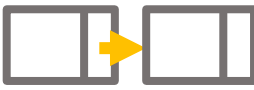


Conceptos de Algoritmos Datos y Programas

CADP – TIPOS DE DATOS - LISTA



Creación de una lista.

Agregar nodos al comienzo de la lista.

Recorrido de una lista.

Agregar nodos al final de la lista.

Buscar un elemento en una lista

Eliminar un elemento de una lista

Insertar un elemento en una lista ordenada



CADP – TEMAS



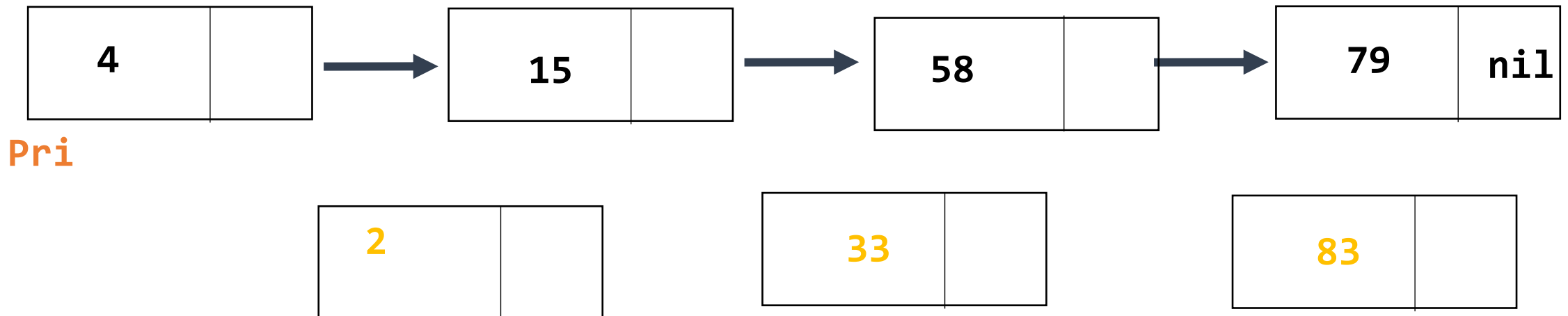
● Operación de INSERTAR un ELEMENTO



Implica agregar un nuevo nodo a una lista ordenada por algún criterio de manera que la lista siga quedando ordenada.

Existen 4 casos:

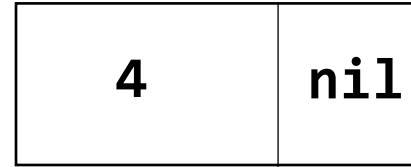
- que la lista esté vacía.
- que elemento vaya al comienzo de la lista (es menor al 1er nodo de la lista)
- que elemento vaya al “medio” de la lista (es menor al último nodo de la lista)
- que elemento vaya al final de la lista (es mayor al último nodo de la lista)





CASO 1: lista vacía

`Pri = nil`



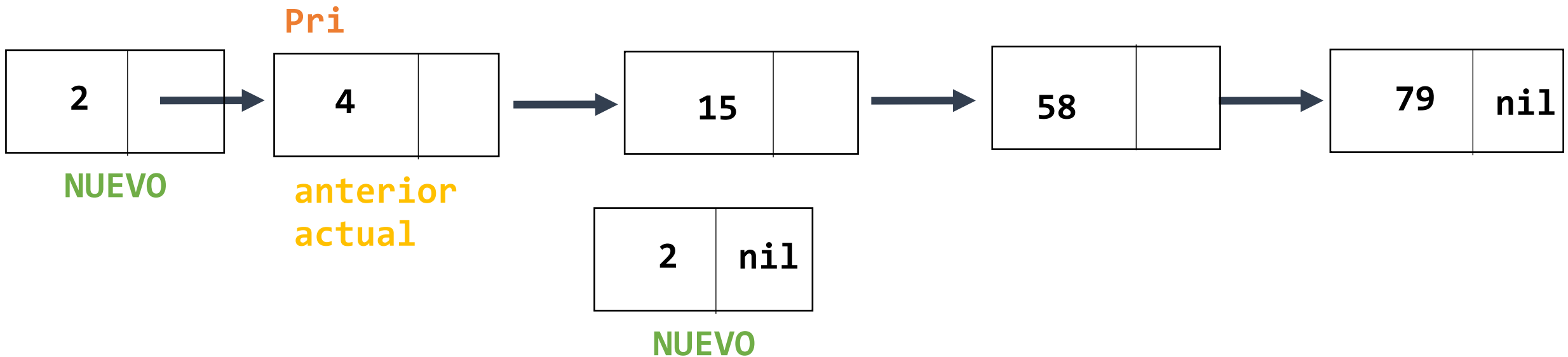
NUEVO

Generar un nuevo nodo (NUEVO).

Asignar a la dirección del puntero inicial (PI) la del nuevo nodo (NUEVO)



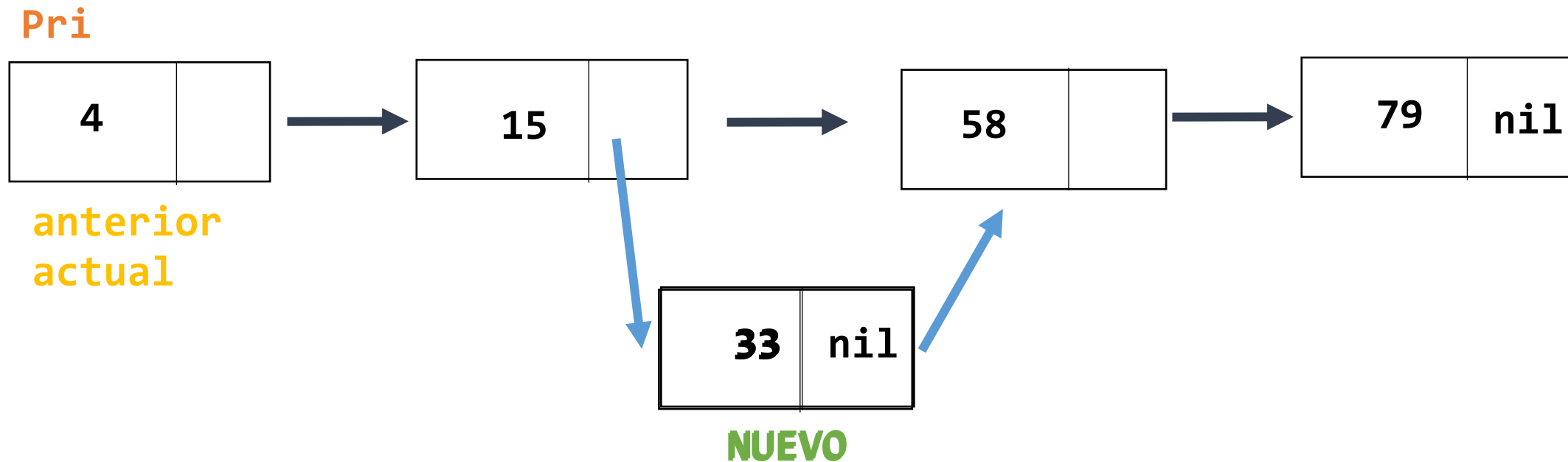
CASO 2: lista no vacía, va al principio



Generar un nuevo nodo (nuevo). Preparar punteros para el recorrido.
Asignar a la dirección del puntero siguiente del nuevo la dirección del nodo inicial (PI).
Actualizar con la dirección del nuevo nodo la dirección del puntero inicial (PI)
OBSERVAR QUE actual HABIA QUEDADO = pri

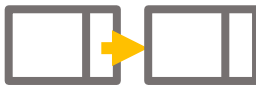


CASO 3: lista no vacía, va en el “medio”



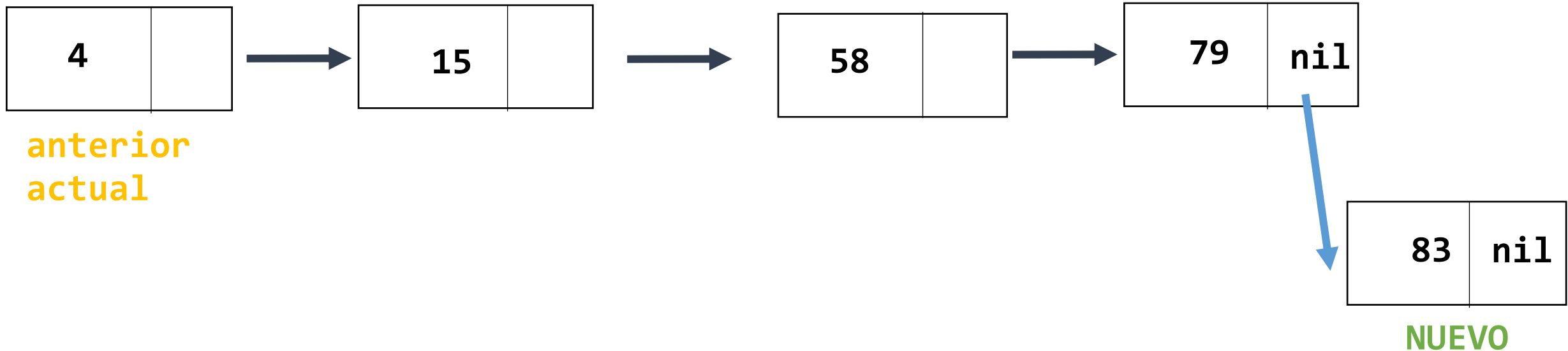
Generar un nuevo nodo (nuevo). Preparo los punteros para el recorrido
Recorro hasta encontrar la posición
Reasigno punteros, el siguiente de anterior es NUEVO y el siguiente de NUEVO es actual.

OBSERVAR QUE actual HABIA QUEDADO <> nil



CASO 4: lista no vacía, va al final

Pri

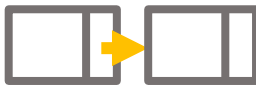


Generar un nuevo nodo (nuevo). Preparo los punteros para el recorrido
Recorro hasta encontrar la posición
Reasigno punteros, el siguiente de anterior es NUEVO y el siguiente de NUEVO es nil.

OBSERVAR QUE actual HABIA QUEDADO = nil

CADP – TIPOS DE DATOS - LISTA

INSERTAR



Generar un nuevo nodo (NUEVO).

Si la lista está vacía

Actualizo la dirección del nodo inicial (pri)

Caso 1 pri=nil

Sino

Preparo los punteros para el recorrido (anterior,actual)

Recorro hasta encontrar la posición.

Si va al principio

Asigno como siguiente del nodo nuevo al nodo inicial

Actualizo la dirección del nodo inicial (pri)

Caso 2 actual=pri

Si va en el medio

La dirección del siguiente del puntero anterior es la dirección del nodo nuevo

La dirección del siguiente del nodo nuevo es la dirección del actual

Caso 3 actual <> nil

sino

La dirección del siguiente del puntero anterior es la dirección del nodo nuevo

Caso 4 actual <> nil

Clase 10-3 La dirección del siguiente del nodo nuevo es la dirección nil



```
Program uno;
```

```
Type listaE= ^datosEnteros;
```

```
    datosEnteros= record
                    elem:integer;
                    sig:listaE;
                end;
```

```
Var
```

```
    pri: listaE;
    num:integer;
```

```
Begin
```

```
    crear (pri);
    cargar (pri); //se dispone
    read (num);
    insertar(pri,num);
```

```
End.
```



```
procedure insertar (Var pI: listaE; valor:integer);
```

```
Var
```

```
    actual,anterior,nuevo:listaE;
```

```
Begin
```

```
    new (nuevo); nuevo^.elem:= valor; nuevo^.sig:=nil;
```

```
    if (pI = nil) then  
        pI:= nuevo
```



Caso 1 pri=nil

```
    else begin  
        actual:= pI; ant:=pI;
```

```
        while (actual <> nil) and (actual^.elem < nuevo^.elem) do  
            begin  
                anterior:=actual;  
                actual:= actual^.sig;  
            end;
```



BUSCO LA
POSICION



```
end;
```



```
if (actual = pI) then
begin
  nuevo^.sig := pI;
  pI := nuevo;
end
```



Caso 2
pri=actual

```
else if (actual <> nil) then
begin
  anterior^.sigg := nuevo;
  nuevo^.sigg := actual;
end;
```



Casos 3 y 4
actual <> nil

```
End;
else
begin
  anterior^.sig := nuevo;
  nuevo^.sig := actual;
end;
End;
```



Caso 4
actual = nil



En el caso 4
cuánto vale
actual?



```
procedure insertar (Var pI: listaE; valor:integer);
Var
  actual,anterior,nuevo:listaE;
Begin
  new (nuevo); nuevo^.elem:= valor; nuevo^.sig:=nil;
  if (pI = nil) then      pI:= nuevo

  else begin
    actual:= pI; ant:=pI;
    while (actual <> nil) and (actual^.elem < nuevo^.elem) do
      begin
        anterior:=actual;
        actual:= actual^.sig;
      end;
    end;
    if (actual = pI) then
      begin
        nuevo^.sig:= pI;  pI:= nuevo;
      end
    else
      begin
        anterior^.sig:= nuevo;  nuevo^.sig:= actual;
      end;
    end;
  End;
```