UNIVERSIDAD DE MÁLAGA

*Departamento de Lenguajes y Ciencias de la Computación*

# LABORATORIO DE PROGRAMACIÓN

**E.T.S.I. Informática- 1º GESTIÓN. 15-SEPTIEMBRE-2006**

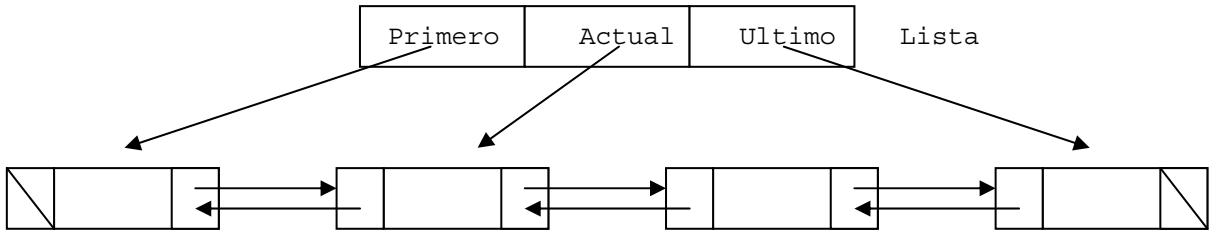
**(EXAMEN PLAN PILOTO EEES)**

APELLIDOS_____NOMBRE_____

DNI _____ ORDENADOR_____

La cadena de supermercados "*La Tienda de la Esquina*" nos ha pedido que le hagamos un programa para la gestión de sus productos. De cada producto se almacenará la siguiente información: referencia (número entero), nombre del producto (cadena de caracteres), tipo de producto (enumerado con los siguientes valores: `Drogueria, Congelado, Fruta, Verdura, Carne, Pescado`) y precio en euros del producto (número real).

Uno de los mayores problemas que tiene el uso de estructuras de almacenamiento dinámicas encapsuladas en un módulo es el problema de ser capaz de recorrer los elementos de la estructura a nivel de usuario. Para solucionar esto vamos a implementar una estructura de lista doblemente enlazada con una cabecera que mantendrá siempre un puntero al primer elemento de la lista, otro puntero al último y otro que usaremos para saber por dónde vamos recorriendo la lista a la hora de listar, buscar, etc.




El puntero `Primero` siempre apuntará al primer elemento de la lista, el puntero `Ultimo` siempre apuntará al último elemento de la lista y el puntero `Actual` solo será actualizado por los procedimientos `PrimeroLista, UltimoLista, AnteriorLista, SiguienteLista`.

Se pide:

- Implementar un módulo llamado `MCadena` para el manejo de cadenas, con las siguientes características:
  - Define las constantes: `MAXCAD (40), FINCAD, y ENTER.`
  - Define los tipos `TCadena` (cadena de 40 caracteres) y `TCompara` (enumerado de valores `Menor, Igual, Mayor`)
  - Ofrece los procedimientos y funciones siguientes:
    - `CopiaCadena`. Copia dos cadenas.
    - `ComparaCadena`. Compara 2 cadenas y retorna un `TCadena`.
    - `IgualCadena`. Devuelve un lógico diciendo si las cadenas son iguales.
    - `MayorCadena`. Devuelve un lógico diciendo si la primera cadena es mayor que la segunda.
    - `MenorCadena`. Devuelve un lógico diciendo si la primera cadena es menor que la segunda.

- Implementar un módulo llamado `MProducto` para el manejo de productos, con las siguientes características:
  - Define las constantes: `SEP (carácter almohadilla), y SP` (espacio en blanco).
  - Define los tipos `TTipoProducto` (enumerado de valores `Drogueria, Congelado, Fruta, Verdura, Carne, Pescado`), `TProducto` (registro con la información del producto).
  - Ofrece los procedimientos y funciones siguientes:
    - `SUC`. Retorna el sucesor de un elemento de tipo `TTipoProducto`.
    - `Cadena_a_TipoProducto.` Transforma una cadena en `TTipoProducto`.
    - `TipoProducto_a_Cadena.` Transforma un `TTipoProducto` en una cadena.
    - `LeerProductoTeclado`. Lee un `TProducto` desde teclado.
    - `LeerProductoFichero.` Lee un `TProducto` desde un fichero de texto.
    - `EscribirProductoPantalla.` Escribe por pantalla un `Tproducto.`
    - `EscribirProductoFichero.` Escribe a fichero de texto un `Tproducto.`
    - `LeeTipoProducto.` Lee un `TTipoProducto` de teclado.
    - `EscribeTipoProducto.` Lee un `TTipoProducto` por pantalla.

- Implementar un módulo llamado `MLista` para el manejo de la lista de productos, con las siguientes características:
  - Define el tipo `TLista` y los tipos auxiliares necesarios para implementar la estructura.
  - Ofrece los procedimientos y funciones siguientes:

- `CrearLista.` Crea una `TLista` vacía.
- `ListaVacia.` Nos dice si una `TLista` está vacía.
- `ListaLlena.` Nos dice si una `TLista` está llena.
- `InsertarLista.` Inserta un Producto al final de la `TLista`.. Si el producto existe (misma referencia) se reemplaza.
- `EliminarLista.` Elimina (si existe) un producto dada su referencia.
- `BuscarLista.` Retorna (si existe) un producto dada su referencia.
- `DestruirLista.` Destruye una `TLista`.
- `PrimeroLista.` Coloca el puntero `Actual` apuntando al primer producto de la lista, si éste existe, es decir, si la lista no está vacía.
- `UltimoLista.` Coloca el puntero `Actual` apuntando al último producto de la lista, si éste existe, es decir, si la lista no está vacía.
- `AnteriorLista.` Coloca (si existe) `Actual` en su anterior, es decir siempre que `Actual` no sea `NULO`.
- `SiguienteLista.` Coloca (si existe) `Actual` en su siguiente, es decir siempre que `Actual` no sea `NULO`..
- `InfoLista.` Retorna (si existe) el producto apuntado por `Actual`.

- Implementar un programa principal que utilizando los módulos anteriores ofrezca un menú con las siguientes opciones:
  - `A. Insertar Producto.` Se leerá un producto desde teclado y se insertará en la lista.
  - `B. Listar Productos.` Se mostrarán por pantalla todos los productos que hay en la lista.
  - `C. Buscar Producto.` Se buscará un producto dada su referencia y se mostrará por pantalla.
  - `D. Borrar Producto.` Se borrará un producto dada su referencia.
  - `E. Cargar Productos desde Fichero.` Se añadirán a la lista todos los productos que haya en el fichero cuyo nombre se pedirá por teclado. (Ver formato de fichero más abajo).
  - `F. Salvar Productos a Fichero.` Se guardarán todos los productos que haya en la lista en un fichero cuyo nombre se pedirá por teclado y que deberá ser creado o borrado su contenido. (Ver formato de fichero más abajo).
  - `X. Salir del Programa.` Se debe pedir confirmación de borrado

Formato del fichero (una linea por producto) donde SEP es el carácter almohadilla (#) y SP un espacio en blanco:
`<Referencia>SP<nombre>SEP<Tipo de Producto>SEP<precio><ENTER>`
Ejemplo:

```
21 Solomillo#Carne#12.30
12 Lubina#Pescado#11.40
33 Boquerones#Pescado#4.30
34 Presa Iberica#Carne#8.35
25 Tarrina de Turron#Congelado#3.60
```

**NOTA.**

- EL AÑADIR OPERACIONES EN LA PARTE DE DEFINICIÓN DE UN MÓDULO, ASÍ COMO EL **USO DE DETALLES DE IMPLEMENTACIÓN DE LA LISTA** DE LOS MODULOS DENTRO DEL PROGRAMA PRINCIPAL SERÁ CAUSA DE **SUSPENSO**, es decir, no se podrá hacer nada del tipo `ptr=l, l =NULL, ptr->sig`, etc., fuera de la parte de implementación del módulo de listas.
- Para hacer sumar con las notas de clase hace falta obtener una puntuación mínima de 4 sobre 10 en este examen.
- Todo **PROGRAMA QUE NO COMPILE** correctamente se considerará con una **CALIFICACIÓN INFERIOR A 4** y por tanto **SUSPENSO**.
- Para **OBTENER 4 PUNTOS** es necesario el correcto funcionamiento de los apartados **A, B, E y X**