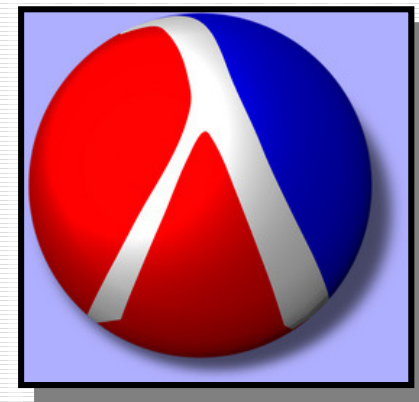


Fundamentos de Programación

Recursividad

Profesor: Daniel Wilches Maradei

Universidad del Valle



¿ Qué es la recursividad ?

- La recursividad es una técnica de programación en la que una función se llama a si misma
- Se usa mucho en Scheme cuando se necesita implementar ciclos
- Ejemplos:
 - función que calcule el máximo de una lista de números
 - función que cuente la cantidad de elementos en una lista
 - función que calcule el factorial o el número de fibonacci

Ejemplo de función recursiva

- Implementemos la sumatoria usando la recursividad.
- Recordemos que la formula de sumatoria es:

$$\Sigma(5) = 5 + 4 + 3 + 2 + 1$$

- O lo que es equivalente:

$$\Sigma(5) = 5 + \Sigma(4)$$

- Esta última definición es una definición recursiva

Ejemplo de función recursiva

- Usemos esta última definición para elaborar nuestra función en Scheme generalizando para cualquier número N:

$$\Sigma(N) = N + \Sigma(N - 1)$$

```
(define (sumatoria N)  
  (+ N (sumatoria (- N 1))))
```

1. Será que esta definición es correcta ?
2. Qué pasa si preguntamos por la sumatoria de 1 ?

Ejemplo de función recursiva

- Las respuestas son:
 1. NO
 2. La función nunca termina, sin importar el número que le pasemos por parámetro.
- Lo que sucede es que según nuestra definición,
 $\text{sumatoria}(1) = 1 + \text{sumatoria}(0)$ y
 $\text{sumatoria}(0) = 0 + \text{sumatoria}(-1)$ y
 $\text{sumatoria}(-1) = -1 + \text{sumatoria}(-2)$ y ... etc ...
Lo cual es claramente un error.
- A esto se le llama recursividad infinita

Ejemplo de función recursiva

- Debemos hacer una modificación a la función para darnos cuenta cuando lleguemos a una **condición de parada**.
- Esta condición es aquella que no cumple con la definición recursiva, es decir $\text{sumatoria}(0)$
- Veamos como nos quedaría nuestra definición (se usan condicionales):

$$\text{Si } N \neq 0 \quad \Sigma(N) = N + \Sigma(N - 1)$$

$$\text{Si } N = 0 \quad \Sigma(N) = 0$$

Ejemplo de función recursiva

- Realicemos entonces la implementación de esta fórmula en Scheme:

$$\text{Si } N \neq 0 \quad \Sigma(N) = N + \Sigma(N - 1)$$

$$\text{Si } N = 0 \quad \Sigma(N) = 0$$

```
(define (sumatoria N)
  (if (= N 0)
      0
      (+ N (sumatoria (- N 1)))))
```

Ejemplo de función recursiva

- Ahora, si preguntamos por la sumatoria de 1, la respuesta sería la esperada:

$\text{sumatoria}(1) = 1 + \text{sumatoria}(0)$, y como
 $\text{sumatoria}(0) = 0$, entonces
 $\text{sumatoria}(1) = 1 + 0 = 1$

- Cuando se evalúa $\text{sumatoria}(0)$ la llamada a la función deja de ser recursiva.

Otras funciones recursivas

- Si se nos pidiera elaborar una función que devuelva “par” si un número es par o cero, e “impar” si el número es impar.
- Cómo abordaríamos el problema ?

Otras funciones recursivas

- Una posibilidad es esta, la más sencilla:

```
(define (par-impar N)
  (if (= (remainder N 2) 0)
      "par"
      "impar")
))
```

Otras funciones recursivas

- Pero podemos abordar el problema con recursividad (aunque nos vamos a complicar más la solución que el problema, así que solo de modo didáctico):

```
(define (par-impar N)
  (if (par? N) "par" "impar"))
```

```
(define (par? N)
  (if (= N 0)
      true
      (not (par? (- N 1)))
  ))
```

Otro ejemplo:

Los números de fibonacci

- Los números de fibonacci son una sucesión definida por la siguiente función:

$$F(n) = \begin{cases} 0 & \text{si } n = 0; \\ 1 & \text{si } n = 1; \\ F(n-1) + F(n-2) & \text{si } n > 1. \end{cases}$$

n	F(n)	n	F(n)
0	0	4	3
1	1	5	5
2	1	6	8
3	2	7	13

Otro ejemplo:

Los números de fibonacci

$$F(n) = \begin{cases} 0 & \text{si } n = 0; \\ 1 & \text{si } n = 1; \\ F(n-1) + F(n-2) & \text{si } n > 1. \end{cases}$$

```
(define (fibonacci n)
  (cond
    ((= n 0) 0)
    ((= n 1) 1)
    .....
  ))
```

Otro ejemplo:

Los números de fibonacci

$$F(n) = \begin{cases} 0 & \text{si } n = 0; \\ 1 & \text{si } n = 1; \\ F(n-1) + F(n-2) & \text{si } n > 1. \end{cases}$$

```
(define (fibonacci n)
  (cond
    ((= n 0) 0)
    ((= n 1) 1)
    (else (+ (fibonacci (- n 1)) (fibonacci (- n 2)))))
  ))
```

Ejercicio

- Escriba una función que calcule el máximo común divisor de dos números.
- La función recursiva usada debe ser:

$$\text{gcd}(n, m) = \begin{cases} n & \text{si } m = n \\ \text{gcd}(n - m, m) & \text{si } m < n \\ \text{gcd}(n, m - n) & \text{si } m > n \end{cases}$$