

PROGRAMACIÓN I

AÑO 2025

Temas que veremos en esta clase

- ✓ Listas con doble enlace
- ✓ Listas doblemente enlazadas

Listas con Doble Enlace

Son listas enlazadas que tienen dos enlaces en lugar de uno.

Son listas cuyos NODOS se encuentran enlazados a otros 2 NODOS por medio de criterios diferentes.

Podemos recorrerla en 2 sentidos (órdenes) diferentes tenemos 2 “vistas” diferentes de los mismos datos

VENTAJAS

- ✓ Pueden accederse siguiendo cualquiera de los dos órdenes, sin utilizar espacio extra...

INCONVENIENTE

- ✓ Ocupan más memoria por nodo que una lista simple.

Listas con Doble Enlace

Una representación posible para el nodo, considerando que los enlaces permitirán recorrer la lista por los dos criterios, podría ser la siguiente:

type

listaD = ^nodoD;

nodoD = **record**

datos: ...;

sig1: listaD;

sig2: listaD;

end;



Pensemos cómo sería la representación de la Lista con Doble Enlace...

Listas con Doble Enlace

Una representación adecuada si se piensa que la lista doble tiene dos comienzos (dos ordenes diferentes) ... podría ser la siguiente:

type

listaD = ^nodoD;

nodoD = **record**

datos: ...;

sig1: listaD;

sig2: listaD;

end;

lista_doble= **record**

orden1: listaD;

orden2: listaD;

end;

var

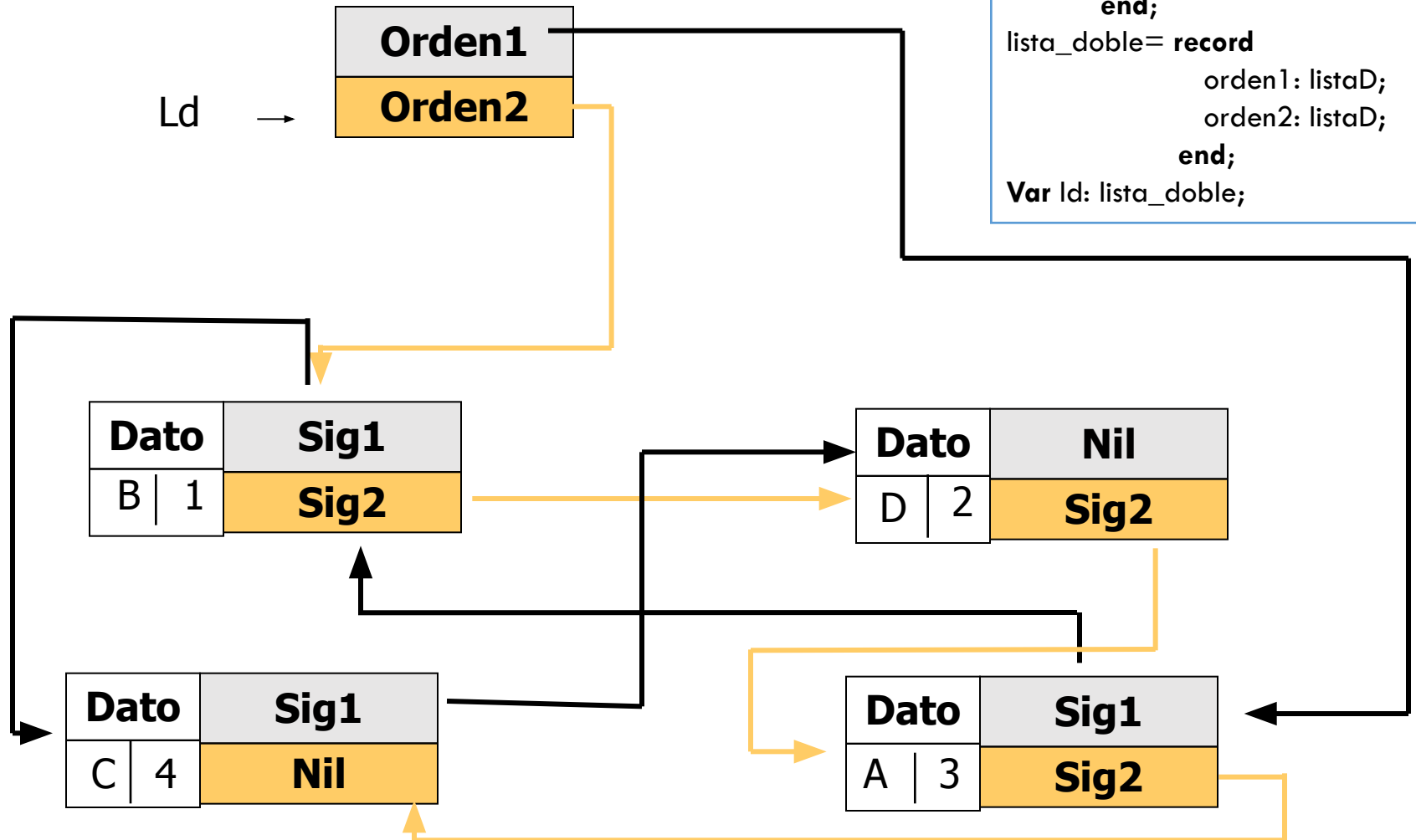
ld : lista_doble;



Pensemos ahora en las
operaciones de Creación y
Recorridos en ambos
sentidos...

Listas con Doble Enlace

Lógicamente se puede ver:



INSERTAR EN UNA LISTA DOBLE ENLACE



Listas con Doble Enlace

La facultad mantiene la información de sus alumnos. Para ello cuenta con una lista ordenada por dos criterios diferentes:

- por apellido
- por número de alumno

Se pide realizar un módulo que reciba un nuevo alumno y lo inserte, manteniendo los órdenes con los que están almacenados los alumnos.

Listas con Doble Enlace de alumnos

type

cadena30 = string[30];

alumno = **record**

 apellido: cadena30;

 numero: integer;

 dni: integer;

end;

listaD = ^nodoD;

nodoD = **record**

 datos: alumno;

 sig_ape: listaD;

 sig_num: listaD;

end;

lista_doble = **record**

 pri_ape: listaD;


 pri_num: listaD;


end;

Listas con Doble Enlace de alumnos

¿Como conviene trabajar?

- Se crea un único nodo.
- Se inserta en orden (siguiendo el orden de apellido).
- Se inserta en orden (siguiendo el orden numérico).

 ¿Qué pasa si la lista está vacía?

 ¿Sirve llamar a los módulos "insertar en orden" para cada orden?

Listas con Doble Enlace de alumnos

type

```
cadena30 = string[30];
alumno = record
    apellido: cadena30;
    numero: integer;
    dni: integer;
end;
listaD = ^nodoD;
nodoD = record
    dato: alumno;
    sig_ape: listaD;
    sig_num: listaD;
end;
lista_doble = record
    pri_ape: listaD;
    pri_num: listaD;
end;
```

```
procedure insertar (var ld: lista_doble; a:alumno);
var nue: listaD;
begin
    new (nue);  nue^.dato:=a;
    nue ^.sig_ape := nil; nue^.sig_num := nil;
    if (ld.pri_ape = nil ) then begin
        {La lista esta vacía?}
        ld.pri_ape := nue;
        ld.pri_num:= nue;
    end
    {Como no esta vacía agrego por los 2 ordenes}
    else begin
        insertar_X_apellido(ld, nue);
        insertar_X_numero(ld, nue);
    end;
end;
```

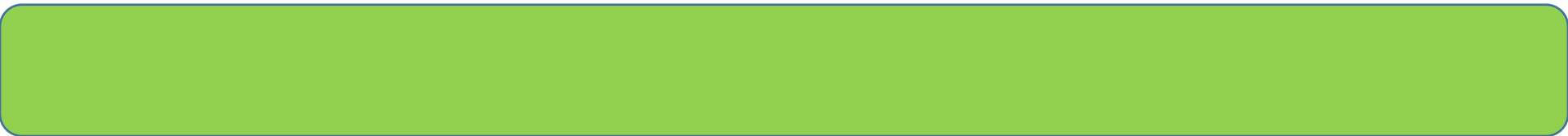
Listas con Doble Enlace

```
procedure insertar_X_apellido (var ld: lista_doble; nue: listaD);  
    var ant, act: listaD; encontro: Boolean;  
  
begin  
    act := ld.pri_ape;  
    encontro := false;  
    while (act <> nil) and (not encontro) do  
        begin  
            if (nue^.dato.apellido < act^.dato.apellido) then encontro := true  
            else begin  
                ant := act;  
                act := act^.sig_ape;  
            end;  
        end;  
    if (act = ld.pri_ape) then ld.pri_ape := nue {Inserto primero}  
    else ant^.sig_ape := nue; {Es al medio o al final}  
    nue^.sig_ape := act;  
end;
```

Listas con Doble Enlace

```
procedure insertar_X_numero (var ld: lista_doble; nue: listaD);  
    var ant, act: listaD; encontro: Boolean;  
  
begin  
    act:= ld.pri_num; encontro:=false;  
    while (act <> nil) and (not encontro) do  
        begin  
            if (nue^.dato. numero < act^.dato. numero)  
                then encontro:=true  
                else begin  
                    ant:= act;  
                    act:= act^.sig_num;  
                end;  
            end;  
            if (act = ld.pri_num) then ld.pri_num := nue {Inserto primero}  
                else ant^.sig_num := nue; {Es al medio o al final}  
            nue^.sig_num := act;  
        end;
```

LISTAS DOBLES EN ORDEN ASCENDENTE Y DESCENDENTE



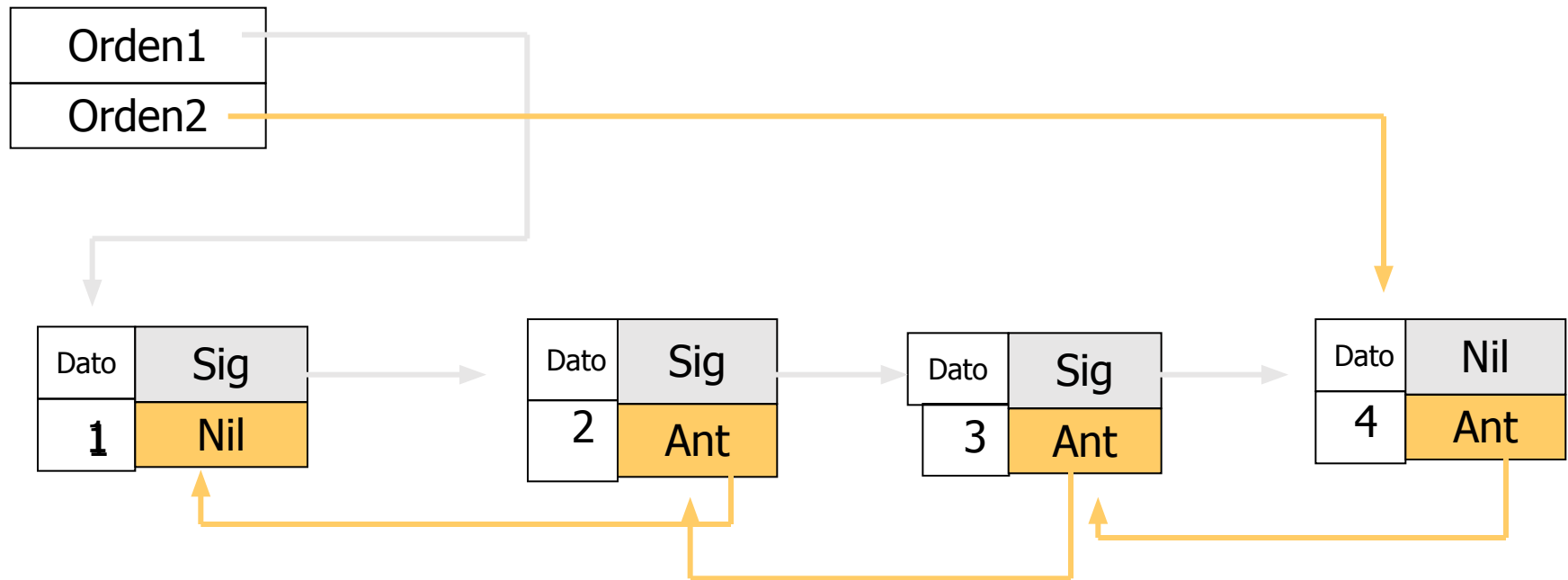
Listas con Doble Enlace CON ANTERIOR Y SIGUIENTE

Caso especial: Cada nodo tiene un “siguiente” y un “anterior” los cuales consideran el mismo criterio.

Por ejemplo: Una lista de números donde hay dos órdenes según el mismo criterio:

Orden 1: Orden Ascendente

Orden 2: Orden Descendente



Listas con Doble Enlace CON ANTERIOR Y SIGUIENTE

Una representación posible considerando que los enlaces permitirán recorrer la lista en dos sentidos de manera tal que un sentido da la inversa del otro.

type

PtrNodo = ^nodo;

nodo = **record**

datos: ...;

ant: PtrNodo;

sig: PtrNodo;

end;

listaDoble = **record**


primero: PtrNodo;

ultimo: PtrNodo;

end;

var

l : listaDoble;



**¿Operación
Crear,
Insertar, etc.?**

Listas Doblemente Enlazadas - Creación

type

PtrNodo = ^nodo;

nodo = **record**

datos: ...;

ant: PtrNodo;

sig: PtrNodo;

end;

listaDoble = **record**

primero: PtrNodo;

ultimo: PtrNodo;

end;

procedure Creacion (**var** l: listaDoble);

begin

l.primero := nil;

l.ultimo := nil;

end;

Listas Doblemente Enlazadas - Creación

CASOS A CONSIDERAR

- El nodo a insertar es el primero
- El nodo a insertar se ubica último
- El nodo a insertar es uno cualquiera

procedure InsertaOrdenado (lista de entrada y el dato);

 pedir espacio

 guardar el dato

 enganchar el nuevo nodo a la lista

si la lista está vacía **entonces**

 este nodo es el primero

sino

 recorre la lista hasta encontrar la posición

Si corresponde insertar en la primera posición

entonces enlazar y actualizar el puntero inicial

sino si es uno cualquiera

entonces enlazar hacia delante y hacia atrás

si es el último **entonces**

 enlazar y actualizar puntero al último

procedure InsertaOrdenado (var l: listaDoble; num: integer);

var act, ant, nuevo : Ptrnodo; encontro:Boolean;

begin

new (nuevo); nuevo^.datos:= num; nuevo^.sig := nil; nuevo^.ant :=nil;

if (l.primerio = nil)

then begin

l.primerio:= nuevo; l.ultimo:= nuevo;

end

else begin

act:= l.primerio;

while (act <> nil) and (act^.datos < num) **do begin**

ant:= act; act:= act^.sig;

end;

if (act = l.primerio)

then begin

nuevo^.sig:=l.primerio; l.primerio^.ant:=nuevo; l.primerio:= nuevo;

end

else if (act <> nil)

then begin

nuevo^.ant:= ante; ante^.sig:= nuevo;

nuevo^.sig:= act; act^.ant:= nuevo;

end

else begin

nuevo^.ant:=l.ultimo; l.ultimo^.sig:=nuevo;

l.ultimo:= nuevo;

end;

end;

end;

type

Ptrnodo = ^nodo;

nodo = **record**

datos:

ant: Ptrnodo;

sig: Ptrnodo;

end;

listaDoble = **record**

primerio: Ptrnodo;

ultimo: Ptrnodo;

end;

Buscar posición

Va al inicio

Va al medio: entre dos nodos
existentes

Va al final – como último nodo

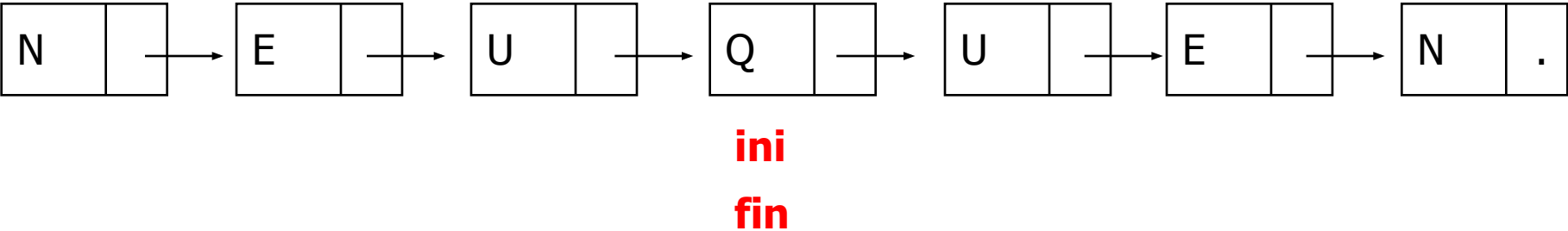
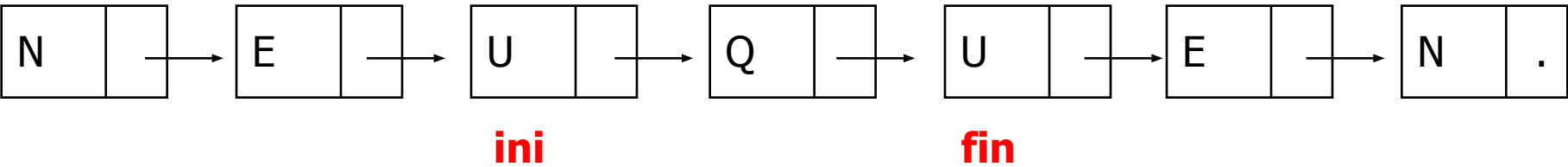
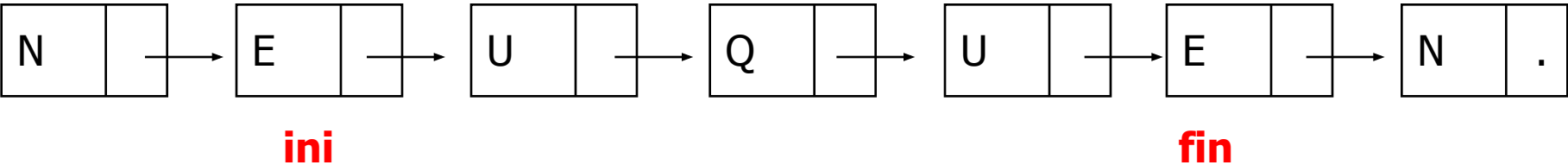
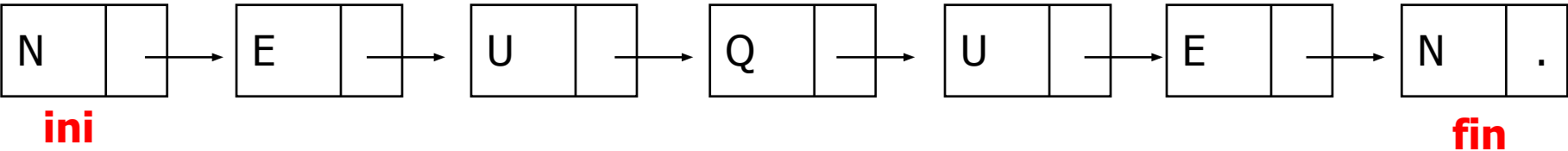
Ver en pascal

LISTAS DOBLES EN ORDEN ASCENDENTE Y DESCENDENTE **EJERCICIO**



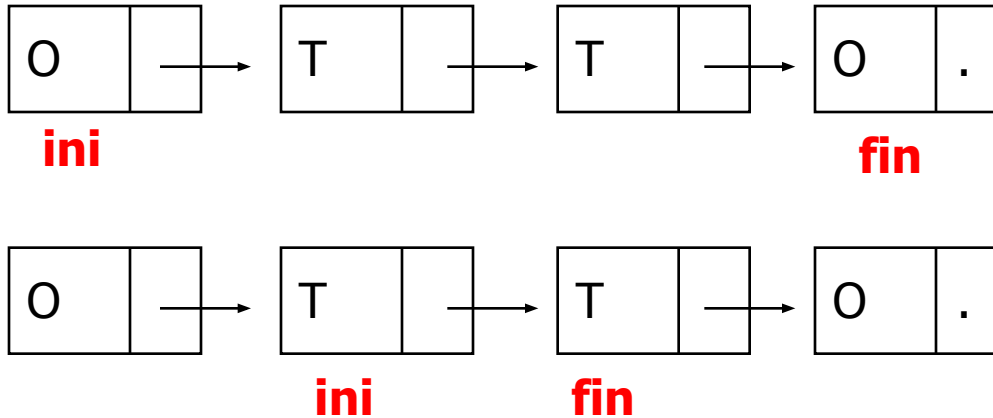
Listas Doblemente Enlazadas- Ejemplo

Supongamos que tenemos almacenados los caracteres de una palabra en una lista, y queremos saber si la palabra es capicúa. ¿Cómo lo resolvemos? ¿Qué casos hay?



Listas Doblemente Enlazadas- Ejemplo

Supongamos que tenemos almacenados los caracteres de una palabra en una lista, y queremos saber si la palabra es capicúa. ¿Cómo lo resolvemos? ¿Qué casos hay?



Listas Doblemente Enlazadas- Ejemplo

```
Function esCapicua (Id:listaDoble):boolean;  
  var ini, fin: PtrNodo;  
      ok, termine: boolean;  
  begin  
    ok:= true;  
    termine:=false;  
    ini:= Id.primeros;  
    fin:= Id.ultimo;  
    while (ok) and (ini <> fin) and (not termine) do  
      if (ini^.datos <> fin^.datos)  
      then ok:= false  
      else if (ini^.sig = fin)  
        then termine:=true  
        else begin  
          ini:= ini^.sig;  
          fin:= fin^.ant;  
        end;  
    esCapicua:=ok;  
  end;
```

```
type  
  PtrNodo = ^nodo;  
  nodo = record  
    datos: .....;  
    ant: PtrNodo;  
    sig: PtrNodo;  
  end;  
  listaDoble = record  
    primero: PtrNodo;  
    ultimo: PtrNodo;  
  end;
```

Ver en pascal esCapicua
pero en lista de enteros

Ejercicio



Un supermercado necesita almacenar sus productos. Cada producto está caracterizado por código de producto, tipo, marca y precio. La estructura donde se almacenarán los productos deberá estar ordenada por código de producto y también por marca. Implementar un programa con:

- a) Un módulo que lea y almacene la información.
- b) Un módulo que reciba la estructura con la información de los productos e informe la cantidad de productos existentes para cada marca.
- c) Un módulo que reciba la estructura con la información de los productos y una código de producto y retorne si existe o no un producto con código recibido.
- d) Un módulo que reciba la estructura con la información de los productos y un precio y retorne si existe o no un producto con el precio recibido.