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.

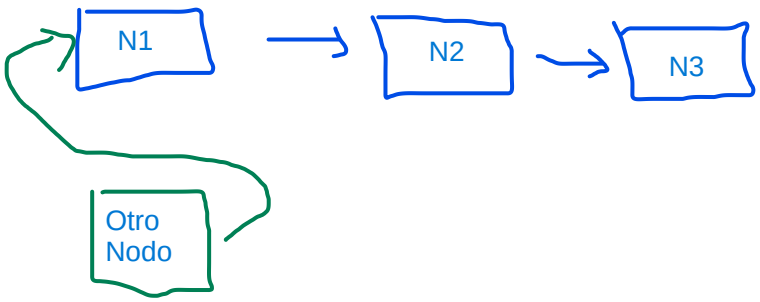
Ejercicio #1

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.



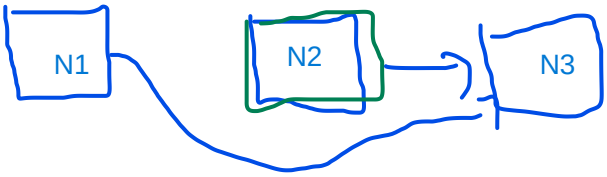
Ejercicio #2

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.



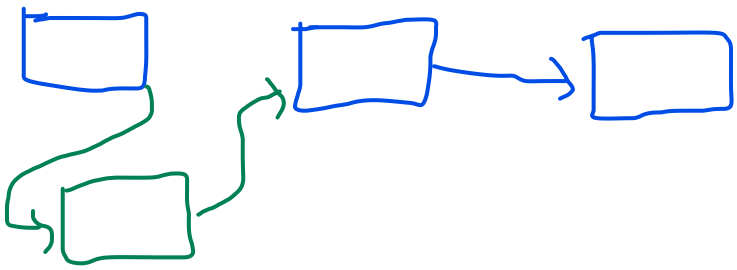
Ejercicio #3

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.

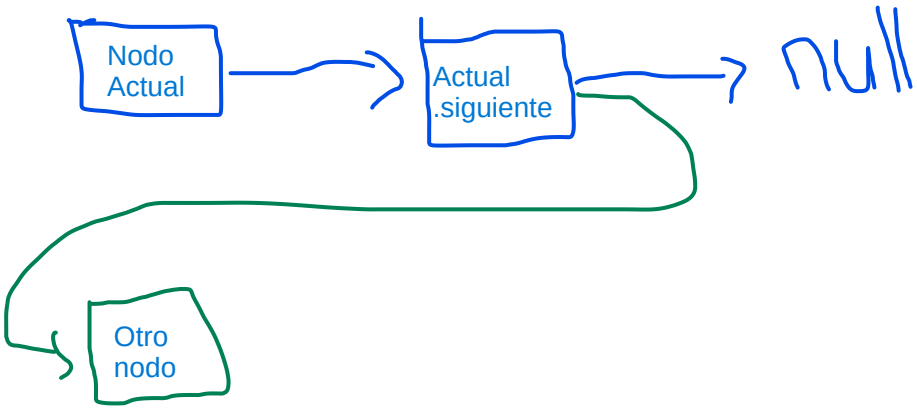
Da error porque asigna a nodoActual una referencia que apunta a Null

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)

a)
Array:
Supongamos que un curso ronda los 40 alumnos, por ende, creamos un array de 40 alumnos.
La clase "alumno" tiene como atributo un byte que cuenta las asistencias.
Esta solución sería de 1 byte por tamaño del objeto, y 4 bytes por la referencia al objeto.
Multiplicado por 40 (tamaño del array), serían 200 bytes, más 4 bytes por referencia al array, esta solución costaría 204 bytes.

LinkedList:
Cada nodoAlumno ocuparía un byte que guarde las asistencias y 4 bytes para referenciar al siguiente nodo.
Esta solución serían 5 bytes por 40 alumnos (al igual que el array), que es 200 bytes. A esto se le deben sumar 4 bytes por referencia al primer nodo. 204 bytes.

b)
En este caso, como no sabemos cual va a ser la cantidad de alumnos, decidiría utilizar LinkedList, ya que puede ser que con el array nos quedemos cortos de tamaño, y agrandarlo llevaría un trabajo mayor.
También puede pasar que nos sobre espacio en el array, desperdiciando memoria, cosa que con la LinkedList no pasaría.
Por último, puede pasar que un alumno se baje del curso más adelante, dejando aún más espacio desperdiciado para el array.