Temas de Intro II

1.- Archivos

2.- Enumerado, Subrango y Registro

3.- Punteros y Listas

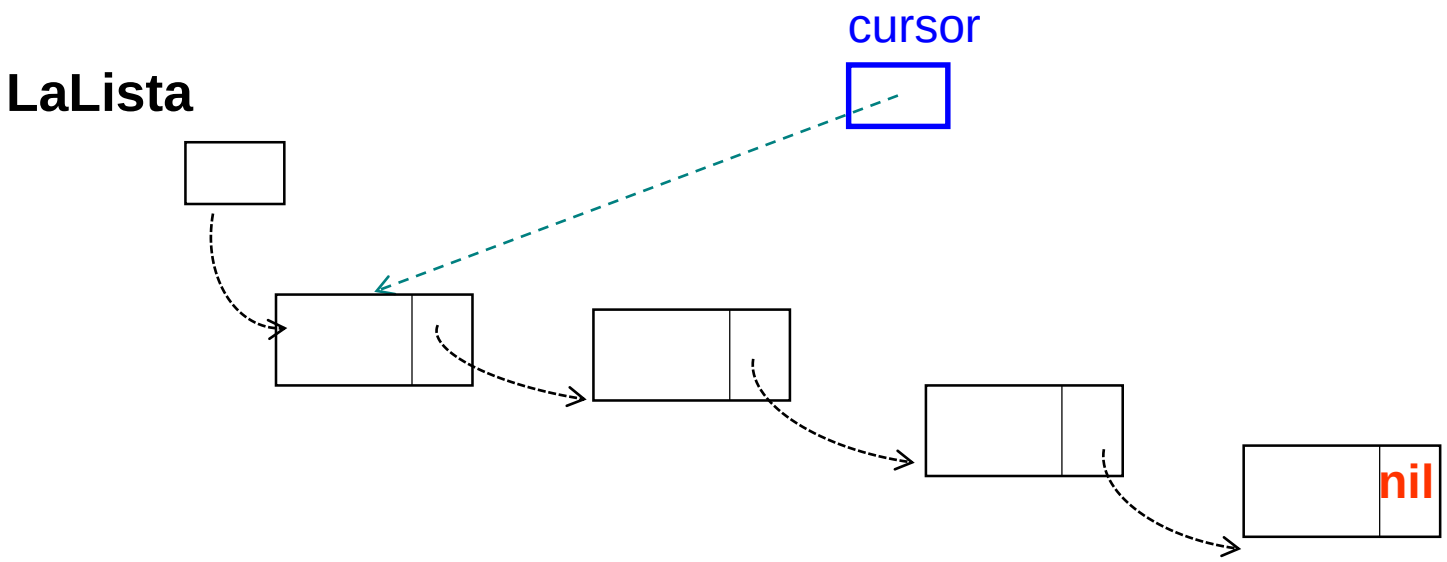
4.- Tipos de Listas

5.- Recursión

6.- Árboles

Repaso de lista simplemente vinculada

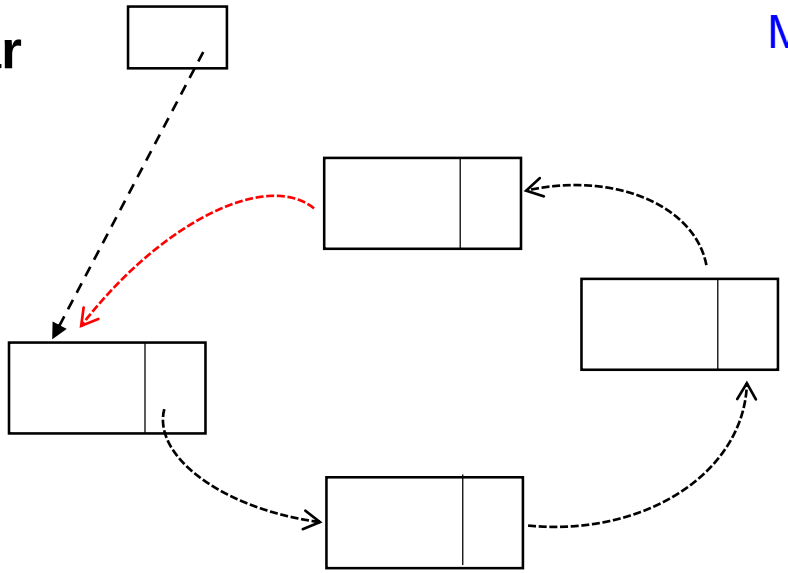
```
type
   PuntNodo = ^NodoLista;
   NodoLista = Record
            Nro :Integer;
            sig :PuntNodo
             end;
var
    Lalista, cursor: PuntNodo;
     ……
```



Lista Circular

```
type
   PuntNodo = ^NodoLista;
   NodoLista = Record
            Nro :Integer;
            sig :PuntNodo
             end;
var
    Lalista, cursor: PuntNodo;
     ……
```

LaListaCircular

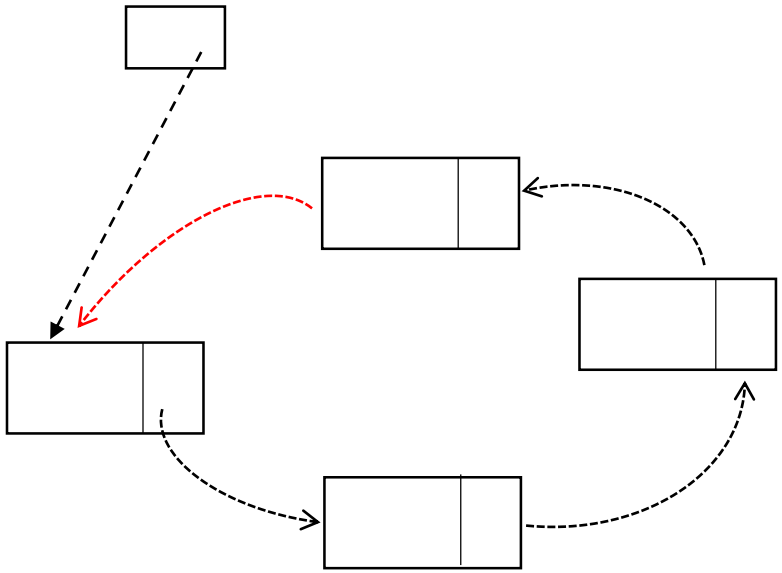


Misma descripción de tipo

Lista Circular

- El puntero inicial puede modificarse

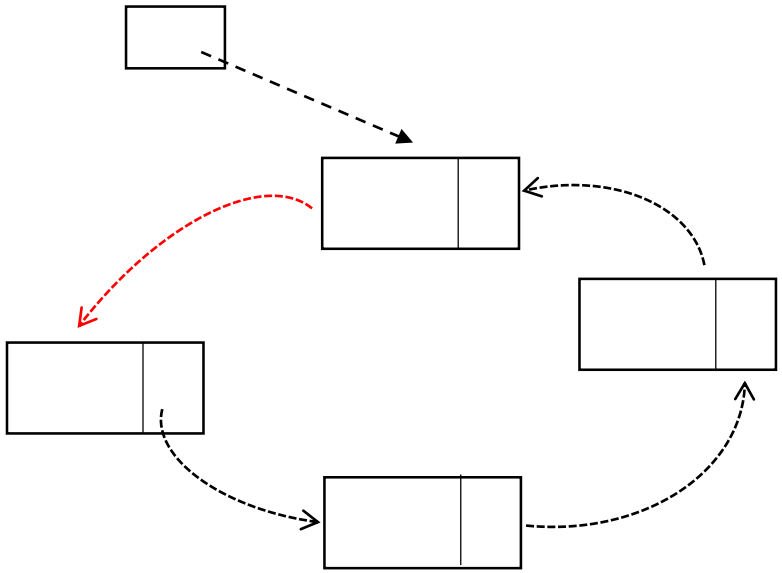
LaListaCircular



Lista Circular

- El puntero inicial puede modificarse
- No hay orden en la lista

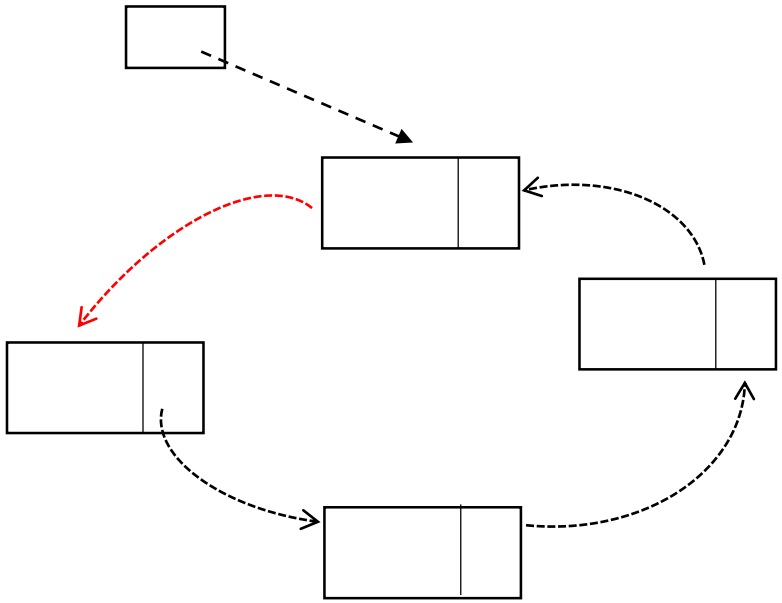
LaListaCircular



Lista Circular

- Hacer un procedimiento que imprima los elementos de una lista circular de enteros

LaListaCircular



Lista Circular

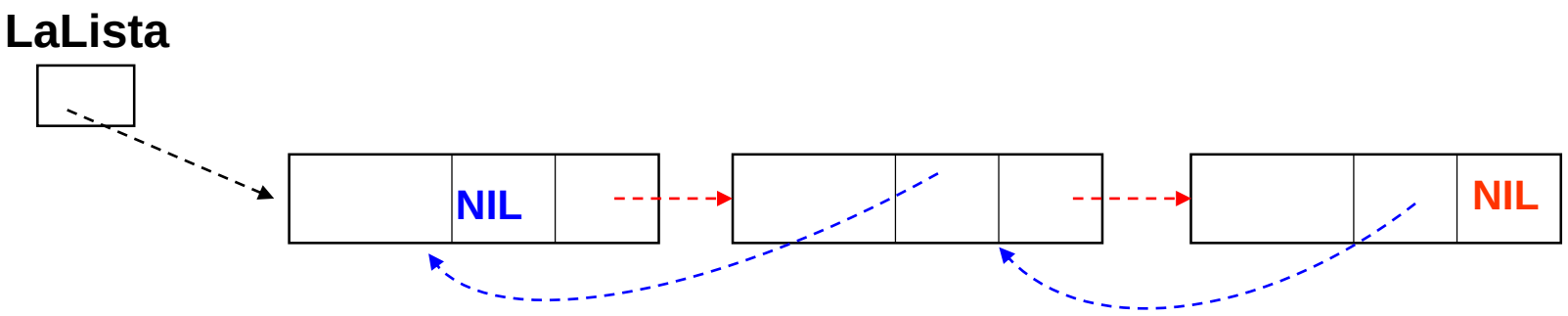
```
Procedure IMPRIMIR ( LaLista: Puntnodo);
var
   cursor: PuntNodo;

begin
  cursor:= Lalista;
  if cursor <> nil then
   begin
    writeln (cursor^.Nro);
    cursor:= cursor^.sig;
     while cursor<> LaLista do
       begin
         writeln (cursor^.Nro);
         cursor:= cursor^.sig;
       end;
    end;
 end.
```

Lista Circular – otra versión

```
Procedure IMPRIMIR ( LaLista: Puntnodo);
var
   cursor: PuntNodo;

begin
  cursor:= Lalista;
  if cursor <> nil then
   begin
    while cursor^.sig<> LaLista do
       begin
         writeln (cursor^.Nro);
         cursor:= cursor^.sig;
       end;
     writeln (cursor^.Nro);
end;
 end.
```

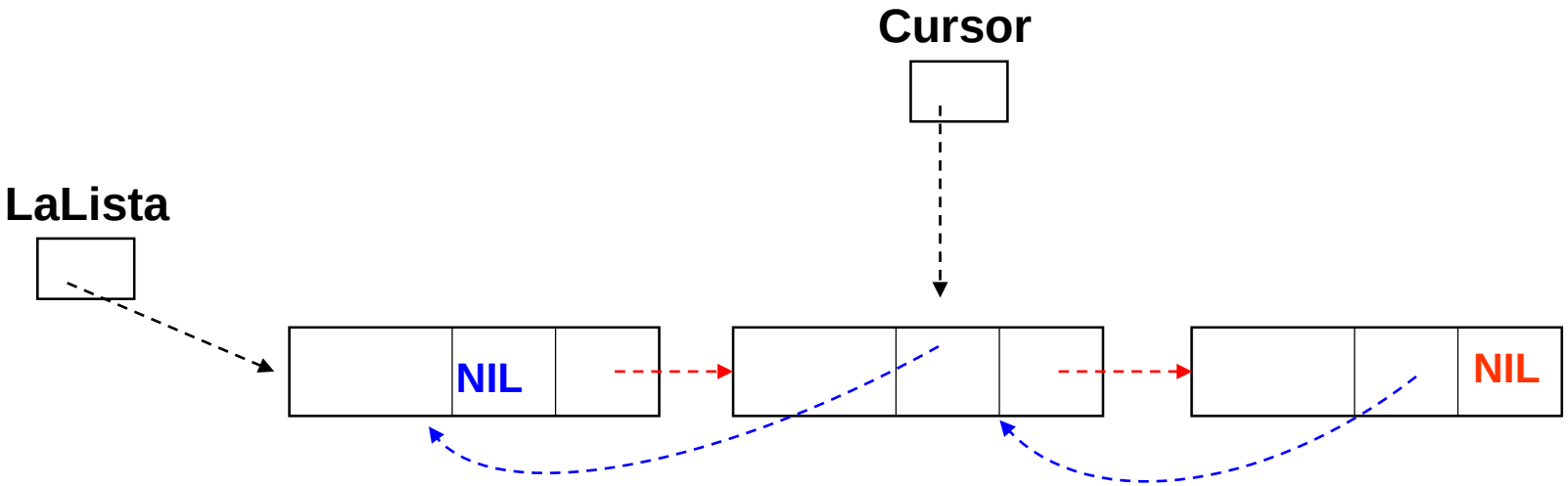
Lista DoblementeVinculada

```
type
   PuntNodo = ^NodoLista;
   NodoLista = Record
            Nro :Integer;
            ant :PuntNodo;
            sig :PuntNodo
end;
var
    Lalista: PuntNodo;
      ……
```



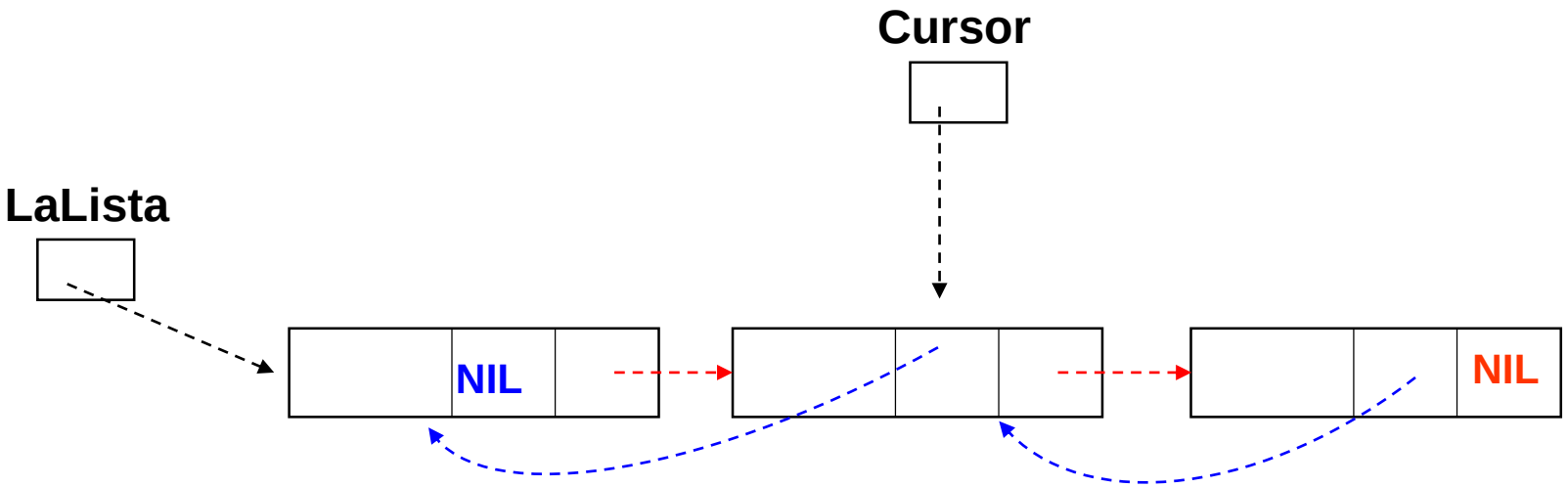
Lista DoblementeVinculada

```
type
   PuntNodo = ^NodoLista;
   NodoLista = Record
             Nro :Integer;
               ant :PuntNodo;
               sig :PuntNodo
   end;
var
    Lalista: PuntNodo;
      ……
```



Borrado en Lista DoblementeVinculada

```
type
   PuntNodo = ^NodoLista;
   NodoLista = Record
            Nro :Integer;
            ant :PuntNodo;
            sig :PuntNodo
   end;
var
    Lalista: PuntNodo;
      ……
```

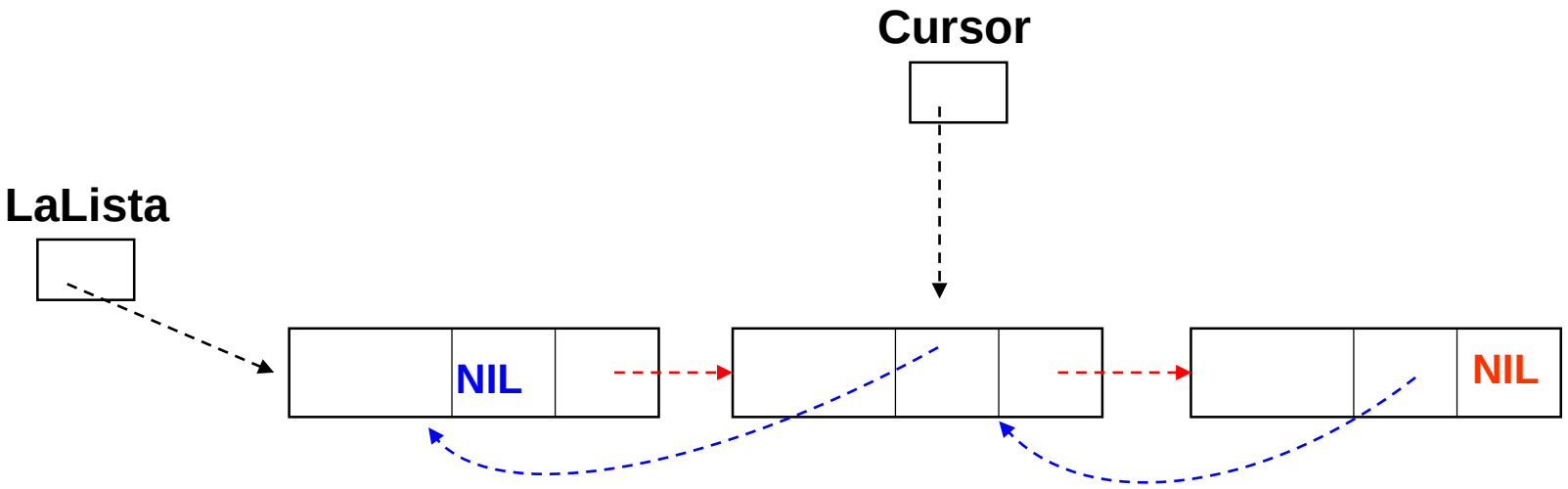


Borrado en Lista DoblementeVinculada

```
type
   PuntNodo = ^NodoLista;
   NodoLista = Record
            Nro :Integer;
            ant :PuntNodo;
            sig :PuntNodo
   end;
var
    Lalista: PuntNodo;
      ……
```

……

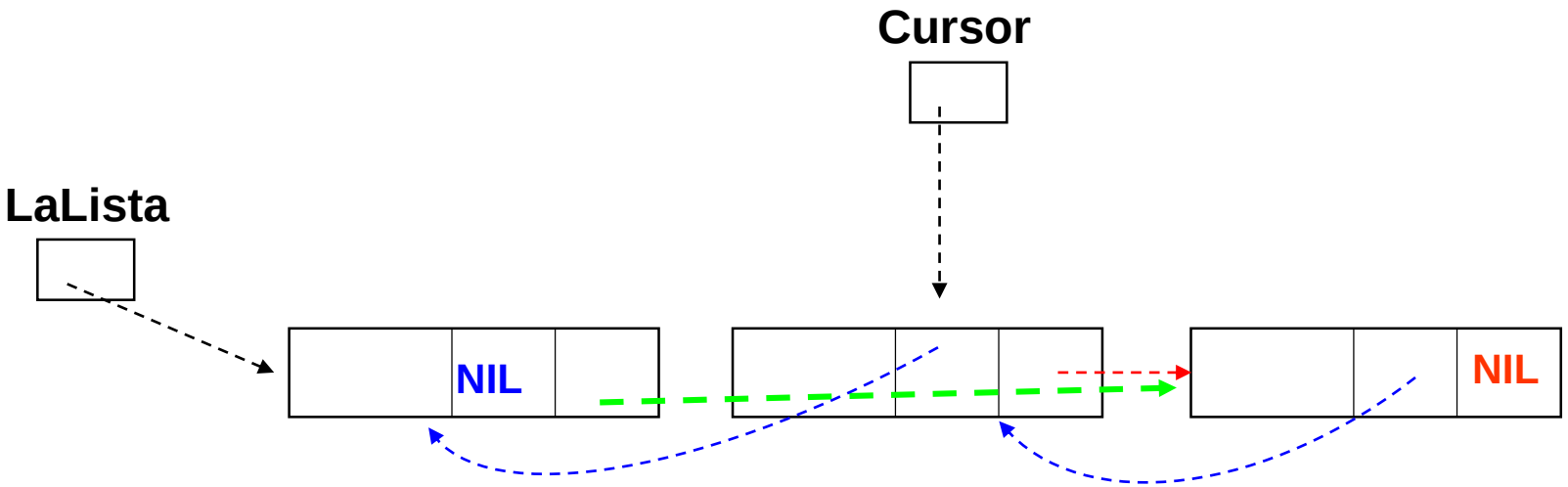
Cursor^.ant^.sig:= cursor^. Sig;



Borrado en Lista DoblementeVinculada

```
type
   PuntNodo = ^NodoLista;
   NodoLista = Record
            Nro :Integer;
            ant :PuntNodo;
            sig :PuntNodo
   end;
var
    Lalista: PuntNodo;
      ……
```

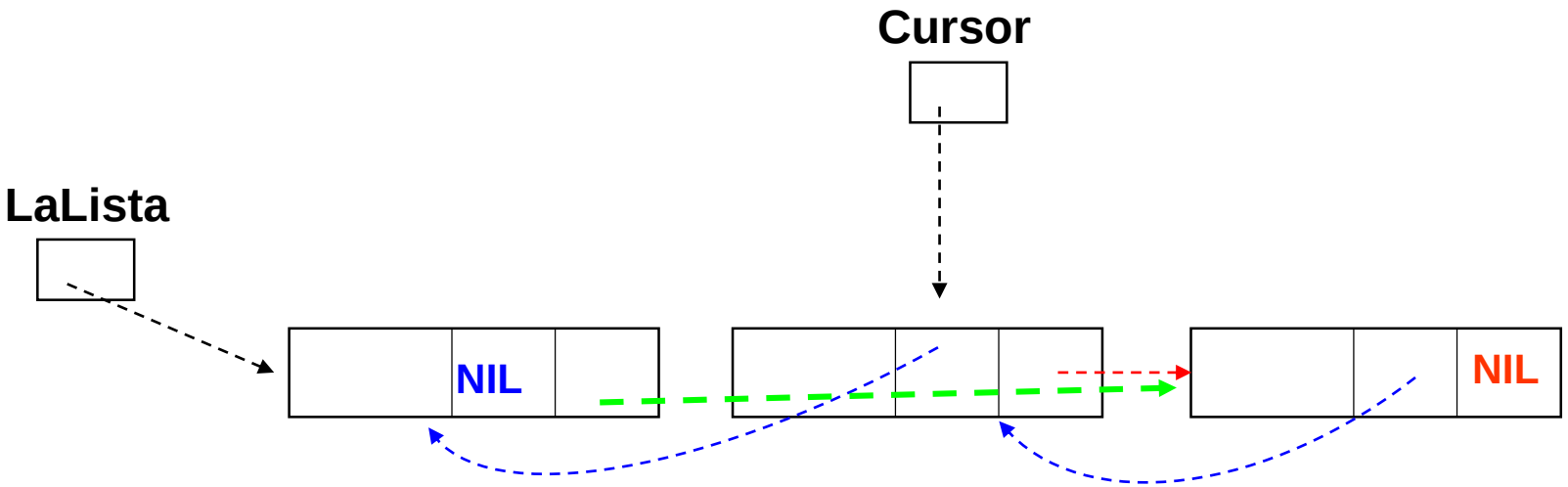
```
……
Cursor^.ant^.sig:= cursor^. Sig;
```



Borrado en Lista DoblementeVinculada

```
type
   PuntNodo = ^NodoLista;
   NodoLista = Record
            Nro :Integer;
             ant :PuntNodo;
             sig :PuntNodo

   end;
var
    Lalista: PuntNodo;
       ……..
```

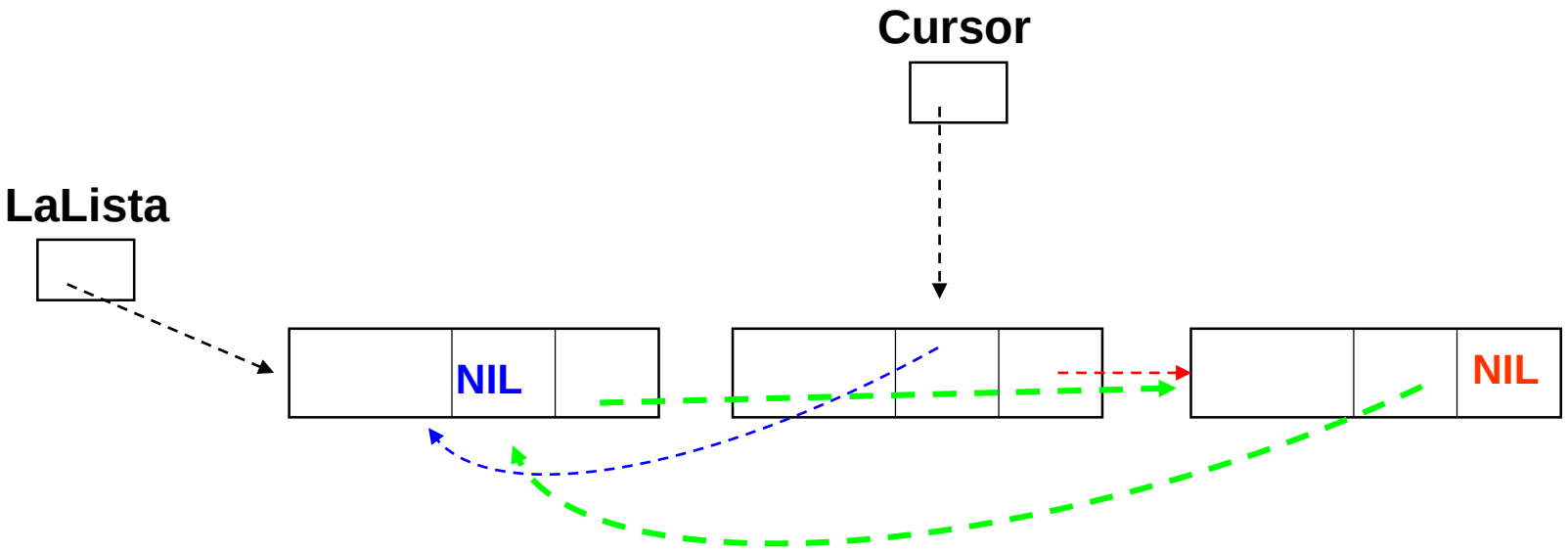
```
……
Cursor^.ant^.sig:= cursor^. Sig;
Cursor^.sig^.ant:= cursor^. ant;
```



Borrado en Lista DoblementeVinculada

```
type
   PuntNodo = ^NodoLista;
   NodoLista = Record
            Nro :Integer;
             ant :PuntNodo;
             sig :PuntNodo

   end;
var
    Lalista: PuntNodo;
       ……..
```

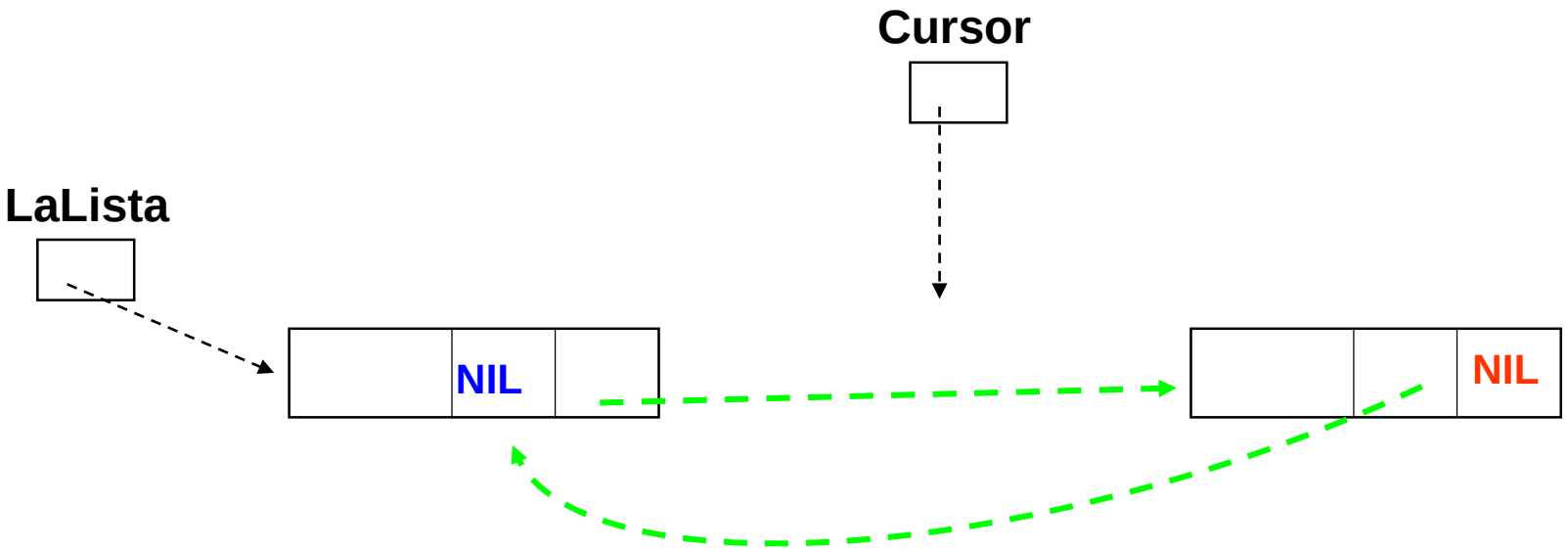
```
……
Cursor^.ant^.sig:= cursor^. Sig;
Cursor^.sig^.ant:= cursor^. ant;
```



Borrado en Lista DoblementeVinculada

```
type
   PuntNodo = ^NodoLista;
   NodoLista = Record
            Nro :Integer;
             ant :PuntNodo;
             sig :PuntNodo

   end;
var
    Lalista: PuntNodo;
       ……..
```

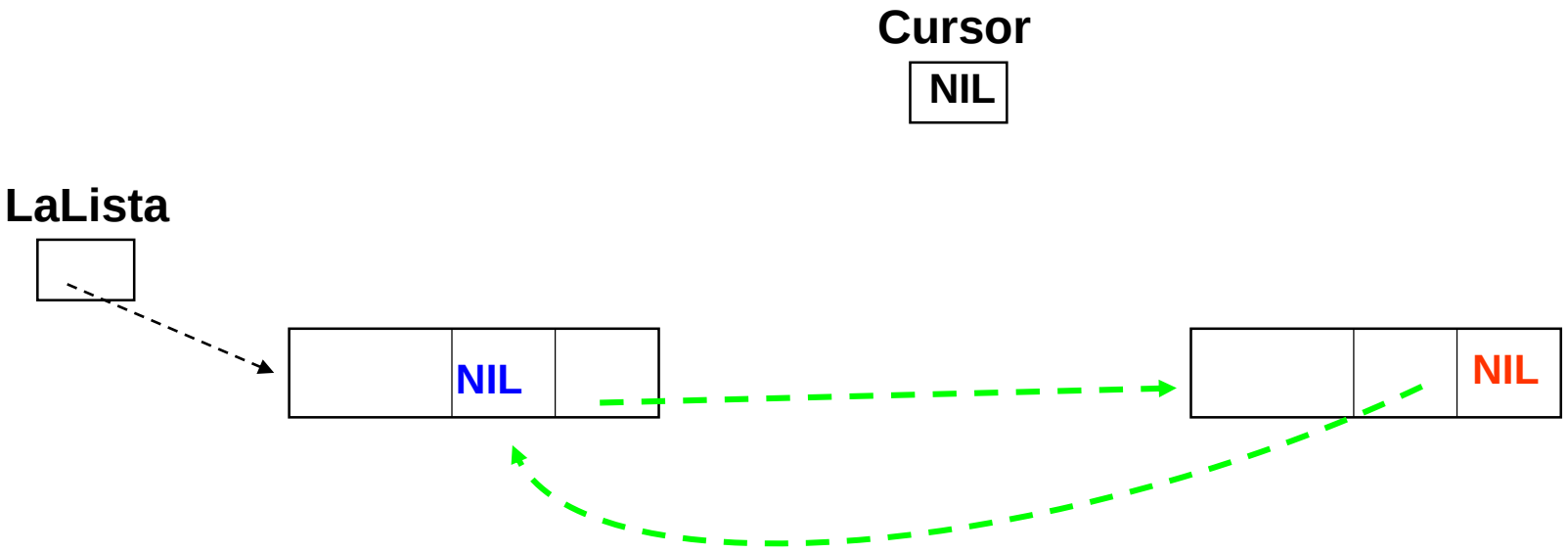
```
……
Cursor^.ant^.sig:= cursor^. Sig;
Cursor^.sig^.ant:= cursor^. ant;
Dispose(cursor);
```



Borrado en Lista DoblementeVinculada

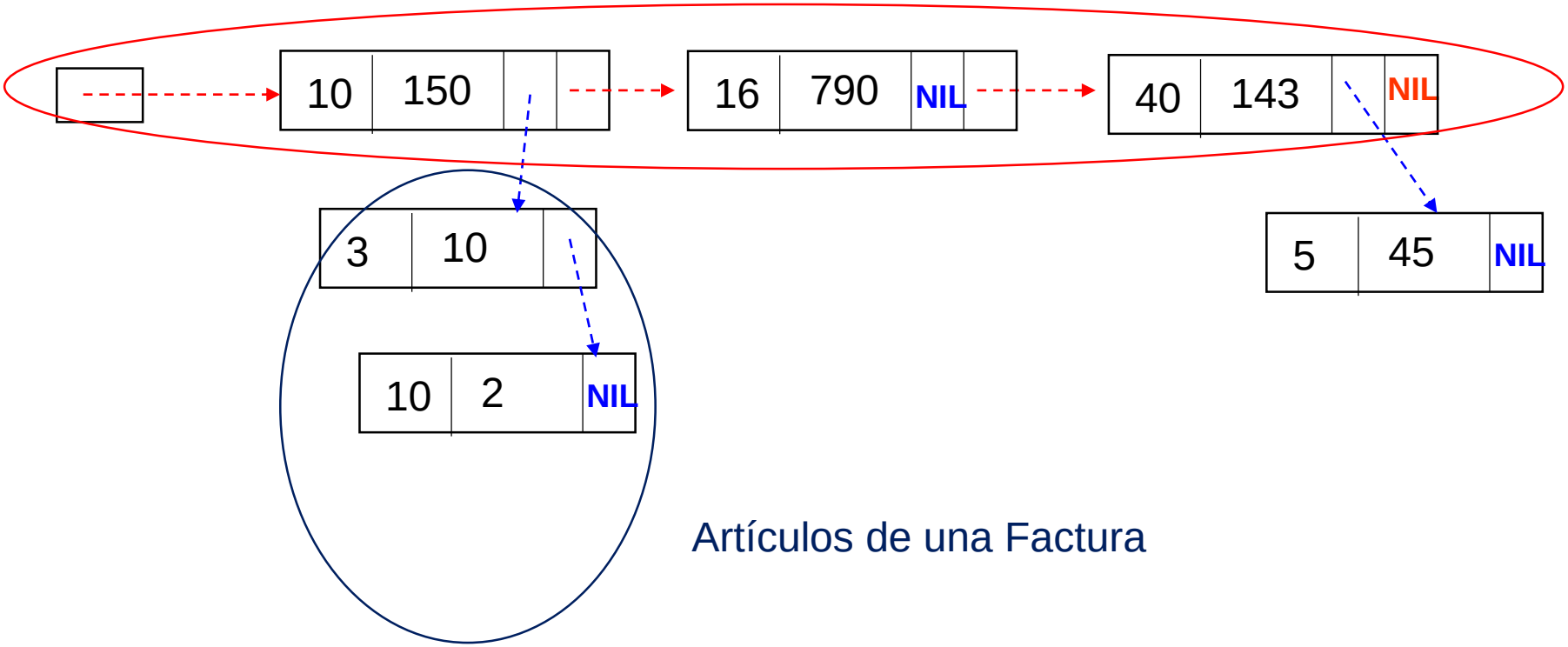
```
type
   PuntNodo = ^NodoLista;
   NodoLista = Record
            Nro :Integer;
             ant :PuntNodo;
             sig :PuntNodo

   end;
var
    Lalista: PuntNodo;
       ……..
```

```
……
Cursor^.ant^.sig:= cursor^. Sig;
Cursor^.sig^.ant:= cursor^. ant;
Dispose(cursor);
Cursor:= nil;
```



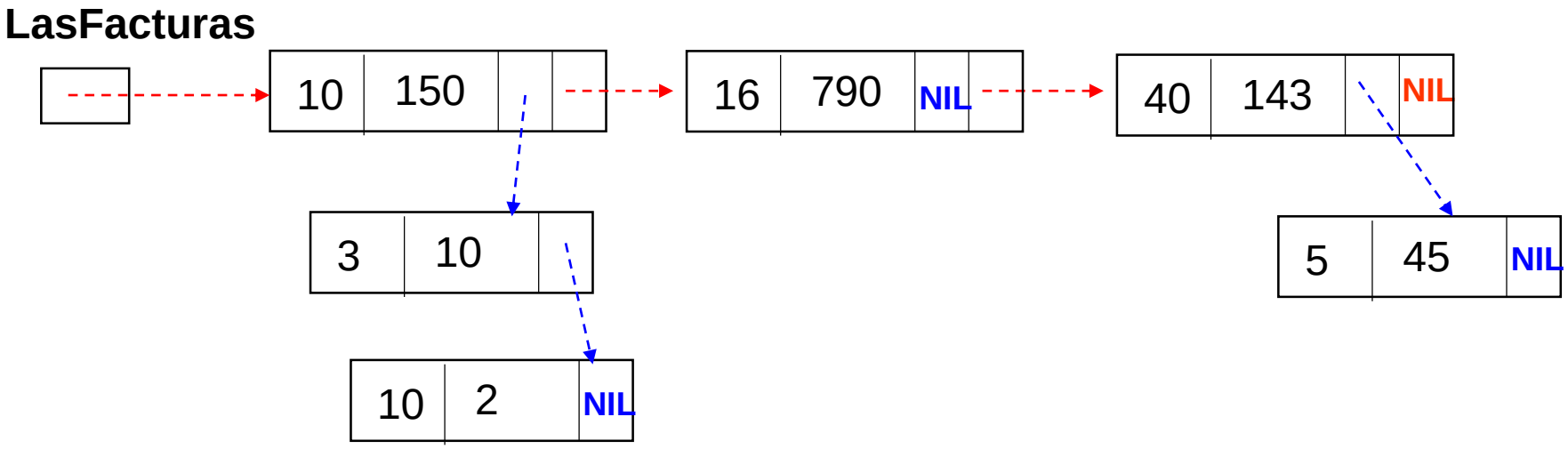
Lista simple en la que cada nodos tiene un puntero a otra lista simple

LasFacturas



Lista simple en la que cada nodos tiene un puntero a otra lista simple

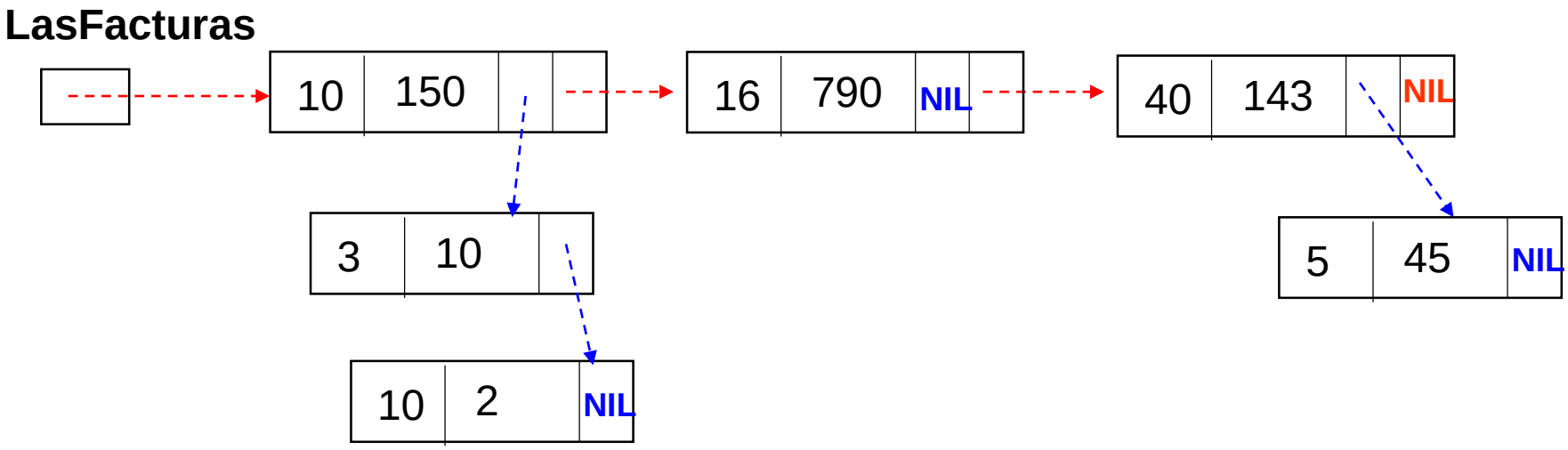
```
type
 PuntFactura = ^NodoFactura;

  NodoFactura = Record
            NroFact :Integer;
            Importe: Real,

            sigFactura: PuntFactura
   end;
```



Lista simple en la que cada nodos tiene un puntero a otra lista simple

```
type
 PuntFactura = ^NodoFactura;
 PuntArtículo = ^NodoArticulo;
  NodoFactura = Record
            NroFact :Integer;
            Importe: Real,
            Articulos: PuntArtículo;
           sigFactura: PuntFactura
   end;
```

```
NodoArticulo = Record
            NroArt:Integer;
            cant: Integer;
            sigArtículo: PuntArticulo
end;

var
    LasFacturas: PuntFactura;
     ……
```



Lista simple en la que cada nodos tiene un puntero a otra lista simple

```
type
 PuntFactura = ^NodoFactura;
 PuntArtículo = ^NodoArticulo;
  NodoFactura = Record
            NroFact :Integer;
            Importe: Real,
            Articulos: PuntArtículo;
           sigFactura: PuntFactura
   end;

   NodoArticulo = Record
              NroArt:Integer;
              cant: Integer;
              sigArtículo:
       PuntArticulo
   end;

   var
      LasFacturas: PuntFactura;
      ……
```

Lista simple en la que cada nodos tiene un puntero a otra lista simple

```
type
 PuntFactura = ^NodoFactura;
 PuntArtículo = ^NodoArticulo;
  NodoFactura = Record
            NroFact :Integer;
            Importe: Real,
           Articulos: PuntArtículo;
          sigFactura: PuntFactura
   end;
```

```
NodoArticulo = Record
            NroArt:Integer;
            cant: Integer;
            sigArtículo: PuntArticulo
end;

var
    LasFacturas: PuntFactura;
     .........
```

```
BEGIN
New(LasFacturas);
New(LasFacturas^.sigFactura);
New(LasFacturas^.Artículos);
LasFacturas^.sigFActura^. sigFActura:= nil;
LasFacturas^.sigFActura^.Articulo:= nil;
LasFacturas^.Artículos^. sigArticulo:= nil;
```

- USAR PUNTEROS AUXILIARES
- MODULARIZAR

Lista simple en la que cada nodos tiene un puntero a otra lista simple

```
type
 PuntFactura = ^NodoFactura;
 PuntArtículo = ^NodoArticulo;
  NodoFactura = Record
            NroFact :Integer;
            Importe: Real,
           Articulos: PuntArtículo;
          sigFactura: PuntFactura
   end;

NodoArticulo = Record
            NroArt:Integer;
            cant: Integer;
            sigArtículo: PuntArticulo
end;

var
    LasFacturas: PuntFactura;
     ……..

   BEGIN
   New(LasFacturas);
   CrearArticulosFactura(LasFacturas);
```

Lista simple en la que cada nodos tiene un puntero a otra lista simple

```
type
 PuntFactura = ^NodoFactura;
 PuntArtículo = ^NodoArticulo;
  NodoFactura = Record
            NroFact :Integer;
            Importe: Real,
           Articulos: PuntArtículo;
          sigFactura: PuntFactura
   end;
```

```
NodoArticulo = Record
            NroArt:Integer;
            cant: Integer;
            sigArtículo: PuntArticulo
end;

var
    LasFacturas: PuntFactura;
     ……..
```

```
BEGIN
New(LasFacturas);
New(LasFacturas^.Artículos);
IngresarArticulos(LasFacturas^.Artículos);
```