# 4️⃣ Reto 4 ED 21/22


> Alumno: Javier Gómez López <javi5454@correo.ugr.es>
> Fecha: 6 de diciembre, 2021.
> Curso: 2º Doble Grado en Ingeniería Informática y Matemáticas, 2021/2022


## Desafío

El enunciado del reto es el siguiente:

- *Diseñar un procedimiento para leer/escribir un árbol binario a/de disco de forma que se recupere la estrucura jerárquica de forma unívoca usando el mínimo número de centinelas que veáis posibles. El reto queda resuelto simplemente rebajando el número de centinelas que yo uśe en clase cuando os comenté el método de lectura/escritura preorden con centinelas, pero tomadlo como una competición entre vosotros para conseguir dar la mejor solución, que será la que use un menor número de datos para el proceso de lectura/escritura de/a disco del árbol binario.*
- *Hay total libertad de diseño en la solución, de forma que pueden usarse diferentes tipos de centinelas, o cualquier idea que tenga sentido. La única restricción es que hagáis lo que hagáis, el árbol debe recuperarse de forma unívoca cuando se ejecute el procedimiento de lectura.*

## Solución

La solución que propongo consiste en utilizar más de un tipo de centinela:

- `n` simboliza que el nodo tiene dos hijos
- `l` simboliza que el nodo solo tiene hijo a la izquierda
- `r` simboliza que el nodo solo tiene hijo a la derecha
- `h` simboliza que el nodo es una hoja, es decir, no tiene hijos.

Con este sistema de centinelas, podemos reescribir las funciones `lee_arbol()` y `escribe_arbol()` de la siguiente manera:

### `escribe_arbol()`

```
template <class Tbase>
void ArbolBinario<Tbase>::escribe_arbol(ostream& flujo, Nodo n) const{
  if(n->izqda == nullptr && n->drcha == nullptr){
      flujo << "h " << n->etiqueta << " ";
  }
  else if(n->drcha == nullptr){
      flujo << "l " << n->etiqueta << " ";
      escribe_arbol(flujo, n->izqda);
  }
  else if(n->izqda == nullptr){
      flujo << "r " << n->etiqueta << " ";
      escribe_arbol(arbol, n->drcha);
  }
  else{
    flujo << "n " << n->etiqueta << " ";
    escribe_arbol(flujo, n->izqda);
    escribe_arbol(flujo, n->drcha);
    }
```

```
}
```

Con esta implementación, podemos especificar la descendencia de cada nodo de manera directa. Si el árbol tiene 2 hijos, el procedimiento es el habitual. Pero si el nodo solo tuviese un hijo por la izquierda, se escribiría por pantalla una `l`, y solo se llamaría de manera recursiva a `escribe_arbol()` con el nodo hijo de la izquierda. De manera análoga se actua con los nodos que solo tienen descendencia por la derecha. En el caso de que el nodo no tenga descendencia, es decir, sea una hoja, simplemente se muestra por pantalla su etiqueta `h` y se abandona dicha rama.

## `lee_arbol()`

```
template <class Tbase>
void ArbolBinario<Tbase>::lee_arbol(istream &flujo, Nodo &n){
  char c;
  flujo >> c;
  if (c=='n'){
    n = new nodo;
    flujo >> n->etiqueta;
    lee_arbol(flujo, n->izqda);
    lee_arbol(flujo, n->drcha);
    n->izqda->padre = n;
    n->drcha->padre = n;
  }
  else if (c == 'r'){
    n = new nodo;
    flujo >> n->etiqueta;
    lee_arbol(flujo, n->drcha);
    n->drcha->padre = n;
  }
  else if (c == 'l'){
      n = new nodo;
      flujo >> n->etiqueta;
      lee_arbol(flujo, n->izqda);
      n->izqda->padre = n;
  }
  else if (c == 'h'){
      n = new nodo;
      flujo >> n->etiqueta;
  }
}
```

Usamos una lógica similar a la que usamos en el método anterior. En este caso, el funcionamiento es el mismo si se introduce una `n` pues en ese caso el nodo tendría dos hijos. Sin embargo, ahora cuando se lea el carácter `l`, se tratará solo con el nodo hijo de la izquierda, pues se conoce que no tiene descendencia por la derecha el nodo padre. De manera análoga ocurre con el carácter `r`. De introducirse una `h`, simplemente creamos un nuevo nodo con su etiqueta, sin continuar por la rama de dicho nodo.