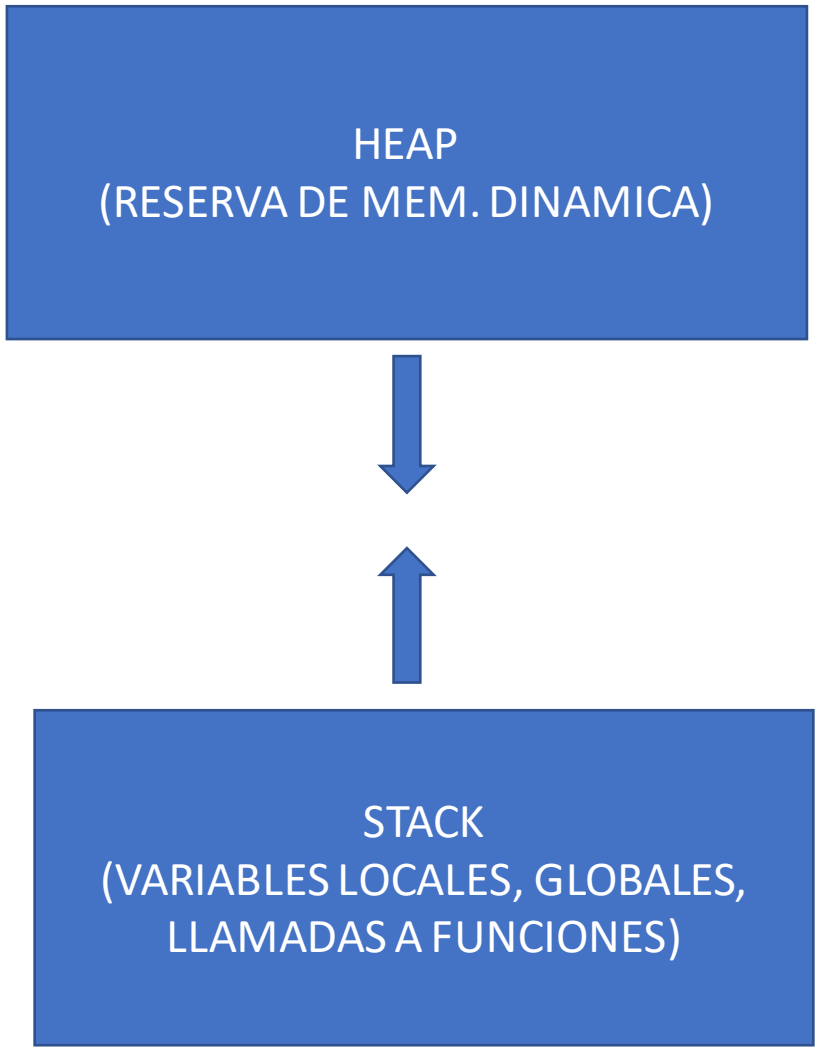
Punteros

- Array de punteros: int *p[5]



Retornar direcciones de memoria del Heap
OK

Retornar direcciones de memoria de una
Variable local de una función ERROR!!!

HEAP

Dir: 0x000FF345

```
int main(){
  int *p=0;
  char letra;
  long i;

  p = malloc(40 bytes)
}
```

Variables en Pila:

int *p = 0

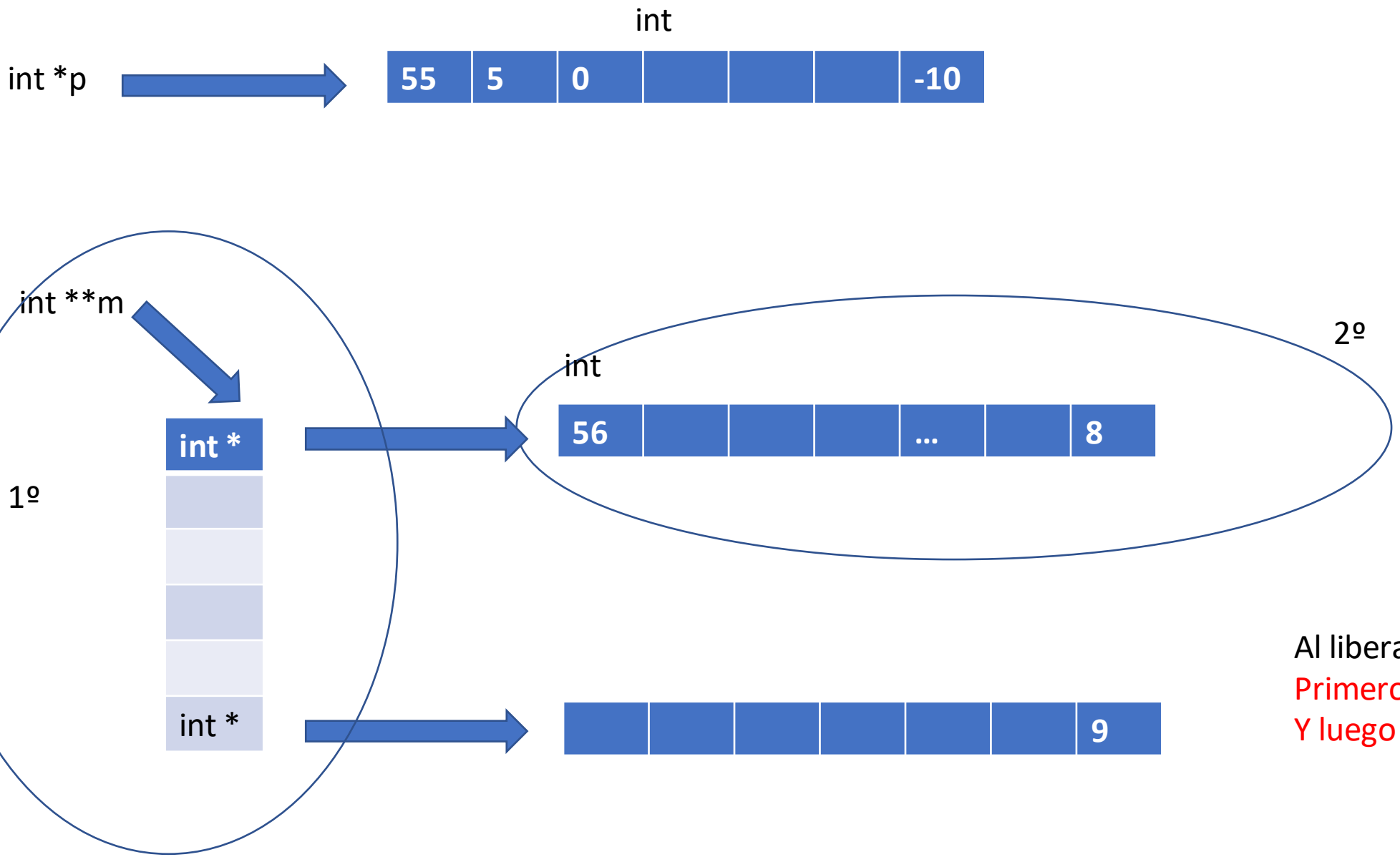
char letra;

long i

P=0x000FF345;

STACK

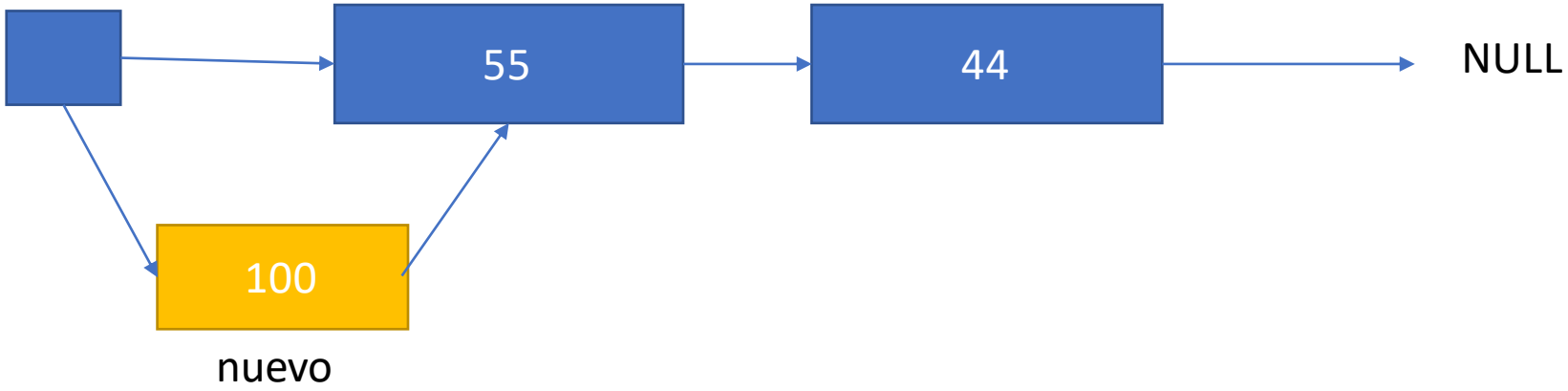
Arrays 1 dim / Tablas de 2 dim



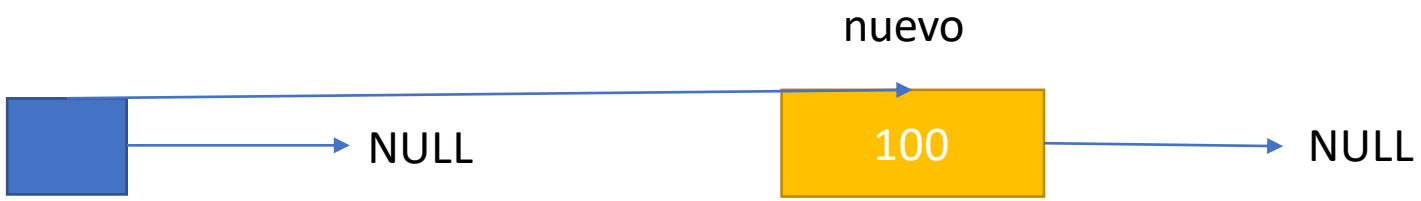
Al liberar se hace al revés
Primero: 2º
Y luego el 1º

Listas: inserción al inicio de un nuevo nodo

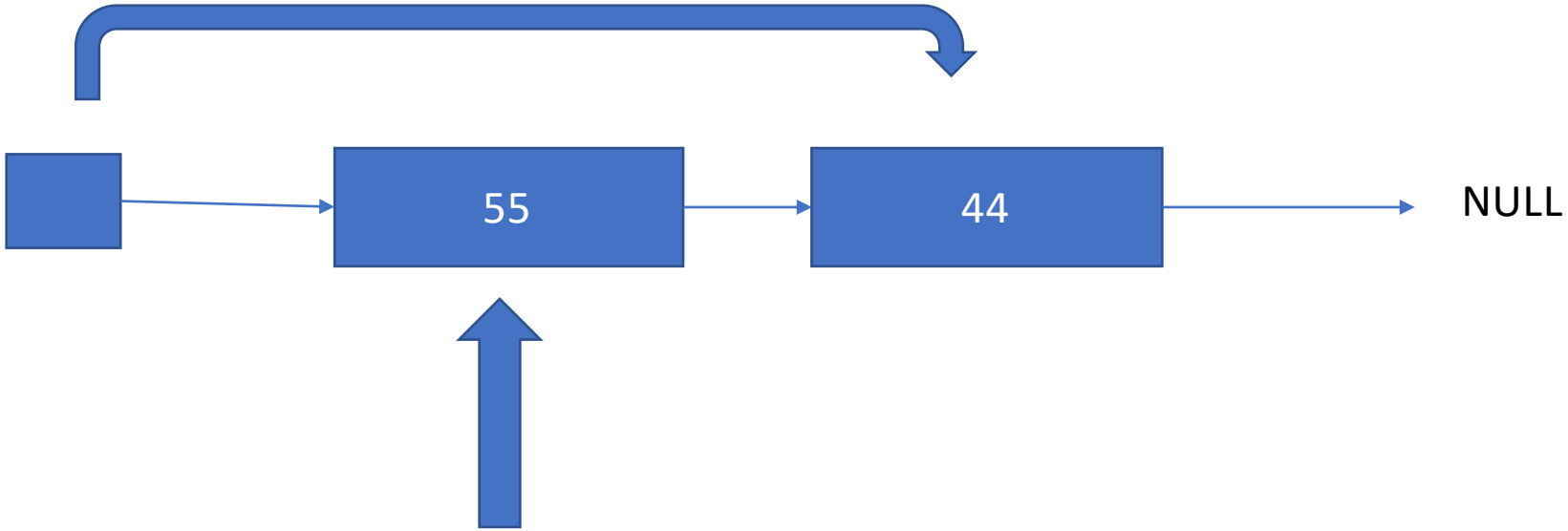
En una lista con elementos ya creados



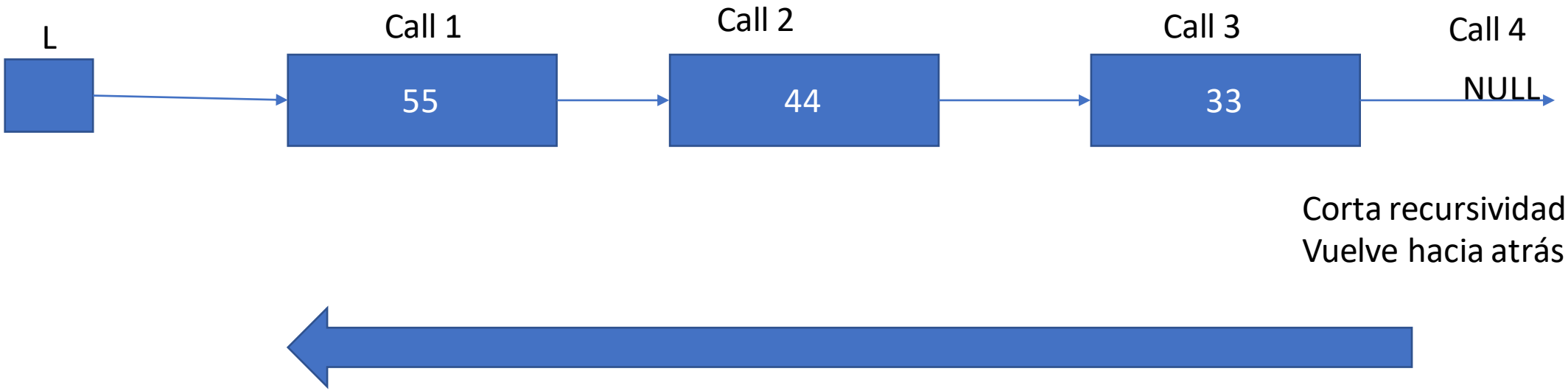
En una lista vacía



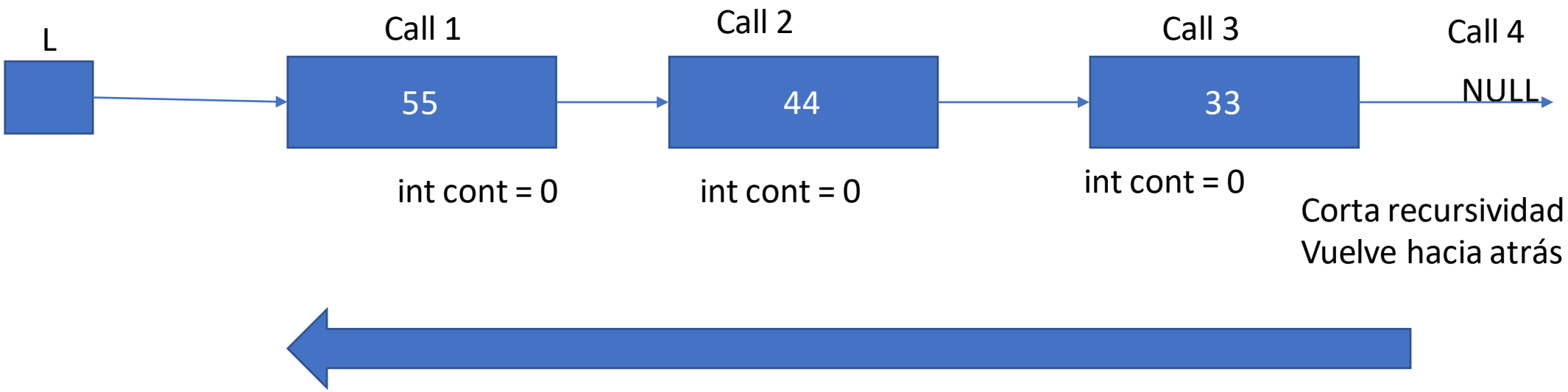
Listas: eliminar el primer nodo



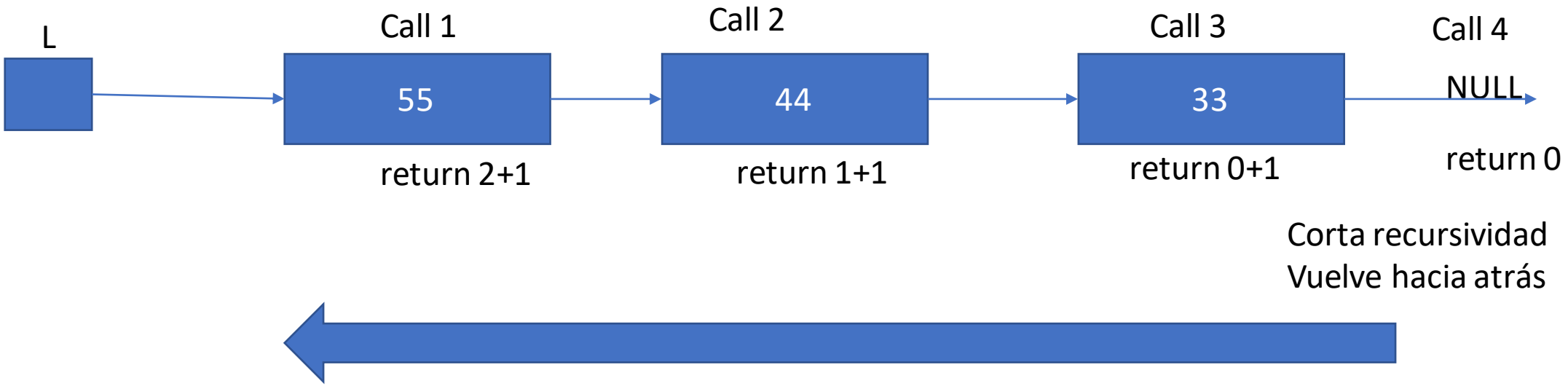
Listas: recorridos recursivos



Listas: variables locales

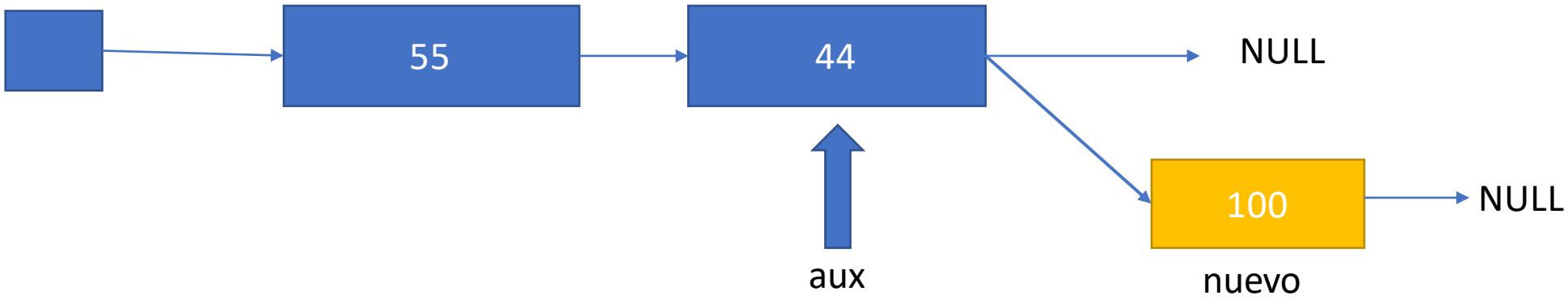


Listas: con valor de retorno

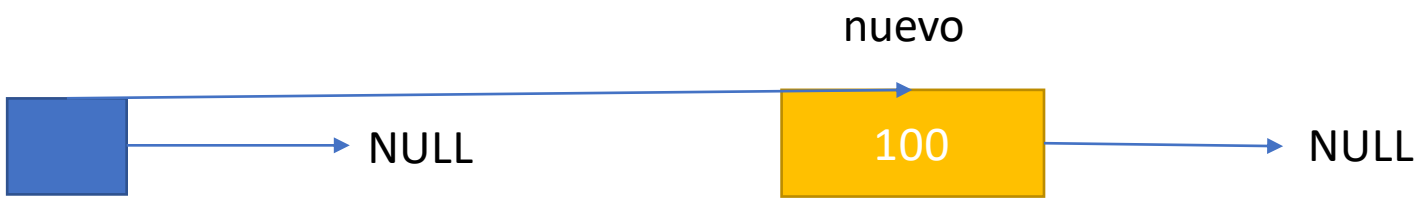


Listas: inserción de un nuevo nodo al final

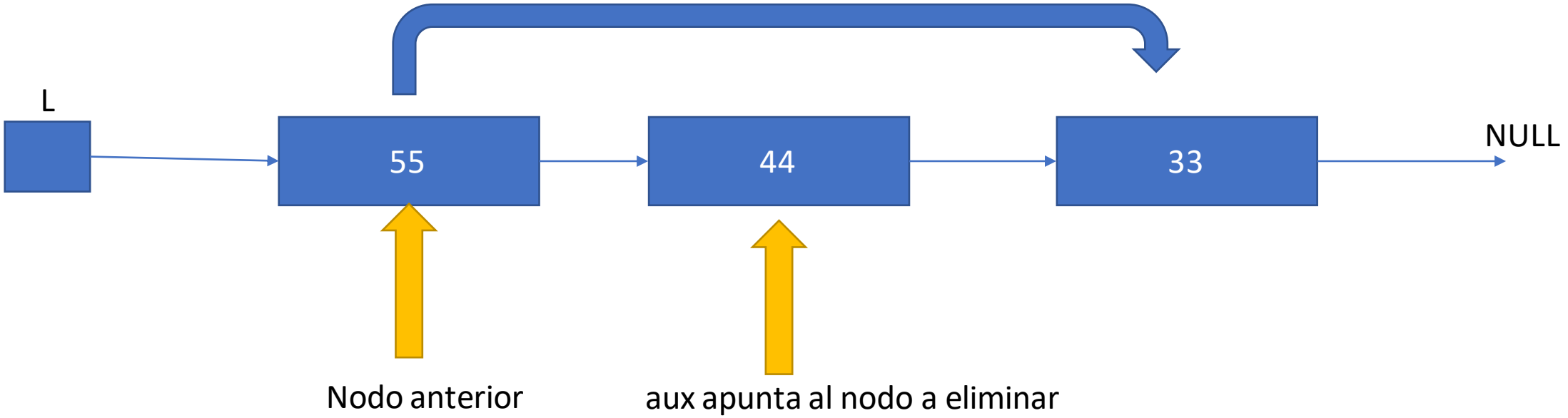
En una lista con elementos ya creados



En una lista vacía

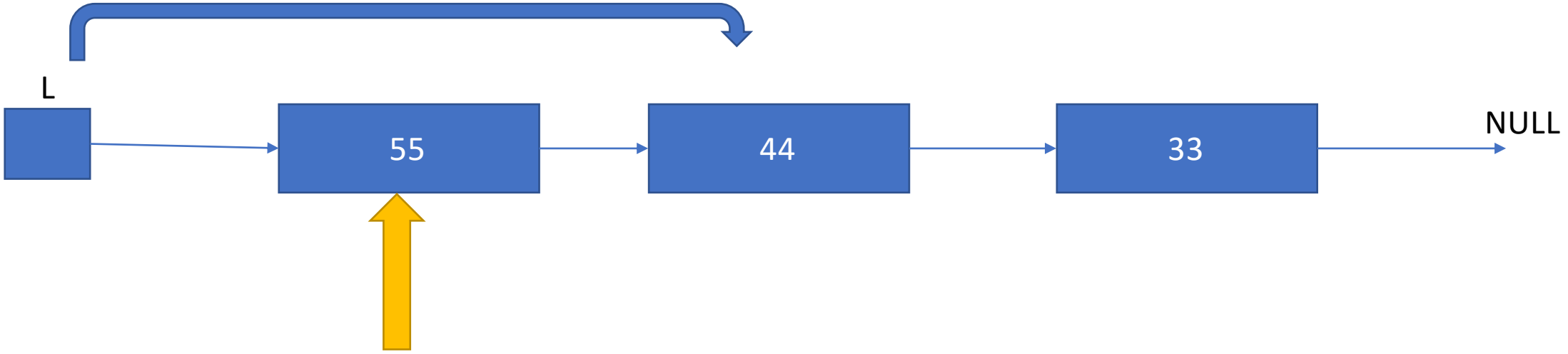


Listas: Borrar un nodo (en medio de la lista)



Borrar el 44

Listas: Borrar un nodo (al principio)

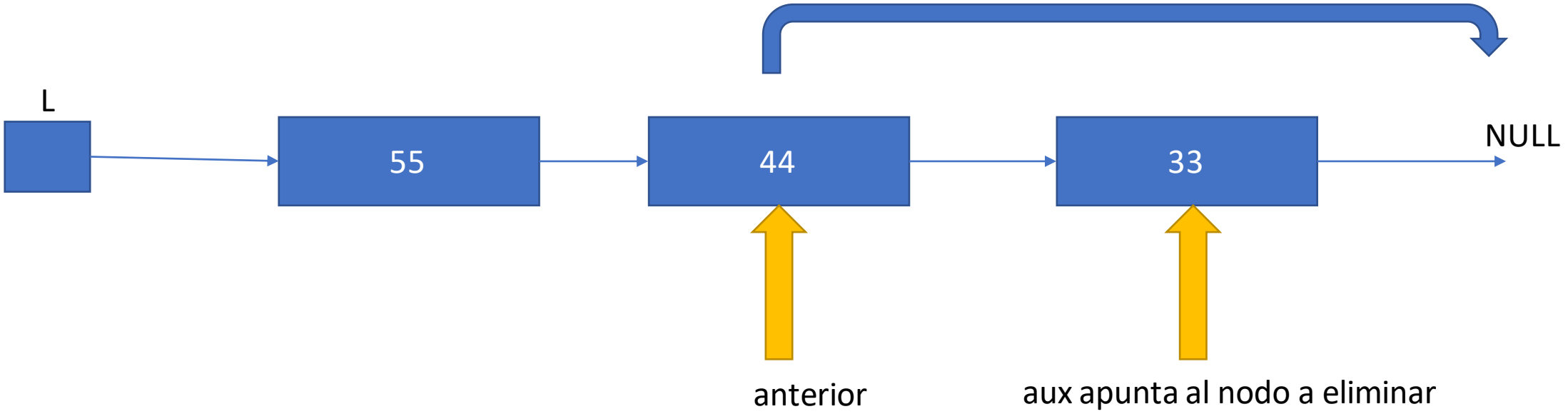


anterior == NULL

Aux apunta al nodo a borra

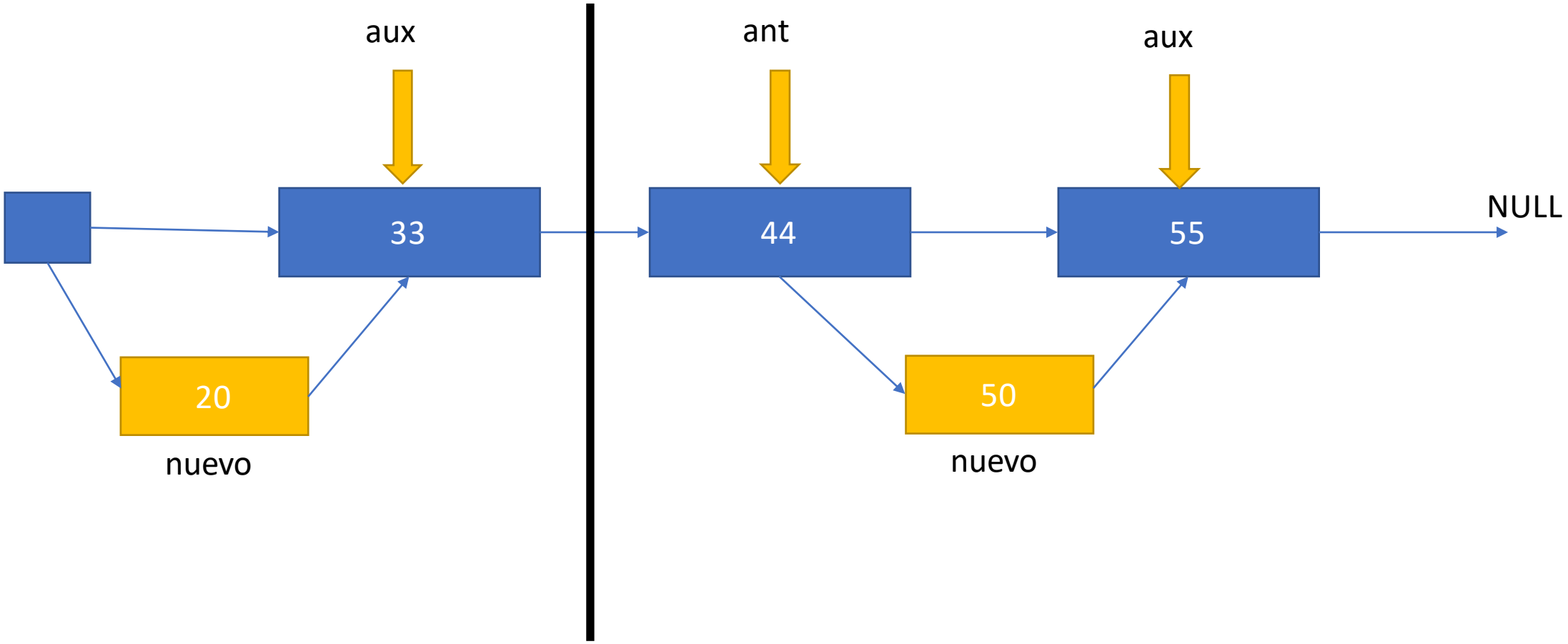
Borrar el 55

Listas: Borrar un nodo (al final)



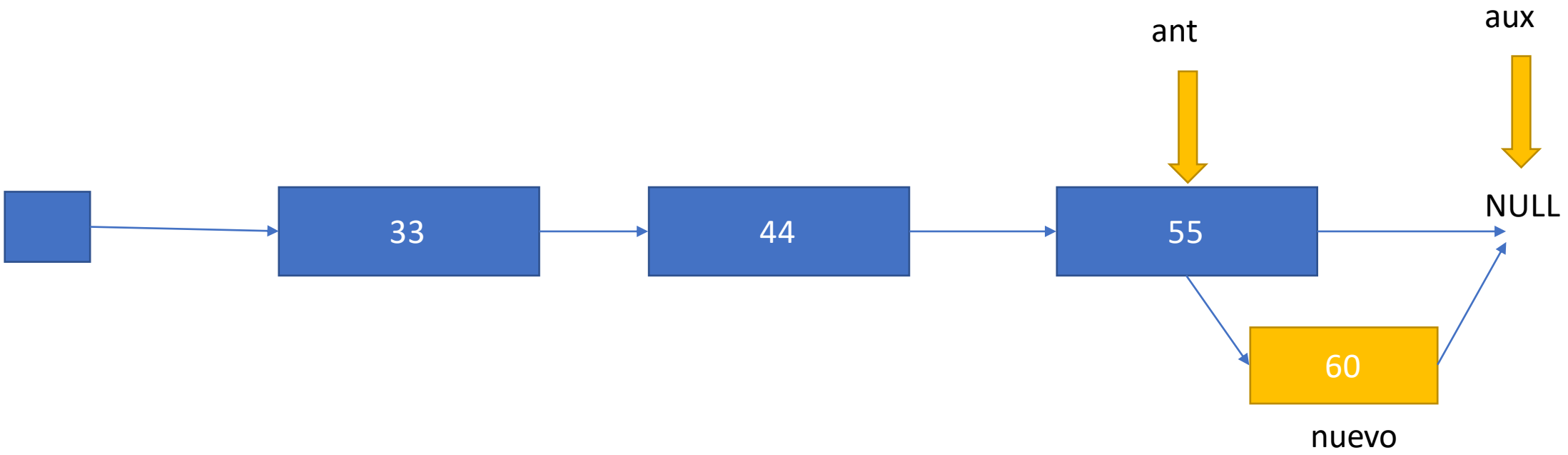
Borrar el 33

Listas: Inserción ordenada (Al inicio y en medio)



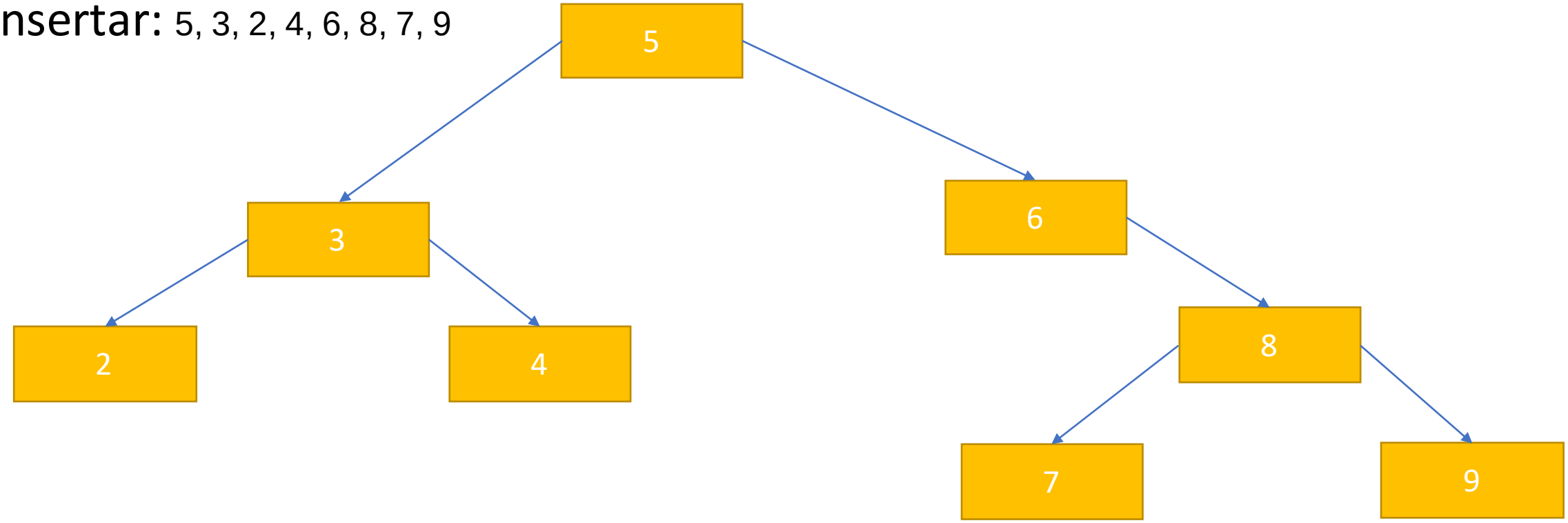
AL **INICIO**: Si la lista está vacía el nuevo nodo es la Lista

Listas: Inserción ordenada (Al final)



Arboles Binarios

- Insertar: 5, 3, 2, 4, 6, 8, 7, 9



RECORRIDOS
PREORDEN (**ORDEN DE INSERCION**): R-I-D: 5 - 3 - 2 - 4 -6 - 8 - 7 - 9
INORDEN (**ORDENADOS SEGÚN CLAVE**): I-R-D: 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9
POSTORDEN (**LIBERAR**): I-D-R: 2 - 4 - 7 - 9 - 8 - 6 - 5