THE IBM 5100 PORTABLE COMPUTER

A Comprehensive Guide For Users and Programmers

HARRY KATZAN, JR.

Computer Consultant
Chairman, Computer Science Department
Pratt Institute

COMPUTER SCIENCE SERIES

VNR VAN NOSTRAND REINHOLD COMPANY
NEW YORK CINCINNATI ATLANTA DALLAS SAN FRANCISCO
LONDON TORONTO MELBOURNE

Van Nostrand Reinhold Company Regional Offices:
New York Cincinnati Atlanta Dallas San Francisco

Van Nostrand Reinhold Company International Offices:
London Toronto Melbourne

Library of Congress Catalog Card Number: 77-2168
ISBN: 0-442-24270-0

Manufactured in the United States of America

Published by Van Nostrand Reinhold Company
450 West 33rd Street, New York. N.Y. 10001

Published simultaneously in Canada by Van Nostrand Reinhold Ltd.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging in Publication Data

Katzan, Harry.
The IBM 5100 portable computer.

(Computer science series)
Includes bibliographical references and index.
1. IBM 5100 (Computer) 2. APL (Computer program language) 3. BASIC (Computer program language) I. Title.
QA76.8.I19K37 001.6'4 77-2168
ISBN 0-442-24270-0

and may be "flowed into," as in the following case:

```
                       10  P9=P9+1
                       20  PRINT USING FLP,30,P9
                       30  :              TITLE    PAGE ###
When the           ┌──40  IF P9=1 GOTO 60
Subroutine is      │   50  RETURN──────────┐   When the subroutine is
"flowed into"      └─→60  ——               │     "transferred into"
                            .               │
                            .               │
                            .               ↓
                      510  GOSUB 10
                            .
                            .
                            .
                      700  GOSUB 10
```

When a subroutine is both flowed into and transferred into, a branch should be made around the RETURN statement in the former case.

7.4 MATRIX OPERATIONS

Matrix (MAT) statements in the BASIC language permit operations that deal with complete arrays, thereby subordinating the detail normally associated with programming to the computer. Each matrix statement begins with the prefix MAT and requires that constituent arrays be declared explicitly or implicitly beforehand. Matrix arithmetic operations are defined only on numeric arrays; matrix assignment and input/output statements are additionally defined on character-string arrays.

The DIM Statement

The DIM statement is used to explicitly declare an array and thereby assign it a name and its row and column bounds. The form of the DIM statement is:

DIM array-name (rows[,columns]) [,array-name(rows[,columns])] . . .

where *array-name* is the name of the numeric or character-string array being declared, and the entries *rows* and *columns* are non-zero positive integer constants specifying the dimensions of the array. The rows entry gives the length of a one-dimensional array and both the *rows* and *columns* entries must be used for two-dimensional arrays. Each element of a numeric array is initialized to zero and each element of a character-string array is initialized to 18 blank characters. The maximum size of an array dimension is 255. The following DIM statement

DIM A(11,23),B(3,4),C(15,1),D(50),E(1,6)

for example, establishes the following numeric arrays:

A has 11 rows and 23 columns
B has 3 rows and 4 columns
C has 15 rows and 1 column
D has 50 elements
E has 1 row and 6 columns

Similarly, the following DIM statement:

DIM F$(3,4),G$(50) ,H$(100,5)

establishes the following character-string arrays:

F$ has 3 rows and 4 columns
G$ has 50 elements
H$ has 100 rows and 5 columns

Numeric and character-string array declarations can be made in the same DIM statement, which must be placed in the program prior to the first reference to the array. Otherwise, there are no restrictions on where a DIM statement must be placed in a program.

Array Replacement

All MAT operations result in the replacement of the elements of an array. A few operations pertain to both numeric and character-string arrays and are referred to as array operations, and use operands denoted as *"array-name."* Operations defined only on numeric arrays are referred to as matrix operations, and use operands denoted as *"matrix-name."* The array replacement operations take the following form:

MAT array-name [(rows [,columns])] = { (scalar-exp) / array-name }

where *array-name* is a previously defined numeric or character-string array and *scalar-exp* is an expression of the same type. The *rows* and *columns* options refer to redimensioning, covered below.

Scalar Replacement. Replacement of the elements of an array with the value of a scalar expression takes the following form:

```
MAT A = (e)
    |    |
    |    └──→Evaluated first
    └───────→Scalar value is assigned to each
               element of A
```

If the dimension of A is A(k), then the scalar replacement statement is equivalent to A(i)=e, for i=1,2, . . . ,k. If the dimension of A is A (m,n), then the scalar replacement statement is equivalent to A(i,j)=e, for i=1,2, . . . ,m and j=1,2, . . . ,n. The following statements demonstrate scalar replacement:

MAT B = (−5.341)
MAT C = (3∗A2↑2+6∗A2−17.1)
MAT D$ = ('ABCD')
MAT E$ = (STR(D$,7,2))

In scalar replacement, the scalar expression must be enclosed in parentheses. Redimensioning may also apply to scalar replacement. Figures 7.16 and 7.17 give examples of scalar replacement. In several of the figures that follow, the MAT PRINT statement is used. This statement is introduced later in the chapter. At this stage, it is sufficient to know that it can be used to print or display a complete array, without having to explicitly print or display each item of the array.

Array Replacement. Replacement of the elements of an array with the elements of another array on an element-by-element basis takes the following simplified form:

MAT A=B

where the two arrays A and B must have the same dimensions, possibly after redimensioning, if specified. If the dimensions of A and B are A(k) and B(k), respectively, then the array replacement statement is equivalent to A(i)=B(i), for

```
0010 DIM A(13),B(2,3)
0020 MAT A=(5.31)
0030 MAT B=(INT(10*RND(123)))
0040 MAT PRINT A;
0050 MAT PRINT B;
RUN

 5.31      5.31      5.31      5.31      5.31      5.31      5.31
 5.31      5.31      5.31      5.31      5.31      5.31



 1      1      1

 1      1      1


READY                                                          28092
```

Figure 7.16 Replacement of the elements of a numeric array with the value of a numberic scalar expression. Note the required parentheses around the expressions.


```
0010 DIM A$(3),B$(2,3)
0020 MAT A$=('TEA FOR TWO')
0030 MAT B$=(STR(A$(2),5,3))
0040 MAT PRINT A$
0050 MAT PRINT B$
RUN

TEA FOR TWO          TEA FOR TWO          TEA FOR TWO



FOR                  FOR                  FOR

FOR                  FOR                  FOR


READY                                                          28065
```

Figure 7.17 Replacement of the elements of a character-string array with the value of a character-string expression. Note the required parentheses around the scalar expressions.

i=1,2, . . . ,k. If the dimensions of A and B are A(m,n) and B (m,n), respectively, then the array replacement statement is equivalent to A(i,j)=B(i,j), for i=1,2, . . . ,m and j=1,2, . . . ,n. The following statements demonstrate array replacement:

MAT P=Q
MAT T$=V$

```
0010 DIM X(13),Y(13),U(2,3),V(2,3)
0020 MAT Y=(&PI)
0030 MAT V=(&PI+1)
0040 MAT X=Y
0050 MAT U=V
0060 MAT PRINT X;
0070 MAT PRINT U;
RUN RD=3

 3.142     3.142     3.142     3.142     3.142     3.142     3.142
 3.142     3.142     3.142     3.142     3.142     3.142



 4.142     4.142     4.142

 4.142     4.142     4.142


READY                                                          27888
```

Figure 7.18 Numeric array replacement.

```
0010 DIM A$(4),B$(4),C$(2,3),D$(2,3)
0020 MAT B$=('AUDIT')
0030 MAT D$=('CONTROL')
0040 MAT A$=B$
0050 MAT C$=D$
0060 MAT PRINT A$
0070 MAT PRINT C$
RUN

AUDIT            AUDIT            AUDIT            AUDIT



CONTROL          CONTROL          CONTROL

CONTROL          CONTROL          CONTROL



 READY                                                        27822
```

Figure 7.19 Character-string array replacement.

Redimensioning may also apply to array replacement. Figures 7.18 and 7.19 give examples of array replacement.

Redimensioning. An array can be redimensioned in any MAT statement that assigns a value to its elements. Replacement and input statements fall into this category. Rediminsioning is achieved by following the replaced array with the new dimensions enclosed in parentheses, as follows:

MAT a(e) = . . .

→ The value of this expression determines the number of elements in the redimensioned array.

or

MAT a(e_1,e_2) = . . .

→ The value of these expressions determine the number of rows and columns, respectively, in the redimensioned array.

A dimension of the redimensioned array can be specified as a scalar numeric expression which is evaluated at the point of reference and truncated to an integer. Redimensioning applies to numeric and character-string arrays and is governed by the following rules:


```
0010 DIM A(10),B(8),C(3,4),D(2,3)
0020 MAT B=(180/&PI)
0030 MAT D=(&PI/180)
0040 MAT A(8)=B
0050 MAT C(2,3)=D
0060 MAT PRINT A;
0070 MAT PRINT C;
RUN RD=4


57.2958     57.2958     57.2958     57.2958     57.2958
57.2958     57.2958     57.2958



1.7453E-2   1.7453E-2   1.7453E-2

1.7453E-2   1.7453E-2   1.7453E-2



READY                                                    27862
```

Figure 7.20 Redimensioning of a numeric array.

1. The total number of elements in the redimensioned array may not exceed the number of elements in the original array.

```
0010 DIM E$(10),F$(3),G$(5,4),H$(2,3)
0020 MAT F$=('INVENTORY')
0030 MAT H$=('CONTROL')
0040 MAT E$(3)=F$
0050 MAT G$(2,3)=H$
0060 MAT PRINT E$
0070 MAT PRINT G$
RUN


INVENTORY        INVENTORY        INVENTORY



CONTROL          CONTROL          CONTROL

CONTROL          CONTROL          CONTROL



READY                                                    27444
```

Figure 7-21 Redimensioning of a character-string array.

2. Redimensioning applies to both one-dimensional and two-dimensional arrays.
3. The number of dimensions in an array can be changed with redimensioning.

The following statements demonstrate redimensioning:

```
MAT A(3,4)=(&PI)
MAT B$(15)=(STR(P$,4,10))
```

Redimensioning also applies to other MAT statements in the same manner. Figures 7.20 and 7.21 give examples of redimensioning.

Matrix Arithmetic

Matrix arithmetic statements permit arithmetic operations to be performed on the elements of a numeric array on an element-by-element basis. Since the operations are numerical, the arrays are referred to as matrices. Matrix arithmetic statements have the following form:

```
                                      ⎧ matrix-name {±} matrix-name ⎫
MAT matrix-name [(rows [,columns])] = ⎨                             ⎬
                                      ⎩ (arith-exp) *matrix-name    ⎭
```

where *matrix-name* denotes a numeric array. In the matrix addition and subtraction operations, all three matrices must have the same dimensions after redimensioning, if specified. *Arith-exp* is a numeric scalar expression.

Matrix Addition. Matrix addition takes the following simplified form:

```
MAT C=A+B
```

and adds matrix B to matrix A on an element-by-element basis and replaces matrix C with the result. If the dimension of A,B, and C is (k) then the matrix addition statement is equivalent to C(i)=A(i)+B(i), for i=1,2, . . . ,k. If the dimensions of A,B, and C are (m,n), then the matrix addition statement is equivalent to C(i,j)=A(i,j)+B(i,j), for i=1,2, . . . ,m and j=1,2, . . . ,n.

Matrix Subtraction. Matrix subtraction takes the following simplified form:

```
MAT C=A-B
```

and subtracts matrix B from matrix A on an element-by-element basis and replaces matrix C with the result. If the dimension of A,B, and C is (k), then the matrix subtraction statement is equivalent to C(i)=A(i)-B(i), for i=1,2, . . . ,k. If the dimensions of A,B, and C are (m,n), then the matrix subtraction statement is equivalent to C(i,j)=A(i,j)-B(i,j), for i=1,2, . . . ,m and j=1,2, . . . ,n.

Scalar Multiplication. Scalar multiplication takes the following simplified form:

```
MAT C = (e) *B
```

where matrices B and C have the same dimensions and e is a numeric scalar expression evaluated at the point of reference. The statement multiplies matrix B by expression e on an element-by-element basis and replaces matrix C with the result. If the dimension of matrices C and B is (k), then the scalar multiplication statement is equivalent to C(i)=e*B(i), for i=1,2, . . . ,k. If the dimensions of C and B are (m,n), then the scalar multiplication statement is equivalent to C(i,j)=e*B(i,j), for i=1,2, . . . ,m and j=1,2, . . . ,n.

Examples. The following statements demonstrate the use of matrix arithmetic:

```
MAT Q=F-R
MAT W(6,24)=(&PI*R↑2)*D
MAT P=(14.731)*N
MAT D(9,1)=I+J
```

Figure 7.22 gives the computer printout for several examples of matrix arithmetic.

Significant Characteristic. One significant characteristic of matrix arithmetic statements is that the same matrix can appear on both sides of the equals sign, as follows:

```
MAT A=A+B
```

Therefore, if it were desired to add a constant, such as 5 to every element of a matrix, then a sequence of statements, such as the following would be used:

```
DIM A(20,30),B(20,30)
.
.
.
MAT B=(5)
MAT A=A+B
```

Similarly, if it were desired to multiply every element of matrix A by 10, one could write:

```
MAT A=(10)*A
```

Logically, a matrix arithmetic operation, such as:

```
MAT A=A+B
```

```
0010 DIM A(10),B(2,3),C(2,3),D(2,3)
0020 MAT A=(5)
0030 MAT C=(6)
0040 MAT D=(2)
0050 MAT A=(2)*A
0060 MAT PRINT FLP,A;
0070 MAT B=C+D
0080 MAT PRINT FLP,B;
0090 MAT B=C-D
0100 MAT PRINT FLP,B;
0110 MAT A(2,3)=(-1)*D
0120 MAT PRINT FLP,A;
```

(A) Program

```
 10   10   10   10   10   10   10   10   10   10


 8    8    8

 8    8    8


 4    4    4

 4    4    4


-2   -2   -2

-2   -2   -2
```

(B) Input

Figure 7.22 Matrix arithmetic demonstrating matrix addition, matrix subtraction, scalar multiplication, and redimensioning.

is interpreted as follows, "Add matrix B to matrix A and replace matrix A with the result." In reality, the operation is performed on an element-by-element basis, equivalent to the following nested FOR loop:

```
FOR I=1 TO M
   FOR J=1 TO N
      A(I,J)=A(I,J)+B(I,J)
   NEXT J
NEXT I
```

where *M* and *N* are the row and column bounds, respectively. Similarly, the statement

MAT A = (3*X+B)*A

is equivalent to the following nested FOR loop:

```
T=3*X+B
FOR I=1 TO M
   FOR J=1 TO N
      A(I,J)=T*A(I,J)
   NEXT J
NEXT I
```

where again, M and N are the row and column bounds, respectively. The scalar expression in matrix arithmetic is *always* evaluated first and its *value does not change* during the matrix operation. The following example demonstrates this point. In the statement:

MAT A=(A(2,3))*A

which is equivalent to the following nested FOR loops:

```
T=A(2,3)
FOR I=1 TO M
   FOR J=1 TO N
      A(I,J)=T*A(I,J)
   NEXT J
NEXT I
```

the elements of matrix A are each multiplied by the same value, namely, the initial value of A(2,3), even though the value of A(2,3) in the resulting matrix is changed part way through the computation. The above concepts also apply to one-dimensional numeric arrays in an analogous fashion.

Matrix Mathematics

Matrix mathematical operations are permitted on previously declared matrices. This facility allows an identity matrix to be established and includes facilities for the transpose, inverse, and matrix multiplication functions.

Identity Function. The identity function permits an identity matrix to be assigned to a square matrix and has the following format:

MAT matrix-name[(rows,columns)]=IDN

where *matrix-name* is a square numeric matrix and *rows* and *columns* are arithmetic expressions evaluated at the point of reference, as explained above, and specify redimensioning. The following statements, for example:

```
DIM A(50),B(3,3)
.
.
.
MAT B=IDN
MAT A(7,7)=IDN
```

would create the following matrices:

A =

1	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	1

B =

1	0	0
0	1	0
0	0	1

Transpose. The mathematical transpose B of matrix A is defined as:

B(j,i)=A(i,j), for i=1,2, . . . ,m and j=1,2, . . . ,n

where the dimensions of A are (m,n) and the dimensions of B are (n,m). The fact that the number of rows of B is equal to the number of columns of A and that the number of columns of B is equal to the number of rows of A is significant, and must be true for the matrix transpose statement that has the following form:

MAT array-name [(rows,columns)] =TRN(array-name)

where *array-name* is a numeric or character-string array and the (rows, columns) option specifies redimensioning. Since no arithmetic is required in the transpose function, the operation applies to both numeric and character-string arrays. Sample transpose statements are:

```
MAT Q=TRN(R)
MAT F(4,3)=TRN(H)
MAT B$=TRN(W$)
```

and the printout of a computer program that uses the transpose is given next under "matrix multiplication." The matrix transpose statement:

```
MAT B=TRN(A)
```


is equivalent to the following nested FOR loops:

```
FOR I=1 TO M
   FOR J=1 TO N
      B(J,I)=A(I,J)
   NEXT J
NEXT I
```

where the M and N are the row and column bounds, respectively, of matrix A and are the column and row bounds, respectively, of matrix B. The same array *cannot* appear on both sides of the equal signs in a matrix transpose statement.

Matrix Multiplication. The multiplication of two matrices A and B is defined as:

$$C(i,j)= \sum_{k=1}^{n} A(i,k)*B(k,j)$$

for i=1,2, . . . ,m and j=1,2, . . . ,p. The dimensions of the matrices are: A(m,n), B(n,p), and C(m,p). The number of columns in matrix A must equal the number of rows in matrix B. For example,

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} * \begin{pmatrix} 5 & 6 \\ 7 & 8 \\ 9 & 10 \end{pmatrix} = \begin{pmatrix} 46 & 52 \\ 109 & 124 \end{pmatrix}$$

The matrix multiplication statement has the form:

MAT matrix-name [(rows,columns)] =matrix-name*matrix-name

where *matrix-name* is a numeric matrix and the (rows, columns) option denotes redimensioning. The matrices specified in the matrix multiplication statement must all be two-dimensional and the same matrix must not appear on both sides of the equals sign; however, the same matrix may appear twice to the right of the equals sign, as follows:

Mat A=A*B	MAT B=A*A
Illegal	*Legal*

The mathematical requirement of conformality of operands also applies to the matrix multiplication statement. As stated above, in a statement of the form:

```
MAT A=B*C
```

the following dimensions must hold:

```
DIM A(M,N),B(M,P),C(P,N)
```

which is summarized as follows:

1. The number of columns in B must equal the number of rows in C.
2. The number of rows in A must equal the number of rows in B.
3. The number of columns in A must equal the number of columns in C.

Moreover, if the above dimensions are true, then the MAT statement of the form:

MAT A=B*C

is equivalent to the following nested FOR loops:

```
FOR I=1 TO M
   FOR J=1 TO N
      S=0
      FOR K=1 TO P
         S=S+B(I,K)*C(K,J)
      NEXT K
      A(I,J)=S
   NEXT J
NEXT I
```

As an example of matrix multiplication, consider the theorem in mathematics that states:

$$(AB)^T = B^T A^T$$

where A and B are matrices and the T denotes transpose. The program in Figure 7.23 gives an example of the theorem.

Matrix Inverse. The inverse of a matrix A is a matrix B that satisfies the following identity:

A*B=B*A=I

where I is the identity matrix. The notion of the inverse of a matrix is easily demonstrated. To compute the inverse of the matrix $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$, a matrix of the form $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is needed such that:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} * \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

After performing the matrix multiplication symbolically and equating elements to the identity matrix, the following simultaneous equations are obtained:

a+3b=1	2a+4b=0
c+3d=0	2c+4d=1


```
0010 DIM A(3,2),B(2,3),C(3,3),D(3,3)
0020 DIM E(2,3),F(3,2),G(3,3)
0030 MAT READ A,B
0040 MAT C=A*B
0050 MAT D=TRN(C)
0060 PRINT FLP,'TRANSPOSE (A*B)'
0070 MAT PRINT FLP,D;
0080 MAT E=TRN(A)
0090 MAT F=TRN(B)
0100 MAT G=F*E
0110 PRINT FLP,'TRANSPOSE(B)*TRANSPOSE(A)'
0120 MAT PRINT FLP,G;
0130 DATA 1,2,3,4,2,1
0140 DATA 3,1,1,2,4,3


TRANSPOSE (A*B)


7      17     8

9      19     6

7      15     5

TRANSPOSE(B)*TRANSPOSE(A)


7      17     8

9      19     6

7      15     5
```

Figure 7.23 An instance of the mathematical theorem $(AB)^T = B^T A^T$ demonstrating matrix transpose and matrix multiplication.

The solution to the simultaneous equations is a=−2, b=1, c=1.5, and d=−.5, so that the inverse matrix is $\begin{pmatrix} -2 & 1 \\ 1.5 & -.5 \end{pmatrix}$.* The form of the matrix inverse statement is:

MAT matrix-name [(rows,columns)]=INV(matrix-name)

where *matrix-name* is a numeric square** matrix, and the (rows, columns) option denotes redimensioning. In the execution of the matrix inverse statement, the

*The reader can verify that $\begin{pmatrix} -2 & 1 \\ 1.5 & -.5 \end{pmatrix} * \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

**In a square matrix, the number of rows equals the number of columns.

inverse is taken of the matrix to the right of the equals sign, which must be non-singular,* and the inverse is assigned to the matrix to the left of the equals sign. The matrix inverse is frequently used in the solution of simultaneous linear equations. For example, consider the system of equations:

$$x_1+x_2+2x_3=3$$
$$x_1+2x_2+3x_3=4$$
$$x_1-x_2-x_3=2$$

If $A=\begin{pmatrix}1 & 1 & 2\\ 1 & 2 & 3\\ 1 & -1 & -1\end{pmatrix}$, $X=\begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix}$, and $B=\begin{pmatrix}3\\ 4\\ 2\end{pmatrix}$, then the system of equations can be expressed as:

$$AX=B$$

Multiplying each side of the matrix equation by the inverse of A (expressed as A^{-1}) and simplifying as follows:

$$A^{-1}AX=A^{-1}B$$
$$IX=A^{-1}B$$
$$X=A^{-1}B$$

the solution X is obtained. Figure 7.24 gives a BASIC program that solves the system of equations that has the following solution: $X=\begin{pmatrix}3\\ 2\\ -1\end{pmatrix}$.

User-Oriented Input and Output

User-oriented array input and output facilities closely parallel those given earlier for single data values. Four statements are involved: READ, INPUT, PRINT, and PRINT USING. The matrix forms of the statements are prefixed with the keyword MAT.

The MAT READ Statement. The MAT READ statement is used to read data values from the internal data set created from DATA statements and assigns those values to specified arrays. The statement has the following form:

```
MAT READ array-name[(rows[,columns])][,array-name[(rows[,columns])]] ...
```

*A matrix is singular if its determinant is zero. This is determined with the DET function. Therefore, a matrix A is non-singular if DET(A)=0. The DET function can be used on matrices up to 50X50; and a determinant is considered to be zero if its value is 1E-20 or less.


```
0010 DIM A(3,3),B(3,1),X(3,1),Q(3,3)
0020 MAT READ A,B
0030 IF DET(A)=0 GOTO 0090
0040 MAT Q=INV(A)
0050 MAT X=Q*B
0060 MAT PRINT X;
0070 STOP
0080 PRINT 'A IS SINGULAR'
0090 DATA 1,1,2,1,2,3,1,-1,-1
0100 DATA 3,4,2
RUN

 3.000000

 2.000000

 1.000000



READY                                                   27818
```

Figure 7.24 Solution to a system of simultaneous linear equations demonstrating the use of the matrix inverse.

where *array-name* is a previously declared numeric or character-string array that may have one or two dimensions and the (rows, columns) option denotes re-dimensioning. If redimensioning is not specified, then the dimension of the specified array is taken from its declaration, and the appropriate number of values are read from the internal data set and assigned to the array in a row-wise order. For example, the statements:

```
DIM R(3,4)
MAT READ R
DATA -7,3,9,6,5,1,4,2,8,-9,0,3
```

would cause the following matrix to be formed in the main storage unit:

$$R=\begin{pmatrix}-7 & 3 & 9 & 6\\ 5 & 1 & 4 & 2\\ 8 & -9 & 0 & 3\end{pmatrix}$$

and the statements:

```
DIM K(15)
MAT READ K
DATA -7,3,9,6,5,1,4,2,8,-9,0,3,-6,7,-1
```

would cause the following one-dimensional array to be constructed:

$$K=(-7\ 3\ 9\ 6\ 5\ 1\ 4\ 2\ 8\ -9\ 0\ 3\ -6\ 7\ -1)$$

The concepts also apply to character-string arrays, as in the following statements:

```
DIM D$(2,3)
MAT READ D$
DATA 'PINTO','VEGA','ARROW'
DATA 'STARFIRE','SKYHAWK','MONZA'
```

that cause the following two-dimensional character-string array to be constructed:

$$D\$ = \begin{pmatrix} \text{'PINTO'} & \text{'VEGA'} & \text{'ARROW'} \\ \text{'STARFIRE'} & \text{'SKYHAWK'} & \text{'MONZA'} \end{pmatrix}$$

Figure 7.25 contains several examples of MAT READ statements as well as corresponding MAT PRINT statements, covered below.

```
0010 DIM C$(15,15),H(20),T(5,12),W(1,1),D$(6)
0020 READ I,J,K,M,N
0030 MAT READ C$(I,J),H(K),T(M,N),W,D$
0040 MAT PRINT FLP,C$,H;T;W;D$;
0050 DATA 2,3,7,4,3
0060 DATA 'PINTO','VEGA','ARROW','STARFIRE','SKYHAWK','MONZA'
0070 DATA -7,3,9,6,5,1,4
0080 DATA 1,1,2,3,5,8,13,21,34,55,89,144
0090 DATA -713.4385
0100 DATA 'A','B','C','D','E','G'
```

```
PINTO                VEGA                 ARROW

STARFIRE             SKYHAWK              MONZA

-7      3      9      6      5      1      4

 1      1      2

 3      5      8

 13     21     34

 55     89     144

-713.4385

ABCDEG
```

Figure 7.25 Examples of the use of the MAT READ and MAT PRINT statements.

The MAT INPUT Statement. The function MAT INPUT statement is identical to the MAT READ statement, except that input is requested from the keyboard instead of being read from the internal data set. The form of the MAT INPUT statement is:

MAT INPUT array-name [(rows[, columns])] [,array-name [(rows[, columns])]] . . .

where *array-name* is a previously declared numeric or character-string array, having either one or two dimensions, and the (rows, columns) option denotes redimensioning. If redimensioning is not specified, then the dimension of the specified array is taken from its declaration, and the appropriate number of values are requested from the keyboard and assigned to the array in a row-wise order. An example of a valid MAT INPUT statement is:

MAT INPUT Q,R(2,N+5)

When the MAT INPUT statement is executed, the user at the keyboard is prompted with a question mark. Values are placed into the input line separated by commas. As each line is filled, it is entered into the computer by pressing the EXECUTE key. If the array is not filled, then the question mark is displayed again. This process is continued until all arrays specified in the MAT INPUT statement have been assigned values. Moreover, the input is assigned successively; after one array is filled, the next value entered is assigned to the next array in the input list. Excess values, after the last array in the list has been filled, are ignored. All values entered must match the corresponding type of variable in the input list. Figure 7.26 includes examples of matrix input.

The MAT PRINT Statement. The MAT PRINT statement is used to print or display a complete array without referring to specific array elements. The statement has the following form:

MAT PRINT [file-ref,] array-name [{; or ,} array-name] . . . [{; or ,}]

where *file-ref* can be FLP for the printer or the designations FL0 through FL9 for tape files 0 through 9.* *Array-name* is a previously declared one or two-dimensional array that contains either numeric or character-string elements. If the file reference is omitted, then the arrays are displayed on the display screen.

*If a file is specified in a MAT PRINT statement, the file must be opened before the statement is executed.

```
0010 DIM A(4),B$(3),C(2,1)
0020 MAT INPUT A,B$,C
0025 PRINT FLP,TAB(10),'OUTPUT'
0030 MAT PRINT FLP,A;B$,C;
RUN
-7,3,9,6
'BOLT','HAMMER','WRENCH'
3.14,2.72
```

0020

(A) Program and input

```
          OUTPUT
          ------


-7       3       9       6


BOLT               HAMMER              WRENCH


 3.14

 2.72
```

(B) Output

Figure 7.26 Matrix input.

An example of valid MAT PRINT statements are:

MAT PRINT FLP, A;B;C
MAT PRINT W;

Each array is printed or displayed by rows with each row starting on a new line. The first row is preceded by two blank lines and succeeding rows are separated from the preceding row by one blank line. If the array reference is followed by a comma,* the array elements are printed or displayed using full print zones. If the array reference is followed by a semicolon, the array elements are printed or displayed using packed print zones. Values are displayed using the same formatting conventions as were given for the PRINT statement. Examples of the use of the MAT PRINT statement were given in Figure 7.25.

The MAT PRINT USING Statement. The MAT PRINT USING statement is used to print or display complete arrays using a specified line image. The form of the line is the same as with the PRINT USING statement. The form of the

*For the last array in an output list, a blank character following the array name is equivalent to a comma.

MAT PRINT statement is:

MAT PRINT USING [file-ref,] statement-number, array-name [{;,}[array-name]] . . . [{;,}]

where:

file-ref is FLP or FL0 through FL9, as specified above,
statement-number is the statement number of the corresponding line image statement, and
array-name is a previously declared one or two-dimensional array that contains either numeric or character-string elements.

An example of a valid print using statement is:

100 PRINT USING 101,A,B
101 : ####.## #.#####| | | |

Each array reference is edited and then printed or displayed in row order according to the specified line image. As with the MAT PRINT statement, the first row of each array begins on a new line, preceded by two blank lines. Each succeeding row begins on a new line and is separated from the preceding row by one blank line. The beginning of each row is printed or displayed according to the start of the line image. If the line image contains more format specifications than the number of elements in the row, then the excess format specifications are ignored. If the number of format specifications is less than the number of elements in the row, then the spacing is controlled by the delimiter following the array reference, as follows:

1. If the delimiter is a comma or a blank and the end of the line image is reached, the current line is printed or displayed and output continues on a line with the start of the line image.
2. If the delimiter is a semicolon and the end of the line image is reached, the output continues on the same line with the start of the line image.

Figure 7.27 gives several examples of the use of the MAT PRINT USING statement. The last row of the last array in a MAT PRINT USING statement, as demonstrated in Figure 7.28, requires special attention. If the trailing delimiter is a comma or a blank character, the line containing the last row is printed or displayed so that the next output will begin on a new line. If the trailing delimiter is a semicolon, then the current line is not printed or displayed so that the next output will be on the same line. The concept is analogous to that of ending a simple PRINT statement with a semicolon.

```
0010 DIM A(5),B(4,3),C(2,6),D$(4)
0020 MAT READ A,B,C,D$
0030 MAT PRINT USING FLP,0040,A
0040 :  ###.##   ####.##
0050 MAT PRINT USING FLP,0060,B,C
0060 :  ###   ####.##   .####||||   #.##||||
0070 MAT PRINT USING FLP,0060,B;C;
0080 MAT PRINT USING FLP,0090,D$
0090 :  #####   #####   #####   #####   #####
0100 DATA 1.23,2.34,3.45,4.56,5.67
0110 DATA 1,2,3,4,5,6,7,8,9,10,11,12
0120 DATA 10,20,30,40,50,60,70,80,90,100,110,120
0130 DATA 'ABLE','BAKER','CHARLY','DAWG'
```

```
   1.23      2.34
   3.45      4.56
   5.67

  1      2.00    .3000E+01

  4      5.00    .6000E+01

  7      8.00    .9000E+01

 10     11.00    .1200E+02

 10     20.00    .3000E+02    4.00E+01
 50     60.00

 70     80.00    .9000E+02    1.00E+02
110    120.00

  1      2.00    .3000E+01

  4      5.00    .6000E+01

  7      8.00    .9000E+01

 10     11.00    .1200E+02

 10     20.00    .3000E+02    4.00E+01    50      60.00

 70     80.00    .9000E+02    1.00E+02   110     120.00

ABLE    BAKER    CHARL    DAWG
```

Figure 7.27 Examples of the use of the MAT PRINT USING statement.

File-Oriented Input and Output

The facilities for file-oriented input and output of complete arrays closely resembles those given earlier for single data values. Two statements are involved: GET and PUT. The matrix forms of the statements are prefixed with the key-


```
0010 DIM A(2,3)
0020 MAT READ A
0030 MAT PRINT USING 0040,A;
0040 :  ##  ##  ##
0050 PRINT 'ALL DONE'
0060 DATA 1,2,3,4,5,6
RUN

   1   2   3

   4   5   6ALL DONE


READY                                    28172
```

Figure 7.28 If the MAT PRINT USING a statement contains a trailing semicolon, then the next output is printed or displayed on the last line of array output.

word MAT. All files referenced with the MAT GET and MAT PUT statements require the use of OPEN and CLOSE statements, as previously introduced, and Input and output processing is the same—except for the fact that entire arrays are being transmitted instead of single values.

The MAT GET Statement. The MAT GET statement is used to read data values from the specified file and assign them in row order to the specified array. The statement has the following form:

MAT GET file-ref,array-name[(rows[,columns])] [,array-name[(rows[,columns])]] . . .[EOF statement-number]

where *file-ref, array-name,* and *(rows,columns)* have the same definitions as given previously. The *EOF statement-number* option specifies the statement number to which program control should be directed if the values in the data file are exhausted before the input list is satisfied. An example of a valid MAT GET statement is:

MAT GET FL2,H,K(15,25)

When the MAT GET statement is executed, data values are read from the specified file until the declared or redimensioned size of the specified array is satisfied. Figure 7.29 demonstrates the case wherein an array is written to a file as single data values and read back as a complete array. The fact that a data file exists as a list of discrete data values is clearly evident with the input and output of complete arrays.

The MAT PUT Statement. The MAT PUT statement is used to write a complete array to a specified data file. The elements of the array are written in row order and exist in the data file as a list of discrete values. The statement has the

```
0010 DIM A(3,4)
0020 MAT READ A
0030 OPEN FL8,'E80',003,OUT
0040 FOR I=1 TO 3
0050 FOR J=1 TO 4
0060 PUT FL8,A(I,J)
0070 NEXT J
0080 NEXT I
0090 CLOSE FL8
0100 MAT A=(0)
0110 OPEN FL8,'E80',003,IN
0120 MAT GET FL8,A
0130 MAT PRINT A;
0140 CLOSE FL8
0150 DATA 1,2,3,4,5,6,7,8,9,10,11,12
RUN

 1      2      3      4

 5      6      7      8

 9      10     11     12




 READY                                    27317
```

Figure 7.29 Example of the use of the MAT GET statement in which an array is written to a file as single data values and read back as an array.

following form:

MAT PUT file-ref, array-name [,array-name] . . .

where *file-ref* and *array-name* have the same definitions as given previously. An example of a valid MAT PUT statement is:

MAT PUT FL4,U,V,W

Data files are written so that the first value written with a MAT PUT statement is the first value read by a subsequent MAT PUT (or GET) statement. This case is demonstrated in Figure 7.30 that gives the combined use of MAT PUT and MAT GET statements.

7.5 PROGRAM CHAINING AND COMMON STORAGE

Program chaining is a computer facility that permits one program to call another program, and common storage is a special area in the main storage unit that is


```
0010 DIM V(5),M(4,4)
0020 MAT V=(1)
0030 MAT M=(25)
0040 PRINT FLP,'VECTOR - MATRIX'
0050 MAT PRINT FLP,V;M;
0060 OPEN FL5,'E80',002,OUT
0070 MAT PUT FL5,V,M
0080 CLOSE FL5
0090 OPEN FL5,'E80',002,IN
0100 MAT GET FL5,M,V
0110 PRINT FLP,'MATRIX - VECTOR'
0120 MAT PRINT FLP,M;V;
0130 CLOSE FL5


VECTOR - MATRIX


 1      1      1      1      1

 25     25     25     25
 25     25     25     25
 25     25     25     25
 25     25     25     25


MATRIX - VECTOR


 1      1      1      1
 1      25     25     25
 25     25     25     25
 25     25     25     25

 25     25     25     25     25
```

Figure 7.30 Example of the use of MAT PUT and MAT GET statements. The first value written to a data file with the MAT PUT statement is the first value read in a subsequent MAT GET (or GET) statement.

effectively used to exchange data betweeen programs that are executed successively. The need for program chaining and common storage facilities is a direct consequence of the fact that the effective use of the main storage unit is dependent upon the size of both programs and data.