Estructura de Datos

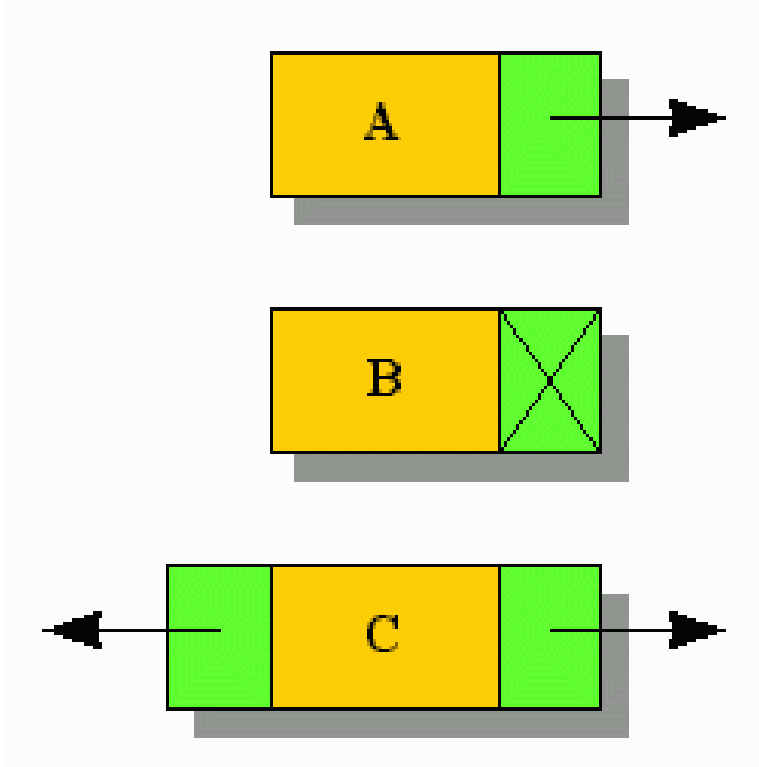
Listas Enlazadas

Conceptos de Lista enlazada

- Una **lista enlazada** es una secuencia de nodos que se interconectan mediante sus campos de enlace.
- **Nodo:** un objeto creado desde una clase auto-referenciada.
- **Clase auto-referenciada:** una clase con al menos un campo cuyo tipo de referencia es el nombre de la clase.
- **Campo de enlace:** un campo cuyo tipo de referencia es el nombre de la clase.
- **Enlace**: la referencia a un campo de enlace.

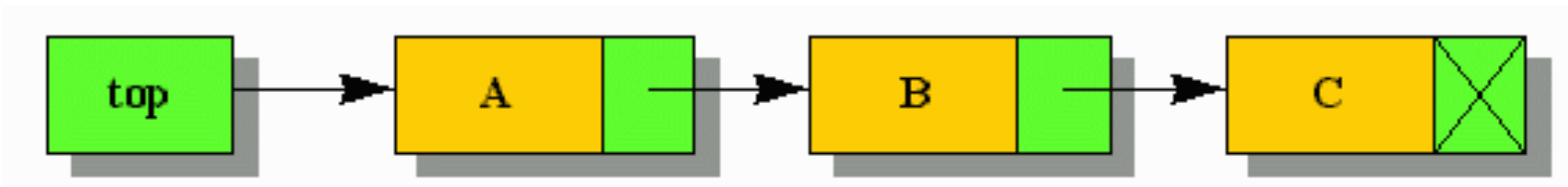
Características de un Nodo

- Cada nodo se divide en áreas de contenido (en naranja) y una o más áreas de enlace (en verde).
- Las áreas de contenido representan todos los campos que no son enlaces, y cada área de enlace representa un campo de enlace.
- Las áreas de enlace de A y C tienen unas flechas para indicar que referencian a otro nodo del mismo tipo (o subtipo).
- La única área de enlace de B incorpora una X para indicar una referencia nula. En otras palabras, B no está conectado a ningún otro nodo.



Lista enlazada Simple

- Una **lista de enlace simple** es una lista enlazada de nodos, donde cada nodo tiene un único campo de enlace.
- Una variable de referencia contiene una referencia al primer nodo, cada nodo (excepto el último) enlaza con el nodo siguiente, y el enlace del último nodo contiene null para indicar el final de la lista.

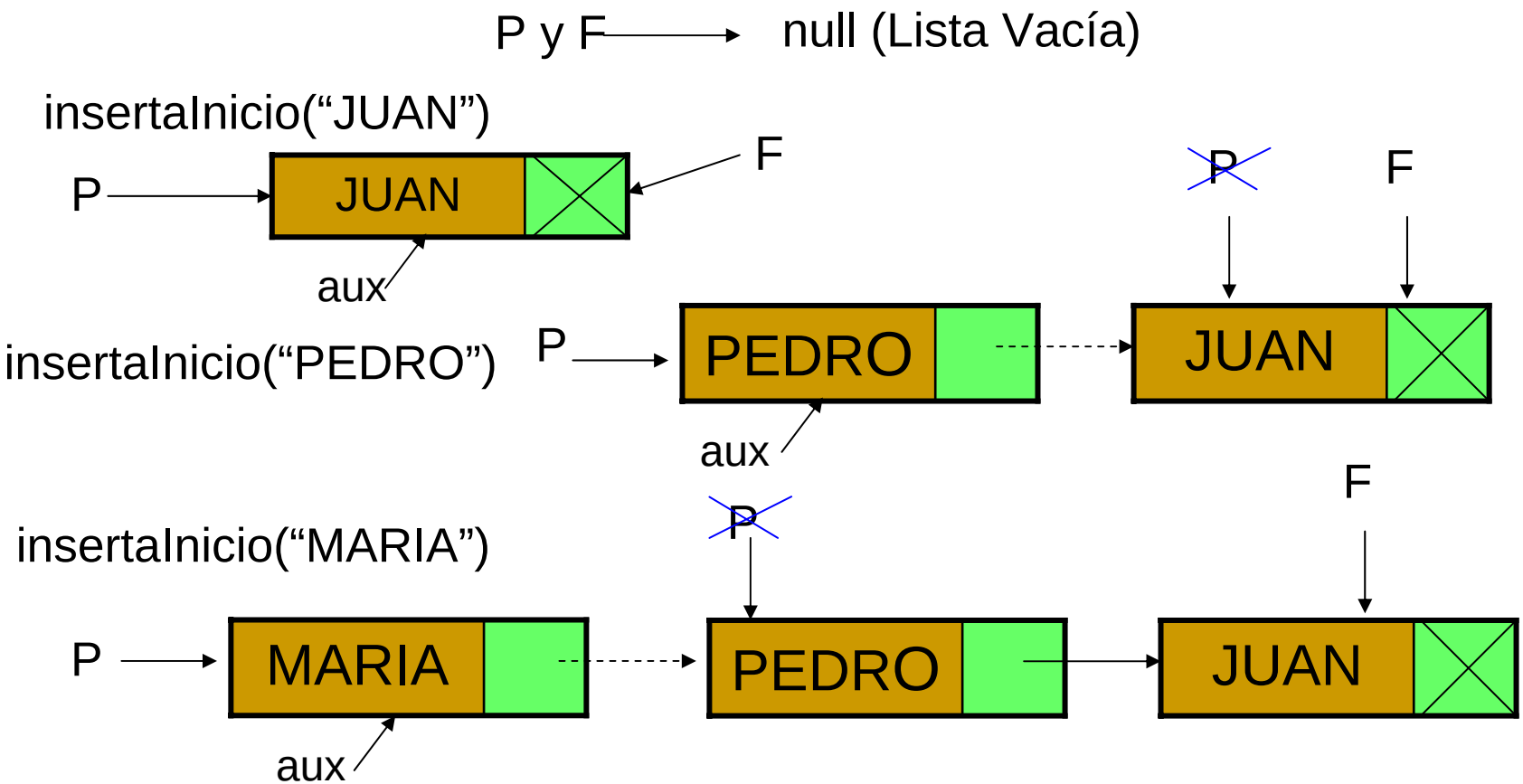


Operaciones en Listas Enlazadas

- Inserción de un nodo:
  - Al inicio de la Lista
  - Al final de la Lista
  - Antes de un nodo específico
  - Después de un nodo específico
- Eliminación de un nodo:
  - Al inicio de la Lista
  - Al final de la Lista
  - Un nodo específico
- Recorrido de la lista
- Búsqueda de un elemento

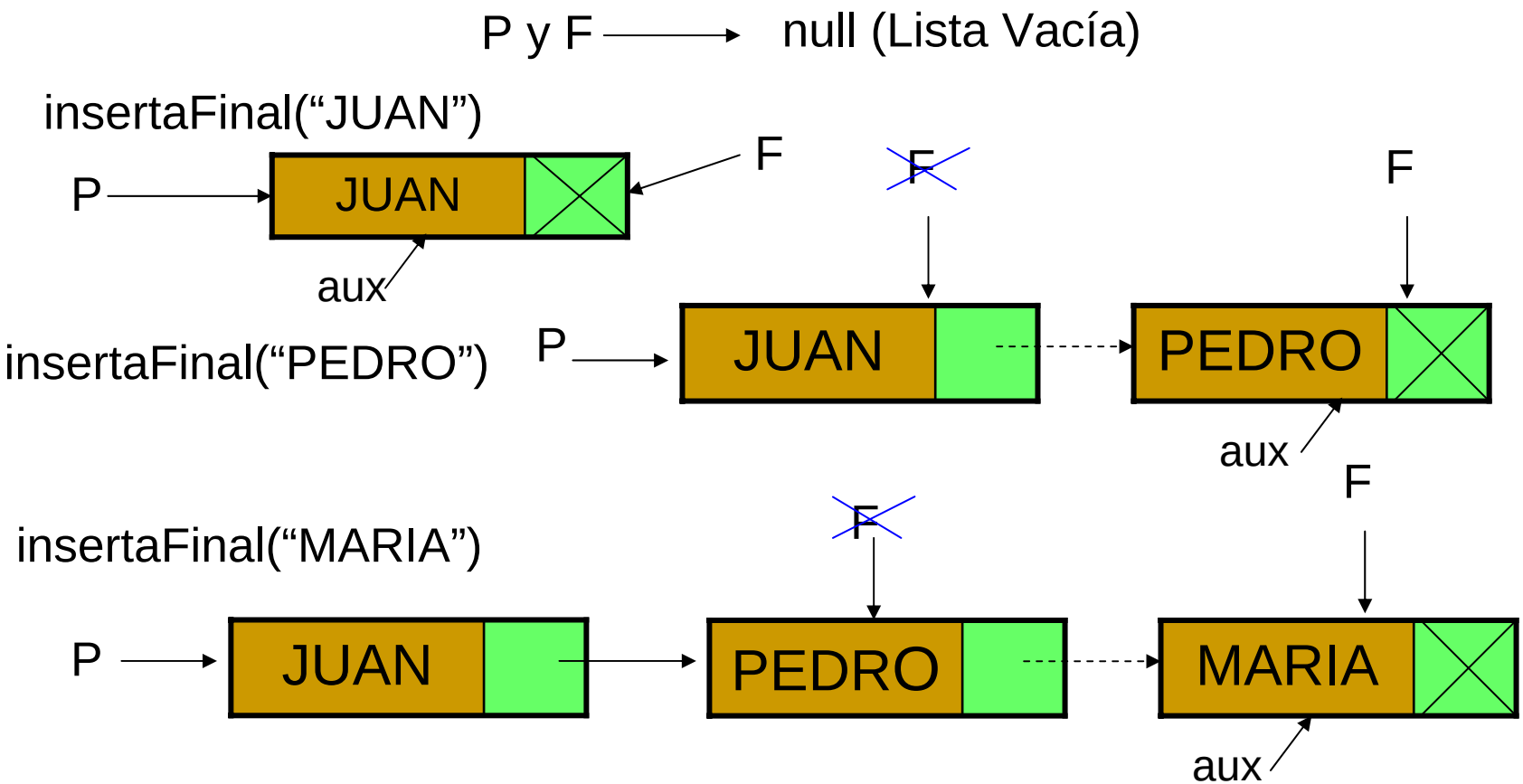
Inserción de un nodo al inicio de la Lista

El nuevo nodo se coloca al principio de la lista, convirtiéndose en el primero de la misma.



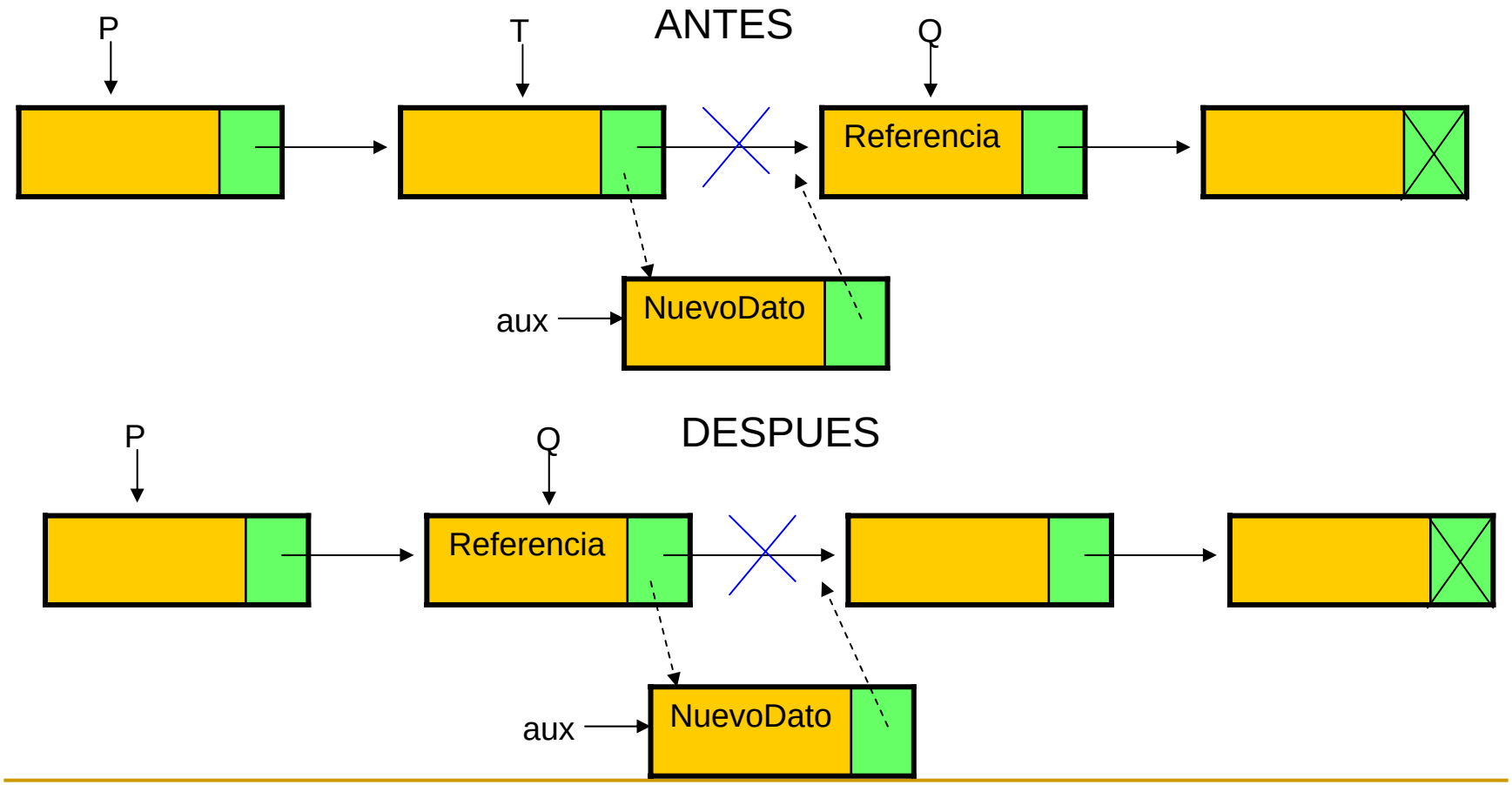
Inserción de un nodo al final de la Lista

El nuevo nodo se coloca al final de la lista, convirtiéndose en el último de la misma.



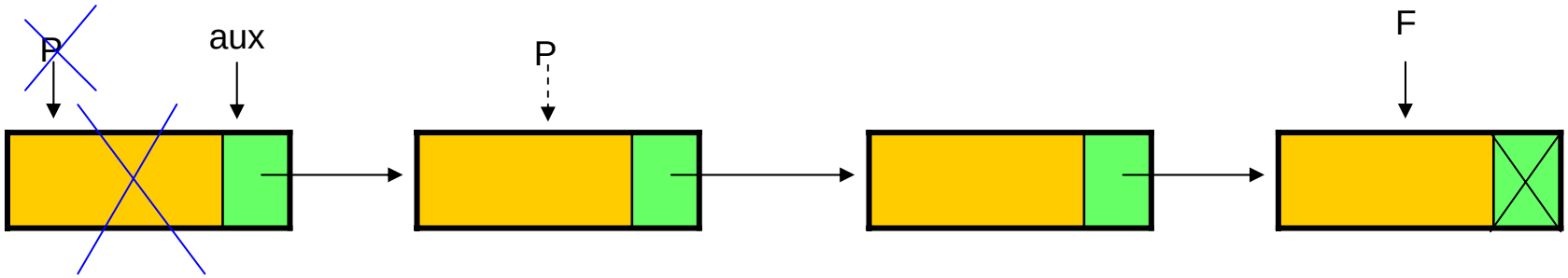
Inserción de un nodo antes/después que otro

El nuevo nodo se coloca antes (después) de otro nodo dado como referencia. Primero se tiene que realizar una búsqueda del nodo dado como referencia.

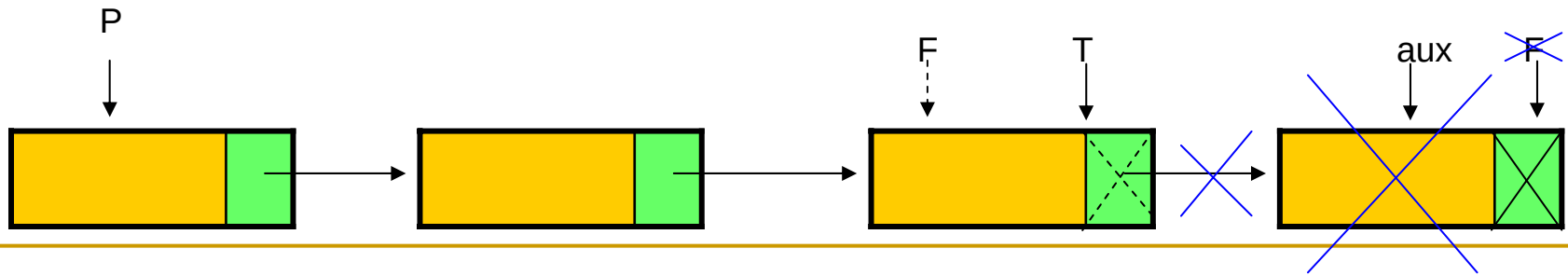


Eliminación de un elemento o nodo

Eliminación primer nodo: se quita el primer elemento de la lista, redefiniendo el valor del puntero al nuevo inicio de la lista.

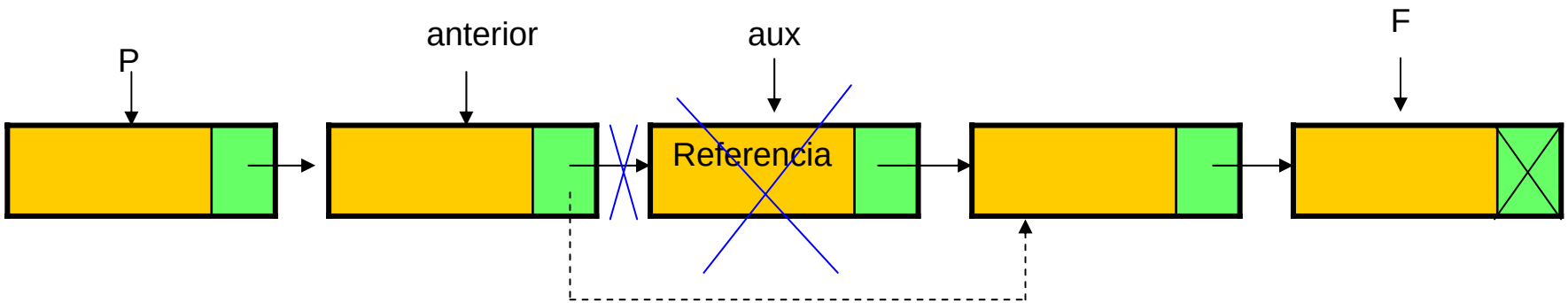


Eliminación del último nodo: se quita el último nodo de la lista, redefiniendo a null el campo liga o enlace del penúltimo nodo de la lista. Para encontrar el penúltimo nodo es necesario recorrer la lista. Se redefine el valor del puntero que indica el fin de la lista.



Eliminación de un elemento o nodo

Eliminación un nodo con información X: se debe buscar al nodo que contenga la información dada como referencia (X) y eliminarlo, estableciendo la liga o enlace correspondiente entre su predecesor y su sucesor.



Recorrido de la lista

- La operación de recorrido consiste en visitar cada uno de los nodos que forman la lista.
- La visita de un nodo puede definirse por medio de una operación muy simple (como escribir su valor), o por medio de operaciones tan complejas como se desee.
- Para recorrer todos los nodos de una lista se comienza con el primero. Tomando el valor del campo enlace o liga se avanza por cada nodo hasta que dicho enlace sea igual a nulo.

```
public void recorrerLista(NodoSimple  p){
NodoSimple aux;
aux=p;
while (aux!=null) {
        System.out.println(aux.info);
      aux=aux.enlace;}
        }
```

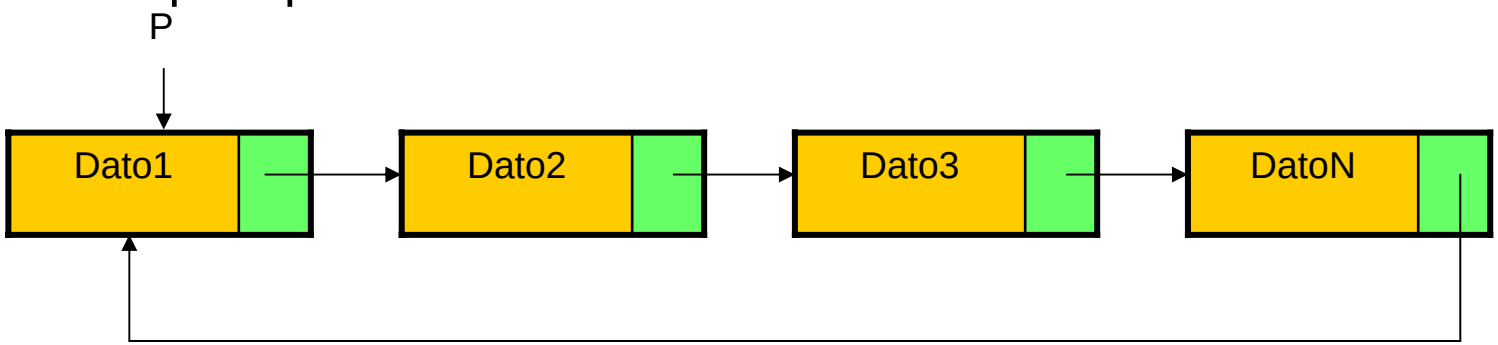
Búsqueda de un elemento

- La operación de búsqueda de un elemento en una lista se realiza de modo secuencial.
- Se deben recorrer los nodos, tomando el campo enlace o liga como referencia al siguiente nodo a visitar.
- La búsqueda puede optimizarse cuando la lista esta ordenada.

```
public void buscarNodo(NodoSimple  p, TipoInfo clavex){
    NodoSimple aux;
    aux=p;
    while (aux!=null) {
            if (aux.clave==clavex) break; //Se encontro valor
             else aux=aux.enlace;}
     if (aux==null) System.out.println("No se encuentra");
      else System.out.println("El elemento esta en la lista");
           }
```

Listas Circulares

- Una lista circular es una lista lineal en la que el último nodo apunta al primero.
- Las listas circulares evitan excepciones en la operaciones que se realicen sobre ellas. No existen casos especiales, cada nodo siempre tiene uno anterior y uno siguiente.
- A la hora de buscar elementos en una lista circular sólo hay que tener una precaución, es necesario almacenar el puntero del nodo en que se empezó la búsqueda, para poder detectar el caso en que no exista el valor que se busca. Por lo demás, la búsqueda es igual que en el caso de las listas simples, salvo que podemos empezar en cualquier punto de la lista.



Consideraciones en Listas Circulares

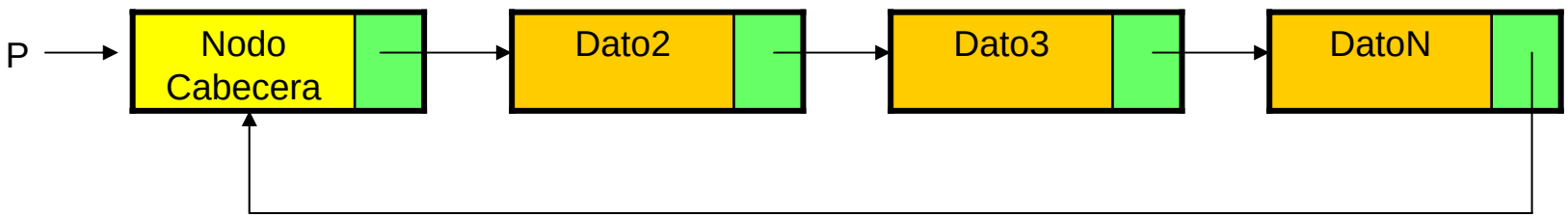
- A pesar de que las listas circulares simplifiquen las operaciones sobre ellas, también introducen algunas complicaciones. Por ejemplo, en un proceso de búsqueda, no es tan sencillo dar por terminada la búsqueda cuando el elemento buscado no existe.

- En el caso de la operación de recorrido es necesario aclarar que se debe considerar algún criterio para detectar cuándo se han visitado todos los nodos para evitar caer en ciclos infinitos.

- Por ese motivo se suele resaltar un nodo en particular, que no tiene por qué ser siempre el mismo. Cualquier nodo puede cumplir ese propósito, y puede variar durante la ejecución del programa.

- Otra alternativa que se usa a menudo, y que simplifica en cierto modo el uso de listas circulares es crear un nodo especial de hará la función de nodo cabecera. De este modo, la lista nunca estará vacía, y se eliminan casi todos los casos especiales.

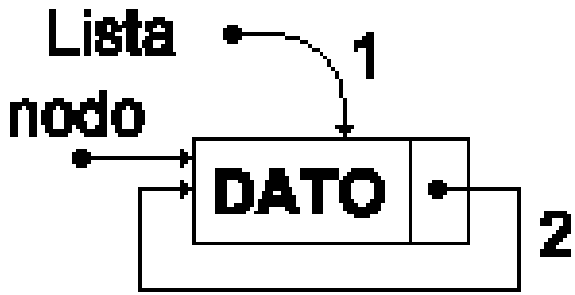
- Un nodo cabecera indica el inicio de la lista y contendrá información especial, de tal manera que se distinga de los demás y así podrá hacer referencia al principio de la lista.



Inserción un elemento en una lista circular

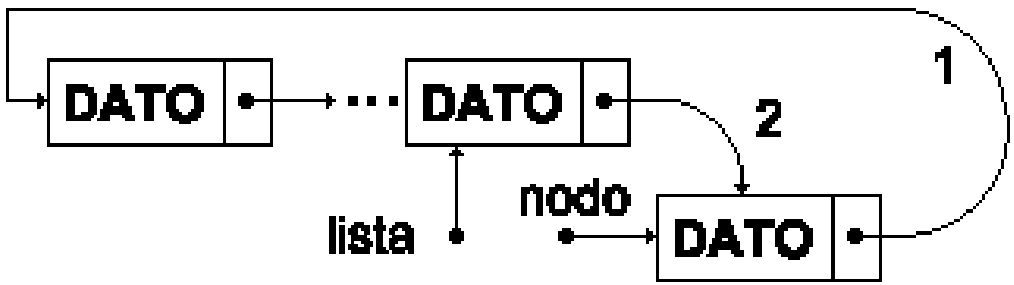
Lista vacía:

1. lista apunta a nodo.
2. lista.siguiente apunte a nodo.



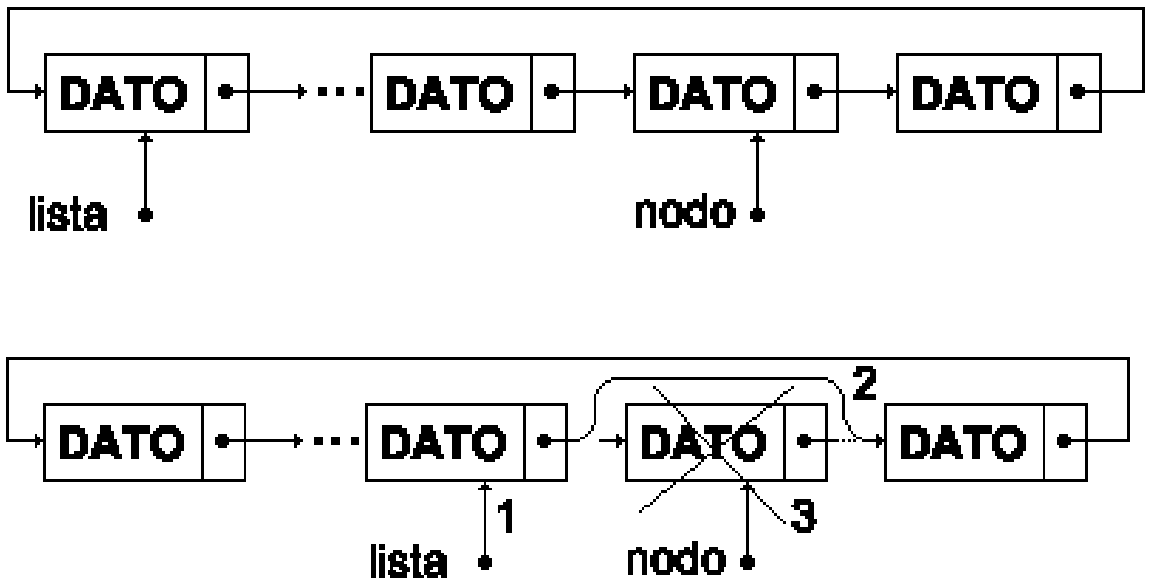
Lista no vacía:

1. Hacemos que nodo.siguiente apunte a lista.siguiente.
2. Después que lista.siguiente apunte a nodo.



Eliminar un elemento en una lista circular

1. El primer paso es conseguir que lista apunte al nodo anterior al que queremos eliminar. Esto se consigue haciendo que lista valga lista.siguiente mientras lista.siguiente sea distinto de nodo.
2. Hacemos que lista.siguiente apunte a nodo.siguiente.
3. Eliminamos el nodo.

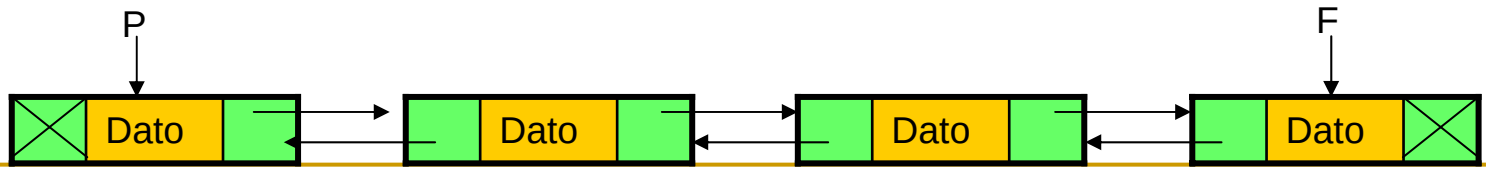


Nota: Hay considerar que cuando la lista circular tiene un único nodo después de eliminar dicho nodo hay que hacer lista igual a nulo.

Listas doblemente enlazadas

- Una lista doblemente enlazada es una colección de nodos, en la cual cada nodo tiene dos enlaces, uno al nodo siguiente o sucesor (enlaceIzq), y otro a su predecesor o anterior (enlaceDer).

- Las listas doblemente enlazadas no necesitan un nodo especial para acceder a ellas, pueden recorrerse en ambos sentidos a partir de cualquier nodo, esto es porque a partir de cualquier nodo, siempre es posible alcanzar cualquier nodo de la lista, hasta que se llega a uno de los extremos.
- Por medio de estos enlaces se puede recorrer la lista en ambos sentidos.
- Se recomienda se mantengan dos referencias o punteros P y F, que apunten al primero y último nodo de la lista respectivamente.



Algoritmo para Inserción de un elemento en una lista doblemente enlazada

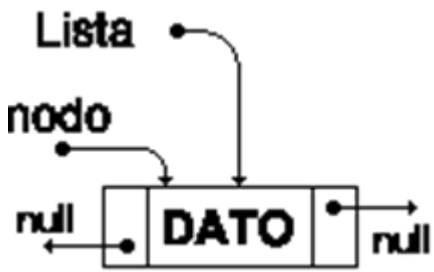
1. Si lista está vacía hacemos que Lista apunte a nodo. Y nodo.anterior y nodo.siguiente a NULL.
2. Si lista no está vacía, hacemos que nodo.siguiente apunte a Lista.siguiente.
3. Después que Lista.siguiente apunte a nodo.
4. Hacemos que nodo.anterior apunte a Lista.
5. Si nodo.siguiente no es NULL, entonces hacemos que nodo.siguiente.anterior apunte a nodo.

- El paso 1 es equivalente a insertar un nodo en una lista vacía.
- Los pasos 2 y 3 equivalen a la inserción en una lista enlazada corriente.
- Los pasos 4, 5 equivalen a insertar en una lista que recorre los nodos en sentido contrario.

Inserción un elemento en una lista doblemente enlazada

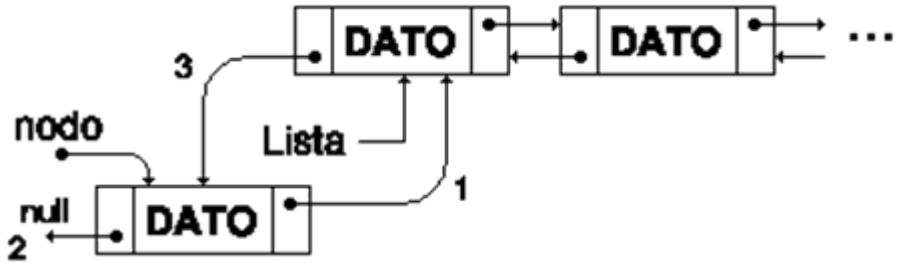
Lista vacía:

1. lista apunta a nodo.
2. lista.siguiente y lista.anterior apunten a null.



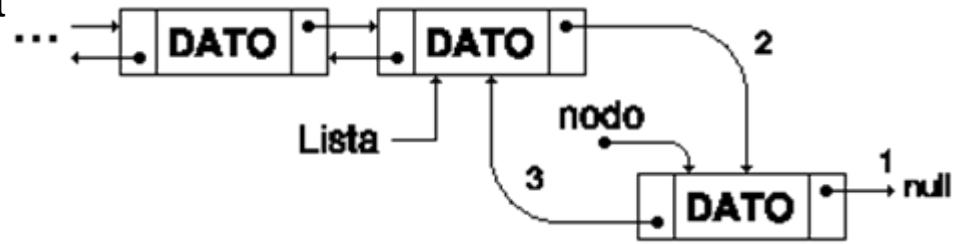
Al Inicio:

1. nodo.siguiente debe apuntar a Lista.
2. nodo.anterior apuntará a lista.anterior.
3. lista.anterior debe apuntar a nodo.



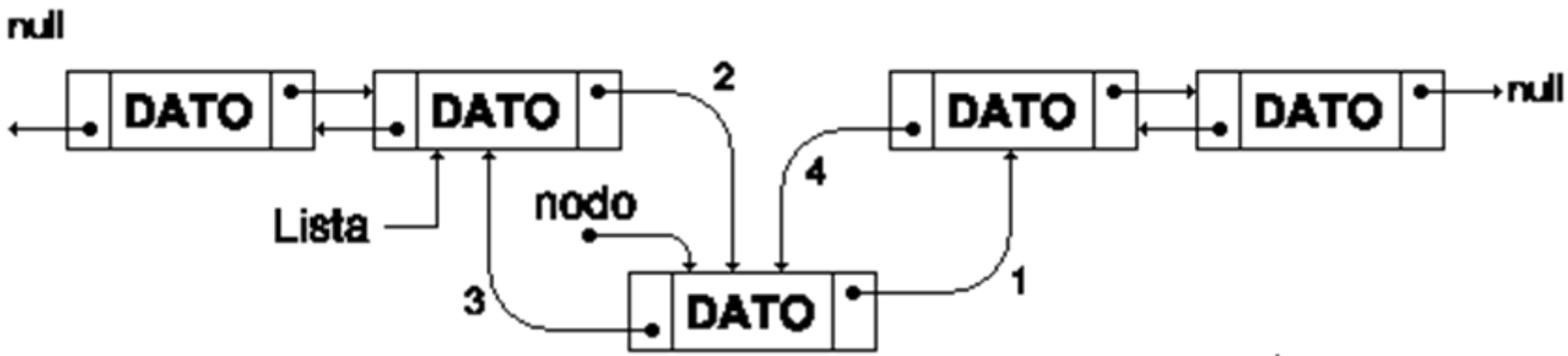
Al Final:

1. nodo.siguiente debe apuntar a Lista.siguiente (NULL).
2. Lista.siguiente debe apuntar a nodo.
3. nodo.anterior apuntará a Lista.



Inserción después de un nodo específico

1. Hacemos que nodo.siguiente apunte a lista.siguiente.
2. Hacemos que Lista.siguiente apunte a nodo.
3. Hacemos que nodo.anterior apunte a lista.
4. Hacemos que nodo.siguiente.anterior apunte a nodo.



Algoritmo para Eliminar un elemento en una lista doblemente enlazada

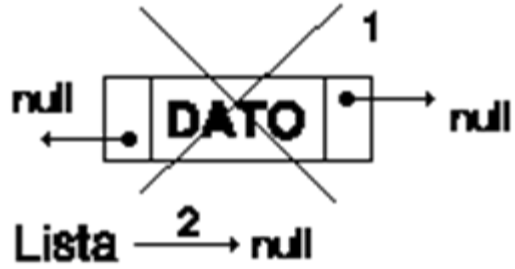
Se tienen dos casos posibles, que el nodo a borrar esté apuntado por Lista o que no. Si lo está, simplemente hacemos que Lista sea Lista.anterior, si no es NULL o Lista.siguiente en caso contrario.

1. 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 Lista.siguiente.
   - Si ambos son NULL, hacemos que Lista sea NULL.
2. Si nodo.anterior no es NULL, hacemos que nodo.anterior.siguiente apunte a nodo.siguiente.
3. Si nodo.siguiente no es NULL, hacemos que nodo.siguiente.anterior apunte a nodo.anterior.
4. Borramos el nodo apuntado por nodo.

Eliminar un elemento en una lista doblemente enlazada

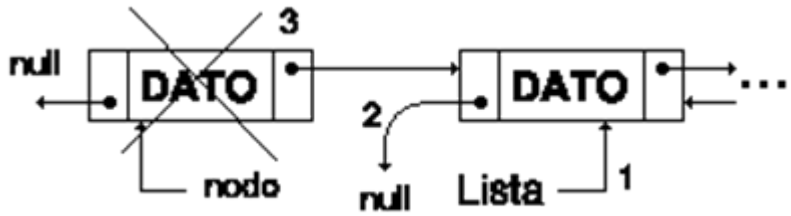
Eliminar un único nodo:

1. Eliminamos el nodo.
2. Hacemos que Lista apunte a NULL.



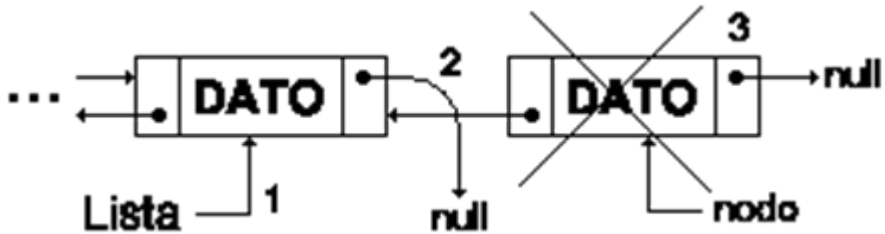
Eliminar el primer nodo:

1. Si nodo apunta a Lista, hacemos que Lista apunte a lista.siguiente.
2. Hacemos que nodo.siguiente.anterior apunte a NULL.
3. Borramos el nodo apuntado por nodo.



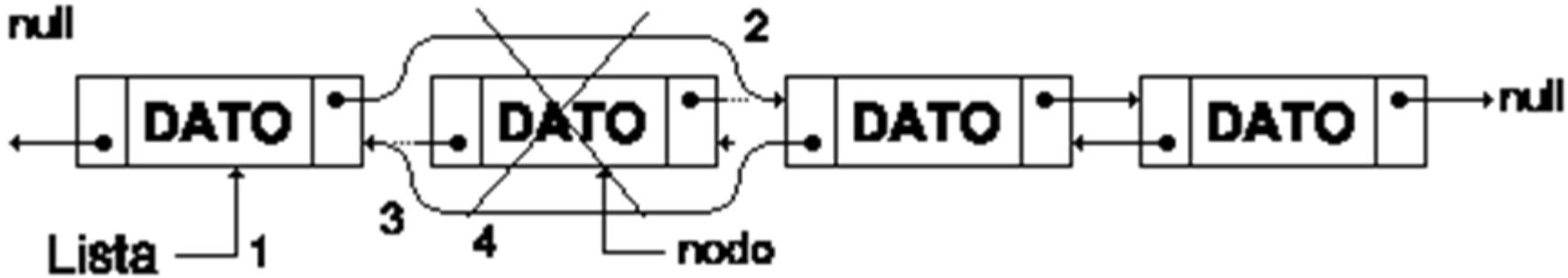
Al Final:

1. Si nodo apunta a Lista, hacemos que Lista apunte a Lista.anterior.
2. Hacemos que nodo.anterior.siguiente apunte a NULL .
3. Borramos el nodo apuntado por nodo.



Eliminar un nodo específico o intermedio

1. Si nodo apunta a Lista, hacemos que Lista apunte a Lista.anterior (o Lista.siguiente).
2. Hacemos que nodo.anterior.siguiente apunte a nodo.siguiente.
3. Hacemos que nodo.siguiente.anterior apunte a nodo.anterior.
4. Borramos el nodo apuntado por nodo.

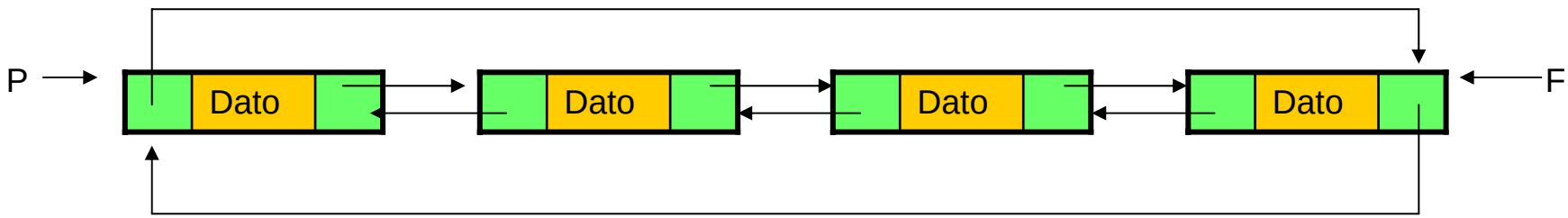


Recorrido y búsqueda de un elemento en una lista doblemente enlazada

- Se tiene la ventaja de que podemos avanzar y retroceder desde cualquier nodo, sin necesidad de volver a uno de los extremos de la lista.
- El recorrido puede ser en el sentido de inicio a final o viceversa.
- Por supuesto, se pueden hacer listas doblemente enlazadas no ordenadas, existen cientos de problemas que pueden requerir de este tipo de estructuras. Pero parece que la aplicación más sencilla de listas doblemente enlazadas es hacer arrays dinámicos ordenados, donde buscar un elemento concreto a partir de cualquier otro es más sencillo que en una lista simple.

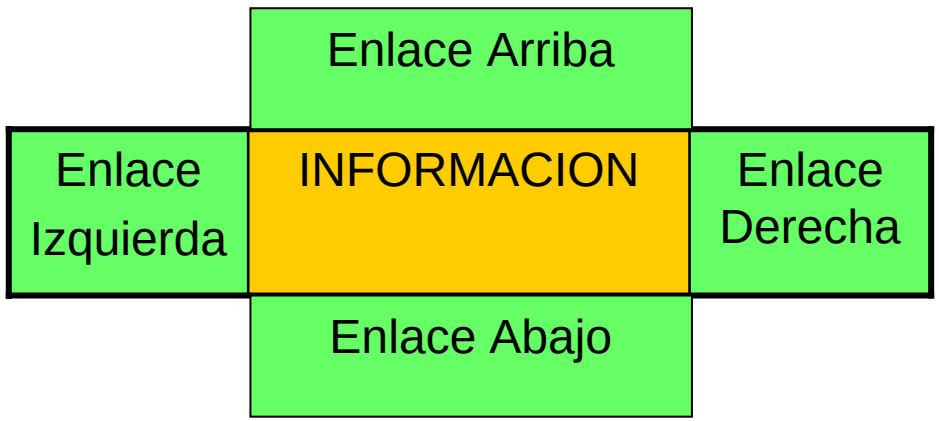
Listas doblemente enlazadas circulares

- En las listas doblemente enlazadas circulares, el campo de enlace izquierdo o anterior del primer nodo de la lista apunta al último, y el campo enlace derecho o siguiente del último nodo apunta al primero.
- La principal ventaja de las listas circulares es que permiten la navegación en cualquier sentido a través de la misma y además, se puede recorrer toda la lista partiendo de cualquier nodo.
- Hay que considerar una manera de controlar el recorrido para evitar caer en ciclos infinitos, ya que no existirán nodos apuntando a nulo. Puede utilizarse un nodo cabecera.



Listas Ortogonales

- En este tipo de lista se utiliza para representar matrices. Los nodos contienen cuatro apuntadores. Uno para apuntar al nodo izquierdo (enlaceIzq),otro para apuntar al derecho (enlaceDer), otro al nodo inferior (enlaceAbajo) y por último un apuntador al nodo superior (enlaceArriba). La estructura de un nodo es:



Aplicaciones de Listas Enlazadas

- Arreglos dinámicos.
- Representación de polinomios.
- Resolución de colisiones Hash.
- Representación de estructuras de datos como colas y pilas.

Tarea:

- Implementar las operaciones básicas para lista simple circular.
- Implementar las operaciones básicas para la listas doblemente enlazadas circulares.
- Implementar las operaciones básicas para una lista ortogonal.