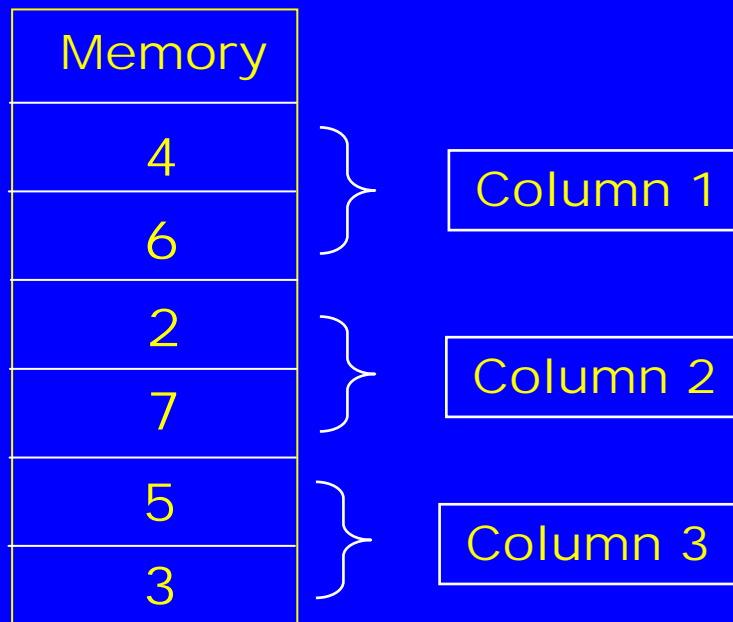


# TWO-DIMENSIONAL ARRAYS

Example:

4	2	5
6	7	3



# Two Dimensional Array Declaration

## Explicit type declaration

```
INTEGER ID(3, 3)
```

```
REAL MSR(100, 100), Z(4:7, 8)
```

```
CHARACTER WORD(5, 5)*3
```

```
LOGICAL TF(5, 7)
```

## Implicit type declaration

```
DIMENSION ALIST(10, 5), KIT(-3:5, 6), XYZ(15, 4)
```

```
INTEGER XYZ
```

```
REAL BLIST(7, 8), KIT
```

# Two Dimensional Array Initialization

- **processing the array column-wise**
  - process the 1st column, then the 2nd column , - - -
  
- **processing the array row-wise**
  - process the 1st row, then the 2nd row , - - -
  
- **Initialization Using the Assignment Statement**
  
- **Initialization Using the READ Statement**

# Two Dimensional Array Initialization

## Initialization Using the Assignment Statement

Example 1:

Declare an integer array ID consisting of 3 rows and 3 columns and initialize array ID row-wise as an identity matrix (i.e. all elements of the main diagonal must be 1 and the rest of the elements must be 0).

Solution:

```
      INTEGER ID(3, 3), ROW, COL
C     INITIALIZING ROW-WISE
      DO 5 ROW = 1, 3
          DO 5 COL = 1, 3
              IF (ROW .EQ. COL) THEN
                  ID(ROW, COL) = 1
              ELSE
                  ID(ROW, COL) = 0
              ENDIF
          5 CONTINUE
```

# Two Dimensional Array Initialization

## Initialization Using the Assignment Statement

Example 2:

Declare a real array X consisting of 2 rows and 3 columns and initialize array X column-wise. Each element of array X should be initialized to its row number.

Solution:

```
REAL X(2, 3)
INTEGER J, K
C  INITIALIZING COLUMN-WISE
DO 5 J = 1, 3
    DO 5 K = 1, 2
        X(K, J) = K
5  CONTINUE
```

# Two Dimensional Array Initialization

## Initialization Using the READ Statement

Example 1: Read all the elements of an integer array MATRIX of size 3X3 column-wise The input data is given as follows:

```
3  4  8
5  9  2
1  6  0
```

The contents of array MATRIX after reading the input data is as follows:

```
3  5  1
4  9  6
8  2  0
```

Solution 1: (Without Array Subscripts)

```
C      INTEGER MATRIX(3, 3)
C      READING COLUMN-WISE
C      READ*, MATRIX
```

# Two Dimensional Array Initialization

## Initialization Using the READ Statement

Solution 2: (Using Implied Loops)

```
      INTEGER MATRIX(3, 3), J, K
C     READING COLUMN-WISE
      READ*, ((MATRIX(K, J), K = 1, 3), J = 1, 3)
```

Solution 3: (Using DO and Implied Loop)

```
      INTEGER MATRIX(3, 3), J, K
C     READING COLUMN-WISE
      DO 28 J = 1, 3
          READ*, (MATRIX(K, J), K = 1, 3)
28    CONTINUE
```

# Printing Two-Dimensional Arrays

Example: Read a 3X3 integer array WHT column-wise and print:

- i. the entire array row-wise in one line;
- ii. the entire array column-wise in one line;
- iii. one row per line;
- iv. one column per line;
- v. the sum of column 3;

If the input is as follows:

```
5 , 2 , 0
3 , 1 , 8
4 , 6 , 7
```

The contents of WHT after reading are as follows:

```
5      3      4
2      1      6
0      8      7
```



## Solution:

```
INTEGER WHT(3, 3), SUM, J, K
READ*, WHT
PRINT*, 'PRINTING THE ENTIRE ARRAY ROW-WISE'
PRINT*, ((WHT(K, J), J = 1, 3), K = 1, 3)
PRINT*, 'PRINTING THE ENTIRE ARRAY COLUMN-WISE'
PRINT*, WHT
PRINT*, 'PRINTING ONE ROW PER LINE'
DO 35 K = 1, 3
    PRINT*, (WHT(K, J), J = 1, 3)
35 CONTINUE
PRINT*, 'PRINTING ONE COLUMN PER LINE'
DO 45 J = 1, 3
    PRINT*, (WHT(K, J), K = 1, 3)
45 CONTINUE
SUM = 0
DO 55 K = 1, 3
    SUM = SUM + WHT(K, 3)
55 CONTINUE
PRINT*, 'SUM OF COLUMN 3 IS', SUM
END
```

The output of the program is as follows :

PRINTING THE ENTIRE ARRAY ROW-WISE

5	3	4	2	1	6	0	8	7
---	---	---	---	---	---	---	---	---

PRINTING THE ENTIRE ARRAY COLUMN-WISE

5	2	0	3	1	8	4	6	7
---	---	---	---	---	---	---	---	---

PRINTING ONE ROW PER LINE

5	3	4
2	1	6
0	8	7

PRINTING ONE COLUMN PER LINE

5	2	0
3	1	8
4	6	7

SUM OF COLUMN 3 IS 17

## Example:

Write a FORTRAN program that reads a two-dimensional array of size 3X3 row-wise. The program finds the minimum element in the array and changes each element of the array by subtracting the minimum from each element. Print the updated array row-wise in one output line.

## Solution:

```
      INTEGER A(3, 3), MIN, J, K
      READ*, ((A(K, J), J = 1, 3), K = 1, 3)
      MIN = A(1, 1)
      DO 3 K = 1, 3
        DO 3 J = 1, 3
          IF (A(K, J) .LT. MIN) THEN
            MIN = A(K, J)
          ENDIF
3     CONTINUE
      DO 4 K = 1, 3
        DO 4 J = 1, 3
          A(K, J) = A(K, J) - MIN
4     CONTINUE
      PRINT*, ((A(K, J), J = 1, 3), K = 1, 3)
      END
```

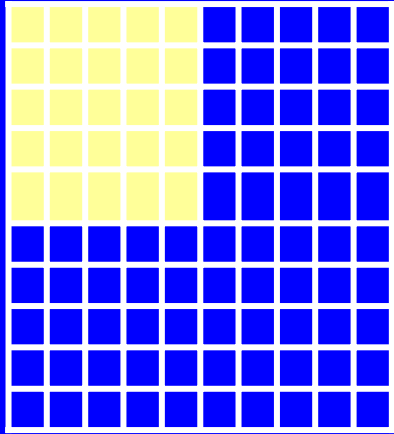
# Two-Dimensional Arrays and Subprograms

Example 1: Counting Zero Elements: Read a 3X2 integer array MAT row-wise. Using a function COUNT, count the number of elements in MAT with the value equal to 0.

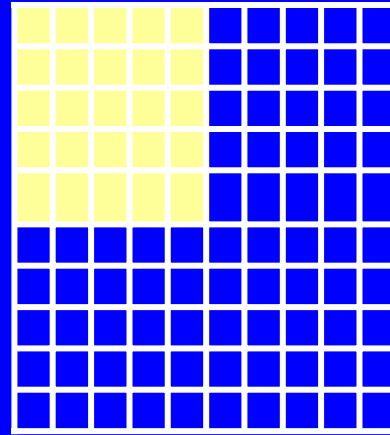
Solution:

```
C  MAIN PROGRAM
  INTEGER MAT(3, 2), COUNT, J, K
C  READING ARRAY MAT ROW-WISE
  READ*, ((MAT(K, J), J = 1, 2), K = 1, 3)
  PRINT*, 'COUNT OF ELEMENTS WITH VALUE 0 IS', COUNT (MAT)
  END
C  FUNCTION SUBPROGRAM
  INTEGER FUNCTION COUNT(MAT)
  INTEGER MAT(3, 2), J, K
  COUNT = 0
  DO 77 K = 1, 3
    DO 77 J = 1, 2
      IF(MAT(K, J) .EQ. 0) COUNT = COUNT + 1
77  CONTINUE
  RETURN
  END
```

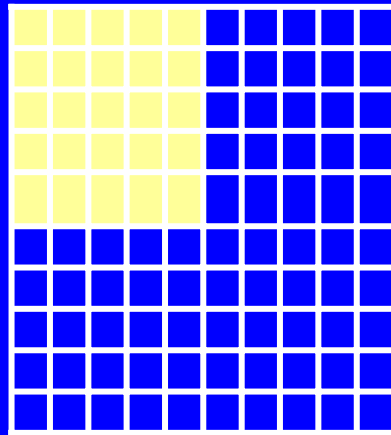
A



B



C



# Two-Dimensional Arrays and Subprograms

Example 2: Addition of Matrices: Write a subroutine CALC(A, B, C, N) that receives 2 two-dimensional arrays A and B of size 10X10. It returns the result of adding the two arrays (matrices) in another array C of the same size.

Solution:

```
C  SUBROUTINE SUBPROGRAM
  SUBROUTINE CALC(A, B, C, N)
  INTEGER A(10, 10), B(10, 10), C(10, 10), N, J, K
  DO 10 K = 1, N
    DO 10 J = 1, N
      C(K, J) = A(K, J) + B(K, J)
10  CONTINUE
  RETURN
  END
C  MAIN PROGRAM
  INTEGER A(10, 10), B(10, 10), C(10, 10), N, J, K
  READ*, N
C  READING ARRAY A ROW-WISE
  READ*, ((A(K, J), J = 1, N), K = 1, N)
C  READING ARRAY B COLUMN-WISE
  READ*, ((B(K, J), K = 1, N), J = 1, N)
  CALL CALC(A, B, C, N)
  DO 10 K = 1, N
    PRINT*, (C(K, J), J = 1, N)
10  CONTINUE
  END
```

# Exercises

```
INTEGER X (3, 3), J
READ*, X
PRINT*, X
PRINT*, (X (J, J), J = 1, 3)
PRINT*, (X (J, 3), J = 1, 3)
END
```

Assume the input is:

1	5	7
7	5	1
3	8	9

The output

1	5	7	7	5	1	3	8	9
1	5	9						
3	8	9						

```

INTEGER A (3, 3), J, K
READ*, ((A (K, J), K = 1, 3), J = 1, 3)
PRINT*, A
PRINT*, ( (A(K, J), J = 1, 2), K = 1, 3)
PRINT*, A(3, 2)
PRINT*, (A (K, 2), K = 3, 1, -2)
END

```

Assume the input is:

1	2	3	
4			
5	6	7	8
9			

The output

1	2	3	4	5	6	7	8	9
1	4	2	5	3	6			
6								
6	4							