

# Estructuras

Laura Pomponio

Departamento de Ciencias de la Computación (DCC)

Escuela de Ciencias Exactas y Naturales (ECEN)



2023

Consideremos que queremos diseñar un programa interactivo en el cual un objeto se desplace

- en el plano en diferentes direcciones o
- apareciendo en forma aleatoria en diferentes lugares o
- que se desplace y mientras lo hace cambie de color o ...etc...

¿cuál sería el estado?

¿cómo haríamos para representar el estado?



Ejemplo ¿Cuál es el estado? ¿Cómo lo representamos?



Ejemplo ¿Cuál es el estado? ¿Cómo lo representamos?

¿Y ahora...?



Y si quisiéramos representar

- contactos de una agenda
- estudiantes de la facultad
- registro de un automóvil
- etc...

¿cómo lo haríamos?



**Necesitamos mayor poder expresivo y mecanismos que nos permitan manipular tipos de datos más complejos.**

### **Necesitamos estructuras.**

Muchos lenguajes de programación nos proveen estructuras, aunque a veces se las nombra diferente.



En matemática utilizamos la noción de producto cartesiano.

$$f : \mathbb{R}^2 \rightarrow \mathbb{R} \quad \text{o bien,} \quad f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$$
$$p \in \mathbb{R} \times \mathbb{R} \quad \text{o bien,} \quad (x, y) \in \mathbb{R} \times \mathbb{R}$$

Queremos poder expresar datos como

---

- $p \in \mathbf{Number} \times \mathbf{Number}$  y

$p$  es 

3	5
---	---

---

- $d \in \mathbf{String} \times \mathbf{Number} \times \mathbf{Number}$  y

$d$  es 

“Alan Turing”	1912	1954
---------------	------	------

---

# Estructura posn

(1)

El lenguaje nos provee una estructura para representar puntos en el plano.

```
(define-struct posn [x y])  
; Un elemento posn representa una posición de  
coordenadas cartesianas.
```

x	y



# Estructura posn

(2)

- Constructor `make-posn`

`; make-posn : Number Number -> posn`  
`(make-posn 3 5) →`

x	y
3	5

- Selector `posn-x`

`; posn-x : posn -> Number`  
`(posn-x (make-posn 3 5)) → 3`

- Selector `posn-y`

`; posn-y : posn -> Number`  
`(posn-y (make-posn 3 5)) → 5`

- Predicado `posn?`

`; posn? : Any -> Boolean`  
`(posn? (make-posn 3 5)) → #true`  
`(posn? 3) → #false`

# Estructura **posn**: leyes de reducción

(3)

## Primera ley

```
(posn-x (make-posn a b))  
== < def. posn-x >  
a
```

## Segunda ley

```
(posn-y (make-posn a b))  
== < def. posn-y >  
b
```



# Estructura posn: leyes de reducción

(4)

## Ejemplo

```
(define P (make-posn 3 5))  
(define Q (make-posn 12 23.5))  
  
(make-posn (posn-x P) (posn-y Q))
```

---

```
(make-posn (posn-x P) (posn-y Q))  
==<def. P>  
(make-posn (posn-x (make-posn 3 5)) (posn-y Q))  
==<def. posn-x>  
(make-posn 3 (posn-y Q))  
==<def. Q>  
(make-posn 3 (posn-y (make-posn 12 23.5))))  
==<def. posn-y>  
(make-posn 3 23.5)
```

¿ Cómo definiríamos una función que reciba un punto en el plano y determine si el punto pertenece a la recta

$$x + 3y + 10 = 0 ?$$

¿  $(2, -4)$   $\in$   $x + 3y + 10 = 0$  ?

# Definición de tipos de estructuras

```
(define-struct <nombre> [ <campo1> ... <campoN> ])
```

siendo

- <nombre> el identificador o nombre de la estructura.
- <campo<sub>i</sub>>, con  $i:1..N$ , un identificador o nombre de un campo de la estructura.

Esta definición **genera automáticamente** la definición de **otras funciones** asociadas.

- constructor: `make-<nombre>`
- selectores: `<nombre>-<campoi>` con  $i:1..N$
- predicado: `<nombre>?`

```
(define-struct Nota [nombre p1 p2 prom])
;Nota es (String, Number, Number, Number)
;donde
;nombre: es el nombre de la persona
;p1: es la nota que la persona sacó en el 1er parcial
;p2: es la nota que la persona sacó en el 2do parcial
;prom: es la nota promedio de los parciales
```

nombre	p1	p2	prom

**Nota**

Por convención utilizaremos **mayúsculas y minúsculas** para nombrar las estructuras que definamos, **comenzando con mayúscula**.



```
(define-struct Nota [nombre p1 p2 prom])  
; Nota es (String, Number, Number, Number)
```

---

Se generan automáticamente las funciones:

- **make-Nota** (constructor)  
;make-Nota: String Number Number Number -> Nota
- **Nota-nombre** (selector)  
;Nota-nombre: Nota -> String
- **Nota-p1** (selector)  
;Nota-p1: Nota -> Number
- **Nota-p2** (selector)  
;Nota-p2: Nota -> Number
- **Nota-prom** (selector)  
;Nota-prom: Nota -> Number
- **Nota?** (predicado)  
;Nota?: Any -> Boolean



- ¿Cómo crearíamos una **Nota**?
- ¿Cómo modificaríamos el nombre de una persona?
- ¿Cómo modificaríamos la nota de un parcial?
- ¿Cómo determinaríamos, entre dos personas, cuál tiene mejor promedio?
- ¿Cómo determinaríamos si un dato es una **Nota** cuyo promedio es mayor a 8?

