

	Facultad de Ingeniería y Ciencias Agrarias	PROGRAMACIÓN ESTRUCTURADA
		Practica N° 05
	<i>Estructuras dinámicas complejas y archivos binarios</i>	

Aclaración: Donde dice “función recursiva”, es OBLIGATORIO que la función sea RECURSIVA. En los ejercicios que dicen solamente “función” la recursividad es optativa.

- Ej. 1:** Realizar una función *recursiva* que imprima el contenido de una lista de enteros en forma inversa. Si la lista contiene 14,17,20,23, entonces el programa deberá imprimir: 23, 20, 17, 14.
- 1.1. Ahora el contenido es una estructura *EJ: nombre, apellido y DNI*. Realizar el ejercicio utilizando dicha estructura.
 - 1.2. Inventar otra estructura más compleja y probar nuevamente.
- Ej. 2:** Realizar una función que inserte un valor entero en una lista de enteros ordenada, preservando el orden.
- 2.1. Ahora el contenido es una estructura *EJ: nombre, apellido y DNI*. Realizar el ejercicio utilizando dicha estructura. Hacer tres insertar ordenado:
 - 2.1.1. insertar ordenado por DNI.
 - 2.1.2. insertar ordenado por apellido.
 - 2.1.3. insertar ordenado con la posibilidad de pasar por parámetro el número de campo por el cual se desea ordenar.
- Ej. 3:** Hacer una función que elimina un nodo de una lista de enteros. La función recibirá como parámetros una lista (*la dirección del puntero al primer elemento de la lista*) y el valor que contiene el nodo a ser eliminado. Si el valor estuviera repetido, se eliminará el primer nodo que lo contenga. Considere que la lista puede estar vacía o que el valor solicitado no se encuentre.
- 3.1. Hacer un eliminar todas las ocurrencias.
 - 3.2. Ahora el contenido es una estructura *EJ: nombre, apellido y DNI*. Realizar el ejercicio utilizando dicha estructura.
 - 3.2.1. eliminar una ocurrencias
 - 3.2.2. eliminar todas las ocurrencias
 - 3.2.3. eliminar por DNI, eliminar por apellido, eliminar por nombre.

	Facultad de Ingeniería y Ciencias Agrarias	PROGRAMACIÓN ESTRUCTURADA
		Practica N° 05
	<i>Estructuras dinámicas complejas y archivos binarios</i>	

- Ej. 4:** Programar una función que elimina un nodo de una lista de enteros. La función recibirá como parámetros el puntero al primer nodo de la lista y la posición del nodo a eliminar. Considere que la lista puede estar vacía o que la posición del nodo a eliminar no exista.
- 4.1. Eliminar por posición el contenido de una estructura compleja.
- Ej. 5:** Programar las funciones `push` y `pop` de la implementación de pilas con listas simplemente enlazadas y las funciones `enqueue` y `dequeue` de la implementación de colas con listas simplemente enlazadas.
- 5.1. Resolver el ejercicio donde el contenido es un entero.
- 5.2. Resolver el ejercicio donde el contenido es una estructura
- 5.3. Hacer una función que elimine un elemento (recibido por parámetro) de una pila utilizando las dos operaciones permitidas. *Ayuda: Debe utilizar un pila auxiliar.*
- 5.4. Hacer una función que elimine un elemento (recibido por parámetro) de una cola utilizando las dos operaciones permitidas. *Ayuda: Debe utilizar un cola auxiliar.*
- Ej. 6:** Desarrollar un programa que carga en un lista el contenido de un archivo CSV con varios campos por registro (*defina usted que campos y en consecuencia la estructura*). Luego, una vez en memoria los datos (*en la lista*), el programa deberá mostrar un menú donde permita ordenar los datos (la lista) por un número de campo ingresado por el usuario. Es clave el desarrollo de varias funciones para realizar un program legible, mantenible y reutilizable.
- 6.1. Desarrollar una función que ordene la lista utilizando una lista auxiliar.
- 6.2. Desarrollar una función que ordene la lista utilizando la misma lista a ordenar (sin lista auxiliar).
- Ej. 7:** Realizar cuatro funciones que impriman un árbol binario de enteros según el recorrido preorden, inorden, postorden y por niveles.
- Ej. 8:** Realizar una función que permita insertar un valor en un árbol binario de enteros de forma tal que su impresión en inorden imprima los números ordenados de menor a mayor. Considere que el árbol recibido puede estar vacío.
- Ej. 9:** Programar una función que permite eliminar un nodo de un árbol binario de enteros. La función recibirá como parámetros el puntero a la raíz del árbol y el entero que se desea eliminar. Se deberá eliminar el nodo que primero se encuentra (en búsqueda postorden)

	Facultad de Ingeniería y Ciencias Agrarias	PROGRAMACIÓN ESTRUCTURADA
		Practica N° 05
	<i>Estructuras dinámicas complejas y archivos binarios</i>	

que contenga dicho entero y todos los nodos que dependen de éste. Considere que el árbol puede estar vacío o que el nodo a eliminar no exista.

Ej. 10: Dada la siguiente declaración:

```
typedef struct {
    char dni[9];
    char nombre [100], apellido [100];
    double monto_adeudado;
    unsigned int dia, mes,anio;
}t_datos;
```

Realizar una función de carga que solicite al usuario que ingrese por teclado dichos datos y los grabe en el archivo binario “deudores.dat”, en modo append. La carga de datos finalizará cuando el usuario ingrese como DNI el cero. Considerar que habrá un registro por cada deuda de cada deudor por lo que habrá registros con igual DNI y distintos o iguales montos en caso que la persona hubiese contraído deudas similares en distintas fechas.

Ej. 11: Dado el struct del ejercicio 9, realizar una función que solicite al usuario un nombre del archivo y a continuación abra y muestre el contenido de ese archivo, en formato de listado por pantalla, separando los datos en columnas y mostrando en cada renglón un registro distinto.

Ej. 12: Dado el struct del ejercicio 9, realizar una función que reciba como parámetros los nombres de dos archivos. El primero contendrá los datos de las deudas, la función deberá abrir dicho archivo y traspasar los datos hacia el segundo archivo sin incluir registros con dni repetido. En caso de que haya repeticiones en el archivo original, deben sumarse todos los montos adeudados y grabarse en un único registro para cada DNI.