

Algoritmos y Estructuras de Datos I

Departamento de Computación - FCEyN - UBA

Secuencias

1 / 42

Secuencias

- **Secuencia:** Varios elementos del mismo tipo T , posiblemente repetidos, ubicados en un cierto orden.
- $\text{seq}\langle T \rangle$ es el tipo de las secuencias cuyos elementos son de tipo T .
- T es un tipo arbitrario.
 - Hay secuencias de \mathbb{Z} , de Bool, de Días, de 5-uplas;
 - también hay secuencias de secuencias de T ;
 - etcétera.

2 / 42

Secuencias. Notación

- Una forma de escribir un elemento de tipo $\text{seq}\langle T \rangle$ es escribir términos de tipo T separados por comas, entre $\langle \dots \rangle$.
 - $\langle 1, 2, 3, 4, 1, 0 \rangle$ es una secuencia de \mathbb{Z} .
 - $\langle 1, 1 + 1, 3, 2 * 2, 5 \bmod 2, 0 \rangle$ es otra secuencia de \mathbb{Z} (igual a la anterior).
- La **secuencia vacía** se escribe $\langle \rangle$, cualquiera sea el tipo de los elementos de la secuencia.
- Se puede formar secuencias de elementos de cualquier tipo.
 - Como $\text{seq}\langle \mathbb{Z} \rangle$ es un tipo, podemos armar secuencias de $\text{seq}\langle \mathbb{Z} \rangle$ (secuencias de secuencias de \mathbb{Z} , o sea $\text{seq}\langle \text{seq}\langle \mathbb{Z} \rangle \rangle$).
 - $\langle \langle 12, 13 \rangle, \langle -3, 9, 0 \rangle, \langle 5 \rangle, \langle \rangle, \langle \rangle, \langle 3 \rangle \rangle$ es un elemento de tipo $\text{seq}\langle \text{seq}\langle \mathbb{Z} \rangle \rangle$.

3 / 42

Secuencias bien formadas

Indicar si las siguientes secuencias están bien formadas. Si están bien formadas, indicar su tipo ($\text{seq}\langle \mathbb{Z} \rangle$, etc...)

- $\langle 'H', 'o', 'l', 'a' \rangle$? Bien formada. Tipa como $\text{seq}\langle \text{Char} \rangle$
- $\langle 1, 2, 3, 4, 5 \rangle$? Bien formada. Tipa como $\text{seq}\langle \mathbb{Z} \rangle$ y $\text{seq}\langle \mathbb{R} \rangle$
- $\langle 1, 2, 3, 4, \frac{1}{0} \rangle$? No está bien formada porque uno de sus componentes está indefinido
- $\langle 1, \text{true}, 3, 4, 5 \rangle$? No está bien formada porque no es homogénea (Bool y \mathbb{Z})
- $\langle \text{true}, \text{false}, \text{true}, \text{true} \rangle$? Bien formada. Tipa como $\text{seq}\langle \text{Bool} \rangle$
- $\langle \frac{2}{5}, \pi, e \rangle$? Bien formada. Tipa como $\text{seq}\langle \mathbb{R} \rangle$
- $\langle \rangle$? Bien formada. Tipa como cualquier secuencia $\text{seq}\langle X \rangle$ donde X es un tipo válido.
- $\langle \langle \rangle \rangle$? Bien formada. Tipa como cualquier secuencia $\text{seq}\langle \text{seq}\langle X \rangle \rangle$ donde X es un tipo válido.

4 / 42

Funciones sobre secuencias

Longitud

- ▶ $length(a : seq\langle T \rangle) : \mathbb{Z}$
 - ▶ Representa la longitud de la secuencia a .
 - ▶ Notación: $length(a)$ se puede escribir como $|a|$ o como $a.length$.
- ▶ Ejemplos:
 - ▶ $|\langle \rangle| = 0$
 - ▶ $|\langle 'H', 'o', 'l', 'a' \rangle| = 4$
 - ▶ $|\langle 1, 1, 2 \rangle| = 3$

5 / 42

Funciones con secuencias

I-ésimo elemento

- ▶ Indexación: $seq\langle T \rangle[i : \mathbb{Z}] : T$
 - ▶ Requiere $0 \leq i < |a|$.
 - ▶ Es el elemento en la i -ésima posición de a .
 - ▶ La primera posición es la 0.
 - ▶ Notación: $a[i]$.
 - ▶ Si no vale $0 \leq i < |a|$ se indefine.
- ▶ Ejemplos:
 - ▶ $\langle 'H', 'o', 'l', 'a' \rangle[0] = 'H'$
 - ▶ $\langle 'H', 'o', 'l', 'a' \rangle[1] = 'o'$
 - ▶ $\langle 'H', 'o', 'l', 'a' \rangle[2] = 'l'$
 - ▶ $\langle 'H', 'o', 'l', 'a' \rangle[3] = 'a'$
 - ▶ $\langle 1, 1, 1, 1 \rangle[0] = 1$
 - ▶ $\langle \rangle[0] = \perp$ (Indefinido)
 - ▶ $\langle 1, 1, 1, 1 \rangle[7] = \perp$ (Indefinido)

6 / 42

Funciones con secuencias

Igualdad

Dos secuencias s_0 y s_1 son iguales si y sólo si

- ▶ Tienen la misma cantidad de elementos
- ▶ Dada una posición, el elemento contenido en la secuencia s_0 es igual al elemento contenido en la secuencia s_1 .
- ▶ Notación: $s_0 = s_1$

Ejemplos:

- ▶ $\langle 1, 2, 3, 4 \rangle = \langle 1, 2, 3, 4 \rangle$? Sí
- ▶ $\langle \rangle = \langle \rangle$? Sí
- ▶ $\langle 1, 2, 3, 4, 5 \rangle = \langle 1, 2, 3, 4 \rangle$? No
- ▶ $\langle 1, 2, 3, 4, 5 \rangle = \langle 1, 2, 4, 5, 6 \rangle$? No

7 / 42

Funciones con secuencias

Cabeza o Head

- ▶ Cabeza: $head(a : seq\langle T \rangle) : T$
 - ▶ Es equivalente a la expresión $a[0]$.
 - ▶ Es el primer elemento de la secuencia a .
 - ▶ Requiere $|a| > 0$.
 - ▶ Si no vale $|a| > 0$ se indefine.
- ▶ Ejemplos:
 - ▶ $head(\langle 'H', 'o', 'l', 'a' \rangle) = 'H'$
 - ▶ $head(\langle 1, 1, 1, 1 \rangle) = 1$
 - ▶ $head(\langle \rangle) = \perp$ (Indefinido)

8 / 42

Funciones con secuencias

Cola o Tail

- Cola: $\text{tail}(a : \text{seq}\langle T \rangle) : \text{seq}\langle T \rangle$
 - Es la secuencia resultante de eliminar su primer elemento.
 - Requiere $|a| > 0$.
 - Si no vale $|a| > 0$ se indefine.
- Ejemplos:
 - $\text{tail}(\langle 'H', 'o', 'l', 'a' \rangle) = \langle 'o', 'l', 'a' \rangle$
 - $\text{tail}(\langle 1, 1, 1, 1 \rangle) = \langle 1, 1, 1 \rangle$
 - $\text{tail}(\langle \rangle) = \perp$ (Indefinido)
 - $\text{tail}(\langle 6 \rangle) = \langle \rangle$

9 / 42

Funciones con secuencias

Agregar al principio o addFirst

- Agregar cabeza: $\text{addFirst}(t : T, a : \text{seq}\langle T \rangle) : \text{seq}\langle T \rangle$
 - Es una secuencia con los elementos de a , agregándole t como primer elemento.
 - Es una función que no se indefine
- Ejemplos:
 - $\text{addFirst}('x', \langle 'H', 'o', 'l', 'a' \rangle) = \langle 'x', 'H', 'o', 'l', 'a' \rangle$
 - $\text{addFirst}(5, \langle 1, 1, 1, 1 \rangle) = \langle 5, 1, 1, 1, 1 \rangle$
 - $\text{addFirst}(1, \langle \rangle) = \langle 1 \rangle$

10 / 42

Funciones con secuencias

Concatenación o concat

- Concatenación: $\text{concat}(a : \text{seq}\langle T \rangle, b : \text{seq}\langle T \rangle) : \text{seq}\langle T \rangle$
 - Es una secuencia con los elementos de a , seguidos de los de b .
 - Notación: $\text{concat}(a, b)$ se puede escribir $a ++ b$.
- Ejemplos:
 - $\text{concat}(\langle 'H', 'o' \rangle, \langle 'l', 'a' \rangle) = \langle 'H', 'o', 'l', 'a' \rangle$
 - $\text{concat}(\langle 1, 2 \rangle, \langle 3, 4 \rangle) = \langle 1, 2, 3, 4 \rangle$
 - $\text{concat}(\langle \rangle, \langle \rangle) = \langle \rangle$
 - $\text{concat}(\langle 2, 3 \rangle, \langle \rangle) = \langle 2, 3 \rangle$
 - $\text{concat}(\langle \rangle, \langle 5, 7 \rangle) = \langle 5, 7 \rangle$

11 / 42

Funciones con secuencias

Subsecuencia o subseq

- Subsecuencia: $\text{subseq}(a : \text{seq}\langle T \rangle, d, h : \mathbb{Z}) : \text{seq}\langle T \rangle$
 - Es una sublista de a en las posiciones entre d (inclusive) y h (exclusive).
 - Cuando $0 \leq d = h \leq |a|$, retorna la secuencia vacía.
 - Cuando no se cumple $0 \leq d \leq h \leq |a|$, ¡se **indefine**!
- Ejemplos:
 - $\text{subseq}(\langle 'H', 'o', 'l', 'a' \rangle, 0, 1) = \langle 'H' \rangle$
 - $\text{subseq}(\langle 'H', 'o', 'l', 'a' \rangle, 0, 4) = \langle 'H', 'o', 'l', 'a' \rangle$
 - $\text{subseq}(\langle 'H', 'o', 'l', 'a' \rangle, 2, 2) = \langle \rangle$
 - $\text{subseq}(\langle 'H', 'o', 'l', 'a' \rangle, 3, 1) = \perp$

12 / 42

Funciones con secuencias

- ▶ Cambiar una posición:
 $setAt(a : seq\langle T \rangle, i : \mathbb{Z}, val : T) : seq\langle T \rangle$
 - ▶ Es una secuencia igual a a , pero con valor val en la posición i .
 - ▶ Requiere $0 \leq i < |a|$
- ▶ Ejemplos:
 - ▶ $setAt(\langle 'H', 'o', 'l', 'a' \rangle, 0, 'X') = \langle 'X', 'o', 'l', 'a' \rangle$
 - ▶ $setAt(\langle 'H', 'o', 'l', 'a' \rangle, 3, 'A') = \langle 'H', 'o', 'l', 'A' \rangle$
 - ▶ $setAt(\langle \rangle, 0, 5) = \perp$ (Indefinido)

13 / 42

Operaciones sobre secuencias

Resumen

- ▶ $length(a : seq\langle T \rangle) : \mathbb{Z}$ (notación $|a|$)
- ▶ indexación: $seq\langle T \rangle[i : \mathbb{Z}] : T$
- ▶ igualdad: $seq\langle T \rangle = seq\langle T \rangle$
- ▶ $head(a : seq\langle T \rangle) : T$
- ▶ $tail(a : seq\langle T \rangle) : seq\langle T \rangle$
- ▶ $addFirst(t : T, a : seq\langle T \rangle) : seq\langle T \rangle$
- ▶ $concat(a : seq\langle T \rangle, b : seq\langle T \rangle) : seq\langle T \rangle$ (notación $a++b$)
- ▶ $subseq(a : seq\langle T \rangle, d, h : \mathbb{Z}) : \langle T \rangle$
- ▶ $setAt(a : seq\langle T \rangle, i : \mathbb{Z}, val : T) : seq\langle T \rangle$

14 / 42

Lemas sobre secuencias

Sea s_0, s_1 secuencias de tipo T y e un elemento de tipo T .
Justificar brevemente por qué cada una de las siguientes afirmaciones son verdaderas:

- ▶ $|addFirst(e, s_0)| = 1 + |s_0|$? Sí
- ▶ $|concat(s_0, s_1)| = |s_0| + |s_1|$? Sí
- ▶ $s_0 = tail(addFirst(e, s_0))$? Sí
- ▶ $s_0 = subseq(s_0, 0, |s_0|)$? Sí
- ▶ $s_0 = subseq(concat(s_0, s_1), 0, |s_0|)$? Sí
- ▶ $e = head(addFirst(e, s_0))$? Sí
- ▶ $e = addFirst(e, s_0)[0]$? Sí
- ▶ $addFirst(e, s_0)[0] = head(addFirst(e, s_0))$? Sí

15 / 42

Repaso: Cuantificadores

El lenguaje de especificación provee formas de predicar sobre los elementos de un tipo de datos

- ▶ $(\forall x : T)P(x)$: Afirma que todos los elementos de tipo T cumplen la propiedad P .
 - ▶ Se lee “Para todo x de tipo T se cumple $P(x)$ ”
- ▶ $(\exists x : T)P(x)$: Afirma que al menos un elemento de tipo T cumple la propiedad P .
 - ▶ Se lee “Existe al menos un x de tipo T que cumple $P(x)$ ”

16 / 42

Ejemplo

- Crear un predicado que sea **Verdadero** si y sólo si una secuencia de enteros sólo posee enteros mayores a 5.

- Solución:

```
pred seq_gt_five(s: seq<Z>) {  
  (∀i : Z)(0 ≤ i < |s| →L s[i] > 5)  
}
```

17 / 42

Ejemplo

- Crear un predicado que sea **Verdadero** si y sólo si todos los elementos con índices pares de una secuencia de enteros s son mayores a 5.

- Solución:

```
pred seq_even_gt_five(s: seq<Z>) {  
  (∀i : Z)(  
    ((0 ≤ i < |s|) ∧ (i mod 2 = 0))  
    →L s[i] > 5)  
}
```

18 / 42

Ejemplo

- Crear un predicado que sea **Verdadero** si y sólo si hay algún elemento en la secuencia s que sea par y mayor que 5.

- Solución:

```
pred seq_has_elem_even_gt_five(s: seq<Z>) {  
  (∃i : Z)(  
    (0 ≤ i < |s| ∧L ((s[i] mod 2 = 0) ∧ (s[i] > 5))  
  )  
}
```

19 / 42

Predicando sobre secuencias

Secuencia vacía o “isEmpty”

- Definir un predicado isEmpty que indique si la secuencia s no tiene elementos.

- Solución

```
pred isEmpty(s: seq<T>) {  
  |s| = 0  
}
```

20 / 42

Predicando sobre secuencias

Pertenencia o "has"

- Definir un predicado `has` que indique si el elemento `e` aparece (al menos una vez) en la secuencia `s`.

- Solución

```
pred has(s: seq<T>, e: T) {  
  (∃i: ℤ)(0 ≤ i < |s| ∧ s[i] = e)  
}
```

- Notación: Podemos utilizar este predicado como $e \in s$

21 / 42

Predicando sobre secuencias

Igualdad o "equals"

- Definir un predicado `equals(s1,s2)` que indique si la secuencia `s1` es igual a la secuencia `s2`.

- Solución

```
pred equals(s1, s2: seq<T>) {  
  s1 = s2  
}
```

22 / 42

Predicando sobre secuencias

Difieren en un elemento o "isSetAt"

- Definir un predicado `isSetAt(s1,s2,e,i)` que indique si la secuencia `s2` cambiándole el elemento de la posición `i` por `e` es igual a `s1`.
- En el caso que **no se cumpla** que $0 \leq i < |s2|$, retornar **Falso** sólo si ambas secuencias **no son** iguales.

- Solución

```
pred isSetAt(s1, s2: seq<T>, e: T, i: ℤ) {  
  (0 ≤ i < |s2| → s1 = setAt(s2, e, i))  
  ∧  
  (¬(0 ≤ i < |s2|) → s1 = s2)  
}
```

23 / 42

Σ - Sumatoria

El lenguaje de especificación provee formas de acumular resultados para los tipos numéricos \mathbb{Z} y \mathbb{R} .

El término

$$\sum_{i=from}^{to} Expr(i)$$

retorna la suma de todas las expresiones $Expr(i)$ entre *from* y *to*. Es decir,

$$Expr(from) + Expr(from + 1) + \dots + Expr(to - 1) + Expr(to)$$

Algunas condiciones:

- $Expr(i)$ debe ser un tipo numérico (\mathbb{R} o \mathbb{Z}).
- $from \leq to$ (retorna 0 si no se cumple).
- *from* y *to* es un **rango** (finito) de valores enteros, caso contrario se **indefine**.
- Si existe i tal que $from \leq i \leq to$ y $Expr(i) = \perp$, entonces toda la sumatoria se **indefine**!

24 / 42

Σ - Ejemplos

Retornar la sumatoria de una secuencia s de tipo $seq\langle T \rangle$.

Solución:

$$\sum_{i=0}^{|s|-1} s[i]$$

Ejemplos:

- Si $s = \langle 1, 1, 3, 3 \rangle$ retornará

$$s[0] + s[1] + s[2] + s[3] = 1 + 1 + 3 + 3 = 8$$

- Si $s = \langle \rangle$, entonces $from = 0$ y $to = -1$, por lo tanto retornará 0

25 / 42

Σ - Ejemplos

Retornar la sumatoria de la posición 1 (únicamente) de la secuencia s .

Solución:

$$\sum_{i=1}^1 s[i]$$

Ejemplos:

- Si $s = \langle 7, 11, 3, 3, 2, 4 \rangle$ retornará $s[1] = 11$.
- Si $s = \langle 7 \rangle$ la sumatoria se indefinirá ya que $s[1] = \perp$.

26 / 42

Σ - Ejemplos

Retornar la sumatoria de los índices pares de la secuencia s .

Solución:

$$\sum_{i=0}^{|s|-1} (\text{if } (i \bmod 2 = 0) \text{ then } s[i] \text{ else } 0 \text{ fi})$$

Ejemplos:

- Si $s = \langle 7, 1, 3, 3, 2, 4 \rangle$ retornará

$$s[0] + 0 + s[2] + 0 + s[4] + 0 = 7 + 0 + 3 + 0 + 2 + 0 = 12$$

- Si $s = \langle 7 \rangle$ retornará $s[0] = 7$.

27 / 42

Σ - Ejemplos

Retornar la sumatoria de los elementos mayores a 0 de la secuencia s .

Solución:

$$\sum_{i=0}^{|s|-1} (\text{if } (s[i] > 0) \text{ then } s[i] \text{ else } 0 \text{ fi})$$

Ejemplos:

- Si $s = \langle 7, 1, -3, 3, 2, -4 \rangle$ retornará

$$s[0] + s[1] + 0 + s[3] + s[4] + 0 = 7 + 1 + 0 + 3 + 2 + 0 = 13$$

- Si $s = \langle -7 \rangle$ retornará 0.

28 / 42

Π - Productoria

El término

$$\prod_{i=from}^{to} Expr(i)$$

retorna el producto de todas las expresiones $Expr(i)$ entre $from$ y to . Es decir,

$$Expr(from) * Expr(from + 1) * \dots * Expr(to - 1) * Expr(to)$$

Algunas condiciones:

- ▶ $Expr(i)$ debe ser un tipo numérico (\mathbb{R} o \mathbb{Z}).
- ▶ $from$ y to define un rango de valores enteros (finito) y $from \leq to$ (retorna 1 si no se cumple).
- ▶ Si existe i tal que $from \leq i \leq to$ y $Expr(i) = \perp$, entonces toda la productoria se **undefine!**

29 / 42

Π - Ejemplos

Retornar la productoria de los elementos mayores a 0 de la secuencia s .

Solución:

$$\prod_{i=0}^{|s|-1} (\text{if } (s[i] > 0) \text{ then } s[i] \text{ else } 1 \text{ fi})$$

Ejemplos:

- ▶ Si $s = \langle 7, 1, -3, 3, 2, -4 \rangle$ retornará

$$s[0] * s[1] * 1 * s[3] * s[4] * 1 = 7 * 1 * 1 * 3 * 2 * 1 = 42$$

- ▶ Si $s = \langle -7 \rangle$ retornará 1.

30 / 42

Funciones auxiliares imprescindibles

Definir una función que permita contar la cantidad de apariciones de un elemento e en la secuencia s :

$$\text{aux } \#apariciones(s: seq\langle T \rangle, e: T): \mathbb{Z} = \sum_{i=0}^{|s|-1} (\text{if } s[i] = e \text{ then } 1 \text{ else } 0 \text{ fi});$$

Ejemplos:

- ▶ $\#apariciones(\langle 5, 1, 1, 1, 3, 3 \rangle, 1) = 3$
- ▶ $\#apariciones(\langle 5, 1, 1, 1, 3, 3 \rangle, 2) = 0$
- ▶ $\#apariciones(\langle \rangle, 5) = 0$

31 / 42

Funciones auxiliares imprescindibles

Definir un predicado que sea verdadero si y sólo si una secuencia es una permutación¹ de otra secuencia.

Ejemplos:

- ▶ $\text{es_permutacion}(\langle 5, 1, 1 \rangle, \langle 5, 1, 1 \rangle) = \text{True}$
- ▶ $\text{es_permutacion}(\langle 5, 1, 1 \rangle, \langle 1, 5, 2 \rangle) = \text{False}$
- ▶ $\text{es_permutacion}(\langle 5, 1, 1 \rangle, \langle 1, 5, 1 \rangle) = \text{True}$

```
pred es_permutacion(s1, s2 : seq⟨T⟩){  
  (∀e : T)(#apariciones(s1, e) = #apariciones(s2, e))  
}
```

¹ mismos elementos y misma cantidad por cada elemento, en un orden potencialmente distinto

32 / 42

Un ejemplo con sumatoria

El predicado de la clase pasada para ver si un número entero es primo:

```
pred esPrimo( $n : \mathbb{Z}$ ) {  
   $n > 1 \wedge (\forall n' : \mathbb{Z})(1 < n' < n \rightarrow n \bmod n' \neq 0)$   
}
```

Podemos hacer otra versión con sumatorias.

1. Por cada número entre 2 y $n - 1$ me fijo si n es divisible por ese número.
2. Cada vez que encuentro un número i que me divide, sumo 1
3. Si al final no acumulé nada, quiere decir que no encontré ningún número entre 2 y $n - 1$ que divida a n

```
pred soy_primo( $n : \mathbb{Z}$ ) {  
   $n > 1 \wedge$   
   $(\sum_{i=2}^{n-1} (\text{if } (n \bmod i = 0) \text{ then } 1 \text{ else } 0 \text{ fi})) = 0$   
}
```

33 / 42

Otro ejemplo con cantidades

Definir una función que retorne la cantidad de números primos menores a un entero n (o 0 si $n < 0$)

1. Por cada número entre 2 y $n - 1$ me fijo si n es primo.
2. Cada vez que encuentro un número primo, acumulo 1
3. Si $n < 0$, debe retornar 0.

```
aux #primosMenores( $n : \mathbb{Z}$ ) =  
   $\sum_{i=2}^{n-1} (\text{if } \text{soy\_primo}(i) \text{ then } 1 \text{ else } 0 \text{ fi});$ 
```

34 / 42

Contando elementos en un conjunto

- La siguiente expresión es muy común en especificaciones de problemas:

$$\sum_{i \in A} \text{if } P(i) \text{ then } 1 \text{ else } 0 \text{ fi.}$$

- Introducimos la siguiente notación como **reemplazo sintáctico** para esta sumatoria:

$$\#\{i \in A : P(i)\}$$

- Por ejemplo, podemos escribir

$$\#\{i : 1 \leq i \leq n - 1 \wedge \text{soy_primo}(i)\}.$$

- Observación: A tiene que ser un conjunto **finito**.

35 / 42

Sumatoria de secuencias de \mathbb{R}

Definir una función que sume los inversos multiplicativos de una lista de reales.

Si no existe el inverso multiplicativo, ignorar el término.

```
aux sumarInvertidos( $s : \text{seq}(\mathbb{R})$ ) :  $\mathbb{R}$  =  
   $\sum_{i=0}^{|s|-1} (\text{if } s[i] \neq 0 \text{ then } \frac{1}{s[i]} \text{ else } 0 \text{ fi});$ 
```

36 / 42

Ejemplo de especificación con sumatorias

Especificar un programa que sume los inversos multiplicativos de una lista de reales, pero que requiera que todos los elementos de la secuencia **tengan** inverso multiplicativo.

```
pred todos_tienen_inverso(s: seq<ℝ>) {  
  (∀ i : ℤ)(0 ≤ i < |s| →L s[i] ≠ 0)  
}  
  
proc sumaInversos(in s: seq<ℝ>, out result: ℝ) {  
  Pre { todos_tienen_inverso(s) }  
  Post { result = sumarInvertidos(s) }  
}
```

37 / 42

Especificaciones y comentarios

- ▶ Los nombres de los predicados/funciones ayudan a describir el significado de las precondiciones y postcondiciones de las especificaciones.
- ▶ Los comentarios (*/*...*/*) también ayudan a describir el significado de las precondiciones y postcondiciones de las especificaciones y son útiles si no hay predicados

Ejemplo:

```
proc menorElemDistintos(in s: seq<ℤ>, out result: ℤ) {  
  Pre { noNegativos(s) ∧ noHayRepetidos(s) }  
  Post {  
    /* result es un índice válido de s */  
    0 ≤ result < |s|L  
    /* s[result] es el menor elemento de s */  
    (∀ i : ℤ)((0 ≤ i < |s|) →L s[result] ≤ s[i])  
  }  
}
```

38 / 42

Matrices

- ▶ Una **matriz** es una **secuencia de secuencias**, todas con la misma longitud (y no ser vacías).
- ▶ Cada posición de esta secuencia es a su vez una secuencia, que representa una fila de la matriz.
- ▶ Definimos **Mat<ℤ>** como un reemplazo sintáctico para **Seq<Seq<ℤ>>**.
- ▶ Una **Seq<Seq<ℤ>>** representa una matriz si todas las secuencias tienen la misma longitud! Definimos entonces:

```
pred esMatriz(m: Seq<Seq<ℤ>>) {  
  (∀ i : ℤ)(0 ≤ i < filas(m) →L  
    |m[i]| > 0 ∧  
    (∀ j : ℤ)(0 ≤ j < filas(m) →L |m[i]| = |m[j]|))  
}
```

- ▶ Notar que podemos reemplazar **Mat<ℤ>** por **Seq<Seq<ℤ>>** en la definición del predicado.

39 / 42

Matrices

- ▶ Tenemos funciones para obtener la cantidad de filas y columnas de una matriz:

```
aux filas(m : Mat<ℤ>) : ℤ = |m|;  
aux columnas(m : Mat<ℤ>) : ℤ  
  = if filas(m) > 0 then |m[0]| else 0 fi;
```

- ▶ En muchas ocasiones debemos recibir matrices **cuadradas**. Definimos también:

```
pred esMatrizCuadrada(m: Seq<Seq<ℤ>>) {  
  esMatriz(m) ∧ filas(m) = columnas(m)  
}
```

40 / 42

Matrices

- **Ejemplo:** Un predicado que determina si una matriz es una **matriz identidad**.

```
pred esMatrizIdentidad(m: Mat( $\mathbb{Z}$ )) {  
  esMatrizCuadrada(m)  $\wedge_L$   
  (  
    ( $\forall i : \mathbb{Z}$ ) ( $0 \leq i < \text{filas}(m) \rightarrow_L m[i][i] = 1$ )  $\wedge$   
    ( $\forall i : \mathbb{Z}$ ) ( $\forall j : \mathbb{Z}$ ) ( $0 \leq i, j < \text{filas}(m) \wedge i \neq j$   
       $\rightarrow_L m[i][j] = 0$ )  
  )  
}
```

Bibliografía

- David Gries - The Science of Programming
 - Chapter 4 - Predicates (cuantificación, variables libres y ligadas, etc.)
 - Chapter 5 - Notations and Conventions for Arrays (secuencias)