Curso : ESTRUCTURAS DE DATOS
Tema : Estructuras de datos Dinámicas Lineales
Sub-tema : LISTAS DOBLEMENTE ENLAZADAS
Profesor : Braulio Barrios Z.

Definición:
Una lista lineal doblemente enlazada es un conjunto finito de elementos llamados "**Nodos**", tales que:

A.- No existe ningún nodo, entonces, la llamaremos "Lista Nula o Lista Vacía",

Ó

B.- Si tiene nodo, en cuyo caso, cada uno de ellos tendrá la siguiente composición:

EXISTE además, un apuntador externo a la estructura llamado "***CABEZA***" que almacena el byte de inicio del bloque de RAM en el cual se encuentra el PRIMER NODO de la estructura.

EXISTE también, un Valor nulo propio de apuntadores al que llamaremos ***NIL*** (**Ω**) que significa que el nodo que lo contenga en su sección ANTERIOR es el Primer nodo de la lista y de igual manera, aquel nodo que contenga este valor en su sección Siguiente es el Ultimo nodo de la lista.

Observe que cada **NODO** tiene una sección de Información (DATO) y dos (2) apuntadores: El de la Derecha (**SIGUIENTE**) que apunta al nodo siguiente, es decir, contiene una dirección de memoria en hexadecimal que es el byte de inicio de un bloque de memoria en el cual se encuentra almacenado el siguiente nodo de la lista; y el de la Izquierda (**ANTERIOR**) que apunta al nodo anterior, es decir, contiene una dirección de memoria en hexadecimal que es el byte de inicio de un bloque de RAM en el cual se encuentra almacenado el Anterior nodo de la lista.

COMO DECLARAR LA ESTRUCTURA DE INFORMACION PARA UNA LISTA ENLAZADA DOBLE?

Como siempre, se los explico utilizando sintaxis tanto de lenguaje PASCAL como de C++.

Figura 1. Declaración en Lenguaje PASCAL:

```
TYPE
     NODO = ^ARTI;
     ARTI = RECORD
               Refe           : longint;
               Articulo       : string[80];
               SIG            : NODO;
               ANT            : NODO;
     END; (* de la declaración del record *)

(* declaración de variables apuntador a la estructura *)

VAR
   Cabeza, P, Q, FIN : NODO;
```

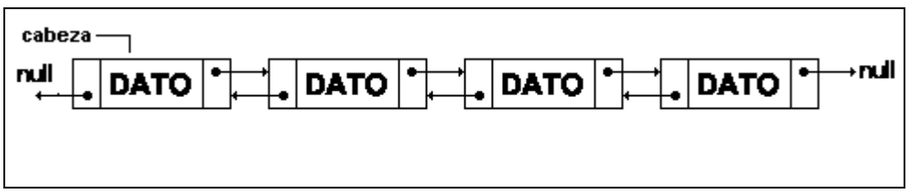
Figura 2. Declaración en Lenguaje C++ :

```
struct NODO
       {
        Long    Refe;
        Char    Articulo[80];
        NODO    *SIG;
        NODO    *ANT;
        };

(* declaración de variables apuntador a la estructura *)
```

```
NODO *Cabeza, *P, *Q, *FIN;
```

De acuerdo con lo anterior, podemos entonces concebir una Estructura de Datos Dinámica Lista Enlazada Doble, como se muestra en el siguiente gráfico:



Según esto, las listas enlazadas dobles, a diferencia de las simples, permiten ser recorridas en ambos sentidos a partir de cualquier nodo hasta llegar a uno de los extremos.

GESTION DE INFORMACION MEDIANTE LISTAS ENLAZADAS DOBLES

Discutiremos, a continuación, como realizar las siguientes tareas de programación sobre una Lista doble, para lo cual utilizaremos la estructura de información descrita en las Figuras 1 y 2.

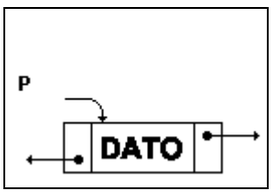
- **Insertar Datos de Artículos.**
- **Insertar Ordenadamente Datos de Artículos**
- **Consultar Artículos.**
- **Retirar (Borrar) Artículos del Inventario**

1.- INSERTAR DE MANERA ORDENADA (ASCENDEMENTE) POR REFERENCIA UN ARTÍCULO EN LA LISTA DOBLE

Al igual que en las listas simples, comenzamos pidiendo memoria al S.O. para el nodo nuevo que queremos insertar en la lista, esto lo hacemos con la sentencia NEW (P), donde P es el apuntador en el cual el S.O me devolverá el byte de inicio del bloque de RAM que ocupará el nodo.

NEW(P)
SI (P= **Ω**) Ent
ESCRIBA("No hay memoria Disponible .....")
SINO
.
.

Si el Sistema Operativo **NO** responde con **NIL** en el apuntador **P** entonces significa que **P** apunta al Nodo nuevo, vacío, listo para introducirle datos



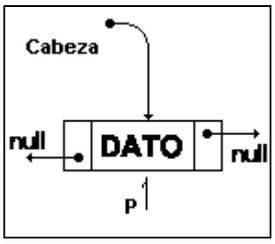
..... Y procedemos a capturarlos por teclado

ESCRIBA("Referencia del articulo ?:"); LEA (P ^.Refe)
ESCRIBA("Nombre del articulo ? :"); LEA (P ^.Articulo)
P ^. ANT ← Ω ;
P ^. SIG ← Ω ;

Ahora debemos "Insertar" ORDENADAMENTE en nuestra lista este nodo que acabamos de llenar con datos.

Si la lista está **VACIA** (o sea si **CABEZA** = **Ω**) entonces este será nuestro *PRIMER NODO* en la lista; en cuyo caso bastará con hacer lo siguiente:

CAB ← P (* pone la cabeza a apuntar al nodo nuevo cuya dirección está en P *)



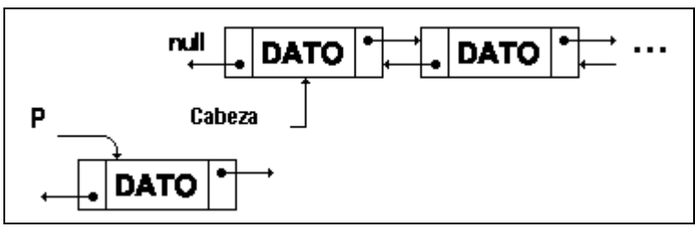
Observe que **CAB ^. SIG** y **CAB^.ANT** están ambos en NIL.

Si la lista NO está vacía entonces las únicas posibilidades que debemos chequear para insertar ORDENADAMENTE nuestro nodo son:

- Insertar el nodo nuevo ANTES de la Cabeza, es decir, en la primera posición
- Insertar el nodo nuevo DESPUES del último
- Insertar el nodo nuevo en cualquier otra posición (En medio de dos nodos)

Insertar un elemento en la primera posición de la lista (ANTES de la cabeza actual):

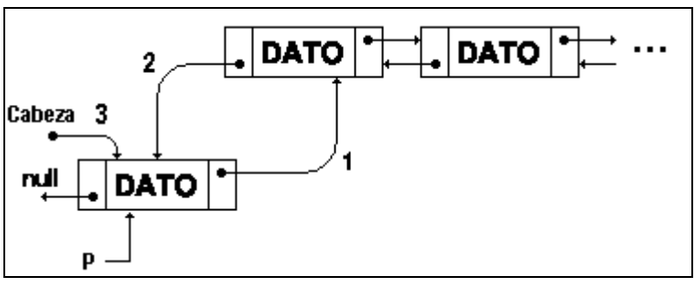
Partimos de una lista NO vacía, y en consecuencia, **Cabeza** apunta al primer nodo de la lista doblemente enlazada. Se entiende que como deseamos crear nuestra lista ordenada ascendentemente por ***referencia*** entonces este artículo que vamos a insertar tendrá una REFERENCIA menor ó igual a la que se encuentre actualmente en el primer nodo de la lista:



El proceso es el siguiente:

1) P ^. SIG ← Cabeza ;
2) Cabeza ^. ANT ← P;
3) Cabeza ← P

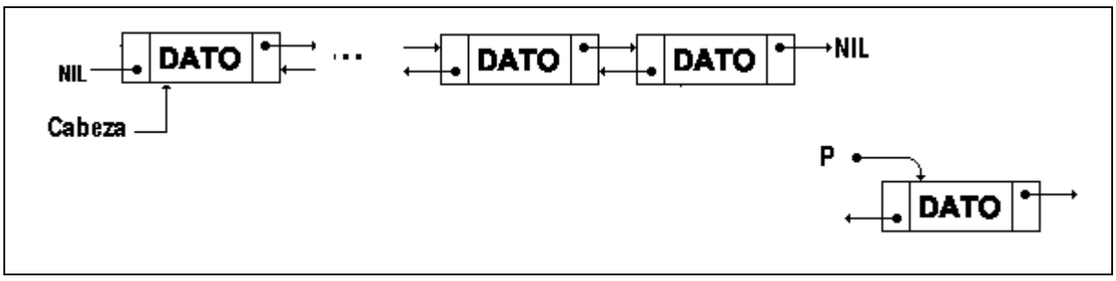
Estos tres pasos y el estado resultante en la lista se muestran en la siguiente ilustración:



Insertar un elemento en la ÚLTIMA posición de la lista (DESPUES del ultimo):

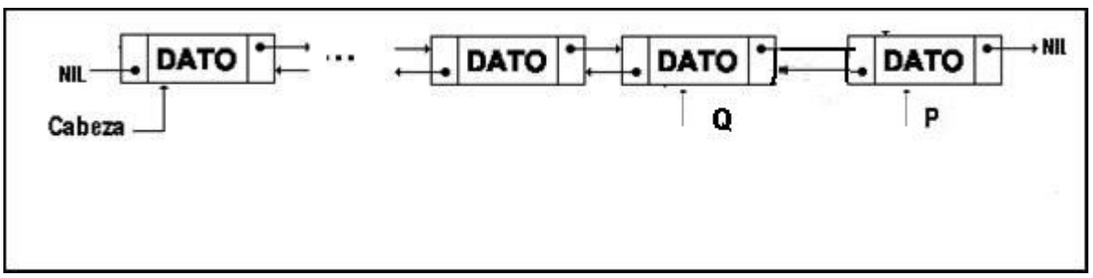
Partimos de una lista NO vacía, y en consecuencia, **Cabeza** apunta al primer nodo de la lista doblemente enlazada. Se entiende que como deseamos crear nuestra lista ordenada ascendentemente por ***referencia*** entonces este artículo que vamos a insertar tendrá una REFERENCIA Mayor ó igual a la que se encuentre actualmente en el Último nodo de la lista, entonces ubicamos un apuntador en el ULTIMO nodo de la lista y

"Conectamos" el nuevo nodo como siguiente del ultimo; Gráficamente se ilustra la situación *antes* del proceso:



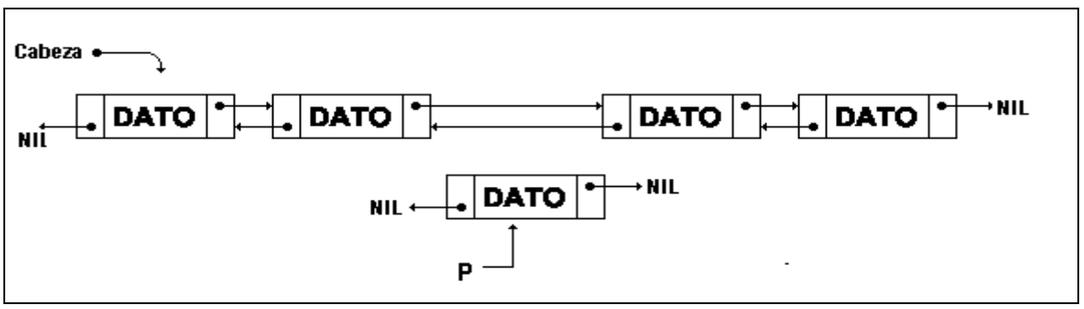
El proceso es el siguiente:

1) **Ubicar apuntador Q en el último nodo**
2) **Q ^. SIG ← P ;**
3) **P^. ANT ← Q**



Insertar un elemento a continuación de un nodo cualquiera de una lista:

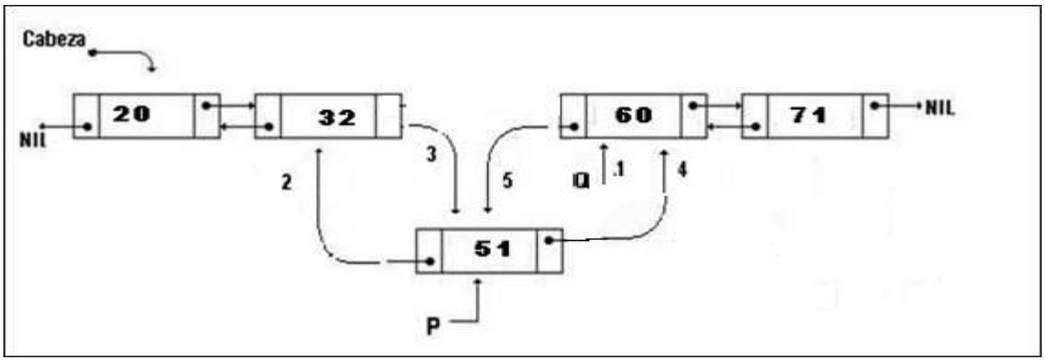
Se trata de *INSERTAR* un nuevo nodo en una lista no vacía, a continuación de uno nodo cualquiera que no sea el último de la lista:



El proceso es el siguiente:

1) **Ubicar apuntador (Q) en el nodo "Siguiente" a donde debe ir el nodo "nuevo" (P)**
2) **Hacemos que el Anterior del nodo Nuevo (P) sea el anterior al cual apunta (Q)**
3) **Hacemos que el Siguiente del Anterior al nodo nuevo (P) es precisamente (P)**
4) **Hacemos que el Siguiente del nodo nuevo (P) sea (Q)**
5) **Hacemos que el Anterior de (Q) sea (P).**

Gráficamente se ilustra la situación *antes* del proceso:



Después de haber considerado todas las posibilidades que se presentarían al momento de insertar el nodo ***ORDENADAMENTE*** en una Lista Enlazada Doble, pasamos a elaborar completamente nuestro algoritmo:

INSERTA_ORDENADA_DOBLE

```
/*Inserta Información de manera ordenada ascendentemente por código en una lista Enlazada Doble*/
REPITA
    NEW(P)
    SI (P= Ω)  Ent
        ESCRIBA("No hay memoria Disponible …..")
    SINO
        /* Capturo los datos  y lleno el nodo …..*/
        ESCRIBA("Referencia del articulo ?:"); LEA (P ^.Refe) ; ESCRIBA("Nombre  del articulo ? "); LEA (P ^.Articulo)
        P ^. ANT ← Ω ;  P ^. SIG  ← Ω ;
        /* La lista está Vacia????? */
        SI  Cabeza =  Ω  Ent
            Cabeza ← P
        SINO
            /*  Va antes del primero???  */
            SI  P ^. REFE < Cabeza ^.REFE  Ent    /* Lo inserto de primero   */
                P ^. SIG   ←  Cabeza; Cabeza ^. ANT  ←   P; Cabeza  ←   P
            SINO
                /*  Va Después del Ultimo???  */
                Q ←  Cabeza
                MQ  Q^.SIG  < > NIL Haga
                    Q ← Q ^. SIG
                FMQ
                SI  P ^. REFE > Q ^. REFE  Ent  /* Lo conecto después del ultimo ….*/
                    Q ^. SIG  ← P;  P ^. ANT ← Q
                SINO     /* Va en medio de dos nodos cualesquiera …..*/
                    Q ←  Cabeza
                    MQ  P ^.REFE  >= Q ^. REFE   Haga
                        Q ← Q ^. SIG
                    FMQ
                    /* Enlazo mi nuevo nodo, primero por la IZQUIERDA ….*/
                    P ^.ANTERIOR ← Q ^.ANTERIOR
                    P ^.ANTERIOR^.SIG ← P
                    /* Enlazo mi nuevo nodo, ahora por la DERECHA…… ….*/
                    P ^.SIG ← Q ;  Q ^.ANTERIOR ← P
                FSI
            FSI
        FSI
    FSI
    ESCRIBA(" DESEA  INSERTAR OTRO ALUMNO?  [S/N] …. "); LEA(MAS);
HASTA( MAS = 'N');
```

FIN INSERTAR ORDENADO

2.- Buscar (Consultar) Información en una lista doblemente enlazada

Esta labor inicia solicitando el dato a consultar (Para nuestro caso la **REFERENCIA** del artículo) y luego comparamos secuencialmente esta referencia con las que se encuentran en cada nodo de la lista, hasta encontrar coincidencia ó hasta terminar de recorrer la lista sin encontrarla.

Inicialmente habrá que considerar la posibilidad de que la lista esté vacía (**Cabeza = NIL**)

De acuerdo con esto, nuestro algoritmo podrá ser escrito como se muestra a continuación:

CONSULTAR ARTICULO

```
REPITA
    ESCRIBA(" Referencia de Artículo que desea Consultar? "); LEA (REF)
    SI Cabeza = Ω Ent
        Escriba(" La lista está Vacía, Debe registrar primero los artículos ….."")
        SINO
            P ← Cabeza; Encontro ← 'n'
            MQ ( P < > Ω ) AND ( Encontro = 'n' ) Haga
                SI P ^. Refe = REF Entonces
                    Encontro ← 'S'; Q ← P /* Guarda en Q la dirección del nodo si encontró */
                SINO
                    P ← P ^. SIG
                FSI
            FINMQ

            SI Encontro = ' n ' Entonces
                Escriba (" No se encuentra ese articulo en Inventario …")
            SINO
                Escriba(" **** Información del Artículo ****")
                Escriba("REFERENCIA ", P ^. Refe)
                Escriba("ARTICULO ", P ^. Articulo)
            FINSI
    FINSI
    Escriba(" Desea Consultar Otro Artículo? [S/N] "); Lea(Desea)
HASTA(Desea = 'N')
```

FIN CONSULTAR ARTICULO

3.- Eliminar un elemento de una lista doblemente enlazada

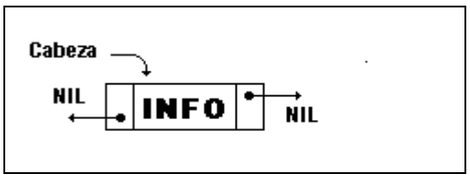
Recordemos que una Lista Enlazada Doble es una *"**Estructura de Datos Dinámica***", por lo tanto, el hecho de retirar un elemento de la estructura implica, a demás, "*Devolver*" al Sistema Operativo la memoria ocupada por el elemento que se ha borrado. En esto consiste el uso ***Dinámico*** de la memoria.

En el momento de BORRAR un nodo de la lista doble podemos encontrarnos con los siguientes casos:

- **Eliminar el único nodo de una lista doblemente enlazada.**
- **Eliminar el primer nodo.**
- **Eliminar el último nodo.**
- **Eliminar un nodo intermedio.**

Eliminar el único nodo en una lista doblemente enlazada:

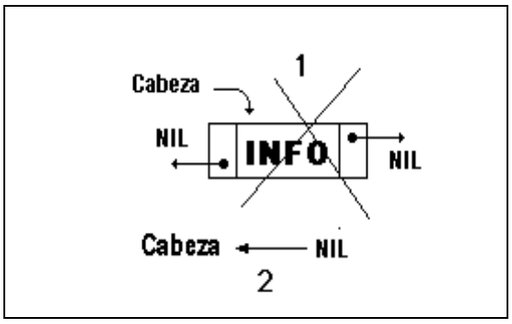
Si la lista tiene UN solo nodo entonces Cabeza apunta a este nodo.



Para Borrarlo hacemos lo Siguiente:

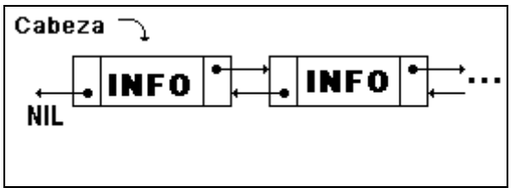
1. *Reasignamos* al Sistema operativo el bloque de memoria RAM que ocupa actualmente el nodo
2. Asignamos **NIL** al apuntador Cabeza

Gráficamente estas operaciones se verían así:



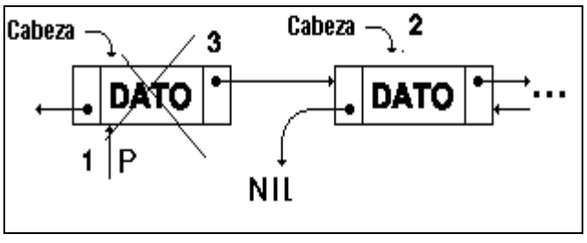
Eliminar el primer nodo de una lista doblemente enlazada:

Partimos del hecho de que la lista no está vacía y tiene más de un nodo.



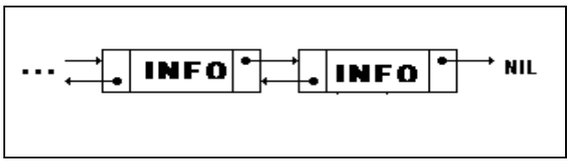
Para Borrarlo hacemos lo Siguiente:

1.- Guardamos la dirección del primer nodo en un apuntador.
2.- *Movemos* la cabeza al segundo nodo.
3.- Reasignamos al Sistema Operativo el bloque de memoria RAM que ocupaba el primer nodo.



Eliminar el último nodo de una lista doblemente enlazada:

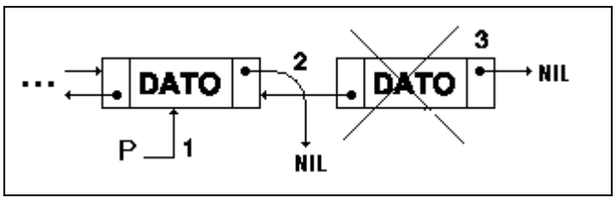
Partimos del hecho de que la lista no está vacía.



Para Borrarlo hacemos lo Siguiente:

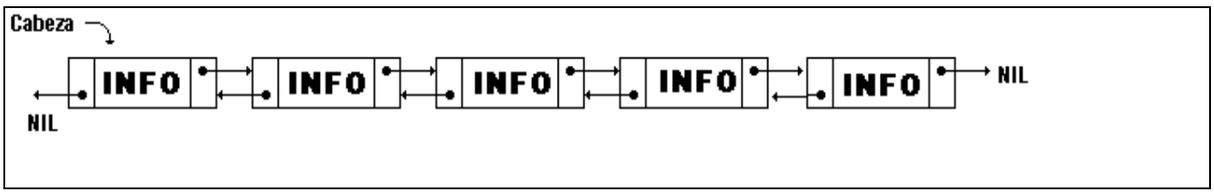
1.- Ubicamos un apuntador en el PENULTIMO Nodo.
2.- Ahora el Penúltimo nodo quedará siendo el ULTIMO, por lo tanto su siguiente es NIL
3.- Reasignamos al Sistema Operativo el bloque de memoria RAM que ocupaba el último nodo.

El siguiente gráfico ilustra la aplicación de estos pasos al proceso de borrado del nodo en la lista.



Eliminar un nodo intermedio de una lista doblemente enlazada:

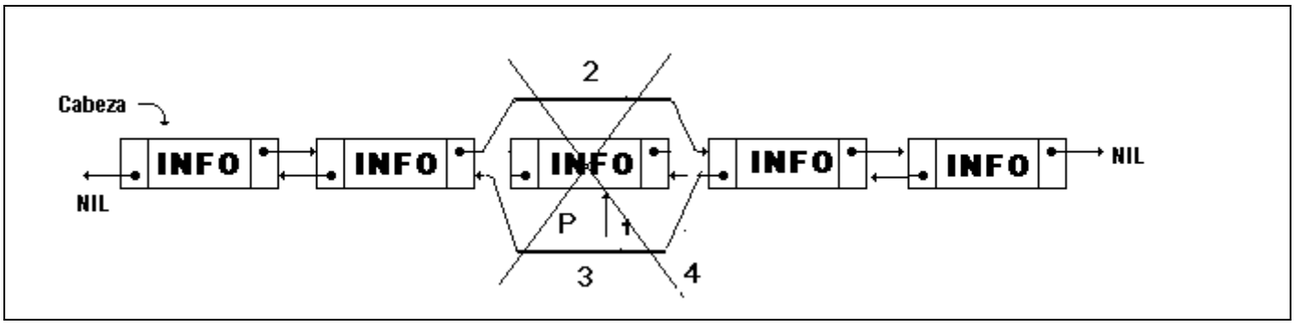
Partimos del hecho de que la lista no está vacía y que se ha descartado que se trate del PRIMERO ó del ULTIMO.



Para Borrarlo hacemos lo Siguiente:

1.- Ubicamos un apuntador en el Nodo que se desea borrar.
2.- Enlazamos el ANTERIOR a este con el SIGUIENTE del que se va a borrar
3.- Enlazamos el SIGUIENTE de con el ANTERIOR del que se va a borrar
4.- Reasignamos al Sistema Operativo el bloque de memoria RAM que ocupaba el nodo borrado.

Lo anterior se ilustra en la siguiente gráfica …….



Ahora sí, vamos a escribir el algoritmo **(*Vaya a la siguiente página*):**

```
BORRAR_ ARTICULO
    Seguir ← 'S'
    MQ (Seguir = 'S' ) Haga
        ESCRIBA(" Referencia de Artículo que desea Borrar? "); LEA (REF)
        SI  Cabeza =  Ω  Ent
            Escriba(" La lista está Vacía, Debe registrar primero los artículos ….."")
            SINO
                P ← Cabeza; Encontro ← 'n'
                MQ  ( P < > Ω )  AND ( Encontro = 'n' ) Haga
                    SI   P ^. Refe = REF Entonces
                        Encontro ← 'S';  Q ← P   /* Guarda en Q la dirección del nodo si encontró */
                        SINO
                            P ← P ^. SIG
                    FSI
                FINMQ
                SI Encontro = ' n ' Entonces
                        Escriba (" No se encuentra ese articulo en Inventario …")
                    SINO
                        Escriba("  **** Información Actual del Artículo ****")
                        Escriba("REFERENCIA ", P ^. Refe)
                        Escriba("ARTICULO     ", P ^. Articulo)
                        Escriba("  Está seguro de Borrar?? [S/N] ); Lea(Confirma)
                        SI   Confirma = 'S' Ent      /* está seguro de borrar el artículo */
                            SI  Q = Cabeza Ent  /* Se trata del primer Nodo ….*/
                                Cabeza  ← Cabeza ^. SIG; Delete(Q)
                                SINO
                                    SI Q ^. SIG = Ω Ent  /* Se trata del ULTIMO nodo …*/
                                            Q ^. ANT  ^.SIG  ←  Ω;  Delete(Q)
                                        SINO   /* Entonces es un nodo intermedio cualquiera*/
                                            Q ^. ANT ^.SIG   ← Q ^.SIG;
                                            Q ^. SIG ^. ANT  ← Q ^.ANT

                                    FSI
                                FSI
                        FSI
                FSI
        FSI

        Escriba(" Desea Borrar OTRO Artículo ?  [S/N]  " ) ;  Lea(Seguir)
    FMQ

FIN BORRAR_ARTICULO
```