**Algoritmos y Estructuras de Datos**

Ejercicios resueltos en la clase del 3 de septiembre
**Tipos Abstractos de Datos**
Segundo Cuatrimestre 2025

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

## Punto2D observado con radio y ángulo

`TAD` Punto2D {

`obs` r : $\mathbb{R}$
`obs` $\theta$ : $\mathbb{R}$
`proc nuevoPunto`(in $x, y : \mathbb{R}$) : $Punto2D$ {
  `requiere` { $Verdadero$ }
  `asegura` { $res.r = radio(x, y) \land esAngulo(res.\theta, x, y)$ }
}

`proc igualdad`(in $P_1 : Punto2D$, in $P_2 : Punto2D$) : bool {
  `asegura` { $res = \text{true} \leftrightarrow (P_1.r = P_2.r \land (\exists k : \mathbb{Z})\ (P_1.\theta = P_2.\theta + 2k\pi))$ }
}

`proc mover`(inout $p : Punto2D, dx, dy : \mathbb{R}$) {
  `requiere` { $p = P_0$ }
  `asegura` {
    $p.r = radio(coordX(P_0.r, P_0.\theta) + dx, coordY(P_0.r, P_0.\theta) + dy) \land$
    $esAngulo(p.\theta, coordX(P_0.r, P_0.\theta) + dx, coordY(P_0.r, P_0.\theta) + dy)$
  }
}

`proc distancia`(in $p1, p2 : Punto2D$) : $\mathbb{R}$ {
  `requiere` { $Verdadero$ }
  `asegura` { $res = distanciaEntre(p1, p2)$ }
}

`proc distanciaAlOrigen`(in $p : Punto2D$) : $\mathbb{R}$ {
  `requiere` { $Verdadero$ }
  `asegura` { $res = p.r$ }
}

`aux radio`$(x, y : \mathbb{R}) : \mathbb{R} = \sqrt{x^2 + y^2}$
`aux angulo`$(x, y : \mathbb{R}) : \mathbb{R} = atan(y/x)$
`pred esAngulo`$(\theta, x, y : \mathbb{R})$ {
  $(x \neq 0 \rightarrow_L \theta = angulo(x, y)) \land$
  $(x = 0 \rightarrow ((y >= 0 \rightarrow \theta = \pi/2) \land (y < 0 \rightarrow \theta = 3 * \pi/2)))$
}
`aux coordX`$(r, \theta : \mathbb{R}) : \mathbb{R} = r * cos(\theta)$
`aux coordY`$(r, \theta : \mathbb{R}) : \mathbb{R} = r * sin(\theta)$
`aux distanciaEntre`$(p1, p2 : Punto2D) : \mathbb{R} =$
  $\sqrt{(coordX(p2.r, p2.\theta) - coordX(p1.r, p1.\theta))^2 + (coordY(p2.r, p2.\theta) - coordY(p1.r, p1.\theta))^2}$

}

### Número Racional

Para este ejercicio pueden asumir que $\div$ es división entera.

`TAD` Fracción {

`obs` numerador : $\mathbb{Z}$
`obs` denominador : $\mathbb{Z}$

`proc nuevaFracción`(in $n, d : \mathbb{Z}$) : $Fracción$ {
`requiere` { $d \neq 0$ }
`asegura` { $res.numerador = n \land res.denominador = d$ }
}

`proc igualdad`(in $f1, f2 : Fracción$) : bool {
`asegura` { $res = true \leftrightarrow F_1.numerador \div F_1.denominador = F_2.numerador \div F_2.denominador$ }
}

`proc suma`(in $f1, f2 : Fracción$) : $Fracción$ {
`requiere` { $Verdadero$ }
`asegura` {
$res.numerador = f1.numerador \times f2.denominador + f2.numerador \times f1.denominador \land$
$res.denominador = f1.denominador \times f2.denominador$
}
}

`proc resta`(in $f1, f2 : Fracción$) : $Fracción$ {
`requiere` { $Verdadero$ }
`asegura` {
$res.numerador = f1.numerador \times f2.denominador - f2.numerador \times f1.denominador \land$
$res.denominador = f1.denominador \times f2.denominador$
}
}

`proc multiplicacion`(in $f1, f2 : Fracción$) : $Fracción$ {
`requiere` { $Verdadero$ }
`asegura` {
$res.numerador = f1.numerador \times f2.numerador \land$
$res.denominador = f1.denominador \times f2.denominador$
}
}

`proc division`(in $f1, f2 : Fracción$) : $Fracción$ {
`requiere` { $f2.numerador \neq 0$ }
`asegura` {
$res.numerador = f1.numerador \times f2.denominador \land$
$res.denominador = f1.denominador \times f2.numerador$
}
}

}

## Conjunto sobre secuencia

TAD Conjunto$\langle T\rangle$ {

obs elems : seq$\langle T\rangle$

proc conjuntoVacio() : Conjunto$\langle T\rangle$ {
requiere $\{ Verdadero \}$
asegura $\{ |res.elems| = 0 \}$
}

proc igualdad(in $c_1, c_2$ : Conjunto$\langle T\rangle$) : bool {
requiere $\{ Verdadero \}$
asegura $\{ res = true \leftrightarrow (\forall e : T)\ (e \in c_1.elems \leftrightarrow e \in c_2.elems) \}$
}

*Solución que vimos en el turno mañana:*

proc agregar(inout $c$ : Conjunto$\langle T\rangle$, in $e : T$) {
requiere $\{ c = C_0 \}$
asegura $\{ c.elems = C_0.elems + +\langle e\rangle \}$
}

*Solución que vimos en el turno noche:*

proc agregar(inout $c$ : Conjunto$\langle T\rangle$, in $e : T$) {
requiere $\{ c = C_0 \}$
asegura {
$e \in c.elems \land$
$siguenLosMismos(C_0, c) \land$
$soloSeAgregó(C_0, c, e)$
}
}

pred siguenLosMismos($C_0, c$ : Conjunto$\langle T\rangle$) {
$(\forall t : T)\ (t \in C_0.elems \rightarrow t \in c.elems)$
}

pred soloSeAgregó($C_0, c$ : Conjunto$\langle T\rangle$, $e : T$) {
$(\forall t : T)\ (t \in c.elems \land t \neq e \rightarrow t \in C_0.elems)$
}

proc pertenece(in $c$ : Conjunto$\langle T\rangle$, in $e : T$) : bool {
requiere $\{ Verdadero \}$
asegura $\{ res = true \leftrightarrow e \in c.elems \}$
}

proc eliminar(inout $c$ : Conjunto$\langle T\rangle$, in $e : T$) {
requiere $\{ c = C_0 \}$
asegura {
$\neg(e \in c.elems) \land$
$noSaquéOtros(C_0, c, e)$
}
}

pred noSaquéOtros($C_0, c$ : Conjunto$\langle T\rangle$, $e : T$) {
$(\forall t : T)\ (t \neq e \rightarrow (t \in C_0.elems \leftrightarrow t \in c.elems))$
}

}

TAD Diccionario$\langle K, V\rangle$ {

obs d : dicc$\langle K, V\rangle$

proc `nuevoDiccionario`() : Diccionario$\langle K, V\rangle$ {
  requiere { $Verdadero$ }
  asegura { $res.d = \langle\rangle$ }
}

proc `definir`(inout $d$ : Diccionario$\langle K, V\rangle$, in $k : K$, in $v : V$) {
  requiere { $d = D_0$ }
  asegura { $d.d = setKey(D_0.d, k, v)$ }
}

proc `obtener`(in $d$ : Diccionario$\langle K, V\rangle$, in $k : K$) : $V$ {
  requiere { $k \in d.d$ }
  asegura { $res = d.d[k]$ }
}

proc `esta`(in $d$ : Diccionario$\langle K, V\rangle$, in $k : K$) : bool {
  requiere { $Verdadero$ }
  asegura { $res = true \leftrightarrow k \in d.d$ }
}

proc `borrar`(inout $d$ : Diccionario$\langle K, V\rangle$, in $k : K$) {
  requiere { $d = D_0 \land k \in d.d$ }
  asegura { $d.d = delKey(D_0.d, k)$ }
}

}

Posición $\texttt{ES}$ tupla $\langle \mathbb{Z}, \mathbb{Z} \rangle$
Matriz$\langle T \rangle$ $\texttt{ES}$ seq$\langle$seq$\langle T \rangle\rangle$

$\texttt{TAD}$ Buscaminas {

obs bombas : Matriz$\langle$bool$\rangle$
obs descubiertas : Matriz$\langle$bool$\rangle$
obs banderines : Matriz$\langle$bool$\rangle$

proc igualdad(in $B_1 : Buscaminas$, in $B_2 : Buscaminas$) : bool {
asegura { $res = \text{true} \leftrightarrow$
$B_1.bombas = B_2.bombas \land$
$B_1.descubiertas = B_2.descubiertas \land$
$B_1.banderines = B_2.banderines$
}
}

proc nuevoJuego(in $t$ : Matriz$\langle$bool$\rangle$) : Buscaminas {
requiere { $esMatriz(t) \land |t| > 0 \land_L |t[0]| > 0$ }
asegura {
$res.bombas = t \land mismasDimensiones(res.descubiertas, t) \land$
$todoFalse(res.descubiertas) \land res.banderines = res.descubiertas$
}
}

pred esMatriz($s$ : Matriz$\langle$bool$\rangle$) {
$(\exists l : \mathbb{Z})\ ((\forall i : \mathbb{Z})\ (0 \leq i < |s| \rightarrow_L |s[i]| = l))$
}

pred mismasDimensiones($A$ : Matriz$\langle$bool$\rangle$, $B$ : Matriz$\langle$bool$\rangle$) {
$|A| = |B| \land_L (\forall i : \mathbb{Z})\ (0 \leq i < |A| \rightarrow_L |A[i]| = |B[i]|)$
}

pred todoFalse($A$ : Matriz$\langle$bool$\rangle$) {
$(\forall i, j : \mathbb{Z})\ (enRango(i, j, A) \rightarrow_L A[i][j] = false)$
}

pred enRango($i, j : \mathbb{Z}, A$ : Matriz$\langle$bool$\rangle$) {
$0 \leq i < |A| \land_L 0 \leq j < |A[0]|$
}

proc estadoDelJuego(in $b$ : Buscaminas) : string {
requiere { $Verdadero$ }
asegura {
$(\textit{reventó}(b) \rightarrow res = \text{“carita muerta”}) \land$
$((\neg \textit{reventó}(b) \land \textit{ganó}(b)) \rightarrow res = \text{“carita cool”}) \land$
$((\neg \textit{reventó}(b) \land \neg \textit{ganó}(b)) \rightarrow res = \text{“carita feliz”})$
}
}

pred reventó($b$ : Buscaminas) {
$(\exists i, j : \mathbb{Z})\ (enRango(i, j, b.bombas) \land_L (b.bombas[i][j] = true \land b.descubiertas[i][j] = true))$
}

pred ganó($b$ : Buscaminas) {
$(\forall i, j : \mathbb{Z})\ (enRango(i, j, b.bombas) \rightarrow_L (b.bombas[i][j] = false \rightarrow b.descubiertas[i][j] = true))$
}

proc jugar(inout $b$ : Buscaminas, in $i, j : \mathbb{Z}$) {
requiere {
$b = B0 \land$
$\neg(\textit{reventó}(b) \lor jugóTodosLosVacios(b)) \land$
$enRango(i, j, b.bombas) \land_L b.descubiertas[i][j] = false \land b.banderines[i][j] = false$
}
asegura {

$b.bombas = B0.bombas \land$
$b.banderines = B0.banderines \land$
$mismasDimensiones(b.descubiertas, B0.descubiertas) \land_L$ (
$b.descubiertas[i][j] = true \land$
$descubiertasSeMantienen(B0, b) \land$
$descubiertasEnCadena(B0, b, i, j)$
)
}
}

`pred descubiertasSeMantienen`$(B0, b : Buscaminas)$ {
$(\forall k, l : \mathbb{Z})$ $(enRango(k, l, b.bombas) \to_L (B0.descubiertas[k][l] = true \to b.descubiertas[k][l] = true))$
}

`pred descubiertasEnCadena`$(B0, b : Buscaminas, i, j : \mathbb{Z})$ {
$(\forall k, l : \mathbb{Z})$ (
$enRango(k, l, b.bombas) \to_L (B0.descubiertas[k][l] = false \to$
$hayCaminoLibreEntre(b, i, j, k, l) \leftrightarrow b.descubiertas[k][l] = true)$
)
}

`pred hayCaminoLibreEntre`$(b : \text{Buscaminas}, i_d, j_d, i_h, j_h : \mathbb{Z})$ {
$b.bombas[i_d][j_d] = false \land b.bombas[i_h][j_h] = false \land$
$(\exists s : \mathsf{seq}\langle Posición\rangle)$ (
$esCaminoEn(s, b) \land |s| > 0 \land_L s[0] = \langle i_d, j_d\rangle \land s[|s| - 1] = \langle i_h, j_h\rangle \land$
$(\forall k : \mathbb{Z})$ $(0 \leq k < |s| - 1 \to_L bombasAlrededor(s[k]_1, s[k]_2, b) = 0)$
)
}

`pred esCaminoEn`$(s : \mathsf{seq}\langle Posición\rangle, b : \text{Buscaminas})$ {
$(\forall i : \mathbb{Z})$ (
$1 \leq i < |s| \to_L (enRango(s[i]_1, s[i]_2, b.bombas) \land sonAdyacentes(s[i-1], s[i]))$
)
}

`pred sonAdyacentes`$(p, q : Posición)$ {
$p_1 - 1 \leq q_1 \leq p_1 + 1 \land p_2 - 1 \leq q_2 \leq p_2 + 1 \land \neg(p_1 = q_1 \land p_2 = q_2)$
}

`aux bombasAlrededor`$(i, j : \mathbb{Z}, b : \text{Buscaminas}) : \mathbb{Z} =$

$$\sum_{k=i-1}^{i+1} \left( \sum_{l=j-1}^{j+1} IfThenElse(enRango(k, l, b.bombas) \land_L b.bombas[k][l] = true, 1, 0) \right)$$

`proc bombasAlrededor`$(\mathsf{in}\ b : \text{Buscaminas}, \mathsf{in}\ i, j : \mathbb{Z}) : \mathbb{Z}$ {
**requiere** $\{ enRango(i, j, b.bombas) \land_L b.descubiertas[i][j] = true \land b.bombas[i][j] = false \}$
**asegura** $\{ res = bombasAlrededor(i, j, b) \}$
}

`proc flipBanderín`$(\mathsf{inout}\ b : \text{Buscaminas}, \mathsf{in}\ i, j : \mathbb{Z})$ {
**requiere** $\{ b = B0 \land enRango(i, j, b.banderines) \land_L b.descubiertas[i][j] = false \}$
**asegura** {
$b.bombas = B0.bombas \land$
$b.descubiertas = B0.descubiertas \land$
$mismasDimensiones(b.banderines, B0.banderines) \land_L$ (
$b.banderines[i][j] = \neg B0.banderines[i][j] \land otrasBanderinesSeMantienen(B0, b, i, j)$
)
}
}

`pred otrasBanderinesSeMantienen`$(B0, b : Buscaminas, i, j : \mathbb{Z})$ {
$(\forall k, l : \mathbb{Z})$ $((enRango(k, l, b.banderines) \land k \neq i \lor l \neq j) \to_L b.banderines[i][j] = B0.banderines[i][j])$
}
}