Matrix

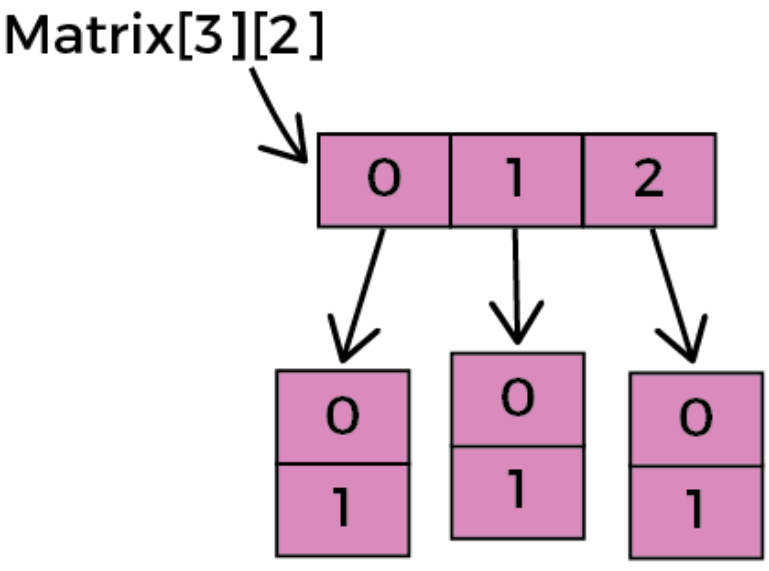
A two-dimensional array is an one-dimensional array, where in each case we have an one-dimensional array.

A two-dimensional array may be ragged (its arrays may all be of differing lengths), but we most often work with (for appropriate parameters M and N) M-by-N two-dimensional arrays that are arrays of M rows, each an array of length N (so it also makes sense to refer to the array as having N columns).

To refer to the entry in row i and column j of a two-dimensional array **m[][]**, we use the notation **m[i][j]**;

to declare a two-dimensional array, we add another pair of square brackets; and to create the array, we specify the number of rows followed by the number of columns after the type name (both within square brackets), as follows:

```
matrix_name : ARRAY_OF type[nb_line][nb_column];
```



example

We refer to such an array as an M-by-N array. By convention, the first dimension is the number of rows and the second is the number of columns. As with one-dimensional arrays. The initialization

of two-dimensional arrays is useful because it masks more code than for one-dimensional arrays.

The following code is initialization of a matrix of integers with zeros :

```
m : ARRAY_OF INTEGER[M][N];
FOR i FROM 0 TO M-1 DO
    FOR j FROM 0 TO N-1 DO
        m[i][j] :=0;
    END_FOR
END_FOR
```

Another example of matrix manipulation, matrix-matrix multiplication

```
a : ARRAY_OF FLOAT[M][N];
b : ARRAY_OF FLOAT[N][O];
c : ARRAY_OF FLOAT[M][O];
FOR i FROM 0 TO M-1 DO
    FOR j FROM 0 TO O-1 DO
        FOR k FROM 0 TO N-1 DO
            // Compute dot product of row i and column j
            c[i][j] := c[i][j] + a[i][k] *b[k][j];
        END_FOR
    END_FOR
END_FOR
```

Previous next