DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1: especificación de TAD's

Algorítmos y Estructuras de Datos

20 de abril de 2025 — Materia de la carrera

Grupo6

Integrante	LU	Correo electrónico
Abizanda, Facundo	1332/21	facundoabizanda@gmail.com
Julian, Alvarez	1944/21	juroalwi@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

Especificación de Berretacoin

Transaccion **ES** Struct$\langle id_tr : \mathbb{Z},\ id_com : \mathbb{Z},\ id_ven : \mathbb{Z},\ mont : \mathbb{Z} \rangle$

Transacciones **ES** Seq$\langle Transaccion \rangle$

Bloque **ES** Struct$\langle id_bl : \mathbb{Z},\ transacciones : Transacciones \rangle$

Cadena **ES** Seq$\langle Bloque \rangle$

TAD $Berretacoin{
obs $cadena : Cadena$

pred balanceUsuarioPositivoEnCadena(id_usuario : $\mathbb{Z}$, cadena: Cadena){

$$(\forall\, i : \mathbb{Z})(0 \leq i < |cadena| \ \longrightarrow_L$$
$$(\forall\, num\_tr : \mathbb{Z})(0 \leq num\_tr < |cadena[i].tr| \ \longrightarrow_L$$
$$($$
$$\text{balanceUsuarioHastaBloque}(id\_usuario,\ cadena[i].id\_bl - 1,\ cadena) +$$
$$\text{balanceUsuarioEnTxsHasta}(id\_usuario,\ num\_tr,\ cadena[i].tr)$$
$$) \geq 0$$
$$) \qquad)$$

}

aux balanceUsuarioEnTxsHasta(id_usuario: $\mathbb{Z}$, num_tr: $\mathbb{Z}$, tr: Transacciones) : $\mathbb{Z}$ =

$$\Big(\sum_{i=0}^{num\_tr} \text{if } tr[i].id\_ven = id\_usuario \text{ then } tr[i].mont \text{ else } 0 \text{ fi} \Big) \ -$$
$$\Big(\sum_{i=0}^{num\_tr} \text{if } tr[i].id\_com = id\_usuario \text{ then } tr[i].mont \text{ else } 0 \text{ fi} \Big)$$

aux balanceUsuarioHastaBloque(id_usuario: $\mathbb{Z}$, hasta_bloque: $\mathbb{Z}$, cadena: Cadena) : $\mathbb{Z}$ =

$$\sum_{j=0}^{|cadena|-1} \left(\begin{array}{l} \textbf{if } cadena[j].id\_bl \leq hasta\_bloque \\ \textbf{then } \text{balanceUsuarioEnTxsHasta}(id\_usuario,\ |cadena[j].tr| - 1,\ cadena[j].tr) \\ \textbf{else } 0 \textbf{ fi} \end{array} \right)$$

pred transaccionValida(id_bl: $\mathbb{Z}$, es_creacion: Bool, tr: Transaccion){

$$(tr.id\_tr \geq 0) \ \wedge \ (tr.id\_ven > 0\,) \ \wedge \ (tr.id\_com \neq tr.id\_ven) \ \wedge \ (tr.mont > 0) \wedge$$
$$($$
$$(\ es\_creacion \ \longrightarrow_L \ tr.id\_com = 0 \ \wedge \ tr.mont = 1) \wedge$$
$$(\ \neg es\_creacion \ \longrightarrow_L \ tr.id\_com > 0)$$
$$)$$

}

pred `bloqueValido`(bl : Bloque, cadena: Cadena){

$(bl.id\_bl \geq 0) \wedge$

$(estaEnPrimerosTresmil(bl.id\_bl, cadena) \longrightarrow_L |bl.tr| \geq 1) \wedge$

$(|bl.tr| \leq 50) \wedge$

$(\forall\, i : \mathbb{Z})\big(0 \leq i < |bl.tr| \longrightarrow_L$
transaccionValida$(bl.id_bl,\ esTrDeCreacion(bl.tr[i].id_tr, bl, cadena),\ bl.tr[i])$
$) \wedge$

$(\forall\, i : \mathbb{Z})\big(0 \leq i < |bl.tr| - 1 \longrightarrow_L$
$bl.tr[i].id\_tr < bl.tr[i+1].id\_tr$
$)$

}

pred `esTrDeCreacion`(id_tr: $\mathbb{Z}$, bl: Bloque, cadena: Cadena){

$estaEnPrimerosTresmil(bl.id\_bl, cadena) \wedge id\_tr = bl.tr[0].id\_tr$

}

pred `estaEnPrimerosTresmil`(id_bl: $\mathbb{Z}$, cadena: Cadena){

$(\exists j : \mathbb{Z})(0 \leq j < |cadena| \wedge_L$
$esPrimerIdBloques(cadena[j].id\_bl, cadena) \wedge id\_bl < cadena[j].id\_bl + 3000)$

}

pred `cadenaValida`(cadena : Cadena){

$(\forall\, i : \mathbb{Z})(0 \leq i < |cadena| \longrightarrow_L$ (bloqueValido$(cadena[i], cadena)) \wedge$

$(\forall\, i : \mathbb{Z})($
$(0 \leq i < |cadena| \wedge_L \neg esUltimoIdBloques(cadena[i].id\_bl, cadena)) \longrightarrow_L$
$(\exists\, j : \mathbb{Z})(0 \leq j < |cadena| \ \wedge_L\ cadena[j].id\_bl = cadena[i].id\_bl + 1)$
$)$

}

pred `esUltimoIdBloques`(id_bloque: $\mathbb{Z}$, cadena : Cadena){

$(\forall\, i : \mathbb{Z})(0 \leq i < |cadena| \longrightarrow_L (cadena[i].id\_bl \leq id\_bloque))$

}

pred `esIesimoIdBloques`(id_bloque: $\mathbb{Z}$, i: $\mathbb{Z}$, cadena: Cadena){

$(\exists\, j : \mathbb{Z})(0 \leq j < |cadena| \ \wedge_L esPrimerIdBloques(j) \wedge cadena[j].id\_bloque + i = id\_bloque)$

}

pred `esPrimerIdBloques`(id_bloque: $\mathbb{Z}$, cadena : Cadena){

$(\forall\, i : \mathbb{Z})(0 \leq i < |cadena| \longrightarrow_L (cadena[i].id\_bl \geq id\_bloque))$

}

pred `vendedoresDistintosEnCreacion`(cadena : Cadena){

$(\forall\, i : \mathbb{Z})(0 \leq i < (\textsf{if } |cadena| \leq 3000 \textsf{ then } |cadena| \textsf{ else } 3000 \textsf{ fi}) \longrightarrow_L$
$(\forall\, j : \mathbb{Z})(0 \leq j < (\textsf{if } |cadena| \leq 3000 \textsf{ then } |cadena| \textsf{ else } 3000 \textsf{ fi}) \ \wedge_L\ i \neq j \longrightarrow$
$\text{head}(cadena[i].tr).id\_ven \neq \text{head}(cadena[j].tr).id\_ven$
$)$
$)$

}

aux `sumaMontosTrs`(trs: Transacciones, desde: $\mathbb{Z}$) : $\mathbb{Z}$ =

$$\sum_{n=desde}^{|trs|} trs[n].mont$$

aux `sumaMontosBloques`(cadena: Cadena) : $\mathbb{Z}$ =

$$\sum_{n=0}^{|cadena|} \left(\begin{array}{l} \textbf{if } estaEnPrimerosTresmil(cadena[i].id\_bl, cadena) \\ \textbf{then } \text{sumaMontosTrs}(cadena[i].tr, 1) \\ \textbf{else } \text{sumaMontosTrs}(cadena[i].tr, 0) \textbf{ fi} \end{array} \right)$$

aux `cantidadTrsCadena`(cadena: Cadena) : $\mathbb{Z}$ =

$$\sum_{n=0}^{|cadena|} |cadena[i].tr| - \text{if } estaEnPrimerosTresmil(cadena[i].id\_bloque, cadena) \text{ then } 1 \text{ else } 0 \text{ fi}$$

proc nuevaBerretacoin() : Berretacoin{

requiere {

$True$

}

asegura {

$res.cadena = \langle\rangle$

}

}

proc maximosTenedores(in berretacoin : Berretacoin) : **seq** $\langle \mathbb{Z} \rangle$ {

requiere {

cadenaValida(*berretacoin.cadena*)

}

requiere {

$(\forall\, i : \mathbb{Z})(i > 0 \ \longrightarrow_L$

balanceUsuarioPositivoEnCadena(i, *berretacoin.cadena*))

}

asegura {

$(\forall\, i : \mathbb{Z})(0 \leq i < |res| \ \longrightarrow_L$

$(\forall\, usuario : \mathbb{Z})(usuario > 0 \longrightarrow_L$

$res[i] \geq$ balanceUsuarioHastaBloque($usuario$, $|berretacoin.cadena| - 1$, *berretacoin.cadena*)

)

)

}

}

proc montoMedio(in berretacoin : Berretacoin) : $\mathbb{Z}${

requiere {

cadenaValida(*berretacoin.cadena*)

}

requiere {

$(\forall\, i : \mathbb{Z})(i > 0 \ \longrightarrow_L$

balanceUsuarioPositivoEnCadena(i, *berretacoin.cadena*))

}

asegura {

$res = sumaMontosBloques(berretacion.cadena)/cantidadTrsCadena(berretacion.cadena)$

}

}

```
proc cotizacionAPesos(in berretacoin : Berretacoin, in cotizaciones : seq ⟨ℤ⟩ ) : seq ⟨ℤ⟩ {
    requiere {
        cadenaValida(berretacoin.cadena)
    }
    requiere {
        (∀ i : ℤ)(i > 0 ⟶_L
          balanceUsuarioPositivoEnCadena(i, berretacoin.cadena))
    }
    requiere {
        |berretacoin.cadena| = |cotizaciones|
    }
    requiere {
        (∀cotizacion ∈ cotizaciones)(cotizacion > 0)
    }
    asegura {
        (∀ i : ℤ)(0 ≤ i < |cotizaciones| ⟶_L
            ((∃ bloque ∈ berretacoin.cadena)(esIesimoIdBloque(bloque.id_bl, i, berretacoin.cadena) ∧_L
            res[i] = sumaMontosTrs(bloque.tr, 0) * cotizaciones[i]))
        )
    }
}
```

```
proc agregarBloque(inout berretacoin : Berretacoin,  in bloque : Bloque ){
    requiere {
        berretacoin = B0
    }
    requiere {
        cadenaValida(berretacoin.cadena)
    }
    requiere {
        (∃ i : ℤ)(0 ≤ i < |berretacoin.cadena| ∧_L berretacoin.cadena[i].id_bl + 1 = bloque.id_bloque)
    }
    requiere {
        (∀ i : ℤ)(i > 0 ⟶_L
          balanceUsuarioPositivoEnCadena(i, berretacoin.cadena ++ bloque))
    }
    requiere {
        bloqueValido(bloque, berretacoin.cadena)
    }
    asegura {
        (bloque ∈ berretacoin.cadena) ∧ ((∀ bl ∈ B0)(bl ∈ berretacoin.cadena)) ∧
        (|berretacoin.cadena| = |B0| + 1)
    }
    asegura {
        cadenaValida(berretacoin.cadena)
    }
    asegura {
        (∀ i : ℤ)(0 < i ⟶_L
          balanceUsuarioPositivoEnCadena(i, berretacoin.cadena)
    }
}
```