```
--|::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
--| MODULO DE DECLARACION DEL TAD 'conjunto generico'.
--| el tipo de los elementos ha de ser un tipo discreto
--|::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
--| Fecha: 31-X-96; corregido 16-10-03
--| Autor: Javier Campos; corregido por Elvira Mayordomo
--| FICHERO: conjuntos.ads


-- especec conjuntos
--   usa booleanos,naturales
--   parámetro formal
--    género elemento
--   fpf
--   género conjunto
--   operaciones
--    vacío: → conjunto
--    esvacío: conjunto → bool
--    añadir: elemento conjunto → conjunto
--    quitar: elemento conjunto → conjunto
--    _∈_: elemento conjunto → bool
--    _∪_, _∩_: conjunto conjunto → conjunto
```

```
--    cardinal: conjunto → nat
--   ecuaciones A,B:conjunto; e,f: elemento
--    añadir(e,añadir(f,A)) = añadir(f,añadir(e,A))
--    añadir(e,añadir(e,A)) = añadir(e,A)
--    esvacío(vacío) = verdad
--    esvacío(añadir(e,A)) = falso
--    quitar(e,vacío) = vacío
--    quitar(e,añadir(e,A)) = quitar(e,A)
--    e≠f → quitar(e,añadir(f,A)) = añadir(f,quitar(e,A))
--    e∈vacío = falso
--    e∈añadir(f,A) = (e=f) ∨ (e∈A)
--    A ∪ vacío = A
--    A ∪  añadir(e,B) = añadir(e,A ∪  B)
--    A ∩ vacío = vacío
--    A ∩ añadir(e,B) = (A ∩ añadir(e,vacío)) ∪ (A ∩B)
--    cardinal(vacío) = 0
--    cardinal(añadir(e,A)) = suc(cardinal(quitar(e,A)))
-- fespec

generic
     type elemento is (<>);
package conjuntos is
```

type conjunto **is private**;
-- coste en memoria: un conjunto de cualquier tamaño
-- ocupa O(número de datos de tipo elemento)
procedure vacio (A:**out** conjunto);
-- Post: A=vacío
-- coste en tiempo O(número de datos de tipo elemento)
function esVacio (A: conjunto) **return** boolean;
-- Post: esVacio(A)=esvacío(A)
-- coste en tiempo O(1)
procedure poner (e:**in** elemento; A:**in out** conjunto);
-- Pre: A=A0
-- Post: A=añadir(e,A0)
-- coste en tiempo O(1)
procedure quitar (e:**in** elemento; A:**in out** conjunto);
-- Pre: A=A0
-- Post: A=quitar(e,A0)
-- coste en tiempo O(1)
function pertenece (e: elemento; A: conjunto) **return** boolean;
-- Post: pertenece(e,A)= $e \in A$
-- coste en tiempo O(1)
procedure union (A,B:**in** conjunto; C:**out** conjunto);
-- Post: $C=A \cup B$

```
    -- coste en tiempo O(número de datos de tipo elemento)
    procedure interseccion (A,B:in conjunto; C:out conjunto);
    -- Post: C=A ∩ B
    -- coste en tiempo O(número de datos de tipo elemento)
    function cardinal (A: conjunto) return integer;
    -- Post: cardinal(A)=cardinal(A)
    -- coste en tiempo O(1)

private
    type elementos is array(elemento) of boolean;
    type conjunto is
        record
            elmto:elementos;
            card:integer;
        end record;

end conjuntos;
```