

1. Módulo Campus

Interfaz

se explica con: CAMPUS

géneros: campus

Operaciones básicas de campus

NUEVOCAMPUS(**in** al : nat, **in** an : nat) $\rightarrow res$: campus

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

Post $\equiv \{res =_{\text{obs}} \text{crearCampus}(al, an)\}$

Complejidad: $O(n^2)$

Descripción: Crea un nuevo campus vacío de alto al x ancho an .

AGREGAROBSTACULO(**in** p : pos), **in/out** c : campus)

Pre $\equiv \{\text{posValida}(p, c) \wedge \neg \text{ocupada?}(p, c) \wedge c =_{\text{obs}} c_0\}$

Post $\equiv \{c =_{\text{obs}} \text{agregarObstaculo}(p, c_0)\}$

Complejidad: $O(1)$

Descripción: Agrega un obstáculo al campus c en la posición p .

FILAS(**in** c : campus) $\rightarrow res$: nat

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

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

Complejidad: $O(1)$

Descripción: Devuelve el alto (filas) del campus c .

COLUMNAS(**in** c : campus) $\rightarrow res$: nat

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

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

Complejidad: $O(1)$

Descripción: Devuelve el ancho (columnas) del campus c .

OCUPADA?(**in** p : pos, **in** c : campus) $\rightarrow res$: bool

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

Post $\equiv \{res =_{\text{obs}} \text{ocupada?}(p, c)\}$

Complejidad: $O(1)$

Descripción: Devuelve *true* si la posición p se encuentra ocupada por algún obstáculo en el campus c , sino retorna *false*.

POSVALIDA?(**in** p : pos, **in** c : campus) $\rightarrow res$: bool

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

Post $\equiv \{res =_{\text{obs}} \text{posValida?}(p, c)\}$

Complejidad: $O(1)$

Descripción: Devuelve *true* si la posición p es válida en el campus c , sino retorna *false*.

ESINGRESO?(**in** p : pos, **in** c : campus) $\rightarrow res$: bool

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

Post $\equiv \{res =_{\text{obs}} \text{esIngreso?}(p, c)\}$

Complejidad: $O(1)$

Descripción: Verifica si la posición p es una entrada del campus c .

INGRESOSUPERIOR?(**in** p : pos, **in** c : campus) $\rightarrow res$: bool

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

Post $\equiv \{res =_{\text{obs}} \text{ingresoSuperior?}(p, c)\}$

Complejidad: $O(1)$

Descripción: Verifica si la posición p es una entrada superior del campus c .

INGRESOINFERIOR?(**in** p : pos, **in** c : campus) $\rightarrow res$: bool

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

Post $\equiv \{res =_{\text{obs}} \text{ingresoInferior?}(p, c)\}$

Complejidad: $O(1)$

Descripción: Verifica si la posición p es una entrada inferior del campus c .

VECINOS(in p : pos, in c : campus) $\rightarrow res$: conj(pos)

Pre $\equiv \{\text{posValida}(p, c)\}$

Post $\equiv \{res =_{\text{obs}} \text{vecinos}(p, c)\}$

Complejidad: $O(1)$

Descripción: Devuelve un conjunto de las posiciones que rodean a p en el campus c

DISTANCIA(in p_0 : pos, in p_1 : pos, in c : campus) $\rightarrow res$: nat

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

Post $\equiv \{res =_{\text{obs}} \text{distancia}(p_0, p_1, c)\}$

Complejidad: $O(1)$

Descripción: Devuelve la distancia, en casilleros, desde la posición p_0 a la posición p_1 .

PROXPOSICION(in p : pos, in d : dir, in c : campus) $\rightarrow res$: pos

Pre $\equiv \{\text{posValida}(p, c)\}$

Post $\equiv \{res =_{\text{obs}} \text{proxPosicion}(p, d, c)\}$

Complejidad: $O(1)$

Descripción: Indica la posición que se encuentra al lado de p , en la dirección d .

INGRESOSMASCERCANOS(in p : pos, in c : campus) $\rightarrow res$: conj(pos)

Pre $\equiv \{\text{posValida?}(p, c)\}$

Post $\equiv \{res =_{\text{obs}} \text{ingresosMasCercanos}(p, c)\}$

Complejidad: $O(1)$

Descripción: Devuelve un conjunto que contiene las posiciones de los ingresos mas cercanos a p en el campus c .

Representación

Representación de Campus

campus se representa con **estr**

donde **estr** es **tupla**(alto: nat, ancho: nat, obstaculos: matriz(bool))

donde **pos** es **tupla**(fila: nat, columna: nat)

Invariante de representacion en castellano:

1. Para toda p de tipo pos , si p esta definida en **obstaculos**, entonces tanto la fila como la columna de p son menores o iguales a *alto* y *ancho* respectivamente.

Rep : **estr** \rightarrow bool

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

1. $(\forall p : pos) p \in \text{claves}(e.\text{obstaculos}) \Rightarrow (p.\text{fila} \leq e.\text{alto} \wedge p.\text{columna} \leq e.\text{ancho})$

Abs : **estr** $e \rightarrow$ campus

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

Abs(e) $\equiv c : \text{campus} /$

$(\forall p : pos) \text{def?}(p, e.\text{obstaculos}) =_{\text{obs}} \text{ocupada?}(p, c) \wedge \text{alto}(c) =_{\text{obs}} e.\text{alto} \wedge \text{ancho}(c) =_{\text{obs}} e.\text{ancho}$

Algoritmos

Lista de algoritmos

1.	nombre	4
2.	AgregarObstaculo	4
3.	Filas	4
4.	Columnas	4
5.	Ocupada?	4
6.	PosValida	5
7.	EsIngreso?	5
8.	IngresoSuperior?	5
9.	IngresoInferior?	5
10.	Vecinos	6
11.	Distancia	6
12.	ProxPosicion	6
13.	ingresosMasCercano	6

```

iNuevoCampus(in al: nat, in an: nat) → res: estr
begin
  | res.alto ← al // O(1)
  | res.ancho ← an // O(1)
  | res.obstaculos ← vacio() // O(n2)
end
Complejidad: O(n2)

```

Algoritmo 1: nombre

```

iAgregarObstaculo(in p: pos, in/out e: estr)
begin
  | Colocar(p, true, e.obstaculos) // O(1)
end
Complejidad: O(1)

```

Algoritmo 2: AgregarObstaculo

```

iFilas(in e: estr) → res: nat
begin
  | res ← e.alto // O(1)
  | return res
end
Complejidad: O(1)

```

Algoritmo 3: Filas

```

iColumnas(in e: estr) → res: nat
begin
  | res ← e.ancho // O(1)
  | return res
end
Complejidad: O(1)

```

Algoritmo 4: Columnas

```

iOcupada?(in p: pos, in e: estr) → res: bool
begin
  | res ← Ocupada?(p, e.obstaculos) // O(1)
  | return res
end
Complejidad: O(1)
Comentarios: La funcion ocupada? que se llama, es la perteneciente al modulo matriz

```

Algoritmo 5: Ocupada?

```

iPosValida(in  $p$ : pos, in  $e$ : estr)  $\rightarrow$  res: bool
begin
    res  $\leftarrow$  true // O(1)
    if  $p.fila > e.alto \vee p.columna > e.ancho$  then // O(1)
        | res  $\leftarrow$  false // O(1)
    end
    if  $0 < p.fila \vee 0 < p.columna$  then // O(1)
        | res  $\leftarrow$  false // O(1)
    end
    return res
end
Complejidad:  $O(1)$ 

```

Algoritmo 6: PosValida

```

iEsIngreso?(in  $p$ : pos, in  $e$ : estr)  $\rightarrow$  res: bool
begin
    res  $\leftarrow$  false // O(1)
    if  $ingresoSuperior(p, e) \vee ingresoInferior(p, e)$  then // O(1)
        | res  $\leftarrow$  true // O(1)
    end
    return res
end
Complejidad:  $O(1)$ 

```

Algoritmo 7: EsIngreso?

```

iIngresoSuperior?(in  $p$ : pos, in  $e$ : estr)  $\rightarrow$  res: bool
begin
    res  $\leftarrow$  false // O(1)
    if  $p.fila == 1$  then // O(1)
        | res  $\leftarrow$  true
    end
    return res
end
Complejidad:  $O(1)$ 

```

Algoritmo 8: IngresoSuperior?

```

iIngresoInferior?(in  $p$ : pos, in  $e$ : estr)  $\rightarrow$  res: bool
begin
    res  $\leftarrow$  false // O(1)
    if  $p.fila == e.alto$  then // O(1)
        | res  $\leftarrow$  true
    end
    return res
end
Complejidad:  $O(1)$ 

```

Algoritmo 9: IngresoInferior?

```

iVecinos(in p: pos, in e: estr) → res: conjunto(bool)
begin
  | res ← vecinos(p, e.obstaculos) //O(1)
  | return res
end

```

Complejidad: $O(1)$

Comentarios: La funcion vecinos que se llama, es la perteneciente al modulo matriz

Algoritmo 10: Vecinos

```

iDistancia(in p0: pos, in p1: pos, in e: estr) → res: nat
begin
  | res ← distancia(p0, p1, e.obstaculos) //O(1)
  | return res
end

```

Complejidad: $O(1)$

Comentarios: La funcion distancia que se llama, es la perteneciente al modulo matriz

Algoritmo 11: Distancia

```

iProxPosicion(in p: pos, in d: dir, in e: estr) → res: pos
begin
  | res ← proxPosicion(p, d, e.obstaculo) //O(1)
  | return res
end

```

Complejidad: $O(1)$

Comentarios: La funcion proxPosicion que se llama, es la perteneciente al modulo matriz

Algoritmo 12: ProxPosicion

```

iIngresosMasCercano(in p: pos, in e: estr) → res: conjunto(pos)
begin
  | res ← ingresosMasCercano(p, e.obstaculos) //O(1)
  | return res
end

```

Complejidad: $O(1)$

Comentarios: La funcion ingresosMasCercano que se llama, es la perteneciente al modulo matriz

Algoritmo 13: ingresosMasCercano
