UNIDAD TEMÁTICA 3: Listas, Pilas y Colas

PRACTICOS DOMICILIARIOS INDIVIDUALES

Ejercicios #1-5 Lista encadenada

Los nodos de una lista simplemente encadenada tienen dos atributos:

- DATOS, de tipo “dato”.
- SIGUIENTE, de tipo “nodo de lista”, que hace referencia al nodo siguiente en la lista.

Ejercicio #1

Sean **nodo1**, **nodo2** y **nodo3** tres nodos consecutivos de una lista (nodo2 es el siguiente a nodo1 y nodo3 es el siguiente a nodo2).

Analice el siguiente fragmento de código (utilice dibujos o diagramas para clarificar qué es lo que sucede):

```
Nuevo nodo otroNodo
otroNodo.siguiente ← nodo1
nodo2.siguiente ← nodo3
```

a) Inserta “otroNodo” en la lista, quedando como anterior a nodo1.
b) Inserta “otroNodo” en la lista, quedando entre nodo1 y nodo2.
c) Elimina nodo2 de la lista.
d) No tiene ningún efecto sobre la lista.

Ejercicio #2

Sean **nodo1**, **nodo2** y **nodo3** tres nodos consecutivos de una lista (nodo2 es el siguiente a nodo1 y nodo3 es el siguiente a nodo2).

Analice el siguiente fragmento de código (utilice dibujos o diagramas para clarificar qué es lo que sucede):

```
Nuevo nodo otroNodo
otroNodo ← nodo1.siguiente
nodo1.siguiente ← nodo3
```

a) Inserta “otroNodo” en la lista, quedando como anterior a nodo1.
b) Inserta “otroNodo” en la lista, quedando entre nodo1 y nodo2.
c) Elimina nodo2 de la lista.
d) No tiene ningún efecto sobre la lista.

Ejercicio #3

Sean **nodo1**, **nodo2** y **nodo3** tres nodos consecutivos de una lista (nodo2 es el siguiente a nodo1 y nodo3 es el siguiente a nodo2).

Analice el siguiente fragmento de código (utilice dibujos o diagramas para clarificar qué es lo que sucede) y responda las preguntas proyectadas en pantalla:

```
Nuevo nodo otroNodo
otroNodo.siguiente ← nodo1.siguiente
nodo1.siguiente ← otroNodo
```

a) Inserta “otroNodo” en la lista, quedando como anterior a nodo1.
b) Inserta “otroNodo” en la lista, quedando entre nodo1 y nodo2.
c) Elimina nodo2 de la lista.
d) Dará error en tiempo de ejecución si nodo1 es el primero o nodo3 es el último.

Ejercicio #4

Analice el siguiente fragmento de código (utilice dibujos o diagramas para clarificar qué es lo que sucede) y responda las preguntas proyectadas en pantalla:

```
Nuevo nodo otroNodo
Nuevo nodo nodoActual
nodoActual ← primero
mientras nodoActual <> nulo hacer
  nodoActual ← nodoActual.siguiente
fin mientras
nodoActual.siguiente ← otroNodo
```

a) Inserta correctamente “otroNodo” en la lista, quedando como último nodo.
b) Inserta correctamente “otroNodo” en la lista, quedando como primer nodo.
c) El algoritmo está mal hecho, ya que dará error en tiempo de ejecución si la lista está vacía.
d) El algoritmo está mal hecho, ya que dará siempre error en tiempo de ejecución.

Ejercicio #5

Analice el siguiente fragmento de código (utilice dibujos o diagramas para clarificar qué es lo que sucede) y responda las preguntas proyectadas en pantalla:

```
Nuevo nodo otroNodo
Nuevo nodo nodoActual
nodoActual ← primero
mientras nodoActual.siguiente <> nulo hacer
  nodoActual ← nodoActual.siguiente
fin mientras
nodoActual.siguiente ← otroNodo
```

a) Inserta correctamente “otroNodo” en la lista, quedando como último nodo.
b) Inserta correctamente “otroNodo” en la lista, quedando como primer nodo.
c) El algoritmo está mal hecho, ya que dará error en tiempo de ejecución si la lista está vacía.
d) El algoritmo está mal hecho, ya que dará siempre error en tiempo de ejecución.

Ejercicio #6

Escenario:

Se desea llevar un registro de asistencia de un cierto curso universitario, el cual contará con una cantidad no determinada inicialmente de alumnos. Para ello, se ha decidido utilizar una lista para representar los alumnos en este curso.

Cada elemento de la lista entonces tendrá un identificador del alumno y un campo que se ha de incrementar cada vez que el alumno concurra a una clase. También se desea registrar el total de clases impartidas en el curso, y con este dato luego para cada alumno obtener el porcentaje de asistencia a las clases.

Las listas pueden implementarse físicamente de dos formas básicas: utilizando un array, o armando una lista encadenada. Se desea la opinión experta de tu Equipo para determinar qué utilizar para resolver eficientemente el problema planteado.

a) ¿Cuál es el costo de memoria en cada caso?
b) ¿Cuáles son las consideraciones que tu Equipo haría referentes a la cantidad de alumnos del curso que soporta cada tipo de estructura? (puedes considerar que, como en la UCU, las inscripciones al curso suelen estar habilitadas desde varias semanas antes de empezar el curso hasta dos semanas después de haber comenzado)

B- Las consideraciones a tomar son el numero de alumnos y el acceso para tomar asistencia.

Por el lado del numero de alumnos, asumiendo que se tiene un numero máximo permitido por curso o se sabe con anterioridad seria mas eficiente un array, permitiendo además un mejor acceso a cada estudiante por índice para evaluar la asistencia.

Si por el contrario se tratara de un acceso no previsto y/o ilimitado a la inscripción de una materia podría ser una mejor opción las listas encadenadas ya que no requieren de memoria contigua, lo que permite una fácil adaptación a diferentes capacidades con la desventaja que consumen un poco mas de memoria al utilizar punteros adicionales, además este tipo de estructuras permiten reorganizar la lista en caso que un estudiante o varios se den de baja de un curso. esta opción seria mas ineficiente a la hora de pasar asistencia ya que se debe seguir la cadena.

A-
Array (Arreglo):

En un array, se reserva un bloque de memoria contigua para almacenar todos los elementos.
El costo de memoria total en un array se calcula multiplicando el tamaño de cada elemento por el número total de elementos.
Cada elemento del array constará de:
Un identificador del alumno (puede ser un número o una cadena).
Un campo para registrar la asistencia (puede ser un entero que se incrementa cada vez que el alumno asiste a una clase).
Además, se necesita un espacio adicional para almacenar el total de clases impartidas en el curso.

Lista encadenada:

En una lista encadenada, se asigna memoria para cada nodo de la lista, y cada nodo contiene datos del alumno y un puntero al siguiente nodo.
El costo de memoria total en una lista encadenada depende del número de nodos en la lista.
Cada nodo constará de:
Un identificador del alumno.
Un campo para registrar la asistencia.
Un puntero al siguiente nodo.
Al igual que en el array, se necesitará espacio adicional para almacenar el total de clases impartidas en el curso.