

Algoritmos y Estructuras de Datos II

Segundo Cuatrimestre de 2016

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo Practico 1

Especificacion

Grupo De TP Algo2

Integrante	LU	Correo electrónico
Fernando Castro	627/12	fernandoarielcastro92@gmail.com
Philip Garrett	318/14	garrett.phg@gmail.com
Gabriel Salvo	564/14	gabrielsalvo.cap@gmail.com
Bernardo Tuso	792/14	btuso.95@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. Especificacion	3
2. Renombres de TADs	3
3. TAD Juego	4
4. TAD Mapa	12

1. Especificacion

Esta es una especificacion del Trabajo Practico 1 del 2^{do} cuatrimestre del 2016 presentada por la catedra para la realizacion del Trabajo Practico 2. Ver enunciado:

<http://www.dc.uba.ar/materias/aed2/2016/2c/descargas/tps/tp1/view>

2. Renombres de TADs

TAD TIPO es STRING

TAD POKEMONES es DICCIONARIO(NAT, TIPO)

TAD POKEMON es TUPLA(NAT, TIPO)

TAD COORDENADA es TUPLA(NAT, NAT)

TAD JUGADOR es NAT

TAD ESTADO es ENUM {CONECTADO,DESCONECTADO}

TAD JUGADORES es DICCIONARIO(JUGADOR, ESTADO)

3. TAD Juego

TAD JUEGO

géneros	juego
exporta	juego, generadores, observadores, rareza
usa	NAT, BOOL, JUGADOR, JUGADORES, ESTADO, POKEMON, POKEMONES, TIPO, COORDENADA, CONJUNTO(COORDENADA), CONJUNTO(JUGADOR)

igualdad observacional

$$(\forall pGo, pGo' : \text{juego}) \quad pGo =_{\text{obs}} pGo' \iff \left(\begin{array}{l} (\text{mapa}(pGo) =_{\text{obs}} \text{mapa}(pGo')) \wedge_L \\ (\text{jugadores}(pGo) =_{\text{obs}} \text{jugadores}(pGo')) \wedge_L \\ (\text{pokemones}(pGo) =_{\text{obs}} \text{pokemones}(pGo')) \wedge_L \\ (\forall j : \text{Jugador}) \text{def?}(j, \text{jugadores}(pGo)) \Rightarrow_L \\ (\text{posicionJugador}(j, pGo) =_{\text{obs}} \text{posicionJugador}(j, pGo')) \wedge \\ \text{cantidadDeSanciones}(j, pGo) =_{\text{obs}} \text{cantidadDeSanciones}(j, pGo') \wedge \\ \text{pokemonesAtrapados}(j, pGo) =_{\text{obs}} \text{pokemonesAtrapados}(j, pGo') \wedge_L \\ (\forall p : \text{Pokemon}) \text{def?}(\Pi_1(p), \text{pokemones}(pGo)) \Rightarrow_L \\ (\text{posicionPokemon}(p, pGo) =_{\text{obs}} \text{posicionPokemon}(p, pGo')) \wedge \\ \text{cuantoLlevaEsperando}(p, pGo) =_{\text{obs}} \text{cuantoLlevaEsperando}(p, pGo') \end{array} \right)$$

observadores básicos

mapa	: Juego	→ Mapa
jugadores	: Juego	→ Jugadores
posicionJugador	: Jugador $j \times$ Juego pGo	→ Coordenada {def?(j, jugadores(pGo)) \wedge_L estaConectado?(j, pGo)}
pokemones	: Juego	→ Pokemones
posicionPokemon	: Pokemon $p \times$ Juego pGo	→ Coordenada {def?($\Pi_1(p)$, pokemones(pGo)) \wedge_L esSalvaje?(p, claves(pGo)) {def?($\Pi_1(p)$, pokemones(pGo)) \wedge_L esSalvaje?(p, pGo)}
cuantoLlevaEsperando	: Pokemon $p \times$ Juego pGo	→ Nat
pokemonesAtrapados	: Jugador $j \times$ Juego pGo	→ Pokemones {def?(j, jugadores(pGo))}
cantidadDeSanciones	: Jugador $j \times$ Juego pGo	→ Nat {def?(j, jugadores(pGo))}

generadores

nuevoJuego	: Mapa	→ Juego
agJugador	: Jugador $j \times$ Coordenada $c \times$ Juego pGo	→ Juego { \neg def?(j, jugadores(pGo)) \wedge_L esPosicionValidaMapa(c, pGo)}
agPokemon	: Pokemon $p \times$ Coordenada $c \times$ Juego pGo	→ Juego {def?($\Pi_1(p)$, pokemones(pGo)) \wedge_L esPosicionValidaPokemon(c, pGo)}
moverJugador	: Jugador $j \times$ Coordenada $c \times$ Juego pGo	→ Juego {def?(j, jugadores(pGo)) \wedge_L esPosicionValidaMapa(c, pGo) \wedge_L estaConectado?(j, pGo)}
conectar	: Jugador $j \times$ Coordenada $c \times$ Juego pGo	→ Juego {def?(j, jugadores(pGo)) \wedge_L \neg estaConectado?(j, pGo) \wedge esPosicionValidaMapa(c, pGo)}
desconectar	: Jugador $j \times$ Juego pGo	→ Juego {def?(j, jugadores(pGo)) \wedge_L estaConectado?(j, pGo)}

otras operaciones

esPosicionValidaMapa	: Coordenada $c \times$ Juego pGo	→ bool
esPosicionValidaPokemon	: Coordenada $c \times$ Juego pGo	→ bool
estaEnElRangoDeOtroPokemon	: Coordenada $c \times$ Conj(Nat) $ps \times$ Juego pGo	→ Bool {esPosicionValidaMapa(c, pGo) \wedge $ps \subseteq$ claves(pokemones(pGo))}
esSalvaje?	: Pokemon $p \times$ Juego pGo	→ Bool {def?($\Pi_1(p)$, pokemones(pGo))}

esSalvajeAux	: Nat $id \times \text{Conj}(\text{Nat}) \text{ } js \times \text{Juego } pGo$	$\longrightarrow \text{Bool}$
estaEnRangoDeAtrapar	: Jugador $j \times \text{Juego } pGo$	$\longrightarrow \text{Bool}$
estaEnRangoDeAtraparAux	: Jugador $j \times \text{Conj}(\text{Nat}) \text{ } ps \times \text{Juego } pGo$	$\longrightarrow \text{Bool}$
pokemonDelRango	: Jugador $j \times \text{Juego } pGo$	$\longrightarrow \text{Pokemon}$
pokemonDelRangoAux	: Jugador $j \times \text{Conj}(\text{Nat}) \text{ } ps \times \text{Juego } pGo$	$\longrightarrow \text{Pokemon}$
jugadoresDelRango	: Coordenada $c \times \text{Juego } pGo$	$\longrightarrow \text{Conj}(\text{Jugador})$
jugadoresDelRangoAux	: Coordenada $c \times \text{Conj}(\text{Jugador}) \text{ } js \times \text{Juego } pGo$	$\longrightarrow \text{Conj}(\text{Jugador})$
restaAbsoluta	: Nat $n1 \times \text{Nat } n2$	$\longrightarrow \text{Nat}$
rareza	: Tipo $t \times \text{Juego } pGo$	$\longrightarrow \text{Nat}$
cantidadPokemonesTipo	: Tipo $t \times \text{Conj}(\text{Nat}) \text{ } ps \times \text{Juego } pGo$	$\longrightarrow \text{Nat}$
dividir	: Nat $n \times \text{Nat } d$	$\longrightarrow \text{Nat} \quad \{\neg(d = 0?)\}$
estaDefTipo	: Tipo $t \times \text{conj}(\text{nat}) \text{ } ps \times \text{Juego } pGo$	$\longrightarrow \text{Bool}$
estaConectado?	: Jugador $j \times \text{Juego } pGo$	$\longrightarrow \text{Bool}$
sacarPokemones	: Conj(nat) $cp \times \text{Juego } pGo$	$\longrightarrow \text{Pokemones}$

axiomas $\forall m: \text{Mapa } \forall j, j1, j2: \text{Jugador } \forall c: \text{Coordenada } \forall pGo: \text{Juego } \forall p: \text{Pokemon } \forall n: \text{Nat } \forall t: \text{Tipo}$

mapa(nuevoJuego(m))	$\equiv m$
jugadores(nuevoJuego(m))	$\equiv \emptyset$
pokemones(nuevoJuego(m))	$\equiv \emptyset$

mapa(agJugador(j, c, pGo))	$\equiv \text{mapa}(pGo)$
jugadores(agJugador(j, c, pGo))	$\equiv \text{Ag}(j, \text{jugadores}(pGo))$
posicionJugador(j1, agJugador(j2, c, pGo))	$\equiv \text{if } j1 = j2 \text{ then } c$
	else
	$\text{posicionJugador}(j1, pGo)$
	fi
pokemones(agJugador(j, c, pGo))	$\equiv \text{pokemones}(pGo)$
posicionPokemon(p, agJugador(j, c, pGo))	$\equiv \text{posicionPokemon}(p, pGo)$
cuantoLlevaEsperando(p, agJugador(j, c, pGo))	$\equiv \text{if } \text{estanEnElMismoRango}(\text{posicionPokemon}(p, pGo), c, \text{mapa}(pGo)) \text{ then } 0$
	else
	$\text{cuantoLLevaEsperando}(p, pGo) + 1$
	fi

<p>pokemonesAtrapados(j1, agJugador(j2, c, pGo))</p>	\equiv if j1 = j2 then vacio else if estanEnElMismoRango(posicionJugador(j1, pGo), c, pGo) then pokemonesAtrapados(j1, pGo) else if (estaEnRangoDeAtrapar(j1,pGo)) then if cuantoLlevaEsperando(pokemonDelRango(j1, pGo), pGo) = 9 then if j1 = dameUno(jugadoresPorAtrapar(pokemonDelRango(j1, pGo), pGo)) then definir(Π_1 (pokemonDelRango((j1, pGo), Π_2 (pokemonDelRango((j1, pGo), pGo)), pokemonesAtrapados(j1, pGo)) else pokemonesAtrapados(j1, pGo) fi else pokemonesAtrapados(j1, pGo) fi else pokemonesAtrapados(j1, pGo) fi ELSE pokemonesAtrapados(j1, pGo) fi ELSE pokemonesAtrapados(j1, pGo) fi
<p>cantidadDeSanciones(j1, agJugador(j2, c, pGo))</p>	\equiv if j1 = j2 then 0 else cantidadDeSanciones(j1) fi
<p>mapa(agPokemon(p, c, pGo)) jugadores(agPokemon(p, c, pGo)) posicionJugador(j, agPokemon(p, c, pGo)) pokemones(agPokemon(p, c, pGo)) posicionPokemon(p1, agPokemon(p2, c, pGo))</p>	\equiv mapa(pGo) \equiv jugadores(pGo) \equiv posicionJugador(j, pGo) \equiv definir(Π_1 (p), Π_2 (p), pokemones(pGo)) \equiv if p1 = p2 then c else posicionPokemon(p1, pGo) fi
<p>cuantoLlevaEsperando(p1, agPokemon(p2, c, pGo))</p>	\equiv if p1 = p2 then 0 else cuantoLlevaEsperando(p1, pGo) fi
<p>pokemonesAtrapados(j, agPokemon(p, c, pGo)) cantidadDeSanciones(j, agPokemon(p, c, pGo))</p>	\equiv pokemonesAtrapados(j, pGo) \equiv cantidadDeSanciones(j, pGo)
<p>mapa(moverJugador(j, c, pGo))</p>	\equiv mapa(pGo)

jugadores(moverJugador(j,c,pGo))	\equiv if movimientoInvalido(posicionJugador(j,pGo),c,mapa(pGo)) then if cantidadDeSanciones(j,pGo) = 4 then borrar(j,jugadores(pGo)) else jugadores(pGo) fi else jugadores(pGo) fi
posicionJugador(j1, moverJugador(j2,c,pGo))	\equiv if j1 = j2 then c else posicionJugador(j1,pGo) fi
pokemones(moverJugador(j,c,pGo))	\equiv if movimientoInvalido(posicionJugador(j,pGo),c,mapa(pGo)) then if cantidadDeSanciones(j,pGo) = 4 then sacarPokemones(claves(pokemonesAtrapados(j,pGo)),pGo) else pokemones(pGo) fi else pokemones(pGo) fi
posicionPokemon(p,moverJugador(j,c,pGo))	\equiv posicionPokemon(pGo)
cuantoLlevaEsperando(p,moverJugador(j,c,pGo))	\equiv if estanEnElMismoRango(posicionPokemon(p,pGo),posicionJugador(j,pGo),mapa(pGo)) then if estanEnElMismoRango(posicionPokemon(p,pGo),c,mapa(pGo)) then cuantoLlevaEsperando(p,pGo) else cuantoLlevaEsperando(p,pGo) + 1 fi else if estanEnElMismoRango(posicionPokemon(p,pGo),c,mapa(pGo)) then 0 else cuantoLlevaEsperando(p,pGo) + 1 fi
cantidadDeSanciones(j1,moverJugador(j2,c,pGo))	\equiv if j1 = j2 then if movimientoInvalido(posicionJugador(j1),c,pGo) then cantidadDeSanciones(j1,pGo) + 1 else cantidadDeSanciones(j1,pGo) fi else cantidadDeSanciones(j1,pGo) fi

```

pokemonesAtrapados(j1, moverJugador(j2, c, pGo))  ≡ if j1 = j2 then
    pokemonesAtrapados(j1, pGo)
else
    if estaEnRangoDeAtrapar(j1, pGo) then
        if estanEnElMismoRango(posicionJugador(j1,
            pGo), c, mapa(pGo)) then
            pokemonesAtrapados(j1, pGo)
        else
            if
                cuantoLlevaEsperando(pokemonDelRango(j1,
                    pGo), pGo)
                = 9 then
                    if j1 = dameUno(jugadoresDelRango(posicionPokemon(pok
                        then
                            definir(Π1(pokemonDelRango(j1,
                                pGo)), Π2(pokemonDelRango(j1,
                                    pGo)), pokemonesAtrapados(j1,
                                        pGo))
                        else
                            pokemonesAtrapados(j1, pGo)
                    fi
                else
                    pokemonesAtrapados(j1, pGo)
            fi
        fi
    fi
    pokemonesAtrapados(j1, pGo)
fi

mapa(conectar(j,c,pGo)) ≡ mapa(pGo)
jugadores(conectar(j,c,pGo)) ≡ definir(j,conectado,jugadores(pGo))
posicionJugador(j1,conectar(j2,c,pGo)) ≡ if j1 = j2 then
    c
    else
        posicionJugador(j1,pGo)
    fi

pokemones(conectar(j,c,pGo)) ≡ pokemones(pGo)
posicionPokemon(p,conectar(j,c,pGo)) ≡ posicionPokemon(p,pGo)
cuantoLlevaEsperando(p,conectar(j,c,pGo)) ≡ if
    estanEnElMismoRango(posicionPokemon(p),c,mapa(pGo)) then
        0
    else
        cuantoLlevaEsperando(p,pGo) + 1
    fi

```


pokemonesAtrapados(j1,conectar(j2,c,pGo))

```

≡ if j1 = j2 then
    pokemonesAtrapados(j1,pGo)
else
    if estaEnRangoDeAtrapar(j1,pGo) then
        if estanEnElMismoRango(
            posicionJugador(j1,pGo),c,mapa(pGo))
        then
            pokemonesAtrapados(j1,pGo)
        else
            if cuantoLlevaEsperando(
                pokemonDelRango(j1,pGo),pGo)
            = 9 then
                if j1 = dameUno(
                    JugadoresDelRango(
                        posicionPokemon(pGo),
                        pGo))
                then
                    definir(Π1(pokemonDelRango(j1,pGo)),
                        Π2(pokemonDelRango(j1,pGo)),
                        pokemonesAtrapados(j1,pGo))
                else
                    pokemonesAtrapados(j1,pGo)
            fi
        else
            pokemonesAtrapados(j1,pGo)
        fi
    fi
    pokemonesAtrapados(j1,pGo)
fi
≡ cantidadDeSanciones(j1,pGo)

```

cantidadDeSanciones(j1,conectar(j2,c,pGo))

mapa(desconectar(j, pGo))
 jugadores(desconectar(j, pGo))
 posicionJugador(j1, desconectar(j2, pGo))
 pokemones(desconectar(j, pGo))
 posicionPokemon(p, desconectar(j, pGo))
 cuantoLlevaEsperando(p, desconectar(j, pGo))
 pokemonesAtrapados(j1, desconectar(j2, pGo))
 cantidadDeSanciones(j1, desconectar(j2, pGo))

```

≡ mapa(pGo)
≡ definir(j, Desconectado, jugadores(pGo))
≡ posicionJugador(j1, pGo)
≡ pokemones(pGo)
≡ posicionPokemon(p, pGo)
≡ cuantoLlevaEsperando(p, pGo)
≡ pokemonesAtrapados(j1, pGo)
≡ cantidadDeSanciones(j1, pGo)

```

esPosicionValidaMapa(c, pGo)
 esPosicionValidaPokemon(c, pGo)

```

≡ c ∈ posiciones(mapa(pGo))
≡ esPosicionValidaMapa(c, pGo) ∧L ¬ estaEnElRangoDeOtroPokemon(
    c, claves(pokemones(pGo)),pGo)

```

estaEnElRangoDeOtroPokemon(c, ps, pGo)

```

≡ if ∅?(ps) then
    false
else
    territorioOcupado(c,posicionPokemon(⟨dameUno(ps),
        obtener(dameUno(ps), pokemones(pGo))⟩, pGo),
        mapa(pGo)) ∨ estaEnElRangoDeOtroPokemon(
            c,sinUno(ps),pGo)

```

restaAbsoluta(p1, p2)
 rareza(t, pGo)

```

≡ if p1 ≤ p2 then p2 - p1 else p1 - p2 fi
≡ 100 - dividir(cantidadPokemonesTipo(t,(claves(pokemones(pGo),
    #claves(pokemones(pGo)))) * 100

```

cantidadPokemonesTipo(t, ps, pGo)

```
≡ if ∅?(ps) then
    0
  else
    if obtener(dameUno(ps)) = t then
      cantidadPokemonesTipo(t, sinUno(ps), pGo)
      + 1
    else
      cantidadPokemonesTipo(t, sinUno(ps), pGo)
    fi
```

dividir(n, d)
sacarPokemones(cp, pGo)

```
≡ if n ≥ d then dividir(n-d, d) + 1 else 0 fi
≡ if ∅?(cp) then
    pokemones(pGo)
  else
    borrar(dameUno(cp), sacarPokemones(sinUno(cp), pokemones(pGo)))
  fi
```

estaDefTipo(t, ps, pGo)

```
≡ if ∅?(ps) then
    false
  else
    if t = obtener(dameUno(ps), pokemones(pGo))
    then
      true
    else
      estaDefTipo(t, sinUno(ps), pGo)
    fi
  fi
```

esSalvaje?(p, pGo)
esSalvajeAux(id, js, pGo)

```
≡ esSalvajeAux(Π1(p), claves(jugadores(pGo)), pGo)
≡ if ∅?(js) then
    true
  else
    if def?(id, pokemonesAtrapados(dameUno(js), pGo))
    then
      false
    else
      esSalvajeAux(id, sinUno(js), pGo)
    fi
  fi
```

estaEnRangoDeAtrapar(j, pGo)

```
≡ estaEnRangoDeAtraparAux(j, claves(pokemones(pGo)))
```

estaEnRangoDeAtraparAux(j, ps, pGo)

```
≡ if ∅?(ps) then
    false
  else
    estanEnElMismoRango(posicionJugador(j, pGo),
      posicionPokemon((dameUno(ps), obtener(dameUno(ps), pokemones(pGo)))
      ∨ estaEnRangoDeAtraparAux(j, sinUno(ps), pGo)
    fi
```

jugadoresDelRango(c, pGo)

```
≡ jugadoresDelRangoAux(c, claves(jugadores(pGo)), pGo)
```

jugadoresDelRangoAux(c,js,pGo)

pokemonDelRango(j,pGo)
pokemonDelRangoAux(j,ps,pGo)

Fin TAD

```
≡ if ∅?(js) then
    ∅
else
    if estaConectado?(dameUno(js),pGo) then
        if estanEnElMismoRango(c,posicionJugador(⟨dameUno(
            obtener(dameUno(js),jugadores(pGo))⟩)))
        then
            Ag(dameUno(js),jugadoresDelRangoAux(c,sinUno(js),
            pGo))
        else
            jugadoresDelRangoAux(c,sinUno(js),pGo)
        fi
    else
        jugadoresDelRangoAux(c,sinUno(js),pGo)
    fi
fi
≡ pokemonDelRangoAux(j,claves(pokemones(pGo)),pGo)
≡ if estanEnElMismoRango(posicionJugador(j,pGo),
    posicionPokemon(⟨dameUno(ps),obtener(dameUno(ps),pokemones
    pGo)⟩)) then
    ⟨dameUno(ps),obtener(dameUno(ps),pokemones(pGo))⟩
else
    pokemonDelRangoAux(j,sinUno(ps),pGo)
fi
```

4. TAD Mapa

TAD MAPA

géneros mapa

exporta mapa, generadores, observadores, movimientoInvalido, estanEnElMismoRango, territorioOcupado

usa NAT, BOOL, COORDENADA, CONJUNTO(COORDENADA)

igualdad observacional

$$(\forall m, m' : \text{Mapa}) \left(m =_{\text{obs}} m' \iff \left(\begin{array}{l} (\text{posiciones}(m) =_{\text{obs}} \text{posiciones}(m')) \wedge \\ (\forall c1: \text{Coordenada}) \\ (\text{conexionesDirectas}(c1, m) =_{\text{obs}} \text{conexionesDirectas}(c1, m')) \end{array} \right) \right)$$

observadores básicos

posiciones : Mapa $m \longrightarrow \text{Conj}(\text{Coordenada})$

conexionesDirectas : Coordenada $c \times \text{Mapa } m \longrightarrow \text{Conj}(\text{Coordenada})$

generadores

crear : $\longrightarrow \text{Mapa}$

agCoordenada : Coordenada $c \times \text{Mapa } m \longrightarrow \text{Mapa}$

conectarCoordenadas : Coordenada $c1 \times \text{Coordenada } c2 \times \text{Mapa } m \longrightarrow \text{Mapa}$
 $\{c \notin \text{posiciones}(m)\}$
 $\{c1 \neq c2 \wedge c1, c2 \in \text{posiciones}(m)\}$

otras operaciones

existeCamino : Coordenada $c1 \times \text{Coordenada } c2 \times \text{Mapa } m \longrightarrow \text{bool}$
 $\{c1, c2 \in \text{posiciones}(m)\}$

existenCaminos : Coordenada $c1 \times \text{Conj}(\text{Coordenada}) \text{ } cs \times \text{Mapa } m \longrightarrow \text{bool}$
 $\{\text{Ag}(c1, cs) \subseteq \text{posiciones}(m)\}$

movimientoInvalido : Coordenada $c1 \times \text{Coordenada } c2 \times \text{Mapa } m \longrightarrow \text{bool}$
 $\{c1, c2 \in \text{posiciones}(m)\}$

distancia : Coordenada $c1 \times \text{Coordenada } c2 \longrightarrow \text{Nat}$

estanEnElMismoRango : Coordenada $c1 \times \text{Coordenada } c2 \longrightarrow \text{bool}$

restaAbsoluta : Coordenada $c1 \times \text{Coordenada } c2 \longrightarrow \text{Nat}$

territorioOcupado : Coordenada $c1 \times \text{Coordenada } c2 \times \text{Mapa } m \longrightarrow c1, c2 \in \text{posiciones}(m)$

axiomas $\forall c, c1, c2: \text{Coordenada } \forall cs: \text{conj}(\text{Coordenada}) \forall m: \text{Mapa}$

posiciones(crear()) $\equiv \emptyset$

posiciones(agCoordenada(c, m)) $\equiv \text{Ag}(c, \text{posiciones}(m))$

posiciones(conectarCoordenadas(c1, c2, m)) $\equiv \text{posiciones}(m)$

conexionesDirectas(c, crear()) $\equiv \text{vacío}$

conexionesDirectas(c, agCoordenada(c1, m)) $\equiv \text{conexionesDirectas}(c, m)$

conexionesDirectas(c, conectarCoordenadas(c1, c2, m)) \equiv **if** $c=c1$ **then**
 $\text{Ag}(c2, \text{conexionesDirectas}(c, m))$
else
if $c=c2$ **then**
 $\text{Ag}(c1, \text{conexionesDirectas}(c, m))$
else
 $\text{conexionesDirectas}(c, m)$
fi

existeCamino(c1, c2, m) \equiv **if** $c2 \in \text{conexionesDirectas}(c1, \text{mapa}) \vee c1 \in \text{conexionesDirectas}(c2, \text{mapa})$ **then**
true

else

$\text{existenCaminos}(\text{conexionesDirectas}(c1, m), c2, m) \vee \text{existenCaminos}(\text{conexionesDirectas}(c2, m), c1, m)$

fi

existenCaminos(cs, c, m) \equiv **if** $\text{vacío?}(cs)$ **then**
false

else

$\text{existeCamino}(\text{dameUno}(cs), c, m) \vee \text{existenCaminos}(\text{sinUno}(cs), c, m)$

fi

```

movimientoInvalido(c1, c2, m)  $\equiv$  distancia(c1,c2)  $\geq$  100  $\vee$  !existenCaminos(c1,c2,m)
distancia(c1, c2)  $\equiv$  (restaAbsoluta( $\Pi_1$ (c1),  $\Pi_1$ (c2))) * (restaAbsoluta( $\Pi_1$ (c1),  $\Pi_1$ (c2))) +
    (restaAbsoluta( $\Pi_2$ (c1),  $\Pi_2$ (c2))) * (restaAbsoluta( $\Pi_2$ (c1),  $\Pi_2$ (c2))))
estanEnElMismoRango(c1, c2)  $\equiv$  distancia(c1,c2)  $\leq$  2 * 2
restaAbsoluta(p1, p2)  $\equiv$  if p1  $\leq$  p2 then p2 - p1 else p1 - p2 fi
territorioOcupado(c1,c2,m)  $\equiv$  if distancia(c1, c2)  $\geq$  5 * 5 then False else True fi

```

Fin TAD