# LENGUAJES DE PROGRAMACIÓN II (ITIG)
# PRÁCTICA 1
Manejo de vectores y reserva de memoria en C (4 a 6 horas)

## Primera parte.

Realizar un programa de aritmética de matrices cuadradas. Este programa debe permitir sumar y multiplicar dos matrices. El máximo tamaño permitido para una matriz será de 10x10 elementos, siendo, por tanto, válidos todos los tamaños inferiores.

El programa no requiere de ningún menú, primero se pedirá el tamaño de las matrices, **n**, a continuación se pedirá la introducción de los elementos de cada matriz, para después realizar la suma y la multiplicación mostrando los resultados correspondientes.

Las matrices se almacenarán en un vector bidimensional de 10x10 elementos. Además, como mínimo se deben realizar cinco funciones: petición del tamaño de las matrices, introducción de datos de la matriz, visualización de una matriz, suma y multiplicación.

La función de *introducción de la matriz* será llamada dos veces, una para cada matriz. Se le pasarán como argumentos la matriz sobre la que se van a introducir los valores y el entero **n**. En las funciones *suma* y *multiplicación*, se pasarán como argumento **n** y tres matrices, los dos operandos y la matriz de resultado. La función de *visualización* sirve para mostrar los resultados de la suma y la multiplicación. Esta función recibirá como argumentos **n** y la matriz que queremos visualizar. La función de petición del valor **n** recibirá como argumento un entero, sobre el que se devolverá el valor del tamaño de la matriz. Por lo tanto, la función será de tipo **void**, pues no necesita devolver ningún valor mediante **return**[1].

No se permite el uso de variables globales, puesto que el programa no las requiere. En todo caso, sólo se permite definir constantes globales. El motivo de esta restricción es que las variables globales dificultan el mantenimiento de los programas, y por lo tanto, conviene acostumbrarse a usarlas lo menos posible. No se deben confundir en *C* las variables que se declaran en la función *main*, las cuales son locales por declararse dentro de una función, con las variables globales, las cuales se declaran fuera de todas las funciones.

## Segunda parte.

Repetir la práctica, pero en este caso usando memoria dinámica. Cada matriz se representará mediante la siguiente estructura:

```
struct matriz
{
    int tam;
    int *vector;
};
```

El campo **tam** representa el tamaño de la matriz (las matrices siguen siendo cuadradas), y el campo **vector** apuntará a un vector unidimensional donde se almacenarán consecutivamente las filas de la matriz. La memoria del vector deberá reservarse y liberarse empleando las funciones **malloc** y **free**, respectivamente.

[1] Con esta función se pretende que el alumno practique el paso parámetros por referencia en C. Es por ello, que esta función debe realizarse de acuerdo con lo que se indica.