Interfaz

se explica con: MAPA

géneros: `mapa`.

Operaciones básicas de mapa

NUEVOMAPA(**in** *largo*: `nat`, **in** *alto*: `nat`, **in** *inicio*: `coordenada`, **in** *llegada*: `coordenada`, **in** *fantasmas*: `conj(coordenada)`, **in** *paredes*: `conj(coordenada)`, **in** *chocolates*: `conj(coordenada)`) → *res* : `mapa`
Pre ≡ {(*inicio* ≠ *llegada* ∧ *todosEnRango*(*paredes* ∪ *fantasmas* ∪ *chocolates* ∪ {*inicio*, *llegada*}, *largo*, *alto*) ∧ {*inicio*, *llegada*} ∩ (*fantasmas* ∪ *paredes*) = ∅ ∧ *disjuntosDeAPares*(*paredes*, *fantasmas*, *chocolates*))}
Post ≡ {*res* = *nuevoMapa*(*largo*, *alto*, *inicio*, *llegada*, *paredes*, *fantasmas*, *chocolates*))}
Complejidad: $O(alto \cdot largo \cdot (\#chocolates + \#fantasmas + \#paredes + 1))$
Descripción: Genera un nuevo mapa
Aliasing: Para construir el mapa hacemos copia de todos los conjuntos

ESCASILLEROPELIGROSO(**in** *m*: `mapa`, **in** *posicion*: `coordenada`) → *res* : `bool`
Pre ≡ {true}
Post ≡ { res = distConFantasmasMasCercano(fantasmas(m), posicion) ≤ 3 }
Complejidad: $O(1)$
Descripción: Devuelve true si el casillero es peligroso, es peligroso si existe un fantasma con distancia ≤ 3 respecto a la posción

ENRANGO(**in** *m*: `mapa`, **in** *posicion*: `coordenada`) → *res* : `bool`
Pre ≡ {true}
Post ≡ {*res* = enRango(posicion, largo(m), alto(m))}
Complejidad: $O(1)$
Descripción: Devuelve true si la posicion se encuentra en rango

CANTCHOCOLATES(**in** *map*: `mapa`) → *res* : `nat`
Pre ≡ {true}
Post ≡ {*res* = #(chocolates(map))}
Complejidad: $O(c)$
Descripción: Devuelve la cantidad de chocolates en el mapa

ESPARED(**in** *map*: `mapa`, **in** *posicion*: `coordenada`) → *res* : `bool`
Pre ≡ {true}
Post ≡ {*res* = true ⟺ *posicion* ∈ paredes(map)}
Complejidad: $O(1)$
Descripción: Devuelve el conjunto de paredes

INICIO(**in** *map*: `mapa`) → *res* : `coordenada`
Pre ≡ {true}
Post ≡ {*res* = inicio(map)}
Complejidad: $O(1)$
Descripción: Devuelve la coordenada de inicio del mapa

LLEGADA(**in** *map*: `mapa`) → *res* : `coordenada`
Pre ≡ {true}
Post ≡ {*res* = llegada(map)}
Complejidad: $O(1)$
Descripción: Devuelve la coordenada de llegada del mapa

IDCHOCOLATE(**in** m : `mapa`, **in** *posicion* : `coordenada`) $\rightarrow$ *res* : `nat`
Pre $\equiv$ {enRango(*posicion*)}
Post $\equiv$ {*posicion* $\in$ chocolates(m) $\iff$ $0 \leq res <$ #chocolates(m)}
Complejidad: $O(1)$
Descripción: Devuelve el id del chocolate en el mapa

Representación
mapa se representa con *mp*

donde *casillero* es tupla(*fantasma*: bool,
peligrosa: bool,
pared: bool,
idChocolate: int)

donde *columna* es array[0...*largo*] de casillero

donde *mp* es tupla(*matriz*: array[0...*alto*] de columna,
#chocolates: nat,
alto: nat,
largo: nat,
inicio: coordenada,
llegada: coordenada)

Invariante de representación
Rep : mp $\longrightarrow$ boolean
Rep(e) $\equiv$ True $\iff$ $(0 \leq e.inicio_1 < e.largo \wedge 0 \leq e.inicio_2 < e.alto\ ) \wedge$
$(0 \leq e.llegada_1 < e.largo \wedge 0 \leq e.llegada_2 < e.alto\ ) \wedge$
$(e.inicio \neq e.llegada) \wedge$
$(\forall$ i: nat$)(0 \leq$ i $< e.largo) \Rightarrow_L ($
$(\forall$ j: nat$)(0 \leq$ j $< e.alto) \Rightarrow_L ($
$(\beta(e.matriz[i][j].pared) +$
$\beta(e.matriz[i][j].fantasma) +$
$\beta(0 \leq e.matriz[i][j].idChocolate < e.\#chocolates) \leq 1) \wedge$
$((\forall\ n$: $\mathbb{N})(0 \leq n < e.\#chocolates) \Rightarrow$
$(\exists!\ i,j$: *nat* $)(\ 0 \leq i < e.largo \wedge 0 \leq j < e.alto\ ) \wedge_L$
$(\ e.matriz[i][j].chocolates = n\ )) \wedge$
$(e.matriz[i][j].peligrosa \Rightarrow_L (\exists\ n,m$: nat$)(0 \leq n < e.largo \wedge 0 \leq m < e.alto) \wedge_L$
$(e.matriz[n][m].fantasma \wedge$ distancia$(\langle i,j\rangle, \langle n,m\rangle) \leq 3)))$
$)$

Función de abstracción
Abs : mp e $\longrightarrow$ Mapa {Rep(e)}
$(\forall\ e$: mp$)$ Abs(e) $=_{\text{obs}}$ m: mapa | largo(m) $= e.largo \wedge$
alto(m) $= e.alto \wedge$
#(chocolates(m)) $= e.\#chocolates \wedge$
$e.inicio = inicio(m) \wedge$
$e.llegada = llegada(m) \wedge$
$(\forall$ i: nat$)(0 \leq$ i $< e.largo) \Rightarrow_L ($
$(\forall$ j: nat$)(0 \leq$ j $< e.alto) \Rightarrow_L ($
$e.matriz[i][j].fantasma \iff \langle i,j\rangle \in fantasmas(m) \wedge$
$e.matriz[i][j].paredes \iff \langle i,j\rangle \in paredes(m) \wedge$
$0 \leq e.matriz[i][j].idChocolate < e.\#chocolates \iff \langle i,j\rangle \in chocolates(m) \wedge$
$e.matriz[i][j].peligrosa \iff distConFantasmasMasCercano(fantasmas(m), \langle i,j\rangle) \leq 3))$

Interfaz

se explica con: PARTIDA

géneros: `partida`.

Operaciones básicas de partida

NUEVAPARTIDA(**in** m : `mapa`) $\rightarrow$ res : `partida`
Pre $\equiv$ {true}
Post $\equiv$ {res = nuevaPartida(m)}
Complejidad: $O(c)$, c es la cantidad de chocolates que contiene el mapa
Descripción: Genera una nueva partida
Aliasing: El mapa se recibe por referencia

MOVER(**in/out** p : `partida`, **in** d : `dirección`)
Pre $\equiv$ {$p_0 = p$}
Post $\equiv$ {p = mover(p_0, d)}
Complejidad: $O(1)$
Descripción: Mueva la posición del jugador un casillero
Aliasing: Se modifica p internamente

GANÓ?(**in** p : `partida`) $\rightarrow$ res : `bool`
Pre $\equiv$ {true}
Post $\equiv$ {res = ganó?(p)}
Complejidad: $O(1)$
Descripción: Devuelve true si el jugador gano la partida

PERDIÓ?(**in** p : `partida`) $\rightarrow$ res : `bool`
Pre $\equiv$ {true}
Post $\equiv$ {res = perdió?(p)}
Complejidad: $O(1)$
Descripción: Devuelve true si el jugador perdio la partida

JUGADOR(**in** p : `partida`) $\rightarrow$ res : `coordenada`
Pre $\equiv$ {true}
Post $\equiv$ {res = perdió?(p)}
Complejidad: $O(1)$
Descripción: Devuelve la posicion del jugador

CANTMOV(**in** p : `partida`) $\rightarrow$ res : `nat`
Pre $\equiv$ {true}
Post $\equiv$ {res = cantMov(p)}
Complejidad: $O(1)$
Descripción: Devuelve la cantidad de movimientos del jugador

Representación
partida se representa con pt
donde pt es tupla($mapa$: mp,
$jugador$: coordenada,
$chocolates$: array[0...c] de bool,
$cantMov$: nat,
$inmunidad$: nat,
$gano$: bool,
$perdio$: bool)

Funciones auxiliares
distancia : coordenada × coordenada $\longrightarrow$ nat
distanciaMinima : coordenada × conj(coordenada) $\longrightarrow$ bool
chocolatesSinComer : pt $\longrightarrow$ conj(coordenada)

distancia$(x, y) \equiv |+x_1 \text{ - } +y_1| + |+x_2 \text{ - } +y_2|$
distanciaMinima$(j, c) \equiv$ **if** $\#(c) = 1$ **then**
distancia(j, dameUno(c))
else
mín(distancia(j, dameUno(c)),distanciaMinima(j, sinUno(c)))
fi
chocolatesSinComer$(e) \equiv$ Esta función devuelve el conjunto de coordenadas de chocolates en la $e.mapa.matriz$ cuyo Id en el array de la partida ($e.chocolates$) aun estan en true

Invariante de representación
Rep : pt $\longrightarrow$ boolean
Rep$(e) \equiv$ True $\iff$ ($mapa.EnRango(e.mapa, e.jugador) \wedge$
$long(e.chocolates) = e.mapa.\#chocolates) \wedge_L$
$e.cantMov = 0 \Rightarrow$
$e.mapa.inicio = jugador \wedge$
if $0 \leq e.mapa.matriz[jugador_1][jugador_2].idChocolate < e.mapa.\#chocolates$ **then**
$e.chocolates[e.mapa.matriz[e.jugador_1][e.jugador_2].idChocolate] = false \wedge$
$e.inmunidad = 10 \wedge$
$(\forall i : \mathbb{N})(0 \leq i < mapa.\#chocolates \wedge i \neq e.mapa.matriz[e.jugador_1][e.jugador_2].idChocolate)$
$\Rightarrow {}_L(e.chocolates[i] = true)$
else
$e.inmunidad = 0 \wedge (\forall i : \mathbb{N})(0 \leq i < mapa.\#chocolates) \Rightarrow {}_L(e.chocolates[i] = true)$
fi $\wedge$
$(e.chocolates[e.mapa.matriz[e.jugador_1][e.jugador_2].idChocolate] = false) \wedge$
$e.inmunidad$=10 $\Rightarrow 0 \leq e.mapa.matriz[e.jugador_1][e.jugador_2].idChocolate < e.mapa.\#chocolates \wedge$
$e.inmunidad \leq 10$ - distanciaMinima($jugador$, chocolatesSinComer(e)) $\wedge$
$e.gano \iff jugador = e.mapa.llegada \wedge$
$e.perdio \iff e.inmunidad = 0 \wedge e.mapa.distConFantasmasMásCercano(e.mapa, e, jugador) \leq 3$

Función de abstracción
Abs : pt e $\longrightarrow$ partida {Rep(e)}
($\forall$ e : pt) Abs(e) $=_{\text{obs}}$ p: partida | mapa(p) = $e.mapa \wedge$
jugador(p) = $e.jugador \wedge$
chocolates(p) = chocolatesSinComer(e) $\wedge$
cantMov(p) = $e.cantMov \wedge$
inmunidad(p) = $e.inmunidad \wedge$
ganó?(p) = $e.gano \wedge$
perdió?(p) = $e.perdio$

Interfaz

se explica con: FICHÍN

géneros: `fichin`.

Operaciones básicas de fichin

NUEVOFICHIN(**in** m : `mapa`) $\rightarrow$ res : `fichin`
Pre $\equiv$ {true}
Post $\equiv$ {res = nuevoFichin(m)}
Complejidad: $O(1)$
Descripción: Genera un fichín
Aliasing: Recibe el mapa por referencia

NUEVAPARTIDA(**in/out** f : `fichin`, **in** j : `jugador`) $\rightarrow$ res : `bool`
Pre $\equiv$ {$f_0 = f$}
Post $\equiv$ {$res = \neg$alguienJugando?(f) $\wedge_L$ res $\Rightarrow_L$ f = nuevaPartida(f_0, j) }
Complejidad: $O(c)$
Descripción: Inicia una nueva partida

MOVER(**in/out** f : `fichin`, **in** d : `dirección`) $\rightarrow$ res : `bool`
Pre $\equiv$ {$f_0 = f$}
Post $\equiv$ {res = alguienJugando?(f) $\wedge_L$ res $\Rightarrow_L$ f = mover(f_0, d) }
Complejidad: $O(|J|)$ donde $|J|$ es el más largo de los nombres de los jugadores
Descripción: Mueve en la dirrección indicada

VERRANKING(**in** f : `fichin`) $\rightarrow$ res : `ranking`
Pre $\equiv$ {true}
Post $\equiv$ {res = ranking(f)}
Complejidad: $O(1)$
Descripción: Devuelve el ranking del fichín
Aliasing: Devuelve el ranking por referencia

OBJETIVO(**in** f : `fichin`) $\rightarrow$ res : `tupla<jugador, nat>`
Pre $\equiv$ {alguienJugando?(f) $\wedge$ def?(jugadorActual(f), ranking(f))}
Post $\equiv$ {res = objetivo(f) }
Complejidad: $O(J \cdot |J|)$ donde J es la cantidad de jugadores y $|J|$ es el más largo de los nombres de los jugadores
Descripción: Devuelve una tupla con el oponente y su puntaje

Representación
partida se representa con fch

donde fch es tupla($mapa$: mp,
$alguienJugando$: bool,
$jugadorActual$: string,
$partidaActual$: pt,
$ranking$: dicc(string, nat))

Invariante de representación
Rep : fch $\longrightarrow$ boolean
Rep(e) $\equiv$ True $\iff$
($e.alguienJugando \iff$ (longitud($e.jugadorActual$) $> 0 \wedge \neg e.partida.gano \wedge \neg e.partida.perdio$)) $\wedge$
($e.pt.gano \Rightarrow$
def?($e.jugadorActual$, $e.ranking$) $\wedge_L$
obtener($e.jugadorActual$, $e.ranking$) $\leq e.partida.cantMov$) $\wedge$
($e.mapa = e.pt.mapa$)

Función de abstracción
Abs : fch e $\longrightarrow$ partida {Rep(e)}
($\forall$ e : fch) Abs(e) $=_{\text{obs}}$ f: fichin | mapa(f) $= e.mapa \wedge$
alguienJugando(f) $= e.alguienJugando \wedge$
ranking(f) $= e.ranking \wedge$
$e.alguienJugando \Rightarrow_L$ partidaActual(f) $= e.partidaActual \wedge$
$e.alguienJugando \Rightarrow_L$ jugadorActual(f) $= e.jugadorActual$