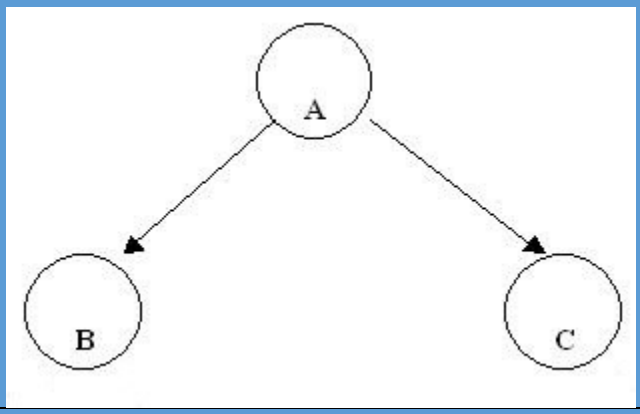
TAD ABB



{inv. : Nodo = {Llave, Valor} ^ {A,B,C} ∈ Nodo ^ B <= A < C ^ A ≠ NIL}

Operaciones Primitivas:

- CrearABB: -> ABB
- BuscarEnABB: ABB x Llave -> Nodo
- EliminarEnABB: ABB x Llave -> ABB
- AgregarEnABB: ABB x Nodo -> ABB
- EstaVacioABB: ABB -> Booleano

CrearABB()

"Crea un árbol binario de búsqueda el cual será la cabeza o root de los elementos que se agreguen después."

{pre: True}

{post: Crea un árbol binario de búsqueda vacío}

BuscarEnABB(abb, llave)

"Devuelve un nodo donde nodo.llave = llave"

{pre: abb y llave ≠ NIL}

{post: nodo donde nodo.llave = llave}

EliminarEnABB(abb, llave)

"Borra un nodo con la primera coincidencia donde nodo.llave = llave"

{pre: abb y llave ≠ NIL}

{post: abb con nodo.llave = llave eliminado}

AgregarEnABB(abb, nodo)

"Agrega un nodo, que contiene llave y valor, en el árbol binario de búsqueda abb"

{pre: abb y nodo ≠ NIL}

{post: nodo ∈ abb}

EstaVacioABB(abb)

"Devuelve un valor booleano de verdadero o falso si el árbol no tiene hijos"

{pre: abb ≠ NIL}

{post: True si el hijo derecho y el hijo izquierdo son NIL y False de lo contrario}