

1. Modulo HippiesYEstudiantes

Interfaz

se explica con: HIPPIESYESTUDIANTES

generos: hipyest

Operaciones basicas de Hippies y Estudiantes

NUEVOHIPPIESYESTUDIANTES() $\rightarrow res : \text{hipyest}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{nuevoHippiesYEstudiantes}()\}$

Complejidad: $\Theta(1)$

Descripción: Crea un nuevo contenedor de Hippies y Estudiantes vacio

DEFINIRHIPPIE(in n : nombre, in p : posicion, in/out he : hipyest)

Pre $\equiv \{he =_{\text{obs}} he_0\}$

Post $\equiv \{he =_{\text{obs}} \text{defHippie}(n, p, he_0)\}$

Complejidad: $O(n_m)$

Descripción: Agrega al hippie n a hippies y estudiantes he

DEFINIRESTUDIANTE(in n : nombre, in p : posicion, in/out he : hipyest)

Pre $\equiv \{he =_{\text{obs}} he_0\}$

Post $\equiv \{he =_{\text{obs}} \text{defEstudiante}(n, p, he_0)\}$

Complejidad: $O(n_m)$

Descripción: Agrega al hippie n a hippies y estudiantes he

POSHIPPIEYESTUDIANTE(in n : string, in he : hipyest) $\rightarrow res : \text{posicion}$

Pre $\equiv \{\text{esta?}(n, he)\}$

Post $\equiv \{res =_{\text{obs}} \text{posHippieYEstudiante}(n, he)\}$

Complejidad: $O(n_m)$

Descripción: Devuelve la posicion del hippie o estudiante n .

ESTA?(in n : string, in he : hipyest) $\rightarrow res : \text{bool}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{esta?}(n, he)\}$

Complejidad: $O(n_m)$

Descripción: Devuelve *true* si n esta en hippies o estudiantes de he .

ESHIPPIE?(in n : string, in he : hipyest) $\rightarrow res : \text{bool}$

Pre $\equiv \{\text{esta?}(n, he)\}$

Post $\equiv \{res =_{\text{obs}} \text{esHip?}(n, he)\}$

Complejidad: $O(n_m)$

Descripción: Devuelve *true* si n esta en hippies de he .

ESEST?(in n : string, in he : hipyest) $\rightarrow res : \text{bool}$

Pre $\equiv \{\text{esta?}(n, he)\}$

Post $\equiv \{res =_{\text{obs}} \neg \text{esHip?}(n, he)\}$

Complejidad: $O(n_m)$

Descripción: Devuelve *true* si n esta en estudiantes de he .

ESTUDIANTES(in he : hipyest) $\rightarrow res : \text{conj}(\text{nombre})$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{\text{alias}(res =_{\text{obs}} \text{estudiantes?}(he))\}$

Complejidad: $O(1)$

Descripción: Devuelve el conjunto de estudiantes.

Aliasing: res no es modificable. Si se agregan hippies o estudiantes, res queda invalidado.

HIPPIES(in he : hipyest) $\rightarrow res : \text{conj}(\text{nombre})$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{\text{alias}(\text{res} =_{\text{obs}} \text{hippies?}(he))\}$

Complejidad: $O(1)$

Descripción: Devuelve el conjunto de hippies.

Aliasing: *res* no es modificable. Si se agregan hippies o estudiantes, *res* queda invalidado.

BORRAR(in *n* : nombre, in/out *he* : hipiest)

Pre $\equiv \{he =_{\text{obs}} he_0\}$

Post $\equiv \{he =_{\text{obs}} \text{borrar}(n, p, he_0)\}$

Complejidad: $O(n_m)$

Descripción: Agrega al hippie *n* a hippies y estudiantes *he*

Representación

Representacion del Conjunto

hipiest) se representa con estr

donde estr es tupla(*estudiantes*: DiccClavesRapidas(nombre, posicion) , *hippies*:
DiccClavesRapidas(nombre, posicion))

Invariante de representacion en castellano:

1. No hay ningun nombre en *estudiantes* que tambien este en *hippies*

$\text{Rep} : \text{estr} \rightarrow \text{bool}$

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

1. $(\forall n : \text{nombre})$
 $(\text{Def?}(n, e.\text{estudiantes}) \Rightarrow \neg \text{Def?}(n, e.\text{hippies})) \wedge$
 $(\text{Def?}(n, e.\text{hippies}) \Rightarrow \neg \text{Def?}(n, e.\text{estudiantes}))$

$\text{Abs} : \text{estr } e \rightarrow \text{HippiesYEstudiantes}$

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

$\text{Abs}(e) \equiv he : \text{HippiesYEstudiantes} /$

$(\forall n : \text{nombre})$

$(\text{Def?}(n, e.\text{estudiantes}) \iff (\text{def?}(n, \text{diccPos}(he)) \wedge_L \neg \text{esHip}(n, he))) \wedge$

$(\text{Def?}(n, e.\text{hippies}) \iff (\text{def?}(n, \text{diccPos}(he)) \wedge_L \text{esHip}(n, he))) \wedge_L$

$(\text{Def?}(n, e.\text{estudiantes}) \Rightarrow_L \text{Obtener}(n, e.\text{estudiantes}) =_{\text{obs}} \text{obtener}(n, \text{diccPos}(he))) \wedge$

$(\text{Def?}(n, e.\text{hippies}) \Rightarrow_L \text{Obtener}(n, e.\text{hippies}) =_{\text{obs}} \text{obtener}(n, \text{diccPos}(he)))$

Algoritmos

Algoritmos de Agentes

Lista de algoritmos

1.	NuevoHippieYEst	3
2.	PosHippieYEstudiante	3
3.	DefinirHippie	3
4.	DefinirEst	3
5.	Esta?	3
6.	EsEst?	3
7.	EsHip?	4
8.	Estudiantes	4
9.	Hippies	4
10.	Borrar	4

iNuevoHippieYEst() \rightarrow res: estr

begin

 res.hippies \leftarrow Vacio()

// $O(1)$

 res.estudiantes \leftarrow Vacio()

// $O(1)$

end

Complejidad: $O(1)$

Algoritmo 1: NuevoHippieYEst

iPosHippieYEstudiante(in n: nombre, in he: estr) \rightarrow res: posicion

begin

if *EsEst?*(n, he) **then**

// $O(n_m)$

 res \leftarrow Obtener(n, he.estudiantes)

// $O(n_m)$

else

 res \leftarrow Obtener(n, he.hippies)

// $O(n_m)$

end

end

Complejidad: $O(n_m)$

Algoritmo 2: PosHippieYEstudiante

iDefinirHippie(in n: nombre, in p: posicion, in/out he: estr)

begin

 Definir(n, p, he.hippies)

// $O(n_m)$

end

Complejidad: $O(n_m)$

Algoritmo 3: DefinirHippie

iDefinirEst(in n: nombre, in p: posicion, in/out he: estr)

begin

 Definir(n, p, he.estudiantes)

// $O(n_m)$

end

Complejidad: $O(n_m)$

Algoritmo 4: DefinirEst

iEsta?(in n: nombre, in he: estr) \rightarrow res: bool **begin**

 res \leftarrow Def?(n, he.hippies) \vee Def?(n, he.estudiantes)

// $O(2n_m)$

end

Complejidad: $O(n_m)$

Algoritmo 5: Esta?

iEsEst?(in n: nombre, in he: estr) \rightarrow res: bool

begin

 res \leftarrow Def?(n, he.estudiantes)

// $O(n_m)$

end

Complejidad: $O(n_m)$

Algoritmo 6: EsEst?

```

iEsHip?(in  $n$ : nombre, in  $he$ : estr)  $\rightarrow$  res: bool
begin
|   res  $\leftarrow$  Def?( $n$ , he.hippies)
end
Complejidad:  $O(n_m)$ 

```

Algoritmo 7: EsHip?

```

iEstudiantes(in  $he$ : estr)  $\rightarrow$  res: puntero(conj(nombre))
begin
|   res  $\leftarrow$  Claves(he.estudiantes)
end
Complejidad:  $O(1)$ 

```

Algoritmo 8: Estudiantes

```

iHippies(in  $he$ : estr)  $\rightarrow$  res: puntero(conj(nombre))
begin
|   res  $\leftarrow$  Claves(he.hippies)
end
Complejidad:  $O(1)$ 

```

Algoritmo 9: Hippies

```

iBorrar(in  $n$ : nombre, in/out  $he$ : estr)
begin
|   if EsEst?( $n$ ,  $he$ ) then
|       res  $\leftarrow$  Borrar( $n$ , he.estudiantes)
|   else
|       res  $\leftarrow$  Borrar( $n$ , he.hippies)
|   end
end
Complejidad:  $O(N)$ 

```

Algoritmo 10: Borrar
