

1. Módulo Conjunto(α)

Interfaz

se explica con: CONJUNTO(α)

géneros: conjRapido(α)

Operaciones básicas de conjunto

VACIO() $\rightarrow res : conjRapido(\alpha)$

Pre $\equiv \{true\}$

Post $\equiv \{res =_{obs} \emptyset\}$

Complejidad: $\Theta(1)$

Descripción: Crea un nuevo conjunto vacio

AGREGAR(in $a : \alpha$ in/out $c : conjRapido(\alpha)$)

Pre $\equiv \{c =_{obs} c_0\}$

Post $\equiv \{c =_{obs} Ag(a, c_0)\}$

Complejidad: $\Theta(1)$

Descripción: Agrega el elemento a al conjunto c

VACIO?(in $c : conjRapido(\alpha)$) $\rightarrow res : bool$

Pre $\equiv \{true\}$

Post $\equiv \{res =_{obs} \emptyset?(c)\}$

Complejidad: $\Theta(1)$

Descripción: Devuelve *true* si el conjunto esta vacio

EN(in $a : \alpha$, in $c : conjRapido(\alpha)$) $\rightarrow res : bool$

Pre $\equiv \{true\}$

Post $\equiv \{res =_{obs} a \in c\}$

Complejidad: $O(N)$

Descripción: Devuelve *true* si el elemento a pertenece al conjunto c

CANTIDAD(in $c : conjRapido(\alpha)$) $\rightarrow res : nat$

Pre $\equiv \{true\}$

Post $\equiv \{res =_{obs} \#c\}$

Complejidad: $O(N^2)$

Descripción: Devuelve la cantidad de elementos definidos en c . En este conjunto se sacrifica la complejidad de *Cantidad* por *Agregar* siempre en $O(1)$

DAMEUNO(in $c : conjRapido(\alpha)$) $\rightarrow res : \alpha$

Pre $\equiv \{\neg \emptyset?(c)\}$

Post $\equiv \{res =_{obs} dameUno(c)\}$

Complejidad: $\Theta(1)$

Descripción: Devuelve un elemento del conjunto.

SINUNO(in/out $c : conjRapido(\alpha)$)

Pre $\equiv \{\neg \emptyset?(c) \wedge c =_{obs} c_0\}$

Post $\equiv \{c =_{obs} sinUno(c_0)\}$

Complejidad: $O(N)$

Descripción: Devuelve el conjunto sin el elemento que devuelve DameUno(c).

BORRAR(in $a : \alpha$, in/out $c : conjRapido(\alpha)$)

Pre $\equiv \{a \in c \wedge c =_{obs} c_0\}$

Post $\equiv \{c =_{obs} c_0 - \{a\}\}$

Complejidad: $O(N)$

Descripción: Devuelve el conjunto sin el elemento a .

Representación

Representacion del Conjunto

$\text{conjRapido}(\alpha)$ se representa con $\text{lista}(\alpha)$

Invariante de representacion en castellano:

Cualquier lista de elementos de tipo α es un conjunto valido de tipo α

$\text{Rep} : \text{lista}(\alpha) \longrightarrow \text{bool}$

$\text{Rep}(l) \equiv \text{true} \iff \text{true}$

$\text{Abs} : \text{lista}(\alpha) \ c \longrightarrow \text{conjRapido}(\alpha)$

$\{\text{Rep}(c)\}$

$\text{Abs}(c) \equiv \text{con} : \text{conj}(\alpha) /$

$(\forall a : \alpha) \ a \in \text{con} \iff \text{En}(a, c)$

Algoritmos

Algoritmos de Agentes

Lista de algoritmos

1.	Vacio	2
2.	Agregar	2
3.	Vacio?	2
4.	En	3
5.	DameUno	3
6.	SinUno	3
7.	Cantidad	3
8.	Borrar	4

$i\text{Vacio}() \rightarrow \text{res} : \text{lista}(\alpha)$

begin

| $\text{res} \leftarrow \text{Vacía}()$

//O(1)

end

Complejidad: O(1)

Algoritmo 1: Vacio

$i\text{Agregar}(\text{in } a : \alpha, \text{in/out } c : \text{lista}(\alpha))$

begin

| $\text{AgregarAtras}(c, a)$

//O(1)

end

Complejidad: O(1)

Algoritmo 2: Agregar

$i\text{Vacio?}(\text{in } c : \text{lista}(\alpha)) \rightarrow \text{res} : \text{bool}$

begin

| $\text{Vacía?}(c)$

//O(1)

end

Complejidad: O(1)

Algoritmo 3: Vacio?

```

iEn(in a:  $\alpha$ , in c: lista( $\alpha$ ))  $\rightarrow$  res: bool
begin
  var
    iterador : itLista
    iterador  $\leftarrow$  CrearIt(l) // O(1)
    res  $\leftarrow$  false // O(1)
    while HaySiguiente(iterador) do // O(1)
      if Siguiente(iterador) == a then // O(1)
        res  $\leftarrow$  true // O(1)
      end
      Avanzar(iterador) // O(1)
    end
  end
// While: O(N)
end
Complejidad: O(N)

```

Algoritmo 4: En

```

iDameUno(in c: lista( $\alpha$ ))  $\rightarrow$  res:  $\alpha$ 
begin
  | Primero(c) // O(1)
end
Complejidad: O(1)

```

Algoritmo 5: DameUno

```

iSinUno(in/out c: lista( $\alpha$ ))
begin
  | Borrar(DameUno(c), c) // O(N)
end
Complejidad: O(N)

```

Algoritmo 6: SinUno

```

iCantidad(in c: lista( $\alpha$ ))  $\rightarrow$  res: nat
begin
  var
    iterador : itLista( $\alpha$ )
    copiaLista : lista( $\alpha$ )
    cantidad : nat
    cantidad  $\leftarrow$  0 // O(1)
    iterador  $\leftarrow$  CrearIt(c) // O(1)
    res  $\leftarrow$  false // O(1)
    copiaLista  $\leftarrow$  Copiar(c) // O(N)
    while HaySiguiente(iterador) do // O(1)
      Fin(copiaLista) // O(1)
      if  $\neg$ En(Siguiente(iterador), copiaLista) then // O(N)
        cantidad  $\leftarrow$  cantidad + 1 // O(1)
      end
      Avanzar(iterador) // O(1)
    end
  end
// While: O( $\sum_{i=1}^n n$ )
end
Complejidad: O( $N^2$ )
Comentarios: O( $\sum_{i=1}^n n$ ) = O( $\frac{1}{2}n(n+1)$ ) = O( $\frac{1}{2}n^2 + \frac{1}{2}n$ ) = O( $n^2$ )

```

Algoritmo 7: Cantidad

```

iBorrar(in a:  $\alpha$ , in/out c: lista( $\alpha$ ))
begin
  var
    iterador : itLista( $\alpha$ )
    iterador  $\leftarrow$  CrearIt(c) // O(1)
    while HaySiguiente(iterador) do // O(1)
      if Siguiente(iterador) == a then // O(1)
        | EliminarSiguiente(iterador) // O(1)
      else
        | Avanzar(iterador) // O(1)
      end
    end
  end
end
// While: O(N)
Complejidad: O(N)

```

Algoritmo 8: Borrar
