Árbol binario de búsqueda

17/4/98

Especificación ArbolBin[TipoEle]

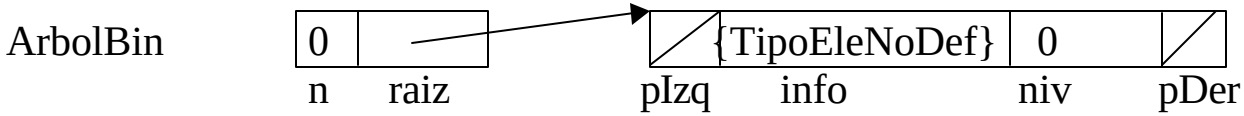
1	**Sintáctica:** creaArbolBin()→ ArbolBin, insArbolBin(ArbolBin,TipoEle)→ ArbolBin, eliArbolBin(ArbolBin, TipoEle)→ ArbolBin, busArbolBin(ArbolBin, TipoEle)→ Lógico, preorden(ArbolBin)→ ArbolBin, enorden(ArbolBin)→ ArbolBin, posorden(ArbolBin)→ ArbolBin, altura(ArbolBin) → Entero+, vacíoArbolBin(ArbolBin)→ Lógico, destruyeArbolBin(ArbolBin)→ .	-**creaArbolBin()**: Crea un árbol binario de búsqueda vacío. -**insArbolBin()**: Ingresa un nuevo elemento al árbol binario en la posición que le corresponde según el orden. -**eliArbolBin()**: Elimina el elemento en el árbol binario, si existe. Si está vacío no hace ninguna eliminación. -**destruyeArbolBin()**: Destruye el árbol binario. -**vacíoArbolBin()**: Regresa Verdadero si el árbol binario está vacío. -**busArbolBin()**: Devuelve Verdadero si el elemento está en el árbol binario, de lo contrario regresa Falso. -**preorden():** Despliega los elementos según el orden raíz, izquierdo y derecho. -**posorden():** Despliega los elementos del árbol binario de búsqueda ordenados ascendentemente. -**enorden():** Despliega los elementos según el orden izquierdo, derecho y raíz. -**altura():** Regresa la altura actual.
2	**Declaraciones:** TipoEle: e, {TipoEleNoDef}	
3	**Semántica:** vacíoArbolBin(creaArbolBin()) = Verdadero vacíoArbolBin(insArbolBin(creaArbolBin(), e)) = Falso busArbolBin(creaArbolBin(), e) = Falso busArbolBin(insArbolBin(creaArbolBin(), e), e)=Cierto eliArbolBin(creaArbolBin()) = creaArbolBin() altura(creaArbolBin()) = 0	

Especificación del TAD ArbolBin en TDSO.

ArbolBin[TipoEle] Clases: Entero, Lógico, ApuntadorA, TipoEle, NodoAB[TipoEle]		
1	**Estructura**: privado: n : Entero+ = 0 raiz : ApuntadorA NodoAB[TipoEle] = Nulo	-**n**: Número actual de elementos. -**raiz**: Apuntador al nodo raíz.
2	**Operaciones:** público: ArbolBin() ~ArbolBin() insArbolBin(TipoEle: e) eliArbolBin(TipoEle: e) busArbolBin(TipoEle: e): Lógico vacíoArbolBin(): Lógico numEle(): Entero+ altura(): Entero preorden() enorden() posorden() despliegue() **privado:** limpiarArbolBin(ApuntadorA NodoAB[TipoEle]: pa) alturaArbolBin(Entero+: c, ApuntadorA NodoAB[TipoEle]: pa): Entero enordenArbolBin(ApuntadorA NodoAB[TipoEle]: pa) preordenArbolBin(ApuntadorA NodoAB[TipoEle]: pa) posordenArbolBin(ApuntadorA NodoAB[TipoEle]: pa)	-**ArbolBin()**. *Constructor*. Crea un árbol vacío. -**~ArbolBin().** *Destructor.* Destruye el árbol . -**insArbolBin()**. *Transformador*. Ingresa un nuevo elemento en la posición que le corresponde según el recorrido enorden. -**eliArbolBin()**. *Transformador*. Elimina el elemento, si el árbol binario de búsqueda no está vacío. -**busArbolBin()**. *Observador*. Regresa Verdadero si el elemento existe. -**vacíoArbolBin()**. *Observador*. Regresa Verdadero si el árbol binario está vacío. -**numEle()**. *Observador*. Regresa el número actual de elementos en la estructura. -**altura()**. *Observador*. Regresa la altura actual de la estructura. -**preorden()**. *Observador*. Despliega los elementos del árbol binario de búsqueda en preorden (RID). -**enorden().** *Observador*. Despliega los elementos del árbol binario de búsqueda en enorden (IRD). -**posorden().** *Observador.* Despliega los elementos del árbol binario de búsqueda en posorden (IDR). -**despliegue().** *Observador*. Despliega el contenido del árbol. -**limpiarArbolBin().** *Transformador.* Devuelve todos los nodos. -**alturaArbolBin().** *Observador.* Devuelve la altura actual del árbol. -**enordenArbolBin().** *Observador.* Recorrido recursivo del árbol. -**preordenArbolBin().** *Observador.* Recorrido recursivo del árbol. -**posordenArbolBin().** *Observador.* Recorrido recursivo del árbol.

NodoAB[TipoEle]		
1	**Estructura**: privado: pIzq : ApuntadorA NodoAB[TipoEle]=Nulo info: TipoEle = {TipoEleNoDef} niv: Entero+ = 0 pDer : ApuntadorA NodoAB[TipoEle]= Nulo	-**pIzq:** Dirección del hijo izquierdo. -i**nfo**: Información del nodo. -**niv:** Nivel del nodo en el árbol binario de búsqueda. -**pDer**: Dirección del hijo derecho.
2	**Operaciones:** público: NodoAB() NodoAB(TipoEle: in, Entero+: ni) NodoAB(TipoEle: in) PIzq(ApuntadorA NodoAB[TipoEle]: izq) PIzq() : ApuntadorA NodoAB[TipoEle] Info(TipoEle: e) Info(): TipoEle Niv(): Entero+ Niv(Entero+: ni) PDer(ApuntadorA NodoAB[TipoEle]: der) PDer(): ApuntadorA NodoAB[TipoEle]	-**NodoAB()**. *Constructores*. Crea un nodo con su información definida y dos apuntadores nulos. -**PIzq().** *Observador y transformador*. Manejo del apuntador al hijo izquierdo. -**Info()**. *Observador y transformador*. Manejo de la información del nodo. -**Niv()**. *Observador y transformador*. Manejo del nivel del nodo en el árbol. -**PDer()**. *Observador y transformador*. Manejo del apuntador al hijo derecho.

Implementación de la clase NodoAB.



27/6/98	ArbolBin() {pos: n = 0 ∧ raiz = Nulo }	
1	n, raiz = 0, Nulo	**-n, raiz.** Definidos en ArbolBin.
1	ArbolBin[Entero] a ⇒ a.n = 0, a.raiz = Nulo	Crea la variable a de este tipo.

Constructor vacío. O(1).

27/6/98	~ArbolBin() {pre: n ≥ 0} {pos: n = 0 ∧ raiz = Nulo }	

1	Si (raiz ≠ Nulo) entonces
	  limpiarArbolBin(raiz)
	  n, raiz = 0, Nulo
	fsi
		-n, raiz, limpiarArbolBin(). Definidos en ArbolBin.

Destruye el árbol binario. O(n).

27/6/98	limpiarArbolBin(ApuntadorA NodoAB[TipoEle]: pa) {pre: n ≥ 0 ∧ pa ≠ Nulo } {pos: n = 0 ∧ raiz = Nulo }	
1	Si (pa → PIzq() ≠ Nulo) entonces limpiarArbolBin(pa → PIzq()) fsi	**-limpiarArbolBin().** Definidos en ArbolBin. **-PIzq(), PDer().** Definidos en NodoAB[TipoEle]. **-r.** ApuntadorA NodoAB[TipoEle]. Variable auxiliar.
2	Si (pa → PDer() ≠ Nulo) entonces limpiarArbolBin(pa → PDer()) fsi	
3	r = pa	
4	DevuelvaDir r	

Devuelve todos los nodos del árbol binario. O(n).

27/6/98

insArbolBin(TipoEle: e)

{pre: e ≠ {TipoEleNoDef} ∧ n ≥ 0} {pos: n ≥ 0}

1
2
	NodoAB[TipoEle] nn(e)
	Si (raiz = Nulo) entonces
	nn.Niv(0)
	raiz, n = DirecciónDe nn, 1
	sino
	pAux, enc = raiz, Falso
	(¬ enc) [Si (e < pAux → Info()) entonces
	Si (pAux→ PIzq( ) ≠ Nulo) entonces // Rama izq.
	pAux = pAux → PIzq( )
	sino
	enc = Verdadero
	nn.Niv(pAux → Niv() + 1)
	pAux→ PIzq(DireccionDe nn)
	fsi
	sino
	Si (pAux→ PDer( ) ≠ Nulo) entonces // Rama der.
	pAux = pAux → PDer( )
	sino
	enc = Verdadero
	nn.Niv(pAux → Niv() + 1)
	pAux→ PDer(DireccionDe nn)
	fsi
	fsi]
	n = n + 1
	fsi
		-pAux: ApuntadorA NodoAB[TipoEle]. Variable auxiliar.
		-Info(), PIzq(), PDer(), NodoAB[TipoEle]**:** Definidos en la clase NodoAB[TipoEle].
		-n, raiz: Definidos en ArbolBin.
		-nn: NodoAB[TipoEle]**:** Nodo que se insertará en el árbol.
		-DireccionDe. Regresa la dirección de memoria de la variable.
		-enc: Lógico. Variable auxiliar para detectar que ya se encontró el punto de inserción del nuevo nodo (nn) en el árbol.

1	a = (0, /), e = 78 ⇒ a = (1, →) (/, 78, 0, /)	Ingresa 78 al ArbolBin vacío.
2	b = (2, →)(/, 78,0,→)(/, 85,1, /), e = 20 ⇒ b = (3, →)(←, 78,0, →)(/, 20,1, /)(/, 85,1, /)	Ingresa 20 como hijo izquierdo de la raíz.

Inserta un nuevo elemento en el ArbolBin. O(lg(n)).

27/6/98 vacíoArbolBin(): Lógico		
{pre: n ≥ 0}		{pos: n ≥ 0}
1	Regresa (n = 0)	**-n.** Definido en ArbolBin.
1	a = (2, →) (/, 34, 0, →) (/, 40, 1, /) ⇒ Falso	No está vacío.
2	b = (0, /) ⇒ Verdadero	La variable b está vacía.

Verifica si el ArbolBin está vacío. O(1).

27/6/98 numEle(): Entero+		
{pre: n ≥ 0}		{pos: n ≥ 0}
1	Regresa n	**-n.** Definido en ArbolBin.
1	a = (2, →)(/, 38, 0, →)(/,42, 1, /) ⇒ 2	Hay dos nodos en el ArbolBin.
2	b = (0, /) ⇒ 0	La variable b tiene 0 elementos.

Devuelve el número de elementos actuales en el árbol binario. O(1).

27/6/98 altura(): Entero		
{pre: n ≥ 0}		{pos: n ≥ 0}
1	Si (raiz ≠ Nulo) entonces Regrese alturaArbolBin(raiz→Niv(), raiz) sino regrese –1 fsi	**-raiz, alturaArbolBin().** Definidos en ArbolBin. **-Niv().** Definido en NodoAB[TipoEle].
1	a = (2, →)(/, 38, 0, →)(/,42, 1, /) ⇒ 1	Altura del árbol = 1

Devuelve la altura actual del árbol binario. O(n).

27/6/98		
	alturaArbolBin(Entero+: c, ApuntadorA NodoAB[TipoEle]: pa): Entero	
{pre: n ≥ 0 ∧ pa ≠ Nulo }		{pos: n ≥ 0}

1
2
 	
3
 	
4
	c = Max(c, pa →Niv())
	Si (pa → PIzq() ≠ Nulo) entonces
	  c = alturaArbolBin(c, pa → PIzq())
	fsi
	Si (pa → PDer() ≠ Nulo) entonces
	  c = alturaArbolBin(c, pa → PDer())
	fsi
	regrese c
		-raiz, alturaArbolBin(). Definidos en ArbolBin.
		-Niv(), PIzq(), PDer(). Definidos en NodoAB[TipoEle].
		-c: Entero. Valor de regreso con la altura actual.

1	a = (2, →)(/, 38, 0, →)(/,42, 1, /) ⇒ 1	Altura del árbol = 1

Devuelve la altura actual del árbol binario.

27/6/98		
	preorden()	
{pre: n ≥ 0}		{pos: n ≥ 0 }
1	Si (raiz ≠ Nulo) entonces preordenArbolBin(raiz) fsi	**-raiz, preordenArbolBin().** Definidos en la clase ArbolBin.

Recorre en preorden los nodos del árbol binario. O(n).

27/6/98 enorden()		
{pre: n ≥ 0}		{pos: n ≥ 0 }
1	Si (raiz ≠ Nulo) entonces enordenArbolBin(raiz) fsi	**-raiz, enordenArbolBin().** Definidos en la clase ArbolBin.

Recorre enorden los nodos del árbol binario. O(n).

27/6/98 posorden()		
{pre: n ≥ 0}		{pos: n ≥ 0 }
1	Si (raiz ≠ Nulo) entonces posordenArbolBin(raiz) fsi	**-raiz, posordenArbolBin().** Definidos en la clase ArbolBin.

Recorre en posorden los nodos del árbol binario. O(n).

27/6/98 preordenArbolBin(ApuntadorA NodoAB[TipoEle]: pa)		
{pre: n ≥ 0 ∧ pa ≠ Nulo }		{pos: n ≥ 0}

1
2
 	
3
	Despliegue pa → Info()
	Si (pa → PIzq() ≠ Nulo) entonces
	  preordenArbolBin(pa → PIzq())
	fsi
	Si (pa → PDer()≠ Nulo) entonces
	  preordenArbolBin(pa → PDer())
	fsi
		-preordenArbolBin(). Definido en ArbolBin.
		-Info(), PIzq(), PDer(). Definidos en NodoAB[TipoEle].

Recorre recursivamente el ArbolBin en preorden.

27/6/98

enordenArbolBin(ApuntadorA NodoAB[TipoEle]: pa)

{pre: n ≥ 0 ∧ pa ≠ Nulo } {pos: n ≥ 0}

2
1
 	
3
	Despliegue pa → Info()
	Si (pa → PIzq() ≠ Nulo) entonces
	  enordenArbolBin(pa → PIzq())
	fsi
	Si (pa → PDer() ≠ Nulo) entonces
	  enordenArbolBin(pa → PDer())
	fsi
		-enordenArbolBin(). Definido en ArbolBin.
		-Info(), PIzq(), PDer(). Definidos en NodoAB[TipoEle].

Recorre recursivamente el ArbolBin enorden.

27/6/98

posordenArbolBin(ApuntadorA NodoAB[TipoEle]: pa)

{pre: n ≥ 0 ∧ pa ≠ Nulo } {pos: n ≥ 0}

3
1
 	
2
	Despliegue pa → Info()
	Si (pa → PIzq() ≠ Nulo) entonces
	  posordenArbolBin(pa → PIzq())
	fsi
	Si (pa → PDer() ≠ Nulo) entonces
	  posordenArbolBin(pa → PDer())
	fsi
		-posordenArbolBin(). Definido en ArbolBin.
		-Info(), PIzq(), PDer(). Definidos en NodoAB[TipoEle].

Recorre recursivamente el ArbolBin en posorden.

27/6/98

busArbolBin(TipoEle: e): Lógico

{pre: n ≥ 0} {pos: n ≥ 0}

1	Si (raiz = Nulo) entonces
	  regrese Falso
	sino
	  pAux, enc = raiz, Falso
	  (¬ enc) [Si (e < pAux → Info()) entonces
	  Si (pAux → PIzq() = Nulo) entonces
	  regrese enc
	  sino
	  pAux = pAux → PIzq()
	  fsi
	  sino
	  Si (e > pAux → Info()) entonces
	  Si (pAux → PDer() = Nulo) entonces
	  regrese enc
	  sino
	  pAux = pAux → PDer()
	  fsi
	  sino
	  regrese Verdadero
	  fsi
	  fsi]
	fsi
		-Info(), PIzq(), PDer(): Definidos en la clase NodoAB[TipoEle].
		-**raiz**: Definido en ArbolBin.
		-pAux: ApuntadorA NodoAB[TipoEle]. Variable auxiliar.
		-enc: Lógico. Variable auxiliar para controlar la búsqueda.

1	b = (0, /) ⇒ Falso	No está el elemento en el árbol
2	a = (2, →)(/,78, 0, →) (/,80, 1, /), e = 80 ⇒ Verdadero	80 está en el árbol

Devuelve el contenido del nodo del ArbolBin que contiene el elemento buscado. O(lg(n)).

27/6/98

eliArbolBin (TipoEle: e)

{pre: n ≥ 0} {pos: n ≥ 0}

```
1  Si (raiz ≠ Nulo ) entonces
     pAux, enc, pp = raiz, Falso, Nulo
     (¬ enc) [ Si (e < pAux → Info( ) ) entonces   / búsqueda del elemento
                 Si (pAux → PIzq( ) ≠ Nulo ) entonces
                    pp, pAux = pAux, pAux → PIzq( )
                 sino
                    enc = Verdadero
                 fsi
               sino  Si (e > pAux → Info( ) ) entonces
                           Si (pAux → PDer( ) ≠ Nulo ) entonces
                              pp, pAux = pAux, pAux → PDer( )
                           sino
                             enc = Verdadero
                           fsi
                         fsi
               fsi    ]
     Si (pAux → Info( ) = e ) entonces          / Lo encontró
        Si (pAux → PIzq( ) = Nulo ∧ pAux → PDer( ) = Nulo ) entonces
          Si (pp = Nulo) entonces              / Es una hoja y es la raíz
            raiz = Nulo                        / Vaciado del árbol
          sino  Si (pAux = pp → PIzq( ) ) entonces
                   pp → PIzq( Nulo )
                sino
```

-pAux, pp, pw: ApuntadorA NodoAB[TipoEle]. Variables auxiliares. Indica el nodo actual, el padre del nodo actual y el nodo más a la izquierda, respectivamente.
-Info(), PIzq(), PDer(): Definidos en la clase NodoAB[TipoEle].
-n, raiz: Definidos en ArbolBin.
-devuelvaDir: Regresa la dirección a la memoria disponible.
-enc: Lógico. Es Verdadero si se llega al final de una rama.

```
                pp → PDer( Nulo )
              fsi
          fsi
      sino Si (pAux → PIzq( ) ≠ Nulo ∧ pAux → PDer( ) = Nulo ) entonces
             Si (pp = Nulo) entonces              / Solo hay subárbol izquierdo
               raiz = pAux → PIzq( )
             sino Si (pAux = pp → PIzq( ) ) entonces
                    pp → PIzq(pAux → PIzq( ) )
                  sino
                    pp → PDer(pAux → PIzq( ) )
                  fsi
             fsi
           sino Si (pAux → PIzq( ) = Nulo ∧ pAux → PDer( ) ≠ Nulo ) entonces
                  Si (pp = Nulo) entonces            / Solo hay subárbol derecho  /
                    raiz = pAux → PDer( )
                  sino Si (pAux = pp → PDer( ) ) entonces
                         pp → PDer(pAux → PDer( ) )
                       sino
                         pp → PIzq(pAux → PDer( ) )
                       fsi
                  fsi
                sino            / Hay subárbol izquierdo y derecho    /
                  pw, pp = pAux → PDer( ), pAux      / Buscar el más a la izq.  /
                  ( pw → PIzq( ) ≠ Nulo )  [ pp, pw = pw, pw → PIzq( ) ]
                  pAux → Info( pw→ Info( ) )     / Cambio de contenido /
                 Si (pp = pAux ) entonces     / Actualización de los apuntadores /
                    pp → PDer( pw → PDer( ))
```

	sino Si (pw → PDer( ) = Nulo) entonces pp → PIzq(Nulo) sino pp → PIzq(pw → PDer( )) fsi fsi pAux → Info(pw → Info()) pAux = pw fsi fsi fsi devuelvaDir pAux n = n – 1 sino Despliegue “Ese elemento no está en el árbol” fsi sino Despliegue “Arbol binario de búsqueda vacío” fsi	
1	b = (0, /), e = 85 ⇒ Arbol binario de búsqueda vacío	No elimina
2	a = (3, →) (/, 78, 0, →) (/,85, 1,→) (/,340, 1, /), e = 85 ⇒ a = (2, →) (/, 78, 0, →) (/, 340, 1, /)	Elimina 85.

Elimina un elemento del árbol binario de búsqueda. O(lg(n)).

27/6/98 despliegue ()		
{pre: n ≥ 0}		{pos: n ≥ 0}

1	Si (n = 0) entonces
	  Despliegue “El árbol binario de búsqueda está vacío”
	sino
	  Despliegue “Elementos del árbol binario de búsqueda”
	  enorden()
	fsi
		-n, enorden(). Definidos en la clase ArbolBin.

1	b = (0, /) ⇒ El árbol binario de búsqueda está vacío	No hay elementos en el ArbolBin.
2	a = (2, →) (←,45, 0, →) (/, 34, 1, /) (/, 48, 1, /) ⇒ Elementos del árbol binario de búsqueda | 34 | 45 | 48	Despliega el contenido del ArbolBin con sus comentarios

Despliega el contenido actual del árbol. O(n)