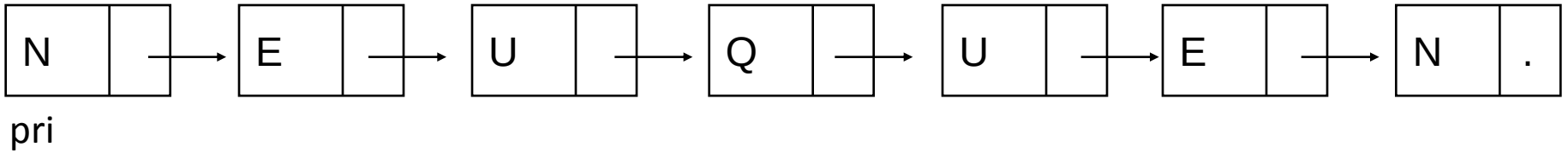
LISTAS DOBLES y multiples

Temas

- ✓ Listas doblemente enlazadas
- ✓ Características y Operaciones
- ✓ Ejemplos
- ✓ Lista múltiples enlaces

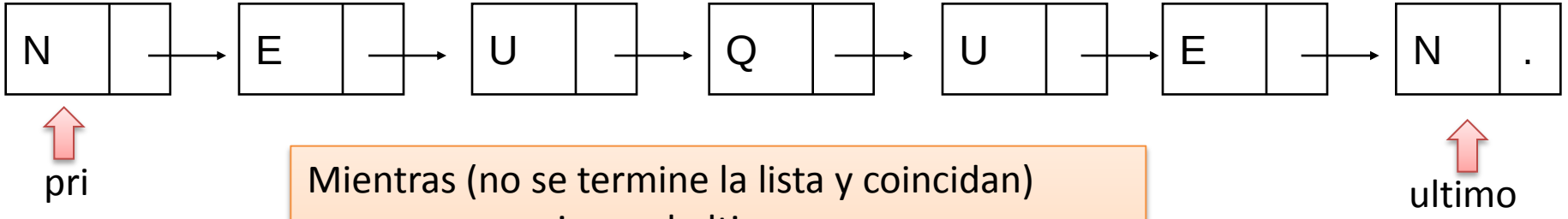
LISTAS DOBLEMENTE ENLAZADAS y MÚLTIPLE ENLACES

Suponga el siguiente problema: en una lista se tiene almacenada una palabra (un carácter en cada nodo) y se quiere realizar un módulo que reciba la lista y devuelva true si la palabra almacenada es capicúa, o false en caso contrario



¿Cómo lo resolvemos?

LISTAS DOBLEMENTE ENLAZADAS y MÚLTIPLE ENLACES



Mientras (no se termine la lista y coincidan)
comparo pri con el ultimo
si (coinciden) entonces
avanzo el pri
avanzo ultimo hasta un lugar anterior.
Si se terminó la lista devuelvo true
Sino false

¿Qué nos convendría tener?

LISTAS DOBLEMENTE ENLAZADAS y MULTIPLE ENLACES

Son listas que tienen más de un enlace.
Cada enlace recorre la lista en un orden:

- Doblemente enlazadas: el criterio de ambos enlaces es el mismo.
- Múltiple enlaces: el criterio de orden es distinto para cada enlace y por lo tanto la lista puede recorrerse teniendo en cuenta cada uno de los criterios.

VENTAJAS

✓Pueden recorrerse de dos formas, ya sea para efectuar una operación con cada elemento o para insertar/actualizar y borrar.

DESVENTAJA

✓Ocupan más memoria por nodo que una lista simple.

LISTAS DOBLEMENTE ENLAZADAS y MULTIPLE ENLACES

Una representación posible considerando que los enlaces permitirán recorrer la lista en dos sentidos de manera tal que un sentido da la inversa del otro.

```
type
        puntero = ^nodo;
        nodo = record
                    ant: puntero;
                    info: .....;
                    sig: puntero;
                end;
       listaDoble = record
          pri: puntero;
          ult: puntero;
      end;
```

¿Cómo implementamos el problema de la palabra?

LISTAS DOBLEMENTE ENLAZADAS y MULTIPLE ENLACES

```
var
  lista: listaDoble;
Function capicua (lis:listaDoble): boolean;
…

Begin
  lista.pri:= nil;
  lista.ult:=nil;
  cargarLista(lista);
  if (capicua(lista) ) then   write (“La palabra es capicua”)
  else write (“La palabra no es capicua”);
End.
```

LISTAS DOBLEMENTE ENLAZADAS y MULTIPLE ENLACES

```
Function capicua (lis:listaDoble): boolean;
Var ok:boolean; ini, fin: puntero;

Begin
   ok:= true;  ini:= lis.pri;  fin:= lis.ult;
   while (ok) and (ini <> fin) and (ini^.ant <> fin) do
       begin
         if (ini^.info <> fin^.info) then ok:= false
         else begin
                   ini:= ini^.sig;  fin:= fin^.ant;
              end;
         end;
  capicua:= ok;
 end;
```

Operaciones en una lista doble?

LISTAS DOBLEMENTE ENLAZADAS y MULTIPLE ENLACES – CREAR

```
....
Var
  lis:listaDoble;

Begin
  crear(lis);
End.
```

```
Procedure crear (var l:listaDoble);
Begin
   l.pri:= nil;
   l.ult:= nil;
End;
```

Cómo agregamos adelante?

LISTAS DOBLEMENTE ENLAZADAS y MULTIPLE ENLACES - AGREGAR

```
….
Var
  lis:listaDoble;
  n:integer;

Begin
  crear(lis);
  read (n);
  agregar (lis,n);
End.
```

LISTAS DOBLEMENTE ENLAZADAS y MULTIPLE ENLACES - AGREGAR

```
Procedure agregar (var l:listaDoble, n:integer);
Begin
   crear un nuevo nodo  (aux)
   si (la lista está vacía) entonces
      asigno al puntero inicial y final la dirección de aux

   sino
      asigno como siguiente de aux al primer nodo
      asigno como anterior del primer nodo a aux
      asigno como primer puntero de la lista a aux
End;
```

LISTAS DOBLEMENTE ENLAZADAS y MULTIPLE ENLACES - AGREGAR

```
Procedure agregar (var l:listaDoble; n:integer);
Var
  aux:puntero;
Begin
  new (aux);   aux^.info := n;  aux^.ant := nil; aux^.sig := nil;

  if (l.pri = nil) then begin
                          l.pri:= aux;   l.ult:= aux;
                        end
  else begin
         aux^.sig := l.pri;  l.pri^.ant := := aux; l.pri:= aux;
        end;
End:
```

Cómo insertamos?

LISTAS DOBLEMENTE ENLAZADAS y MULTIPLE ENLACES - INSERTAR

```
....
Var
  lis:listaDoble;
  n:integer;
Begin
  crear(lis);
  read (n);
  insertar (lis,n);
End.
```

LISTAS DOBLEMENTE ENLAZADAS y MULTIPLE ENLACES - INSERTAR

```
Procedure insertar (var l:listaDoble, n:integer);
Begin
  crear un nuevo nodo  (aux)
  si (la lista está vacía) entonces
    {asigno como primer y último elemento a aux}

  sino
   { busco el lugar
    determino si el elemento debe agregarse adelante, al medio o al final
     realizo las actualizaciones correspondientes dependiendo del caso}
End;
```

```
procedure InsertaOrdenado ( l: listaDoble; x: integer);
var
    act, nuevo : puntero;
begin
   new (nuevo);       nuevo^.info := x;
   nuevo^.sig := nil;
   nuevo^.ant :=nil;

   if (l.pri = nil) then
    begin
      l.pri:= nuevo;  l.ult:= nuevo;
    end
   else begin
   act:= l.pri;

   while (act <> nil) and (x > act^.info) do
        act:= act^.sig;
   if (act = l.pri) then begin
                        nuevo^.sig:=l.pri;
                        l.pri^.ant:=nuevo;
                        l.pri:= nuevo;
                       end
    else if (act <> nil) then begin
                          nuevo^.ant:=  act^.ant;
                          act^.ant^.sig:=  nuevo;
                          nuevo^.sig:=  act;
                          act^.ant:= nuevo;
                         end
          else begin
               nuevo^.ant:=l.ult;
               l.ult^.sig:=nuevo;
               l.ult:= nuevo;
          end;
end;
```

LISTAS DE MULTIPLE ENLACES

VENTAJAS

✓Pueden recorrerse siguiendo diferentes enlaces, lo que permite vistas diferentes de la misma información. Por ejemplo TRES ordenes diferentes.

DESVENTAJA

✓Ocupan más memoria por nodo, un puntero por cada orden que se quiere representar

Analicemos el siguiente problema

LISTAS DE MULTIPLE ENLACES

Una empresa maneja la información de los empleados. De cada empleado se conoce nombre, edad, y el número de oficina en la que trabaja.

A la empresa le interesa mantener la información de sus empleados ordenados por nombre y por oficina. esta información está ordenada por número de oficina en la que trabaja el empleado, y por apellido y nombre.

¿Cómo representamos esto?
¿Tengo dos listas?
¿Cómo busco un empleado?
¿Cómo agrego un nuevo empleado?

LISTAS DE MULTIPLES ENLACES

```
type
    empleado = record
       nombre: string;  edad: integer; oficina:integer;
     end;
      puntero = ^nodo;
      nodo = record
                    nom: puntero;
                    dato: empleado;
                    ofi: puntero;
                end;
     listaDoble = record
         pri-Nombre: puntero;
         pri-Ofi: puntero;
    end;
```

LISTAS DE MULTIPLES ENLACES

Nombre: Ana Edad:25 Oficina: 6	Nombre: Pedro Edad:25 Oficina: 5	Nombre: Lucas Edad:33 Oficina: 1	Nombre: Juan Edad:25 Oficina: 8

¿Cómo se organiza esta información en una única lista?

LISTAS DE MULTIPLES ENLACES

Nombre: Ana
Edad:25
Oficina: 6

Lis.pri-Nom

Nombre: Lucas
Edad:33
Oficina: 1

Lis.pri-Ofi

Nombre: Juan
Edad:25
Oficina: 8

Nombre: Pedro
Edad:25
Oficina: 5

LISTAS DE MULTIPLES ENLACES

Si quiero recorrer **por nombre** empezaré por Ana luego el siguiente es Juan, luego Lucas y por último Pedro.

Si quiero recorrer **por oficina** empezaré por Lucas luego el siguiente Pedro y luego Ana y por último Juan.

Para **insertar** creo un solo nodo y acomodo ambos enganches.