Archivo Haskell: ABB.hs

```
--Arboles Binarios de Búsqueda en Haskell:
-----------------------------------------

--Definición:
-------------

data Arbol a = Nodo a (Arbol a) (Arbol a) | Nil
                                deriving Show
-- Se puede leer como:
-- "Un arbol sobre elementos del tipo a puede ser construído de dos formas:
-- (1) aplicando la función constructora Nodo a tres parámetros:
--                              : el primero de tipo a
--                  : el segundo de tipo Arbol sobre a
--                              : el tercero de tipo Arbol sobre a
--
-- o (2) usando la constante Nil para indicar árbol vacío.

-----------------------------------------------------------------
-- Función para construir un ABB
-----------------------------------------------------------------
-- En el caso en que inserte un elemento en un arbol vacío (Nil) se construye
-- un árbol con un Nodo que contendrá: elemento y dos subárboles vacíos.
-- En el otro caso, si no es vacío, entonces contiene un Nodo con valor en la raiz y dos
--              subárboles izq, der.
-- Este valor raiz se usa para decidir si se tiene que insertar elemento
-- en el subárbol izquierdo o en el subárbol derecho.
-----------------------------------------------------------------------

insertArbol :: Ord a => a -> Arbol a -> Arbol a
insertArbol elemento Nil                    = Nodo elemento Nil Nil
insertArbol elemento (Nodo raiz izq der) | elemento < raiz = Nodo raiz (insertArbol elemento izq) der
                                         | elemento  > raiz = Nodo raiz izq (insertArbol elemento der)
                                         | elemento == raiz = Nodo raiz izq der
-----------------------------------------------------------------------
-- Función para armar desde una lista de elementos un ABB
-- Ej [1,2,6] da un arbol: Nodo 6 (Nodo 2 (Nodo 1 Nil Nil) Nil) Nil
--    foldr trabaja así:  insertArbol 1 ( insertArbol 2 (insertArbol 6 Nil))

deListaaArbol :: Ord a => [a] -> Arbol a
deListaaArbol =  foldr insertArbol Nil
-------------------------------------------------------------------------
-- La función foldr inserta un operador entre todos los elementos de la lista
-- empezando a la derecha de la lista, si la lista es vacía aplica Nil
-----------------------------------------------------------------------
terminales :: Arbol a -> [a]
terminales Nil = []
terminales (Nodo raiz Nil Nil) = [raiz]
```

```
terminales (Nodo raiz izq der) = terminales (izq) ++ terminales (der)
-----------------------------------------
-- Dado un ABB determinar el número de nodos
-----------------------------------------
numNodos :: Arbol a -> Int
numNodos Nil = 0
numNodos (Nodo raiz izq der) = 1 + numNodos izq + numNodos der
-----------------------------------------------------------------

-- Dado un ABB y un elemento determinar si elemento está en el ABB
-----------------------------------------------------------------
buscar :: Ord a => a -> Arbol a -> Bool
buscar x  Nil = False
buscar x  (Nodo raiz izq der) = if x == raiz then True
                                else if x < raiz then buscar x izq
                                     else buscar x der
-----------------------------------------------------------------
-- Dado un ABB  determinar su altura
-----------------------------------------------------------------
altura :: Arbol a -> Int
altura Nil = 0
altura (Nodo raiz izq der) = maximo (altura izq ,altura der) + 1
                             where
                                  maximo (a,b) = if a >= b then a
                                                 else b


-----------------------------------------------------------------
-- Recorridos de un ABB
-----------------------------------------------------------------
preorden :: Arbol a -> [a]
preorden Nil = []
preorden (Nodo raiz izq der) = [raiz] ++ preorden izq ++ preorden der


-----------------------------------------------------------------
inorden :: Arbol a -> [a]
inorden Nil = []
inorden (Nodo raiz izq der) = inorden izq ++ [raiz] ++ inorden der

-----------------------------------------------------------------
posorden :: Arbol a -> [a]
posorden Nil = []
posorden (Nodo raiz izq der) = preorden izq ++ preorden der ++ [raiz]
```