# 11074 - Programación I

División Computación - Departamento de Ciencias Básicas

## ARREGLOS MULTIDIMENSIONALES

# Arreglos bidimensionales

## Matrices

# ¿Qué es una matriz?

En matemática, una **matriz** es un arreglo bidimensional de números.

En el ejemplo, vemos una matriz A de *m* filas y *n* columnas.

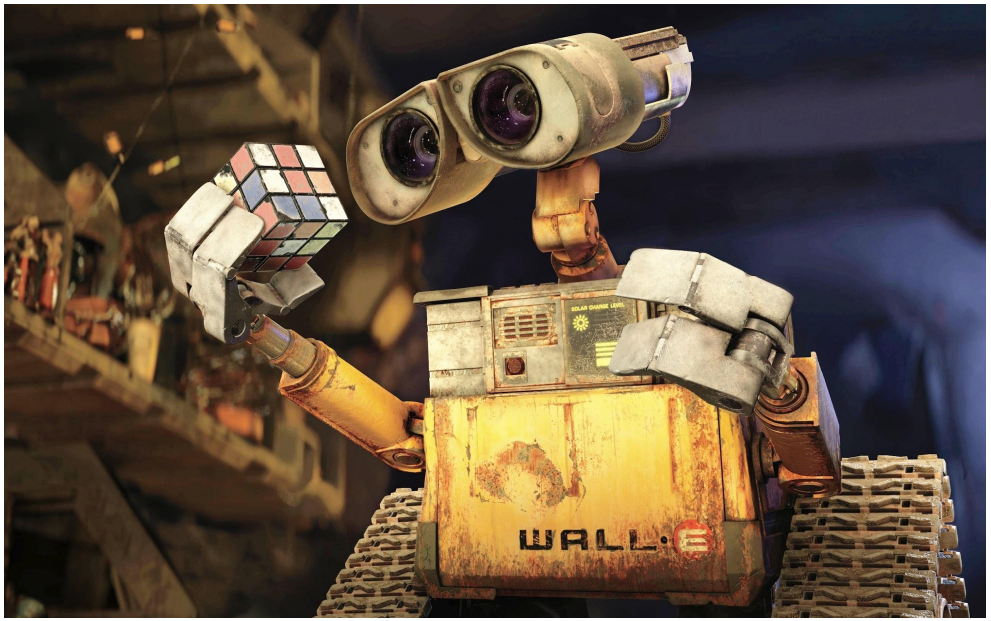
$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

# Aplicaciones de las matrices en computación

Tienen infinidad de aplicaciones.

Ejemplos:

- Procesamiento de imágenes
- Inteligencia artificial
- Robótica
- Grafos
- Base de datos
- Etc, etc.



La cantidad de dimensiones permitidas para un arreglo está acotada por el lenguaje de programación y, básicamente, por el espacio que la estructura ocupa en la memoria.

Un arreglo bidimensional es también llamado **Tabla o Matriz**. Es un array con 2 índices. Estos índices especifican 2 posiciones: el primero para indicar la **fila** y el segundo la **columna**.

También puede pensarse como un vector de vectores. Es un grupo de elementos homogéneo, con un orden interno y en el que se necesitan dos índices para referenciar un único elemento de la estructura.

De acuerdo a la definición, para referenciar un elemento en particular de una matriz es necesario, además del nombre del arreglo, dos valores de índice que indican la posición en fila y columna. Esto es:

Matriz [ pos1] [pos2 ] (pos1 corresponde a la fila y pos2 a la columna)

Ejemplo de cómo se estructura una matriz M de 3 filas y 4 columnas y cómo se identifica a cada uno de sus elementos.

| | | | |
|---|---|---|---|
| M[0,0] | M[0,1] | M[0,2] | M[0,3] |
| M[1,0] | M[1,1] | M[1,2] | M[1,3] |
| M[2,0] | M[2,1] | M[2,2] | M[2,3] |

## Definición

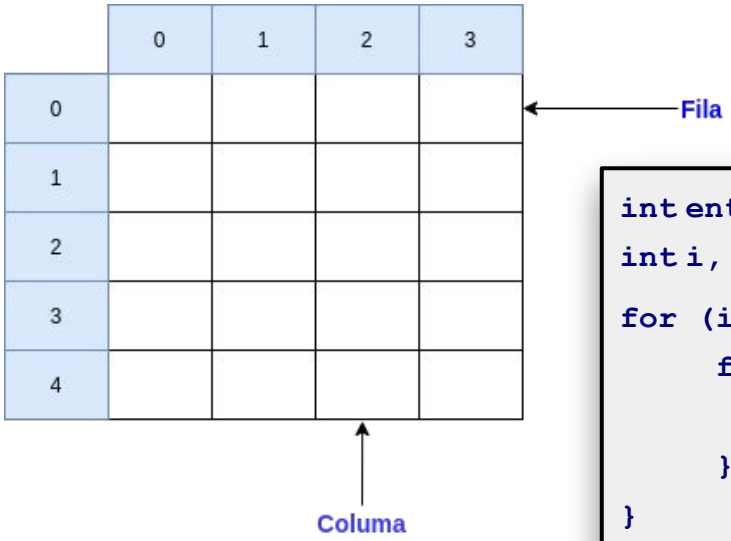
**type nombre [cantidad filas] [cantidad columnas]**

En el ejemplo anterior :

**int M [3] [4];**

# Cargando datos en la matriz

Como las matrices matemáticas, tienen dos dimensiones:




R1:Inicializar una matriz de ***M*** filas por ***N*** columnas con 0

```
int enteros[M][N];
int i, j;
for (i=0; i<M; i++){
     for (j=0; j<N; j++){
          enteros[i][j] = 0;
     }
}
```

*Colección de operaciones:*

- Dimensión (cantidad de filas, y cantidad de columnas)
- Indexación
- Igualdad
- Modificación de un elemento
- Ordenamiento

*Manipulación de Tablas*

Se pueden recorrer por filas o por columnas. Se hace por medio de bucles anidados.

**Cómo** se almacena un arreglo multidimensional en memoria principal, es una cuestión que, lejos de ser compleja, tampoco es trivial. El problema radica en cómo “acomodar” un arreglo multidimensional en memoria, siendo ésta una estructura lineal por construcción. Así, es posible distinguir dos esquemas (que dependen del lenguaje de programación):

- **Orden por filas**(row-mayor order): los elementos de la matriz se ubican por filas en la memoria principal, antes de pasar a la fila siguiente (**C** aplica este esquema).
- **Orden por columnas**(column-mayor order): los elementos de la matriz se ubican por columnas en la memoria principal, antes de pasar a la columna siguiente. (Lenguaje **Fortran** por ejemplo)·

En ocasiones el tipo de orden tiene implicaciones en cuestiones de desempeño (performance) cuando se recorre un arreglo multidimensional. Por ejemplo, en lenguajes como C, recorrer una matriz por columnas y filas podría mostrar peor desempeño que hacerlo por filas y columnas.

# Ejemplo: declara, inicializa y muestra matriz

```c
#include<stdio.h>

int main(void)
{
    // La matriz que quiero es la siguiente:  0 1
    //                                        2 3
    //                                        4 5
    // declaro e inicializo una matriz de enteros de 3 filas y 2 columnas
    int x[3][2] = {{0,1}, {2,3}, {4,5}};
    // forma alternativa de inicializar equivalente a la anterior
    //int x[3][2] = {0,1,2,3,4,5};

    // muestro los elementos de la matriz
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 2; j++)
        {
            printf("Elemento x[%i][%i]: ",i, j);
            printf("%d\n",x[i][j]);
        }
    return (0);
}
```

# Arreglos multidimensionales

Existen, como ya vimos, arreglos de una dimensión:Vector y de dos dimensiones: Matriz. Ahora conoceremos el de tres dimensiones: Tensor o 3D, y al resto de 4 o más ya lo llamaremos n-dimensionales.

. Los arreglos multidimensionales son una ampliación de las matrices de dos dimensiones y utilizan subíndices adicionales para la indexación. Un arreglo 3D, por ejemplo, utiliza tres subíndices. Los dos primeros son como una matriz, pero la tercera dimensión representa páginas u hojas de elementos.

```
// Declaración de un calendario como un arreglo de 3 dimensiones

// en este caso se estarían representando 80 años, cada uno con hasta

// 12 meses y cada mes con hasta 31 días

int calendario[12][31][80];

calendario[3][10][49] = 105; // Aquí le asignamos un 105 al día 11 de abril del año 50
```

Ejemplo de cómo se estructura un tensor T de 3 filas, 4 columnas y 2 páginas y cómo se identifica a cada uno de sus elementos.

| | | | |
|---|---|---|---|
| M[0,0,0] | M[0,1,0] | M[0,2,0] | M[0,3,0] |
| M[1,0,0] | M[1,1,0] | M[1,2,0] | M[1,3,0] |
| M[2,0,0] | M[2,1,0] | M[2,2,0] | M[2,3,0] |

| | | | |
|---|---|---|---|
| M[0,0,1] | M[0,1,1] | M[0,2,1] | M[0,3,1] |
| M[1,0,1] | M[1,1,1] | M[1,2,1] | M[1,3,1] |
| M[2,0,1] | M[2,1,1] | M[2,2,1] | M[2,3,1] |

*Colección de operaciones:*

- Dimensión (cantidad de filas, de columnas y de dimensiones,)
- Indexación
- Igualdad
- Modificación de un elemento
- Ordenamiento

*Manipulación de Tensores*

Se pueden recorrer por filas, por columnas o  por dimensión. Se hace por medio de bucles anidados.

# Ejemplo tensor (array 3d)

```c
#include <stdio.h>

int main(void)
{
    int T [2][3][2];
    int fila, columna, dim;
    for (dim=0; dim<2; dim++) {
        for (fila=0; fila <2; fila++)  {
            for (columna=0; columna <3; columna++) {
            printf("\n Ingrese un numero: ");
            scanf("%d", &T[fila][columna][dim]);
            }
     }
    }
```

```
// muestro los elementos por pantalla
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 2; ++j) {
            for (int k = 0; k < 3; ++k) {
                printf("Elemento T[%i][%i][%i] = %d\n",  j, k, i, T[j][k][i]);
            }
        }
    }
    return (0);
}
```

"Enseñar no es transferir conocimientos, sino crear las posibilidades de su construcción; quien enseña aprende al enseñar y quien aprende enseña al aprender"

**Paulo Freire**