



Trabajo Práctico 1: Especificación de TADs

Cómo funcionan las Blockchains: \$BerretaCoin

20 de abril de 2025

Algoritmos y Estructuras de Datos

Grupo: Ameri

Integrante	LU	Correo electrónico
Falbo, Tiziana	863/23	tfalbo@dc.uba.ar
Herrera, Facundo	1175/22	facundoherreracp@gmail.com
Marsico, Bautista	1001/24	bautimarsico@gmail.com
Rankov, Jorge	714/23	jrankov@dc.uba.ar



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

bloque ES struct(transacciones: seq<struct (idTransaccion: \mathbb{Z} , idComprador: \mathbb{Z} , idVendedor: \mathbb{Z} , monto: \mathbb{Z})>, idBloque: \mathbb{Z})

TAD \$BerretaCoin {

obs blockchain: Seq<bloque>

```
proc nuevoBerretaCoin(): BerretaCoin {  
  asegura { res.blockchain = <> }  
}
```

```
proc agregarBloque (inout B: BerretaCoin, in S: bloque) {  
  requiere {  $B = B_0 \wedge (0 < |S.transacciones| \leq 50) \wedge \text{bloqueValido}(B, S) \wedge \text{sonTransaccionesValidas}(S, B)$  }  
  asegura { B.blockchain =  $B_0.blockchain ++ \{S\}$  }  
}
```

```
proc maximosTenedores (in B: BerretaCoin): Seq< $\mathbb{Z}$ > {  
  asegura {  $(\forall i: \mathbb{Z})(0 \leq i < |res|) \rightarrow_L \text{esUsuario}(res[i], B)$  }  
  asegura {  $(\forall id: \mathbb{Z})(id \in res) \rightarrow_L \neg (\exists \text{otro: } \mathbb{Z})(\text{esUsuario}(\text{otro}, B) \wedge$   
     $(\text{montoDeUsuario}(\text{otro}, B) > \text{montoDeUsuario}(id, B)))$  }  
  asegura {  $(\forall i: \mathbb{Z})(\forall j: \mathbb{Z})(0 \leq i < |res|) \wedge_L (0 \leq j < |res|) \wedge_L (i \neq j) \rightarrow res[i] \neq res[j]$  }  
}
```

```
proc montoMedio (in B: BerretaCoin): Float {  
  requiere {  $|B.blockchain| \neq 0$  }  
  asegura { res = promedio (B.blockchain) }  
}
```

```
proc cotizacionAPesos (in cotizaciones: Seq< $\mathbb{Z}$ >, in B: BerretaCoin): Seq< $\mathbb{Z}$ > {  
  requiere {  $(\forall C \in \text{cotizaciones}) \rightarrow (C > 0)$  }  
  requiere {  $|\text{cotizaciones}| = |B.blockchain|$  }  
  asegura {  $|res| = |\text{cotizaciones}|$  }  
  asegura {  $(\forall i: \mathbb{Z})(0 \leq i < |res|) \rightarrow_L res[i] = \text{cotizarBloque}(i, \text{cotizaciones}, B.blockchain)$  }  
}
```

Predicados

```
pred bloqueValido (B: BerretaCoin, S: bloque) {  
  (|B.blockchain| < 3000) → S.transacciones[0].idComprador = 0 ∧  
  (|B.blockchain| ≥ 3000) → S.transacciones[0].idComprador ≠ 0 ∧  
  (∀ transaccion ∈ S.transacciones) →L [(transaccion.idComprador ≠ transaccion.idVendedor) ∧  
  (transaccion.id > 0 ∧ transaccion.idComprador ≥ 0 ∧ transaccion.idVendedor > 0 ∧ transaccion.monto > 0)] ∧  
  estaOrdenada(S) ∧ esIdDeBloqueConsecutivo(B, S)  
}
```

```
pred estaOrdenada (S: bloque) {  
  (∀i: ℤ) (0 ≤ i < |S.transacciones|) →L (S.transacciones[i].idTransaccion = i)  
}
```

```
pred esIdDeBloqueConsecutivo (B: BerretaCoin, S: bloque) {  
  (|B.blockchain| = 0) → (S.idBloque = 0) ∧  
  (|B.blockchain| ≠ 0) → (S.idBloque = B.blockchain[|B.blockchain|-1].idBloque + 1)  
}
```

```
pred sonTransaccionesValidas (S: bloque, B: BerretaCoin) {  
  (∀id:ℤ)(∀j:ℤ) (0 ≤ j < |S.transacciones|) ∧L [esUsuario(id, B) ∨ esUsuarioDeBloque(id, S)]  
    → montoDeUsuarioHastaTransaccion(id, S, B, j) ≥ 0  
}
```

```
pred esUsuario (id: ℤ, B: BerretaCoin) {  
  (∃i: ℤ)(∃j: ℤ) (0 ≤ i < |B.blockchain| ∧L 0 ≤ j < |B.blockchain[i].transacciones|) ∧L  
    (id = B.blockchain[i].transacciones[j].idVendedor)  
}
```

```
pred esUsuarioDeBloque (S: bloque, id: ℤ) {  
  (∃i: ℤ) (0 ≤ i < |s|) ∧L (S.transacciones[i].idVendedor = id)  
}
```

Auxiliares

aux MontoDeUsuario (id: \mathbb{Z} ; B: BerretaCoin): \mathbb{Z} =

$$\sum_{j=0}^{|B.blockchain|-1} \left[\left(- \sum_{i=0}^{|B.blockchain[j].transacciones|-1} (ifThenElse(B.blockchain[j].transacciones[i].idComprador = id, B.blockchain[j].transacciones[i].monto, 0)) \right) \right]$$

+

$$\left(\sum_{i=0}^{|B.blockchain[j].transacciones|-1} (ifThenElse(B.blockchain[j].transacciones[i].idVendedor = id, B.blockchain[j].transacciones[i].monto, 0)) \right)$$

aux montoDeUsuarioHastaTransaccion (id: \mathbb{Z} , S: bloque, B: BerretaCoin, pos: \mathbb{Z}) =

montoDeUsuario(id, B) +

$$\left[\left(- \sum_{i=0}^{pos} (ifThenElse(S.transacciones[i].idComprador = id, S.transacciones[i].monto, 0)) \right) \right]$$

+

$$\left(\sum_{i=0}^{pos} (ifThenElse(S.transacciones[i].idVendedor = id, S.transacciones[i].monto, 0)) \right)$$

aux montoTotalDeTransacciones (bloques: seq<bloque>): \mathbb{Z}

$$\sum_{j=0}^{|bloques|-1} \left(\sum_{i=0}^{|bloques[j].transacciones|-1} (ifThenElse(bloques[j].transacciones[i].idComprador \neq 0), bloques[j].transacciones[i].monto, 0)) \right)$$

aux cantidadTotalDeTransacciones (bloques: seq<bloque>): \mathbb{Z}

$$\sum_{j=0}^{|bloques|-1} \left(\sum_{i=0}^{|bloques[j].transacciones|-1} (ifThenElse(bloques[j].transacciones[i].idComprador \neq 0, 1, 0)) \right)$$

aux promedio (bloques: seq<bloque>): Float =

$$\frac{montoTotalDeTransacciones(bloques)}{cantidadTotalDeTransacciones(bloques)}$$

aux cotizarBloque (posicion: \mathbb{Z} , cotizaciones: Seq< \mathbb{Z} >, blockchain: Seq<bloque>): \mathbb{Z} =

$$\left(\sum_{i=0}^{|blockchain[posicion]|-1} blockchain[posicion].transacciones[i].montos \right) * cotizaciones[posicion]$$

}