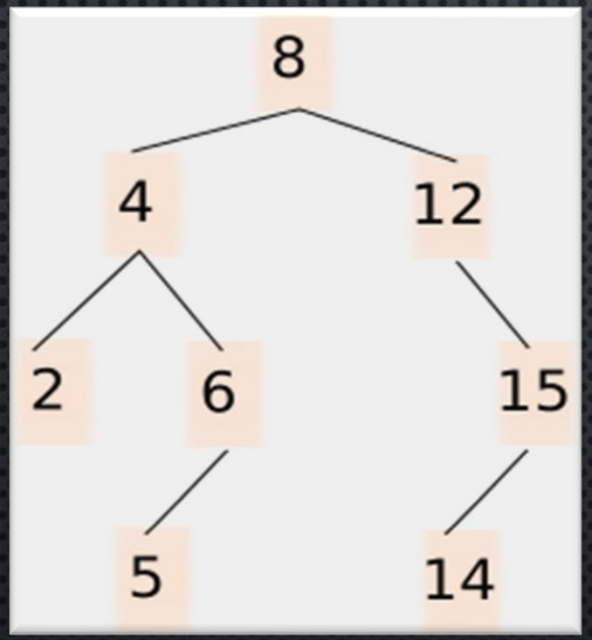
ARBOLES: ARBOLES BINARIOS DE BUSQUEDA (ABB)

Arbol Binarios de Búsqueda.- Son árboles binarios que cumplen que la etiqueta de cada nodo es mayor que todos las etiquetas del subárbol hijo izquierda, y menor que las etiquetas del subárbol hijo derecha. No existen etiquetas repetidas.



OPERACIONES:

- Preguntar por si exsite un elemento
- Insertar un nuevo elemento
- Borrar un elemento

La clase set de la STL es un ABB.

Los recorridos inorden de un ABB da una secuencia de las etiquetas ordenadas.

ARBOLES: ARBOLES BINARIOS DE BUSQUEDA (ABB)

Ejercicio. 1- Construir un ABB con las siguiente claves {5,7,13,20,80,1,3,0, 12}

ARBOLES: ARBOLES BINARIOS DE BUSQUEDA (ABB)

Algoritmo para Buscar un elemento en un ABB.

```
template <class T>
struct info_nodo{
  T et;
  info_nodo<T> *padre;
  info_nodo<T> *hizq;
  info_nodo<T> *hder;
};
```

```
template <class T>
info_nodo<T> * Buscar(info_nodo<T>*n, const T &x){
1.   if (n==nullptr)
2.        return nullptr;
3.   else
4.      if (x==n->et)
5.           return n;
6.      else if (x<n->et)
7.              return Buscar(n->hizq,x);
8.        else
9.              return Buscar(n->hder,x);
10.
}
```

¿Eficiencia?

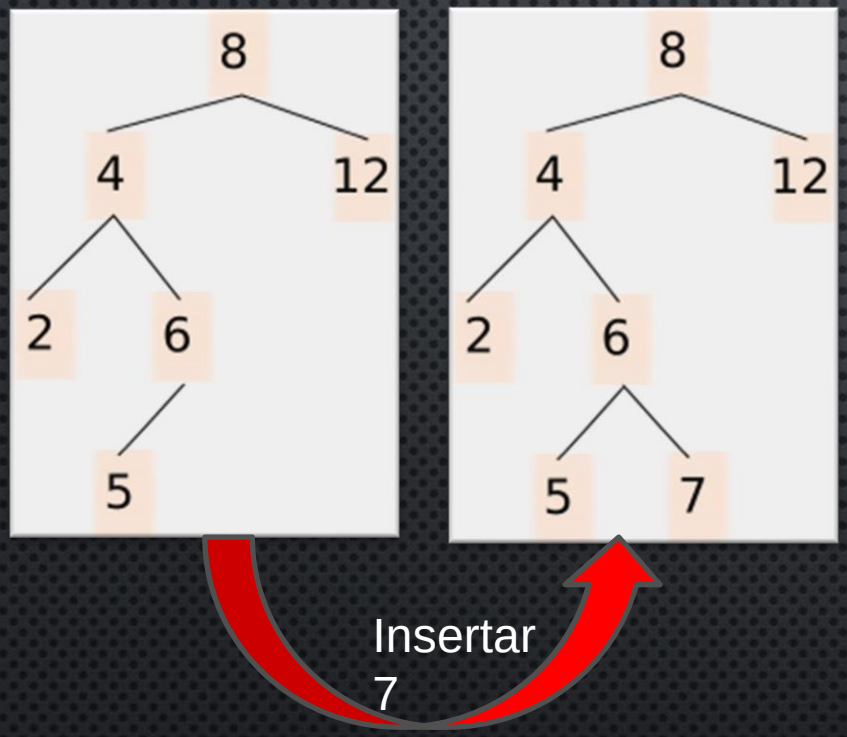
ARBOLES: ARBOLES BINARIOS DE BUSQUEDA (ABB)

Ejercicio – Obtener la implementación del algoritmo para Buscar un elemento en un ABB de forma iterativa

```
template <class T>
struct info_nodo{
  T et;
  info_nodo<T> *padre;
  info_nodo<T> *hizq;
  info_nodo<T> *hder;
};
```

ARBOLES: ARBOLES BINARIOS DE BUSQUEDA (ABB)

Insertar un nuevo elemento en un ABB




```
template <class T>
bool Insertar(info_nodo<T>* &n, const T &x){
1. bool res=false;
2. if (n==nullptr){
3.    n=new info_nodo(x);
4.    return true;
5. }
6. else if (x<n->et){
7.        res = Insertar(n->hizq,x);
8.        if (res)
9.            n->hizq->padre=n;
10.      return res;
11.}
12.else if(x>n->et){
13.       res = Insertar(n->hder,x);
14.       if (res)
15.           n->hder->padre=n;
16.       return res;
17.}
18.else
19.   return false;
20.}
```

ARBOLES: ARBOLES BINARIOS DE BUSQUEDA (ABB)

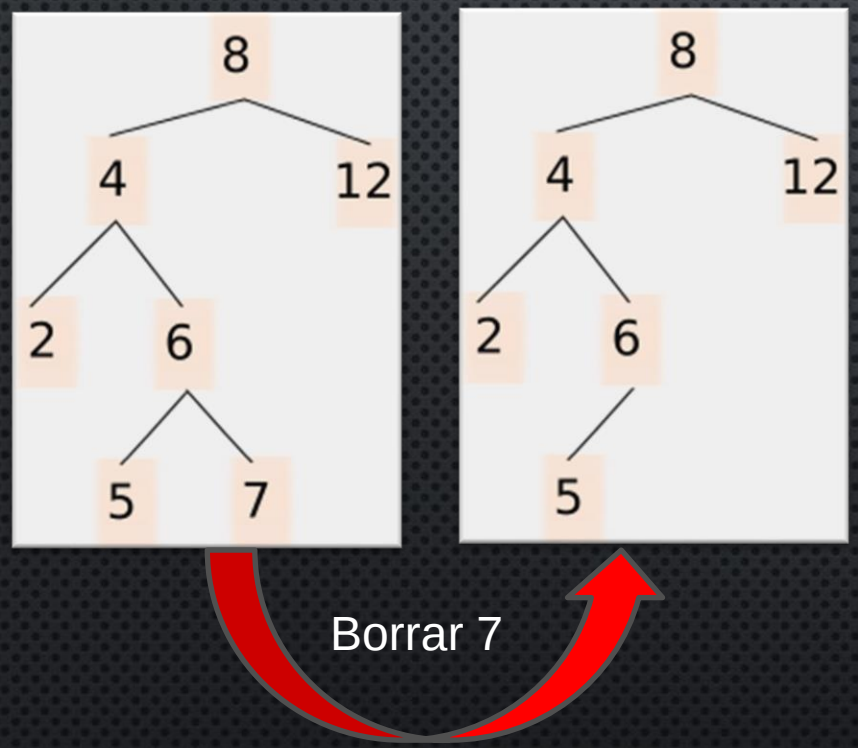
Ejercicio – Obtener la implementación del algoritmo para insertar un elemento en un ABB de forma iterativa

```
template <class T>
struct info_nodo{
  T et;
  info_nodo<T> *padre;
  info_nodo<T> *hizq;
  info_nodo<T> *hder;
};
```

ARBOLES: ARBOLES BINARIOS DE BUSQUEDA (ABB)

Borrado de una clave en un ABB.

Posibilidad 1.- El elemento a borrar está en una hoja (CASO 0)



Queremos borrar el elemento que apunta n

CODIGO.-

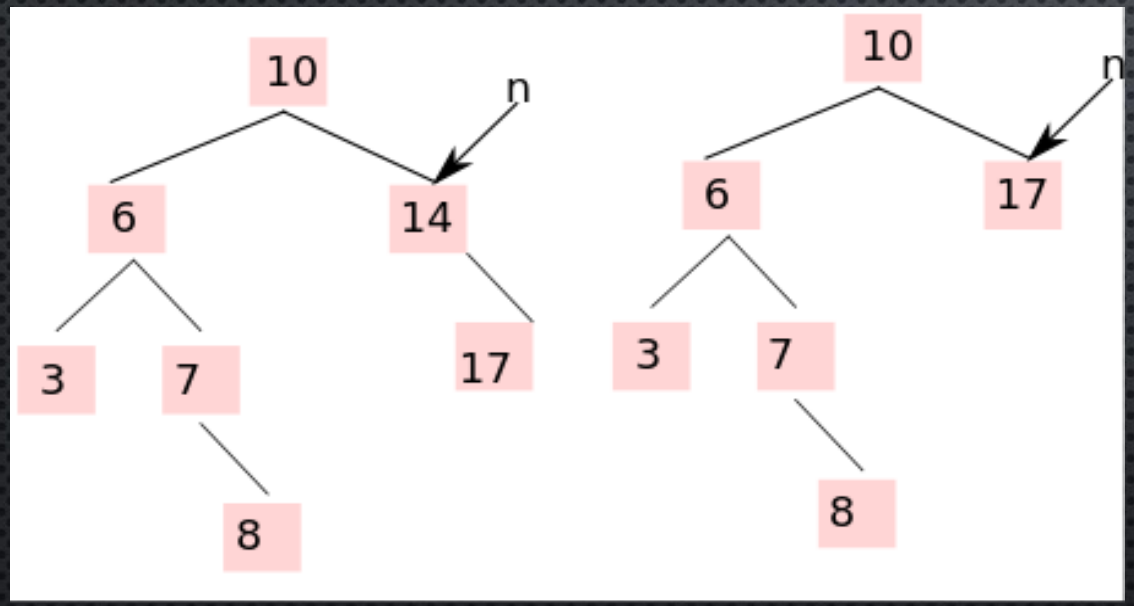
```
1. info_nodo<T> *aux =n;
2. if (aux->padre!=nullptr){
3.   if (aux->padre->hder==n)//es n el  hijo der.
4.      aux->padre->hder=nullptr;
5.   else
6.      aux->padre->hizq=nullptr;
7. }
8. delete aux;
```

ARBOLES: ARBOLES BINARIOS DE BUSQUEDA (ABB)

Borrado de una clave en un ABB.

Posibilidad 2.- El elemento a borrar no está en un nodo hoja

CASO 1.- El nodo a borrar solamente tiene un hijo a la derecha



Queremos borrar el elemento que apunta n

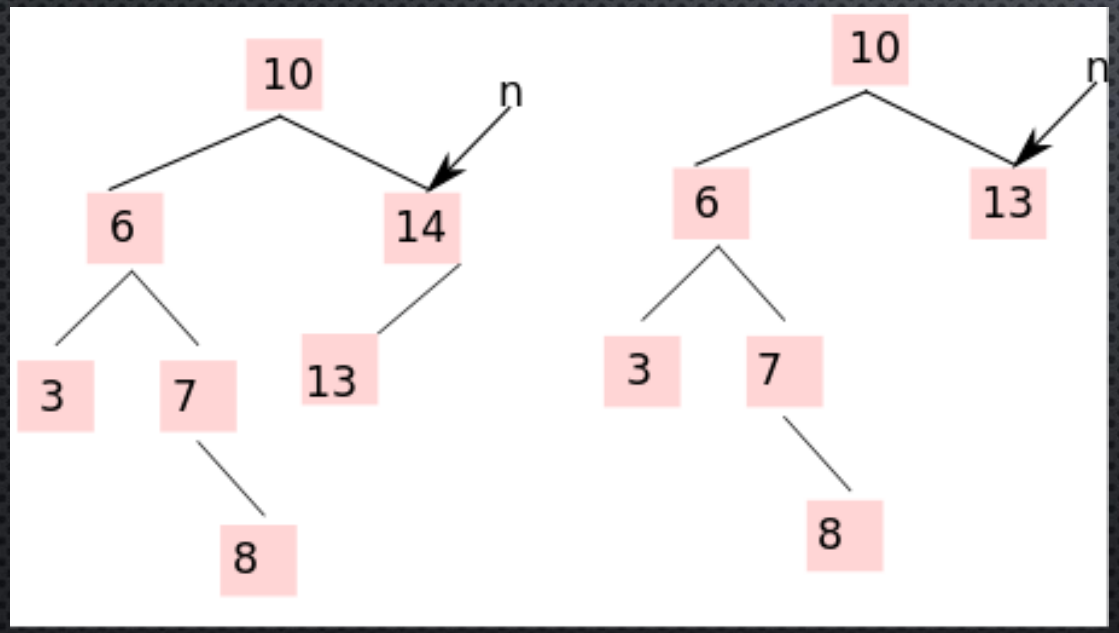
CODIGO.-

```
1. info_nodo<T> *padre =n->padre;
2. if (padre!=nullptr){
3.   if (padre->hder==n){
4.     padre->hder=n->hder;
5.     n->hder->padre=padre;
6.   }
7.   else{
8.      padre->hizq=n->hder;
9.      n->hder->padre=padre;
10.  }
11.info_nodo<T>*aux=n;
12.n=n->hder; delete aux;
```

ARBOLES: ARBOLES BINARIOS DE BUSQUEDA (ABB)

Borrado de una clave en un ABB.

Posibilidad 2.- El elemento a borrar no está en un nodo hoja
CASO 2.- El nodo a borrar solamente tiene un hijo a la izquierda



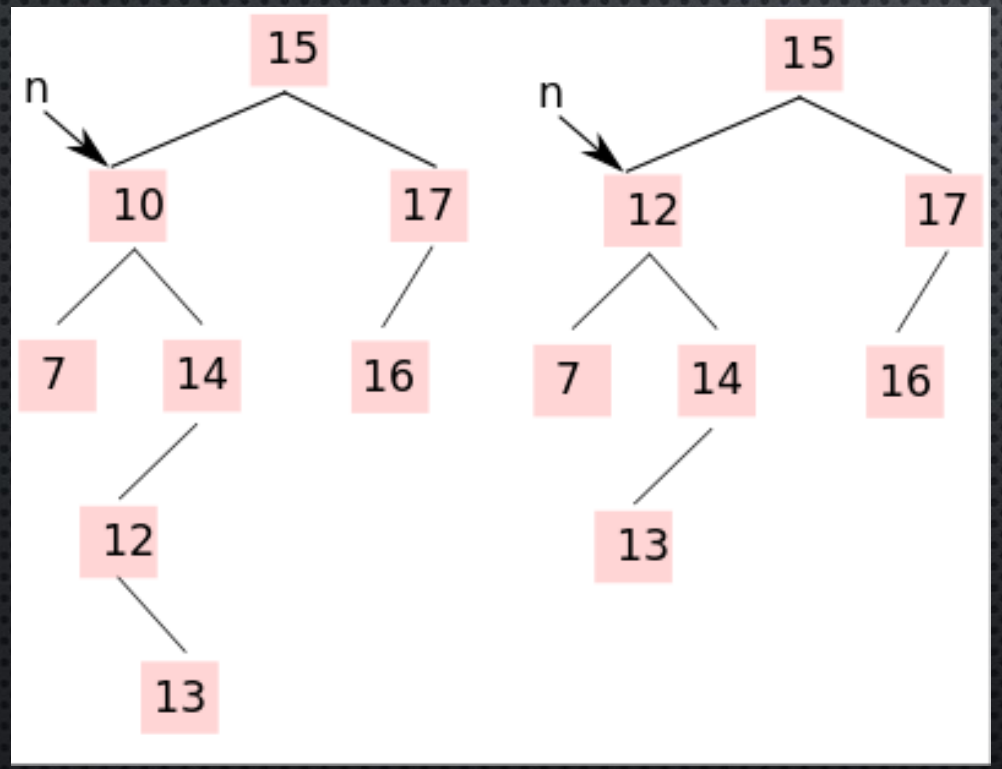
Queremos borrar el elemento que apunta n
CODIGO.-

```
1. info_nodo<T> *padre =n->padre;
2. if (padre!=nullptr){
3.   if (padre->hder==n){
4.     padre->hder=n->hizq;
5.     n->hizq->padre=padre;
6.   }
7.   else{
8.     padre->hizq=n->hizq;
9.     n->hizq->padre=padre;
10.  }
11.info_nodo<T>*aux=n;
12.n=n->hizq; delete aux;
```

ARBOLES: ARBOLES BINARIOS DE BUSQUEDA (ABB)

Borrado de una clave en un ABB.

Posibilidad 2.- El elemento a borrar no está en un nodo hoja
CASO 3.- El nodo a borrar tiene dos hijos (hijo a la izquierda e hijo a la derecha)




```
Queremos borrar el elemento que apunta n
CODIGO.-
1. info_nodo<T>*aux=n->hder;
2. while (aux->hizq!=nullptr)
3.    aux = aux->hizq;
4. n->et=aux->et;
5. Borrar(aux,aux->et);
```

ARBOLES: ARBOLES BINARIOS DE BUSQUEDA (ABB)

Borrado de una clave en un ABB.

```
void Borrar(info_nodo<T>*n, int x){
1.  if (n!=nullptr){
2.   if (n->et==x)
3.    EliminarRaiz(n);
4.   else
5.    if (n->et<x)
6.     Borrar(n->hder,x);
7.    else
8.     Borrar(n->hizq,x);
9.  }
10.}
void PutPadre(info_nodo<T>*n,
              info_nodo<T>*nuevo){
1.   info_nodo<T>*padre=n->padre;
2.   if (padre!=nullptr){
3.      if (padre->hder==n)
4.        padre->hder=nuevo;
5.      else padre->hizq=nuevo;
6.      nuevo->padre=padre;
7.   }}
```

```
void EliminarRaiz(info_nodo<T>*n, int x){
1. //CASO 0
2. if (n->hizq==nullptr && n->hder==nullptr){
3.    PutPadre(n,nullptr); delete n;
4. else
5.  if (n->hizq==nullptr && n->hder!=nullptr){
6.    //CASO 1
7.    PutPadre(n,n->hder);
8.    info_nodo<T>*aux = n;
9.    n=n->hder;   delete aux;
10. }
11. else
12.  if (n->hizq!=nullptr && n->hder==nullptr){
13.     //CASO 2
14.     PutPadre(n,n->hizq);
15.     info_nodo<T>*aux=n;
16.     n=n->hizq; delete aux;
17.  }
18.  else{ //CASO 3
19.   info_nodo<T>*aux=n->hder;
20.   while (aux->hizq!=nullptr)
21.      aux = aux->hizq;
22.   n->et=aux->et;
23.   Borrar(aux,aux->et);}}
```