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

Recursión

Repetición por autoreferencia, cuando un procedimiento o función se invoca a si mismo (de manera directa o indirecta)

Es la forma en la cual se especifica un proceso basado en su propia definición pero con menor complejidad. De esta forma en algún momento se llega a un proceso muy simple que puede resolverse fácilmente.

Ejemplos

1- Definición por inducción

El cero pertenece a los Naturales, y si N pertenece a los naturales N+1 también

2- Estructura de datos recursivas

El nulo (nil) es una Lista y un Nodo seguido de una Lista es también una lista

Ejemplo: Factorial

Definición por inducción

- Factorial(0):=1
- Factorial (N):=N * Factorial (N-1)

para todo N >0

Solución iterativa

```
Function Factorial (n:integer): integer;
var
   fac_aux: integer;
begin
   fac_aux:= 1;
   if N > 0 then
    begin
      fac_aux:= N;
      while n > 1 do begin
          n := n -1;
          fac_aux:= fac_aux  * n;
      end;
   end;
  Factorial:=   fac_aux;
end;
```

Solución recursiva

```
Function Factorial(N:integer):integer;
begin
   if N = 0
      then
        Factorial:= 1
      else
        Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(2):integer;
begin
   if 2 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (1) * 2;
end;
```

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(2):integer;
begin
   if 2 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (1) * 2;
end;
```

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(2):integer;
begin
   if 2 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (1) * 2;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(2):integer;
begin
   if 2 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (1) * 2;
end;
```

```
Function Factorial(1):integer;
begin
   if 1 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (0) * 1;
end;
```

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(2):integer;
begin
   if 2 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (1) * 2;
end;
```

```
Function Factorial(1):integer;
begin
   if 1 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (0) * 1;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(2):integer;
begin
   if 2 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (1) * 2;
end;
```

```
Function Factorial(1):integer;
begin
   if 1 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (0) * 1;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(2):integer;
begin
   if 2 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (1) * 2;
end;
```

```
Function Factorial(1):integer;
begin
   if 1 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (0) * 1;
end;
```

```
Function Factorial(0):integer;
begin
   if 0 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (0) * -1;
end;
```

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(2):integer;
begin
   if 2 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (1) * 2;
end;
```

```
Function Factorial(1):integer;
begin
   if 1 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (0) * 1;
end;
```

```
Function Factorial(0):integer;
begin
   if 0 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (0) * -1;
end;
```

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(2):integer;
begin
   if 2 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (1) * 2;
end;
```

```
Function Factorial(1):integer;
begin
   if 1 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (0) * 1;
end;
```

```
Function Factorial(0):integer;
begin
   if 0 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (0) * -1;
end;
```

1

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(2):integer;
begin
   if 2 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (1) * 2;
end;
```

```
Function Factorial(1):integer;
begin
   if 1 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (0) * 1;
end;
```

Function Factorial(**0**) → **1**

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(2):integer;
begin
   if 2 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (1) * 2;
end;
```

```
Function Factorial(1):integer;
begin
   if 1 = 0
    then  Factorial:= 1
    else  Factorial:= 1 * 1;
end;
```

1

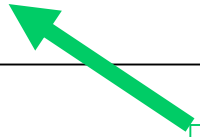
Function Factorial(**0**) →

1

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(2):integer;
begin
   if 2 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (1) * 2;
end;
```



Function Factorial(1) → 1

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial(2):integer;
begin
   if 2 = 0
    then  Factorial:= 1
    else  Factorial:= 1  * 2;
end;
```

2

```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (2) * 3;
end;
```

Function Factorial(**2**) →

2

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

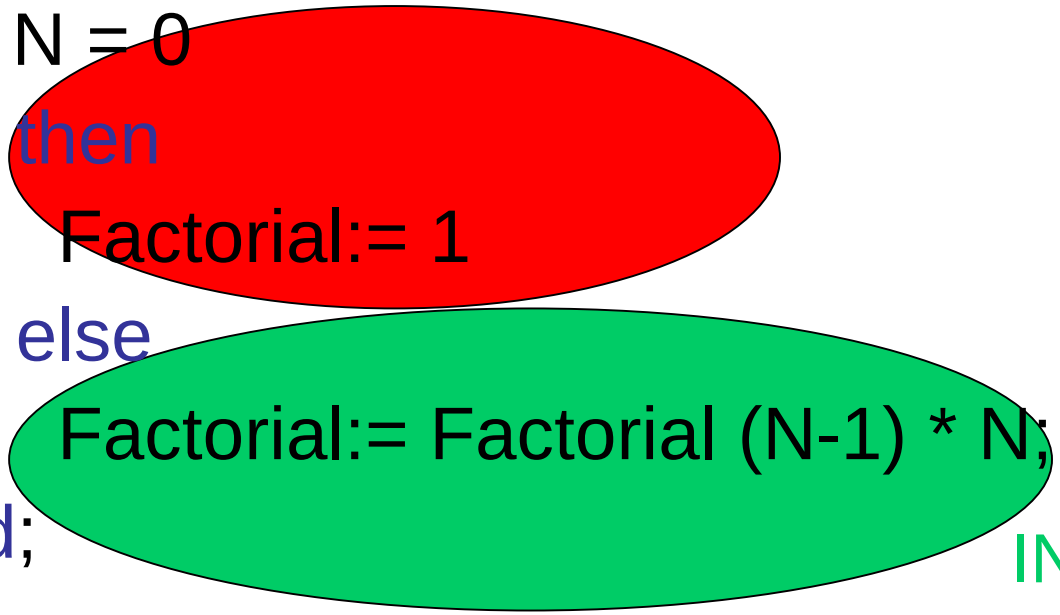
```
Function Factorial(3):integer;
begin
   if 3 = 0
    then  Factorial:= 1
    else  Factorial:= 2 * 3;
end;
```

```
Function Factorial(N:integer):integer;
begin
   if N = 0
    then  Factorial:= 1
    else  Factorial:= Factorial (N-1) * N;
end;
```

Function Factorial(3) → **6**

Características a tener en cuenta es Soluciones recursivas

```
Function Factorial(N:integer):integer;
begin
   if N = 0
      then
        Factorial:= 1
      else
        Factorial:= Factorial (N-1) * N;
end;
```



CORTE

INVOCAR RECURSIVAMENTE más cerca del FINAL (problema más “chico”)

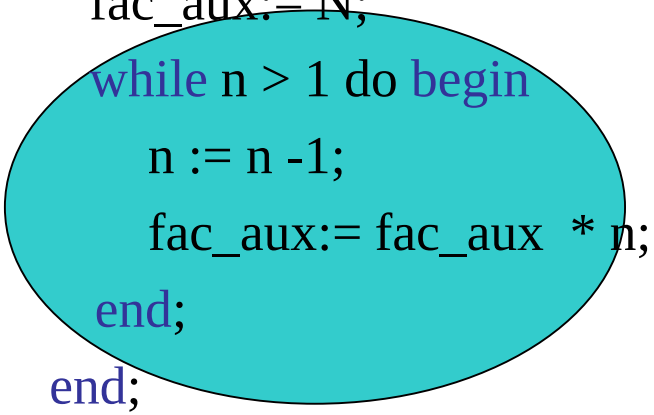
Características a tener en cuenta es Soluciones recursivas

- Hay una condición de corte, es decir que por ese camino NO se invoca a la misma función/procedimiento. Si es función, se asigna resultado

- Si no se da el CORTE, es decir, se invoca a la misma función/procedimiento la próxima vez se estará mas cerca del corte

```
Function Factorial(N:integer):integer;
begin
  if N = 0
    then
      Factorial:= 1
    else
      Factorial:= Factorial (N-1) * N;
end;
```

```
Function Factorial (n:integer): integer;
var
   fac_aux: integer;
begin
  fac_aux:= 1;
  if N > 0 then
    begin
      fac_aux:= N;
      while n > 1 do begin
          n := n -1;
          fac_aux:= fac_aux  * n;
      end;
   end;
Factorial:=   fac_aux;
end;
```



```
Function Factorial(N:integer):integer;
begin
   if N = 0
      then
        Factorial:= 1
      else
        Factorial:= Factorial (N-1) * N;
end;
```

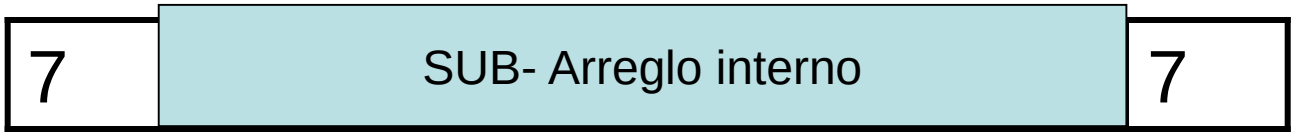
Ejercicio

Problema: Determinar si un arreglo de caracteres es capicúa

7	6	3	1	1	3	6	7

Ejemplo Recursión

Estrategia: pensar en que un arreglo es capicúa si el primer elemento es igual al último y el sub-arreglo interno es capicúa.

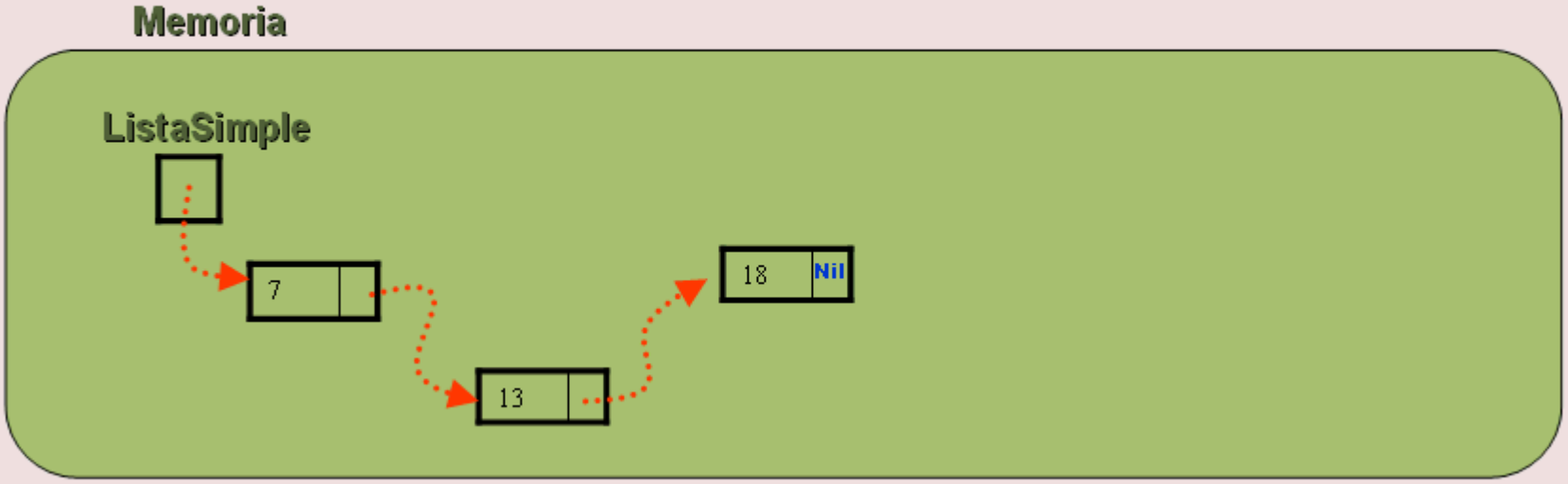


```
Function EsCapicua(ar:arreglo; init,fin:integer):boolean;
begin
if init >= fin
        then
          EsCapicua:= true
        else
            if ar[init] = ar[fin]
            then
                        EsCapicua:= Escapicua( ar, init+1, fin-1)
            else
                        EsCapicua:= false

end;
```

7	6	3	1	1	3	6	7

Ejemplo: Recorrer lista con recursión



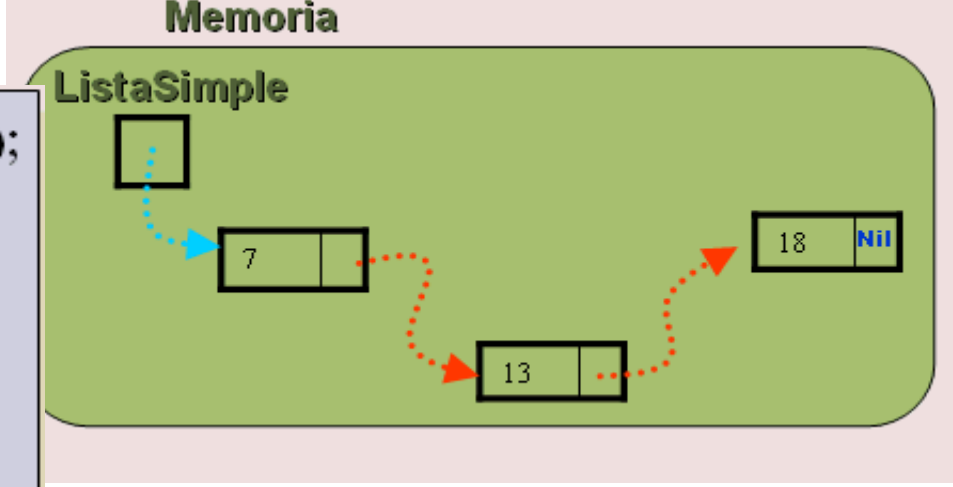
Mostrar Datos Ordenados

Mostrar Datos Invertidos

```
Procedure ImpLista(P:PuntLista);
begin
   if (P <> nil) then begin
      writeln(P^.Dato);
      ImpLista(P^.Sig);
   end;
end;
```

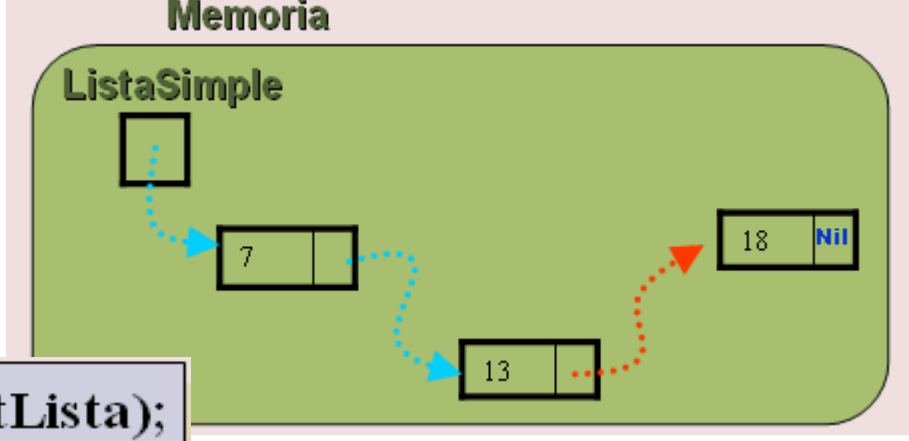
```
Procedure ImpListaInv(P:PuntLista);
begin
   if (P <> nil) then begin
      ImpListaInv(P^.Sig);
      writeln(P^.Dato);
   end;
end;
```

```
Procedure ImpListaInv(P:PuntLista);
begin
   if (P <> nil) then begin
      ImpListaInv(P^.Sig);
      writeln(P^.Dato);
   end;
end;
```



```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        wr
    end;
end;
```

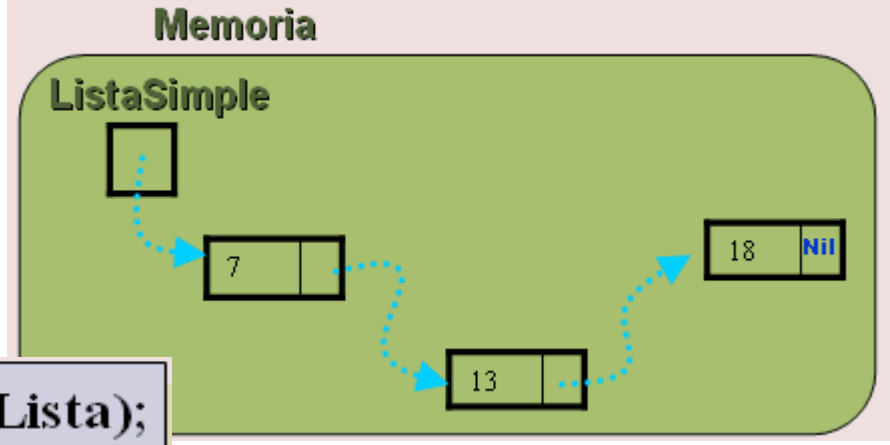
```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        writeln(P^.Dato);
    end;
end;
```

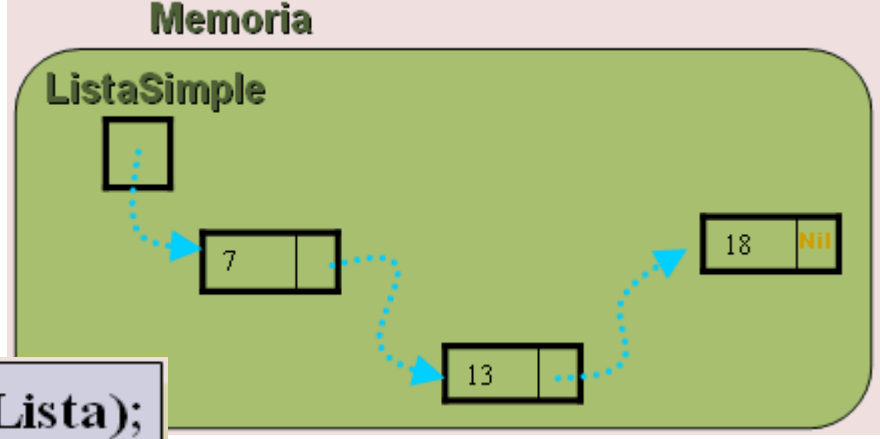


```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        wr
    end;
end;
```

```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        wr
    end;
end;
```

```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        writeln(P^.Dato);
    end;
end;
```



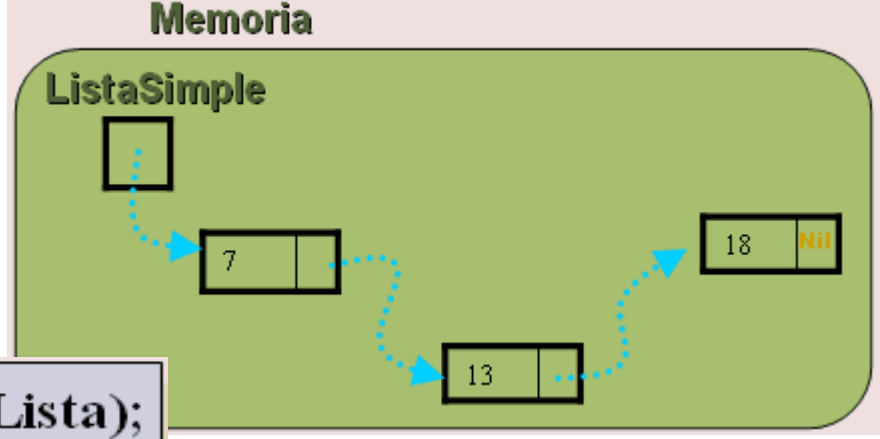



```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        wr
    end;
end;
```

```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        wr
    end;
end;
```

```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        w
    end
end;
```

```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        writeln(P^.Dato);
    end;
end;
```




```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        wr
    end;
end;
```

```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        wr
    end;
end;
```

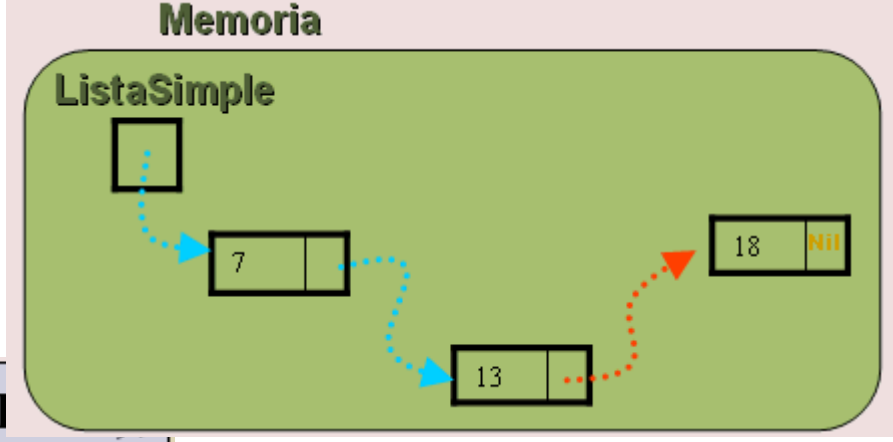
```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        w
    end
end;
```

```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        writeln(P^.Dato);
    end;
end;
```

```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        wr
    end;
end;
```

```
Procedure ImpListaInv(P:Puntl
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        wr
    end;
end;
```

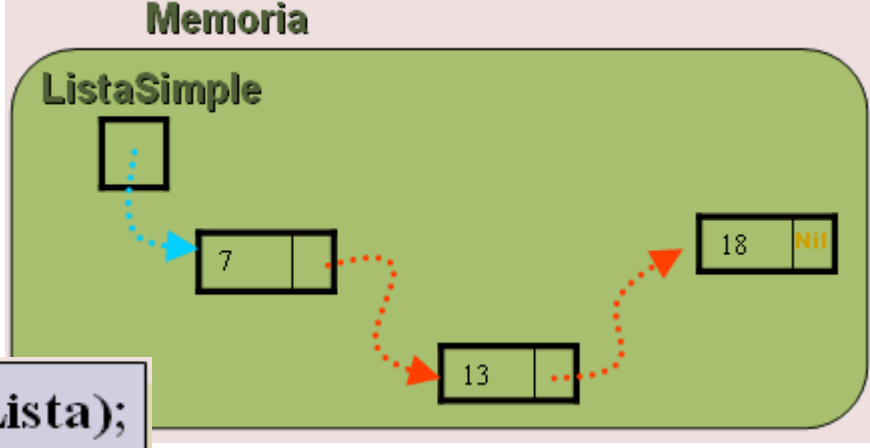
```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        writeln(P^.Dato);
    end;
end;
```



18

```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        wr
    end;
end;
```

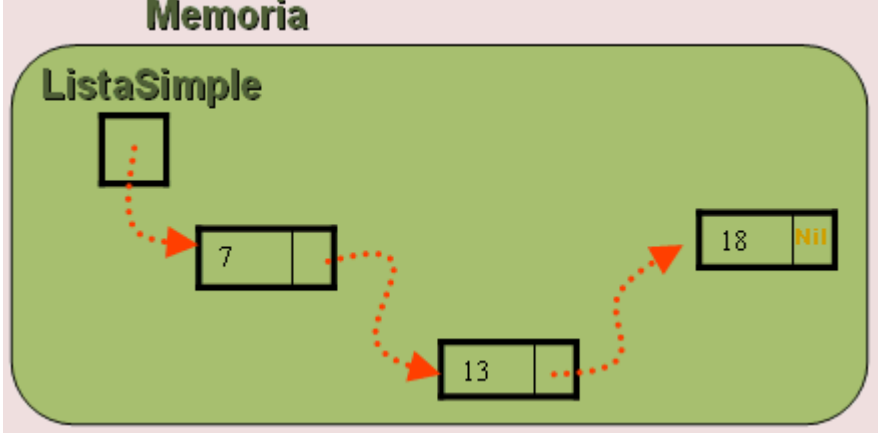
```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        writeln(P^.Dato);
    end;
end;
```



18

13

```
Procedure ImpListaInv(P:PuntLista);
begin
    if (P <> nil) then begin
        ImpListaInv(P^.Sig);
        writeln(P^.Dato);
    end;
end;
```



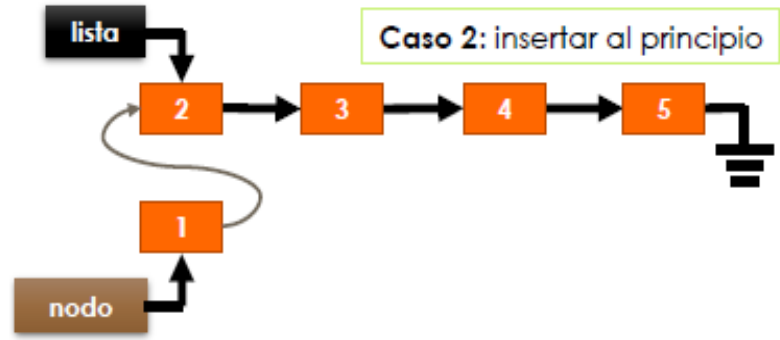
18

13

7

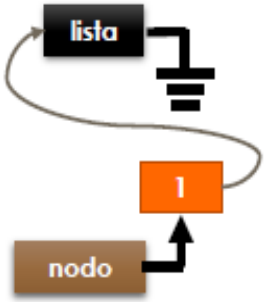
Procedimiento recursivo para insertar ordenado en una lista

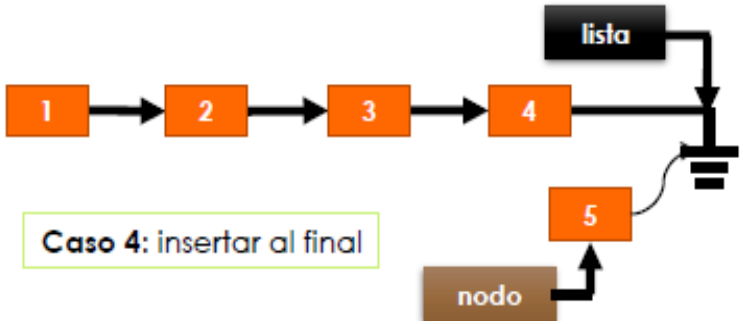
```
Procedure insertarordnodolista(var lista:punterolista;nodo:punterolista);
Begin
//Caso 1 y 4                Caso 2 y 3
if (lista=NIL)      or      (lista^.dato>=nodo^.dato) then begin
                nodo^.ste:=lista;
                lista:=nodo;
end
Else //la llamada recursiva reemplaza la búsqueda de la posición
                insertarordnodolista(lista^.ste,nodo);
End;
```



Caso 2: insertar al principio

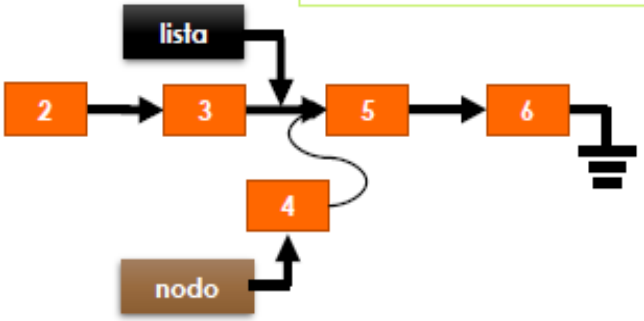
Caso 1: lista vacia





Caso 4: insertar al final

Caso 3: insertar al medio



Observación: a diferencia de los procedimientos recursivos, las funciones recursivas siempre deben retornar un resultado.

```
Función Factorial(N:integer):integer;
begin
   if N = 0
      then
        Factorial:= 1
      else
        Factorial:= Factorial (N-1) * N;
end;
```

```
Procedure ImprimirListaInv(lista:Puntlista);
Begin
   if lista<>nil
     then begin
            ImprimirListaInverso(lista^.sig);
            write(lista^.dato);
     end;
end;
```

Cuando usar ó no recursión

Usar Recursión cuando

- ✓ La estructura de la función/procedimiento es recurrente y el algoritmo queda más sencillo (Ej: ImprimirListaInv())
- ✓ La estructura de datos es recursiva

NO Usar Recursión cuando

- X Hay dificultad en la comprensión del algoritmo
- X Produce excesivas demandas de memoria o tiempo de ejecución

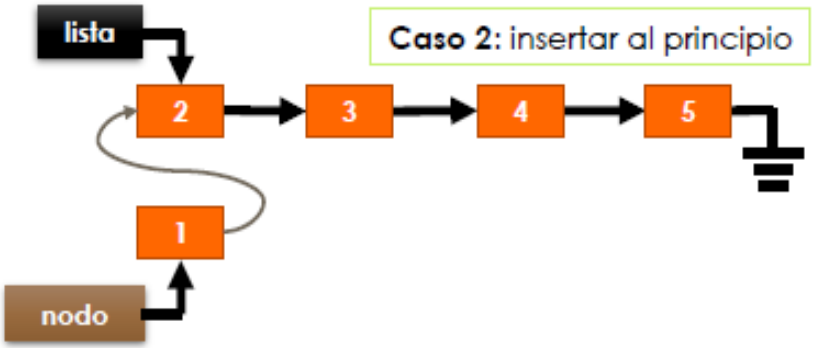
```
Procedure ImprimirArr(Arr:Arreglo;n:integer);
Begin
if n<=MAX then begin
          write(Arr[n]);
          ImprimirArreglo(Arr,n+1);
          end;
End;
```

Implica N llamados recursivos

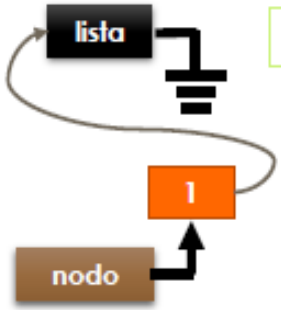
```
Procedure ImprimirArr (Arr:Arreglo);
var n:integer;
Begin
          for n:=1 to MAX do write(Arr[n]);
End;
```

Ejemplos de recursividad

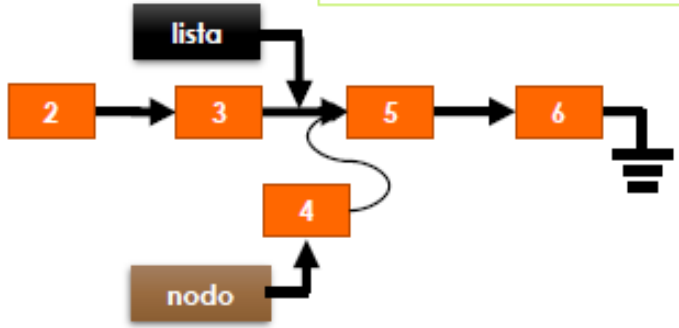
```
Procedure insertarordnodolista(var lista:punterolista;nodo:punterolista);
Begin
//Caso 1 y 4                    Caso 2 y 3
if (lista=NIL)        or        (lista^.dato>=nodo^.dato) then begin
                    nodo^.ste:=lista;
                    lista:=nodo;
end
Else //la llamada recursiva reemplaza la búsqueda de la posición
                    insertarordnodolista(lista^.ste,nodo);
End;
```



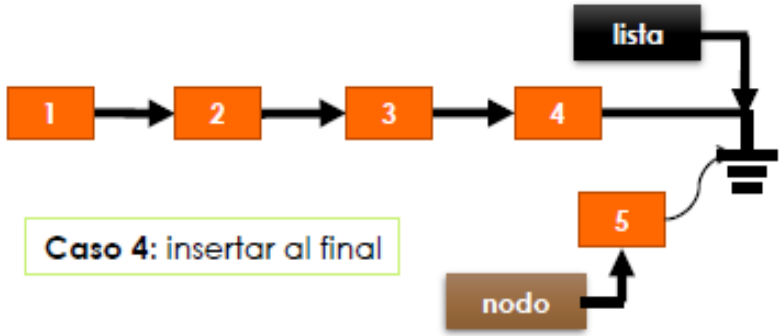
Caso 2: insertar al principio



Caso 1: lista vacia



Caso 3: insertar al medio



Caso 4: insertar al final