

Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2020

Versión 2: 6 de noviembre de 2020

TPI - “Juego de la vida toroidal”

Fecha de entrega: 13 de noviembre de 2020 (hasta las 17hs)

Grupos: deben ser de exactamente 2 integrantes

Cambios versión 3

1. *primosLejanos* Se agregó la precondition *sonPeriodicosOMueren(t1,t2)*.

Cambios versión 2

1. *esPeriodico* Se cambió la precondition *esToroide(t) ∧ p = P₀* por *esToroide(t) ∧ p = P₀ ∧ (cumpleEvolucionCiclica(t) ∨ eventualmenteMuere(t))*
2. *test en SeleccionNatural* Uno de los tests que les dimos para seleccionNatural no cumple la precondition (porque es periódico). Es el segundo test, por favor ignoren ese caso al hacer sus pruebas.

1. Ejercicios

1. Implementar las funciones especificadas en la sección **Especificación**. La implementación de cada ejercicio DEBE SEGUIR OBLIGATORIAMENTE ESTA ESPECIFICACIÓN.
2. No está permitido el uso de librerías de C++ fuera de las clásicas: math, vector, tuple, pair, las de input-output, etc. Consultar con la cátedra por cualquier librería adicional. Implementar los tests necesarios para asegurar el correcto funcionamiento de los funciones. Encontrarán dentro de los tests un ejemplo para cada una de las funciones los cuales **No pueden modificar**.
3. Completar (agregando) los tests estructurales necesarios para cubrir todas las líneas del archivo *solucion.cpp*. Utilizar la herramienta de cobertura de código de CLion para dicha tarea.

2. Compilación

1. Bajar del campus de la materia Archivos TPI.
2. Descomprimir el ZIP.
3. Seleccionar File → Open... y seleccionar la carpeta con el nombre toroide que se encuentra en TPI.
4. Dentro del archivo que se descarguen desde la página de la materia van a encontrar los siguientes archivos y carpetas:
 - **definiciones.h**: Aquí están las definiciones de los tipos del TPI.
 - **ejercicios.cpp**: Aquí es donde van a volcar sus implementaciones.
 - **ejercicios.h**: *headers* de las funciones que tienen que implementar.
 - **auxiliares.cpp** y **auxiliares.h**: Donde es posible volcar funciones auxiliares.
 - **main.cpp**: Punto de entrada del programa.
 - **tests**: Estos son algunos Tests Suites provistos por la materia. Aquí deben completar con sus propios Tests para lograr la cobertura pedida.
 - **lib**: Todo lo necesario para correr Google Tests. Aquí no deben tocar nada.
 - **CMakeLists.txt**: Archivo que necesita CLion para la compilación y ejecución del proyecto. **NO** deben sobrescribirlo al importar los fuentes desde CLion.

Es importante recalcar que la especificación de los ejercicios elaborada por la materia es la guía sobre la que debe basarse el equipo a la hora de implementar los problemas.

3. Términos y condiciones

El trabajo práctico se realiza de manera grupal con grupos de exactamente 2 personas. Para aprobar el trabajo se necesita:

- Que todos los ejercicios estén resueltos.
- Que las soluciones sean correctas.
- Que todos los tests provistos por la cátedra funcionen (no pueden ser modificados).
- Que las soluciones sean prolijas: evitar repetir implementaciones innecesariamente y usar adecuadamente funciones auxiliares.
- Que los test cubran todas las líneas de las funciones.

4. Pautas de Entrega

La fecha de entrega del TPI es el **13 de Noviembre de 2020 (hasta las 17hs)**.

El trabajo debe ser subido al campus en la sección Trabajos Prácticos en la fecha estipulada.

La entrega debe contener:

- El proyecto completo, se requiere que estén todos los archivos necesarios para compilarlo.
- Los tests necesarios que garanticen el cubrimiento de líneas de código.
- **Importante: Utilizar la especificación diseñada para este TP.**

5. C++

Uso de pair

En especificación tenemos el tipo $\mathbb{Z} \times \mathbb{Z}$, el cual implementaremos en C++ con `pair`. El container `pair` pertenece a la librería estándar de C++ y está definido en el header `<utility>`. Este es un container permite agrupar dos elementos de cualquier tipo. En nuestro caso, vamos a utilizarlo para dos valores enteros.

Para asignar o acceder al primero se utiliza la propiedad `first`, y para el segundo, `second`. Pueden ver ejemplos en <https://www.geeksforgeeks.org/in-cpp-stl/>

Uso de % (módulo)

En especificación usamos `mod` para obtener el resto de la división de un número por otro. El C++ podemos utilizar el operador `%` para tal fin. Se debe tener en cuenta que este operador tiene ciertas restricciones sobre los parámetros que recibe. Pueden ver ejemplos y leer sobre dichas restricciones en: <https://www.geeksforgeeks.org/modulo-operator-in-c-cpp-with-examples/>

6. Especificación

Implementar funciones en C++ que cumplan las siguientes especificaciones respetando el **renombré** de tipo:

```
type toroide = seq<seq<Bool>>
type rectangulo = seq<seq<Bool>>
```

6.1. Ejercicios

Ejercicio 1.

```
proc toroideValido (in t: toroide, out result: Bool) {
  Pre {True}
  Post {result = true ↔ esToroide(t)}
  pred esToroide (t: toroide) {
```

```

    esRectangulo(t)  $\wedge$  filas(t)  $\geq$  3  $\wedge$  columnas(t)  $\geq$  3
  }
  pred esRectangulo (r: seq<seq<Bool>>) {
    filas(r) > 0  $\wedge$  columnas(r) > 0  $\wedge$  ( $\forall f : \mathbb{Z}$ )(0  $\leq$  f < |r|  $\rightarrow_L$  |r[f]| = |r[0]|)
  }
  aux filas (t: seq<seq<Bool>>) :  $\mathbb{Z}$  = |t|;
  aux columnas (t: seq<seq<Bool>>) :  $\mathbb{Z}$  = if filas(t) > 0 then |t[0]| else 0 fi;
}

```

Ejercicio 2.

```

proc toroideMuerto (in t: toroide, out result: Bool) {
  Pre {esToroide(t)}
  Post {result = true  $\leftrightarrow$  ( $\forall f : \mathbb{Z}$ )( $\forall c : \mathbb{Z}$ )(enRangoToroide(f, c, t)  $\rightarrow_L$  t[f][c] = false)}
  pred enRangoToroide (f:  $\mathbb{Z}$ , c:  $\mathbb{Z}$ , t: seq<seq<Bool>>) {
    enRango(f, t)  $\wedge_L$  enRango(c, t[f])
  }
  pred enRango (i:  $\mathbb{Z}$ , s: seq<T>) {
    0  $\leq$  i < |s|
  }
}

```

Ejercicio 3.

```

proc posicionesVivas (in t: toroide, out vivas: seq< $\mathbb{Z} \times \mathbb{Z}$ >) {
  Pre {esToroide(t)}
  Post {sinRepetidos(vivas)  $\wedge$  posicionesValidas(vivas, t)  $\wedge_L$  estanLasVivas(vivas, t)  $\wedge$  cantidadVivas(t) = |vivas|}
  pred sinRepetidos (s: seq<T>) {
    ( $\forall i : \mathbb{Z}$ )(enRango(i, s)  $\rightarrow_L$  #apariciones(s, s[i]) = 1)
  }
  pred posicionesValidas (listapos: seq< $\mathbb{Z} \times \mathbb{Z}$ >, t: toroide) {
    ( $\forall p : \mathbb{Z} \times \mathbb{Z}$ )(p  $\in$  listapos  $\rightarrow_L$  enRangoToroide(p0, p1, t))
  }
  pred estanLasVivas (vivas: seq< $\mathbb{Z} \times \mathbb{Z}$ >, t: toroide) {
    ( $\forall p : \mathbb{Z} \times \mathbb{Z}$ )(p  $\in$  vivas  $\leftrightarrow$  estaViva(p0, p1, t))
  }
  pred estaViva (f:  $\mathbb{Z}$ , c:  $\mathbb{Z}$ , t: seq<seq<Bool>>) {
    enRangoToroide(f, c, t)  $\wedge_L$  t[f][c] = true
  }
  aux cantidadVivas (t: seq<seq<Bool>>) :  $\mathbb{Z}$  =  $\sum_{f=0}^{filas(t)-1} \sum_{c=0}^{columnas(t)-1}$  if estaViva(f, c, t) then 1 else 0 fi;
}

```

Ejercicio 4.

```

proc densidadPoblacion (in t: toroide, out result :  $\mathbb{R}$ ) {

```

```

Pre {esToroide(t)}
Post {result = cantidadVivas(t)/superficieTotal(t)}
aux superficieTotal (t: toroide) :  $\mathbb{Z}$  = filas(t) * columnas(t);
}

```

Ejercicio 5.

```

proc cantidadVecinosVivos (in t: toroide, In f: $\mathbb{Z}$ , In c: $\mathbb{Z}$ , Out result:  $\mathbb{Z}$ ) {
  Pre {esToroide(t)  $\wedge$  enRangoToroide(f, c, t)}
  Post {result = vecinosVivos(t, f, c)}
  aux vecinosVivos (t: toroide, f:  $\mathbb{Z}$ , c:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  =  $\sum_{i=-1}^1 \sum_{j=-1}^1$  if (i  $\neq$  0  $\vee$  j  $\neq$  0)  $\wedge$  vecinaViva(t, f, c, i, j) then 1 else 0 fi;
  pred vecinaViva (t: toroide, f:  $\mathbb{Z}$ , c:  $\mathbb{Z}$ , i:  $\mathbb{Z}$ , j:  $\mathbb{Z}$ ) {
    vivaToroide(f + i, c + j, t)
  }
}

```

Ejercicio 6.

```

proc evolucionDePosicion (in t: toroide, in pos :  $\mathbb{Z} \times \mathbb{Z}$ , out result : Bool) {
  Pre {esToroide(t)  $\wedge$  enRangoToroide(pos0, pos1, t)}
  Post {result = true  $\leftrightarrow$  debeVivir(t, pos0, pos1)}
  pred debeVivir (t: toroide, f:  $\mathbb{Z}$ , c:  $\mathbb{Z}$ ) {
    (estaViva(f, c, t)  $\rightarrow$  2  $\leq$  vecinosVivos(t, f, c)  $\leq$  3)  $\wedge$  (estaMuerta(f, c, t)  $\rightarrow$  vecinosVivos(t, f, c) = 3)
  }
  pred estaMuerta (f:  $\mathbb{Z}$ , c:  $\mathbb{Z}$ , t: toroide) {
    enRangoToroide(f, c, t)  $\wedge_L$  t[f][c] = false
  }
}

```

Ejercicio 7.

```

proc evolucionToroide (inout t: toroide) {
  Pre {esToroide(T0)  $\wedge$  t = T0}
  Post {esEvolucionToroide(t, T0)}
  pred esEvolucionToroide (tf: toroide, ti: toroide) {
    mismaDimension(tf, ti)  $\wedge_L$  ( $\forall f : \mathbb{Z}$ ) (( $\forall c : \mathbb{Z}$ ) (enRangoToroide(f, c, ti)  $\rightarrow_L$  (estaViva(f, c, tf)  $\leftrightarrow$  debeVivir(ti, f, c))))
  }
  pred mismaDimension (t1: toroide, t2: toroide) {
    filas(t1) = filas(t2)  $\wedge$  columnas(t1) = columnas(t2)
  }
}

```

Ejercicio 8.

```

proc evolucionMultiple (in t: toroide, in k:  $\mathbb{Z}$ , out result: toroide) {
  Pre {esToroide(t)  $\wedge$  k  $\geq$  1}

```

```

Post {esEvolucionNivelK(result, t, k)}
pred esEvolucionNivelK (tf: toroide, ti: toroide, k:  $\mathbb{Z}$ ) {
   $k \geq 0 \wedge_L (\exists s: seq\langle toroide \rangle)(todosValidos(s) \wedge |s| = k + 1 \wedge_L s[0] = ti \wedge sonTicksConsecutivos(s) \wedge s[k] = tf)$ 
}
pred todosValidos (ts: seq<toroide>) {
   $(\forall i: \mathbb{Z})(0 \leq i < |ts| \longrightarrow_L esToroide(ts[i]))$ 
}
pred sonTicksConsecutivos (ts: seq<toroide>) {
   $(\forall i: \mathbb{Z})(0 \leq i < |ts| - 1 \rightarrow_L esEvolucionToroide(ts[i + 1], s[i]))$ 
}
}

```

Ejercicio 9.

```

proc esPeriodico (in t: toroide, inout p:  $\mathbb{Z}$ , out result: Bool) {
  Pre {esToroide(t)  $\wedge$   $p = P_0 \wedge (cumpleEvolucionCiclica(t) \vee eventualmenteMuere(t))$ }
  Post {(cumpleEvolucionCiclica(t)  $\rightarrow$  result = true  $\wedge$  tienePeriodoP(t, p))  $\wedge$ 
  ( $\neg$ cumpleEvolucionCiclica(t)  $\rightarrow$  (result = false  $\wedge$   $p = P_0$ ))}
  pred cumpleEvolucionCiclica (t: toroide) {
     $(\exists k: \mathbb{Z})(k > 0 \wedge esEvolucionNivelK(t, t, k))$ 
  }
  pred eventualmenteMuere (t: toroide) {
     $(\exists k: \mathbb{Z})k > 0 \wedge_L seExtingueEnK(t, k)$ 
  }
  pred seExtingueEnK (t: toroide, k:  $\mathbb{Z}$ ) {
     $(\exists tmuerto: toroide)(esToroide(tmuerto) \wedge mismaDimension(t, tmuerto) \wedge cantidadVivas(tmuerto) = 0 \wedge_L$ 
     $esEvolucionNivelK(tmuerto, t, k) \wedge (\forall k': \mathbb{Z})(0 \leq k' < k \longrightarrow_L \neg esEvolucionNivelK(tmuerto, t, k')))$ 
  }
  pred tienePeriodoP (t: toroide, p:  $\mathbb{Z}$ ) {
     $esEvolucionNivelK(t, t, p) \wedge (\forall q: \mathbb{Z})(1 \leq q < p \rightarrow \neg esEvolucionNivelK(t, t, q))$ 
  }
}

```

Ejercicio 10.

```

proc primosLejanos (in t1: toroide, in t2: toroide, out primos: Bool) {
  Pre {esToroide(t1)  $\wedge$  esToroide(t2)  $\wedge_L mismaDimension(t1, t2) \wedge sonPeriodicosOMueren(t1, t2)$ }
  pred sonPeriodicosOMueren (t1: toroide, t2: toroide) {
     $(cumpleEvolucionCiclica(t1) \vee eventualmenteMuere(t1)) \wedge$ 
     $(cumpleEvolucionCiclica(t2) \vee eventualmenteMuere(t2))$ 
  }
  Post {primos = true  $\leftrightarrow (\exists k: \mathbb{Z})(k > 0 \wedge_L esEvolucionNivelK(t2, t1, k) \vee esEvolucionNivelK(t1, t2, k))$ }
}

```

Ejercicio 11.

```

proc seleccionNatural (in ts: seq<toroide>, out res:  $\mathbb{Z}$ ) {
  Pre  $\{|ts| > 0 \wedge_L \text{ todosValidos}(ts) \wedge \text{ todosVivos}(ts) \wedge \text{ todosSeExtinguen}(ts)\}$ 
  Post  $\{0 \leq res < |ts| \wedge_L \text{ sobreviveATodos}(res, ts)\}$ 
  pred todosVivos (ts: seq<toroide>) {
     $(\forall i : \mathbb{Z})(0 \leq i < |ts| \longrightarrow_L \text{ cantidadVivas}(ts[i]) > 0)$ 
  }
  pred todosSeExtinguen (ts: seq<toroide>) {
     $(\forall i : \mathbb{Z})(0 \leq i < |ts| \longrightarrow_L (\exists k : \mathbb{Z})k > 0 \wedge_L \text{ seExtingueEnK}(ts[i], k))$ 
  }
  pred sobreviveATodos (i:  $\mathbb{Z}$ , ts: seq<toroide>) {
     $(\forall j : \mathbb{Z})(0 \leq j < i \longrightarrow_L \text{ muereAntesEstricto}(ts[j], ts[i])) \wedge$ 
     $(\forall h : \mathbb{Z})(i < h < |ts| \longrightarrow_L \text{ muereAntes}(ts[h], ts[i]))$ 
  }
  pred muereAntesEstricto (tin: toroide, tduracell: toroide) {
     $(\exists k1 : \mathbb{Z})((\exists k2 : \mathbb{Z})(\text{seExtingueEnK}(tin, k1) \wedge \text{seExtingueEnK}(tduracell, k2) \wedge k1 < k2))$ 
  }
  pred muereAntes (tin: toroide, tduracell: toroide) {
     $(\exists k1 : \mathbb{Z})((\exists k2 : \mathbb{Z})(\text{seExtingueEnK}(tin, k1) \wedge \text{seExtingueEnK}(tduracell, k2) \wedge k1 \leq k2))$ 
  }
}

```

Ejercicio 12.

```

proc fusionar (in t1: toroide, in t2: toroide, out res: toroide) {
  Pre  $\{\text{esToroide}(t1) \wedge \text{esToroide}(t2) \wedge \text{mismaDimension}(t1, t2)\}$ 
  Post  $\{\text{esToroide}(res) \wedge \text{mismaDimension}(res, t1) \wedge_L \text{ interseccionVivas}(res, t1, t2)\}$ 
  pred interseccionVivas (tf: toroide, t1: toroide, t2: toroide) {
     $(\forall f : \mathbb{Z})(\forall c : \mathbb{Z})(\text{enRangoToroide}(f, c, tf) \longrightarrow_L (\text{estaViva}(f, c, tf) \leftrightarrow (\text{estaViva}(f, c, t1) \wedge \text{estaViva}(f, c, t2))))$ 
  }
}

```

Ejercicio 13.

```

proc vistaTrasladada (in t1: toroide, in t2: toroide, out res: Bool) {
  Pre  $\{\text{esToroide}(t1) \wedge \text{esToroide}(t2) \wedge_L \text{ mismaDimension}(t1, t2)\}$ 
  Post  $\{res = \text{true} \leftrightarrow \text{esTrasladada}(t1, t2)\}$ 
  pred esTrasladada (t1: toroide, t2: toroide) {
     $(\exists i : \mathbb{Z})(\exists j : \mathbb{Z})(\text{traslacion}(t1, t2, i, j))$ 
  }
  pred traslacion (t1: toroide, t2: toroide, i:  $\mathbb{Z}$ , j:  $\mathbb{Z}$ ) {
     $(\forall f : \mathbb{Z})(\forall c : \mathbb{Z})(\text{enRangoToroide}(f, c, t1) \longrightarrow_L \text{ estaViva}(f, c, t1) \leftrightarrow \text{vivaToroide}(f + i, c + j, t2))$ 
  }
  pred vivaToroide (f:  $\mathbb{Z}$ , c:  $\mathbb{Z}$ , t: toroide) {
     $\text{estaViva}(\text{filaToroide}(f, t), \text{columnaToroide}(c, t), t)$ 
  }
}

```

```

aux filaToroide (f:  $\mathbb{Z}$ , t: toroide) :  $\mathbb{Z} = f \bmod \text{filas}(t)$ ;
aux columnaToroide (c:  $\mathbb{Z}$ , t: toroide) :  $\mathbb{Z} = c \bmod \text{columnas}(t)$ ;
}

```

Ejercicio 14.

```

proc menorSuperficieViva (in t: toroide, out res:  $\mathbb{Z}$ ) {
  Pre {esToroide(t)  $\wedge_L$  cantidadVivas(t) > 0}
  Post {( $\exists sr0 : \text{rectangulo}$ )( $\exists t0 : \text{toroide}$ )(esSubRecToroideValido(sr0,t0,t)  $\wedge_L$ 
    superficieTotal(sr0) = res  $\wedge$  ( $\forall sr1 : \text{rectangulo}$ )( $\forall t1 : \text{toroide}$ )(esSubRecToroideValido(sr1,t1,t)  $\longrightarrow_L$ 
    superficieTotal(sr1)  $\geq$  res))}
  pred esSubRecToroideValido (subrec: rectangulo, tt: toroide, torig: toroide) {
    dimensionesValidasSubRec(subrec,tt)  $\wedge$  esTrasladada(tt,torig)  $\wedge_L$  cubreLosVivos(subrec,tt)
  }
  pred dimensionesValidasSubRec (subrec: rectangulo, t: toroide) {
    esRectangulo(subrec)  $\wedge_L$  filas(subrec)  $\leq$  filas(t)  $\wedge$  columnas(subrec)  $\leq$  columnas(t)
  }
  pred cubreLosVivos (subrec: rectangulo, t: toroide) {
    ( $\exists i : \mathbb{Z}$ )( $\exists j : \mathbb{Z}$ )(desplazamientoValido(i,j,subrec,t)  $\wedge_L$  cantidadVivas(t) = cantidadVivas(subrec))
  }
  pred desplazamientoValido (i:  $\mathbb{Z}$ , j:  $\mathbb{Z}$ , subrec: rectangulo, t: toroide) {
    ( $\forall f : \mathbb{Z}$ )( $\forall c : \mathbb{Z}$ )(enRangoToroide(f,c,subrec)  $\wedge$  enRangoToroide(f+i,c+j,t)
     $\longrightarrow_L$  estaViva(f,c,subrec)  $\leftrightarrow$  estaViva(f+i,c+j,t))
  }
}

```