREA(18)
```

This statement defines two arrays in the program. The last element in array R is R(10). The last element in array AREA is AREA(18).

You can include as many arrays in a DIM statement as will fit on a line (maximum of 255 characters on a line). You can also include more than one DIM statement in a program. However, any array name can be entered only once and in only one DIM statement. If you use an array name more than once, you will get an error message when you try to run the program.

This is valid: 10 DIM X(10),Y(20)
This is invalid: 10 DIM X(10),Y(20),X(15)

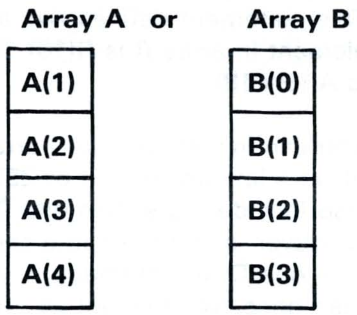
Each element in a numeric array is set to zero when you start running a program. The values may change as program statements are executed.

You can use an array in a program without including it in a DIM statement. If you do, it will be dimensioned to (10). It will have 10 elements with BASE 1 or 11 elements with BASE 0.

If you do not specify the BASE by using an OPTION statement, your System/23 uses BASE 0 for the program.

The DIM statement (continued)

The arrays we have been looking at are *one-dimensional arrays*. (We will discuss two-dimensional arrays in a later chapter.) One-dimensional arrays are like a list.



The subscript for each element is a number that represents the position of the element by row.

A(1)
Row 1
B(0)
Row 0

Remember: With BASE 1, the first position in an array is position 1. With BASE 0, the first position in an array is position 0.

Using arrays in programs

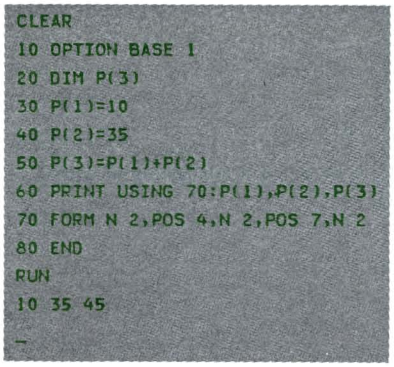
Let's see how you can use an array. Enter the following:

```
CLEAR
10 OPTION BASE 1
20 DIM P(3)
30 P(1)=10
40 P(2)=35
50 P(3)=P(1)+P(2)
60 PRINT USING 70:P(1),P(2),P(3)
70 FORM N 2,POS 4,N 2,POS 7,N 2
80 END
```

The three variables P(1), P(2), and P(3) all have individual values. They can be used just like any other variables in a program. The only difference is that each of these values is a separate element of the P array.

Go ahead and run this program:

```
RUN
```



```
CLEAR
10 OPTION BASE 1
20 DIM P(3)
30 P(1)=10
40 P(2)=35
50 P(3)=P(1)+P(2)
60 PRINT USING 70:P(1),P(2),P(3)
70 FORM N 2,POS 4,N 2,POS 7,N 2
80 END
RUN
10 35 45
_
```

How else can you use an array? Look at this program.

```
10 OPTION BASE 1
20 DIM P(3)
30 DATA 10,35
40 READ P(1),P(2)
50 P(3)=P(1)+P(2)
60 PRINT USING 70:P(1),P(2),P(3)
70 FORM N 2,POS 4,N 2,POS 7,N 2
80 END
```

Again, P(1), P(2), and P(3) all have individual values. P(1) and P(2) are being assigned values with the READ and DATA statements.

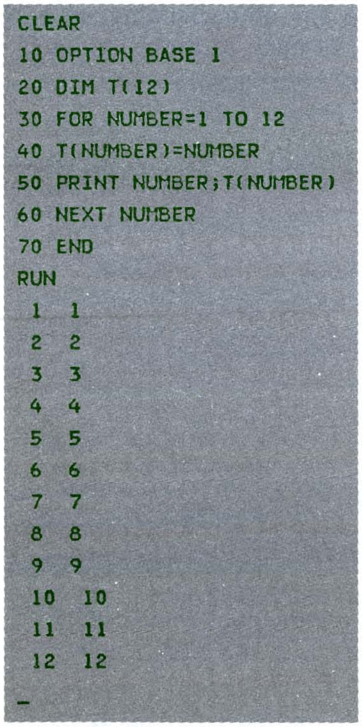
Using arrays in programs (continued)

If you look closely at these two programs, you will see that not much was gained by using an array. You still had to assign a value, one at a time, to each element.

But, subscripted variables and arrays can be very useful in a FOR/NEXT loop. Keep in mind, as you look at the following program, that array subscripts can also be variables.

Enter this program:

```
CLEAR
10 OPTION BASE 1
20 DIM T(12)
30 FOR NUMBER=1 TO 12
40 T(NUMBER)=NUMBER
50 PRINT NUMBER;T(NUMBER)
60 NEXT NUMBER
70 END
```



```
CLEAR
10 OPTION BASE 1
20 DIM T(12)
30 FOR NUMBER=1 TO 12
40 T(NUMBER)=NUMBER
50 PRINT NUMBER;T(NUMBER)
60 NEXT NUMBER
70 END
RUN
 1  1
 2  2
 3  3
 4  4
 5  5
 6  6
 7  7
 8  8
 9  9
 10  10
 11  11
 12  12
_
```

Before you run this program, lets look at what the program does. The DIM statement in line 20 indicates that there are 12 elements in the T array.

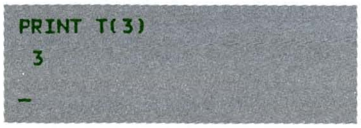
This program lets you assign all the values to the T array in one loop. The FOR-NEXT loop is run 12 times. Each time, the value of NUMBER is assigned to the T array.

Go ahead and run the program:

```
RUN
```

After you run the program, T(1)=1, T(2)=2, T(3)=3, etc. Let's be sure. Enter:

```
PRINT T(3)
```

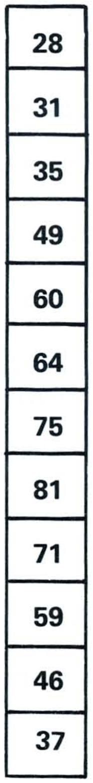


```
PRINT T(3)
 3
_
```

Now look at this program:

```
10 OPTION BASE 1
20 DIM T(12)
30 DATA 28,31,35,49,60,64
40 DATA 75,81,71,59,46,37
50 FOR I=1 to 12
60 READ T(I)
70 NEXT I
80 END
```

What happens if you run this program? You assign values with READ and DATA, and the T array looks like this:



Using RD in the OPTION statement

So far, you have been using the OPTION statement to indicate BASE 0 or BASE 1. There is another use for OPTION.

You may remember some of our earlier programs in which the results were often long decimal numbers. For example, in Book I, an accumulated savings amount was displayed as 121.550625000001.

By using RD 02 in the OPTION statement, this number would be displayed as 121.55. The 02 tells your System/23 to display all numbers rounded with only two digits to the right of the decimal point. The RD 02 applies only to the numbers that are displayed without using a FORM statement.

Let's look at a few more examples.

Number	**OPTION RD 02**	**OPTION RD 01**
18.41030	**18.41**	**18.4**
35.617	**35.62**	**35.6**
7	**7.00**	**7.0**

The number of digits to the right of the decimal point can be any number from 00 to 15.

You can include *both* the BASE and the RD in the same option statement. For example,

```
OPTION BASE 1,RD 02
or
OPTION RD 00,BASE 1
```

You can enter only one OPTION statement in a program.

Your turn!

Enter the following:

```
CLEAR
10 OPTION BASE 1,RD 02
20 P=100
30 I=.12
40 FOR N=1 TO 6
50 AMOUNT(N)=P*(1+I)**N
60 PRINT AMOUNT(N)
70 NEXT N
80 END
```

Now, add a statement on line 15 that will allow six elements in the AMOUNT array.

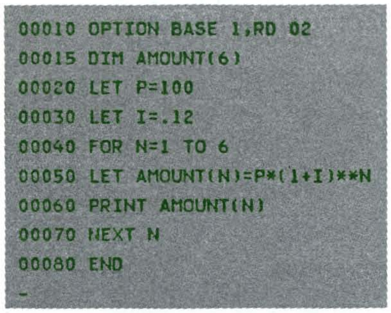
Answer: ________________________________

You should have entered:

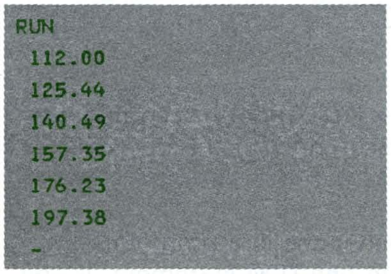
```
15 DIM AMOUNT(6)
```

List and run your program:

```
LIST
RUN
```



```
00010 OPTION BASE 1,RD 02
00015 DIM AMOUNT(6)
00020 LET P=100
00030 LET I=.12
00040 FOR N=1 TO 6
00050 LET AMOUNT(N)=P*(1+I)**N
00060 PRINT AMOUNT(N)
00070 NEXT N
00080 END
-
```



```
RUN
 112.00
 125.44
 140.49
 157.35
 176.23
 197.38
 -
```

Remember: If you do not dimension an array in a DIM statement, the array will be dimensioned to (10). Also remember from Book I: If you do not assign a value to a numeric variable in a program, the value of the variable is zero. So, if you changed BASE 1 to BASE 0 in this program, AMOUNT(0) would equal 0.

Numeric arrays

Chapter summary

A numeric array is a group of numeric variables which are all referred to by the same name. The array name is just like any other numeric variable name.

Each variable in the array is an element. An element is referred to by the array name and a subscript. The subscript indicates the position in the array.

A one-dimensional array can be thought of as a list. The subscript is a number that indicates the position by row.

You indicate the number of elements in an array by using the DIM statement. The OPTION statement lets you select BASE 0 or BASE 1 for the arrays in a program. OPTION also lets you select the number of digits to be displayed to the right of the decimal point with RD. RD applies to numbers that are displayed without using a FORM statement.

DIM and OPTION work like this:

```
10 OPTION BASE 0
20 DIM A(10)
```

Array A has 11 elements: A(0) – A(10).

```
10 OPTION BASE 1,RD 02
20 DIM A(10)
```

Array A has 10 elements: A(1)–A(10). Unformatted numbers are displayed with two digits to the right of the decimal point.

An array can be dimensioned only once in a program. Several arrays can be included in one DIM statement. If you do not include an array in a DIM statement, it will be dimensioned to (10).

Numeric arrays

Exercises

Question 1

Which of the following are valid program statements?

```
a. 10 DIM A(5)
b. 20 DIM AREA(3),RADIUS(3)
c. 30 DIM AREA(3),RADIUS(3),AREA(6)
d. 40 OPTION BASE 1
e. 50 OPTION BASE 0,RD 3
```

Question 2

What would be displayed if the following program were run?

```
10 OPTION BASE 1,RD 00
20 DIM X(10)
30 FOR I=1 TO 8
40 X(I)=10*I
50 NEXT I
60 FOR I=1 TO 10
70 PRINT X(I)
80 NEXT I
90 END
```

Answer: ____________________

Question 3

Add a statement to the following program, on line 15, to allow twelve variables in the AGE array.

```
10 OPTION BASE 1
20 FOR N = 1 TO 12
30 INPUT AGE(N)
40 NEXT N
50 END
```

Answer: ______________________________

Question 4

Add a statement to the following program, on line 5, to allow forty variables in the SCORE array.

```
10 DIM N(30),XYZ(15),SCORE(39)
20 FOR ABC = 0 TO 39
30 SCORE(ABC)=100
40 NEXT ABC
50 END
```

Answer: ______________________________

Numeric arrays

Answers

Question 1

a. Valid

b. Valid

c. Invalid—AREA cannot be dimensioned twice

d. Valid

e. Valid

Question 2

```
 10
 20
 30
 40
 50
 60
 70
 80
 0
 0
```

Remember: If you don't assign a value to a numeric variable, it equals zero.

Question 3

```
15 DIM AGE(12)
```

Question 4

```
5 OPTION BASE 0
```

Note that the OPTION BASE 0 is not required. Your System/23 automatically uses BASE 0 if there is no OPTION statement.

Chapter 2. Character arrays

Introduction

In Chapter 1 of this book, you learned how to use numeric arrays. You can also group several character variables in an array.

In this chapter, you will learn how to use *character arrays*. You will learn how to refer to a character array by its array name.

You will also learn how to specify the number of variables in an array, and the number of characters allowed for each variable.

Objectives

Upon completion of this chapter, you should be able to do the following:

- Refer to a collection of character variables by using an array.
- Specify the number of character variables in an array by using the DIM statement.
- Specify the number of characters allowed in a variable by using the DIM statement.

If you are familiar with these tasks, try the exercises located at the end of this chapter. If not, read through the chapter before going on to the exercises.

Character arrays

Array names and subscripts

In the last chapter, you learned about one-dimensional numeric arrays. The numeric array P, with four elements, looks like this:

Array P
P(0)
P(1)
P(2)
P(3)
P(4)

BASE 0

Array P
P(1)
P(2)
P(3)
P(4)

BASE 1

Well, you can also group character variables in an array. A character array is named just like a character variable. The name must begin with a letter and end with a $. It can have up to seven additional letters or numbers.

The elements of a character array are named by subscripted variables, like this:

$$\underbrace{\textbf{P\$}}_{\textbf{Array name}}\textbf{(}\underset{\nwarrow\ \textbf{Subscript}}{\textbf{1}}\textbf{)}$$

The first part of the subscripted variable is the array name. This can be any valid character variable name. The second part, in parentheses, is the subscript. Just like numeric array elements, the subscript indicates the position in the array.

The OPTION statement

A character array also begins with either position 0 or 1. For example,

Array P$	Array R$
P$(0)	R$(1)
P$(1)	R$(2)
P$(2)	R$(3)
P$(3)	R$(4)
P$(4)	
BASE 0	**BASE 1**

As with numeric arrays, the OPTION statement indicates which position the arrays in a program start with.

Let's look at an example. Enter the following:

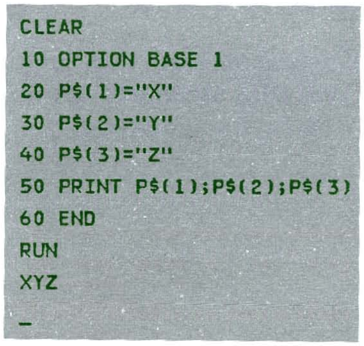
```
CLEAR
10 OPTION BASE 1
20 P$(1)="X"
30 P$(2)="Y"
40 P$(3)="Z"
50 PRINT P$(1);P$(2);P$(3)
60 END
```

Now run the program:

```
RUN
```

The values of P$(1), P$(2), and P$(3) are:

X	**P$(1)**
Y	**P$(2)**
Z	**P$(3)**



```
CLEAR
10 OPTION BASE 1
20 P$(1)="X"
30 P$(2)="Y"
40 P$(3)="Z"
50 PRINT P$(1);P$(2);P$(3)
60 END
RUN
XYZ
_
```

Character arrays

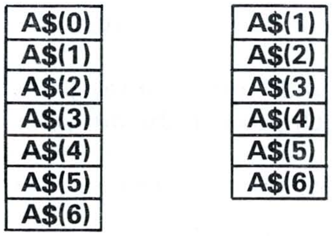
The DIM statement

The DIM statement performs *two* tasks for character arrays. Like numeric arrays, the DIM statement indicates how many elements are in an array.

```
20 DIM A$(6)
```

This statement dimensions the A$ array to (6). It could have 6 or 7 elements, depending on the OPTION statement.

Array A$** **Array A$



BASE 0 **BASE 1**

You can include numeric and character arrays in one DIM statement. But remember, you can't place the same array in a DIM statement more than once. And you'll get an error message when you try to run the program.

This is valid: 10 DIM Y(10),X$(5)
This is invalid: 10 DIM X$(5),Y$(6),X$(8)

The second use for the DIM statement is to state how long a character string can be. That is, the DIM statement can indicate the maximum number of characters in a string.

```
20 DIM A$*20,B$(6)*14
```

This statement sets the maximum number of characters allowed for A$ at 20. Each element of the B$ array is limited to 14 characters.

In Book III, you learned to specify the length of a character string being printed or displayed by using a C in the FORM statement.

```
10 DATA "NUTS",100,.29
20 READ ITEM$,QUANTITY,COST
30 PRINT USING 40:ITEM$,QUANTITY,COST
40 FORM C 10,POS 12,N 4,N 7.2
50 END
```

The value of ITEM$ is allowed ten positions by the C 10 in line 40. This limits the space for printing but not the actual length of the character string.

To specify the maximum length of a character string in a program, you use the DIM statement.

Look at the following program:

```
10 DIM ITEM$*10
20 PRINT "ENTER ITEM,QUANTITY,COST"
30 INPUT ITEM$,QUANTITY,COST
40 PRINT USING 50:ITEM$,QUANTITY,COST
50 FORM C 12,N 4,N 7.2
60 END
```

Line 10 sets the maximum size of ITEM$. When you input a value for ITEM$ in line 30, the value can be no longer than ten characters. If you want a maximum of five characters, you would enter:

```
10 DIM ITEM$*5
```

The DIM statement (continued)

Your turn!

What would you enter on line 10 if you wanted a maximum length of 12 characters for the value of ITEM$?

Answer: ____________________________

Your answer should be:

```
10 DIM ITEM$*12
```

Now look at the FORM statement in line 50:

```
50 FORM C 12,N 4,N 7.2
```

The FORM used for ITEM$ is C 12. This allows you 12 positions to display the value of ITEM$. But the length of ITEM$ has already been limited to a maximum of 10 characters by the original line 10.

```
10 DIM ITEM$*10
```

So, when line 40 is executed, the value of ITEM$ will never take up more than 10 positions.

Notice: Any spaces in a character string are counted as characters in the length of the string. But, the quotation marks around a string are *not* counted. For example,

10 SS$="123456789"
20 SS$="123 45 6789"

In line 10, SS$ has a length of 9. In line 20, its length is 11.

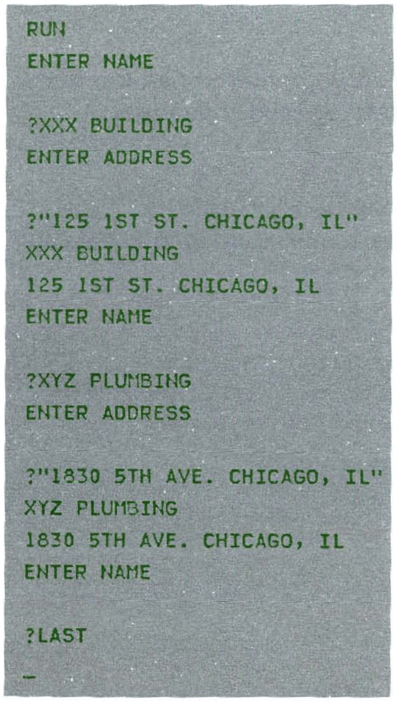
Let's look at another example. Enter the following program:

```
CLEAR
10 DIM MANE$*25,ADDRESS$*65
20 PRINT "ENTER NAME"
30 INPUT NAME$
40 IF NAME$="LAST" THEN GOTO 100
50 PRINT "ENTER ADDRESS"
60 INPUT ADDRESS$
70 PRINT USING 80:NAME$,ADDRESS$
80 FORM POS 1,C 25,POS 26,C 65
90 GOTO 20
100 END
```

Notice in line 10 that you can include more than one character variable in a DIM statement. Each name entered in line 30 is limited to 25 characters. Each address entered in line 60 is limited to 65 characters. Now run the program, using the following input entries:

```
RUN
ENTER NAME
?XXX BUILDING
ENTER ADDRESS
?"125 1ST ST. CHICAGO, IL"
ENTER NAME
?XYZ PLUMBING
ENTER ADDRESS
?"1830 5TH AVE. CHICAGO, IL"
ENTER NAME
?LAST
```

Notice that we used quotation marks in the addresses because of the commas.



```
RUN
ENTER NAME

?XXX BUILDING
ENTER ADDRESS

?"125 1ST ST. CHICAGO, IL"
XXX BUILDING
125 1ST ST. CHICAGO, IL
ENTER NAME

?XYZ PLUMBING
ENTER ADDRESS

?"1830 5TH AVE. CHICAGO, IL"
XYZ PLUMBING
1830 5TH AVE. CHICAGO, IL
ENTER NAME

?LAST
_
```

The DIM statement (continued)

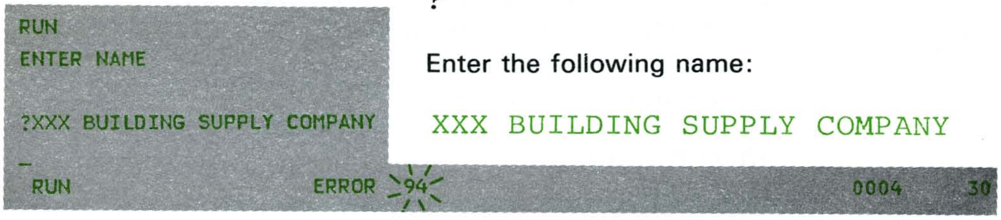
What happens if you enter a name that is longer than 25 characters? Let's run the program again and see:

```
RUN
ENTER NAME

?
```

Enter the following name:

```
XXX BUILDING SUPPLY COMPANY
```



```
RUN
ENTER NAME

?XXX BUILDING SUPPLY COMPANY
_
 RUN                          ERROR 94                              0004    30
```

The program stops at line 30, because in line 10, you defined the limit for NAME$ to be 25 characters. You entered the name XXX BUILDING SUPPLY COMPANY, which is too long. Any name you enter *must* be 25 characters or less.

Error code 0004 means that what you entered has too many characters. Press the Error Reset key, and then enter:

```
GO END
```

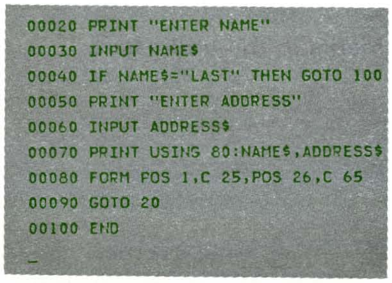
What happens if you don't state a maximum length for your character variables? Let's see.

Enter the following:

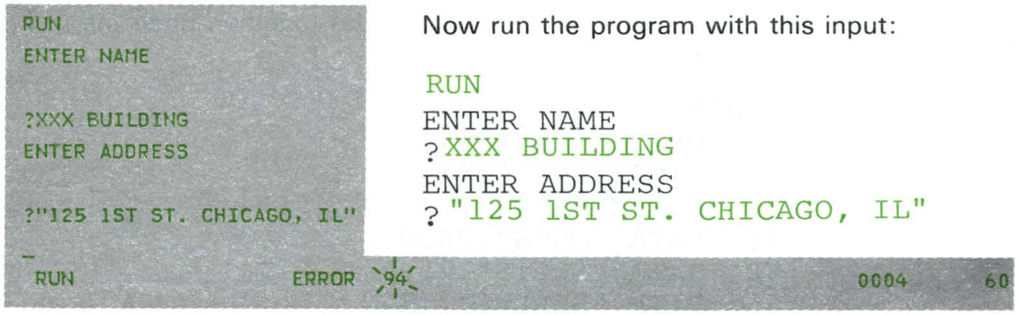
```
DEL 10
```

And list the new version of your program:

```
LIST
```



```
00020 PRINT "ENTER NAME"
00030 INPUT NAME$
00040 IF NAME$="LAST" THEN GOTO 100
00050 PRINT "ENTER ADDRESS"
00060 INPUT ADDRESS$
00070 PRINT USING 80:NAME$,ADDRESS$
00080 FORM POS 1,C 25,POS 26,C 65
00090 GOTO 20
00100 END
_
```



```
RUN
ENTER NAME

?XXX BUILDING
ENTER ADDRESS

?"125 1ST ST. CHICAGO, IL"
_
 RUN                    ERROR 94                                0004     60
```

Now run the program with this input:

```
RUN
ENTER NAME
?XXX BUILDING
ENTER ADDRESS
?"125 1ST ST. CHICAGO, IL"
```

The program stops again, because if you do not dimension a character variable, its maximum length is 18 characters.

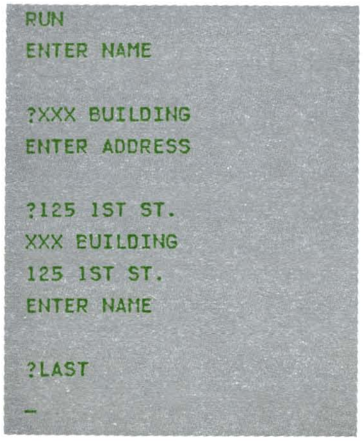
The first entry XXX BUILDING is accepted because it is only 12 characters long. But 125 1ST ST. CHICAGO, IL is too long and it causes an error 0004.

Press the Error Reset key, and enter:

```
GO END
```

Run the program again, but use the following input:

```
RUN
ENTER NAME
?XXX BUILDING
ENTER ADDRESS
?125 1ST ST.
ENTER NAME
?LAST
```



```
RUN
ENTER NAME

?XXX BUILDING
ENTER ADDRESS

?125 1ST ST.
XXX BUILDING
125 1ST ST.
ENTER NAME

?LAST
_
```

Remember: If you do not state a maximum number of characters for a string, the maximum is 18.

The DIM statement (continued)

Now let's put the two uses for DIM together. Enter the following program:

```
CLEAR
10  OPTION BASE 1
20  DIM ITEM$(3)*10
30  DIM NUMBER(3)
40  DATA "NUTS",2000
50  DATA "BOLTS",2500
60  DATA "SCREWS",1500
70  FOR X=1 TO 3
80  READ ITEM$(X),NUMBER(X)
90  PRINT ITEM$(X),NUMBER(X)
100 NEXT X
110 END
```

Before we run this program, let's look at the DIM statements.

In line 20, we state that there are three variables in the ITEM$ array. The value of each variable can be no longer than 10 characters.

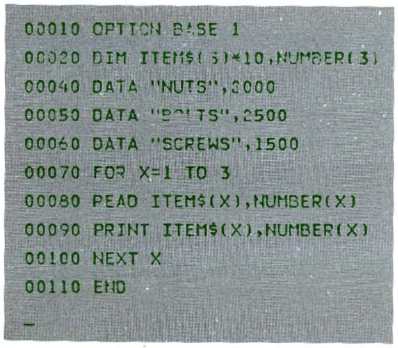
In line 30, we state that there are three variables in the NUMBER array.

We can combine these two statements. Enter:

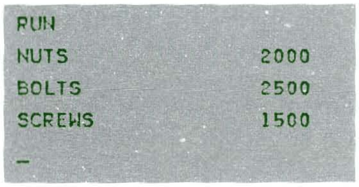
```
20 DIM ITEM$(3)*10,NUMBER(3)
DEL 30
```

Now list your program:

```
LIST
```



```
00010 OPTION BASE 1
00020 DIM ITEM$(3)*10,NUMBER(3)
00040 DATA "NUTS",2000
00050 DATA "BOLTS",2500
00060 DATA "SCREWS",1500
00070 FOR X=1 TO 3
00080 READ ITEM$(X),NUMBER(X)
00090 PRINT ITEM$(X),NUMBER(X)
00100 NEXT X
00110 END
_
```



```
RUN
NUTS                    2000
BOLTS                   2500
SCREWS                  1500
_
```

Now we're ready to run the program. Enter:

RUN

As you can see, the FOR-NEXT loop in lines 70 through 100 is run three times. Each time, we are reading and displaying an element of the character array ITEM$ and an element of the numeric array NUMBER.

Your turn!

What would you enter on line 20 to allow the following?

- 18 values, each 7 characters long, in the ITEM$ array
- 36 values in the NUMBER array

Answer: ______________________________

Your answer should be:

```
20 DIM ITEM$(18)*7,NUMBER(36)
```

You already know that if you do not assign a value to a numeric variable, its value is 0. If you do not assign a value to a character variable, its value is *null*. That is, it has no characters in it.

Remember, the DIM statement performs two tasks for character variables. With DIM, you limit the number of elements in an array, like DIM A$(6). With DIM, you also limit the number of characters in a string or in each array element, like DIM A$*12, B$(6)*10.

Character arrays

Chapter summary

A character array is a group of character variables which are all referred to by the same name. The array name is just like any other character variable name.

Each variable in the array is an element. An element is referred to by the array name and a subscript. The subscript indicates that position in the array.

You indicate the number of elements in an array *and* the maximum number of characters in a character variable by using the DIM statement.

If you do not indicate the number of elements in an array, there will be 10 elements in BASE 1 or 11 elements in BASE 0. If you do not indicate the maximum number of characters in a character variable, the maximum will be 18. The initial value of a character variable when you run a program is null.

Exercises

Use the following program to answer questions 1 and 2:

```
10 DIM I$*14,A$*25
20 INPUT I$,A$
30 PRINT USING 40:I$,A$
40 FORM C 14,POS 20,C 25
50 END
```

Question 1

Which of the following would be valid input entries for I$?

```
a. PICTURE FRAMES
b. 55
c. TELEVISION STANDS
d. 50,60
```

Question 2

Which of the following would be valid input entries for A$?

```
a. XYZ CORPORATION
b. BUILDING AND SUPPLY COMPANY
c. 55 1ST AVE.
d. "CHICAGO, IL"
```

Exercises (continued)

Question 3

Add a statement to this program, on line 20, that will do the following:

- Allow 10 values in the numeric array X
- Allow 5 values, each 13 characters long, in the character array P$
- Allow the character variable W$ to be 15 characters long

```
10 OPTION BASE 1
30 INPUT W$
40 FOR I=1 TO 10
50 INPUT X(I)
60 NEXT I
70 FOR J=1 TO 5
80 INPUT P$(J)
90 NEXT J
100 END
```

Answer: ______________________________________

Answers

Question 1

a. Valid

b. Valid—remember that you can input strings without using quotes

c. Invalid—I$ can only be up to 14 characters long

d. Invalid—because of the comma, this is two entries

Question 2

a. Valid

b. Invalid—A$ can only be up to 25 characters long (spaces are characters)

c. Valid

d. Valid

Question 3

```
20 DIM X(10),P$(5)*13,W$*15
```

Chapter 3. Two-dimensional arrays

Introduction

In Chapters 1 and 2, you learned how to store values in a *one-dimensional* array. Each individual element was thought of as an item in a list.

In this chapter, you will learn how to store values in a *two-dimensional* array. A *two-dimensional array* can be thought of as a table of values in rows and columns.

Objectives

Upon completion of this chapter, you should be able to do the following:

- Refer to a collection of numeric or character variables by using a two-dimensional array.
- Specify the size of a two-dimensional by array using the DIM statement.

If you are familiar with these tasks, try the exercises located at the end of this chapter. If not, read through the chapter before going on to the exercises.

Two-dimensional arrays

Numeric arrays

In Chapters 1 and 2, you learned that each element in a *one-dimensional* array is referred to by the array name and, in parentheses, its position in the array.

Here is a one-dimensional array A, in BASE 1, with four elements.

Array A

A(1)
A(2)
A(3)
A(4)

A *two-dimensional* array can be thought of as a *matrix*, or table, of values. The values are arranged in rows and columns in the table.

As with a one-dimensional array, each value in a two-dimensional array is an *element*. The variables are *subscripted*. A subscripted variable in a two-dimensional array looks like this:

$$\underset{\substack{\uparrow \\ \textbf{Array} \\ \textbf{name}}}{\mathbf{B}}\mathbf{(}\underbrace{\mathbf{1,2}}_{\nwarrow\ \textbf{Subscript}}\mathbf{)}$$

The numbers in the subscript represent the element's position by row and column. B(1,2) is in row 1, column 2.

When you enter OPTION BASE 0 in a program, a two-dimensional array begins with row 0, column 0. When you enter OPTION BASE 1, a two-dimensional array begins with row 1, column 1.

Look at the following statements:

```
10 OPTION BASE 1
20 DIM B(3,2),C(3,4)
```

The elements of the B array are arranged like this:

	Column	
Row	**1**	**2**
1	**B(1,1)**	**B(1,2)**
2	**B(2,1)**	**B(2,2)**
3	**B(3,1)**	**B(3,2)**

The values of the C array are arranged like this:

	Column			
Row	**1**	**2**	**3**	**4**
1	**1**	**31**	**35**	**2**
2	**60**	**64**	**7**	**81**
3	**71**	**4**	**46**	**37**

The value of C(1,1) is 1. The value of C(1,4) is 2.

Your turn!

What is the value of C(2,3)?

Answer:______________________________________

Your answer should be 7. C(2,3) is in row 2, column 3.

Two-dimensional arrays

Character arrays

Here is a one-dimensional array ITEM$ with three elements:

Array ITEM$ (in BASE 1)

NUTS
BOLTS
SCREWS

Here is a two-dimensional array STOCK$ with six elements:

Array STOCK$ (in BASE 1)

	Column		
Row	**1**	**2**	**3**
1	NUTS	BOLTS	SCREWS
2	NAILS	HAMMERS	WIRES

STOCK$(2,3) is WIRES. Just like two-dimensional numeric arrays, the subscript for each element is its row, column.

Note: You can store numeric variables or character variables in an array, but not both.

This is a valid numeric array:

1	2	3
30	40	50

This is an invalid numeric array. It is a valid character array:

1	A	-3
NUTS	15	8

The DIM statement

To specify the number of elements in a two-dimensional array, you use the DIM statement. The DIM statement for a two-dimensional array is almost like DIM for a one-dimensional array.

10 DIM P(20,10)

↗ ↖

Rows Columns

If you are using BASE 1, array P has 200 elements. They are arranged in 20 rows and 10 columns.

As with a one-dimensional numeric array, each element is set to zero when you run the program. The values may change as program statements are executed.

10 DIM R(6),AREA(6,2)

This statement states that there are two arrays in the program. Array R is one-dimensional. It has six values (in BASE 1). Array AREA is two-dimensional. It has six rows with two columns of values (in BASE 1).

If you use BASE 0, Array AREA would look like this:

Array AREA (in BASE 0)

	Column		
	0	**1**	**2**
Row			
0	**AREA(0,0)**	**AREA(0,1)**	**AREA(0,2)**
1	**AREA(1,0)**	**AREA(1,1)**	**AREA(1,2)**
2	**AREA(2,0)**	**AREA(2,1)**	**AREA(2,2)**
3	**AREA(3,0)**	**AREA(3,1)**	**AREA(3,2)**
4	**AREA(4,0)**	**AREA(4,1)**	**AREA(4,2)**
5	**AREA(5,0)**	**AREA(5,1)**	**AREA(5,2)**
6	**AREA(6,0)**	**AREA(6,1)**	**AREA(6,2)**

Two-dimensional arrays

The DIM statement (continued)

The size of a two-dimensional character array is also specified in the DIM statement.

```
10 DIM STOCK$(3,2)
             ↗ ↖
          Rows Columns
```

This statement states that array STOCK$ has six elements in BASE 1. They are arranged in three rows and two columns.

Now look at this statement:

10 DIM ITEM$(3,8),STOCK$(2,3)*8

Do you know what it does?

It states that array ITEM$ has three rows and eight columns of character values. Each string is limited to a maximum of 18 characters because you did not specify a length.

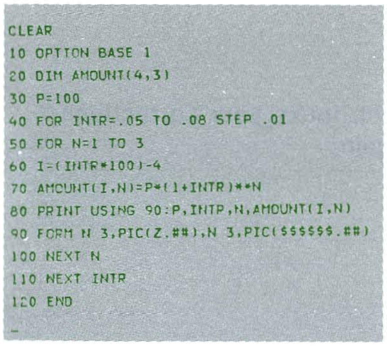
Array STOCK$ has two rows and three columns. Each string is limited to a maximum of 18 characters.

Using two-dimensional arrays in a program

Let's look at an example of how a two-dimensional array can be used in a program.

Do you remember the program that we used to calculate an amount of savings? Well, we're going to change that program to include an array.

Enter the following program:



```
CLEAR
10 OPTION BASE 1
20 DIM AMOUNT(4,3)
30 P=100
40 FOR INTR=.05 TO .08 STEP .01
50 FOR N=1 TO 3
60 I=(INTR*100)-4
70 AMOUNT(I,N)=P*(1+INTR)**N
80 PRINT USING 90:P,INTR,N,AMOUNT(I,N)
90 FORM N 3,PIC(Z.##),N 3,PIC($$$$$$.##)
100 NEXT N
110 NEXT INTR
120 END
-
```

```
CLEAR
10 OPTION BASE 1
20 DIM AMOUNT(4,3)
30 P=100
40 FOR INTR=.05 TO .08 STEP .01
50 FOR N=1 TO 3
60 I=(INTR*100)-4
70 AMOUNT(I,N)=P*(1+INTR)**N
80 PRINT USING 90:P,INTR,N,AMOUNT(I,N)
90 FORM N 3,PIC(Z.##),N 3,PIC($$$$$$.##)
100 NEXT N
110 NEXT INTR
120 END
```

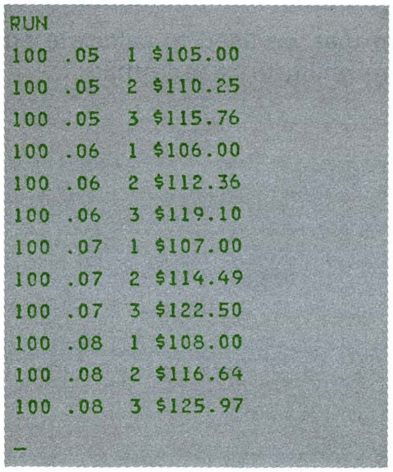
What will happen when you run this program?

The initial principal (P) is 100. First the interest is set equal to five percent. The amount is calculated for each of three years (lines 40-90).

Then the interest rate is changed to six percent, and the amounts are recalculated. The program continues to calculate amounts. Each amount is stored as an element of the AMOUNT array.

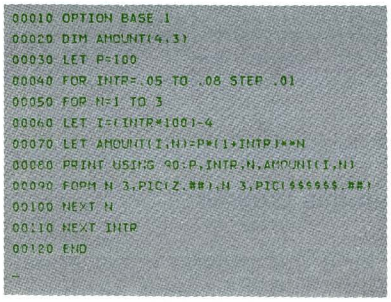
Two-dimensional arrays

Using two-dimensional arrays in a program (continued)



```
RUN
100 .05  1 $105.00
100 .05  2 $110.25
100 .05  3 $115.76
100 .06  1 $106.00
100 .06  2 $112.36
100 .06  3 $119.10
100 .07  1 $107.00
100 .07  2 $114.49
100 .07  3 $122.50
100 .08  1 $108.00
100 .08  2 $116.64
100 .08  3 $125.97

_
```



```
00010 OPTION BASE 1
00020 DIM AMOUNT(4,3)
00030 LET P=100
00040 FOR INTR=.05 TO .08 STEP .01
00050 FOR N=1 TO 3
00060 LET I=(INTR*100)-4
00070 LET AMOUNT(I,N)=P*(1+INTR)**N
00080 PRINT USING 90:P,INTR,N,AMOUNT(I,N)
00090 FORM N 3,PIC(Z.##),N 3,PIC($$$$$$.##)
00100 NEXT N
00110 NEXT INTR
00120 END

_
```

Look at line 60.

```
60 I=(INTR*100)-4
```

This statement multiplies the interest rate (e.g. .05) by 100. Then, 4 is subtracted, which produces the numbers 1, 2, 3, and 4 for the "row" number in the amount array. Now run the program:

```
RUN
```

If you didn't get the correct results, list the program in your work area:

```
LIST
```

If any of your lines are incorrect, make your corrections now. Then run the program again:

```
RUN
```

Chapter summary

There are two types of arrays used in BASIC programs:

- One-dimensional arrays
  - Variables are stored as items in a list.
  - Only one type of data (numeric or character) can be in each array.
  - Individual elements are referred to as A(1), ITEM$(3), etc.

- Two-dimensional arrays
  - Variables are stored as items in a table of rows and columns.
  - Only one type of data (numeric or character) can be in each array.
  - Individual elements are referred to as A(1,1), ITEM$(3,1), etc.

The DIM statement states the number of elements in an array. It also states the maximum number of characters in a character variable. For example,

```
DIM XYZ(3,4),X$(4,2)*24

Rows Columns  Rows Columns  Characters
```

Two-dimensional arrays

Exercises

Question 1

What will be stored in the following arrays after these programs are run?

```
a. 10 OPTION BASE 1
   20 DIM W(2,2)
   30 W(1,1)=0
   40 W(1,2)=2
   50 W(2,1)=15
   60 W(2,2)=-3
   70 END
```

Array W


```
b. 10 OPTION BASE 1
   20 DIM NAME$(2,2)
   30 NAME$(1,2)="DOE"
   40 NAME$(2,2)="SMITH"
   50 NAME$(1,1)="JOHN"
   60 NAME$(2,1)="JIM"
   70 END
```

Array NAME$


```
c. 10 OPTION BASE 1
   20 DIM X(2,5)
   30 FOR NUMBER=1 TO 5
   40 X(1,NUMBER)=2
   50 X(2,NUMBER)=4
   60 NEXT NUMBER
   70 END
```

Array X

Question 2

What will appear on the screen if the following programs are run?

```
a. 10 OPTION BASE 1
   20 DIM TOTAL(2,3)
   30 FOR I=1 TO 2
   40 TOTAL(1,I)=100
   50 TOTAL(2,I)=200
   60 PRINT TOTAL(1,I),TOTAL(2,I)
   70 NEXT I
   80 END

b. 10 OPTION BASE 1
   20 DIM A$(2,2)*6
   30 A$(1,1)="ABC"
   40 A$(1,2)="CBA"
   50 A$(2,1)=A$(1,1)
   60 A$(2,2)=A$(1,2)
   70 PRINT USING 80:A$(2,1),A$(2,2)
   80 FORM C 6,C 6
   90 END
```

Answer: a. ____________________

____

b. ____________________

Two-dimensional arrays

Answers

Question 1

a.

0	2
15	-3

b.

JOHN	DOE
JIM	SMITH

c.

2	2	2	2	2
4	4	4	4	4

Question 2

a.
```
 100                  200
 100                  200
```

b.
```
ABC   CBA
```

Chapter 4. Matrix operations

Introduction

In Chapter 3, you learned that another word for an array is a *matrix*. In this chapter, you will learn how to perform several matrix operations.

We will show you how to assign values to an entire array. We will also show you how to display and print all of the contents of an array.

Objectives

Upon completion of this chapter, you should be able to do the following:

- Display the contents of an entire array on the screen by using the PRINT MAT statement.
- Print the contents of an entire array with the printer by using the PRINT #255:MAT statement.
- Assign values to every element in an array by using a MAT assignment statement.

If you are familiar with these tasks, try the exercises located at the end of this chapter. If not, read through the chapter before going on to the exercises.

Using PRINT and PRINT #255:

You already know how to display values by using the PRINT statement, like this:

```
10 PRINT XYZ(1),XYZ(2),XYZ(3)
```

You also know how to use the PRINT USING and FORM statements, like this:

```
10 PRINT USING 20:XYZ(1),XYZ(2),XYZ(3)
20 FORM N 6,POS 8,N 6,POS 16,N 6
```

Now we will show you how to display an entire array by using one statement.

```
10 PRINT MAT SA
           ↗   ↑
Matrix    Array name
```

In this statement, MAT means *mat*rix, or array. The name of the array is SA. This statement tells your System/23 to display all of the elements of the SA array.

Let's see how this works in a program. Let's write a program that stores each of eight employees' sickness days in an array called SA. The numbers will be assigned by READ and DATA statements. The program will display the numbers by using a PRINT MAT statement.

First of all, make sure there is nothing in the work area. Enter:

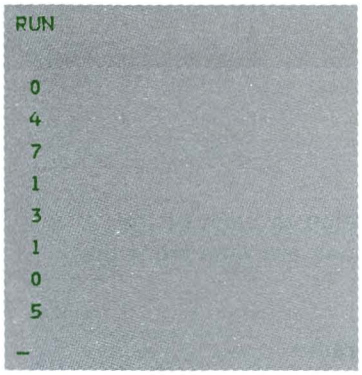
```
CLEAR
```

Now, enter the following:

```
10 OPTION BASE 1
20 DIM SA(8)
30 DATA 0,4,7,1,3,1,0,5
40 FOR I=1 TO 8
50 READ SA(I)
60 NEXT I
70 PRINT MAT SA
80 END
```

Go ahead and run the program:

```
RUN
```



```
RUN

 0
 4
 7
 1
 3
 1
 0
 5
_
```

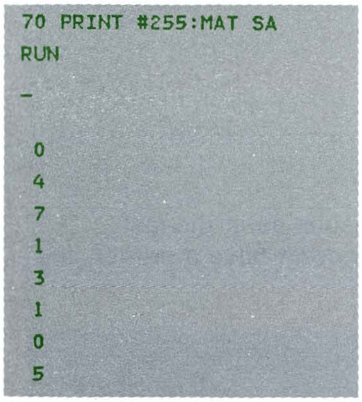
As you can see, all eight elements of the SA array are displayed. And it only requires one program statement.

Now, if you have a printer attached to your System/23 system, enter the following:

```
70 PRINT #255:MAT SA
```

And run your new program:

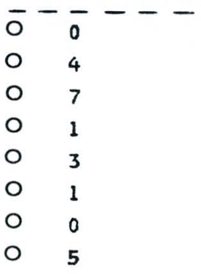
```
RUN
```



```
70 PRINT #255:MAT SA
RUN
_

 0
 4
 7
 1
 3
 1
 0
 5
```

This time, the results are printed by the printer.



```
0
4
7
1
3
1
0
5
```

Remember:

PRINT MAT SA *displays* an entire array.

PRINT #255:MAT SA *prints* an entire array.

Matrix operations

Using READ/DATA and INPUT

List the program in the work area:

```
LIST
```

```
00010 OPTION BASE 1
00020 DIM SA(8)
00030 DATA 0,4,7,1,3,1,0,5
00040 FOR I=1 TO 8
00050 READ SA(I)
00060 NEXT I
00070 PRINT MAT SA
00080 END
-
```

Now we'll show you how to combine lines 40, 50 and 60 into one statement. Enter the following:

```
DEL 40,60
```

Now, enter:

```
40 READ MAT SA
```

This READ statement assigns a value to each of the elements in the SA array. The values are read from the DATA statement in line 30.

List the new version of your program:

```
LIST
```

```
00010 OPTION BASE 1
00020 DIM SA(8)
00030 DATA 0,4,7,1,3,1,0,5
00040 READ MAT SA
00070 PRINT MAT SA
00080 END
-
```

Run the program:

```
RUN
```

```
RUN


 0
 4
 7
 1
 3
 1
 0
 5
-
```

If you have a printer, the same results as in the last example should be printed. If you don't have a printer, the results should be displayed.

Remember:

READ MAT SA assigns a DATA value to each element in the SA array.

The same method applies to the INPUT statement.

Enter this program:

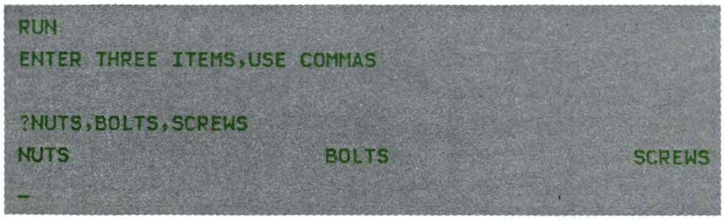
```
CLEAR
10 OPTION BASE 1
20 DIM ITEM$(3)
30 PRINT "ENTER THREE ITEMS, USE COMMAS"
40 INPUT MAT ITEM$
50 PRINT ITEM$(1),ITEM$(2),ITEM$(3)
60 END
```

Look at line 40. This statement allows you to input values from the keyboard for each element of the ITEM$ array.

Now run the program with this input:

```
RUN
ENTER THREE ITEMS, USE COMMAS
?NUTS,BOLTS,SCREWS
```

The input values are assigned to the ITEM$ array, one element at a time.



```
RUN
ENTER THREE ITEMS,USE COMMAS

?NUTS,BOLTS,SCREWS
NUTS                BOLTS               SCREWS
_
```

Matrix operations

Using MAT assignments

Here are some more ways to assign values to an array.

Look at the following statements:

```
10 OPTION BASE 1
20 DIM Y(3)
30 MAT Y=(5)
          \/
          Required
```

These statements indicate that numeric array Y has three elements. Line 30 sets each element in array Y equal to 5.

Array Y

5
5
5

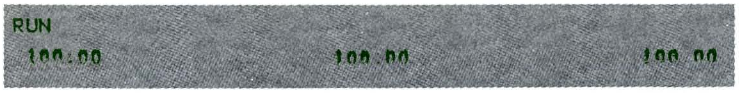
Note that the parentheses are required around the value (5).

Let's try this. Enter the following:

```
CLEAR
10 OPTION BASE 1, RDƀ2
20 DIM SCORE(8)
30 MAT SCORE=(100)
40 PRINT SCORE(1),SCORE(2),SCORE(8)
50 END
```

Now, run the program:

```
RUN
```



```
RUN
 100.00                          100.00                          100.00
```

This method also works with character arrays.

```
10 OPTION BASE 1
20 DIM NAME$(2,3)
30 MAT NAME$=("IBM")
             \     /
              Required
```

Each of the elements in the NAME$ array is set to IBM.

Array NAME$

IBM	**IBM**	**IBM**
IBM	**IBM**	**IBM**

Note: Parentheses and quotes are required around ("IBM").

When you assign values to a two-dimensional array, the values are assigned by row and column. For example,

```
10 OPTION BASE 1
20 DIM A(2,3)
30 DATA 1,2,3,4,5,6
40 READ MAT A
50 END
```

The A array contains the following:

Column	**1**	**2**	**3**
Row			
1	1	2	3
2	4	5	6

Using MAT assignments (continued)

You can set all of the values in a numeric array to zero by using ZER. It's done like this:

```
10 MAT A=ZER
```

Here is another way to assign values to an entire array.

```
10 DIM A(2,2),B(2,2)
20 MAT A=B
```

The values of the elements in array B are now assigned also to the corresponding elements in array A.

If array B looks like this,

10	**15**
20	**25**

What will array A look like? It will look like this:

10	**15**
20	**25**

Remember: When assigning values from one array to another, the arrays must both be the same type (numeric or character). The arrays must also have the same dimension.

See "MAT assignment" in your *BASIC Language Reference* for more details on MAT assignments.

Now, notice the difference in these two statements:

10 MAT A=(10)	**10 MAT A = B**
Parentheses required- Each element of array A is equal to 10.	**No parentheses- Each element of array A is equal to its corresponding element of array B.**

Your turn!

Look at the following program:

```
10 OPTION BASE 0
20 DIM NUM(8),SCORE(8)
30 MAT SCORE=(100)
40 PRINT MAT NUM
50 END
```

What would you enter on line 35 to set each element of array NUM equal to its corresponding element in array SCORE?

Answer: ______________________________

You should have said:

```
35 MAT NUM=SCORE
```

Matrix operations

Chapter summary

You use the PRINT MAT statement to display all of the elements in an array. You use the PRINT #255:MAT statement to print all of the elements in an array. For example,

```
10 PRINT MAT A
20 PRINT #255:MAT B
```

You can assign values to all of the elements of an array by using MAT assignments.

- MAT A=(12) - each element in Array A equals 12.
- MAT A$=("JOHN") - each element in array A$ equals JOHN.
- MAT A=ZER - each element in array A equals 0.
- MAT A=B - each element in array A equals the corresponding element in array B.
- MAT A$=B$ - each element in array A$ equals the corresponding element in array B$.
- READ MAT A - the values in a DATA list are assigned to each element of the A array.
- INPUT MAT A - the values input from the keyboard are assigned to each element of the A array.

Exercises

Question 1

What will be displayed if the following programs are run?

```
a. 10 OPTION BASE 1
   20 DIM X(3)
   30 MAT X=(55)
   40 PRINT USING 50:MAT X
   50 FORM N 2,SKIP 2,N 2,X 3,N 2
   60 END

b. 10 OPTION BASE 1
   20 DIM X$(2)
   30 INPUT MAT X$
   40 PRINT MAT X$
   50 END
```

(The input items in b. are "DOGS" and "CATS".)

Answer: a. ____________________

____

b. ____________________

____

Matrix operations

Exercises (continued)

Question 2

What will be printed if the following programs are run?

```
a. 10 OPTION BASE 1
   20 DIM A(2,2),B(2,2)
   30 MAT A=ZER
   40 MAT B=A
   50 PRINT #255:B(2,1)
   60 END

b. 10 OPTION BASE 1
   20 DIM A(2,2)
   30 DATA 5,10,-3,0
   40 READ MAT A
   50 PRINT #255:A(1,1),A(2,2)
   60 END
```

Answer: a. ____________________

b. ____________________

Answers

Question 1

a.
```
55

55   55
```

b.
```
DOGS
CATS
```

Question 2

a.
```
 0
```

b.
```
 5                                      0
```

READER'S COMMENT FORM

SA34-0124-0

IV. Arrays and Subscripts

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or the IBM branch office serving your locality.

Corrections or clarifications needed:

Page Comment

Please indicate your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Cut or Fold Along Line

Reader's Comment Form

Cut Along Line

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Systems Publications, Dept 27T
P.O. Box 1328
Boca Raton, Florida 33432

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

SA34-0124-0
Printed in U.S.A.

SA34-0124-0
Printed in U.S.A.