

LISTAS SIMPLES

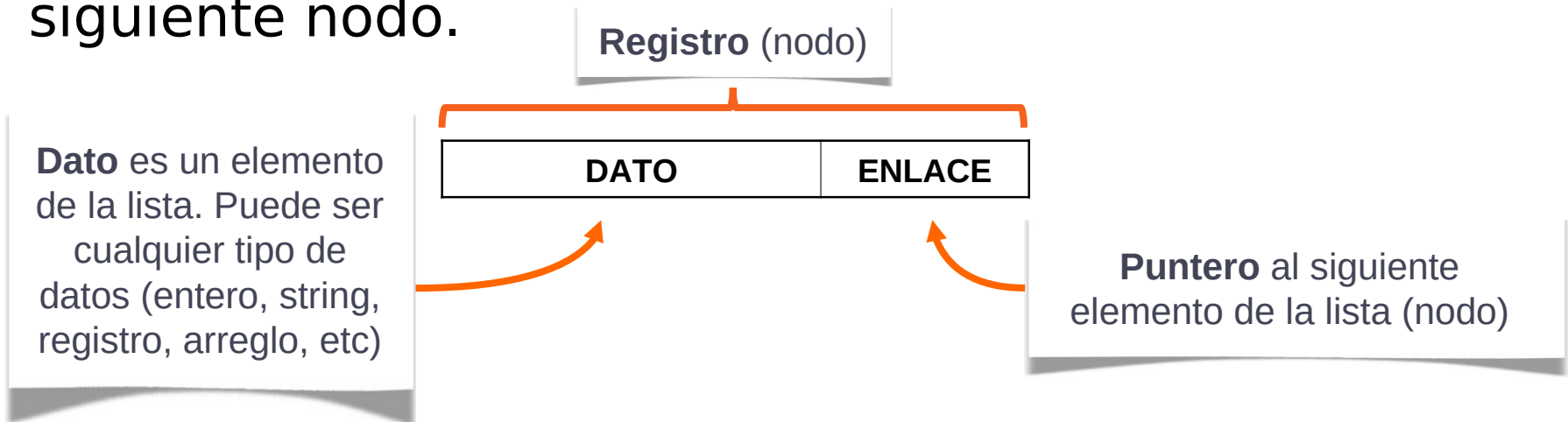
Explicación Práctica

Programación I - 2024

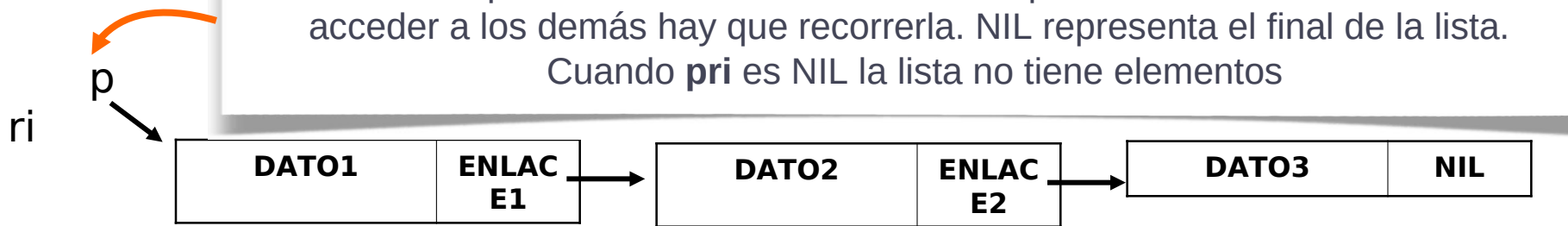
Facultad de Informática y Facultad de Ingeniería - UNLP

Listas - Definición

Una **lista** es una colección de **NODOS**, donde cada nodo está representado por un registro que contiene información de un elemento de la lista y un enlace al siguiente nodo.



Estructura de la lista:

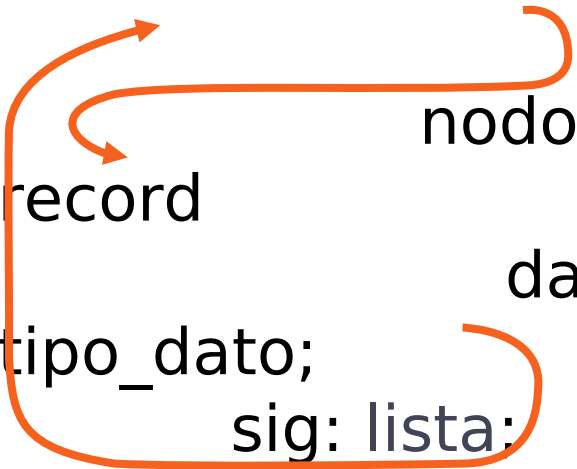


Listas - Características

- Una lista enlazada es **estructura de datos dinámica**
- Crece o decrece dinámicamente, liberando o reservando memoria para agregar y eliminar sus elementos.
- Respecto de la memoria que ocupa, es una estructura de datos más eficiente que un arreglo cuando no es posible saber con anticipación la cantidad de elementos que podrá almacenar.
- Respecto del acceso a sus elementos, como su acceso es secuencial es menos eficiente que un arreglo cuando se conoce la posición del elemento.

Definición y uso en Pascal

```
TYPE          lista  =  
  ^nodo;  
  nodo =  
  record  
    dato: tipo_dato;  
    sig: lista;  
  end;
```



Definición típica de una lista en Pascal

Notar la naturaleza recursiva de la definición de la lista

```
VAR  
  pri: lista;
```



Variable que almacena la lista (apunta al primer elemento)

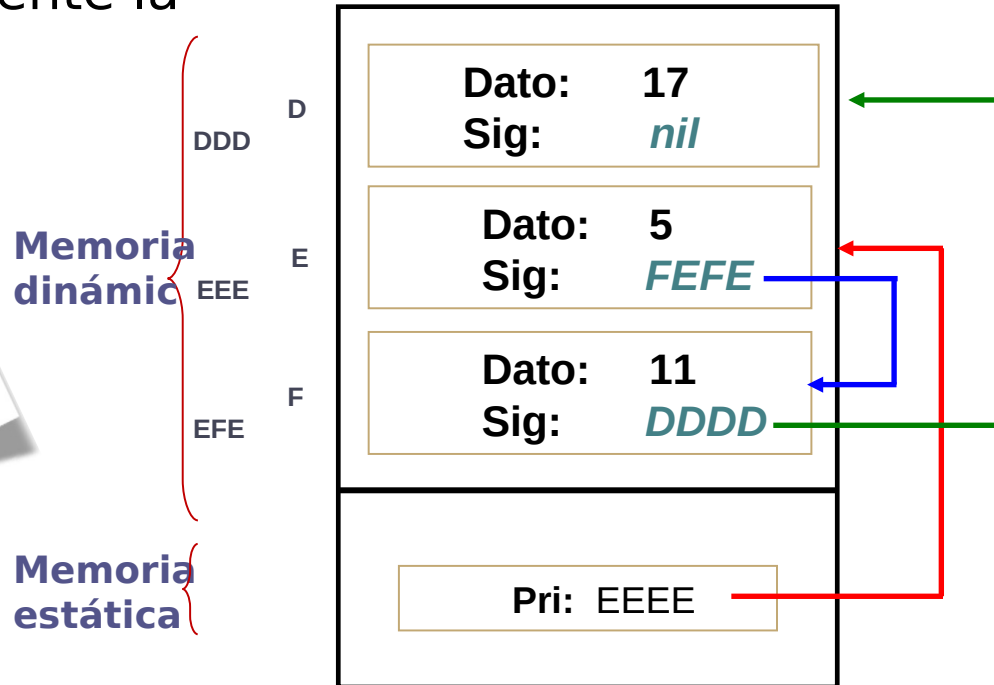
Definición y uso en Pascal

Ejemplo: Supongamos una lista con los elementos: 5, 11 y 17
¿Cómo se ve lógicamente?



¿Cómo se ve físicamente la lista?

Recordar: el campo **sig** del registro es un puntero en el que tenemos que reservar espacio para almacenar el nodo en memoria dinámica



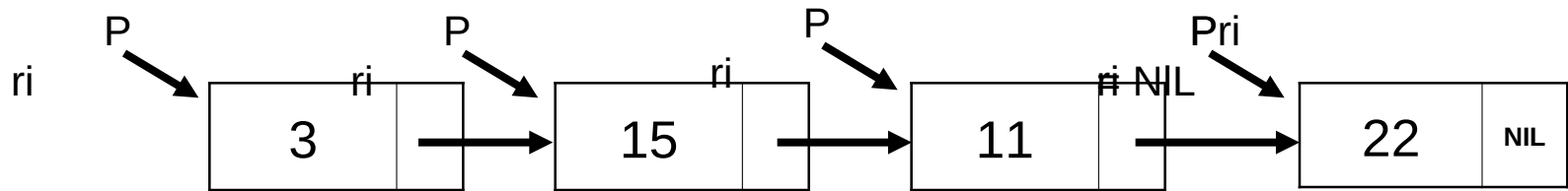
Ejercicio 1

Leer una secuencia de 20 números y crear una lista agregando **siempre al principio**. Una vez generada dicha lista debe imprimirla en pantalla.

Ejercicio 1 - Análisis del problema

Supongamos que llegan los números: 22
11 15 3

¿Cómo se va formando la lista?



Los elementos quedan en el orden inverso al que se los lee.

Ejercicio 1 - Solución

Program uno;

{TIPOS DEFINIDOS POR EL USUARIO}

const

cant = 20;

type

lista = ^nodo;

nodo = record

dato: integer;

sig : lista;

end;

{PROCEDIMIENTOS Y FUNCIONES}

procedure agregar(var pri:lista; num:integer);

var

nue:lista;

begin

new (nue); *{ reserva memoria para el nodo }*

nue^.dato := num; *{ guarda dato en el nodo }*

nue^.sig := pri; *{ apunta al "viejo" inicio }*

pri := nue; *{ la lista apunta al nuevo nodo }*

end;

procedure **imprimir** (l: lista);

begin

writeln('----- LISTADO -----');

while (l <> nil) do begin

writeln ('Numero: ', l^.dato);

l := l^.sig;

end;

end;

{VARIABLES DEL PROGRAMA PRINCIPAL}

var

pri : lista;

num, i : integer;

begin **{PROGRAMA PRINCIPAL}**

pri := nil;

for i:= 1 to cant do begin

read (num);

agregar(pri, num);

end;

imprimir (pri);

end.

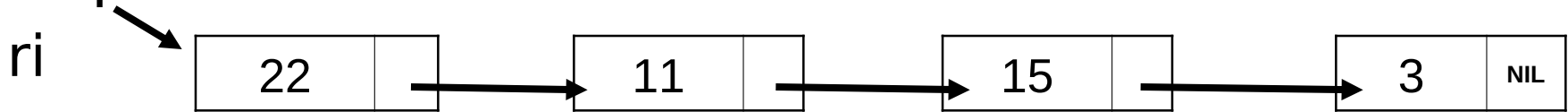
Ejercicio 2

Realizar un **modulo** para el programa del ejercicio anterior que reciba la lista, un valor entero y le agregue un elemento **al final** de la misma.

Ejercicio 2 - Análisis del problema

Los números agregados en el ej. 1 son: 22,
11, 15, 3

La lista creada tiene el puntero Pri al elemento 22:



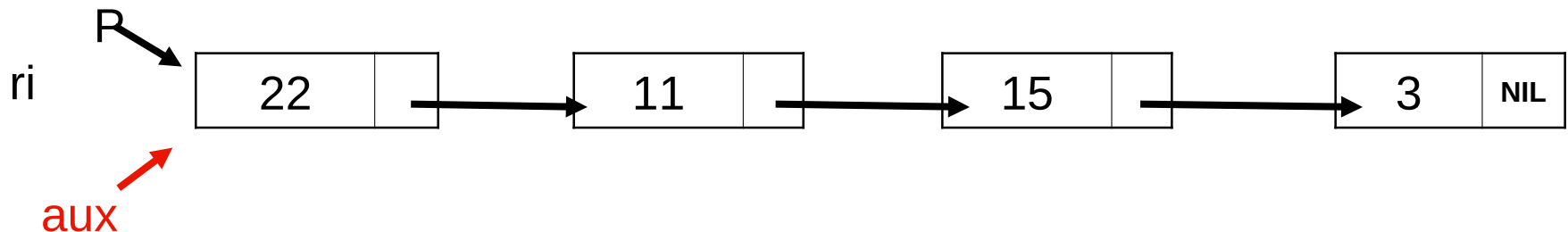
¿Cómo obtengo el último nodo de la lista para enlazar uno nuevo?

Se recorre la lista hasta el final y se alcanza el último nodo para que en vez de apuntar a NIL apunte al nuevo.

Ejercicio 2 - Análisis del problema

```
{ recorrido de la lista hasta el final }  
aux := Pri; { preserve el principio de la lista }  
while (aux <> nil) do begin { mientras no llegue al final de la lista }  
    aux := aux^.sig; { avanzar al próximo elemento }  
end;
```

Cuando sale del **while** llegamos al final de la lista, pero que pasa con el último nodo que necesitamos para enlazar con el nuevo?



aux apunta a NIL y no último nodo, **aux** debería tener el valor anterior a NIL. Necesito una variable adicional con el nodo anterior de **aux**

Ejercicio 2 - Análisis del problema

{ recorrido de la lista hasta el final }

aux := Pri;

{ preserva el principio de la lista }

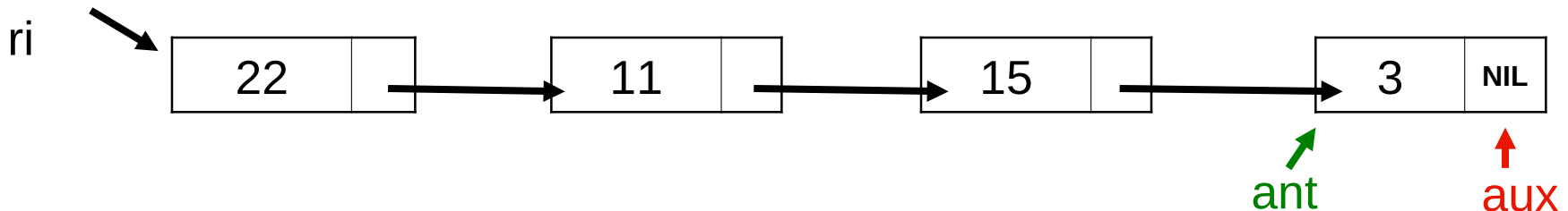
while (aux <> nil) do begin *{ mientras no llegue al final de la lista }*

 ant := aux; *{ guarda nodo actual antes de pasar al siguiente }*

 aux := aux^.sig; *{ avanzar al próximo elemento }*

end;

Hay que llevar los dos punteros hasta llegar al final de la lista:



Al finalizar el **while** utilizamos **ant** para apuntar **ant^.sig** al nuevo nodo para que se convierta en el ultimo de la lista.

Ejercicio 2 - Solución

```
procedure agregar_final (var l: lista; n: integer);
var
  aux, nue, ant: lista;
begin
  new (nue);           { reserva memoria para el nodo de la lista}
  nue^.dato := n;      { guarda el dato en el nodo de la lista}
  nue^.sig := nil;     { como nue será el ultimo nodo, no hay siguiente}
  if (l = nil) then    {La lista está vacía?}
    l := nue           { el nodo es el primero de la lista}
  else begin           { ya hay elementos en la lista}
    aux := l;
    while (aux <> nil) do begin { mientras no llegue al final de la lista}
      ant := aux;          { guarda el nodo actual}
      aux := aux^.sig;     { avanzar al próximo elemento}
    end;
    ant^.sig := nue; { enlaza el último nodo con el nuevo}
  end;
end;
```