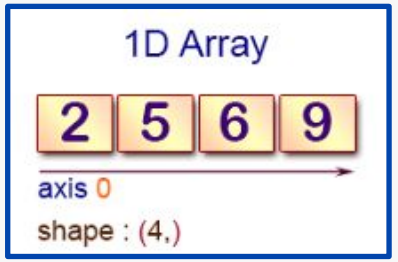
MATERIA: Fundamentos de los Computadores Digitales - Mecatrónica

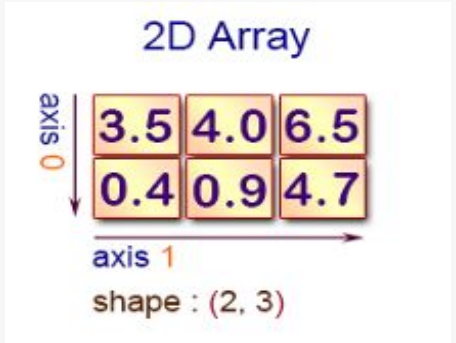
ARRAY

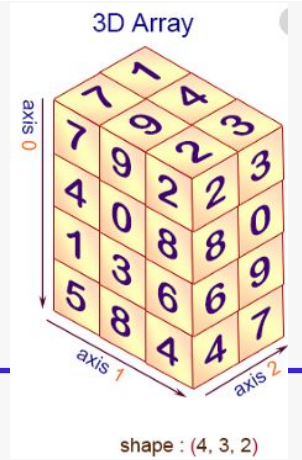
Vectores - matrices - arreglos

TIPOS DE ARREGLOS

- **ARREGLO**
  - **Lineal**
    - Array unidimensional
    - Matriz lineal
    - Vector
  - **No Lineal**
    - Matriz bidimensional
    - Matriz tridimensional
    - Matriz n-dimensional







EL PROBLEMA

Ejercicio: Calcular el promedio de las notas de 3 personas

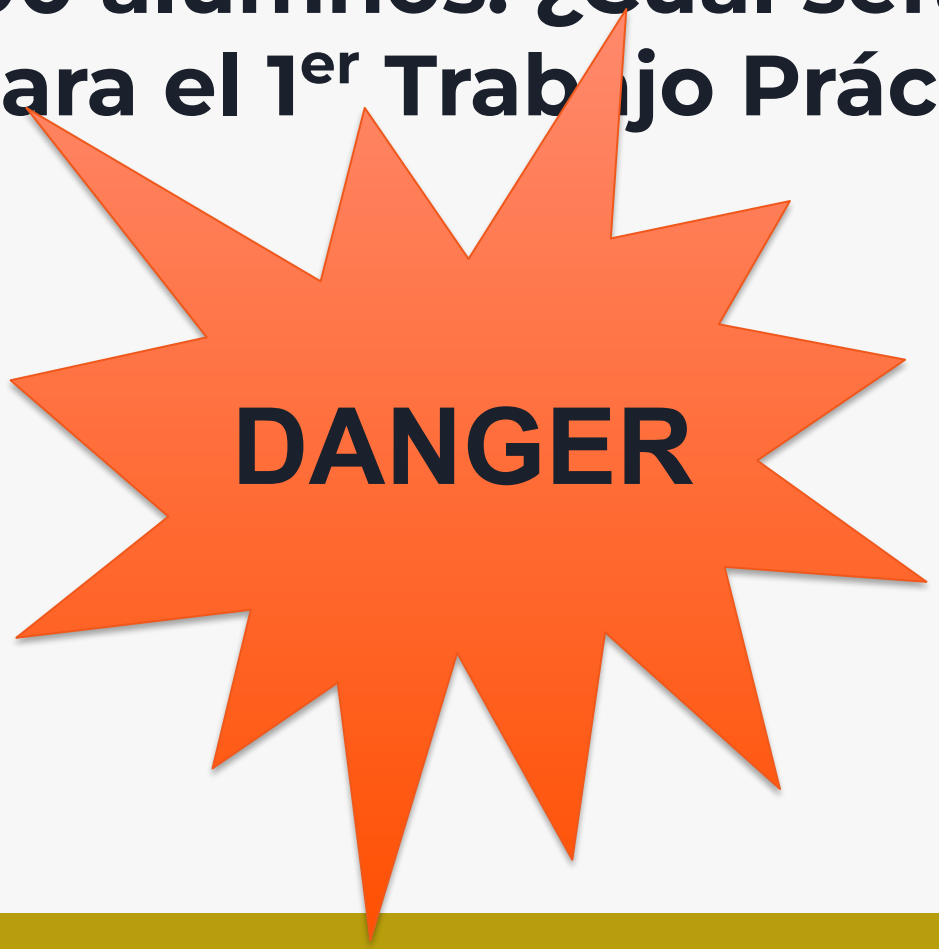
/* ¿Qué variables necesito? */

```
float promedio;
float nota1;
float nota2;
float nota3;
```

Promedio = (nota1 + nota2 + nota3) / 3

Ejercicio: Si tengo 100 alumnos. ¿Cual será el promedio del aula para el 1er Trabajo Práctico

```
float promedio;
float nota1;
float nota2;
float nota3;
 ...
float nota100;
```

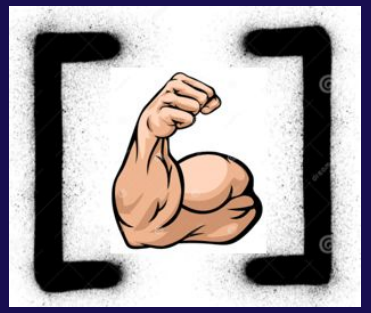


¿Cuántas variables tengo que declarar?

LA SOLUCIÓN

UNLZ Universidad Nacional de Lomas de Zamora

ARREGLOS



```
float nota [100];
```

Tipo de dato

Nombre de la variable

Tamaño

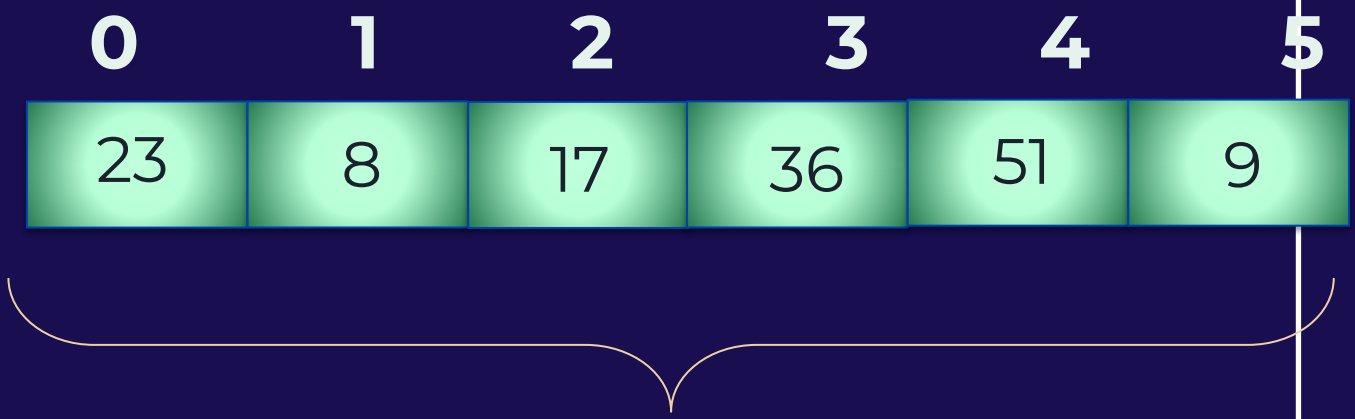
nota1 nota2 nota3
nota4 nota5 nota6
nota7 nota8 nota9
nota10 nota11 nota12
ntoa13 nota14 nota15
nota16 nota 7 nota18
nota19 nota20 nota21
nota22 nota23 nota24
nota25 nota26 nota27
nota28. etc.

...

UNLZ Universidad Nacional de Lomas de Zamora

OPERACIÓN DE ASIGNAR

```
nota [0] = 23;
nota [1]  = 8;
nota [2] = 17;
nota [3] = 36;
nota [4] = 51;
nota [5] = 9;
```




```
int nota [6];
```

Facultad de INGENIERÍA UNLZ

VECTORES EN C

UNLZ
Universidad Nacional
de Lomas de Zamora

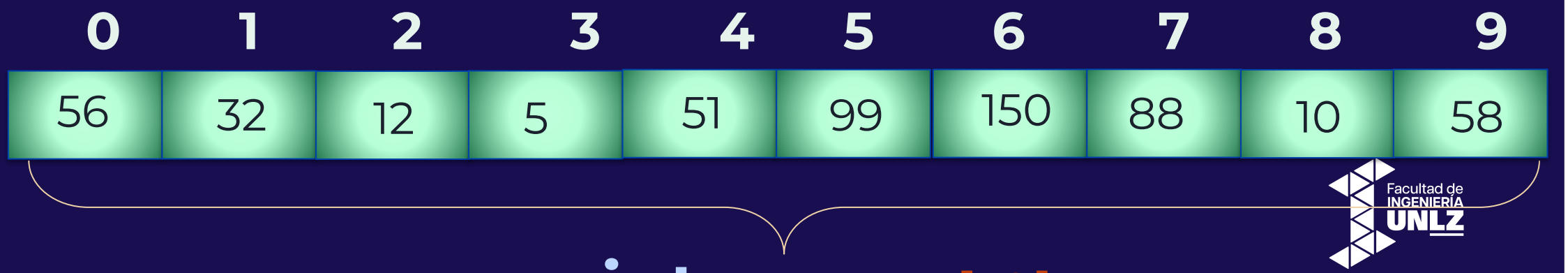
Facultad de
INGENIERÍA
UNLZ

Arreglos – Vectores - Array en C

- Un vector (arreglo de una dimensión) se declara en C usando **[ ]**

```
int v_num[10]; /* declara un vector de enteros, de 10 elementos */
```

- Los subíndices en C comienzan en **0** por lo tanto si quiero mostrar los 10 valores del vector, debería recorrer desde la posición **0** hasta la **9**.



Arreglos – Vectores - Array

- Un vector también puede inicializarse al momento de declararlo

```
int       vec_num[5] = {1, 6, 8, 3, 9};
float     array_num[] = {3.5, 12.2, 7};
double    vec_dou[10] = {0};
```

- Notar que en el caso de **array_num** no se indicó el tamaño explícitamente, pero al iniciarlo con 3 elementos el compilador lo genera de ese tamaño.

- También podemos indicar el tamaño e inicializar una cantidad de valores menor al tamaño, en cuyo caso los valores no inicializados explícitamente, se inicializan, implícitamente, en cero. Por ejemplo el vector **vec_dou** tendrá sus 10 elementos inicializados en cero.
- Otra alternativa es declarar vectores usando constantes para definir su tamaño.

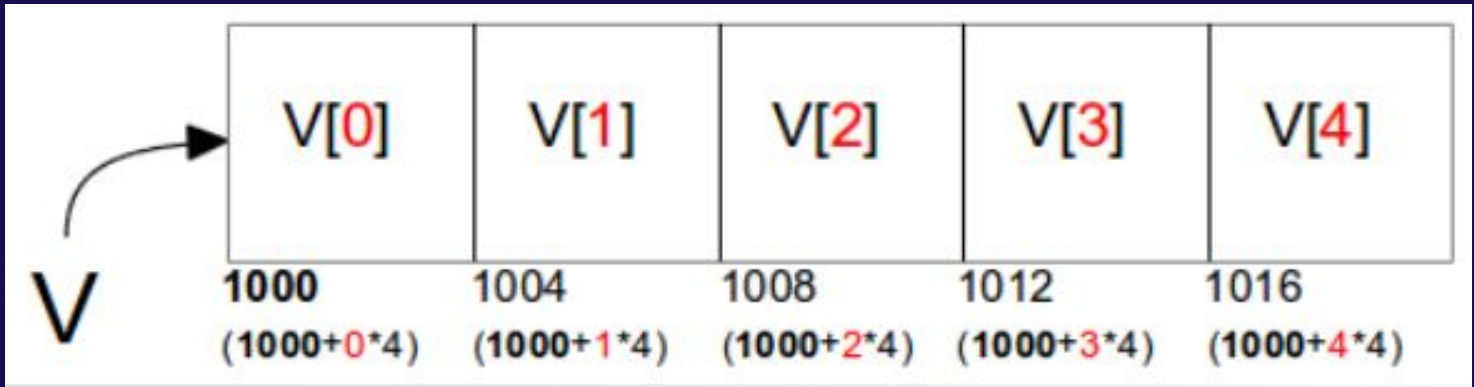
```
#define DIMEN 10

int vec_num [DIMEN];
```

Arreglos según C

Para lenguaje C un arreglo no es más que un sector de memoria contigua con capacidad para guardar la cantidad de elementos declarados.

Así si declaro int v[5] guarda lugar en memoria para almacenar 5 int consecutivos.



Esto explica, porqué el primer subíndice es el cero, ya que el primer elemento es el que está **desplazado** cero "elementos" con respecto al inicio del vector.

Ejemplos para cargar datos en vector

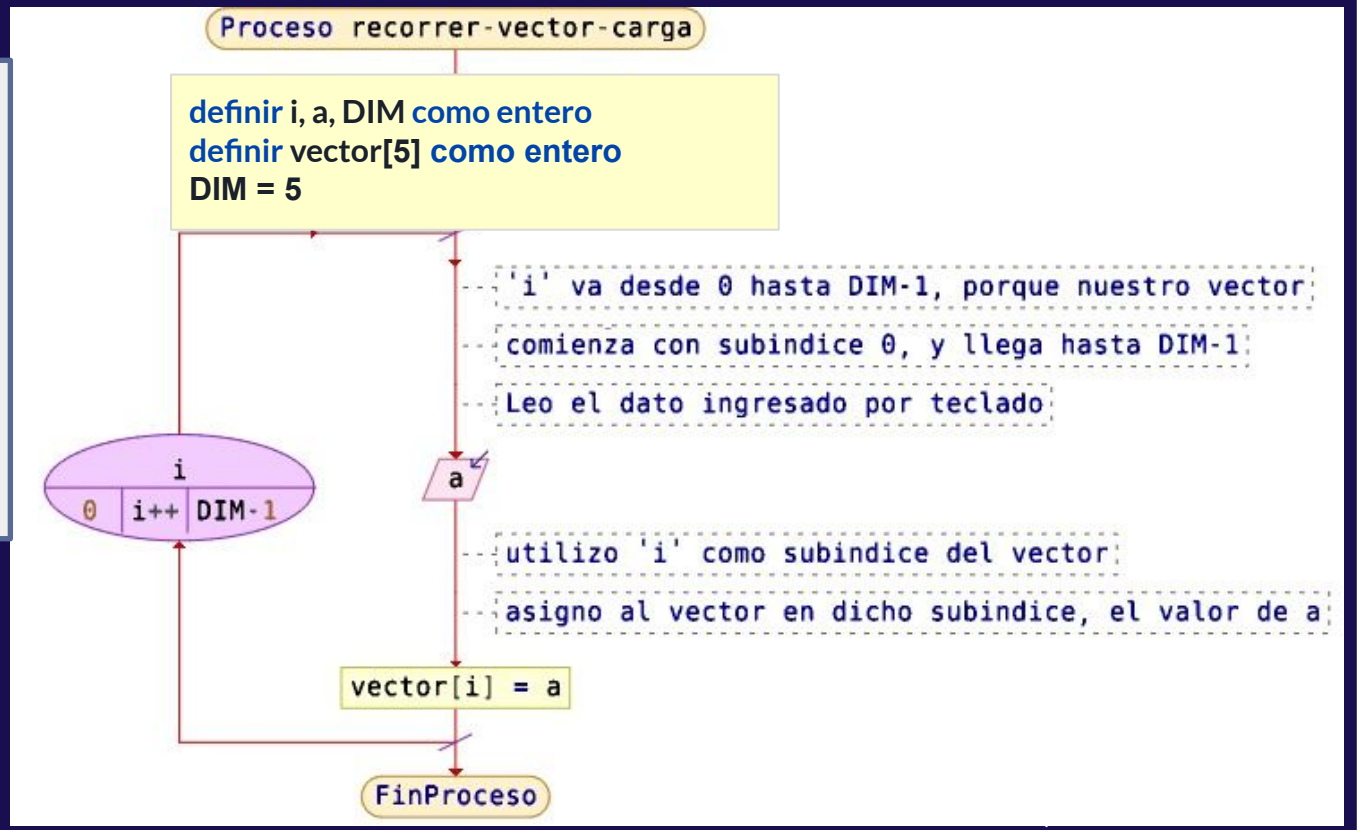
Ejemplo carga de datos en C:

```c
int pos;
int vector[5];

for (pos=0; pos<5; pos++)
{
    printf("Ingresar datos para la posicion %d: ", pos);
    scanf("%d", &vector[pos]);
}
```

Salida:

Ingresar datos para la posicion 0: 20
Ingresar datos para la posicion 1: 25
Ingresar datos para la posicion 2: 36
Ingresar datos para la posicion 3: 98
Ingresar datos para la posicion 4: 100



VER EJEMPLO EN C

Ejemplos para imprimir datos en vector

Ejemplo cargo e imprimir datos en C:

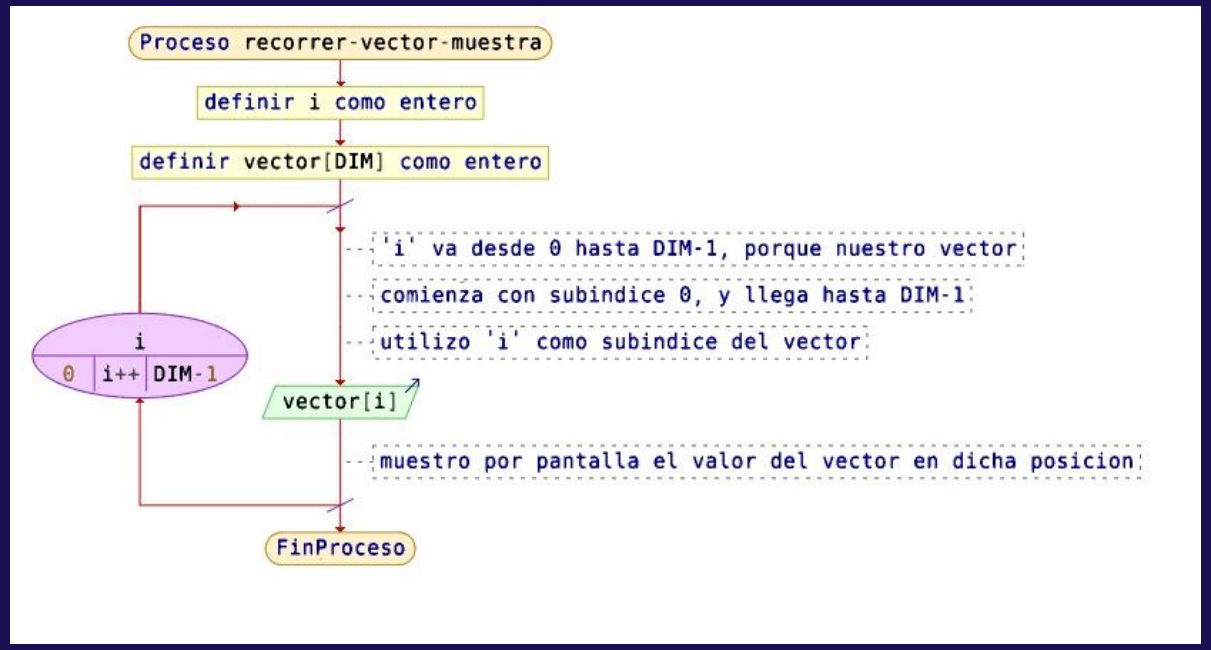
```c
int pos;
int vector[5];

for (pos=0; pos<5; pos++)
{
    printf("Ingresar datos para la posicion %d: ", pos);
    scanf("%d", &vector[pos]);
}
```

```c
for (pos=0; pos<5; pos++)
{
  printf("La posicion: %d tiene el valor: %d \n", pos, vector[pos]);
}
```

Salida:

La posicion: 0 tiene el valor: 20
La posicion: 1 tiene el valor: 25
La posicion: 2 tiene el valor: 36
La posicion: 3 tiene el valor: 98
La posicion: 4 tiene el valor: 100



VER EJEMPLO EN C

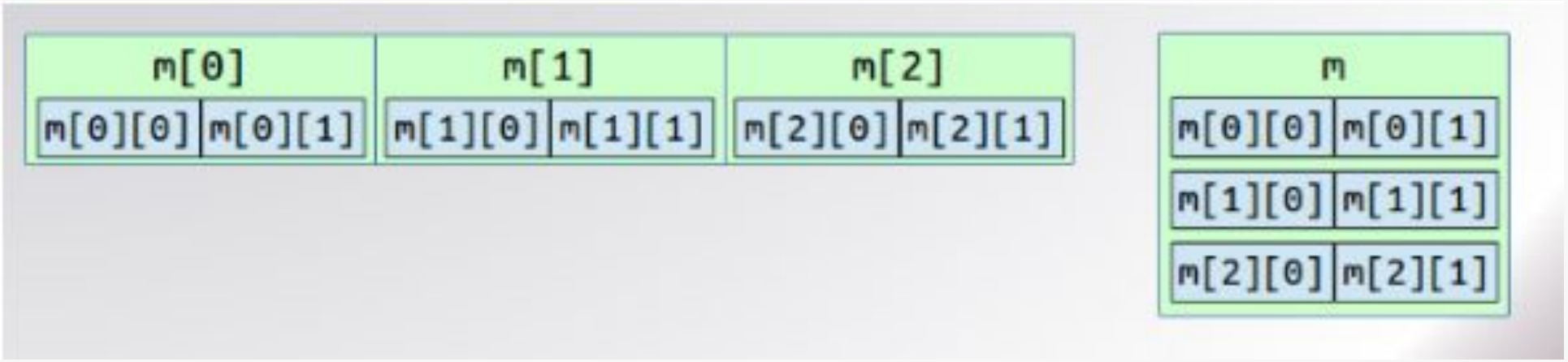
MATRICES

Matrices o array Multidimensional

- No hay matrices como tales, solo vector de vector. Por ejemplo si declaramos

```
double mat [3] [2] ; // mat[fila] [columnas]
```

- Entonces mat es un vector de 3 elementos, donde cada uno de ellos es a su vez un vector de 2 doubles.
- Como cada elemento es un vector de 2 doubles implica que en memoria se guardará la “matriz” por filas (cada fila es un elemento del “vector” mat)

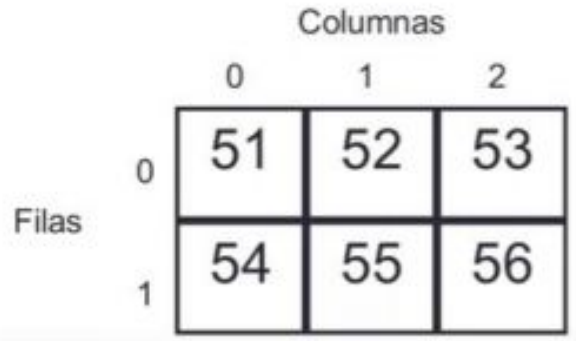


Matrices o array Multidimensional

- **tipoDeDato** nombreDelArray [numeroFilas] [numeroColumnas];

- Ejemplos:
- **char** pantalla[25][80];
- **int** puestos[6][8];

- **int** tabla[2][3] = {51 , 52 , 53 , 54 , 55 , 56};



- Otra opción más amigable:

- **int** tabla[2][3] = { {51, 52, 53} , {54, 55, 56} };

Matrices o array Multidimensional

```
int main(){
    int matriz[2][3] = {51,52,53 , 54,55,56};
    int filas,columnas;

    for(filas=0;filas<2;filas++){
        for(columnas=0;columnas<3;columnas++){
            printf("%i ",matriz[filas][columnas]);
        }
        printf("\n");
    }



    getch();
    return 0;
}
```