Febrero 2010

(1 punto) Ejercicio 1 - (ELIMINATORIO)
Escribe un programa que almacene un número en una variable, *num*, de tipo *int* y que escriba en pantalla la suma de sus dígitos. Por ejemplo, si *num* vale 45, se deberá mostrar en pantalla el 9.

(4 puntos) Ejercicio 2 - Vectores
Escribe un programa que muestre por pantalla un menú con las siguientes opciones y que se ejecute mientras el usuario no pulse la opción de salir **(0.5 puntos)**. Las opciones del menú serán:
1. Leer un vector
2. Escribir un vector
3. Eliminar un elemento del vector
4. Rotar los elementos del vector una posición hacia la derecha
5. Salir

Estructura tu programa en tres ficheros, *principal.c* (contendrá el menú y el main), *funciones.c* (contendrá las funciones relacionadas con vectores) y *funciones.h* (contendrá los prototipos de las funciones relacionadas con vectores). El número de elementos del vector se leerá desde teclado. El tamaño máximo del vector será 100 elementos.

El programa deberá implementar al menos las siguientes funciones:
- **(0.5 puntos) int menu()**. Función que muestre al usuario un menú con las opciones de leer vector, escribir vector, eliminar elemento, rotar vector, rotar vector varias posiciones y salir, y devuelva la opción seleccionada por el usuario.
- **(0.25 punto) void leerVector(double Vector[], int nElementos)**. Función que rellene un vector con doubles leídos desde teclado.
- **(0.25 punto) void escribirVector(double Vector[], int nElementos)**. Función que muestre por pantalla un vector de *nElementos* elementos.
- **(1.25 punto) int eliminarElemento(double Vector[], int nElementos, double ele)**. Función que elimine de *Vector* el primer elemento menor que el elemento *ele* desplazando aquellos elementos que sean necesarios, de forma que no queden huecos en el vector. Por ejemplo, si tenemos el vector [5.1, 7.2, 1.2, 6.3, 8.2] y *ele* vale 3.5 , el vector quedaría [5.1, 7.2, 6.3, 8.2]. Si se ha borrado el elemento, la función devolverá 1 y en caso de no haberse encontrado ninguna elemento menor que *ele*, la función devolverá 0.
- **(1.25 puntos) void rotarVector(double Vector[], int nElementos)**. Función que rota una posición hacia la derecha los elementos del vector. Por ejemplo, si contiene los valores *¡¥2/2-8/3-4/5-7/6^*, la función devolverá el vector con los valores *¥7/6-2/2-8/3-4/5¡^*.

(1.5 puntos) Ejercicio 3 - cadenas
Escribe un programa, *cadenas.c*, que lea por teclado una cadena de caracteres, llámala *cadenaOrigen*, y almacene en otra cadena, *cadenaResultado*, la primera cadena sin espacios en blanco y con el primer carácter tras un espacio en blanco en mayúsculas. La cadena origen podrá tener espacios en blanco al principio, en medio y/o al final. Por ejemplo, si *cadenaOrigen* contiene *" ha llegado a su localidad el camion del tapicero ",* copiará en *cadenaResultado* el valor *"HaLlegadoASuLocalidadElCamionDelTapicero"*.

(3.5 puntos) Ejercicio 4 - matrices
Crea un programa, *matrices.c*, que realice de forma secuencial (sin usar menú) las siguientes operaciones sobre matrices (considera que el programa podrá trabajar como máximo con matrices de 20x20) **(0.5 puntos)**:
- Pedir por teclado las dimensiones de la matriz

- Leer la matriz.
- Escribir la matriz en pantalla.
- Calcular si es triangular superior.
- Pedir por teclado dos columnas, *c1* y *c2*, a intercambiar.
- Intercambiar los valores de las columnas *c1* y *c2*.
- Escribir la matriz resultado en pantalla.

El programa deberá implementar al menos las siguientes funciones:

- **(0.25 puntos) void leerMatriz(int Matriz[][MAX], int nFil, int nCol)**. Función que rellene una matriz con valores de tipo *int* leídos desde teclado.
- **(0.25 puntos) void escribirMatriz(int Matriz[][MAX], int nFil, int nCol)**. Función que imprime una matriz por pantalla.
- **(1.5 puntos) int triangularSuperior(int Matriz[][MAX], int nFil, int nCol)**. Función que comprueba si una matriz cuadrada es triangular superior. La función devolverá -1 si la matriz no es cuadrada, 1 si la matriz es triangular superior y 0 si no lo es. *Una matriz cuadrada de orden n se dice que es triangular superior si los elementos situados por debajo de la diagonal principal son todos ceros*. Así, por ejemplo, la siguiente matriz es triangular superior.

$$U = \begin{matrix} 1 & 7 & 5 & 13 & 4 \\ 0 & 12 & 1 & 6 & 14 \\ 0 & 0 & 2 & 8 & 11 \\ 0 & 0 & 0 & 10 & 9 \\ 0 & 0 & 0 & 0 & 3 \end{matrix}$$

- **(1 puntos) void intercambiaColumnas(int Matriz[][MAX], int nFil, int c1, int c2)**. Función que intercambia los elementos de las columnas *c1* y *c2* de la matriz.

Notas:

Para que los ejercicios sean puntuados, han de funcionar correctamente y han de seguir las especificaciones reseñadas en cada uno de ellos.

- **Excepto en las funciones específicas de entrada/salida, no deben pedirse o mostrarse valores de variables dentro de las funciones, salvo en el *main*. En caso contrario no se evaluará el ejercicio aunque funcione correctamente.**
- **La salida por pantalla será clara y ordenada, en caso contrario restarás un punto a la nota final.**
- **Si los nombres de los archivos no se corresponden con los indicados en el enunciado del examen restarás un punto de tu nota final.**
- **Salvo los ejercicios 1 y 3, aquellos ejercicios/apartados que se implementen sin usar funciones no puntuarán, aunque funcionen correctamente.**
- **No se podrá hacer uso de variables globales.**
- **No olvides firmar tu examen y poner la hora de entrega.**