```
Pseudocódigos para la inserción en un árbol B
Elaboró: MI Fco. Javier Rodríguez G
24.11.2020, 19:00
Rev: 1.2


NOTAS

max_order <- Lo introduce el usuario. Es el número máximo de hijos.
max = max_order - 1
min = floor( max / 2 )
{min y max son constantes estáticas de la clase Node}

{Los tamaños de los arreglos son:}
keys[max_order]
children[max_order + 1]

ALGORITMOS

Insert( root, key ): Bool
{
   Si root = Nil {
      root = new_node()
      root.keys[0] = key
      root.cnt = 1
   } si no {
      Si root.cnt == max {
         old_root = root
         root = new_node()
         root.children[0] = old_root
         root.leaf = false
         root = split_node( root, 0 )
      }
      root = insert_node( root, key )
   }

   Devuelve (root <> Nil)
}

insert_node( node, key ): Node
{
   i = node.cnt

   Si node.leaf = True {
      Mientras i > 0 AND key < node.keys[i-1] {
         node.keys[i] = node.keys[i-1]
         --i
      }

      node.keys[i] = key
      ++node.cnt

      (escribe "node" al disco)

   } si no {
      Mientras i > 0 AND key < node.keys[i-1] {
         --i
      }

      (lee del disco "node.children[i]")

      Si node.children[i].cnt == max {
         node.leaf = False

         node = node_split( node, i )

         Si key > node.keys[i] {
            ++i
         }
      }

      insert_node( node.children[i], key )
   }

   Devuelve node
}

split_node( parent, index ): Node
{
   left = parent.children[ index ]

   right = new_node()
   right.leaf = left.leaf;
   right.cnt = min

   Para( j = 0; j < min; ++j ){
      right.keys[ j ] = left.keys[ j + min + 1 ]
   }

   Si left.leaf = False {
      Para( j = 0; j <= min; ++j ){
         right.children[ j ] = left.children[ j + min + 1 ]
      }
   }

   left.cnt = min

   Para( j = parent.cnt; j > index; --j ){
      parent.children[ j + 1 ] = parent.children[ j ]
   }

    parent.children[ index + 1 ] = right

   Para( j = parent.cnt; j > index; --j ){
      parent.keys[ j ] = parent.keys[ j - 1 ]
   }

    parent.keys[ index ] = left.keys[ min ]

   ++parent.cnt

   return parent
}
```