Sesión 6: Matrices (2h)

En esta sesión de laboratorio seguiremos trabajando el tema 3 de la asignatura (tipos de datos estructurados) y, en concreto, el tipo de dato *matriz*. En la sesión anterior se trabajó con el tipo de dato *vector* que permitía manipular un conjunto de elementos del mismo tipo como una *tabla unidimensional*. En esta sesión se hace una extensión en el número de dimensiones de este tipo de dato y se muestra como manipular *tablas bidimensionales* o *matrices*. El objetivo de esta sesión es que el alumno se familiarice con las matrices y sea capaz de manipularlas correctamente y de aplicar algoritmos básicos sobre ellas (recorrido por filas y por columnas, suma de matrices, traspuestas, intercambios de filas o columnas, etc.).

Ejercicio 1:

Indique en el fichero *sesion6_ej1.c* cómo se declaran e inicializan en C las siguientes variables, teniendo en cuenta la información que van a almacenar (la inicialización se puede realizar en la propia declaración o con un fragmento de código):

(a) La variable `m1` debe almacenar una matriz de 5 filas y 4 columnas de enteros y debe estar inicializada a 1 en todas sus posiciones.

(b) La variable `sudoku` debe almacenar la información que hay en un tablero de sudoku y debe estar inicializada como muestra la siguiente figura.

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

Las casillas que en la imagen se muestran vacías, deben estar inicializadas a -1. Use la constante `VACIA` (con el valor -1) en la inicialización.

(c) La variable `mat_frac` debe almacenar una matriz de NxN fracciones, donde N es una constante de valor 10. Inicialice esta variable a través de un fragmento de código de forma que la posición `(i,j)` de la matriz esté inicializada al valor `(i+1)/(j+1)`.

Compruebe si las declaraciones anteriores son correctas añadiendo al fichero *sesion6_ej1.c* el código necesario para mostrar por pantalla el valor que almacenan las variables anteriores. En el caso del sudoku, no hay que dibujar el tablero, únicamente se debe mostrar el valor de las casillas (para la casillas vacías se muestra un blanco en lugar de un -1).

Ejemplo de ejecución:

```
Valor de m1:
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1

Valor de sudoku:
5 3     7
6     1 9 5
  9 8         6
8       6       3
4     8   3     1
7       2       6
  6         2 8
      4 1 9     5
        8     7 9

Valor de mat_frac:
 1/ 1  1/ 2  1/ 3  1/ 4  1/ 5  1/ 6  1/ 7  1/ 8  1/ 9  1/10
 2/ 1  2/ 2  2/ 3  2/ 4  2/ 5  2/ 6  2/ 7  2/ 8  2/ 9  2/10
 3/ 1  3/ 2  3/ 3  3/ 4  3/ 5  3/ 6  3/ 7  3/ 8  3/ 9  3/10
 4/ 1  4/ 2  4/ 3  4/ 4  4/ 5  4/ 6  4/ 7  4/ 8  4/ 9  4/10
 5/ 1  5/ 2  5/ 3  5/ 4  5/ 5  5/ 6  5/ 7  5/ 8  5/ 9  5/10
 6/ 1  6/ 2  6/ 3  6/ 4  6/ 5  6/ 6  6/ 7  6/ 8  6/ 9  6/10
 7/ 1  7/ 2  7/ 3  7/ 4  7/ 5  7/ 6  7/ 7  7/ 8  7/ 9  7/10
 8/ 1  8/ 2  8/ 3  8/ 4  8/ 5  8/ 6  8/ 7  8/ 8  8/ 9  8/10
 9/ 1  9/ 2  9/ 3  9/ 4  9/ 5  9/ 6  9/ 7  9/ 8  9/ 9  9/10
10/ 1 10/ 2 10/ 3 10/ 4 10/ 5 10/ 6 10/ 7 10/ 8 10/ 9 10/10
```

Ejercicio 2:

Se desea mantener la información de distancias en kilómetros entre 5 capitales según la tabla siguiente:

	Londres	Madrid	París	Roma	Viena
Londres	0	1720	456	1845	1473
Madrid	1720	0	1272	1965	2399
París	456	1272	0	1468	1280
Roma	1845	1965	1468	0	1130
Viena	1473	2399	1280	1130	0

Para representar dicha tabla en C, se ha utilizado una matriz cuadrada de 5 filas x 5 columnas. Según la tabla anterior cada capital tiene asociado un identificador entre 0 y 4 (Londres el 0, Madrid el 1, Paris el 2, Roma el 3 y Viena el 4).
En la siguiente declaración, la variable `distancias` representa dicha matriz:

```
#define MAX_CAPITALES 5
int distancias[MAX_CAPITALES][MAX_CAPITALES];
```

a) Modificar el fichero *sesion6_ej2a.c* y añadir la información de una capital más. En concreto se debe añadir la información de Berlín (con identificador 5). La distancia en kilómetros entre Berlín y las otras capitales son:

Berlín – Londres: 1093
Berlín – Madrid: 2322
Berlin – París: 1054
Berlín - Roma: 1519
Berlín – Viena: 682

Además, se debe añadir al fichero *sesion6_ej2a.c* el código necesario para que lea del teclado dos identificadores de capitales (valor entre 0 y `MAX_CAPITALES-1`) y muestre por pantalla la distancia entre ambas capitales. El programa debe controlar que los identificadores de capitales están dentro del rango permitido.
Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Introduce el identificador de las dos capitales (id1, id2): 5, 6
Introduce el identificador de las dos capitales (id1, id2): 3, 4

La distancia entre las capitales 3 y 4 es 1130
```

```
Introduce el identificador de las dos capitales (id1, id2): 5, 5

La distancia entre las capitales 5 y 5 es 0
```

```
Introduce el identificador de las dos capitales (id1, id2): 5, 2

La distancia entre las capitales 5 y 2 es 1054
```

b) Se define un camino como una secuencia de identificadores de capitales, donde el primer elemento de la secuencia indica el origen, los siguientes las capitales por las que se pasa y el último el destino. Por ejemplo, un camino con origen en París que pasa por Viena y finaliza en Londres sería la secuencia { 2, 4, 0 }.
Completar el código del fichero *sesion6_ej2b.c* para que lea un camino del teclado y muestre por pantalla el total de kilómetros que tiene. El programa debe controlar que la longitud del camino es menor de `MAX_CAMINO` y que los identificadores de todas las ciudades del camino son válidos.
Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Introduce el numero de ciudades a visitar: 11
Introduce el numero de ciudades a visitar: -1
Introduce el numero de ciudades a visitar: 2
Introduce el id de las ciudades a visitar (id1, id2, ...): 2,3

La distancia del camino es: 1468
```

```
Introduce el numero de ciudades a visitar: 1
Introduce el id de las ciudades a visitar (id1, id2, ...): 2

La distancia del camino es: 0
```

```
Introduce el numero de ciudades a visitar: 3
Introduce el id de las ciudades a visitar (id1, id2, ...): 2,3,5

La distancia del camino es: 2987
```

```
Introduce el numero de ciudades a visitar: 4
Introduce el id de las ciudades a visitar (id1, id2, ...):
3,4,6,1
Introduce el id de las ciudades a visitar (id1, id2, ...):
3,4,2,1

La distancia del camino es: 3682
```

Ejercicio 3:

a) Complete el programa del fichero *sesion6_ej3a.c* para que pida al usuario que introduzca dos matrices de como máximo NxN naturales y las almacene en `m1` y `m2`. El programa debe comprobar que las dimensiones de las matrices están dentro del rango permitido (entre 0 y N-1). A continuación el programa debe comprobar si las dos matrices tienen las mismas dimensiones y, si es así, mostrar por pantalla la matriz resultante después de sumar `m1` y `m2`.

Ejemplos de ejecución (en negrita los datos introducidos por el usuario):

```
Introduzca las dimensiones de la matriz m1 (filas, columnas): 3,3
Introduzca los datos de la matriz m1:
1 2 3
4 5 6
7 8 9

Introduzca las dimensiones de la matriz m2 (filas, columnas): 3,4
Introduzca los datos de la matriz m2:
1 1 1 1
1 1 1 1
1 1 1 1

Matrices con dimensiones diferentes: no se pueden sumar
```

```
Introduzca las dimensiones de la matriz m1 (filas, columnas): -1,4
Introduzca las dimensiones de la matriz m1 (filas, columnas): 3,4
Introduzca los datos de la matriz m1:
1 2 3 4
5 6 7 8
9 10 11 12
Introduzca las dimensiones de la matriz m2 (filas, columnas): 3,4
Introduzca los datos de la matriz m2:
1 1 1 1
1 1 1 1
1 1 1 1

Datos de la matriz suma:
  2   3   4   5
  6   7   8   9
 10  11  12  13
```

b) Complete el programa del fichero *sesion6_ej3b.c* para que determine si la matriz `m1` es simétrica. La matriz `m1` se inicializa en la propia declaración.
Ejemplos de ejecución:

```
Datos de la matriz m1:
  1   2   3
  2   1   4
  3   4   1

La matriz es SIMETRICA
```

```
Datos de la matriz m1:
  1   2   3   0
  2   1   4   0
  3   4   1   0

La matriz NO es SIMETRICA
```

```
Datos de la matriz m1:
  1   2   2
  2   1   4
  3   4   1

La matriz NO es SIMETRICA
```

c) Complete el programa del fichero *sesion6_ej3c.c* para que calcule la traspuesta de la matriz `m1`. La matriz `m1` se inicializa en la propia declaración.
Ejemplos de ejecución:

```
Datos de la matriz m1:
  1   2   2
  2   1   4
  3   4   1
```

```
Datos de la matriz m1:
  1   2   3
  2   1   4
  3   4   1
  0   0   0

Datos de la matriz traspuesta:
  1   2   3   0
  2   1   4   0
  3   4   1   0
```

```
Datos de la matriz m1:
  1   2   3   0
  2   1   4   0
  3   4   1   0

Datos de la matriz traspuesta:
  1   2   3
  2   1   4
  3   4   1
  0   0   0
```