

Listas Enlazadas

Estructuras de Datos

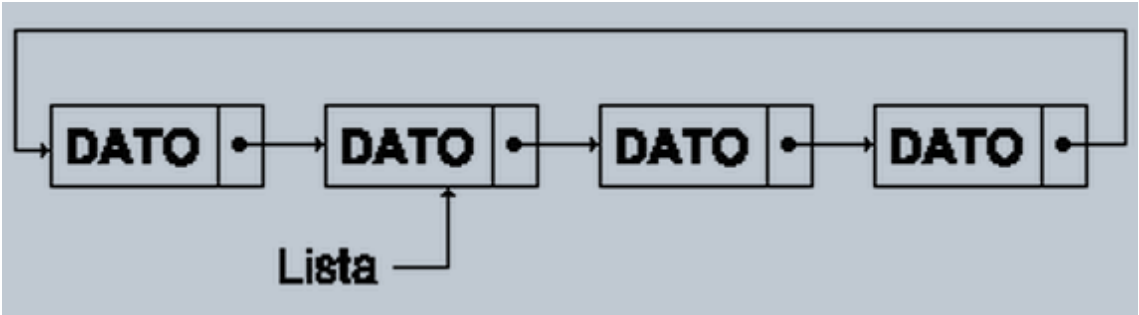
1

Listas Circulares

- Listas Circulares Simples
- Listas Circulares Dobles

Listas Circulares simples

- Una *lista circular* es una lista lineal en la que el último nodo apunta al primero.
- No existen casos especiales, cada nodo siempre tiene uno anterior y uno siguiente.
- Se añade un nodo especial de cabecera, de ese modo se evita la única excepción posible, la de que la lista esté vacía.



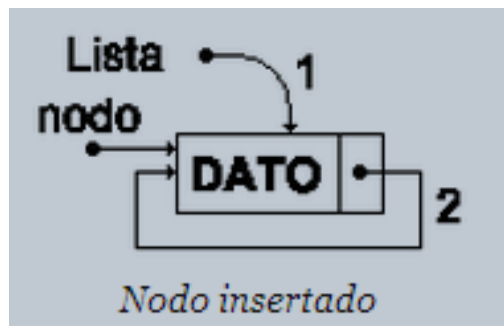
Operaciones básicas con listas circulares

Las listas circulares son como las listas abiertas en cuanto a las operaciones que se pueden realizar sobre ellas:

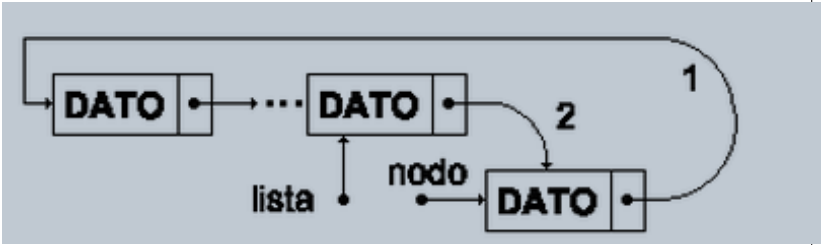
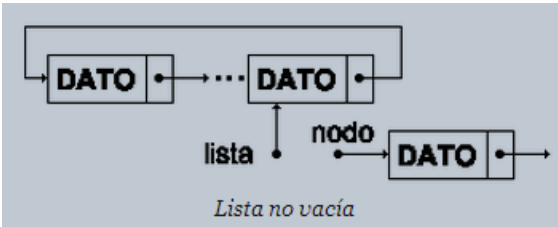
- Añadir o insertar elementos.
- Buscar o localizar elementos.
- Borrar elementos.
- Moverse a través de la lista, siguiente.

Añadir un elemento

- El único caso especial a la hora de insertar nodos en listas circulares es cuando la lista esté vacía.



Inserción en una lista no vacía



Buscar o localizar un elemento de una lista circular

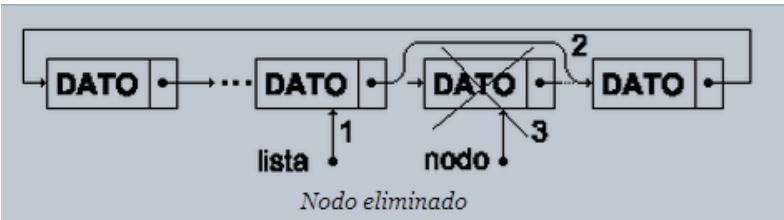
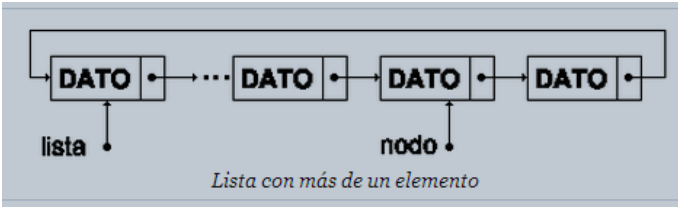
- Para buscar elementos en una lista circular sólo hay que tener una precaución:

es necesario almacenar el indicador del nodo en que se empezó la búsqueda, para poder detectar el caso en que no exista el valor que se busca.

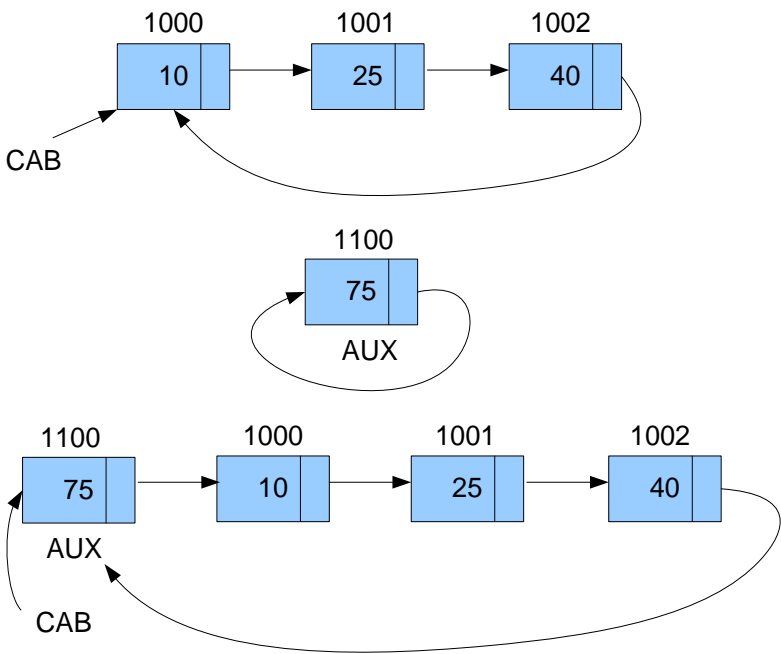
Eliminar un elemento de una lista circular

- Existen tres casos diferentes:
 1. Eliminar un nodo que no sea el señalado por lista.
 2. Eliminar el nodo señalado por lista, y que no sea el único nodo.
 3. Eliminar el único nodo de la lista.
- En el primer caso se necesita localizar el nodo anterior al que se quiere borrar. Como el principio de la lista puede ser cualquier nodo, entonces será **lista** quien apunte al nodo anterior al que se va a eliminar.
- Esto anula la excepción del segundo caso, ya que lista nunca será el nodo a eliminar, salvo que sea el único nodo de la lista.
- Una vez localizado el nodo anterior y apuntado por lista, se hará que **lista.next** señalea **nodo.next**. Y a continuación se borra nodo.
- En el caso de que sólo exista un nodo, será imposible localizar el nodo anterior, así que simplemente se elimina el nodo, y se hace que **lista** valga NULL.

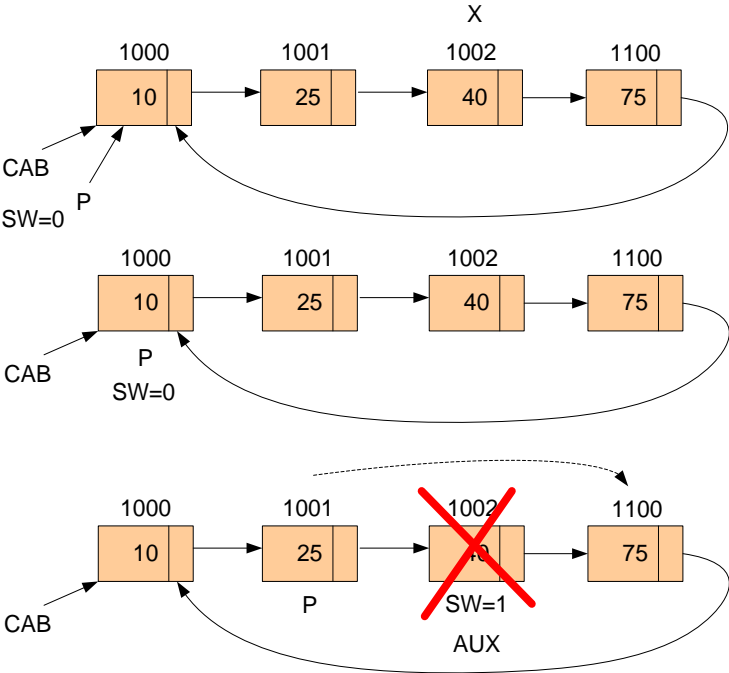
Eliminación en lista con varios nodos:



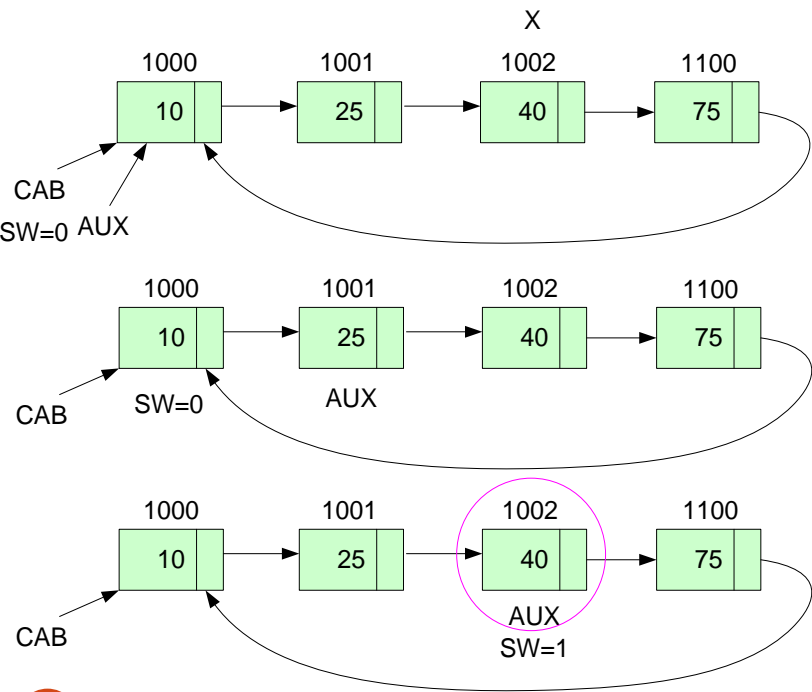
OPERACIONES: INSERTAR AL INICIO



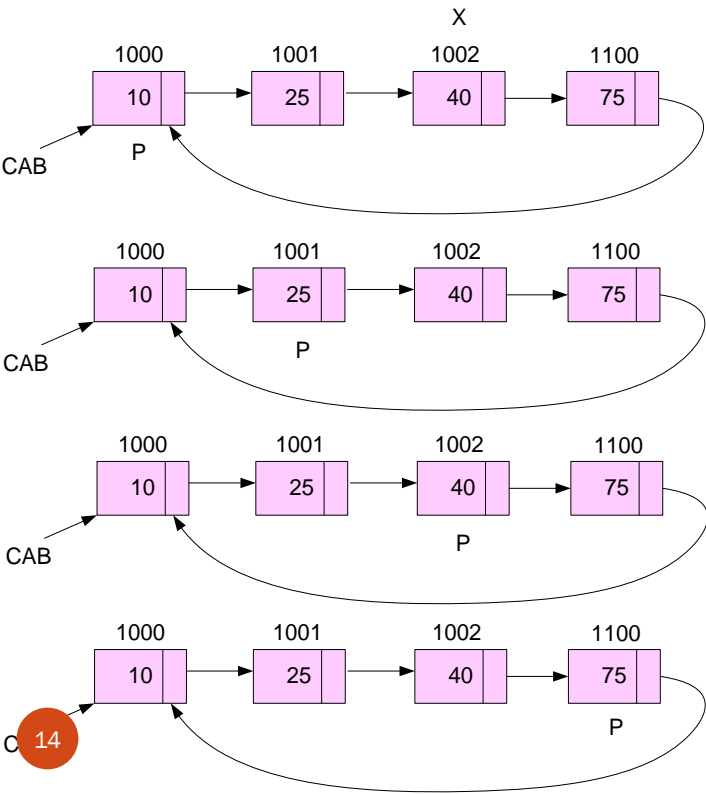
OPERACIONES: ELIMINAR UN ELEMENTO



OPERACIONES: BUSCAR UN ELEMENTO



OPERACIONES: MOSTRAR ELEMENTOS



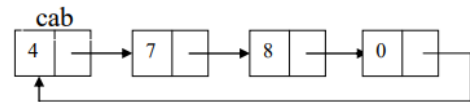
P=10
Mostrar 10
P=25
Mostrar 25
P=40
Mostrar 40
P=75
Mostrar 75

CLASE Nodo.

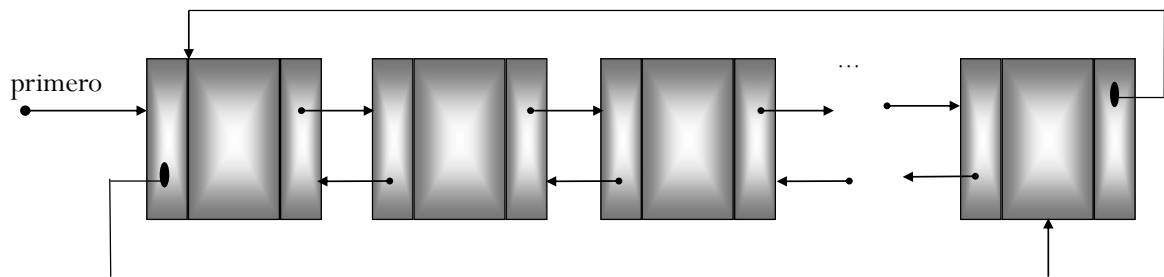
```
public class Nodo {
    int info;
    Nodo enlace;
    public Nodo(int a)
    {
        info=a;
    }
}
```

CLASE ListaC,

```
public class ListaC {
    Nodo cab;
    void insertar(int a){
        if(cab==null){
            cab=new Nodo(a);
            cab.enlace=cab;
        }
        else{
            Nodo temp=new Nodo(a);
            Nodo aux=cab;
            while(aux.enlace!=cab)
                aux=aux.enlace;
            aux.enlace=temp;
            temp.enlace=cab;
        }
    }
    void mostrar() {
        if(cab!=null){
            Nodo aux=cab;
            do
            {
                System.out.print(aux.info+" ");
                aux=aux.enlace;
            }
            while(aux!=cab);
        }
    }
}
```



Lista doblemente enlazada circular



El campo *siguiente* del último nodo apunte al primer nodo de la lista y el campo anterior del primer nodo apunte al último nodo de la lista.

Ventajas y desventajas

- **Ventajas**

- No es preciso conocer la cantidad de elementos en tiempo de compilación.
- Ni las inserciones ni las eliminaciones implican realizar corrimientos de los elementos de la lista.

- **Desventajas**

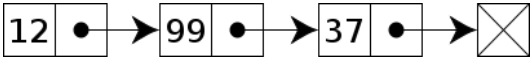
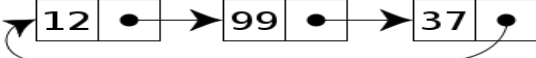
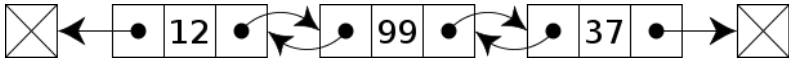
- No permite el acceso directo a un elemento arbitrario de la lista. Para acceder al *i-ésimo* elemento, debemos recorrer la lista, comenzando por el primer nodo, hasta llegar al elemento deseado.

CLASIFICACIÓN DE LAS LISTAS ENLAZADAS

Simplemente enlazadas	Doblemente enlazadas	Circular simplemente enlazada	Circular doblemente enlazada
Cada nodo (elemento) contiene un único enlace que conecta ese nodo al nodo siguiente o nodo sucesor. La lista es eficiente en recorridos directos («adelante»).	Cada nodo contiene dos enlaces, uno a su nodo predecesor y el otro a su nodo sucesor. La lista es eficiente tanto en recorrido directo («adelante») como en recorrido inverso («atrás»).	Una lista enlazada simplemente en la que el último elemento (cola) se enlaza al primer elemento (cabeza) de tal modo que la lista puede ser recorrida de modo circular («en anillo»).	Una lista doblemente enlazada en la que el último elemento se enlaza al primer elemento y viceversa. Esta lista se puede recorrer de modo circular (en anillo) tanto en dirección directa («adelante») como inversa («atrás»).

Por cada uno de estos cuatro tipos de estructuras de listas, se puede elegir una implementación basada en arrays (asignación fija o estática) o una implementación basada en punteros (asignación dinámica de memoria mediante punteros).

TIPOS DE LISTAS ENLAZADAS

TIPO	NODO	LISTA
Simple	Dato sig	
Simple circular	Ant dato sig	
Doblememnte enlazada	Ant dato sig	
Doblemente enlazada circular	Ant dato sig	