DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo práctico

Especificacion de TADs

September 11, 2025 — Algoritmos y Estructuras de Datos

BobElConstructorPorCopia

Integrante	LU	Correo electrónico
Choque, Leandro	252/25	leandroch2002@gmail.com
Musi, Santino	965/24	santinomusi1@gmail.com
Rojas, Damian	209/25	dam.rojas1@gmail.com
Martell, Juan Bautista	622/25	Juanbamartell@hotmail.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

1 Supongo que acá iría una descripción

Breve descripción.

Luego veremos bien el formato, esto de momento es para tener un esqueleto.

2 Especificacion

TAD EdR {

obs aula : $Aula$

obs solucion : $Paso$

obs entregas : seq$\langle Alumno \rangle$

proc EdR(in $dimensionAula : \mathbb{Z}$, in $s : Paso$, in $cantEstudiantes : \mathbb{Z}$) : EdR {

requiere {

$(dimensionAula > 0) \wedge_L$

$rtaValida(s) \wedge_L$

$cantValidaEstudiantes(dimensionAula, cantEstudiantes)$

}

asegura {

$(|res.aula| = dimensionAula) \wedge_L$

$aulaCuadrada(res.aula) \wedge_L$

$noHayAlumnosJuntos(res.aula) \wedge_L$

$(cuantosEstudiantes(res.aula) = cantEstudiantes) \wedge_L$

$examenesSinResponder(res.aula) \wedge_L$

$(res.solucion = s) \wedge_L$

$(res.entregas = \langle\rangle)$

}

}

pred rtaValida($s : Paso$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |s| \rightarrow_L s[i] \in \text{conj}\langle$"0","1","2","3","4","5","6","7","8","9"$\rangle)$

}

pred cantValidaEstudiantes($a : Aula, e : \mathbb{Z}$) {

$(e \leq ifThenElseFi(mod(|a|, 2) == 0, \frac{|a|^2}{2}, \frac{|a|+1}{2} * |a|))$

}

pred aulaCuadrada($a : Aula$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |a| \rightarrow_L |a[i]| = |a|)$

}

pred noHayAlumnosJuntos($a : Aula$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |a| \rightarrow_L$

$(\forall j : \mathbb{Z})\ (0 \leq j < |a[i]| - 1 \rightarrow_L (a[i][j] \neq \langle\rangle \rightarrow a[i][j+1] = \langle\rangle)))$

}

aux cuantosEstudiantes($a : Aula$) : $\mathbb{Z} = \sum_{i=0}^{|a|-1} \sum_{j=0}^{|a[i]|-1} ifThenElseFi(a[i][j] \neq \langle\rangle, 1, 0)$

pred examenesSinResponder($a : Aula$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |a| \rightarrow_L (\forall j : \mathbb{Z})\ (0 \leq j < |a[i]| \rightarrow_L examenSinResponder(a[i][j].examen)))$

}

pred examenSinResponder($e : Examen$) {

$(|e| = 1 \wedge (\forall i : \mathbb{Z})\ (0 \leq i < |e| \rightarrow e[0][i] =$ ""$))$

}

proc igualdad(in $edr1, edr2 : EdR,$) : Bool {

```
   requiere { True }
   asegura {
        (res = True) ↔
        (edr1.aula = edr2.aula) ∧
        (edr1.solucion = edr2.solucion) ∧
        (edr1.entregas = edr2.entregas)
   }
}
proc copiarse(in alumno : Alumno, inout edr : EdR) : {
   requiere {
        (perteneceAlumno(alumno, edr.aula) ∧_L
        edr = edr0 ∧_L
        alumno.Cordenada.x = x0 ∧_L
        alumno.Cordenada.y = y0∧_L
   }
   asegura {
        (∃i, j : ℤ)(cordenadaValidas(i, j, edr.Aula) ∧_L
        adyacente(< i, j >, alumno.Cordenada) ∧_L
        (∃k : ℤ)(alumno.examen[|alumno.examen| − 1][k] = "" ∧
        edr.Aula[i][j].examen[|examen|][k] ≠ "" ∧
        edr.Aula[i][j].examen[|examen|][k] = edr0.Aula[x0][y0].examen[|alumno.examen|][k])∧_L

        (∀p : ℤ) ((∀q : ℤ) (cordenadaValida(p, q, edr0.Aula) ∧_L
        (x0 ≠ p ∧ y0 ≠ q) →_L edr.Aula[p][q] = edr0.Aula[p][q])) ∧_L
        edr0.solucion = edr.solucion ∧_L
        edr0.entregas = edr.entregas
   }
}
pred perteneceAlumno(e : alumno, a : Aula) {
     (∃i, j : ℤ)(cordenadaValida(< i, j >, edr.aula) ∧_L
     alumno.Cordenada.x = i ∧ alumno.Cordenada.y = j ∧_L
     alumno.examen = a[i][j].examen)
}
pred cordenadaValida(c : Cordenada, a : Aula) { 0 ≤ c.x < |a| ∧ 0 ≤ c.y < |a| }

pred adyacente(c0, c1 : Cordenada, a : Aula) {
     cordenadaValida(c0, a) ∧ cordenadaValida(c1, a) ∧
     (c0.x ≠ c1.x ∧ c0.y ≠ c1.y) ∧
     ((−2 ≤ c1.x − c0.x ≤ 2 ∧ c1.y − c0.y = 0) ∨
     (c1.x − c0.x = 0 ∧ 0 < c1.x − c0.x ≤ 2))
}
proc consultarDarkWeb(in s : Paso, in posiblesAccesos : ℤ, inout edr : EdR) {
   requiere {
        rtaValida(s) ∧_L
        posiblesAccesos ≥ 0∧_L
   }
   asegura { res }
}
proc resolver(in alumno : Cordenada, inout edr : EdR) : Paso {
```

```
        requiere {
            (existeAlumno(alumno)) ∧_L
            (∃i : ℤ)(0 ≤ i < |alumno.examen[0]|∧_L alumno.examen[|alumno.examen|][k] ==
            "")
        }
        asegura { res }
    }
    proc entregar(in alumno : Cordenada) : EdR {
        requiere { True }
        asegura { res }
    }
    proc chequearCopias(in alumnos : seq < Alumno >) : seq < Alumno > {
        requiere { True }
        asegura { res }
    }
    proc corregir(inout edr, EdR) : seq << Alumno, Nota >> {
        requiere { True }
        asegura { res }
    }
}
```