



Lenguajes de programación

Recursividad en el lenguaje SCHEME



Recursividad

- En términos simples: Un módulo es recursivo, si contiene una llamada a sí mismo.
- Es la abstracción para la estructura de repetición en el lenguaje.

Ejemplo clásico

- Definición recursiva del factorial de un número:

$N!$ vale 1 si $N = 0$

$N!$ vale $N(N-1)!$ si $N > 0$*

- EJERCICIO: Programar la función en Scheme.

Pensando recursivamente...

1

- Analizar cuál es el caso más simple o pequeño del problema que se quiere resolver...
- Este caso, debe de tener una solución clara y directa... no recursiva...
- Este caso se considera el CASO BASE de la recursividad, y determina una condición de salida de la repetición implícita que se da en la recursividad.

Pensando recursivamente...

2

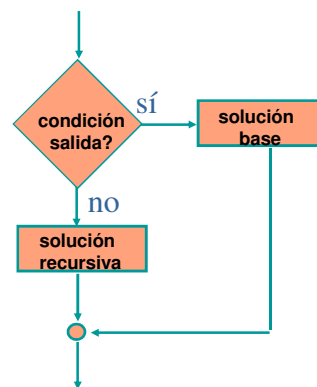
- Analizar cómo se resuelve el problema general, *suponiendo* que ya se tiene un procedimiento que resuelve el siguiente caso más pequeño o simple del problema...
- Este caso, plantea la solución recursiva del problema...
- La solución al siguiente caso más pequeño, la da la llamada recursiva...

Programando la Recursividad

- Típicamente, una rutina recursiva tiene la siguiente forma:

Si se cumple el caso más pequeño
Generar el resultado para
ese caso

Si NO
Generar el resultado de la
solución recursiva



Ejemplo de Recursividad

- El cálculo del Factorial para un número.

```
(define (factorial n)
  (if (<= n 1) 1
      (* n (factorial (- n 1)))))
```

(factorial 3)

(factorial 3)

= 3 * (factorial 2)

Ejemplo de Recursividad

```
(define (factorial n)
  (if (<= n 1) 1
      (* n (factorial (- n 1)))))
```

(factorial 3)

(factorial 2)

= 3 * = 2 * (factorial 1)

Ejemplo de Recursividad

```
(define (factorial n)
  (if (<= n 1) 1
      (* n (factorial (- n 1)))))
```

(factorial 3)

= 3 *

(factorial 2)

= 2 *

(factorial 1)

= 1

Ejemplo de Recursividad

```
(define (factorial n)
  (if (<= n 1) 1
      (* n (factorial (- n 1)))))
```

(factorial 3)

= 3 *

(factorial 2)

= 2 * 1

Ejemplo de Recursividad

```
(define (factorial n)
  (if (<= n 1) 1
      (* n (factorial (- n 1)))))
```

(factorial 3)

= 3 * 2

Ejemplo de Recursividad

```
(define (factorial n)
  (if (<= n 1) 1
      (* n (factorial (- n 1)))))
```

(factorial 3)

= 6



Fundamentación

- ¿Porqué funciona el pensamiento recursivo?
- Base formal: La INDUCCIÓN MATEMÁTICA como método de comprobación.



Ejercicio

- Implementar una función que sirva para elevar un número a cierta potencia entera no negativa (a^b).

RECURSIVIDAD TERMINAL (Tail recursive)

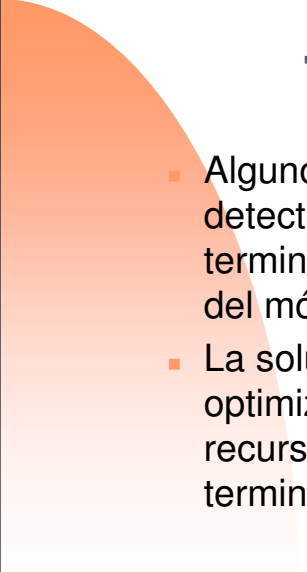
- Se dice que un módulo recursivo tiene RECURSIVIDAD TERMINAL, cuando la solución recursiva al problema NO contiene operaciones adicionales a la llamada recursiva.

Ejemplo

```
(define (factorial n)
  (if (<= n 1) 1
      (* n (factorial (- n 1)))))
```

• En este caso, no se conoce el resultado final hasta que se ejecuta la última operación al regresar de las llamadas recursivas.

No es
recursividad
terminal pues
la solución
recursiva usa
el resultado
de la llamada
para hacer una
operación
adicional.



¿Porqué es importante la RECURSIVIDAD TERMINAL?

- Algunos intérpretes son capaces de detectar cuando se utiliza la recursividad terminal, y hacer más eficiente la ejecución del módulo.
- La solución a algunos problemas, puede optimizarse convirtiendo su solución recursiva para que utilice recursividad terminal.



Eficiencia de la recursividad terminal

- Dado que el resultado final se obtiene al llegar al fondo de la recursividad (caso base), ya no es necesario regresar por cada llamada recursiva.
- Si el intérprete lo detecta, no se almacenan direcciones de regreso en el stack, y se usa un espacio constante de memoria.

Ejemplo

NO Terminal

```
(define (factorial n)
  (if (<= n 1) 1
      (* n (factorial (- n 1)))))
```

Terminal

```
(define (factorial n)
  (factorial-aux n 1))

(define (factorial-aux n r)
  (if (<= n 1) r
      (factorial-aux (- n 1) (* n r))))
```

Ejemplo de Recursividad terminal

- El cálculo del Factorial para un número.

```
(define (factorial n)
  (factorial-aux n 1))
```

```
(define (factorial-aux n r)
  (if (<= n 1) r
      (factorial-aux (- n 1) (* n r))))
```

• (factorial 3)

(factorial-aux 3 1)

= (factorial-aux 2 3)

Ejemplo de Recursividad terminal

```
(define (factorial n)
  (factorial-aux n 1) )

(define (factorial-aux n r)
  (if (<= n 1) r
      (factorial-aux (- n 1) (* n r) )))
```

(factorial-aux 2 3)

= (factorial-aux 1 6)

Ejemplo de Recursividad terminal

```
(define (factorial n)
  (factorial-aux n 1) )

(define (factorial-aux n r)
  (if (<= n 1) r
      (factorial-aux (- n 1) (* n r) )))
```

(factorial-aux 1 6)

= 6

Sin embargo...

- La solución a un problema por medio de las reglas del pensamiento recursivo, no siempre llevan a una solución con recursividad terminal.
- Un módulo que no tiene recursividad terminal, puede ser transformado a un módulo con recursividad terminal.

Transformación NO TERMINAL -> TERMINAL

- Construir un **nuevo módulo**, que contenga la **interface original del módulo** no terminal, pero que sólo sirva para llamar a un **módulo auxiliar** que será el que contenga la **recursividad terminal**.
- El módulo auxiliar contendrá **parámetros extra** que se requerirán para calcular el resultado.
- La primera llamada al módulo auxiliar debe **inicializar** los parámetros extra.

Transformación NO TERMINAL -> TERMINAL

- El módulo auxiliar, contendrá la lógica para resolver el problema recursivamente, pero cuidando de hacer la **llamada recursiva en forma terminal**.
- El **resultado del caso base** se obtendrá directamente de los valores de los parámetros extra.

Ejemplo

```
(define (factorial n)
  (if (<= n 1) 1
      (* n (factorial (- n 1)))))
```

Resultado del
caso base

```
; nuevo módulo (wrapper)
(define (factorial n)
  (factorial-aux n 1))
; módulo auxiliar
(define (factorial-aux n r)
  (if (<= n 1) r
      (factorial-aux (- n 1) (* n r))))
```

Operación adicional

Parámetro extra
(acumulador del resultado)

Llamada en forma
terminal



EJEMPLO

- *Transformar la función recursiva normal que eleva un número a una potencia entera a su versión terminal*