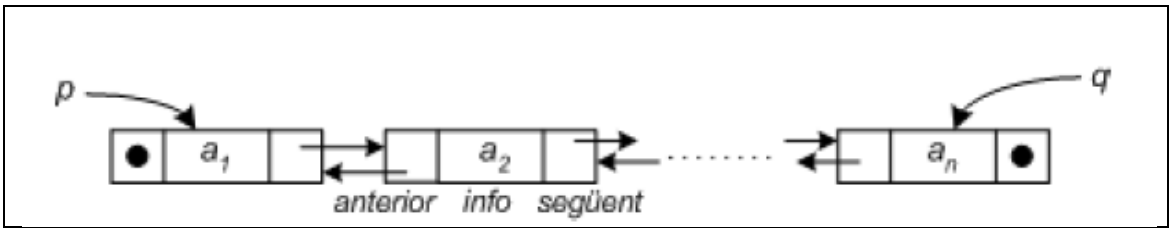
PRÁCTICA FINAL: LISTAS DOBLEMENTE ENCADENADAS

OBJETIVO

Diseñar y codificar un programa en **lenguaje C** para representar una **lista doblemente encadenada** en la que cada nodo contiene información sobre un determinado **producto**.

Gráficamente:



PUNTOS OBLIGATORIOS A IMPLEMENTAR

1. Definición de una **estructura de datos** para representar **PRODUCTOS**:
    a. Cada producto tiene un **identificador** numérico, múltiple de 10 (Ejemplo: 10, 20, 30, 40, etc.).
    b. Cada producto tiene un **nombre** descriptivo de 50 caracteres como máximo.
    c. Cada producto tiene un precio de **coste** (número real).
    d. Cada producto tiene un precio de **venta** (número real).
    e. Cada producto tiene una variable para indicar si está en **catálogo**.

2. Definición de una **estructura de datos** para representar **NODOS**:
    a. Cada nodo contiene una referencia al elemento **anterior** de la lista.
    b. Cada nodo contiene una referencia al elemento **posterior** de la lista.
    c. Cada nodo contiene un producto en el **área de datos**.

3. Definición de una **estructura de datos** para representar la **LISTA**:
    a. Se debe almacenar una **referencia** al 1er nodo.
    b. Se debe almacenar una **referencia** al último nodo.

4. **Inserción** de un producto por el **inicio** de la lista:
    a. Solicitar la información correspondiente al usuario.
    b. Considerar los casos especiales, si existen.

5. **Inserción** de un producto por el **final** de la lista:
    a. Solicitar la información correspondiente al usuario.
    b. Considerar los casos especiales, si existen.

6. **Eliminación** del primer **producto** de la lista:
    a. Considerar los casos especiales, si existen.

7. **Eliminación** del último **producto** de la lista:
    a. Considerar los casos especiales, si existen.

8. **Longitud** de la lista:
    a. Opción 1: Contar todos los nodos.
    b. Opción 2: Contar solo aquellos nodos que tengan un producto con ID superior a un determinado valor. El valor se solicita al usuario.

9. Determinar el **estado** de la lista:
    a. Indicar si la lista está vacía.
    b. Indicar si la lista está llena.

10. **Recorrido** de la lista:
    a. Desde el primer nodo hasta el último.
    b. Desde el último nodo hasta el primero.

c. Se debe preguntar al usuario la dirección que se aplicará en el recorrido.

11. Almacenamiento del contenido de la lista en **fichero:**
    a. Solicitar nombre de archivo al usuario.
    b. El fichero destino es de formato libre.

12. **Eliminar** lista:
    a. Borra todos los nodos de la lista.

13. **Menú** inicial
    a. Similar a:

```
OPCIONES

1.Inserción de nodo por el inicio
2. Inserción de nodo por el final
3. Eliminación del primer nodo
4. Eliminación del último nodo
5. Ver longitud de la lista
6. Ver estado de la lista
7. Recorrido de la lista
8. Guardar en fichero
9. Eliminar lista
```

PUNTOS OPCIONALES A IMPLEMENTAR

1. Validar las **entradas de datos** del usuario. Si el programa espera, por ejemplo, un número entero, verificar que el usuario proporciona ese valor y no otro.

   Para validar las entradas, se pueden utilizar funciones de la **librería <ctype.h>.**

2. Utilizando el campo identificador de cada producto, aplicar un **ALGORITMO DE ORDENACIÓN** sobre la lista, para obtener una nueva lista ordenada.

PONDERACIÓN

PREGUNTA	***PUNTUACIÓN***
1. Estructura producto	1.0 puntos
2. Estructura nodo	1.0 puntos
3. Estructura lista	0.5 puntos
4. Inserción inicio	0.5 puntos
5. Inserción final	0.5 puntos
6. Eliminación inicio	0.5 puntos
7. Eliminación final	0.5 puntos
8. Longitud lista	0.5 puntos
9. Estado lista	0.5 puntos
10. Recorrido lista	1.0 puntos
11. Guardar en fichero	1.0 puntos
12. Eliminar lista	1.0 puntos
13. Menú	0.5 puntos
Estructura, diseño general	0.5 puntos
Modularidad, reutilización código	0.5 puntos
TOTAL:	10.0 puntos
1. Entradas	1.5 puntos
2. Ordenación	1.5 puntos
TOTAL:	3.0 puntos

CONSIDERACIONES

1. El código fuente debe contener, en la parte superior:
    a. Nombre y apellidos:
        i. Manuel López
    b. Turno (mañana/tarde)
        i. Analista Programador Mati
    c. Máquina
        i. AULA02PC25

2. Se pueden codificar todas las funciones adicionales/de soporte que sean necesarias.

3. Se pueden utilizar variables globales.

4. Programa LIBRE DE:

    a. ERRORES DE COMPILACIÓN
    b. ERRORES DE EJECUCIÓN

MATERIAL A ENTREGAR

1. Código fuente.

2. Archivo de proyecto ".dev"

3. Breve documento descriptivo en formato de texto argumentando:

    a. similitudes y diferencias entre la estructura utilizada y otras estructuras, como las pilas o las colas.

    b. ventajas e inconvenientes del uso de las listas doblemente encadenadas.

    c. pseudocódigo correspondiente a las definiciones de estructuras de datos.

    d. recursos utilizados en la utilización de la práctica.

REFERENCIAS

1. Librería <ctype>:

http://www.cplusplus.com/reference/cctype/