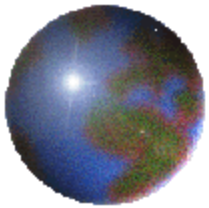


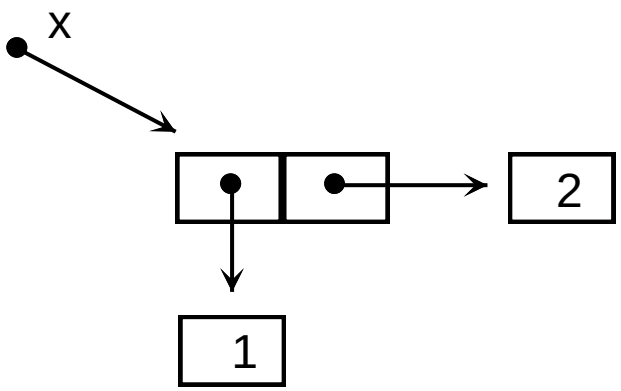
Programación Funcional
Lisp-DrScheme (2° parte)

Dr. Oldemar Rodríguez Rojas
Escuela de Informática
Universidad de Nacional

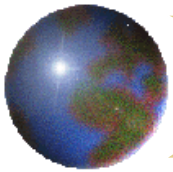
Pares y Aritmética Simbólica

- Lisp posee una estructura compuesta denominada par las cuales de manipulan con las funciones: cons, cdr y car
- Ejemplos:

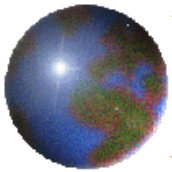
```
> (define x (cons 1 2))
X
> x
(1 . 2)
> (car x)
1
> (cdr x)
2
```



.....



```
>(define y (cons 3 -8))
Y
>y
(3 . -8)
>(define z (cons x y))
Z
>z
((1 . 2) 3 . -8)
>(car z)
(1 . 2)
>(cdr z)
(3 . -8)
```



```
>(car (car z))
1
>(caar z)
1
>(cdr (car z))
2
>(cdar z)
2
>(cddr z)
-8
```

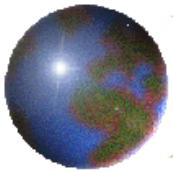
Ejemplo: La aritmética de los Racionales Q

```
; CONSTRUCTOR

(define (racional a b)
  (cons a b))

(define (denominador r)
  (cdr r))

(define (numerador r)
  (car r))
```

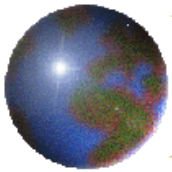


; *DEFINE SUMA DE NUMEROS RACIONALES*

```
(define (suma r1 r2)
  (racional (+ (* (numerador r1) (denominador r2))
               (* (denominador r1) (numerador r2)))
               (* (denominador r1) (denominador r2))))
```

; *DEFINE RESTA DE NUMEROS RACIONALES*

```
(define (resta r1 r2)
  (racional (- (* (numerador r1) (denominador r2))
               (*(denominador r1) (numerador r2)))
               (* (denominador r1) (denominador r2))))
```



; *DEFINE PRODUCTO DE NUMEROS RACIONALES*

```
(define (producto r1 r2)
  (racional  (* (numerador r1) (numerador r2))
             (* (denominador r1) (denominador r2))))
```

; *DEFINE LA DIVISIÓN NUMEROS RACIONALES*

```
(define (division r1 r2)
  (racional  (* (numerador r1) (denominador r2))
             (* (denominador r1) (numerador r2))))
```

; *IMPRIME NUMEROS RACIONALES*

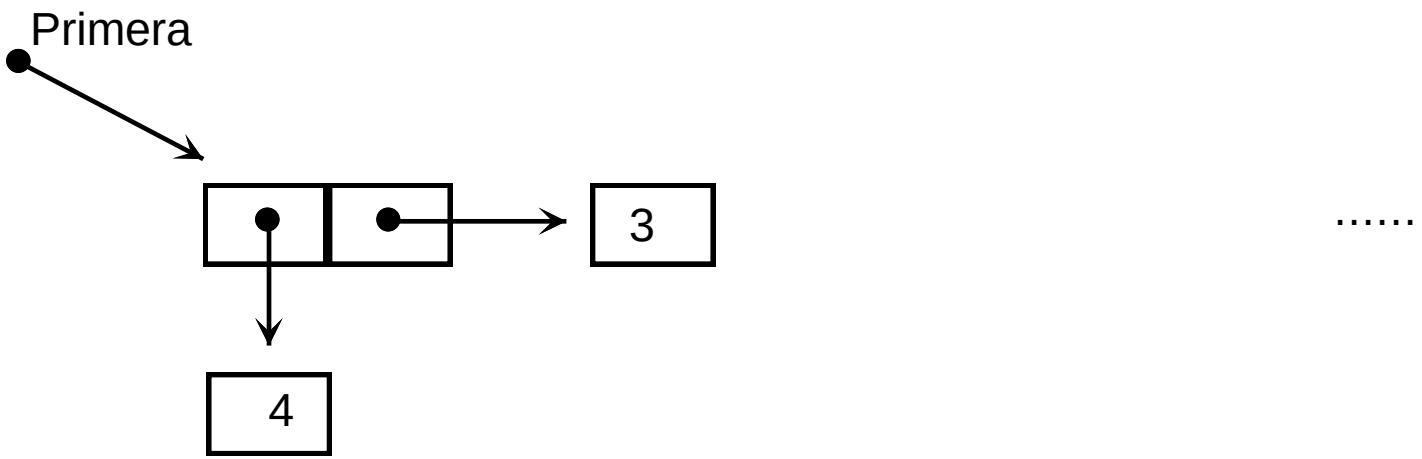
```
(define (imprime r)
  (newline)
  (write (numerador r))
  (write-char #\/ )
  (write (denominador r)))
```

Ejemplos de Uso

```
>(define r1 (racional 1 2))
R1
>(define r2 (racional 3 4))
R2
>(imprime r1)
½
>(imprime r2)
¾
>(define z (suma r1 r2))
Z
>(imprime z)
10/8
>(define p (producto r1 z))
P
>(imprime p)
10/16
```

Listas Enlazadas vrs Pares

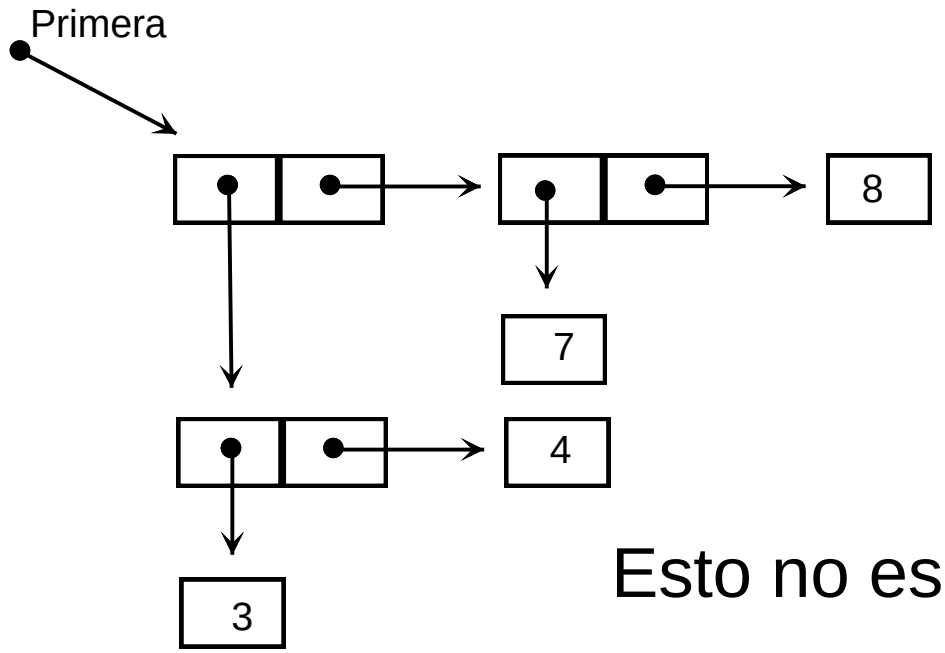
```
>(cons 3 4)
(3 . 4)
```



Listas Enlazadas vrs Pares

>(cons (cons 3 4) (cons 7 8))

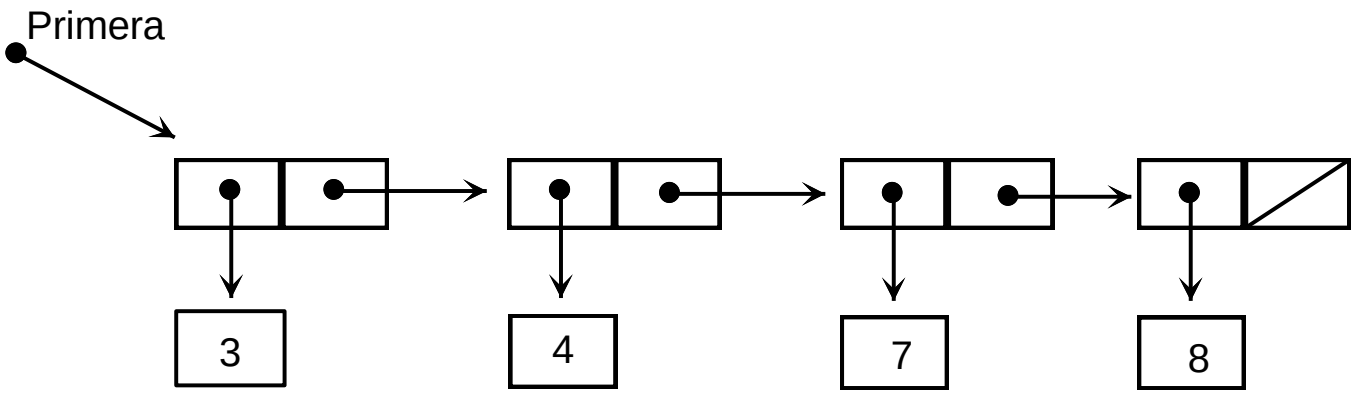
((3 . 4) 7 . 8)



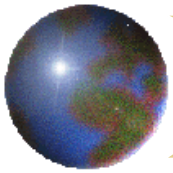
Esto no es una Lista

Listas Enlazadas vrs Pares

```
>(cons 3 (cons 4 (cons 7 (cons 8 '()))))
(3 4 7 8)
```



Esto si es una lista



La función para construir listas es:

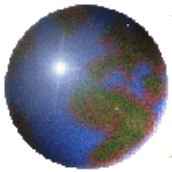
- (list a1 a2 a3 . . . . an)

- *Ejemplo:*

```
=>(list 3 4 7 8)
(3 4 7 8)
; es equivalente a (cons 3 (cons 4 (cons 7 (cons 8
  '())))
```

Más Ejemplos:

```
>(define L1 (list 'dos 'mas 'tres 'es 4))
L1
>L1
(dos mas tres es 4)
>(car L1)
Dos
>(cdr L1)
(mas tres es 4)
```



Operaciones con listas

```
define (longitud L)
  (if (null? L)
      0
      (+ 1 (longitud (cdr L)))))

(define (n-esimo n L)
  (if (= n 1)
      (car L)
      (n-esimo (- n 1) (cdr L))))
```

```
(define (suma-lista L)
  (if (null? L)
      0
      (+ (car L) (suma-lista (cdr L)))))

(define (pega L1 L2)
  (if (null? L1)
      L2
      (cons (car L1) (pega (cdr L1) L2))))
```

```
(define (cuadrado-lista L)
  (if (null? L)
      '()
      (cons (* (car L) (car L)) (cuadrado-lista (cdr L)))))


(define (multiplica-listas L1 L2)
  (if (null? L1)
      '()
      (cons (* (car L1) (car L2)) (multiplica-listas (cdr L1)  (cdr L2)))))
```

Ejemplos:

```
>(define L1 (list 3 4 5 -1))
L1
>(define L2 (list 1 1 2 2))
L2
>(longitud L1)
4
>(n-esimo 3 L1)
5
>(pega L1 L2)
(3 4 5 -1 1 1 2 2)
>(cuadrado-lista L1)
(9 16 25 1)
>(suma-lista L2)
6
>(multiplica-listas L1 L2)
(3 4 10 -2)
```

Más Ejemplos

```
(define (aplica f L)
  (if (null? L) '()
      (cons (f (car L)) (aplica f (cdr L)))))

(define (cuenta-pares L)
  (if (null? L) 0
      (if (= (modulo (car L) 2) 0)
          (+ 1 (cuenta-pares (cdr L)))
          (+ 0 (cuenta-pares (cdr L))))))

(define (invierte L)
  (if (null? L) '()
      (append (invierte (cdr L)) (list (car L)))))
```