EjercicioEnlazadaSimple

Estructuras de Datos

Tema 2: listas, pilas y colas

1º Grado en Ingeniería de la Computación
© Profesor Dr. Carlos Grima Izquierdo (www.carlosgrima.com)
URJC (www.urjc.es)

Programar una lista enlazada simple (clase "ListaEnlazada") con métodos similares (con algunas excepciones) a los del EjercicioListaContigua2 resuelta. Tendremos lo siguiente:

- Estructura "Nodo". Cada elemento de la ListaEnlazada realmente será un objeto de la estructura Nodo. Esta estructura tendrá:
  - Un entero que representa al auténtico elemento.
  - Un puntero al nodo siguiente de la lista enlazada. Será NULL si estamos en el último nodo de la lista.
- Clase "ListaEnlazada":
  - Puntero al primer nodo de la lista (atributo privado). Tendrá NULL si la lista enlazada está vacía.
  - Atributo privado "n": número actual de elementos en la lista.
  - Método privado llamado getNodo() que, ante una posición, devuelva el puntero al nodo en esa posición. Tener este método simplificará la programación de los métodos (descritos a continuación) "setValor()", "getValor()", "insertar()" y "eliminar()", maximizando la cohesión.
  - Constructor sin parámetros que creará una lista enlazada vacía.
  - Métodos públicos "setValor()" y "getValor()", con las mismas características y cabecera que en la actividad EjercicioListaContigua2 resuelta.
  - Método público "getN()", con las mismas características y cabecera que en la actividad 4.2.
  - Métodos públicos "insertar()" y "eliminar()", con las mismas características y cabecera que en la actividad EjercicioListaContigua2 resuelta. A la hora de programar cada uno de los dos, posiblemente tendrás que distinguir varios casos (piensa en cada uno de los dos métodos si se pueden integrar entre sí algunos casos):
    - Caso "normal": cuando queremos insertar/eliminar en una posición "i" que no sea ni el principio ni el final de la lista, en una lista no vacía.
    - Caso especial de que queramos insertar en una lista vacía.
    - Casos especiales de que queramos insertar al principio o al final de una lista no vacía.
    - Casos especiales de que queramos eliminar el primer elemento o el último en una lista no vacía.
  - Destructor, que destruirá todos los nodos de la lista enlazada (¡ojo: no sólo el primero!).
- Biblioteca "impresionListasEnlazadas", que contendrá el procedimiento "imprimirListaEnlazada()", muy parecida al de la actividad EjercicioListaContigua2 resuelta, con la salvedad de que ahora imprime una lista enlazada en vez de una contigua. Para acceder a un elemento de la lista enlazada, llamará a "getValor()", lo cual

hará que la complejidad temporal de “imprimirListaEnlazada()” sea mucho mayor que la de “imprimirListaContigua()”. En la próxima actividad intentaremos mejorar esto.
- Un “main()” para probar todo exhaustivamente, que haga lo que se ve en la caja negra.

Ten en cuenta las siguientes consideraciones:

- Fíjate en que hemos quitado (con respecto a EjercicioListaContigua2) algunos métodos y atributos privados que ya no nos sirven, como por ejemplo “isLlena()” (una lista enlazada nunca puede estar llena), “insertarAlFinal()”, “eliminarAlFinal()”, el atributo “capacidad” (en una lista enlazada el concepto de capacidad ya no tiene sentido, pues nunca se llena al no tener que estar los nodos almacenados de forma contigua en memoria), y el “INCREMENTO” (al no haber capacidad, tampoco necesitamos el concepto de “incremento”). Los métodos “concatenar()” y “buscar()” los añadiremos en la siguiente actividad.
- La posición del elemento que se encuentra en el primer nodo de la lista es 0, y la del último es n-1. Como en todas las listas.
- Recuerda que hay que poner la complejidad temporal (en el peor caso) de todos los métodos.

La prueba de caja negra será: