



FUNDAMENTOS DE PROGRAMACIÓN más de estructuras

**Escuela de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad del Valle**

Recordemos

- Una estructura es un dato compuesto, representa **una** pieza de información compuesta por varios campos.
- Al realizar la definición de una estructura se definen también (automáticamente) las operaciones para trabajar con dicha estructura:
 - crear (make)
 - consultar (selectores)
 - preguntar (x es una estructura)



Recordemos

○ Ejemplo 1:

- estructura llamada posn (ya está definida en scheme)
- representa un punto (x, y) en el plano cartesiano (una sola pieza de información).
- Definición:

```
(define-struct posn (x y))
```

- Creación de varios puntos:
 - el punto (0,0): `(make-posn 0 0)`
 - el punto (2,1): `(make-posn 2 1)`
- Selectores
 - el campo x del punto (0,0): `(posn-x (make-posn 0 0))`
 - el campo y del punto (2,1): `(posn-y (make-posn 2 1))`
- ¿X es un posn?: `(posn? X)`



Recordemos

○ Ejemplo:

- estructura llamada cheque
- contiene la información necesaria para hacer una transacción (en una sola pieza de información).
- Definición:
`(define-struct cheque (num banco monto))`
- Creación de algunos cheques:
 - `(make-cheque 1 'conavi 500000)`
 - `(make-cheque 22 'bogota 200000)`
- Selectores para un cheque X:
 - el campo num : `(cheque-num X)`
 - el campo banco: `(cheque-banco X)`
 - el campo monto: `(cheque-monto X)`
- ¿X es un cheque?: `(cheque? X)`



Ejemplo

- La función que calcula la distancia de dos puntos A y B.

Recordemos la fórmula: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, en donde los puntos son A=(x1, y1) y B=(x2, y2)

```
;distancia: posn posn -> numero
```

```
;calcula la distancia entre dos puntos  
representados por estructuras posn
```

```
;ejemplo: distancia entre (1,2) y (3,4)
```

```
;(distancia (make-posn 1 2) (make-posn 3 4)) da  
2.84
```

```
;Programa:
```

```
(define (distancia A B)
```

```
  (sqrt (+
```

```
    (expt (- (posn-x A) (posn-x B)) 2)
```

```
    (expt (- (posn-y A) (posn-y B)) 2))))
```



Ejemplo (2)

- diseñe un programa que calcula el total pagado, la entrada son dos estructuras de tipo cheque. El total es la suma de los montos:

```
;calcula-total: cheque cheque -> numero
;calcula la suma de los montos de los cheques
;ejemplos: Si el monto del cheque A es 100 y el
            del cheque B es 200 el programa retorna 300
;monto de cheque A + monto de cheque B
;Programa:
(define (calcula-total A B)
  (+ (cheque-monto A) (cheque-monto B)))
```



Ejemplo (3)

- Desarrolle un programa que determine si dos cheques pertenecen al mismo banco.

La función tiene dos entradas chequeA y chequeB

```
; mismo-banco?: cheque cheque -> booleano  
; función que determina si dos cheques  
  pertenecen al mismo banco.
```

- Análisis de datos:
 - Si el valor del campo banco del cheque A es igual al valor del campo banco del cheque B, la respuesta es true
 - En el caso contrario false.
- Plantilla:

```
; (define (mismo-banco chequeA chequeB)  
;   (cond  
;     []  
;     []  ))
```



Ejemplo (3)

○ Ejemplos:

- Si cheque A es: (make-cheque 1 'conavi 200)
y el cheque B es (make-cheque 23 'conavi 30000) la salida del programa es true.
- Si cheque A es: (make-cheque 1 'bancolombia 200)
y el cheque B es (make-cheque 23 'conavi 30000) la salida del programa es false.

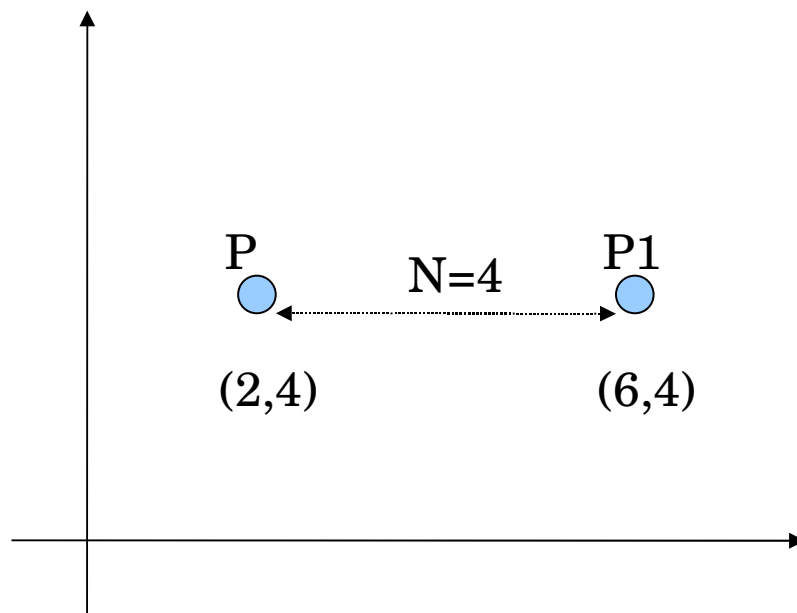
○ Programa:

```
(define (mismo-banco chequeA chequeB)
  (cond
    [(symbol=? (cheque-banco chequeA)
               (cheque-banco chequeB)) true]
    [else false]))
```



Funciones que retornan estructuras

- Escriba una función que tome como entrada un punto en el plano cartesiano (posn) y un número N. La función traslada al punto N unidades en el eje x.



Funciones que retornan estructuras

- La función retorna un posn, que es el punto trasladado

```
; trasladar-en-x: posn numero -> posn  
; función que traslada un punto n unidades
```

- Dado que la función no tiene condicionales y retorna un posn, la plantilla es la siguiente:

```
; plantilla:  
; (define (trasladar-en-x p n)  
;   (make-posn <coordenada x>  
;             <coordenada y>)) )
```



Funciones que retornan estructuras

○ Ejemplos:

- trasladar el punto (2,4) 6 unidades:

```
(trasladar-en-x (make-posn 2 4) 6) "debe retornar  
(make-posn 8,4)"
```

- trasladar el punto (0,0) 4 unidades

```
(trasladar-en-x (make-posn 0 0) 4) "debe retornar  
(make-posn 4,0)"
```

○ Programa:

```
(define (trasladar-en-x p n)  
  (make-posn (+ (posn-x p) n)  
              (posn-y p)))
```



Funciones que reciben varios tipos de estructuras

- Realice las definiciones de datos para tres tipos de cuentas bancarias, una cuenta bancaria puede ser :
 - **nomina** con dos campos: saldo y número de transacciones.
 - **ahorros** con dos campos: saldo y número de transacciones
 - **credito** con tres campos: saldo, límite y número de transacciones.
- A pesar de las diferencias entre los tipos de cuentas, podemos decir que todas son cuentas bancarias, es por esto que podemos decir (en comentario):
 - ; una cuenta_bancaria es:
 - ; de tipo nomina, de tipo ahorros, de tipo credito.



Funciones que reciben varios tipos de estructuras

- La anterior es una definición de datos para `cuenta_bancaria`. En adelante podemos referenciar a una entidad de tipo nomina, ahorros o credito, como una cuenta bancaria y puede existir una función que reciba como entrada una `cuenta_bancaria`.

- Ejemplo:

```
;retorna_saldo: cuenta_bancaria -> numero  
;propósito: función que retorna el saldo de  
una cuenta bancaria  
;encabezado:  
(define (retorna_saldo cuenta)...) 
```



Funciones que reciben varios tipos de estructuras

○ Análisis de datos:

- Si la cuenta es de tipo nómina, entonces se retorna el saldo usando el selector (nomina-saldo cuenta)
- Si la cuenta es de tipo ahorros, entonces se retorna el saldo usando el selector: (ahorros-saldo cuenta)
- Si la cuenta es de tipo crédito, entonces se retorna el saldo usando el selector: (credito-saldo cuenta)

○ Plantilla:

```
; (define (retorna_saldo cuenta)
;   (cond
;     [<pregunta> <respuesta>]
;     [<pregunta> <respuesta>]
;     [<pregunta> <respuesta>]))
```



Funciones que reciben varios tipos de estructuras

○ Ejemplos:

- Si la cuenta es: `(make-nomina 300000 2)` la función debe retornar 300000
- Si la cuenta es: `(make-ahorros 500000 7)` la función debe retornar 500000
- Si la cuenta es: `(make-credito 10000 200000 7)` la función debe retornar 10000

○ Programa:

```
(define (retorna_saldo cuenta)
  (cond
    [(nomina? cuenta) (nomina-saldo cuenta)]
    [(ahorros? cuenta) (ahorros-saldo cuenta)]
    [(credito? cuenta) (credito-saldo cuenta)]))
```




Funciones que reciben varios tipos de datos

La verificación de los tipos de las entradas nos permite hacer validación:

```
;; distancia-evaluada: posn posn -> number  
;; función que calcula la distancia de dos puntos  
haciendo validación en la entrada.
```

```
(define (distancia-evaluada A B)  
  (cond  
    [(and (posn? A) (posn? B)) (distancia A B)]  
    [else  
      (error 'error "A y B deben ser de tipo posn")])  
  ))
```



Funciones que reciben varios tipos de datos

Veamos la siguiente definición de datos para un pixel:

; un pixel es:

; 1. un número si está ubicado en el eje x (porque el valor en y siempre es cero.)

; 2. un posn si está en cualquier parte .

Proponga una función en scheme que calcule la distancia entre un pixel P y el origen (0,0).



Ejercicio

- Desarrolle una función llamada *retiro*, la cual toma como entrada una cuenta_bancaria y un monto a retirar. La función retorna el resultado de hacer un retiro de una cuenta bancaria (o intentarlo).
 - En el caso que la cuenta sea de tipo de nómina se pueden hacer retiros hasta que el saldo sea 0, al realizar un retiro se descuenta el monto y se calcula el nuevo saldo, también se aumenta en uno el número de transacciones. El retiro no se puede realizar en el caso en que, como resultado del descuento el saldo sea menor que cero (saldo negativo); si esto pasa el programa debe retornar un mensaje de error y **NO** modificar el saldo.



Ejercicio

- En el caso de una cuenta de tipo ahorros el proceso es similar al anterior, pero la cuenta no se puede dejar con menos de \$20.000.
- En el caso de una cuenta de tipo crédito, al hacer un retiro no se resta al saldo, sino que se suma al crédito, la deuda del cliente no puede superar el límite del crédito.

