Temas de Intro II

1.- Archivos

2.- Enumerado, Subrango y Registro

3.- Punteros y Listas

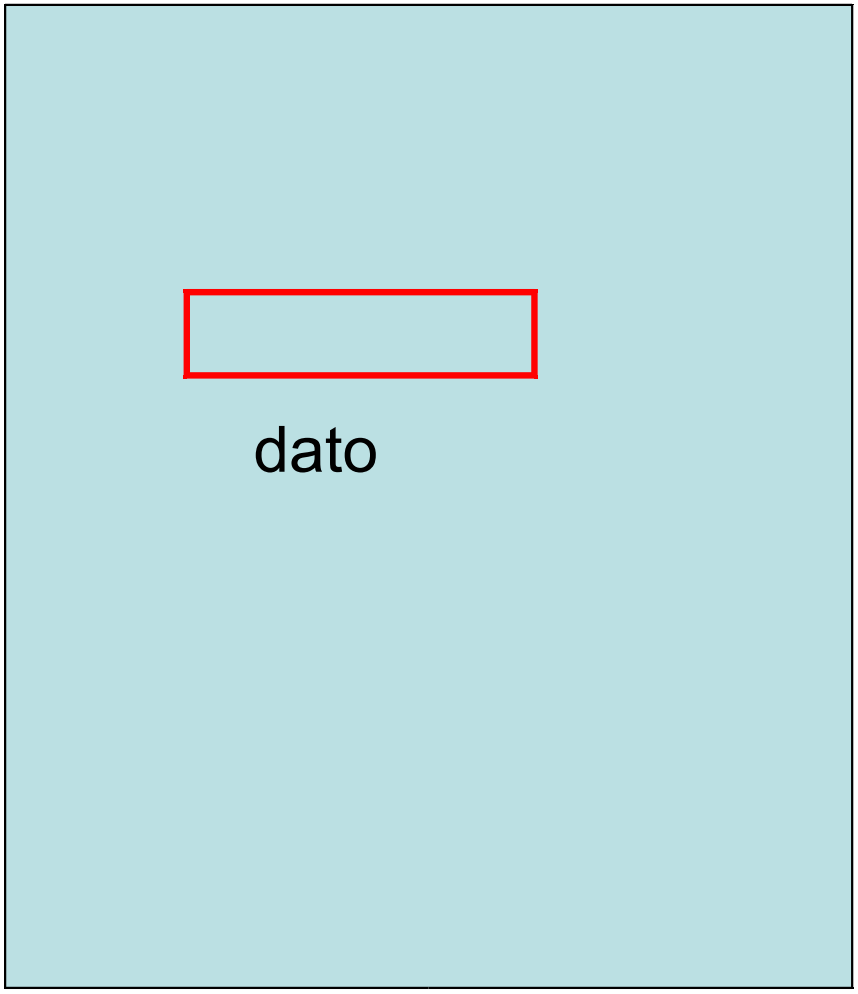
4.- Listas y Listas de Listas

5.- Recursión

6.- Árboles

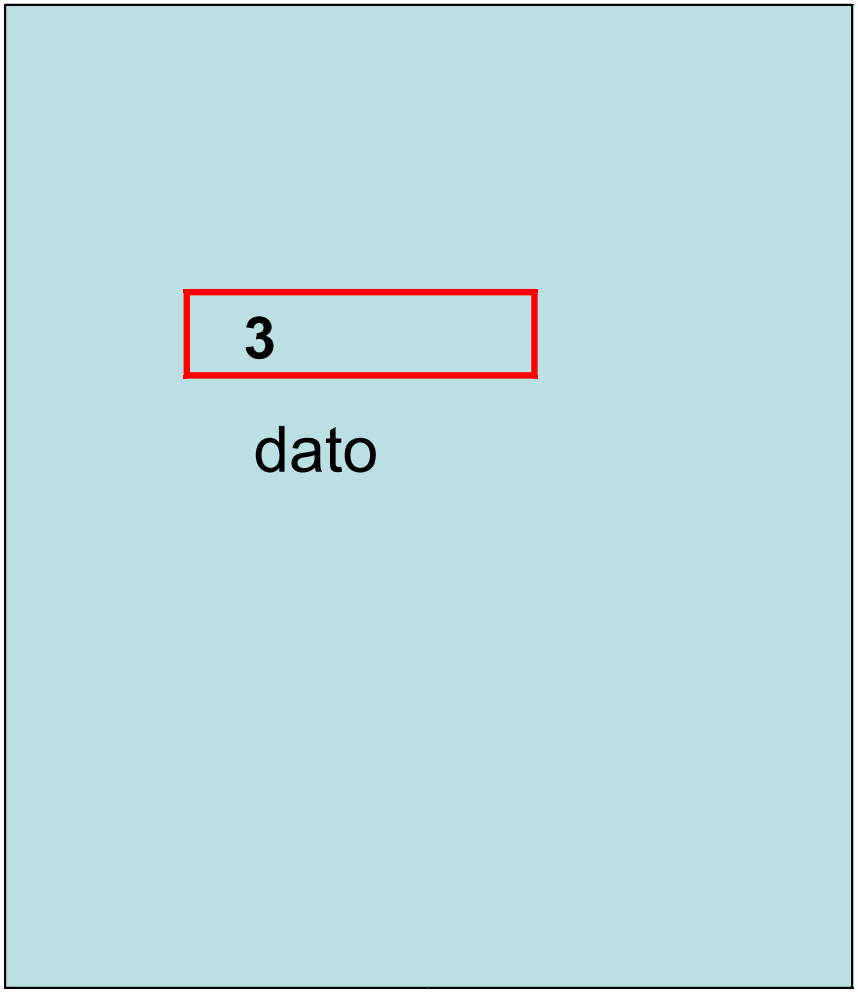
Asignación en Memoria estática

```
Type
.....
Var
      dato: integer;
     ......
Begin


End
```



Asignación en Memoria estática

```
Type
.....
Var
      dato: integer;
     ......
Begin
dato:=3;
End
```



Tipos de Asignación en Memoria

Estática

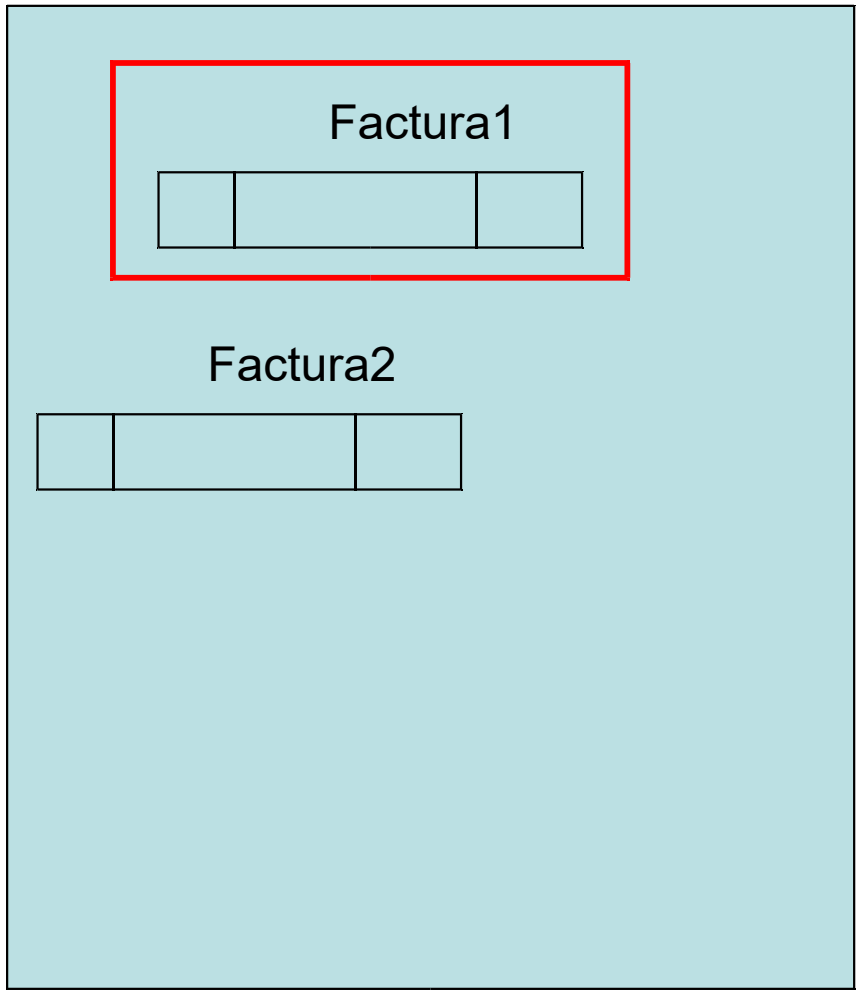
- Se asigna memoria cuando se define la variable
- No se puede usar más de lo definido, tampoco menos. Es necesario marcar límites
- Ejemplo: arreglos, matrices, variable de tipo integer, record.

Dinámica

- Se asigna memoria en ejecución (por demanda)
- No se asignan ni más ni menos de lo necesario
- Su uso es un poco mas engorroso
- Se hace mediante el mecanismo de Punteros

Asignación en Memoria dinámica

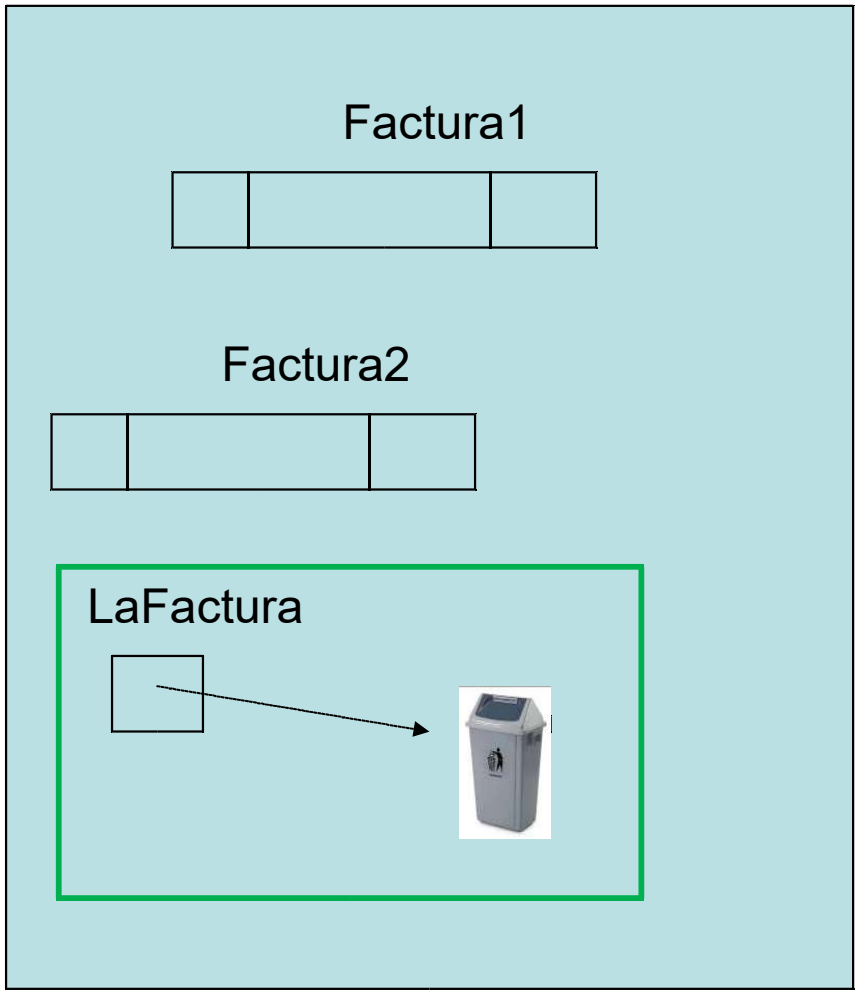
```
Type
 TipoFactura= Record
    Número:integer;
    Importe: real;
    Tipo: char
 End;

Var
 Factura1: TipoFactura;
 Factura2: TipoFactura;
```



Asignación en Memoria dinámica

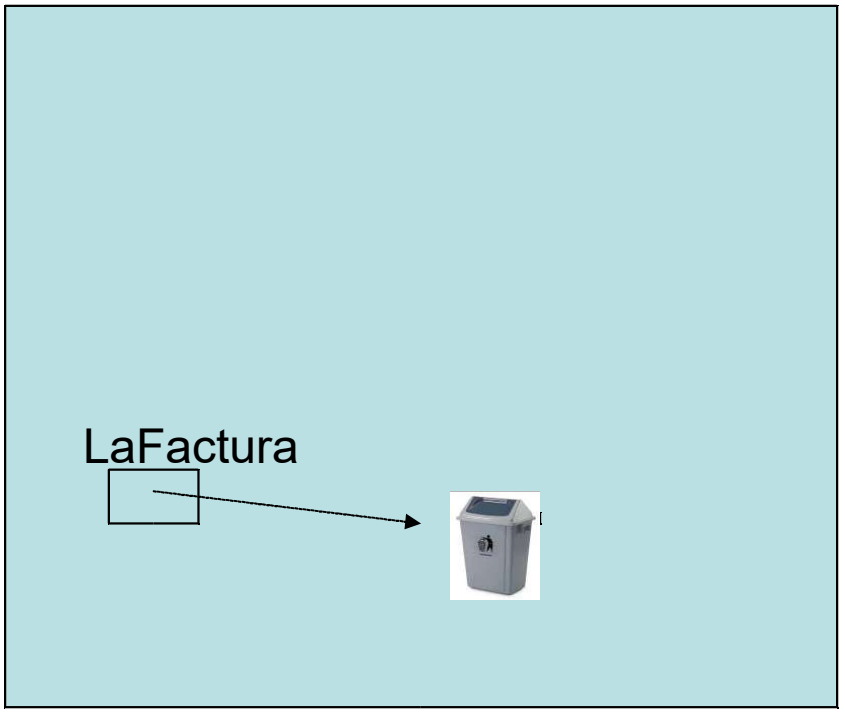
```
Type
 TipoFactura= Record
    Número:integer;
    Importe: real;
    Tipo: char
 End;

Var
 Factura1: TipoFactura;
 Factura2: TipoFactura;
 laFactura:  ^TipoFactura;
```

PUNTERO



Asignación en Memoria dinámica

```
Type
 TipoFactura= Record
     Número:integer;
     Importe: real;
     Tipo: char
 End;

Var

  laFactura:  ^TipoFactura;
```

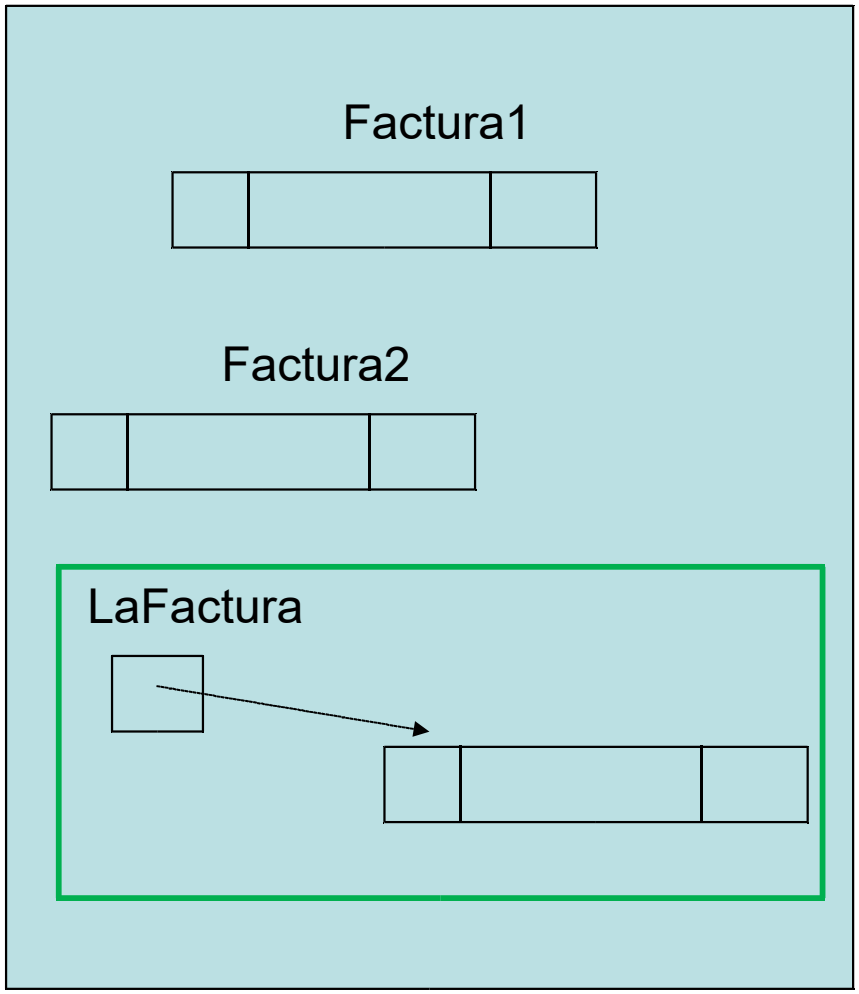


PUNTERO

ES UN TIPO DE VARIABLE EN EL CUAL SE ALMACENA LA DIRECCIÓN A UN DATO(SIMPLE O COMPUESTO). PERMITE MANEJAR DIRECCIONES “APUNTANDO” A UN ELEMENTO.

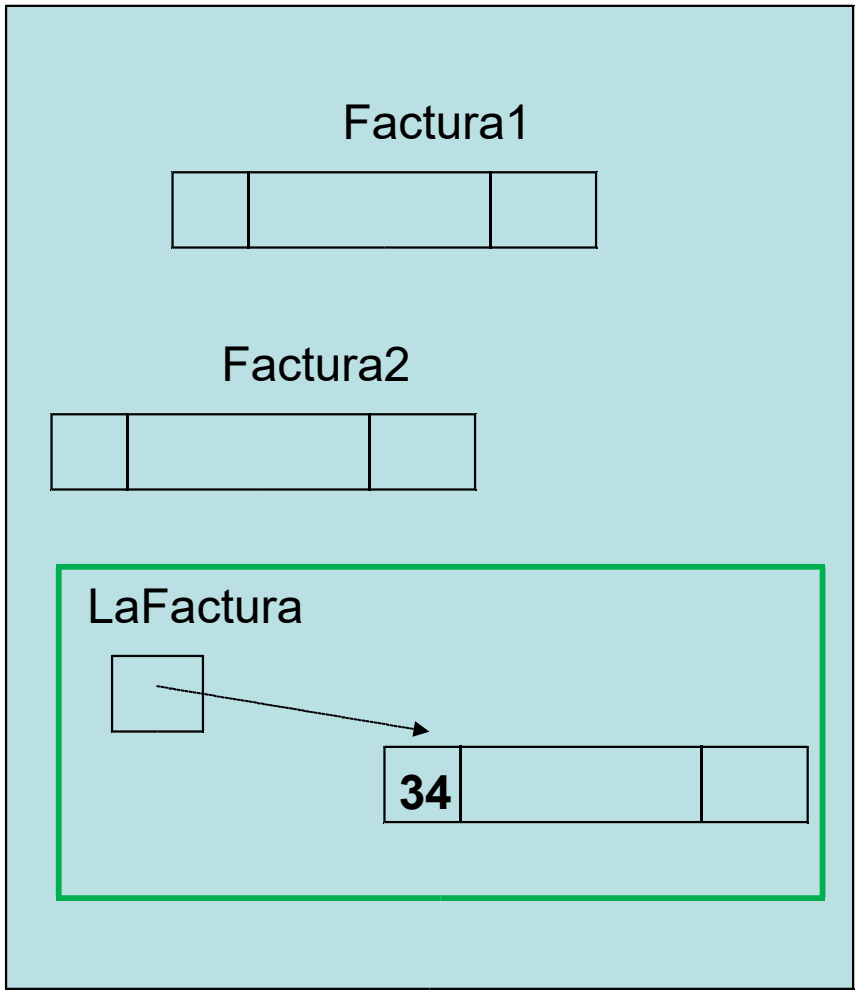
Asignación en Memoria dinámica

```
Type
 TipoFactura= Record
    Número:integer;
    Importe: real;
    Tipo: char
 End;

Var
 Factura1: TipoFactura;
 Factura2: TipoFactura;
 laFactura:  ^TipoFactura;

 Begin
    New(laFactura);
 …….
```



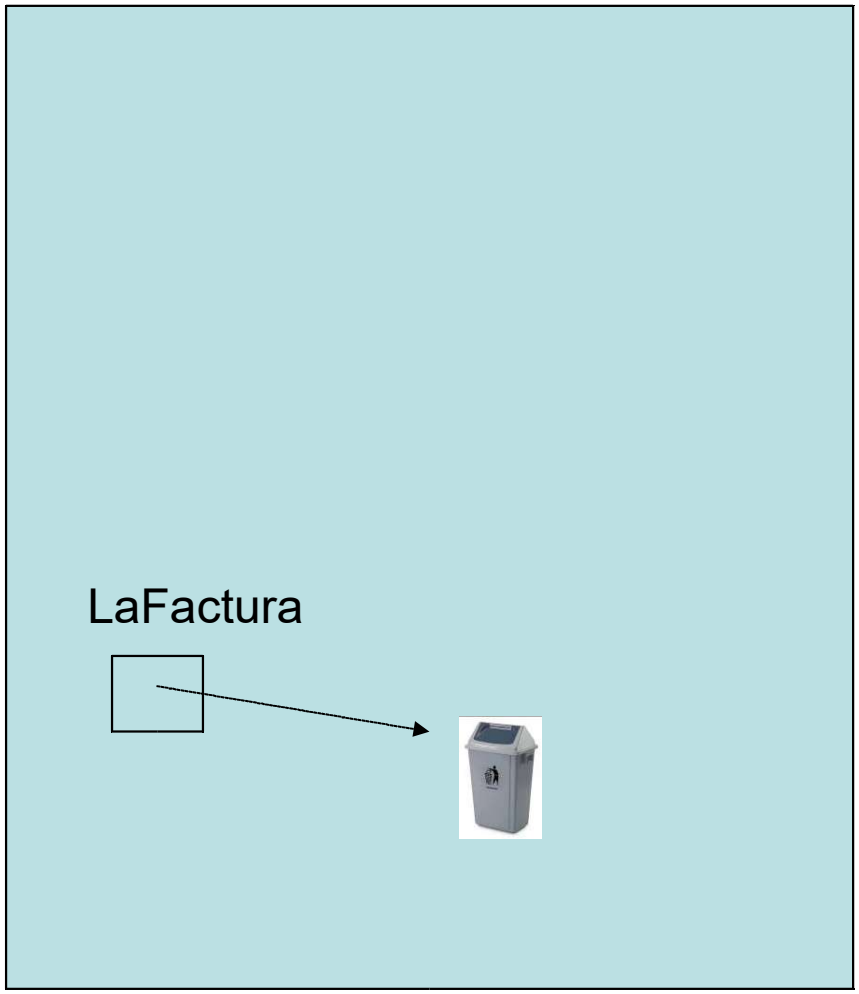
Asignación en Memoria dinámica

```
Type
 TipoFactura= Record
    Número:integer;
    Importe: real;
    Tipo: char
 End;

Var
 Factura1: TipoFactura;
 Factura2: TipoFactura;
 laFactura:  ^TipoFactura;

  Begin
    New(laFactura);
    laFactura^.Número:=34;
  …….
```



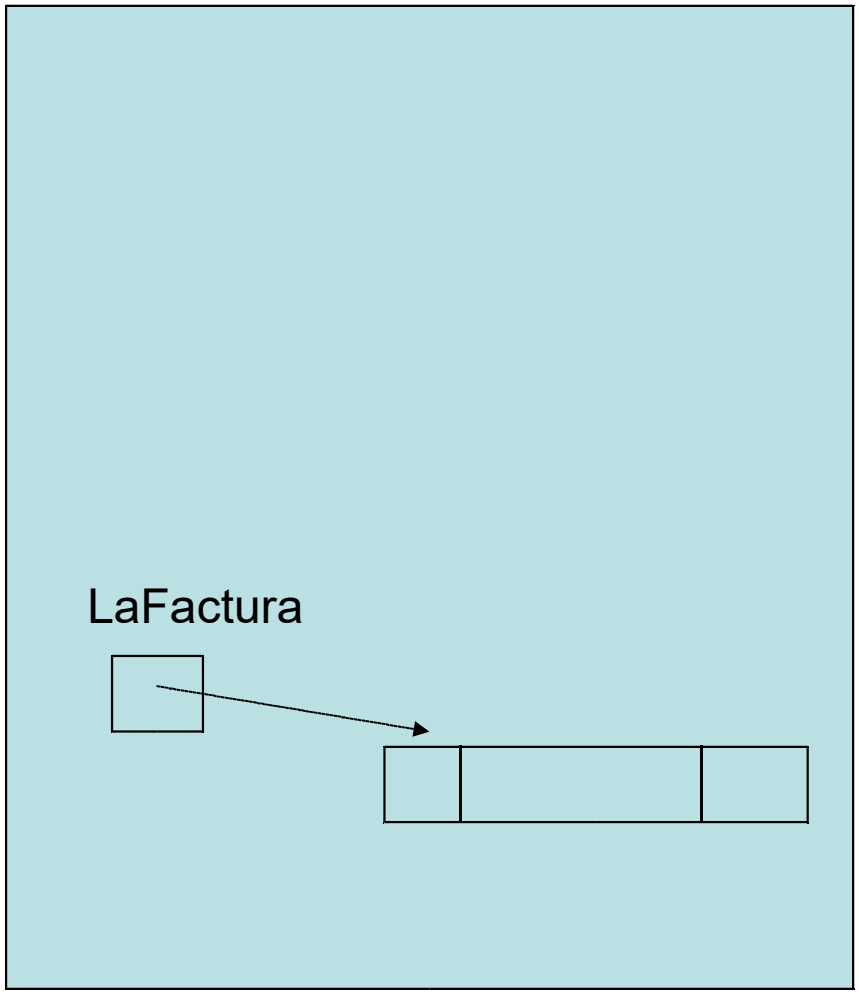
Doble new

```
Type
 TipoFactura= Record
     Número:integer;
     Importe: real;
     Tipo: char
 End;

Var
   laFactura:  ^TipoFactura;
```



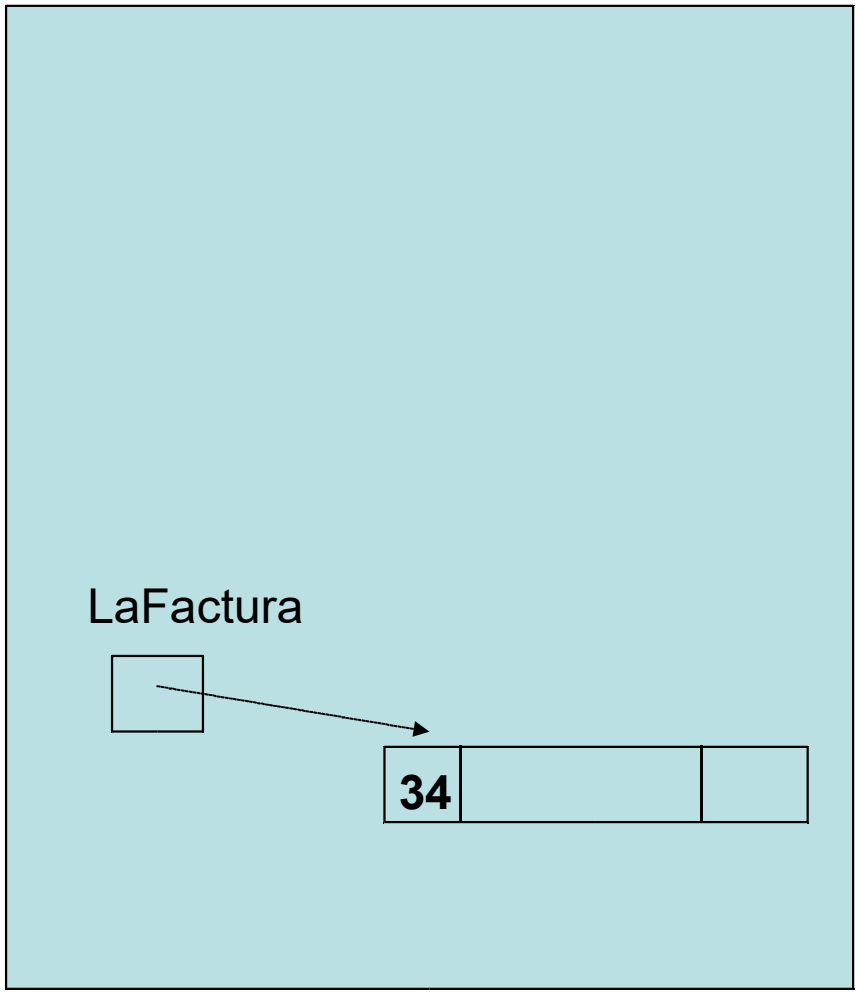
Doble new

```
Type
 TipoFactura= Record
     Número:integer;
     Importe: real;
     Tipo: char
 End;

Var
  laFactura:  ^TipoFactura;
   Begin
      new(laFactura);
   …….
```



Doble new

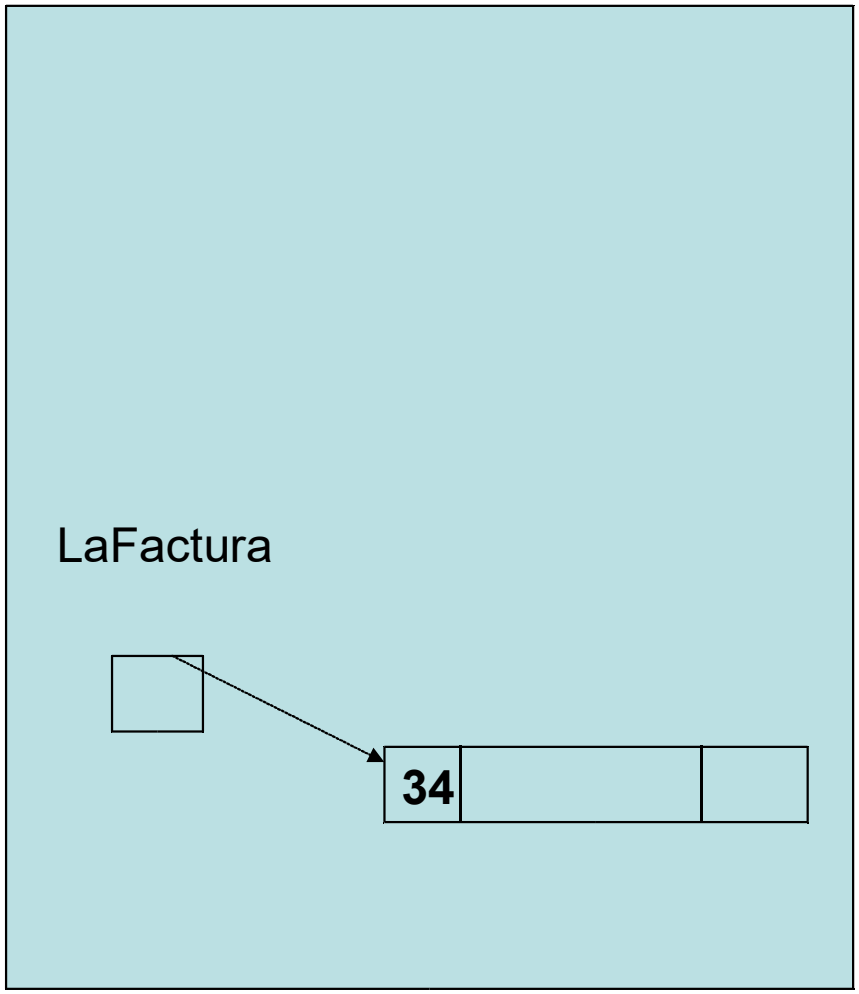
```
Type
 TipoFactura= Record
    Número:integer;
    Importe: real;
    Tipo: char
 End;

Var
  laFactura:  ^TipoFactura;
   Begin
     new(laFactura);
     laFactura^.Número:= 34;
```



Doble new

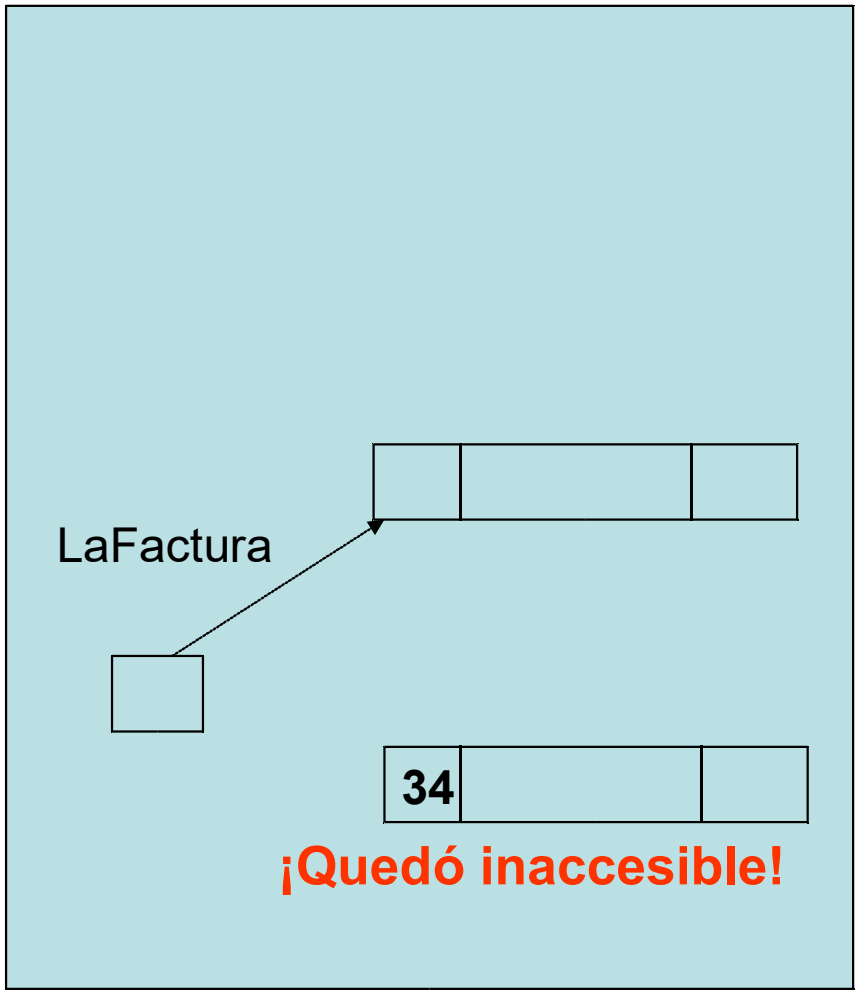
```
Type
 TipoFactura= Record
    Número:integer;
    Importe: real;
    Tipo: char
 End;

Var
  laFactura:  ^TipoFactura;
   Begin
     new(laFactura);
     laFactura^.Número:= 34;
     new(laFactura);
   ……
```

¿ Qué sucede?



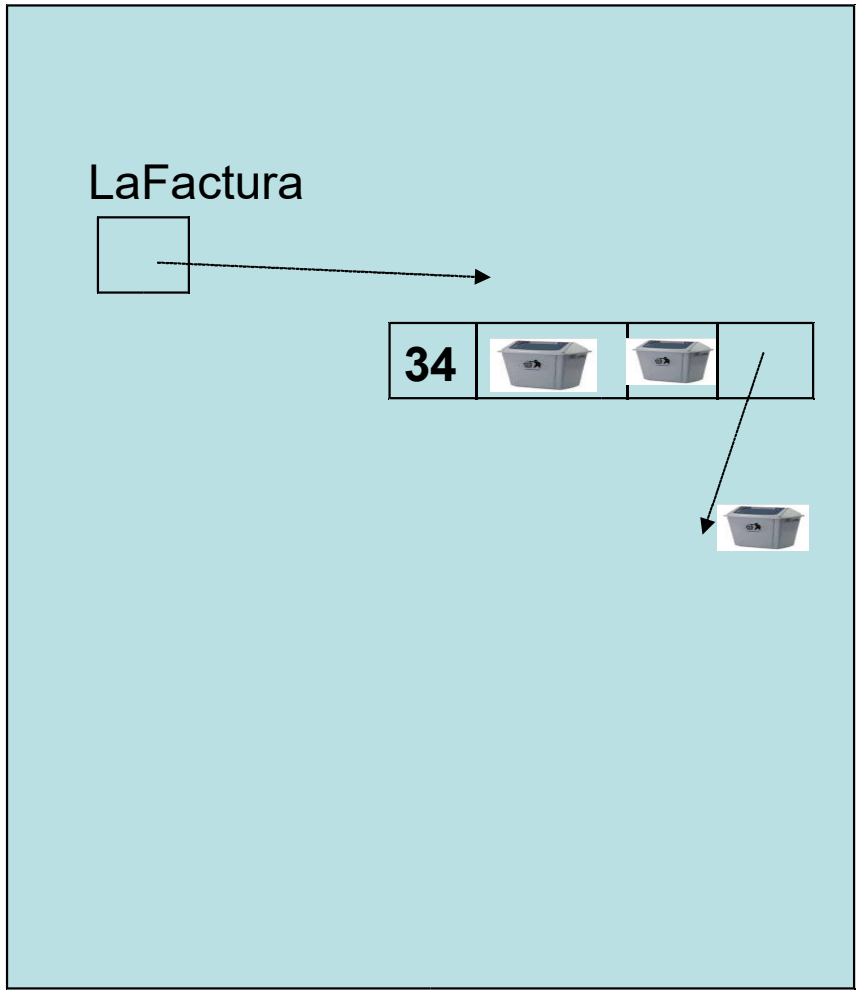
Doble new

```
Type
 TipoFactura= Record
    Número:integer;
    Importe: real;
    Tipo: char
 End;

Var
  laFactura:  ^TipoFactura;
   Begin
     new(laFactura);
     laFactura^.Número:= 34;
     new(laFactura);
   ……
```



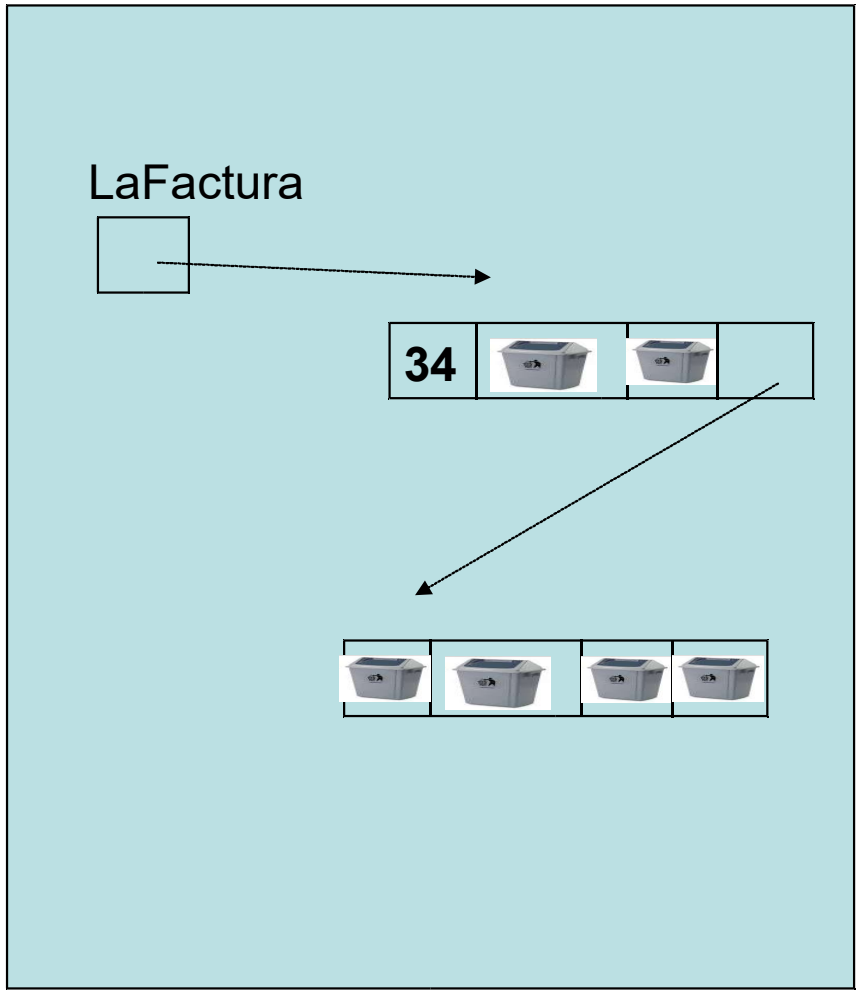
Una lista de Facturas

```
Type
ListaFactura= ^ TipoFactura
TipoFactura = Record
    Número:integer;
    Importe: real;
    Tipo: char
    siguienteFactura: ListaFactura
End;
Var
   laFactura:  ^TipoFactura;

 Begin
   new(laFactura);
   laFactura^.Número:= 34;
```



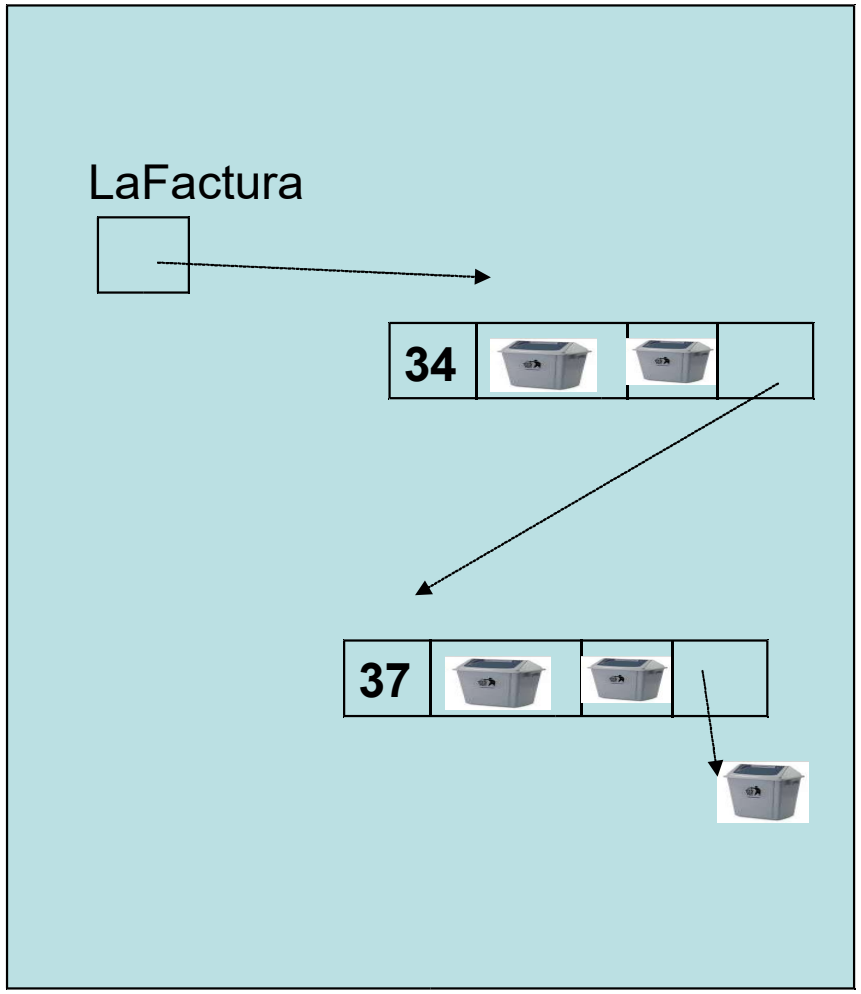
Una lista de Facturas

```
Type
ListaFactura= ^ TipoFactura
TipoFactura = Record
    Número:integer;
    Importe: real;
    Tipo: char
    siguienteFactura: ListaFactura
End;
Var
    laFactura:  ^TipoFactura;

 Begin
    new(laFactura);
    laFactura^.Número:= 34;
    new(laFactura^.siguienteFactura);
```



Una lista de Facturas

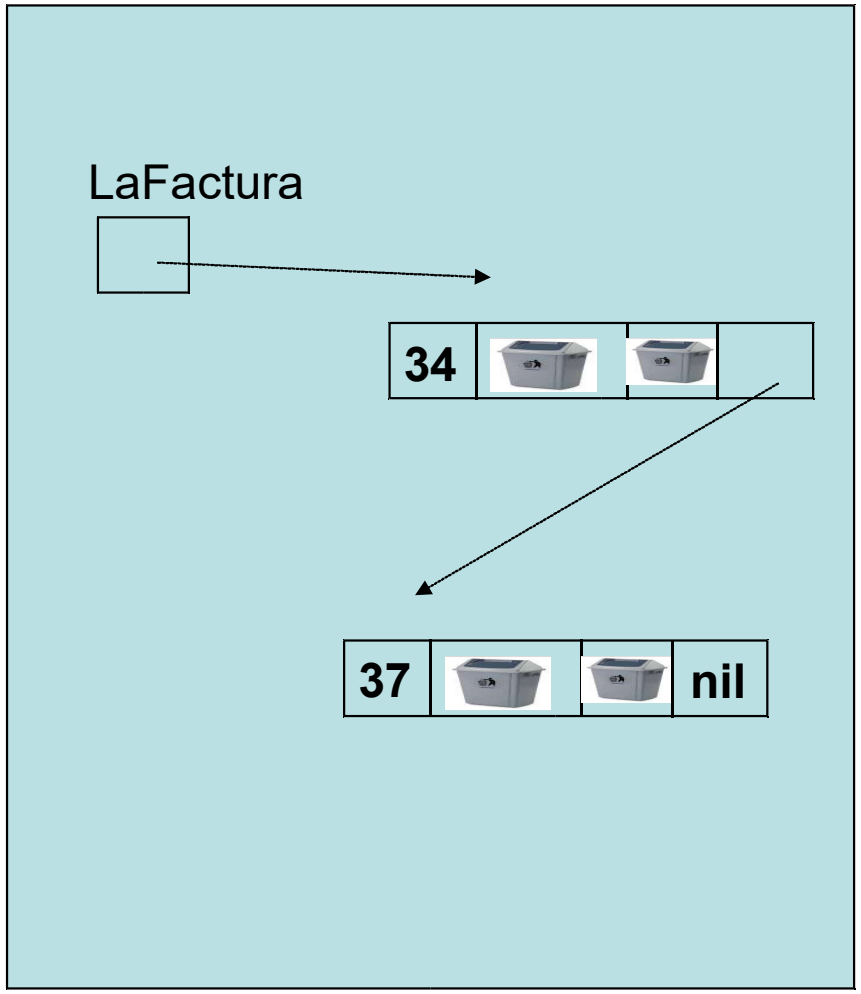
```
Type
ListaFactura= ^ TipoFactura
TipoFactura = Record
    Número:integer;
    Importe: real;
    Tipo: char
    siguienteFactura: ListaFactura
 End;
  laFactura:  ^TipoFactura;
Var
Begin
  new(laFactura);
  laFactura^.Número:= 34;
  new(laFactura^ .siguienteFactura);
```




```
  laFactura^.siguienteFactura^.Número := 37;
```

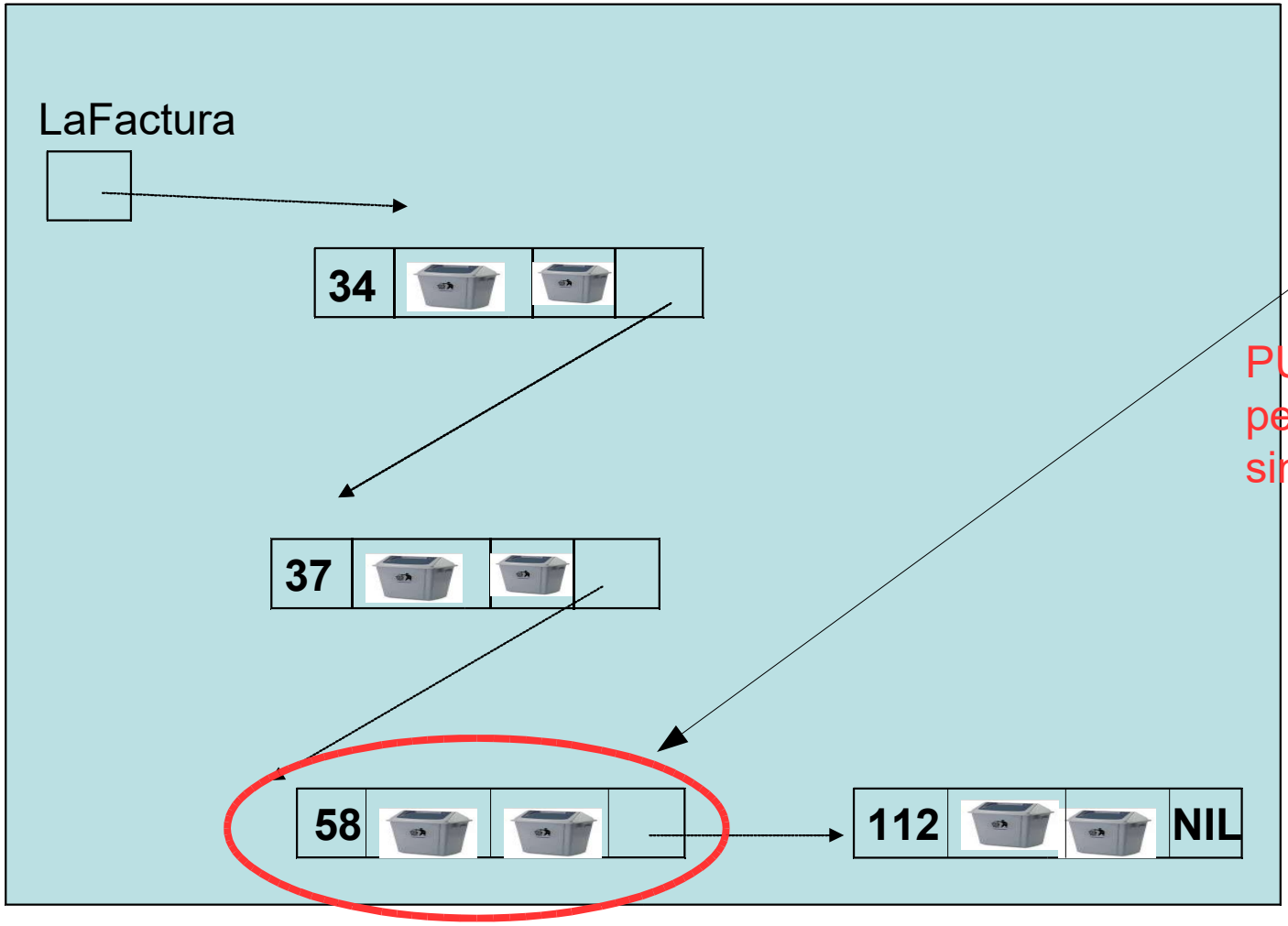
Una lista de dos nodos con Facturas

```
Type
ListaFactura= ^ TipoFactura
TipoFactura = Record
    Número:integer;
    Importe: real;
    Tipo: char
    siguienteFactura:= ListaFactura
 End;
  laFactura:  ^TipoFactura;
Var
Begin
  new(laFactura);
  laFactura^.Número:= 34;
  new(laFactura^ .siguienteFactura);
```




```
  laFactura^.siguienteFactura^.Número := 37;
  laFactura^.siguienteFactura^.siguienteFactura := nil;
```

Una lista de varios nodos con Facturas



FacturaActual

PUNTERO auxiliar que permita recorrer la lista, sin perder el inicio.

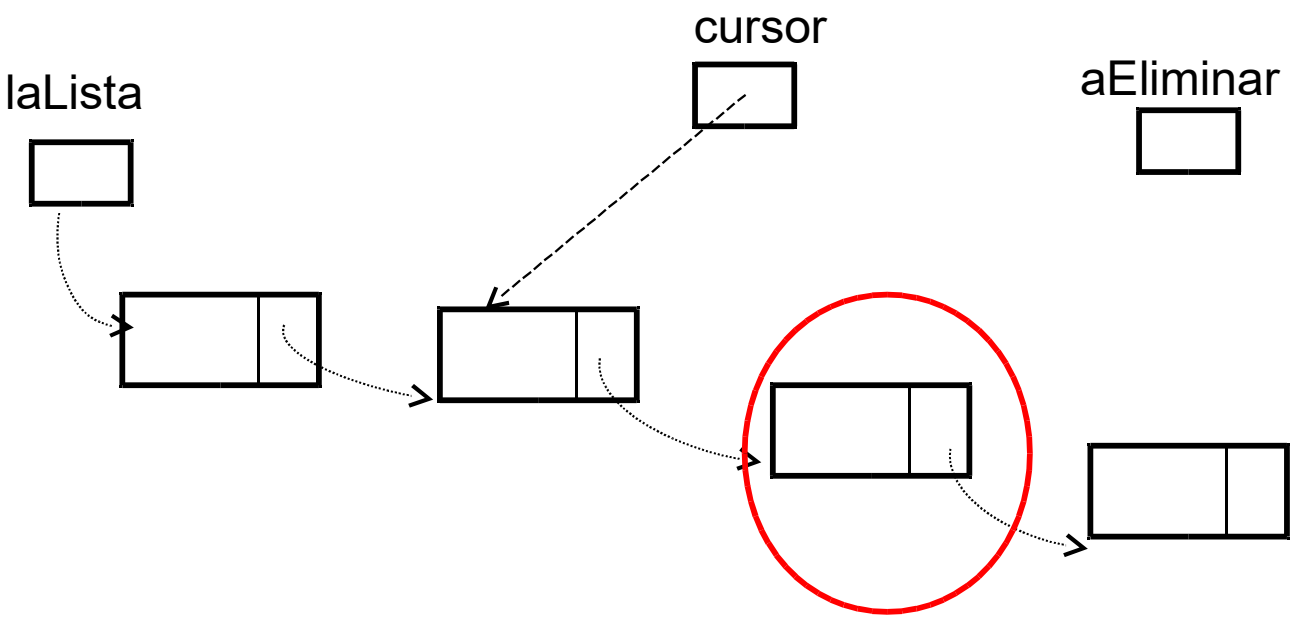
```
program CrearListaSinProcedures;
{ Este programa crea una lista de enteros de longitud variable hasta que el usuario entre un valor menor o igual a cero}

type
  PuntNodo = ^NodoLista;
  NodoLista = Record
            Nro :Integer;
           sig :PuntNodo
            end;
var
   Lalista, cursor: PuntNodo;
                valor: integer;
begin
Lalista:= nil;
readln(valor);
 if valor > 0  then begin  { creo primer nodo de la lista}
                  new(Lalista);
                  laLista^.Nro:= valor;
                  laLista^.sig:= nil;  {no hay otro nodo}
                  cursor:= Lalista; {guardo en curso el INICIO de la lista}
                  readln(valor);
                   While (valor > 0) do  begin
                     new(cursor^.sig);
                      cursor:= cursor^.sig;
                      cursor^.Nro:=valor;
                      cursor^.sig:=nil;
                      readln(valor);
              end; { del while}
  end; { del if}
end.
```

```
program CrearListaSinProcedures;
{ Este programa crea una lista de enteros de longitud variable hasta que el usuario entre un valor menor o igual a cero}
…….
Lalista:= nil;
readln(valor);
 if valor > 0  then begin  { creo primer nodo de la lista}
              new(Lalista);
              laLista^.Nro:= valor;
              laLista^.sig:= nil;  {no hay otro nodo}
              cursor:= Lalista; {guardo en curso el INICIO de la lista}
              readln(valor);
               While (valor > 0) do  begin
                  new(cursor^.sig);
                   cursor:= cursor^.sig;
                   cursor^.Nro:=valor;
                   cursor^.sig:=nil;
                   readln(valor);
                  end; { del while}
  end; { del if}
end.
```

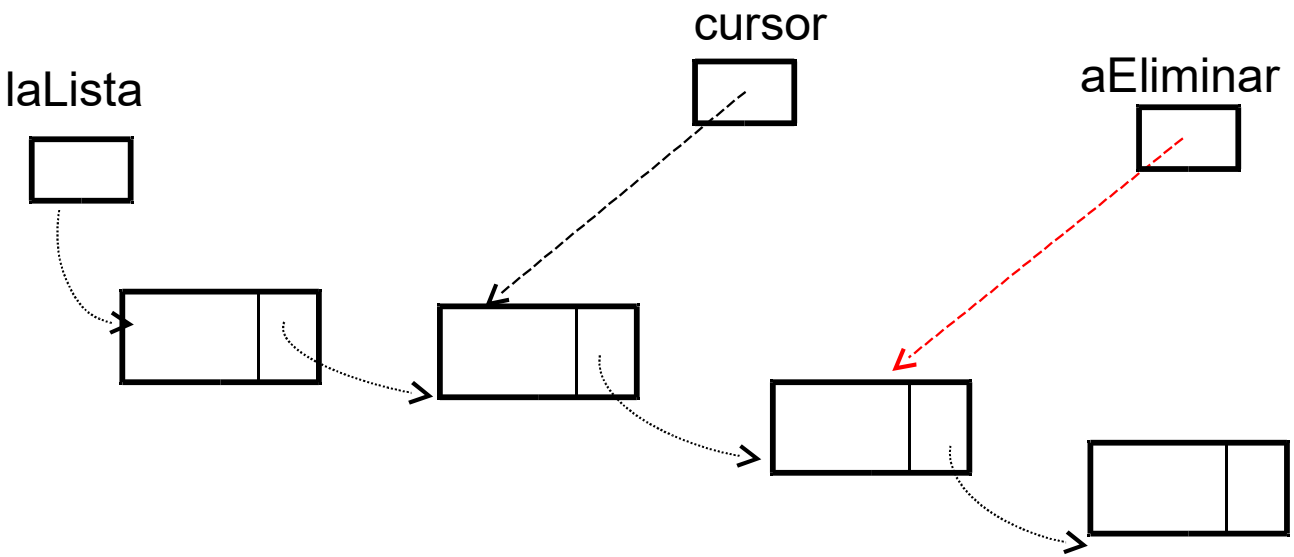
Para recordar:

- Diferenciar el puntero INICIAL de los auxiliares para los recorridos.
- Cuidar el uso del INICIAL. NUNCA perder el primer nodo
- Nombres significativos a los punteros para no confundirse ni confundir
- Crear los punteros con new o inicializar a nil.

Borrado de un nodo

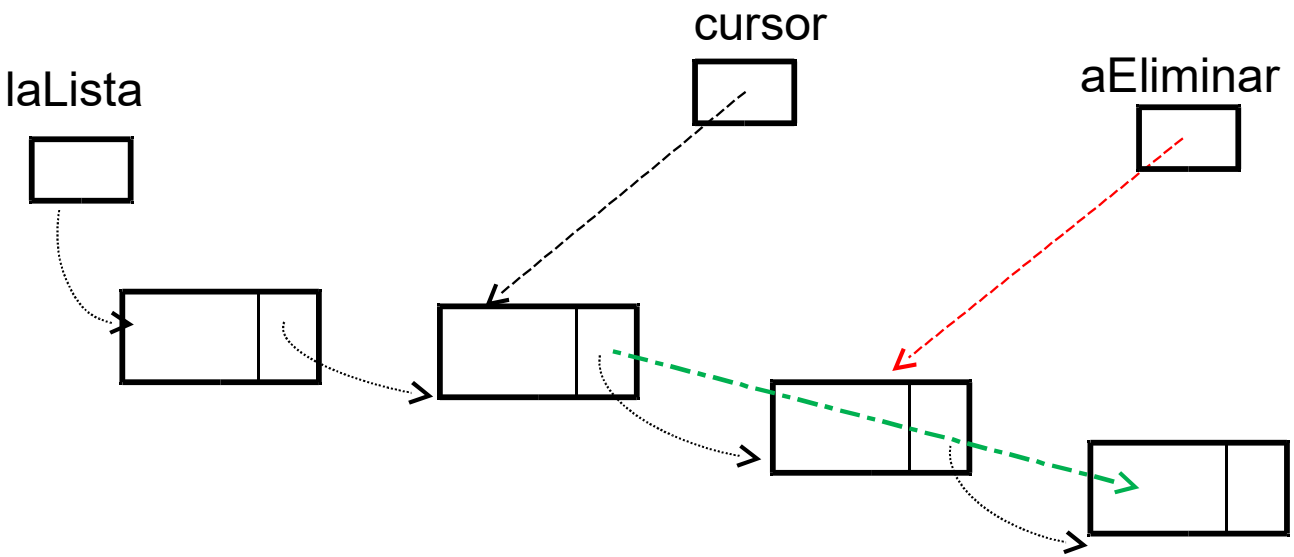


Borrado de un nodo



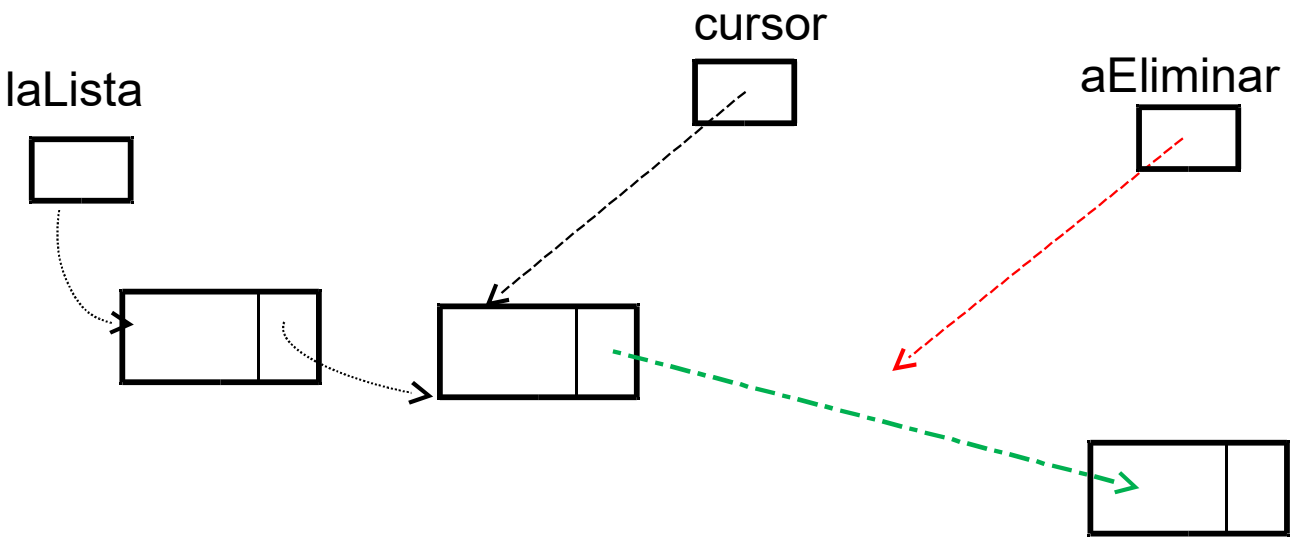
……
aEliminar:= cursor^.sig;

Borrado de un nodo



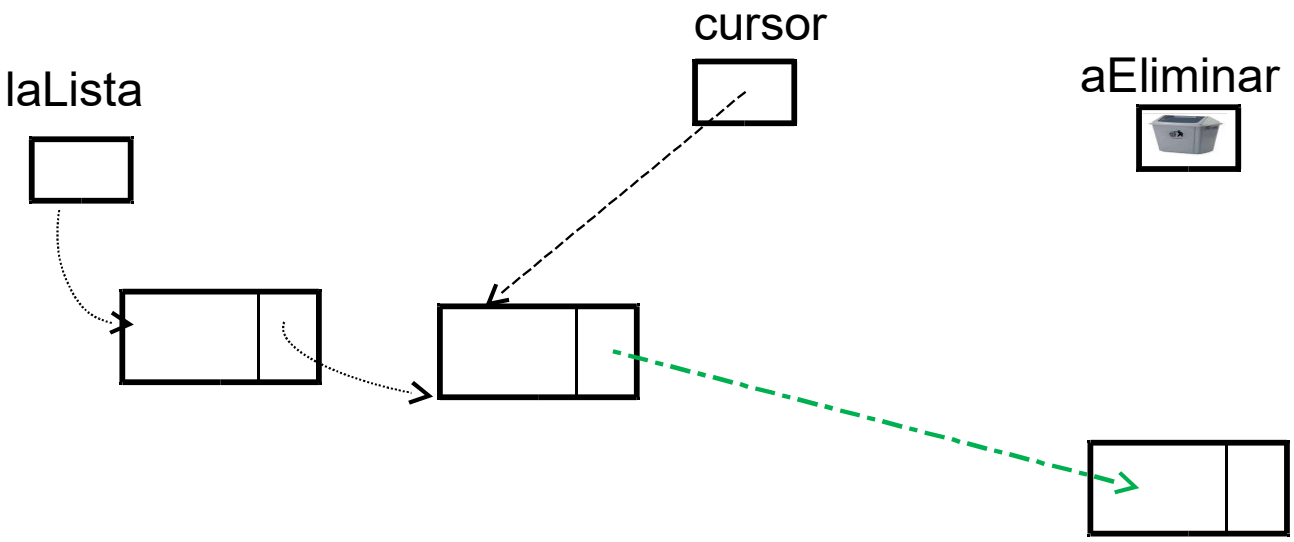

```
……
aEliminar:=  cursor^.sig;
cursor^.sig := cursor^.sig^.sig;    // cursor^.sig := aEliminar^.sig
```

Borrado de un nodo



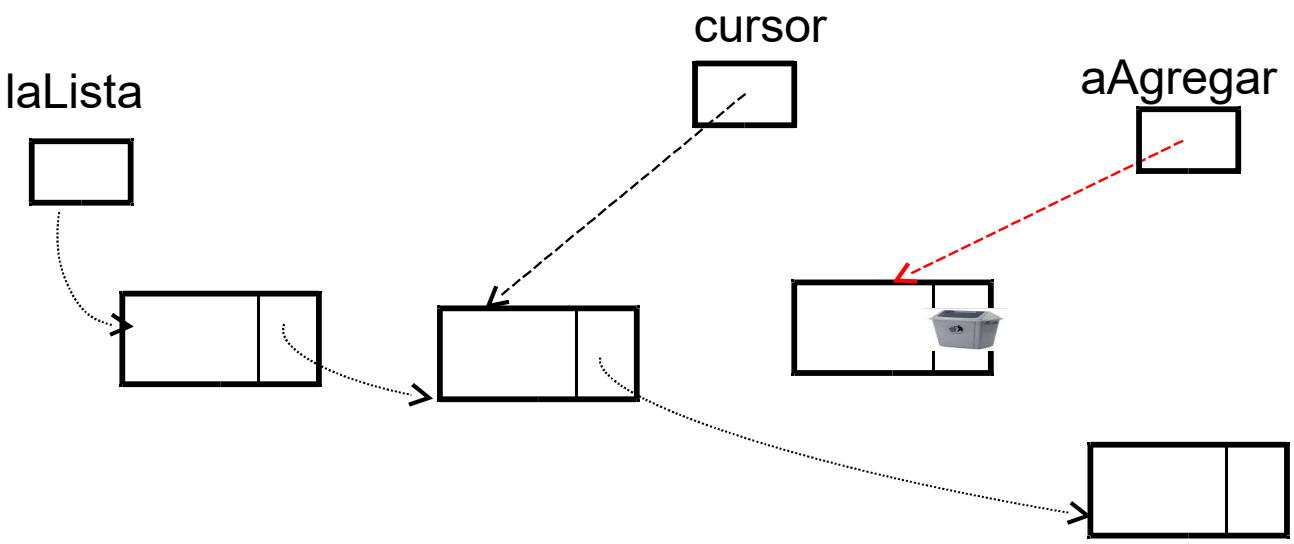

```
……
aEliminar:=  cursor^.sig;
cursor^.sig := cursor^.sig^.sig;
Dispose(aliminar);
```

Borrado de un nodo

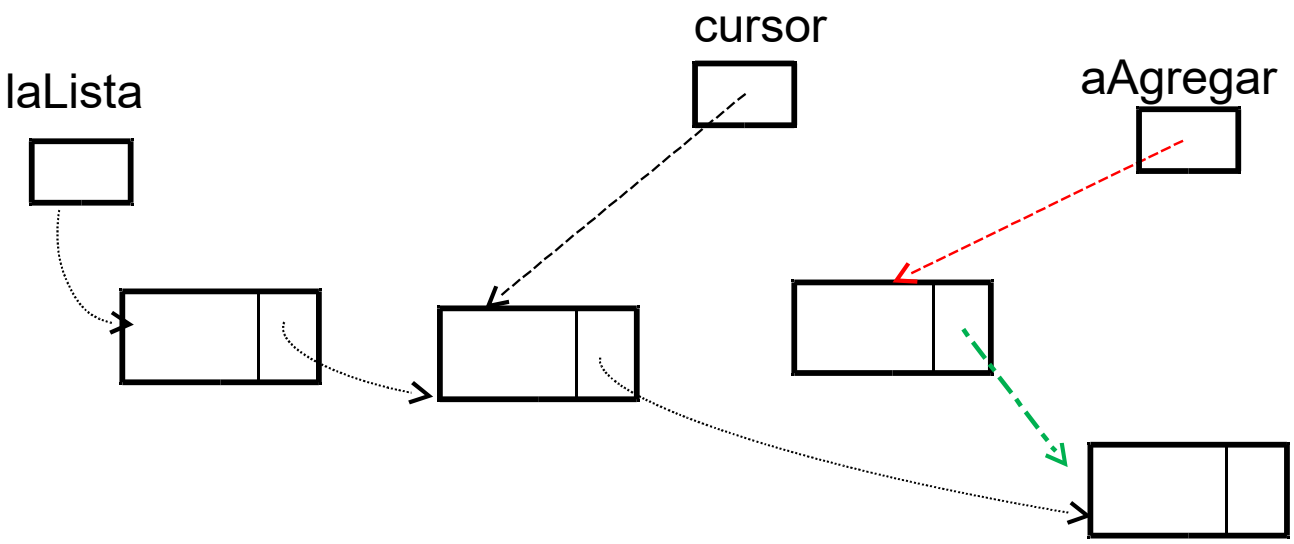



```
……
aEliminar:=  cursor^.sig;
cursor^.sig := cursor^.sig^.sig;
Dispose(aEliminar);
aEliminar:= nil;
```

Alta de un nodo

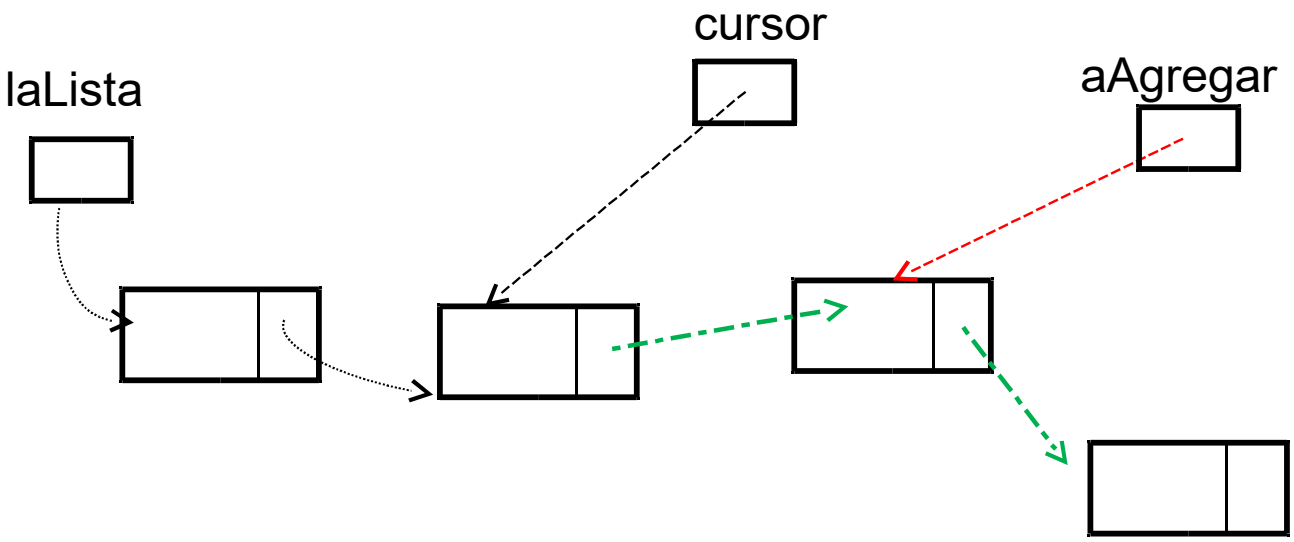


Alta de un nodo




```
……
aAgregar^.sig:= cursor^.sig;
```

Alta de un nodo




```
……
aAgregar^.sig:= cursor^.sig;
cursor ^.sig := aAgregar ;
```

Punteros como parámetros

Lo que se pasa por parámetro es la variable tipo puntero

- POR REFERENCIA: se pasa la propia variable, por lo que **TODA** modificación impactará de igual, forma que se estuviera en el módulo invocador.

- POR COPIA: se pasa una copia de la variable, por lo que toda modificación de **SU Valor NO** impactará, pero **SI** todos los cambios que se hagan en los **nodos** de la lista.