



Mapeo de Procedimientos a Listas

- Repetición de un procedimiento a cada elemento de una lista:
 - MAP
 - FOR-EACH



Mapeo de Procedimientos a Listas

- MAP: aplica el procedimiento a cada elemento de la lista y devuelve una lista con los resultados.
 - $(\text{map } (\text{lambda } (x) (+ x 2)) '(1\ 2\ 3)) \rightarrow (3\ 4\ 5)$
- También es posible tener múltiples argumentos
 - $(\text{map cons } '(1\ 2\ 3) '(10\ 20\ 30)) \rightarrow ((1\ .\ 10)\ (2\ .\ 20)\ (3\ .\ 30))$



Mapeo de Procedimientos a Listas

- FOR-EACH: aplica el procedimiento a cada elemento de la lista pero devuelve <void>.
 - (for-each display
 (list "un" "dos " "tres"))



Estructuras

- **define-struct:** permite crear una estructura con los campos que se indican.
- Se crea tres operaciones:
 - un constructor: `make-<nom-struct>`
 - métodos selectores: `<nom-struct>-<campo>`
 - métodos accensores: `set-<nom-struct>-<campo>!`



Estructuras

- (define-struct posn (x y))
 - make-posn: (make-posn 1 2) \rightarrow #(struct:posn 1 2)
 - posn-x: (posn-x (make-posn 1 2)) \rightarrow 1
 - posn-y: (posn-y (make-posn 1 2)) \rightarrow 2
- (define punto (make-posn 1 2))
 - set-posn-x!: (set-posn-x! punto 2) \rightarrow (2 2)
 - set-posn-y!: (set-posn-y! punto 1) \rightarrow (2 1)



Ingreso y Salida de datos

- read-char: permite leer un carácter desde el puerto indicado. Por defecto, la consola
- read-line: lee toda una línea de caracteres desde el puerto indicado y devuelve un string
- write-char: escribe un carácter al puerto indicado
- write: escribe la expresión en el puerto indicado en formato de máquina. Ej: los strings con comillas dobles y los caracteres precedidos con #\
#\"
- display: muestra el resultado de la expresión



Puertos de Archivos

- open-input-file: abre un archivo y devuelve un puerto de lectura
- open-output-file: abre un archivo y devuelve un puerto de escritura
- close-input-port / close-output-port: cierran los puertos.



Puertos de Archivos

- `Hola.txt`
`hola!!`
- `(define i (open-input-file "hola.txt"))`
- `(read-char i) → #\h`
- `(define j (read i))`
- `j → ola!!`
- `(close-input-file i)`



Puertos de Archivos

- (define o (open-output-file "saludo.txt"))
- (display "hola" o)
- (write-char #\space o)
- (display 'mundo! o)
- (newline o)
- (close-output-port o)

→ saludo.txt
hola mundo!



Puertos de Strings

- Las lecturas sobre puertos de string finalizan en los separadores de las mimas.
- (define i (open-input-string "hola mundo"))
- (read-char i) → #\h
- (read i) → ola
- (read i) → mundo