

```
*****
*****                                     *****
*****                                     *****
*****                                     *****
```

Se explica con: Mapa
Géneros: Mapa

```
*****
*****                                     *****
*****                                     *****
*****                                     *****
```

Crear() → res: Mapa
Pre ≡ {true}
Post ≡ {res =obs vacío}
Complejidad: O(1)
Descripcion: Crea un mapa vacío

Agregar(in/out m : Mapa, in e : estacion)
Pre ≡ {e ∉ estaciones(m) ∧ m₀ =obs m}
Post ≡ {m =obs agregar(e, m₀)}
Complejidad: O(#estaciones(m) + long(e))
Descripcion: Agrega una estación al mapa

Conectar(in/out m : Mapa, in e1 : estacion, in e2 : estacion, in r : restriccion)
Pre ≡ {e1, e2 ∈ estaciones(m) ∧ ¬conectadas?(m, e1, e2) ∧ m₀ =obs m}
Post ≡ {m =obs conectar(e1, e2, r, m)}
Complejidad: O(long(e1) + long(e2) + S)
Descripcion: Conecta 2 estaciones del mapa entre con una restriccion para ese vinculo.

Estaciones(in m : Mapa) → res : itConj(string)
Pre ≡ {true}
Post ≡ {alias(esPermutación?(SecuSuby(res), estaciones(m))) ∧ vacia?(Anteriores(res))}
Complejidad: O(1)
Descripcion: Devuelve un iterador al conjunto de las estaciones del mapa.
Aliasing: No se pueden modificar los contenidos del iterador.

Conectadas?(in m : Mapa, in e1 : estacion, in e2 : estacion) → res : bool
Pre ≡ {e1, e2 ∈ estaciones(m)}
Post ≡ {res =obs conectadas?(m, e1, e2)}
Complejidad: O(long(e1) + long(e2))
Descripcion: Indica si 2 estaciones estan conectadas.

idSenda(in m : Mapa, in e1 : estacion, in e2 : estacion) → res : nat
Pre ≡ {e1, e2 ∈ estaciones(m) ∧ conectadas?(m, e1, e2)}
Post ≡ {res es el id de la senda en nuestra representación}
Complejidad: O(long(e1) + long(e2))
Descripcion: Devuelve el id de la senda entre e1 y e2.

Sendas(in m : Mapa) → res : itVectorPointer(restriccion)
Pre ≡ {true}
Post ≡ {res itera sobre las restricciones de todas las sendas del mapa}
Complejidad: O(1)
Descripcion: Devuelve las restricciones de las sendas del mapa

Copiar(in m : Mapa) → res : Mapa
Pre ≡ {true}
Post ≡ {res =obs m}
Complejidad: O(Copiar(m.sendas) + Copiar(m.estaciones) + Copiar(m.conexiones))
Descripcion: Devuelve una copia del mapa.
Aliasing: El nuevo mapa comparte las restricciones ya agregadas al mapa original.

```

*****
*****                                Representación                                *****
*****

```

Mapa se representa con estr,
donde estr es: tupla(

```

    sendas      : VectorPointer(Restriccion),
    conexiones  : DiccString(DiccString(Int)),
    estaciones  : conj(string)
)

```

Rep: $\wedge(\text{estr}) \rightarrow \text{boolean}$

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

claves(e.conexiones) \subset e.estaciones \wedge

(($\forall e_1 : \text{string}$)

$e_1 \in \text{claves}(e.\text{conexiones}) \wedge$

claves(obtener(e_1 , e.conexiones)) \subset e.estaciones \wedge

(($\forall e_2 : \text{string}$)

($e_1 \in \text{claves}(e.\text{conexiones}) \wedge e_2 \in \text{claves}(\text{obtener}(e_1, e.\text{conexiones})) \iff$

$e_2 \in \text{claves}(e.\text{conexiones}) \wedge e_1 \in \text{claves}(\text{obtener}(e_1, e.\text{conexiones})) \wedge$

($e_1 \in \text{claves}(e.\text{conexiones}) \wedge e_2 \in \text{claves}(\text{obtener}(e_1, e.\text{conexiones})) \implies$

$\text{obtener}(e_2, \text{obtener}(e_1, e.\text{conexiones})) < \text{longitud}(e.\text{sendas}))$))

Abs: $\wedge(\text{estr}) \rightarrow \text{mapa}$

{Rep(e)}

($\forall e : \wedge(\text{estr})$) Abs(e) =obs m /

(e.estaciones =obs estaciones(m)) \wedge

(($\forall e_1, e_2 : \text{string}$)

definido?(e.conexiones, e_1) \wedge

(definido?(obtener(e.conexiones, e_1), e_2) =obs conectadas?(e_1 , e_2 , m)) \wedge

(conectadas?(e_1 , e_2 , m) \implies *(e.sendas)[i] =obs restriccion(e_1 , e_2 , m)))

```

*****
*****                                Algoritmos                                *****
*****

```

iCrear() \rightarrow res : estr

res \leftarrow {sendas: Vacia(), conexiones: Vacio(), estaciones: Vacio()}

end function

iAgregar(in/out m : estr, in e : estacion)

Agregar(m.estaciones, e)

end function

iConectar(in/out m : estr, in e1 : estacion, in e2 : estacion, in r : restriccion)

var i : nat \leftarrow Longitud(m.sendas)

AgregarAtras(m.sendas, &r)

if \neg Definido?(m.conexiones, e1) then

Definir(m.conexiones, e1, Vacio())

end if

if \neg Definido?(m.conexiones, e2) then

Definir(m.conexiones, e2, Vacio())

end if

Definir(Obtener(m.conexiones, e1), e2, i)

Definir(Obtener(m.conexiones, e2), e1, i)

end function

iEstaciones(in m : estr) \rightarrow res : itConj(string)

res \leftarrow CreaIt(m.estaciones)

end function

iConectadas?(in m : estr, in e1 : estacion, in e2 : estacion) \rightarrow res : bool

res \leftarrow Definido?(m.conexiones, e1) \wedge Definido?(obtener(m.conexiones, e1), e2)

end function

iidSenda(in m : estr, in e1 : estacion, in e2 : estacion) \rightarrow res : nat

res \leftarrow Obtener(Obtener(m.conexiones, e1), e2)

end function

```
iSendas(in m : estr) → res : itVectorPointer(Restriccion)
    res ← CrearIt(m.sendas)
end function

iCopiar(in m : estr) → res : estr
    res ← { sendas : Copiar(m.sendas),
            conexiones : Copiar(m.conexiones),
            estaciones : Copiar(m.estaciones) }
end function
```