

### **Estructuras de Datos**





**Ejercicio 4.-** Extender la especificación de los árboles generales vista en clase con las siguientes operaciones:

- num nodos: árbol → natural, para calcular cuántos nodos hay en un árbol general;
- num hojas: árbol → natural, para ver la cantidad total de hojas que tiene un árbol general;
- max\_hijos: árbol → natural, que obtiene cuál es la mayor cantidad de hijos en un mismo nodo que hay en un árbol general.
- reflejar: árbol → árbol, que obtiene la imagen especular de un árbol;
- frontera: árbol → lista, que genera una lista formada por los elementos almacenados en las hojas del árbol, tomados de izquierda a derecha.

#### Solución Operaciones:

```
Apartado a)
num nodos: árbol→natural
func num nodos (a:árbol) dev n:natural
              n \leftarrow 1 + \text{num nodos b (hijos(a))}
finfunc
num nodos b: bosque → natural
func num nodos b (b:bosque) dev n:natural
var prim:arbol
       si vacio?(b) entonces n \leftarrow 0
              prim←primero(b)
       sino
                      n←
                             num nodos (prim)
                             + num nodos b(resto(b))
       fin si
finfunc
```



### **Estructuras de Datos**



Soluciones Ejercicios de Árboles binarios y generales

```
Apartado b)
num hojas: árbol→natural
func num hojas (a:árbol) dev n:natural
              si vacio?(bosque(a)) entonces n\leftarrow 1
              sino n\leftarrownum hojas b(hijos(a))
finfunc
num hojas b: bosque → natural
func num hojas b (b:bosque) dev n:natural
var prim:arbol
        si vacio?(b) entonces n \leftarrow 0
        sino
        prim←primero(b)
        n← num hojas(prim)+num hojas b(resto(b))
        fin si
finfunc
Apartado c)
máx hijos: árbol→natural
func máx hijos (a:árbol) dev n:natural
              num hijos, max hijos b:natural
var
              num hijos \leftarrow num hijos(a)
              max_hijos_b ← max_hijos_b(bosque(a))
              si (num hijos > max hijos b)
                     entonces n← num hijos
              sino n← máx hijos b
```



## Estructuras de Datos

Soluciones Ejercicios de Árboles binarios y generales

#### finsi

#### finfunc

```
máx hijos b: bosque→natural
```

```
func máx_hijos _b (b:bosque) dev n:natural

var prim:arbol num_hijos_p, max_hijos_r:natural

si vacio?(b) entonces max_hijos_b ← 0

sino prim ← primero(b)

num_hijos_p ← num_hijos(prim)

max_hijos_r ← max_hijos_b(resto(b))

finsi

si (num_hijos_p > max_hijos_r

entonces n ← num_hijos_p

sino n ← max_hijos_r

finsi
```

#### finfunc

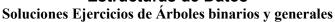
```
Apartado d)
reflejar: árbol→árbol

func reflejar (a:árbol) dev ar:árbol
ar←raiz(a)•reflejar_b(bosque(a))
```

#### finfunc



## Estructuras de Datos

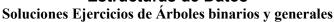




```
bosque imagen:bosque → bosque
proc bosque imagen (b, bimag:bosque)
{procedimiento auxiliar que va creando el bosque imagen de b en bimag}
var prim:árbol
       mientras !vacio(b) hacer
                                   prim←primero(b)
                                   b \leftarrow resto(b)
                                   bimag←reflejar(prim):bimag
       finmientras
       finsi
reflejar b:bosque→bosque
func reflejar b(b:bosque) dev br:bosque
       bimagen:bosque
var
       bimagen←[]
       bosque imagen(b, bimagen)
       {procedimiento auxiliar que va creando el bosque imagen de b}
       br←bimag
finfunc
Apartado d)
frontera: árbol→lista
func frontera(a:árbol) dev l:lista
       si num hijos(a) =0 entonces 1 \leftarrow [raiz(a)]
       sino l←frontera b(hijos(a))
finfunc
```



#### Estructuras de Datos





```
func frontera_b (b:bosque) dev l:lista
    si vacio?(b) entonces 1←[]
    sino 1←frontera(primero(b))++frontera_b(resto(b)
    finsi
```

#### finfunc

**Ejercicio 5.-**Llamaremos a un árbol general de naturales "creciente" en cada nivel del árbol, la cantidad de nodos que hay en ese nivel es igual al valor del nivel más uno; es decir, el nivel 0 tiene exactamente un nodo, el nivel 1 tiene exactamente dos nodos, el nivel 2 tiene exactamente 2 nodos. Se pide:

- Especificar completamente el TAD árbol general,
- Comprobar si un árbol general es "creciente",
- Buscar el nodo con mayor cantidad de hijos de un árbol creciente.

Necesitaremos una función auxiliar que cuente el total de nodos de un nivel dado k:

```
func nodos_nivel_k (a:árbol, k:natural) dev n:natural

si k=0 entonces n←1

sino n←nodos_nivel_k_b(hijos(a), k-1)

finsi

finfunc

func nodos_nivel_k_b(b:bosque, k:natural) dev n:natural

si vacio?(b) entonces n←0

sino si k=0 entonces n←long(b) {tamaño del bosque}

sino

n←nodos_nivel_k(primero(b), k) + nodos_nivel_k_b(resto(b), k)

finsi
```

finfunc



# **Estructuras de Datos Soluciones Ejercicios de Árboles binarios y generales**

```
func creciente (a:árbol) dev b:boolean
       creciente desde k(a,1)
finfunc
func creciente desde k (a: árbol,k:natural) dev b:boolean
       si nodos nivel k(a, k) = 0 entonces b \leftarrow T
       sino si nodos_nivel_k(a, k) !=k+1 entonces b\leftarrowF
              sino b←creciente_desde_k(a, k+1)
       finsi
finfunc
o tb. {versión iterativa}
func creciente (a:árbol) dev b:boolean
       k←0 es creciente←T
mientras nodos nivel k(a, k) !=0 \land es creciente hacer
       si nodos_nivel_k(a, k) !=k+1 entonces es_creciente ←F
       sino k←k+1
       finsi
finmientras
finfunc
```