Árboles binarios de búsqueda

Cómo eliminar nodos

ABB – Eliminar un nodo

- Eliminar un nodo de un árbol puede parecer una operación trivial, aunque luego de analizar en profundidad lo que tendría que suceder veremos que no es sencillo.
- El algoritmo resulta más fácil de ver si separamos las diferentes posibilidades.
- Eliminar
  - Una hoja (nodo sin hijos)
  - Un nodo con un solo hijo (subárbol izquierdo o derecho)
  - Un nodo con ambos hijos.

```
procedure eliminar(var arbol:TArbol; e:TElemento);
begin
        //busca e en el árbol. Retorna en nElimnar el nodo a eliminar…
        buscar(arbol, e, nEliminar, nPadre);

        if(nEliminar=nil)then
            writeln('el nodo no se encuentra en el árbol')
        else
            begin
            // verificamos bajo qué situación nos encontramos
            if(nEliminar^.izq = nil)and(nEliminar^.der = nil)then
                //
                eliminarHoja(arbol, nEliminar, nPadre)
            else if(nEliminar^.der = nil)then
                //
                eliminarConHijoIzquierdo(arbol, nEliminar, nPadre)
            else if(nEliminar^.izq = nil)then
                //
                eliminarConHijoDerecho(arbol, nEliminar, nPadre)
            else
                //
                eliminarConAmbosHijos(nEliminar, nPadre);
            end;
end;
```

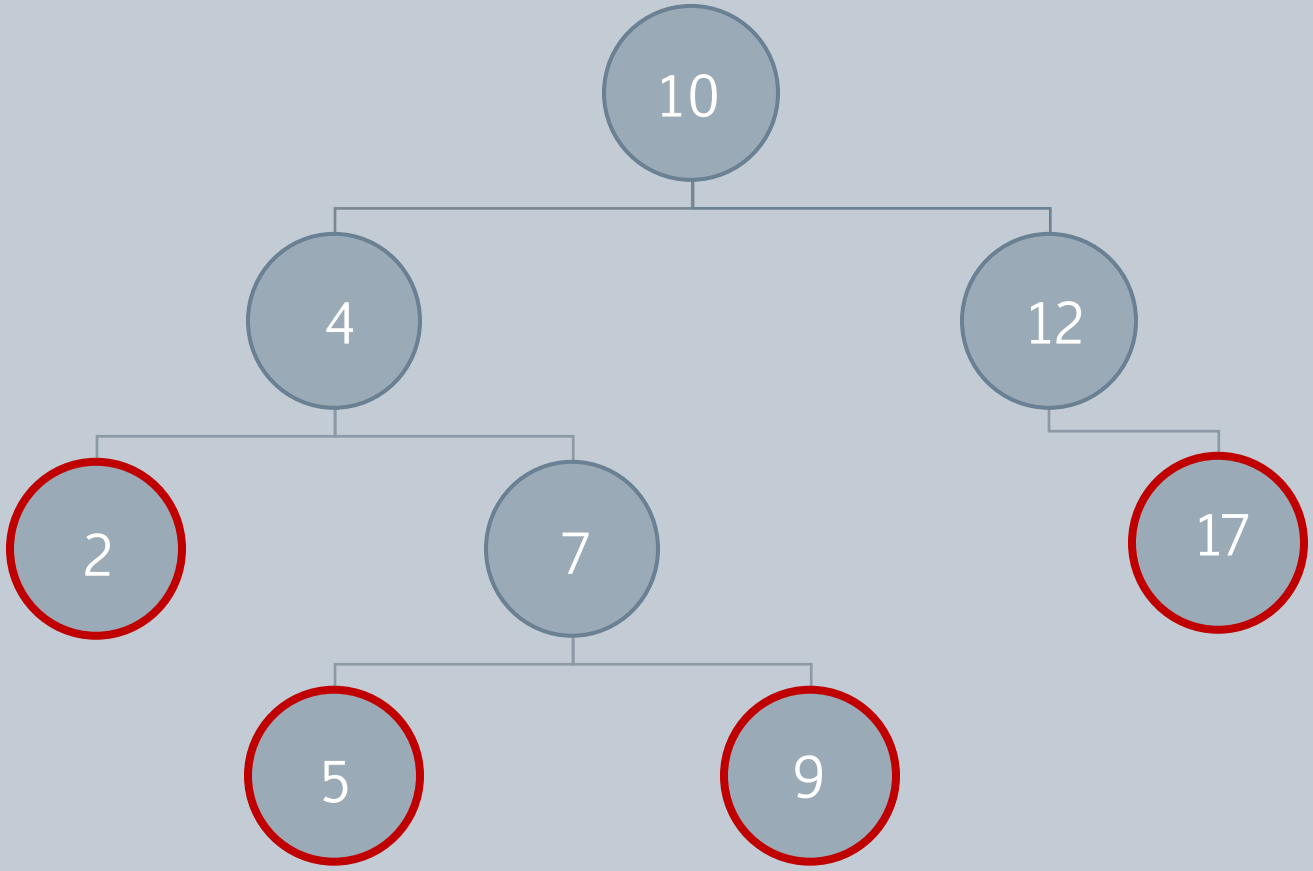
ELIMINAR UNA HOJA

El caso más sencillo.

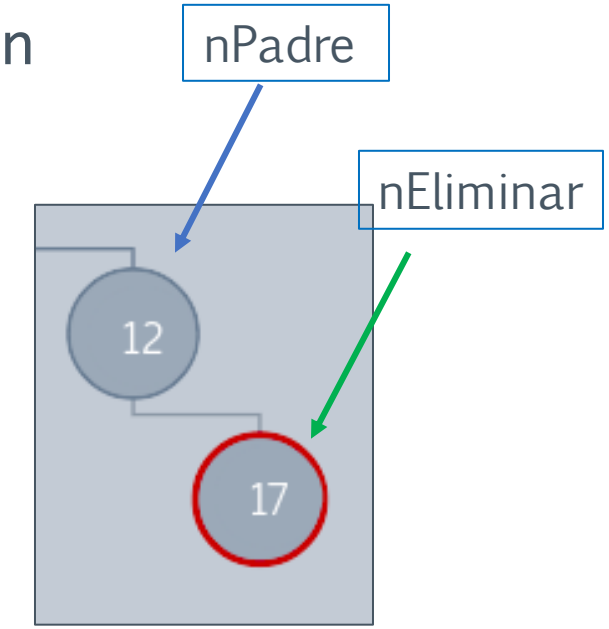
Se debe borrar el nodo y establecer en nulo el puntero del padre

Si borro el nodo 2, el padre (nodo 4) deberá establecer a nulo su hijo izquierdo.

Si borro 17, el hijo derecho de 12 apunta a nulo.



```
procedure eliminarHoja(var arbol,nEliminar,nPadre:TArbol);
begin
    //es raiz
    if(nPadre=nil)then
        arbol := nil
    else
        begin
        //verifico cual tengo que borrar
        if(nPadre^.der = nEliminar)then
            nPadre^.der := nil
        else
            nPadre^.izq := nil;
        end;
    dispose(nEliminar);
end;
```



ELIMINAR NODO CON UN HIJO.

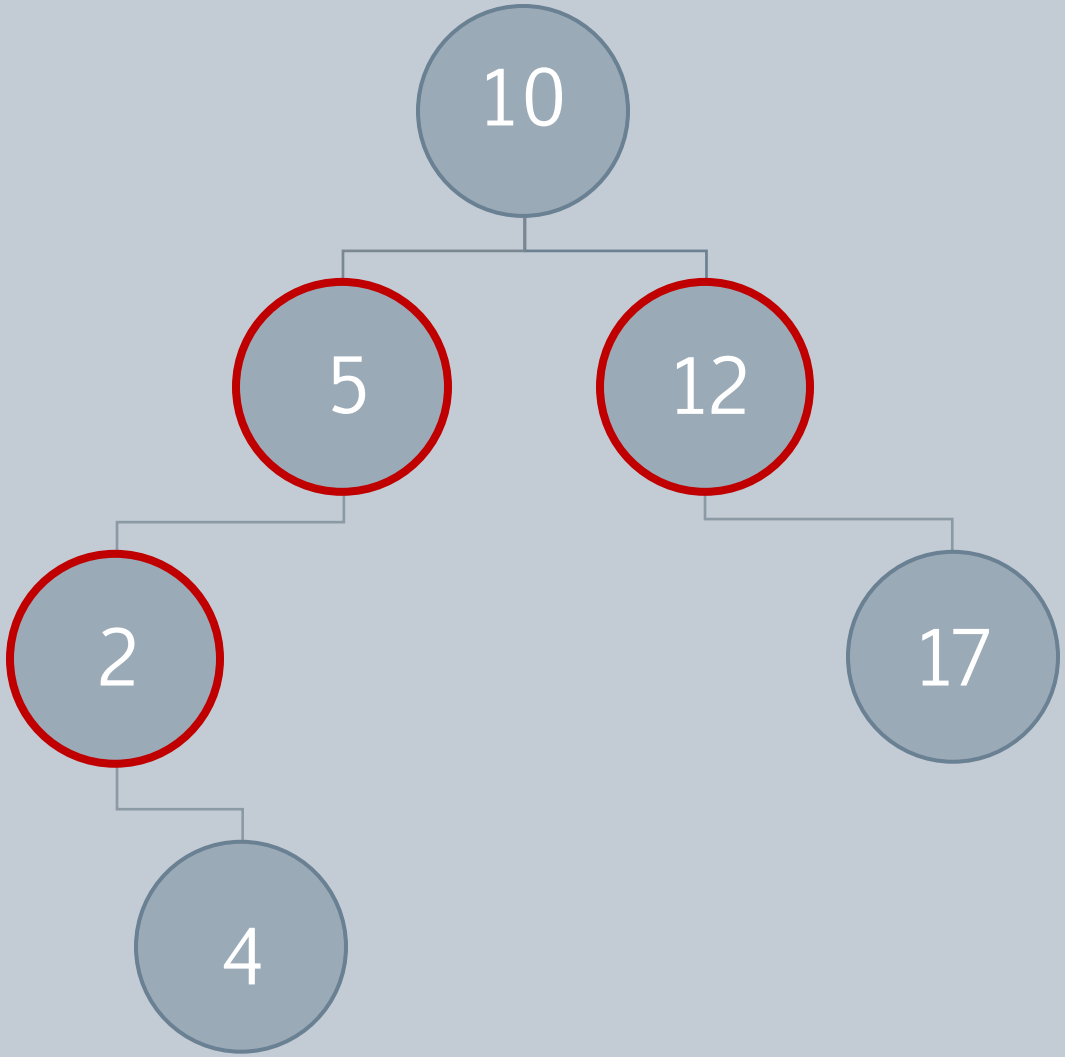
Borrar el nodo.

El subárbol que tiene pasa a ocupar el lugar del nodo borrado.

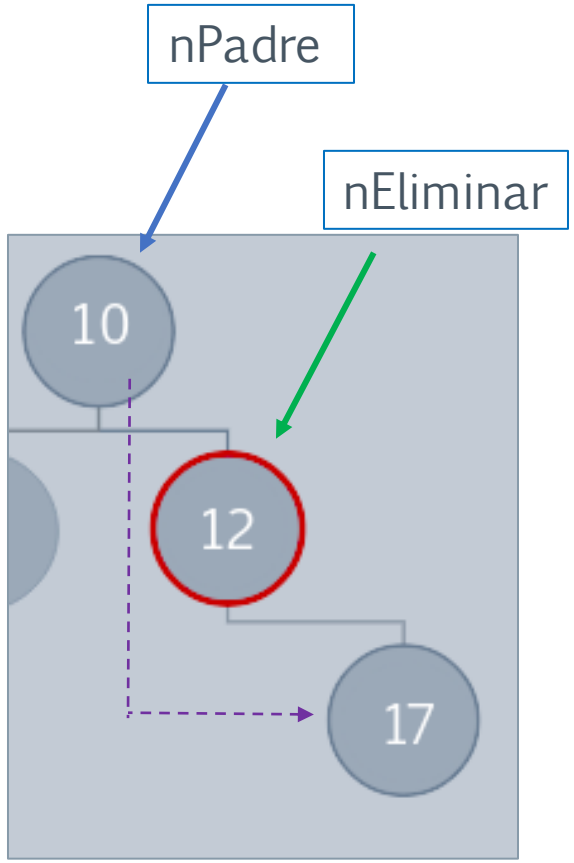
Si borro el nodo 5, el padre tendrá como nuevo hijo izquierdo al nodo 2

Si borro el nodo 2, el 5 tendrá como HI a 4

Si borro 12, 10 tendrá como ID a 17



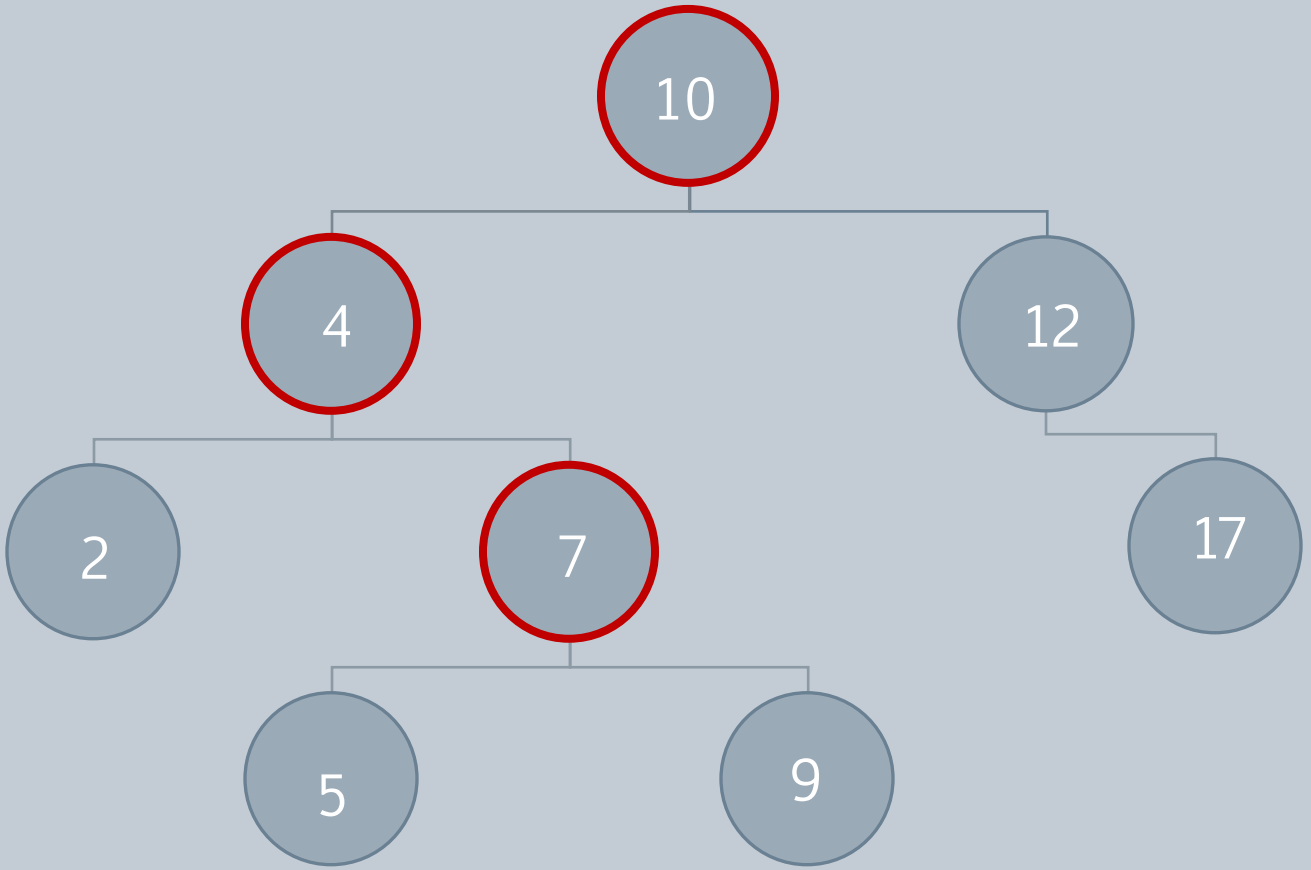
```
procedure eliminarConHijoIzquierdo(var arbol, nEliminar, nPadre:TArbol);
begin
        if(nPadre= nil)then
                arbol := nEliminar^.izq
        else
                begin
                //verifico cual le asigno al padre
                if(nPadre^.izq = nEliminar)then
                        nPadre^.izq := nEliminar^.izq
                else
                        nPadre^.der := nEliminar^.izq;
                end;
        dispose(nEliminar);
end;
```



ELIMINAR NODO CON AMBOS HIJOS.

Se utiliza la técnica de buscar el inmediato sucesor o el inmediato antecesor.

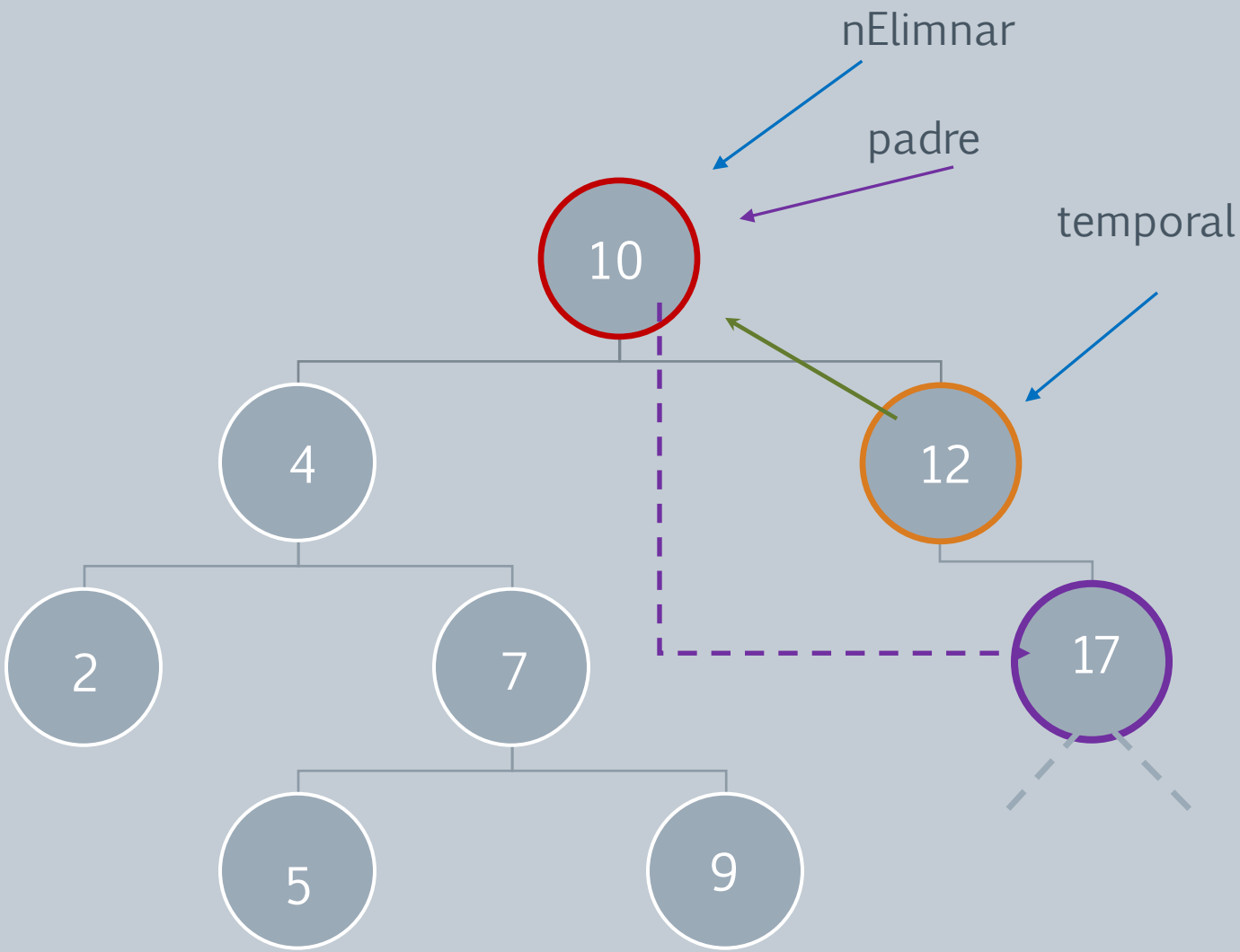
- **Inmediato sucesor** sería el nodo más chico del subárbol de la derecha (de los mayores)
- **Inmediato antecesor** sería el nodo más grande del subárbol de la izquierda (de los menores)



ELIMINAR NODO CON AMBOS HIJOS.

Eliminar el **nodo 10** con la técnica del inmediato sucesor.

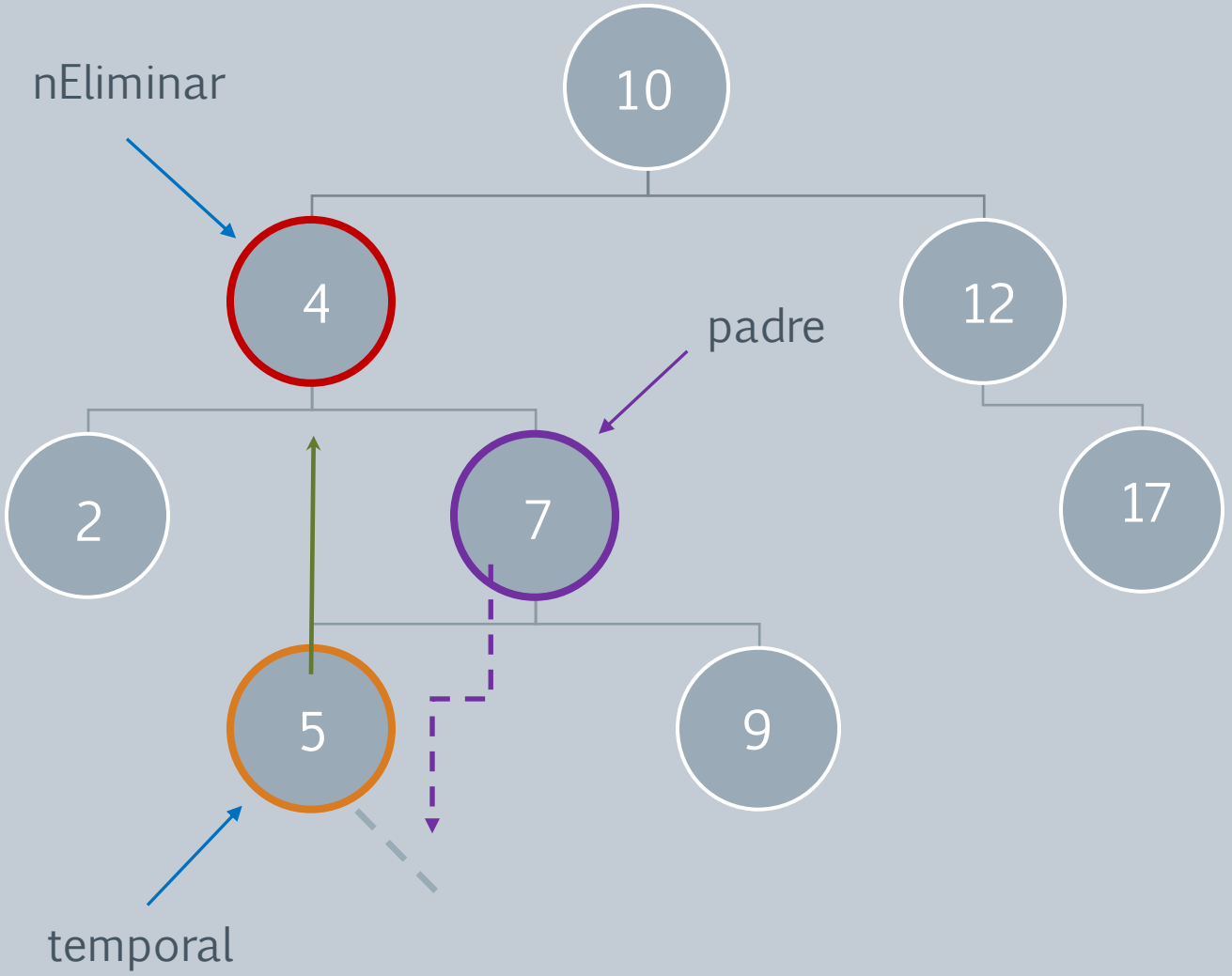
- **Buscar el Inmediato sucesor.**
- Se establece que el **12 es el nodo inmediato sucesor**.
- Se **copia** el contenido del nodo 12 al 10.
- Se enlaza el **nodo padre** (hijo derecho) con el hijo derecho del **nodo temporal (17)**.
- **Se elimina el temporal**



ELIMINAR NODO CON AMBOS HIJOS.

Eliminar el **nodo 4** con la técnica del inmediato sucesor.

- **Buscar el Inmediato sucesor.**
- Se establece que el **5 es el nodo inmediato** sucesor..
- Se **copia** el contenido del nodo 5 al 4.
- Se enlaza el **nodo padre** (hijo izquierdo) con el hijo derecho del **nodo temporal**
- **Se elimina el temporal**



```
procedure eliminarConAmbosHijos(var nEliminar, padre:TArbol);
var temporal:TArbol;
begin
    //busco el menor del subárbol derecho (más grandes)
    padre:= nEliminar;
    temporal := nEliminar^.der;
    //busco el nodo temporal (más chico)
    while(temporal^.izq <> nil)do
        begin
        padre:= temporal;
        temporal := temporal^.izq;
        end;
    nEliminar^.info := temporal^.info;
    // verifico con quien conecto al padre
    if(padre = nEliminar)then //no se avanzo en la búsqueda del temp.
        padre^.der := temporal^.der
    Else
        padre^.izq := temporal^.der;
    //
    dispose(temporal);
end;
```

Bibliografía

- Data Structures. Nalle Dale.
- Estructuras de datos en C. A, Tenenbaum - Y, Langsam – M, Augenstein Un libro
- Estructuras de datos y algoritmos. M, Weiss

Herramientas:

- http://btv.melezinek.cz/binary-search-tree.html
- https://www.cs.usfca.edu/~galles/visualization/BST.html