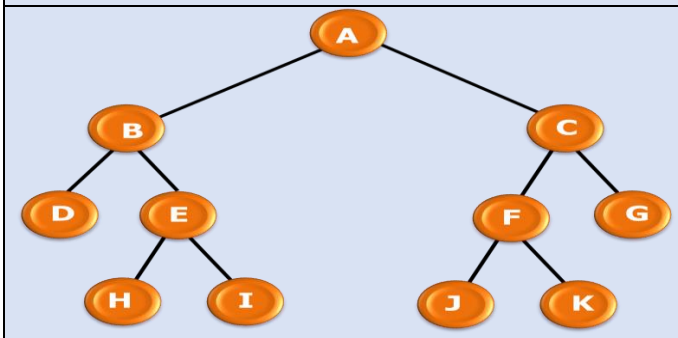
TAD ABB



{*inv.: Se deben cumplir la siguiente propiedad de orden. (1) Todos los datos de su subárbol izquierdo son menos o iguales que el dato que ocupa la raíz. (2) Todos los datos de su subárbol derecho son mayores que el dato que ocupa la raíz. (3) Los subárboles izquierdo y derecho también son ABB*}

Operaciones Primitivas:

- CrearABB: ->ABB
- DarRaiz: *ABB* ->Nodo
- ModificarRaiz: *ABBxNodo* ->ABB
- Buscar: *ABBxLlave* ->Nodo
- Eliminar: *ABBxLlave* ->ABB
- Insertar: *ABBxNodo* ->ABB
- Sucesor: *ABBxLlave* ->Nodo
- Predecesor: *ABBxLlave* ->Nodo
- EsHoja: *ABBxLlave* ->Booleano
- ArbolMinimo: *ABB* ->Nodo
- ArbolMaximo: *ABB* ->Nodo

CrearABB()

"Crea un árbol binario de búsqueda en el que la raíz es el primer elemento y no existe, por lo tanto, es nula"

{*pre:* TRUE}

{post*: Crea un árbol binario de búsqueda vacío*}

DarRaiz()

"Da la raíz del árbol binario de búsqueda"

{*pre:* raíz!=nulo}

{*post: Se da la raíz del ABB*}

ModificarRaiz(Nodo**)**

"Modificar el nodo que se encuentra como raíz del ABB"

{*pre:* TRUE}

{*post: Se modifica el nodo raíz del ABB por un nuevo nodo*}

Buscar(Llave**)**

"Busca un nodo específico de acuerdo con la llave dada"

{*pre:* TRUE}

{*post: Si algún nodo.llave = Llave -> Se da este nodo,*
Si no -> se devuelve nulo}

Eliminar(Llave**)**

"Elimina el nodo del ABB de acuerdo con la llave dada"

{*pre:* TRUE}

{*post: Si algún nodo.llave = llave -> Se elimina este nodo,*
Si no -> no se hace ninguna modificación}

Insertar(Nodo**)**

"Se inserta un nuevo en el árbol binario de búsqueda"

{*pre:* TRUE}

{*post: El ABB tiene un nuevo nodo en su árbol*}

Sucesor(Llave**)**

"Se da el nodo sucesor del nodo al que le pertenece la Llave"

{*pre:* TRUE}

{*post: Si la Llave ∈ algún nodo en el árbol y este nodo != hoja -> Se da el nodo mínimo del hijo derecho de este nodo (sucesor),*
Si la llave ∉ algún nodo en el árbol -> Se devuelve un valor nulo}

Predecesor(Llave**)**

"Se da el nodo predecesor del nodo al que pertenece la Llave"

{*pre:* TRUE}

{*post: Si la Llave ∈ algún nodo en el árbol y este nodo != hoja -> Se da el nodo máximo del hijo izquierdo de este nodo (predecesor),*
Si la llave ∉ algún nodo en el árbol -> Se devuelve un valor nulo}

EsHoja(Llave**)**

"Da a conocer si el nodo al que le pertenece la llave es hoja"

{*pre:* TRUE}

{*post: Si la Llave ∈ algún nodo en el árbol y los hijos de este son nulos ->* True,
Si la Llave ∈ algún nodo en el árbol y algún hijo de este no es nulo -> False}

ArbolMinimo()

"Da el nodo con el valor menor en el ABB"

{*pre:* TRUE}

{*post: Si la raíz != nulo -> Se da el nodo con el valor mínimo en el ABB,*
De lo contrario -> Se devuelve un valor nulo}

ArbolMaximo()

"Da el nodo con el valor máximo en el ABB"

{*pre:* TRUE}

{*pre: Si la raíz != nulo -> Se da el nodo con el valor máximo en el ABB,*
De lo contrario -> Se devuelve un valor nulo}