



```
struct CeldaArbolGeneral {
    Tbase etiqueta;
    CeldaArbolGeneral *padre,
    *izquierdo,
    *derecho;
};
```

void insertarHijoIZqda (CeldaArbolGeneral *T1,
CeldaArbolGeneral *n,
CeldaArbolGeneral *T2)

{ if (T2 != 0)

{ T2 -> hermanoIzqda = n -> hermanoDcha;

T2 -> padre = n;

n -> hermanoDcha = T2;

T2 = 0;

}

}

CeldaArbolGeneral * PoderHijoIZqda (CeldaArbolGeneral *T,
CeldaArbolGeneral *n)

{

CeldaArbolGeneral * Res = 0;

if (n -> hermanoIzqda != 0)

{

Res = n -> hermanoIzqda;

n -> hermanoIzqda = Res -> hermanoDcha;

Res -> padre = Res -> hermanoDcha = 0;

}

return Res;

}

void InsertarHermanos (CeldArbolGeneral * Td,
CeldArbolGeneral * n,
CeldArbolGeneral * d T2)

{
if (T2 != 0) {
T2 -> hermanos = n -> hermanos;
T2 -> padre = n -> padre;
n -> hermanos = T2;
T2 = 0;
}
}

CeldArbolGeneral * PodarHermanos (CeldArbolGeneral * T,
CeldArbolGeneral * n)

{
CeldArbolGeneral * Res = 0;
if (n -> hermanos != 0)
{
Res = n -> hermanos;
n -> hermanos = Res -> hermanos;
Res -> padre = Res -> hermanos = 0;
}
return Res;
}

void ListarPreorden (CelulaArbolGeneral *T)

{

if (T != 0)

{

cout << T->etiqueta;

for (CelulaArbolGeneral *aux = T->izqda;

aux != 0; aux = aux->heredrcha)

ListarPreorden (aux);

}

}

void ListarPostorden (CelulaArbolGeneral *T)

{

if (T != 0)

{

for (CelulaArbolGeneral *aux = T->izqda;

aux != 0; aux = aux->heredrcha)

ListarPostorden (aux);

cout << T->etiqueta ;

}

}

CeldaArbolGeneral * CrearRaiz (Const Tabla & t)

{

CeldaArbolGeneral n = new CeldaArbolGeneral;

n → etiqueta = e;

n → padre = n → hermanoIzda = n → hermanoDcha = 0;

}

void Destruir (CeldaArbolGeneral * t)

{

if (t != 0)

{

CeldaArbolGeneral * n, *aux;

n = t → hermanoIzda;

while (n != 0)

{

aux = n;

n = n → hermanoDcha;

Destruir (aux);

}

delete t;

{

}

void destruir (CelulaArbolGeneral * t)

{

if (t != 0)

{

destruir (t -> hijoIzda);

destruir (t -> hermanoDcha);

delete t ;

}

}

int numeroNodos (CelulaArbolGeneral * T)

{

return (T == 0) ? 0 :

(1 + numeroNodos (T -> hijoIzda) +

numeroNodos (T -> hermanoDcha));

}

void ListarInorden (ArbolGeneral * T)

```
{  
    if (T != 0)  
    {  
        ListarInorden (left(T));  
        cout << T->etiqueta;  
        aux = aux -> hermanoDcha;  
    }  
    while (aux != 0)  
    {  
        ListarInorden (aux);  
        aux = aux -> hermanoDcha;  
    }  
}
```

void ListarNiveles (ColaArbolGeneral *T)

```
if (T != 0)
{
    Cola C;
    C.push(T);
    while (!C.vacia())
    {
        ColaArbolGeneral *aux;
        aux = C.front();
        C.pop();
        cout << aux->etiqueta;
        for (aux = aux->izquierda;
             aux != 0;
             aux = aux->hermano)
            C.push(aux);
    }
}
```

3