Escuela de
Ciencia y Tecnología
ECyT_UNSAM

Programación

Listas simplemente enlazadas

David López

Eliminar nodo

Idea general

- Para suprimir un nodo, es necesario “desprenderlo” de la lista y luego ejecutar un free() sobre él.
- Esto se logra con un puntero que apunte a dicho nodo.

Estrategia

Se pueden dar 3 casos principales:

- **Caso 1)** Trivial: Si la lista está vacía no hacemos nada. Recibimos NULL y retornamos NULL.
- **Caso 2)** El nodo a eliminar es el primero.
- **Caso 3)** El nodo a eliminar está en el medio o al final (requiere búsqueda)

```c
nodo *eliminar (nodo* l, int d) {
    nodo *ret = l, *aux;

    if (l == NULL) { /* 1er caso: Si la lista es nula no hay que hacer nada */
        printf ("La lista esta vacia\n");
    }
    else { /* La no lista es nula */
        if (l->dato == d){ /* 2do caso: el nodo a borrar es el primero */
            ret = l->sig;
            free (l);
            printf ("Elemento eliminado\n");
        }
        else { /* 3er caso: el nodo a borrar no es el primero */
            while (l->sig != NULL && l->sig->dato != d)
                l = l->sig;
            if (l->sig != NULL) { /* Si l->sig es NULL es porque el dato no existe en l */
                aux = l->sig;
                l->sig = aux->sig;
                free (aux);
                printf ("Dato %d eliminado\n", d);
            }
            else { /* Si l->sig es NULL */
                printf ("El dato %d no se encuentra en la lista\n", d);
            }
        }
    }
    return ret; /* Va a ser el nuevo principio de la lista */
}
```

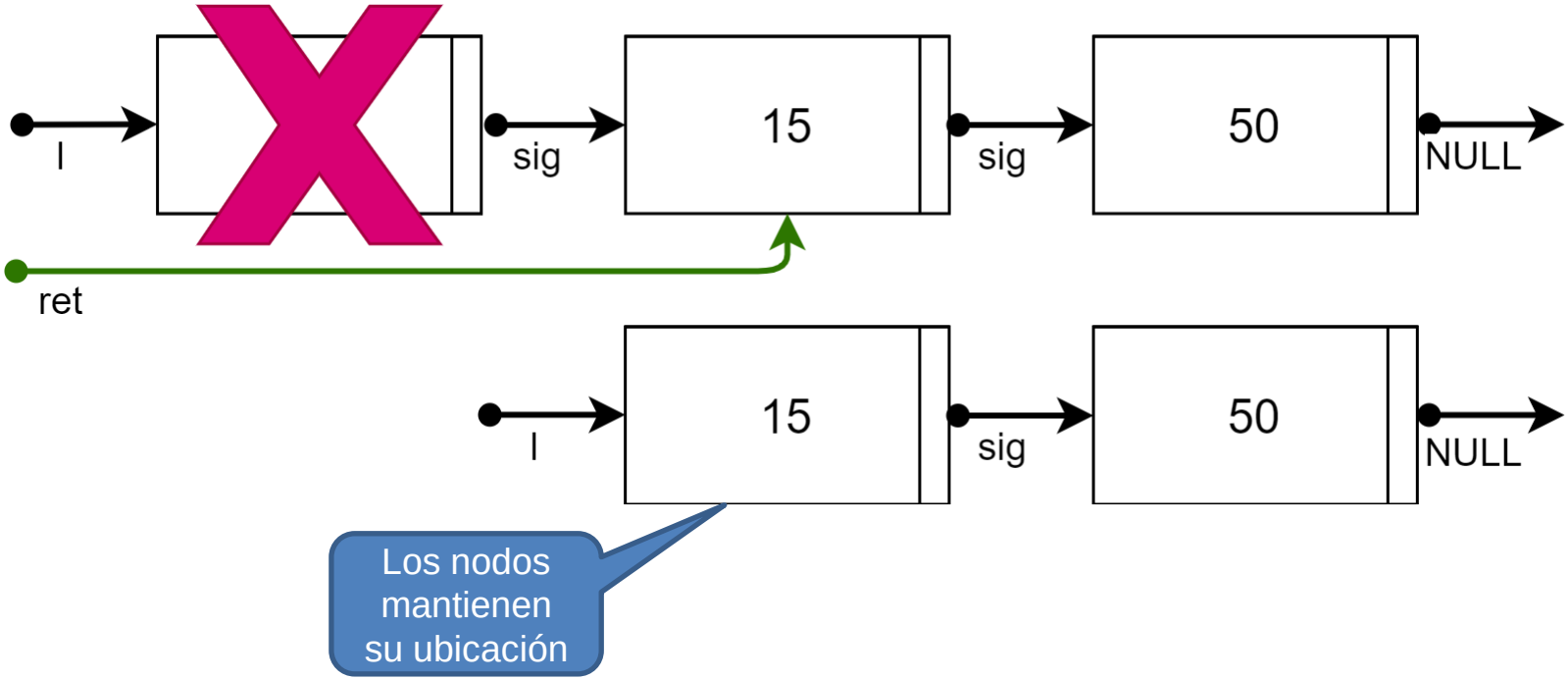
Caso 1: Lista vacía

→ Devuelvo NULL

```
1  nodo *eliminar (nodo* l, int d) {
2      nodo *ret = l, *aux;
3
4      if (l == NULL) { /* 1er caso: Si la lista es nula no hay que hacer nada */
5          printf ("La lista esta vacia\n");
6      }
7      else {⟺}
27     return ret; /* Va a ser el nuevo principio de la lista */
28 }
```

Caso 2: nodo a eliminar es el primero

Ejemplo: eliminar el nodo con valor 7

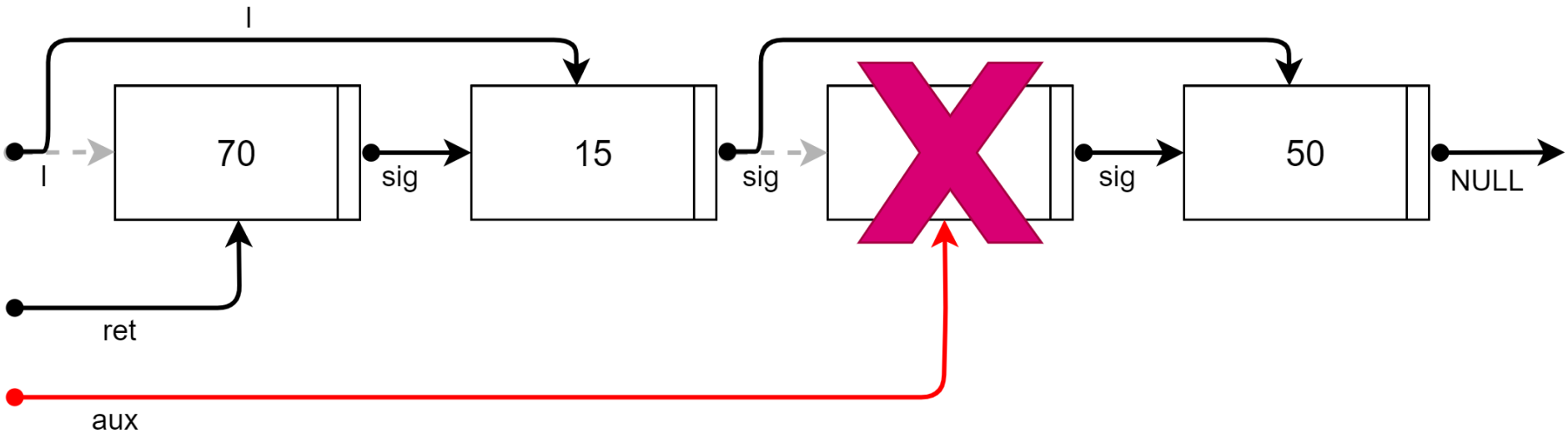


Caso 2: nodo a eliminar es el primero

```
1  nodo *eliminar (nodo* l, int d) {
2      nodo *ret = l, *aux;
3
4      if (l == NULL) { /* 1er caso: Si la lista es nula no hay que hacer nada */
5          printf ("La lista esta vacia\n");
6      }
7      else { /* La no lista es nula */
8          if (l->dato == d){ /* 2do caso: el nodo a borrar es el primero */
9              ret = l->sig;
10             free (l);
11             printf ("Elemento eliminado\n");
12         }
13         else {⟺}
26     }
27     return ret; /* Va a ser el nuevo principio de la lista */
28 }
```

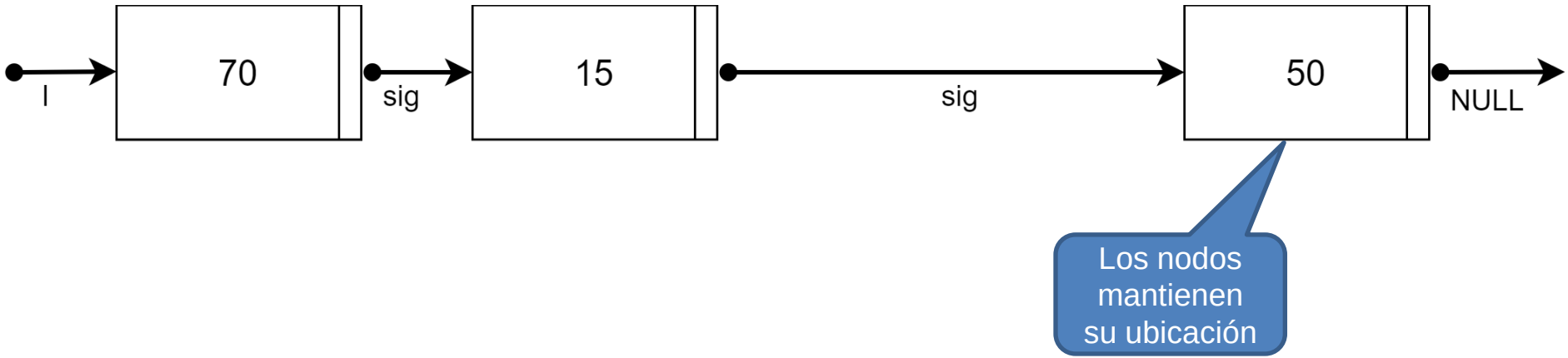
Caso 3: eliminar nodo intermedio o final

Ejemplo: eliminar el nodo con valor 20



Caso 3: eliminar nodo intermedio o final

Resultado



Caso 3: eliminar nodo intermedio o final

```
1  nodo *eliminar (nodo* l, int d) {
2      nodo *ret = l, *aux;
3
4      if (l == NULL) {⟺}
7      else { /* La no lista es nula */
8          if (l->dato == d){⟺}
13         else { /* 3er caso: el nodo a borrar no es el primero */
14             while (l->sig != NULL && l->sig->dato != d)
15                 l = l->sig;
16             if (l->sig != NULL) { /* Si l->sig es NULL es porque el dato no existe en l */
17                 aux = l->sig;
18                 l->sig = aux->sig;
19                 free (aux);
20                 printf ("Dato %d eliminado\n", d);
21             }
22             else { /* Si l->sig es NULL */
23                 printf ("El dato %d no se encuentra en la lista\n", d);
24             }
25         }
26     }
27     return ret; /* Va a ser el nuevo principio de la lista */
28 }
```