



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 ELIMINAR un ELEMENTO



Implica recorrer la lista desde el comienzo pasando nodo a nodo hasta encontrar el elemento y en ese momento eliminarlo (dispose). El elemento puede no estar en la lista.

Si la lista está desordenada seguramente la búsqueda se realizará hasta encontrar el elemento o hasta que se termina la lista.

Si la lista está ordenada seguramente la búsqueda se realizará hasta que se termina la lista o no se encuentre un elemento mayor al buscado.

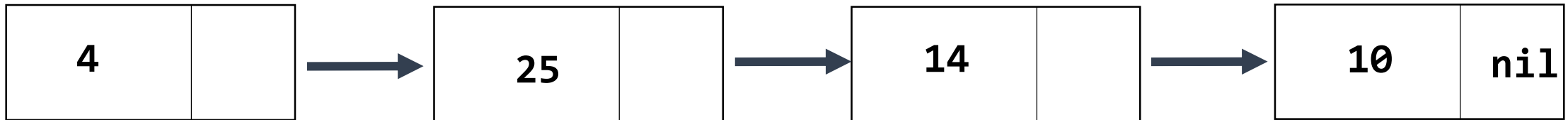
Existen 3 casos:

- que elemento a eliminar no se encuentre en la lista
- que elemento a eliminar sea el primero de la lista
- que elemento a eliminar no sea el primero en la lista



anterior

actual



Pri

num = 20

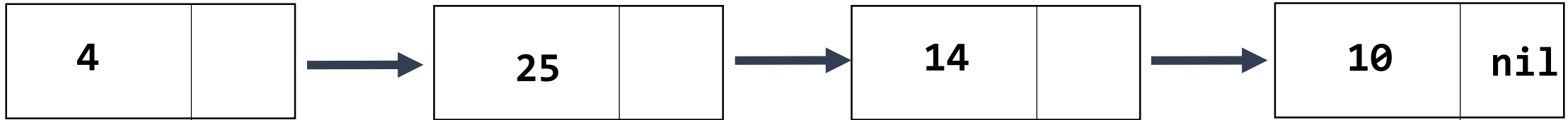
Caso 1:

Recorrí toda la lista y el elemento a eliminar no se encuentra.

OBSERVAR QUE actual QUEDÓ EN nil



anterior
actual



Pri

num = 4

Caso 2:

Empiezo a recorrer la lista.

Mientras (no encuentro el elemento a borrar) y (no se termine la lista)

el puntero anterior toma la dirección del puntero actual

avanzo el puntero actual

Como (el elemento está) y (es el primer elemento)

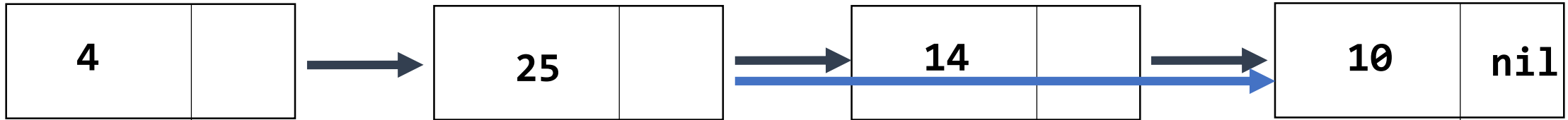
actualizo el puntero inicial de la lista

elimino la dirección del puntero actual

OBSERVAR QUE `actual` HABIA QUEDADO IGUAL A `pri`



anterior
actual



Pri

num = 14

Caso 3:

Empiezo a recorrer la lista.

Mientras (no encuentro el elemento a borrar) y (no se termine la lista)

el puntero anterior toma la dirección del puntero actual

avanzo el puntero actual

Como (el elemento está) y (NO es el primer elemento)

actualizo el siguiente del puntero anterior con el siguiente de actual

elimino la dirección del puntero actual

OBSERVAR QUE actual HABIA QUEDADO <> nil y de pri



ELIMINAR EN UN LISTA DESORDENADA

Comienzo a recorrer la lista desde el nodo inicial.

mientras ((no sea el final de la lista)y(no encuentre el elemento))

el puntero anterior toma la dirección del puntero actual
avanzo el puntero actual

si (encontré el elemento) entonces


si (es el primer nodo) entonces

actualizo el puntero inicial de la lista

elimino la dirección del puntero actual 

sino

actualizo el siguiente del puntero anterior con el siguiente de actual

elimino la dirección del puntero actual 



ELIMINAR EN UN LISTA DESORDENADA

Comienzo a recorrer la lista desde el nodo inicial.

mientras ((no sea el final de la lista)y(no encuentre el elemento))

 el puntero anterior toma la dirección del puntero actual
 avanzo el puntero actual

si (encontré el elemento) entonces

 si (es el primer nodo) entonces

 actualizo el puntero inicial de la lista

 sino

 actualizo el siguiente del puntero anterior con el siguiente de actual

 elimino la dirección del puntero actual



```
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);
    eliminar(pri,num);
```

```
End.
```



```
procedure eliminar (Var pI: listaE; valor:integer);
Var
    actual,ant:listaE;

Begin
    actual:=pI;
    while (actual <> nil) and (actual^.elem <> valor) do begin
        ant:=actual;
        actual:= actual^.sig;
    end;
    if (actual <> nil) then
        if (actual = pI) then
            pI:= pI^.sig;
        else
            ant^.sig:= actual^.sig;

            dispose (actual);

    End;
```

**Qué modifiko si el elemento
puede repetirse?**

CADP – TIPOS DE DATOS - LISTA

ELIMINAR



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

```
Var
```

```
    actual, ant: listaE;
```

```
Begin
```

```
    actual:=pI;
```

```
    while (actual <> nil) do begin
```

```
        if (actual^.elem <> valor) then begin
```

```
            ant:=actual;  actual:= actual^.sig;
```

```
        end;
```

```
        else begin
```

```
            if (actual = pI) then
```

```
                pI:= pI^.sig;
```

```
            else
```

```
                ant^.sig:= actual^.sig;
```

```
            dispose (actual);
```

```
            actual:= ant;
```

```
        end;
```

```
End;
```

Qué modifico si la lista está
ordenada y ele elemento un
única vez ?