**Operaciones básicas con listas**

Las operaciones con listas son las acciones que se pueden realizar sobre las listas, que son estructuras de datos que permiten almacenar una colección de elementos. Las listas son mutables, lo que significa que se pueden modificar después de su creación (Silvia Guardati, 2017).

Los tipos de operaciones básicas son:

* Insertar: Se pueden agregar elementos al final de la lista, al principio de la lista o en una posición determinada.
* Buscar es una operación que comprueba si un elemento determinado se encuentra en una lista. Si el elemento se encuentra en la lista, la operación devuelve un valor booleano que indica que el elemento está presente. Si el elemento no se encuentra en la lista, la operación devuelve un valor booleano que indica que el elemento no está presente.
* Eliminar: es una operación mutable que elimina un elemento de una lista. Se puede eliminar un elemento por su posición o por su valor.

**Operaciones básicas con listas Simples**

**Insertar Nodo en una lista vacía**

* Suponer que tenemos un nodo a ser insertado.

**  
Figura 5** *Nodo a insertar*

* Un puntero denominado Lista que apunte a Null.

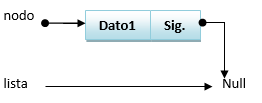
**  
Figura 6** *Nodo auxiliar llamado lista que apunta a NULL*

Proceso:

* Considerar condiciones iniciales.

**  
Figura 7** *Lista apunta a NULL*

* Nodo---> siguiente debe apuntar a NULL

****

**Figura 8** *Nodo siguiente apunta a NULL*

* Lista apunta a nodo

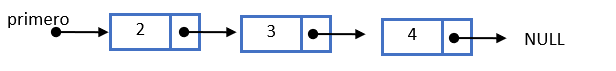
**  
Figura 9** *Lista apunta al Nodo*

* Lista=nodo

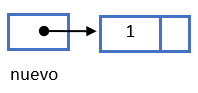
**  
Figura 10** *Nodo va a hacer igual a lista*

**Insertar en la cabeza de la lista**

A continuación, se muestra una lista con 4 elementos insertados (2,3,4), y se desea ingresar el numero 1 por la cabeza, para ello se siguen los siguientes pasos:

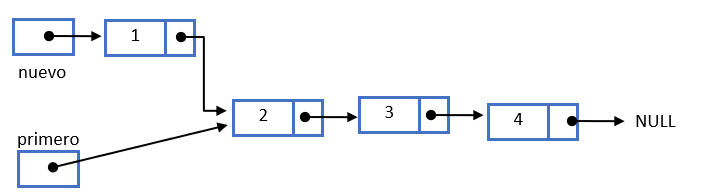
**  
Figura 11** *Lista simple*

* Crear un nodo e inicializar el campo dato al nuevo elemento. La dirección del nodo creado se asigna a *nuevo*.

****

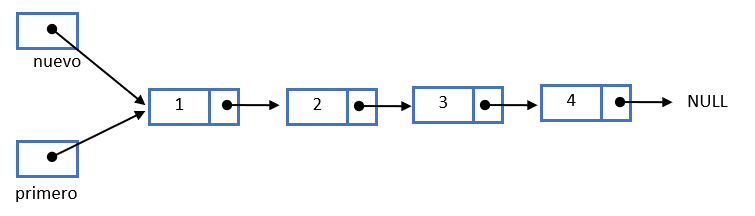
**Figura 12** *Asignación de la dirección del nodo creado a nuevo*

* Hacer que el campo *enlace* del nodo creado apunte a la cabeza (primero) de la lista.

****

**Figura 13** *Nuevo Nodo apunte a la cabeza*

* Hacer que *primero* apunte al nodo que se ha creado

****

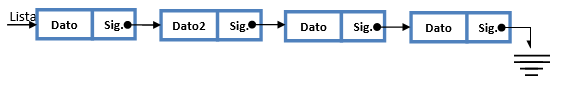
**Figura 14** *Inserción del nuevo nodo*

* En este momento, la función termina su ejecución, la variable local nuevo desaparece y sólo permanece el puntero primero al inicio de la lista.

**Insertar en la cola de la lista**

Condiciones:

* Disponer de una lista con cierto número de elementos.

****

**Figura 15** *Lista Simple con datos*

* Disponer de un nuevo a ser insertado al final de la lista.

****

**Figura 16** *Nuevo nodo*

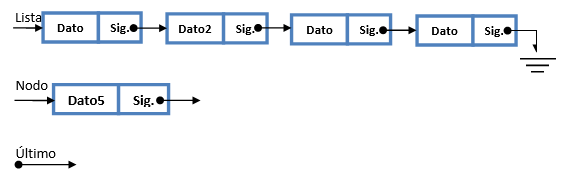
* Disponer de un apuntador denominado: "ultimo".

****

**Figura 17** *Auxiliar denominado Ultimo*

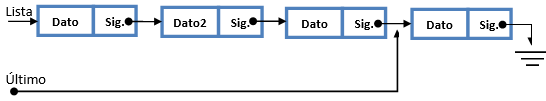
**Proceso:**

* Integrar en un solo diagrama las condiciones iniciales.

****

**Figura 18** *Diagrama de condiciones iniciales*

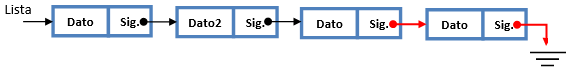
* Hacer que el apuntador "último" apunte al último nodo de la lista.

****

**Figura 19** *Ultimo apunta al último nodo*

* Nodo siguiente y último siguiente apuntan al último nodo de la lista.

Y además el puntero denominado nodo siguiente de la lista que apunta al valor de Null pasa a denominarse último siguiente.

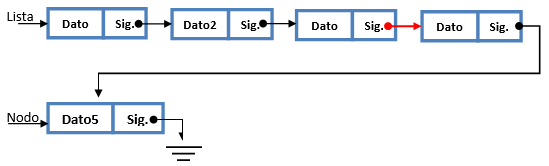
****

**Figura 20** *nodo->siguiente = ultimo siguiente*

* Hacer que el apuntador nodo->siguiente del nuevo nodo apunte y tome el valor de Null.

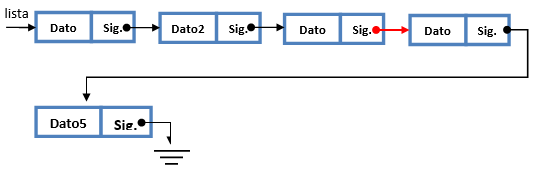
nodo->siguiente = Null

* Hacer que el apuntador ultimo->siguiente que tenía el valor de Null apunte al nuevo nodo.

****

**Figura 21** *ultimo->siguiente y nodo->siguiente apuntan al nuevo nodo a la vez*

* ultimo->siguiente=nodo->siguiente

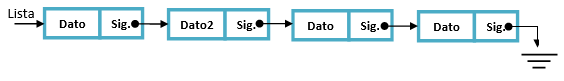
****

**Figura 22** *Inserción del nuevo Nodo*

**Insertar un nuevo nodo en cualquier posición**

Condiciones Iniciales:

* Disponer de una lista.

**  
Figura 23** *Lista Simple*

* Disponer de un nuevo nodo a ser insertado.

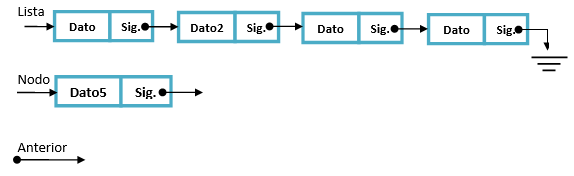
**  
Figura 24** *Nuevo Nodo*

* Disponer de un apuntador denominado anterior

**  
Figura 25** *Puntero anterior*

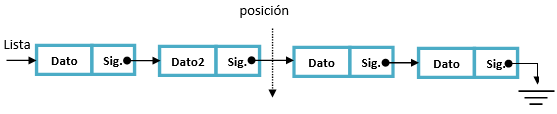
Proceso:

* Integrar en un solo diagrama las condiciones iniciales.

****

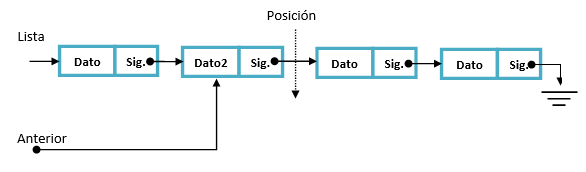
**Figura 26** *Diagrama de condiciones iniciales*

* Determinamos la posición en la cual se va a insertar el nuevo nodo.

****

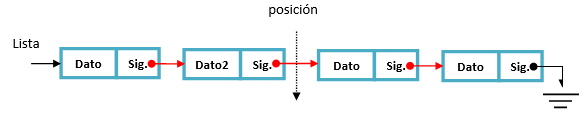
**Figura 27** *Determinación de nuevo nodo a insertar*

* Hacer que el apuntador anterior apunte al nodo->anterior que se encuentra en la lista al cual se va a insertar el nuevo nodo.

****

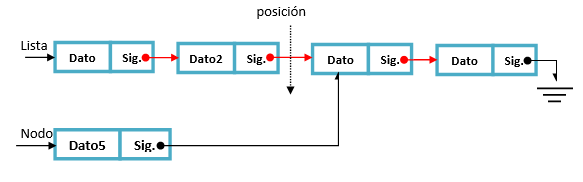
**Figura 28** *Apuntador apunta al nodo anterior donde se desea insertar el nodo*

* Apuntador nodo->siguiente de la lista y anterior->siguiente apuntan a un mismo nodo de la lista.

****

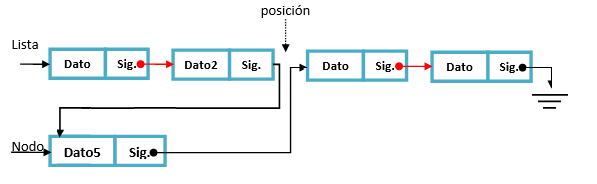
**Figura 29** *nodo->siguiente = anterior->siguiente*

* Hacer que el apuntador nodo->siguiente del nuevo nodo apunte al nodo que se encuentra ubicado en la posición siguiente en la cual se va a ubicar el nuevo nodo**.**

****

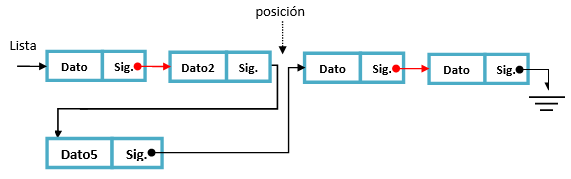
**Figura 30** *Nodo Siguiente apunte a la ubicación de la inserción del nuevo nodo*

* Apuntador anterior->siguiente de la lista y apuntador nodo->siguiente apuntan a un mismo nodo.

****

**Figura 31** *Nodo Siguiente = nodo->siguiente*

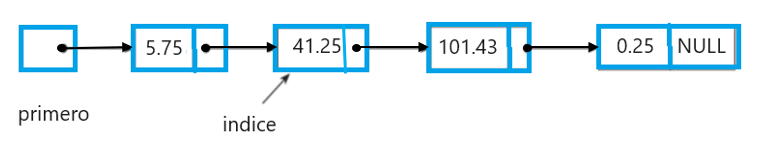
* Apuntador nodo->siguiente y apuntador nodo->siguiente apuntan al nuevo nodo.
* nodo->siguiente = nodo->siguiente

****

**Figura 32** *Inserción del nuevo nodo*

**Buscar Nodo**

La operación búsqueda de un elemento en una lista enlazada recorre la lista hasta encontrar el nodo con el elemento. El algoritmo que se utiliza, una vez encontrado el nodo, devuelve el puntero al nodo (en caso negativo, devuelva NULL). Otro planteamiento consiste en devolver true si encuentra el nodo y false si no está en la lista

****

**Figura 33** *Búsqueda de un nodo.*

La función buscar Lista de la clase Lista utiliza el puntero índice para recorrer la lista, nodo a nodo. Primero, se inicializa índice al nodo cabeza (primero), a continuación, se compara el dato del nodo apuntado por índice con el elemento buscado, si coincide la búsqueda termina, en caso contrario índice avanza al siguiente nodo. La búsqueda termina cuando se encuentra el nodo, o bien cuando se ha terminado de recorrer la lista y entonces índice toma el valor NULL.

**Eliminar Nodo**

Eliminar un nodo de una lista enlazada supone enlazar el nodo anterior con el nodo siguiente al que se desea eliminar y liberar la memoria que ocupa. El algoritmo se enfoca para eliminar un nodo que contiene un dato, sigue estos pasos:

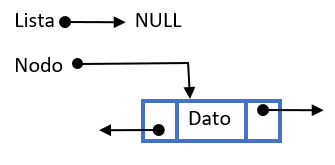
* Búsqueda del nodo que contiene el dato. Se ha de obtener la dirección del nodo a eliminar y la dirección del anterior.
* El enlace del nodo anterior que apunte al nodo siguiente al que se elimina.
* Si el nodo a eliminar es el cabeza de la lista (primero), se modifica primero para que tenga la dirección del siguiente nodo.
* Por último, la memoria ocupada por el nodo se libera.

Naturalmente, eliminar () es una función miembro y pública de la clase Lista, recibe el dato del nodo que se quiere borrar. Desarrolla su propio bucle de búsqueda con el fin de disponer de la dirección del nodo anterior.

**Operaciones con Listas Doblemente Enlazadas**

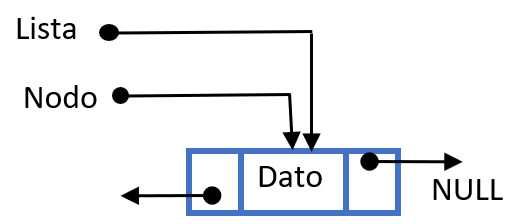
**Insertar Nodo en una lista vacía**

Partiremos de que ya tenemos el nodo a insertar y, por supuesto un puntero que apunte a él, además el puntero que define la lista, que valdrá NULL:

****

**Figura 34** *Diagrama del nuevo nodo y el puntero que apunta a NULL*

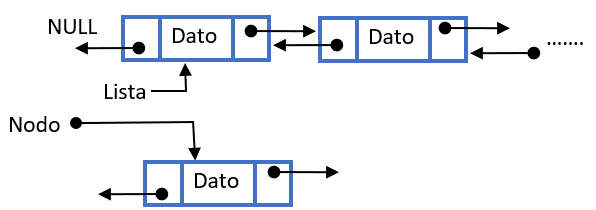
* lista apunta a nodo.

****

**Figura 35** L*ista->siguiente y lista->anterior apunten a NULL*

**Insertar Nodo en una lista por la Cabeza**

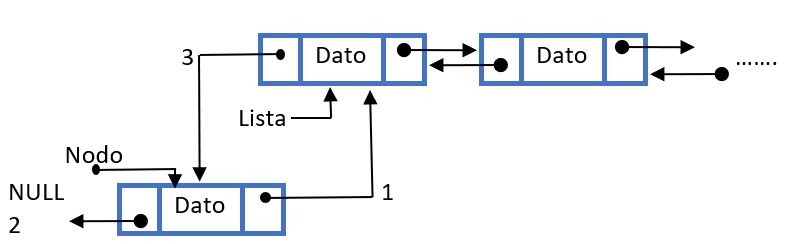
Partimos de una lista no vacía. Para simplificar, consideraremos que lista apunta al primer elemento de la lista doblemente enlazada:

****

**Figura 36** *Diagrama del nuevo nodo a insertar y la lista no vacía*

El proceso es el siguiente:

* nodo->siguiente debe apuntar a Lista.
* nodo->anterior apuntará a Lista->anterior.
* Lista->anterior debe apuntar a nodo.

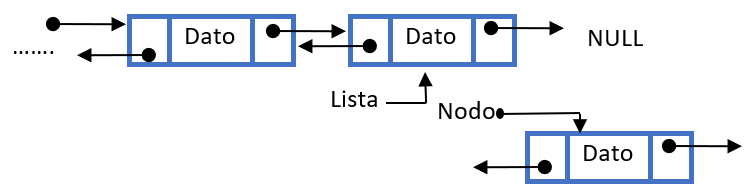
****

**Figura 37** *Diagrama del comportamiento de nuevo nodo*

Recuerda que Lista no tiene por qué apuntar a ningún miembro concreto de una lista doblemente enlazada, cualquier miembro es igualmente válido como referencia.

**Insertar Nodo en una lista por la Cola.**

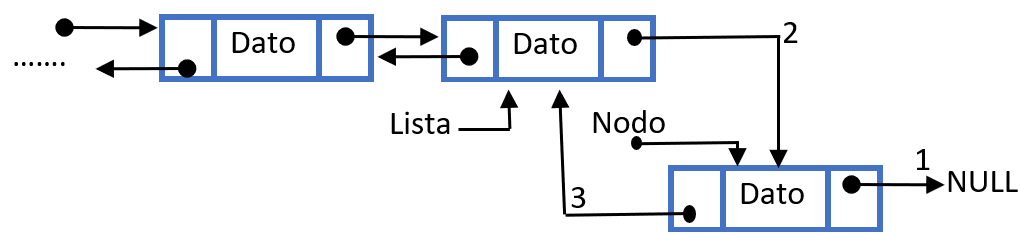
Igual que en el caso anterior, partiremos de una lista no vacía, y de nuevo para simplificar, que Lista está apuntando al último elemento de la lista:

****

**Figura 38** *Diagrama del nuevo nodo a insertar y la lista no vacía*

El proceso es el siguiente:

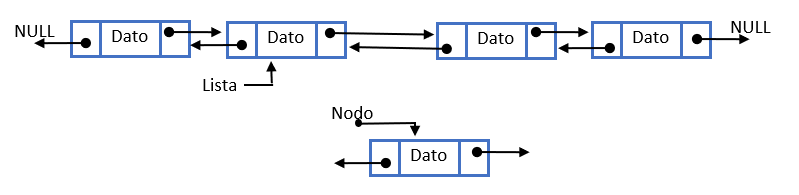
* nodo->siguiente debe apuntar a Lista->siguiente (NULL).
* Lista->siguiente debe apuntar a nodo.
* nodo->anterior apuntará a Lista.

****

**Figura 39** *Diagrama del comportamiento del nodo que se desea insertar*

**Insertar un nuevo nodo en cualquier posición**

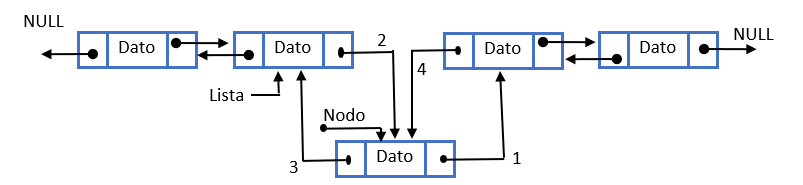
Bien, este caso es más genérico, ahora partimos de una lista no vacía, e insertaremos un nodo a continuación de uno nodo cualquiera que no sea el último de la lista:

****

**Figura 40** *Diagrama de la lista y la posición donde se desea ingresar el nuevo nodo*

El proceso sigue siendo muy sencillo:

* Hacemos que nodo->siguiente apunte a lista->siguiente.
* Hacemos que Lista->siguiente apunte a nodo.
* Hacemos que nodo->anterior apunte a lista.
* Hacemos que nodo->siguiente->anterior apunte a nodo

****

**Figura 41** *Diagrama del comportamiento del nodo que se desea insertar*

**Buscar Nodo**

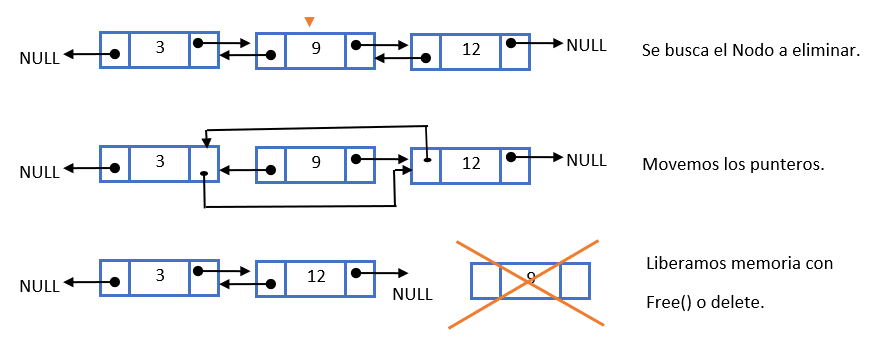
Para recorrer una lista procederemos de un modo parecido al que usábamos con las listas abiertas, ahora no necesitamos un puntero auxiliar, pero tenemos que tener en cuenta que Lista no tiene por qué estar en uno de los extremos:

* Retrocedemos hasta el comienzo de la lista, asignamos a lista el valor de lista->anterior mientras lista->anterior no sea NULL.
* Abriremos un bucle que al menos debe tener una condición, que el índice no sea NULL.
* Dentro del bucle asignaremos a lista el valor del nodo siguiente al actual.

**Eliminar Nodo**

De nuevo tenemos los dos casos posibles, que el nodo a borrar esté apuntado por Lista o que no. Si lo está, simplemente hacemos que Lista sea Lista->anterior, si no es NULL o Lista->siguiente en caso contrario.

* Si nodo apunta a Lista,
  + Si Lista->anterior no es NULL hacemos que Lista apunte a
  + Lista->anterior. Si Lista->siguiente no es NULL hacemos que Lista apunte a NULL
  + Lista->siguiente. Si ambos son NULL, hacemos que Lista sea NULL.
* Si nodo->anterior no es NULL, hacemos que nodo->anterior->siguiente apunte a nodo->siguiente.
* Si nodo->siguiente no es NULL, hacemos que nodo->siguiente->anterior apunte a nodo->anterior.
* Borramos el nodo apuntado por nodo.

****

**Figura 42** *Diagrama de la eliminación de un nodo de una lista*

**Listas circulares Simples**

**Insertar un elemento en una lista circular vacía.**

Condiciones Iniciales:

* Disponer de un nodo a ser insertado.

****

**Figura 43** *Nodo que se desea insertar*

* Disponer de un puntero que defina la lista, el cual valdrá NULL.

****

**Figura 44** *Puntero que apunta a Null*

Proceso:

* Integrar en un solo diagrama las condiciones iniciales.

****

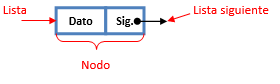
**Figura 45** *Diagrama del nodo a ingresar y el puntero*

* Hacer que Lista apunte al nodo.

****

**Figura 46** *Puntero apunta al nodo*

* Lista=nodo

****

**Figura 47** *El nuevo nodo se apunta a si mismo*

* Lista->siguiente apunte al nodo.

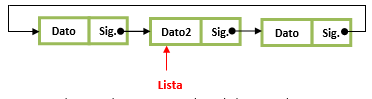
****

**Figura 48** *Nodo insertado en la lista vacía*

**Insertar un elemento en una lista circular no vacía**

**Condiciones Iniciales:**

* Disponer de una lista circular y de un apuntador denominado lista el cual apunte a cualquier nodo de la lista.

****

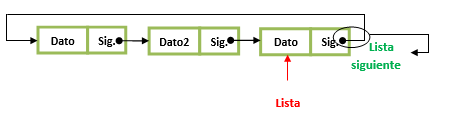
**Figura 49** *Diagrama de la lista circular*

* Disponer de un nodo a ser insertado en la lista circular.

****

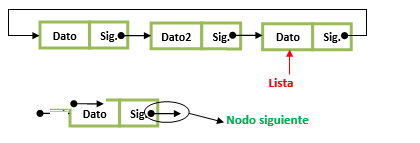
**Figura 50** *Nodo que se desea insertar*

* Ubicar el apuntador a la lista apuntando al nodo en donde va a ser insertado el nuevo nodo.

****

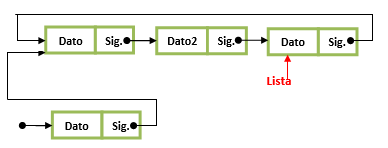
**Figura 51** *El apuntador apunta a la ubicación donde va a ser insertado*

* Integrar las condiciones iniciales.

****

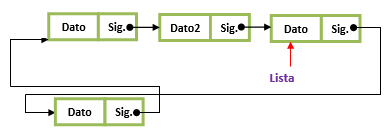
**Figura 52** *Diagrama de condiciones iniciales*

* Hacer que nodo siguiente apunte a la cabecera de lista siguiente.

****

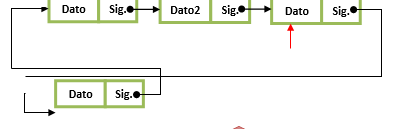
**Figura 53** *Nodo nuevo apunta al nodo que siguiente de donde se va a insertar*

* Hacer que la lista siguiente apunte al nodo a ser insertado

****

**Figura 54** *La lista apunta al nodo insertado*

* Lista->siguiente = nodo

****

**Figura 55** *Inserción del nodo*

**Buscar un nodo en una lista circular**

La operación buscar en una lista simple circular es similar a la operación buscar en una lista simple lineal, con la diferencia de que la lista circular no tiene un final. Esto significa que, al llegar al final de la lista, se debe volver al primer nodo para continuar la búsqueda.

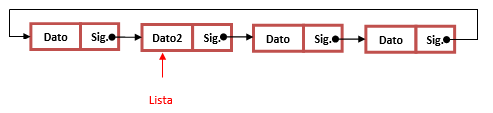
**Eliminar Nodo**

* Disponer una lista circular

****

**Figura 56** *Lista circular*

* Hacer que el apuntador auxiliar lista apunte al nodo anterior que deseamos eliminar.

****

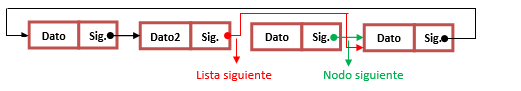
**Figura 57** *El puntero auxiliar apunta a la posición anterior donde se desea hacer la inserción*

* Hacer que el apuntador lista valga lista -> siguiente, mientras lista siguiente sea distinto de nodo.

****

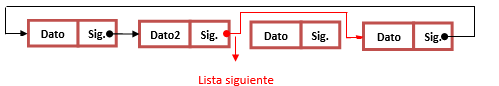
**Figura 58** *Lista ya no apunta al nodo que se desea eliminar*

* Hacer que lista->siguiente apunte a nodo->siguiente

****

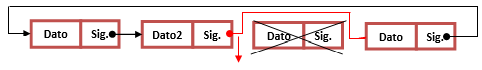
**Figura 59** *Diagrama del comportamiento lista*

* Lista->siguiente toma la dirección de nodo->siguiente

****

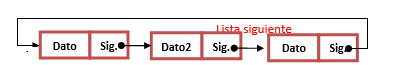
**Figura 60** *Los nodos dejan de apuntar al nodo que se desea eliminar*

* Eliminar el nodo

****

**Figura 61** *Eliminación del nodo*

* Presentar la nueva lista.

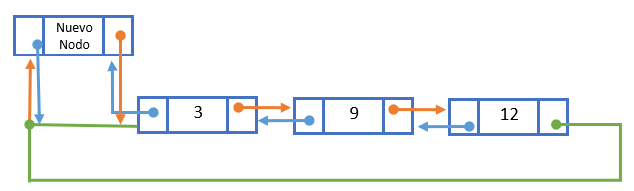
****

**Figura 62** *Diagrama de la lista sin el nodo*

**Listas Dobles circulares**

**Insertar Nuevo nodo a la lista**

* Crear un nuevo nodo. El nuevo nodo debe tener un puntero al nodo anterior y otro al nodo siguiente.

****

**Figura 63** *Diagrama de la inserción de un nodo a una lista doble circular*

* Obtener el nodo anterior y siguiente del punto de inserción. Podemos obtener estos nodos iterando por la lista hasta llegar al punto de inserción.
* Realizar los enlaces necesarios. Los enlaces necesarios son los siguientes:
  + El puntero al nodo anterior del nuevo nodo se establece en el nodo anterior del punto de inserción.
  + El puntero al nodo siguiente del nuevo nodo se establece en el nodo siguiente del punto de inserción.
  + El puntero al nodo siguiente del nodo anterior del punto de inserción se establece en el nuevo nodo.
  + El puntero al nodo anterior del nodo siguiente del punto de inserción se establece en el nuevo nodo.

**Eliminar Nodo**

* Obtener el nodo a eliminar. Podemos obtener el nodo a eliminar iterando por la lista hasta encontrar el nodo que queremos eliminar.
* Realizar los enlaces necesarios. Los enlaces necesarios son los siguientes:
* El puntero al nodo siguiente del nodo anterior del nodo a eliminar se establece en el nodo siguiente del nodo a eliminar.
* El puntero al nodo anterior del nodo siguiente del nodo a eliminar se establece en el nodo anterior del nodo a eliminar.

**Buscar Nodo**

Pasos para buscar un elemento en una lista circular doble:

* Inicializar un nodo auxiliar. El nodo auxiliar se usará para iterar por la lista.
* Inicializar el nodo auxiliar al primer nodo de la lista.
* Recorrer la lista. Mientras el nodo auxiliar no sea Nodo y el elemento del nodo auxiliar no sea igual al elemento que buscamos, avanzaremos al siguiente nodo de la lista.
* Si el elemento del nodo auxiliar es igual al elemento que buscamos, devolver el nodo auxiliar.
* Si el nodo auxiliar es Nodo, devolver Nodo.

**Referencias**

[1] Abstraction and Specification in Program Development, Barbara Liskov and John Guttag.

[2] Data Structures and Algorithms in Java, Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser.