ÁRBOLES DE BÚSQUEDA

Los árboles de búsqueda son un tipo particular de árboles binarios, que pueden definirse cuando el tipo de los elementos del árbol posee una relación < de orden total. Tienen la propiedad de que todos los elementos almacenados en el subárbol izquierdo son menores estrictos que el elemento raíz, que a su vez es menor estricto que todos los elementos del subárbol derecho. Además ámbos subárboles son árboles de búsqueda.

En algunas variantes se admite la existencia de elementos repetidos, en cuyo caso todos los elementos del subárbol izquierdo serían menores o iguales que la raíz.

1.- ESPECIFICACIÓN:

```
especificación ArbolBusqueda;
usa
  arbolBin
parámetros
  géneros
    telemento
  operaciones
    función esIgual(d1:telemento, d2:telemento) devuelve booleano
    {Devuelve el valor verdad si d1=d2, en caso contrario
    devuelve el valor falso }
    función esMenor(d1:telemento, d2:telemento) devuelve booleano
    {Devuelve el valor verdad si d1<d2, en caso contrario
    devuelve el valor falso }
operaciones
  acción insertar(e/s A:arbolBin; ent d:telemento)
  {Inserta en el árbol de búsqueda A el elemento d. Si
  el elemento ya está en el árbol la operación no produce
  ningún efecto}
  acción borrar(e/s A: arbolBin; ent d:telemento)
  {Borra del árbol de búsqueda A el elemento d}
  función está?(A: arbolBin; d:telemento) devuelve booleano
  {Devuelve el valor verdad si el elemento d está en el
  árbol de búsqueda A y falso en caso contrario}
```

2.- IMPLEMENTACIÓN DINÁMICA:

Consideramos la implementación dinámica vista para el TAD Árbol Binario y la completamos con la siguiente:

```
acción insertar(e/s A:arbolBin; ent d:telemento);
{Inserta en el árbol de búsqueda A el elemento d. Si
el elemento ya está en el árbol la operación no produce
ningún efecto}
variables
  iz,de : arbolBin
principio
  si arbolVacio(A) entonces {se realiza la inserción}
      iniciarArbol(iz)
      iniciarArbol(de)
      enraizar(A, iz, de, d)
  si_no
    si esMenor(d,raiz(A)) entonces
        iz ← dest(A).izdo
        insertar(iz,d)
    si_no
      si esMenor(raiz(A),d) entonces
        de ← dest(A).dcho
        insertar(de,d)
      fsi {Si ya existe no se hace nada}
    fsi
  fsi
fin
```

```
acción borrar(e/s A:arbolBin; ent d:telemento)
  {Borra del árbol de búsqueda A el elemento d}
  variables
    aux:puntero a Celda;
  principio
    si not arbolVacio(A) entonces
      si esMenor(d,raiz(A)) entonces
          aux ← dest(A).izdo
          borrar(aux,d)
      si_no
        si esMenor(raiz(A),d) entonces
          aux ← dest(A).dcho
          borrar(aux,d)
        si_no // esIgual(d,raiz(A)
          si arbolVacio(izquierdo(A)) entonces
          {Lo reemplaza por el hijo derecho}
            aux ←A
            A ← dest(A).dcho
            liberar(aux)
          si_no {Lo reemplaza por el máximo del subárbol
                izquierdo}
            dest(A).dato ← maximo(izquierdo(A))
            aux ← dest(A).izdo
            borrar(aux, dest(A).dato)
          fsi
        fsi
      fsi
    fsi
  fin
      función maximo(A:arbolbin) devuelve telemento
      {Devuelve el máximo valor de los elementos del árbol de
      búsqueda A que debe ser no vacío }

      principio
        si arbolVacio(derecho(A)) entonces
          devuelve(raiz(A))
        si_no
          devuelve(maximo(derecho(A)))
        fsi
      fin { De la función max }
```

```
función esta?(A:arbolBin; d:telemento) devuelve booleano
  {Devuelve el valor verdad si el elemento d está en el
árbol de búsqueda A y falso en caso contrario}

principio
  si arbolVacio(A) entonces
    devuelve(falso)
  si_no
    si esIgual(d,raiz(A)) entonces
        devuelve(verdad)
    si_no
      si esMenor(d,raiz(A)) entonces
        devuelve(esta?(izquierdo(A),d))
      si_no
        devuelve(esta?(derecho(A),d))
      fsi
    fsi
  fsi
fin
```