U.D.3

Estructuras de almacenamiento

(PARTE 2: MATRICES)

Nota: Ejemplos en lenguaje C

ÍNDICE

7. *Arrays* multidimensionales. Matrices.

- Matriz: Colección de elementos del mismo tipo que se distribuyen en forma de filas y columnas.
  - Una matriz es un vector de vectores.

- Las matrices suelen usarse para relacionar 2 magnitudes.

Ejemplo: Queremos almacenar las notas de los 30 alumnos de una clase en cada uno de los 3 trimestres.

Matrices

Matrices multidimensionales: Relacionan 3 o más magnitudes:

Ejemplo: Almacenar el sueldo base de los empleados de una empresa que depende de su antigüedad (1-30 años), de su categoría (5 categorías posibles) y de su departamento (1-3).

```
float asesueldos[30][5][3];
```

Uso de *arrays* multidimensionales

- *ARRAYS* MULTIDIMENSIONALES:
  - Más de 2 dimensiones.
    - Ejemplo de 3 con curso, sexo y facultad.
  - La mayoría de las aplicaciones no utilizan tablas con más de 3 dimensiones ya que visualizar el esquema es imposible y la representación del problema es fundamental para el programador.
- *ARRAYS* PARALELOS:
  - En muchas ocasiones se requiere el proceso simultáneo de más de una tabla que teniendo el mismo número de elementos, el tipo de datos de los mismos es distinto.
    - Ejemplo: lista de los socios de un club deportivo con los siguientes datos personales: Nombre (tabla de cadenas), Dirección (idem), Edad (de enteros) y teléfono (idem).

Matrices

- Definición:

```
float notas[3][5];
```

- Índices (0..2) (0 … 4)

	0	**1**	**2**	**3**	**4**
0	5.5	6	3	2	5
1	6	8	4.5	3	4.4
2	9.2	9	2	1	6.3

- Acceso a la nota del alumno 4 en el segundo trimestre:

```
printf("%f", nota[1][3]);
```

8. Operaciones con matrices

- Recorrer una matriz es realizar un tratamiento a cada uno de los componentes del *array*.
- Una matriz puede recorrerse por filas o por columnas.
- Inicialización de una matriz:

```
for (i=0; i<NTRIM; i++)
   for (j=0; j<NALUM; j++)
      notas[i][j]=0;
```

Recorrido matriz por filas

- Leer los datos y almacenarlos en una matriz (por filas):

```
for (i=0; i<NTRIM; i++)
{
  for (j=0; j<NALUM; j++)
  {
    printf("Nota alumno %d en el trim. %d"
                , j+1, i+1);
    scanf("%f", & nota[i][j]);
    fflush(stdin);
  }
}
```

Recorrido matriz por columnas

- Escribir los datos de una matriz (por columnas):

```
for (j=0; i<NALUM; j++)
{
  for (i=0; j<NTRIM; i++)
  {
    printf("Nota alumno %d en el trim. %d:
                %d", j+1, i+1, nota[i][j]);
  }
}
```

9. Paso de matrices como argumento a una función

- Un matriz es un *array*, por lo que puede pasarse como argumento a una función.

- Una función nunca puede devolver una matriz.

- Las matrices siempre son pasadas por referencia, es decir, cualquier cambio realizado en los elementos de la matriz será visible al salir de la función.

Matrices como parámetros en C

La cabecera de la función podría ser de estas 2 formas:

```
void inicializar (int matriz[NFILAS][NCOL])
void inicializar (int matriz[ ][NCOL])
```

En la llamada de la función sólo aparecerá el nombre de la matriz:

```
inicializar (matriz);
```

Almacenamiento de una matriz en Lenguaje C

5.5	m[0][0]
6	m[0][1]
3	m[0][2]
2	m[0][3]
5	m[0][4]
6	m[1][0]
8	m[1][1]
4.5	m[1][2]
3	m[1][3]
4.4	m[1][4]

La matriz se almacena en memoria por filas:

	0	**1**	**2**	**3**	**4**
0	5.5	6	3	2	5
1	6	8	4.5	3	4.4

Inicialización de matrices en C

```
int m [3] [4] =
{
  1, 2, 3, 4,
  5, 6, 7, 8,
  9, 10, 11, 12
};
```

```
int m [3] [4] =
{
  { 1, 2, 3, 4 },
  { 5, 6, 7, 8 },
  { 9, 10, 11, 12}
};
```

Casos idénticos.

```
int m1 [3] [2] =
{
  2, 3,
  4,
  5, 6
};
```

```
int m2 [3] [2] =
{
  { 2, 3 },
  { 4 },
  { 5, 6 }
};
```

Casos diferentes.

```
int m [] [4] =
{
  { 1, 2, 3, 4 },
  { 5, 6, 7, 8 }
};
```

No es necesario indicar el índice máximo de la primera dimensión.