UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA

DEPARTAMENTO DE
INFORMÁTICA Y ANÁLISIS NUMÉRICO

PROGRAMACIÓN DECLARATIVA

INGENIERÍA INFORMÁTICA

ESPECIALIDAD DE COMPUTACIÓN

CUARTO CURSO

PRIMER CUATRIMESTRE

Tema 4.- Recursión e iteración

Primera parte: Scheme

- **Tema 1.-** Introducción al Lenguaje Scheme
- **Tema 2.-** Expresiones y Funciones
- **Tema 3.-** Predicados y sentencias condicionales
- **Tema 4.-** Iteración y Recursión
- **Tema 5.-** Tipos de Datos Compuestos
- **Tema 6.-** Abstracción de Datos
- **Tema 7.-** Lectura y Escritura

Segunda parte: Prolog

- **Tema 8.-** Introducción al Lenguaje Prolog
- **Tema 9.-** Elementos Básicos de Prolog
- **Tema 10.-** Listas
- **Tema 11.-** Reevaluación y el "corte"
- **Tema 12.-** Entrada y Salida

Primera parte: Scheme

Tema 1.- Introducción al Lenguaje Scheme

Tema 2.- Expresiones y Funciones

Tema 3.- Predicados y sentencias condicionales

Tema 4.- Iteración y Recursión

Tema 5.- Tipos de Datos Compuestos

Tema 6.- Abstracción de Datos

Tema 7.- Lectura y Escritura

Índice

Índice

1. Forma especial iterativa “do”

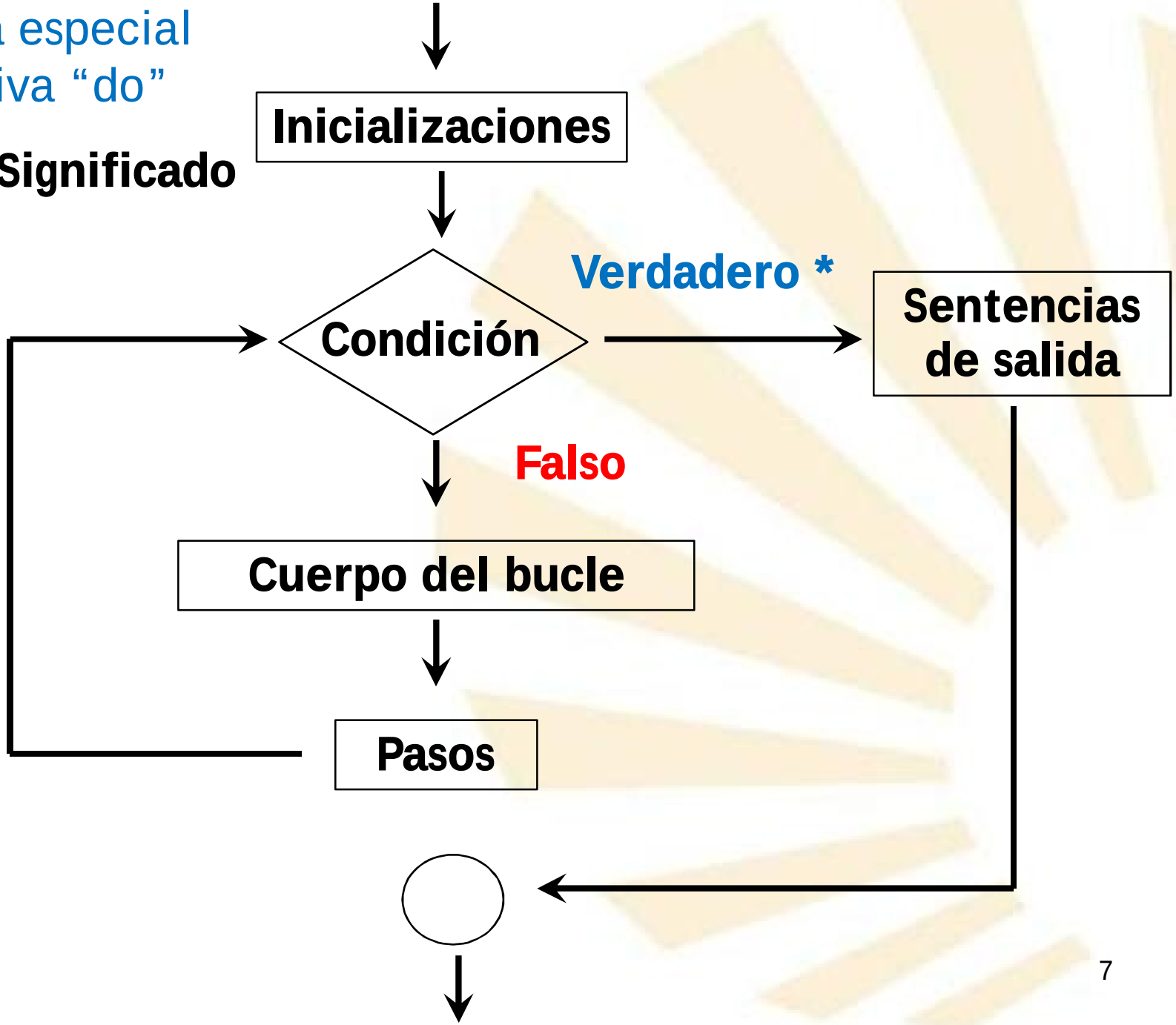
- **Sintaxis**

```
(do
    ;; Inicializaciones
    (
     (<variable 1>  <inicialización 1>  <paso 1>)
     (<variable 2>  <inicialización 2>  <paso 2>)
                    ...
     (<variable n>  <inicialización n>  <paso n>)
    )
    ;; Condición de salida y sentencias asociadas
    ( <condición> <sentencia> ...)

    ;; Cuerpo del bucle
    <sentencia> ...
)
```

1. Forma especial iterativa “do”

- **Significado**



1. Forma especial iterativa “do”

- **Semántica**
  - El ámbito de cada *variable_i* abarca el cuerdo del bucle **do** y las sentencias de **paso.**
  - La **evaluación** de las sentencias de **inicialización** se realiza en un **orden no determinado**
  - La *variable_i* no puede usarse en las sentencias de inicialización.
  - Las sentencias de paso pueden usar los **valores** de las **variables** que tengan en el **paso anterior**.
  - La **evaluación** de las sentencias de **paso** se realiza en un **orden no determinado**.

1. Forma especial iterativa “do”

- **Ejemplo (1/9):** factorial (versión nº 1)

```
(define (factorial-con-set n)
    (if (= n 0)  1
            (do
                ;; Inicializaciones
                 (
                   (i              n   (- i 1))
                   (resultado  1)          ;; no tiene “paso”
                 )
                 ;; Condición y sentencia de salida
                 ((= i 1)    resultado)
                 ;;  Cuerpo del bucle
                  (set! resultado (* resultado i))
            )
    )
)
```

Se recomienda evitar el uso de set!

1. Forma especial iterativa “do”

- **Ejemplo (2/9)**: factorial (versión nº 2)

```
(define (factorial n)
    (if (= n 0)  1
        (do
            ;; Inicializaciones
             (
                 (i              n   (- i 1))
                 (resultado  1   (* resultado i))
              )
            ;; Condición y sentencia de salida
            ((= i 1)    resultado)
            ;; no hay cuerpo del bucle
    )
)
```

1. Forma especial iterativa “do”

- **Ejemplo (3/9):** Fibonacci

```
(define (fib-iter n)
  (if (or (= n 0) (= n 1))  1
      (do
          ;; Inicializaciones
          (
           (actual     1   (+ actual anterior))
           (anterior   1    actual)
            (i              n    (- i 1)
          )
           ;; Condición y sentencia de salida
          ((= i 1)   actual)
          ;; no hay cuerpo del bucle
         )
  )
)
```

1. Forma especial iterativa “do”

- **Ejemplo (4/9)** $\sum_{n=a}^{b} n^3 = a^3 + (a+1)^3 + (a+2)^3 + \ldots + (b-1)^3 + b^3$

```
(define (sumar-cubos inicial final)
  (define (cubo x)
    (* x x x)
    )
  (do  ;; variables
       (
        (i          inicial    (+ i 1))
        (resultado  0.0        (+ resultado (cubo i)))
       )
    ;; Condición y sentencia de salida
    ((> i final) resultado)
    ;; No hay cuerpo
    )
  )
(suma-cubos 1 3)   ➔ 36
```

1. Forma especial iterativa “do”

- **Ejemplo: series no paramétricas**

$$\sum_{n=1}^{\infty}\frac{1}{n^2}=\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\ldots=\frac{1}{1}+\frac{1}{4}+\frac{1}{9}+\ldots=\frac{\pi^2}{6}$$

$$\sum_{\substack{n=1\\ n=n+4}}^{\infty}\frac{1}{n\times(n+2)}=\frac{1}{1\times3}+\frac{1}{5\times7}+\frac{1}{9\times11}+\ldots=\frac{\pi}{8}$$

1. Forma especial iterativa “do”

- **Ejemplo (5/9)** $\sum_{n=1}^{\infty}\frac{1}{n^2}=\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\ldots=\frac{1}{1}+\frac{1}{4}+\frac{1}{9}+\ldots=\frac{\pi^2}{6}$

```
(define (sumar-reciprocos-cuadrados-perfectos final)
  (define (termino n)
    (/  1.0    (* n n))
    )
  ;;
  (do (
        (i              1.0     (+ i 1))
        (resultado   0.0   (+ resultado (termino i)))
      )
    ;; Condición de salida: número de elementos
    ((> i final) resultado)
    ;; No hay cuerpo
    )
  )
(sumar-reciprocos-cuadrados-perfectos 1000) ➔ 1,6449...
```

1. Forma especial iterativa “do”

- **Ejemplo(6/9)** $\sum_{n=1}^{\infty} \frac{1}{n\times(n+2)} = \frac{1}{1\times3}+\frac{1}{5\times7}+\frac{1}{9\times11}+\ldots=\frac{\pi}{8}$ $n=n+4$

```
(define (sumar-pi-octavos-v1 cota)
  (define (termino n)
    (/     1.0   (* n (+ n 2)))
    )
  ;;
  (do (
        (i              1.0        (+ i 4))
        (resultado 0.0             (+ resultado (termino i)))
      )
    ;;
    ((<  (termino i) cota) resultado)
     ;; No hay cuerpo
    )
  )
```

(8.0 (sumar-pi-octavos-v1 0.00001))* ➔ *3.135263603..*

1. Forma especial iterativa “do”

- **Ejemplo(6/9)** $\sum_{n=1}^{\infty} \frac{1}{n\times(n+2)} = \frac{1}{1\times3}+\frac{1}{5\times7}+\frac{1}{9\times11}+\ldots=\frac{\pi}{8}$

$n=n+4$

```
(define (sumar-pi-octavos-v1 cota)
  (define (termino n)
    (/     1.0   (* n (+ n 2)))
    )
  ;;
  (do (
        (i              1.0        (+ i 4))
        (resultado 0.0             (+ resultado (termino i)))
      )
    ;;
    ((< (termino i) cota) resultado)
     ;; No hay cuerpo
    )
  )
```

Ineficiencia

```
(* 8.0 (sumar-pi-octavos-v1 0.00001)) ➔ 3.135263603...
```

1. Forma especial iterativa “do”

- **Ejemplo(7/9)** $\sum_{n=1}^{\infty} \frac{1}{n\times(n+2)} = \frac{1}{1\times3}+\frac{1}{5\times7}+\frac{1}{9\times11}+\ldots=\frac{\pi}{8}$ $n=n+4$

```
(define (sumar-pi-octavos-v2 cota)
  (define (termino n)
    (/     1.0   (* n (+ n 2)))
    )
  ;;
  (do  (
        (i              5.0          (+ i 4))
        (valor          (termino 1)  (termino i))
        (resultado   0.0             (+ resultado valor))
      )
    ;; Condición y sentencia de salida
    ((< valor cota) resultado)
    ;; No hay cuerpo
    )
  )
(* 8.0 (sumar-pi-octavos-v2  0.00001)) ➔ 3.135263603
```

1. Forma especial iterativa “do”

- **Ejemplo(8/9)**

$$\sum_{\substack{n=1 \\ n=n+4}}^{\infty} \frac{1}{n\times(n+2)} = \frac{1}{1\times3}+\frac{1}{5\times7}+\frac{1}{9\times11}+\ldots=\frac{\pi}{8}$$

La serie anterior es equivalente a la serie

$$\sum_{n=1}^{\infty} \frac{1}{(4n-3)\times(4n-1)} = \frac{1}{1\times3}+\frac{1}{5\times7}+\frac{1}{9\times11}+\ldots=\frac{\pi}{8}$$

1. Forma especial iterativa “do”

- **Ejemplo(8/9)** $\sum_{n=1}^{\infty}\frac{1}{(4n-3)\times(4n-1)}=\frac{1}{1\times3}+\frac{1}{5\times7}+\frac{1}{9\times11}+\ldots=\frac{\pi}{8}$

```
(define (sumar-pi-octavos-v3 cota)
    (define (termino n)
        (/   1.0
           (*    (- (* 4 n) 3)   (- (* 4 n) 1)    )
        )
    )
     (do  (
            (i              2.0          (+ i 1))
            (valor          (termino 1)  (termino i))
            (resultado      0.0          (+ resultado valor))
          )
        ;; Condición y sentencia de salida
        ((< valor cota) resultado)
        ;; No hay cuerpo
        )
)
```

1. Forma especial iterativa “do”

- **Ejemplo: series paramétricas**

$$seno(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

1. Forma especial iterativa “do”

- **Ejemplo (9/9): series paramétricas**

```
(define (serie-seno   x    iteraciones)
  (define (termino x n)
    (*  (expt -1 n)
        (/   (expt  x (+ (* 2 n) 1))
             (factorial (+ (* 2 n) 1))
        )
    )
  )
  ;;
  (do   (
         (n              0      (+ n 1))
         (resultado 0.0    (+ resultado (termino x n)))
    )
    ;;
    ((> n iteraciones) resultado)
    ;; No hay cuerpo
    )
  )
```

Índice

2. Recursión

- Descripción
- Recursión simple
- Recursión múltiple
- Recursión de cola
- Forma especial “let con nombre”

2. Recursión

- Descripción
- Recursión simple
- Recursión múltiple
- Recursión de cola
- Forma especial “let con nombre”

2. Recursión

- Descripción
  - Una función es "***recursiva***" si **se llama** a sí misma
  - **Fases** para definir una función recursiva
    1. Determinar cómo realizar **un paso**.
    2. **Descomponer** el problema en un paso y en un sub-problema **idéntico** más simple.
    3. Determinar cuándo se ha de **parar** la ejecución de la función.

2. Recursión

- Descripción
- Recursión simple
- Recursión múltiple
- Recursión de cola
- Forma especial “let con nombre”

2. Recursión

- Recursión simple
  - Una función posee **recursividad simple**
    - si solamente puede **llamarse** a sí misma **una única vez, a lo sumo**
    - por cada **línea de ejecución**.
  - **Línea de ejecución**
    - Sucesión de sentencias que se pueden ejecutar consecutivamente dentro de una función

2. Recursión

- Recursión simple
  - **Ejemplo (1/7)**
    - Factorial (versión recursiva): $n \in N$

$$f(n) = n! = \begin{cases} 1 & \text{Si } n=0 \vee n=1 \\ n \times (n-1)! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1 & \text{Si } n \geq 2 \end{cases}$$

2. Recursión

• Recursión simple

o **Ejemplo (1/7)**

- Factorial (versión recursiva): $n \in N$

```
(define (factorial n)
        (if (or (= n 0) (= n 1))
                1
                (* n (factorial (- n 1))))
        )
)

(factorial 4) ➔ 24
```

2. Recursión

- Recursión simple
  - **Ejemplo (2/7)**
    - Potencia: $a \in R, b \in N$

$$a^b = \begin{cases} 1 & \text{Si } b = 0 \\ a \times a^{b-1} & \text{Si } b \geq 1 \end{cases}$$

2. Recursión

- Recursión simple
  - **Ejemplo (2/7)**
    - Potencia: $a \in R, b \in N$

```
(define (potencia a b)
        (if (= b 0)
                1
                (* a (potencia a (- b 1))))
        )
)

(potencia  2 3) ➔ 8
```

2. Recursión

• Recursión simple

o **Ejemplo (3/7)**

$$\sum_{n=a}^{b} n^3 = a^3 + (a+1)^3 + (a+2)^3 + \ldots + (b-1)^3 + b^3$$

```
(define (suma-cubos inicial final)
        ;; función auxiliar
        (define (cubo x) (* x x x))
        ;; cuerpo de suma-cubos
        (if (> inicial final)
                0
                (+
                    (cubo inicial)
                    (suma-cubos  (+ inicial 1) final)
                )
        )
)
(suma-cubos 1 3)        ➔ 36
```

2. Recursión

- Recursión simple
  - **Ejemplo: series no paramétricas**

$$\sum_{n=1}^{\infty}\frac{1}{n^2} = \frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\ldots=\frac{1}{1}+\frac{1}{4}+\frac{1}{9}+\ldots=\frac{\pi^2}{6}$$

$$\sum_{\substack{n=1\\ n=n+4}}^{\infty}\frac{1}{n\times(n+2)} = \frac{1}{1\times3}+\frac{1}{5\times7}+\frac{1}{9\times11}+\ldots=\frac{\pi}{8}$$

$$\sum_{n=1}^{\infty}\frac{1}{(4n-3)\times(4n-1)} = \frac{1}{1\times3}+\frac{1}{5\times7}+\frac{1}{9\times11}+\ldots=\frac{\pi}{8}$$

2. Recursión

• Recursión simple

o **Ejemplo (4/7)**

$$\sum_{n=1}^{\infty}\frac{1}{n^2} = \frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\ldots=\frac{1}{1}+\frac{1}{4}+\frac{1}{9}+\ldots=\frac{\pi^2}{6}$$

```
(define (suma-reciprocos-cuadrados-perfectos n)
  (define (termino n)
        (/  1.0     (* n n))
    )
  (if (<= n 0)  0.0
      (+ (termino n)
         (suma-reciprocos-cuadrados-perfectos (- n 1))
      )
    )
  )
```

2. Recursión

- Recursión simple

  - **Ejemplo (5/7)**

$$\sum_{\substack{n=1 \\ n=n+4}}^{\infty} \frac{1}{n\times(n+2)} = \frac{1}{1\times3}+\frac{1}{5\times7}+\frac{1}{9\times11}+\ldots=\frac{\pi}{8}$$

```
(define (suma-pi-octavos-v1 n)
  (define (termino n)
    (/  1.0     (* n (+ n 2)))
  )
 (define (auxiliar-suma-pi-octavos   i  n)
  (if (> i n)
    0.0
    (+ (termino i)
       (auxiliar-suma-pi-octavos (+ i 4) n)
       )
    )
  )
   ;; Llamada a la función auxiliar
    (auxiliar-suma-pi-octavos 1 n)
)
```

2. Recursión

- Recursión simple

$$\sum_{n=1}^{\infty}\frac{1}{(4n-3)\times(4n-1)}=\frac{1}{1\times3}+\frac{1}{5\times7}+\frac{1}{9\times11}+\ldots=\frac{\pi}{8}$$

  - **Ejemplo (6/7)**

```
(define (suma-pi-octavos-v2 n)
    (define (termino n)
        (/   1.0
           (*    (- (* 4 n) 3)   (- (* 4 n) 1)    )
        )
    )
   (define (auxiliar-suma-pi-octavos   i  n)
      (if (> i n)      0.0
                    (+ (termino i)
                       (auxiliar-suma-pi-octavos (+ i 1) n)
                    )
      )
   )
   ;; Llamada a la función auxiliar
    (auxiliar-suma-pi-octavos 1 n)
)
```

2. Recursión

• Recursión simple

o **Ejemplo (7/7): serie paramétrica**

$$seno(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

```
(define (serie-seno x n)
  (define (termino x n)
    (* (expt -1 n)
       (/ (expt x (+ (* 2 n) 1))
          (factorial (+ (* 2 n) 1))
       )
    )
  )
 ;;
 (if (< n 0)
     0.0
     (+ (termino x n) (serie-seno x (- n 1)))
     )
 )
```

2. Recursión

- Descripción
- Recursión simple
- Recursión múltiple
- Recursión de cola
- Forma especial “let con nombre”

2. Recursión

- Recursión múltiple
  - Una función posee **recursividad múltiple** si
    - puede **llamarse** a sí misma **más de una vez**
    - en una **línea de ejecución**, al menos.

2. Recursión

- Recursión múltiple
  - **Ejemplo (1/2)**
    - Fibonacci (versión recursiva doble)

$$f(n) = \begin{cases} 1 & Si \;\; n=0 \vee n=1 \\ f(n-1) + f(n-2) & Si \;\; n \geq 2 \end{cases}$$

2. Recursión

- Recursión múltiple
  - **Ejemplo (1/2)**
    - Fibonacci (versión recursiva doble)

```
(define (fibonacci n)
   (if (or (= n 0) (= n 1))
         1
         (+
              (fibonacci (- n 1))
              (fibonacci (- n 2))
         )
    )
)
```

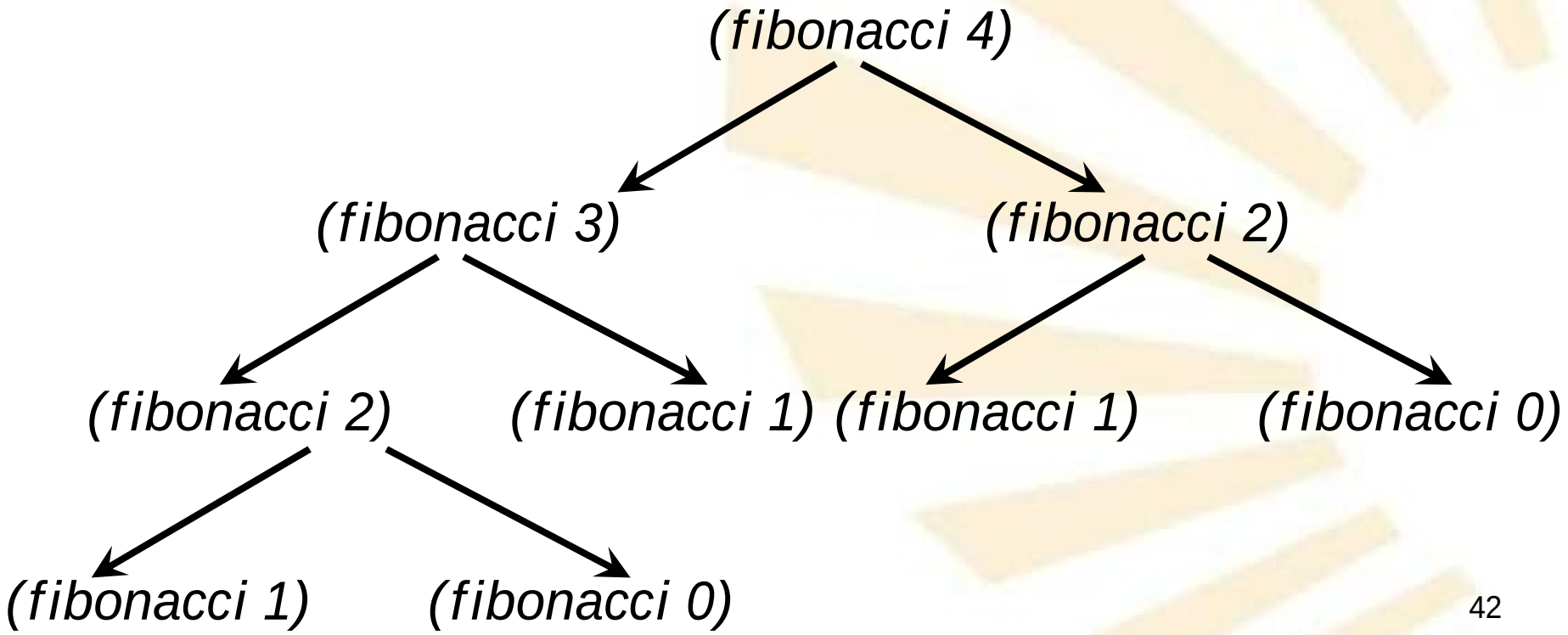
(fibonacci 4) ➔ *5*

2. Recursión

• Recursión múltiple

o **Ejemplo (1/2)**

- Fibonacci (versión recursiva doble)
  - ❑ Árbol de activación

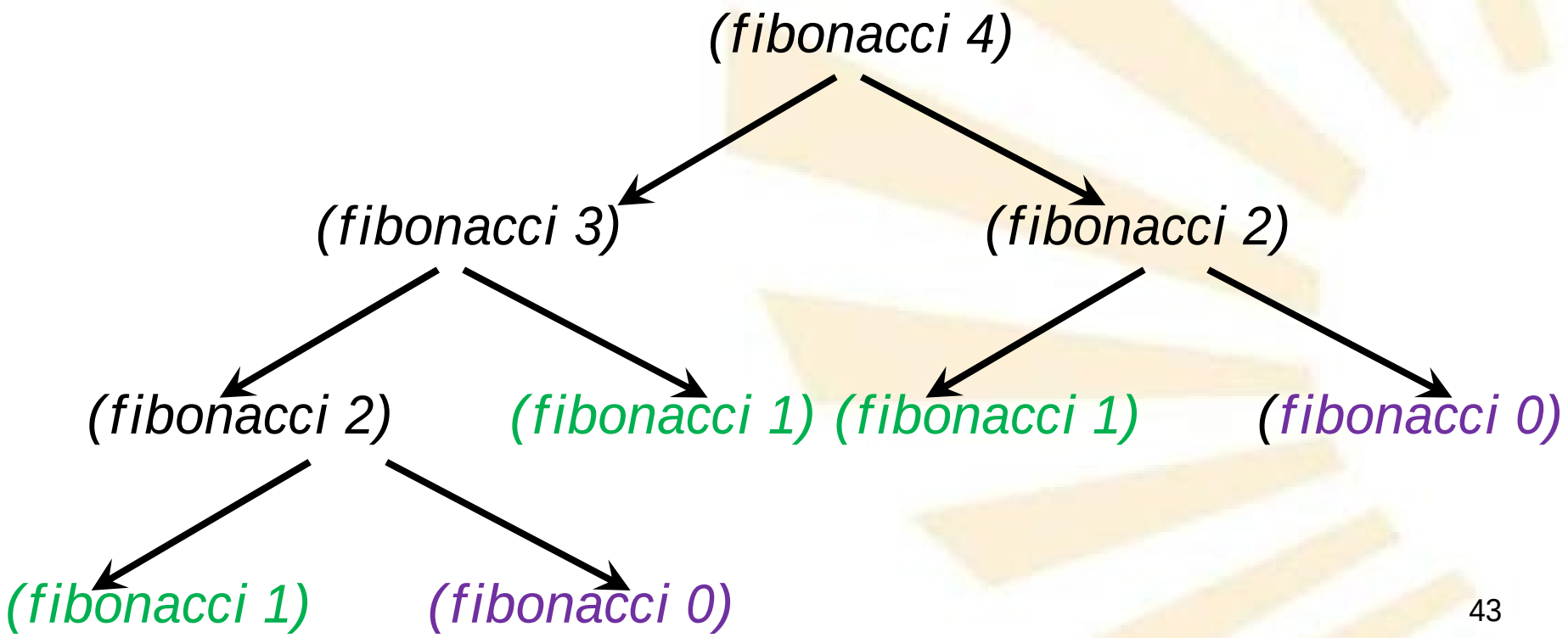


2. Recursión

- Recursión múltiple

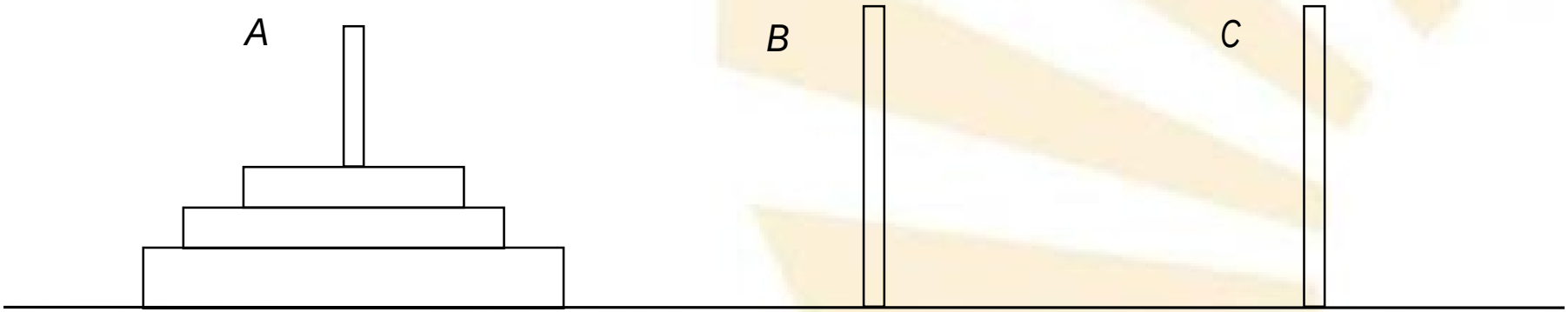
Ineficiencia de la recursión múltiple: repetición de cálculos

  - **Ejemplo (1/2)**
    - Fibonacci (versión recursiva doble)
      - ❑ Árbol de activación



2. Recursión

- Recursión múltiple
  - **Ejemplo (2/2)**
    - Torres de Hanoi



2. Recursión

• Recursión múltiple

o **Ejemplo (2/2)**

- Torres de Hanoi: objetivo
  - ❑ **Trasladar** todos los discos desde la varilla A hasta la varilla B
  - ❑ Se deben respetar las **reglas** de los movimientos
  - ❑ Se puede utilizar la varilla **auxiliar** C

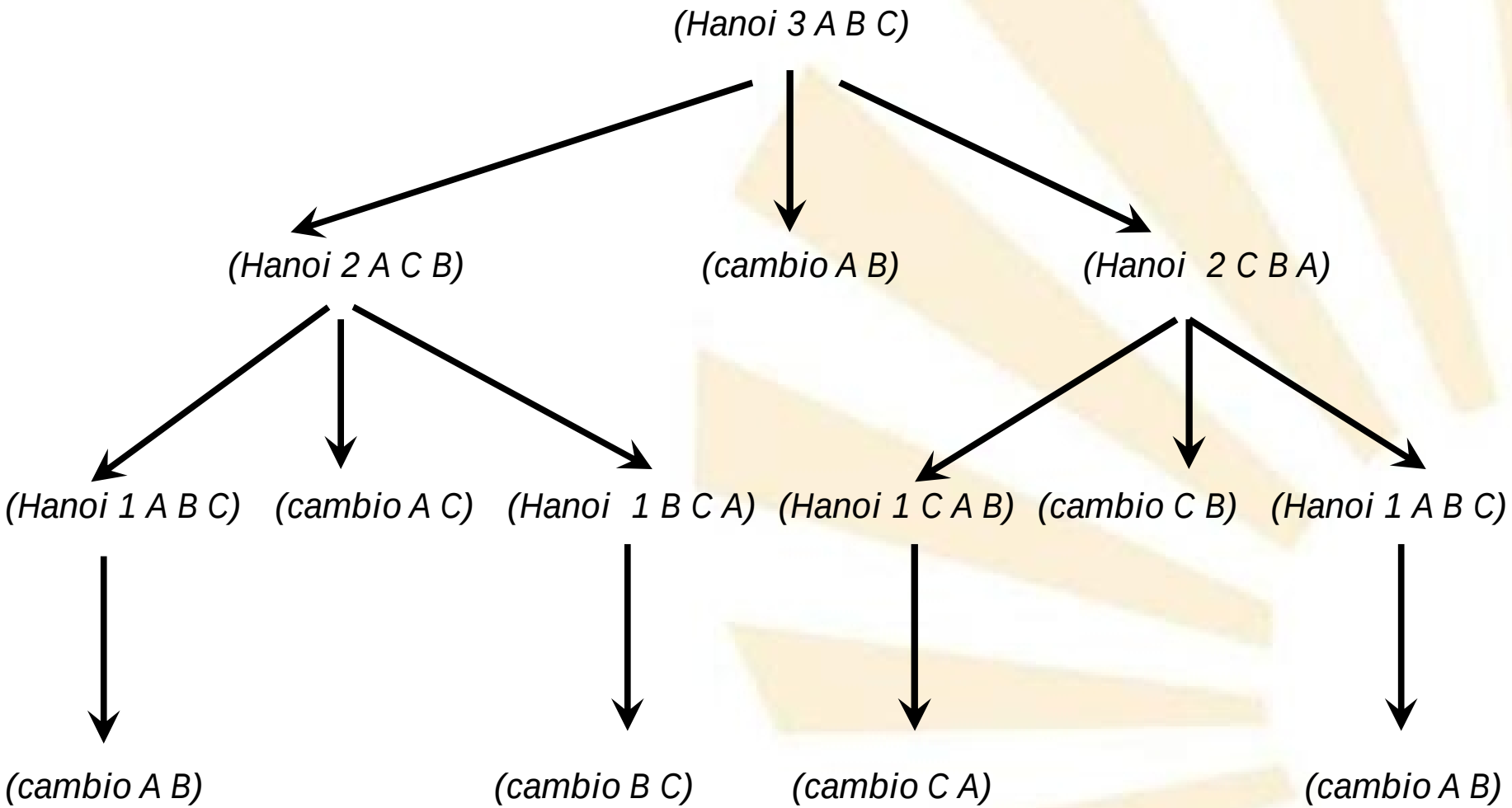
2. Recursión

- Recursión múltiple
  - **Ejemplo (2/2)**
    - Torres de Hanoi: reglas
      1. Todos los discos son de **tamaños diferentes**.
      2. Inicialmente cada uno de los discos **descansa** sobre uno mayor.
      3. Sólo se puede **mover un disco** cada vez.
      4. **Ningún** disco se puede colocar sobre uno más pequeño.

- **Torres de Hanoi**

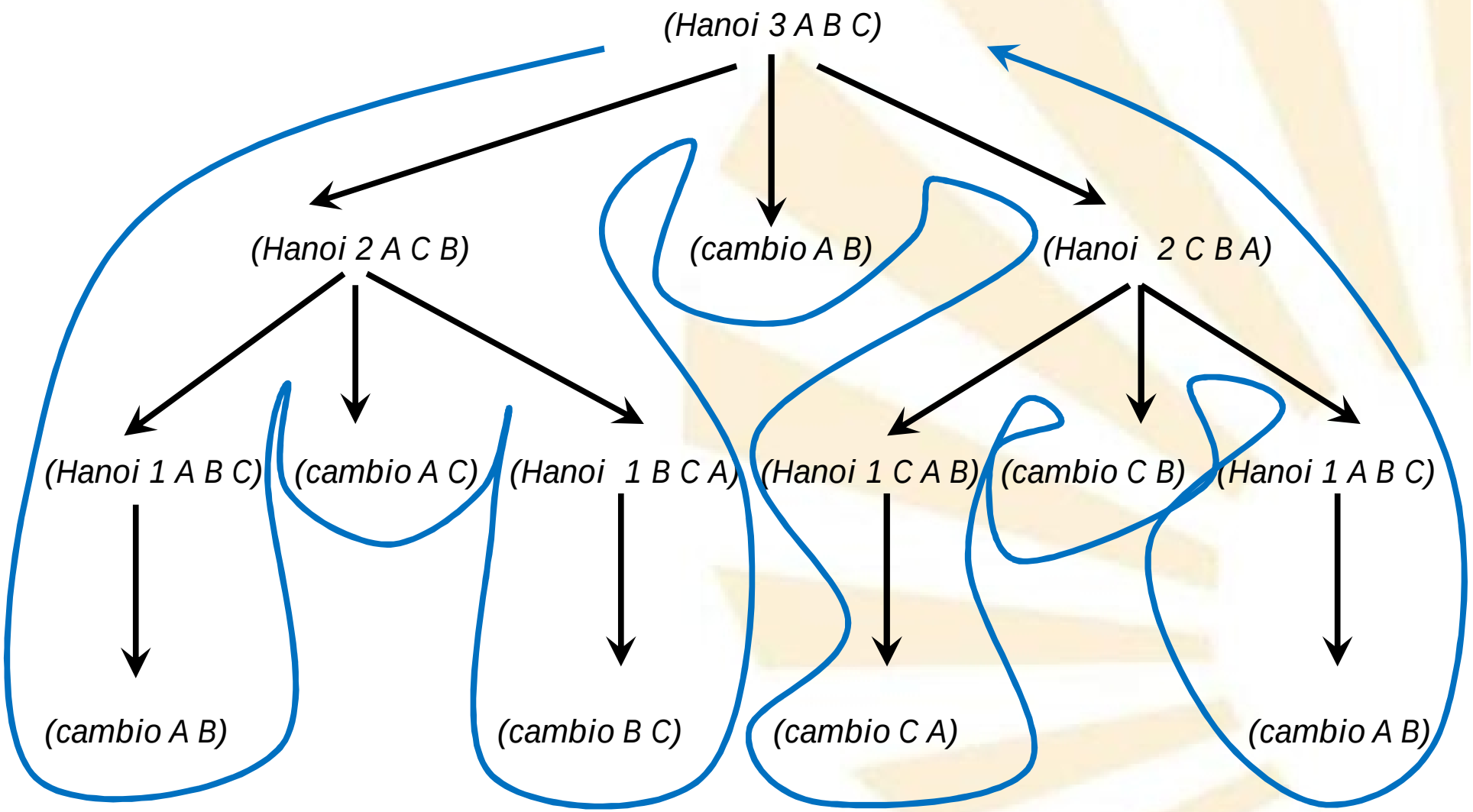
```
(define (Hanoi n a b c)
        (define (cambio x y)
                (display x)
                (display " --> ")
                (display y)
                (display "; ")
                1
        )
        ;; cuerpo de "Hanoi"
        (if (= n 1) (cambio a b)
          (+
                (Hanoi (- n 1) a c b)
                (cambio a b)
                (Hanoi (- n 1) c b a)
           )
        )
)
```

- **Torres de Hanoi: árbol de activación**



- **Torres de Hanoi**

Recorrido en profundidad



2. Recursión

- Recursión múltiple
  - **Ejemplo (2/2)**
    - **Torres de Hanoi**: 7 movimientos
      1. *A → B*
      2. *A → C*
      3. *B → C*
      4. *A → B*
      5. *C → A*
      6. *C → B*
      7. *A → B*

2. Recursión

- Descripción
- Recursión simple
- Recursión múltiple
- Recursión de cola
- Forma especial “let con nombre”

2. Recursión

- Recursión de cola
  - Concepto de “reducción”
  - Definición de “recursión de cola”
  - Ejemplo
  - Comparación
  - Más ejemplos

2. Recursión

- Recursión de cola
  - **Concepto de “reducción”**
    - **Transformación** de un problema en **otro más simple**
    - de forma que **no se requiera realizar ningún cálculo adicional** una vez que éste haya sido resuelto.

2. Recursión

- Recursión de cola
  - **Definición de "recursión de cola"**
    - Una función es recursiva de cola si todas sus transformaciones son **reducciones**
    - Término en inglés: ***"tail recursive"***

2. Recursión

• Recursión de cola

o **Ejemplo (1/5)**

▪ Factorial

```
(define (fac-cola n)
  ;; función auxiliar
  (define (fac-aux n resultado)
        (if (= n 0)  resultado
                     (fac-aux (- n 1) (* n resultado))
        )
    )
    ;; cuerpo de "fac-cola"
    (fac-aux n 1)
)

(fac-cola 4)   ➔ 24
```

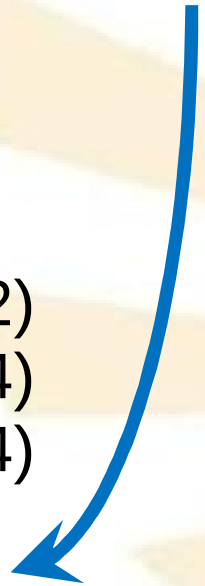
2. Recursión

- Recursión de cola
  - **Ejemplo (1/5)**
    - Factorial
      - ❑ Proceso "recursivo de cola"

(fac-cola 4)

(fac-aux 4 1)
(fac-aux 3 4)
(fac-aux 2 12)
(fac-aux 1 24)
(fac-aux 0 24)

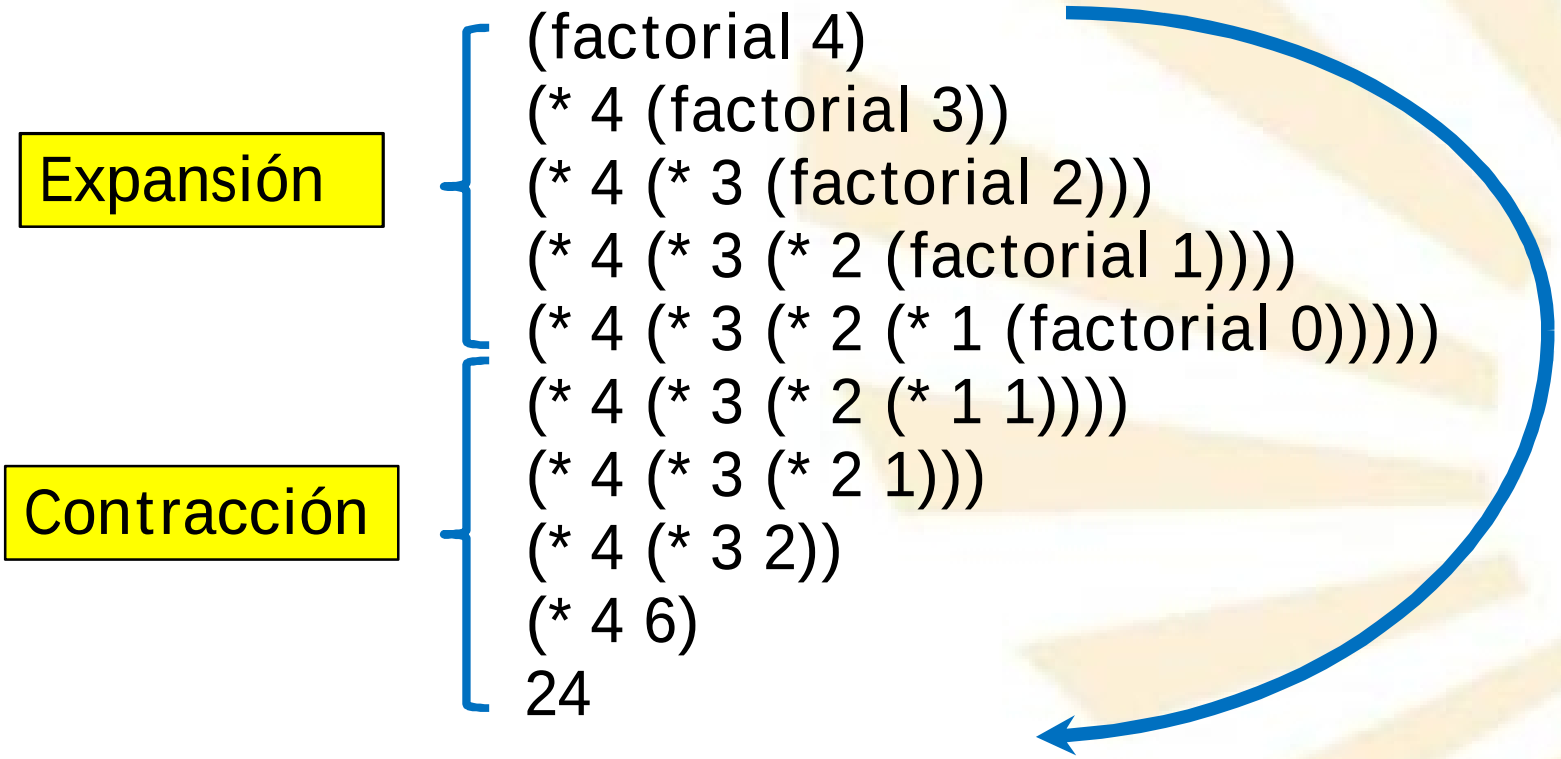
24



2. Recursión

- Recursión de cola
  - **Ejemplo (1/5)**
    - Factorial

❑ Proceso "recursivo lineal"



2. Recursión

- Recursión de cola
  - **Comparación**
    - Proceso “recursivo lineal”
    - Proceso “recursivo de cola”

2. Recursión

- Recursión de cola
  - **Comparación**
    - Proceso "recursivo lineal"
      - ❑ **Expansión**
        - ✓ Se efectúan las llamadas recursivas
        - ✓ Se encadenan una serie de operaciones "diferidas"
      - ❑ **Contracción**
        - ✓ Evalúa las operaciones "diferidas"

2. Recursión

- Recursión de cola
  - **Comparación**
    - Proceso "recursivo lineal"
      - ❑ **Observación**
        - ✓ El intérprete debe
          - ➢ almacenar los operadores y argumentos de las operaciones diferidas
          - ➢ recordar el orden en el que se deben evaluar dichas operaciones diferidas

2. Recursión

- Recursión de cola
  - **Comparación**
    - Proceso "recursivo lineal"
      - ❑ **Observación**
        - ✓ Necesita conocer **información adicional** "escondida"
          - ➢ mantenida por el intérprete
          - ➢ no contenida en las variables
        - ✓ Está información indica **qué** operaciones diferidas están pendientes de realizar

2. Recursión

- Recursión de cola
  - **Comparación**
    - Proceso "recursivo de cola"
      - Las **variables** del programa suministran una **descripción completa** del estado en el que se encuentra el proceso en cualquier instante.
      - En cada paso, el número de operaciones es conocido **"a priori".**
      - **No** necesita "operaciones diferidas"

2. Recursión

- Recursión de cola
  - **Ejemplo (2/5)**

$$\sum_{n=1}^{\infty}\frac{1}{n^2} = \frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\ldots=\frac{1}{1}+\frac{1}{4}+\frac{1}{9}+\ldots=\frac{\pi^2}{6}$$

```
(define (suma-reciprocos-cuadrados-perfectos n)
  (define (termino n)
    (/  1.0     (* n n))
    )
  (define (auxiliar n resultado)
    (if (<= n 0)
        resultado
        (auxiliar   (- n 1)  (+ resultado (termino n)))
     )
    )
 ;;
 (auxiliar   n  0.0)
 )
```

2. Recursión

- Recursión de cola
  - **Ejemplo (3/5)** $\sum_{n=1}^{\infty} \frac{1}{(4n-3)\times(4n-1)} = \frac{1}{1\times3} + \frac{1}{5\times7} + \frac{1}{9\times11} + \ldots = \frac{\pi}{8}$

```
(define (suma-pi-octavos n)
   (define (termino n)
      (/     1.0
             (*   (- (* 4 n)   3)    (- (* 4 n) 1)    )
       )
   )
  (define (auxiliar i n resultado)
        (if (> i n)   resultado
                      (auxiliar  (+ i 1) n (+ resultado (termino i)))
   )
   )
   ;;
  (auxiliar   1 n   0.0)
 )
```

2. Recursión

- Recursión de cola

$$seno(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

o **Ejemplo (4/5)**

```
(define (serie-seno x n)
  (define (termino x n)
    (* (expt -1 n)
       (/  (expt x (+ (* 2 n) 1))   (factorial (+ (* 2 n) 1)) )
     )
  )
  (define (auxiliar x n resultado)
    (if (< n 0)
      resultado
       (auxiliar x (- n 1) (+ (termino x n) resultado))
       )
    )
   ;;
     (auxiliar x n 0.0)
)
```

2. Recursión

- Recursión de cola
  - **Ejemplo (5/5)**
    - Raíz cuadrada usando el **método de Newton**

$$x_0 = 1$$

$$x_{n+1} = \frac{\frac{x}{x_n} + x_n}{2} \quad si\ n \geq 1$$

$$\lim_{n \to \infty} x_n = y = \sqrt{x}$$

2. Recursión

- Recursión de cola
  - **Ejemplo (5/5)**
    - Raíz cuadrada usando el método de Newton

$$x_0 = 1.0 \qquad x_1 = \frac{\frac{2}{1} + 1}{2} = 1.5$$

$$y = \sqrt{2} \qquad x_2 = \frac{\frac{2}{1.5} + 1.5}{2} = 1.41\overline{6}$$

$$x_3 = \frac{\frac{2}{1.41\overline{6}} + 1.41\overline{6}}{2} = 1.41421568$$

- **Ejemplo (5/5)**
  - Raíz cuadrada usando el **método de Newton**

```
(define (raíz x cota_error)
        (define (siguiente x y)
                (/  (+ (/ x y)  y) 2. )
        )
        (define (raíz-aux x y)
           (if (< (abs (- (* y y) x)) cota_error)
                        y
                        (raíz-aux x  (siguiente x y))
                )
        )
        ; cuerpo de "raíz"
        (raíz-aux x 1.0)
)
```

(raíz 2 0.001) ➔ 1.41421568

- **Ejemplo (5/5)**
  - Raíz cuadrada usando el **método de Newton**

(raíz 2 0.001)

(raíz-aux 2 1.0)

(raíz-aux 2 1.5)

(raíz-aux 2 1.41666...)

(raíz-aux 2 1.41411568)

➔ *1.41421568*

2. Recursión

- Descripción
- Recursión simple
- Recursión múltiple
- Recursión de cola
- Forma especial “let con nombre”

2. Recursión

- Forma especial “let con nombre”
  - **Sintaxis**

```
(let  <nombre-función>
    ;; Asignación inicial a los parámetros
        (
          (<parámetro 1>  <inicialización 1>)
          (<parámetro 2>  <inicialización 2>)
                    ...
          (<parámetro n>  <inicialización n>)
        )
      ;; cuerpo del “let con nombre”,
      ;; se puede llamar  recursivamente a
      ;; <nombre-función>
        <sentencia> ...
)
```

2. Recursión

- Forma especial “let con nombre”
  - **Significado**
    - Se evalúan en un orden no determinado las **sentencias de inicialización** de los parámetros
    - Se ejecuta el **cuerpo** de “let con nombre”
    - Si hay una **llamada recursiva** entonces los valores de los argumentos reales se asignan a los parámetros de ***let.***
    - Término en inglés: *named let*

2. Recursión

- Forma especial “let con nombre”
  - **Ejemplo**

```
(define (factorial-let-nombre n)
   (let   fact-let
         ;; Asignación inicial a los parámetros
          (
            (i              n)
            (resultado   1)
          )
         ;; cuerpo de let con nombre
          (if (or (= i 0) (= i 1))
                   resultado
                  ;; llamada recursiva
                  (fact-let (- i 1) (* resultado i))
          )
   )
)
```

Índice

3. Funciones pasadas como parámetros

- **Descripción**
- **Ejemplos**

3. Funciones pasadas como parámetros

- **Descripción**
  - **Definición de la función *f* que recibe a otra función *g* como parámetro formal**

    (**define** (*f* pf_1 pf_2 … pf_i ***g*** pf_{i+1} … pf_n)

    <*cuerpo de f*>

)

    donde pf_1, pf_2,…, pf_i , ***g***, pf_{i+1} , …, pf_n

    son los parámetros formales de ***f***

  - **Observación**
    - Solamente hay que **poner un nombre** al **parámetro** que va a actuar como **función**

3. Funciones pasadas como parámetros

- **Descripción**
  - **Uso de *g* dentro de la función *f***

    (*g* x_1 x_2 ... x_k)

    donde x_1, x_2 , ... , x_k

    son los parámetros reales de *g*
  - **Observación**
    - Solamente hay que **llamar** a la función con los parámetros reales que le correspondan.

3. Funciones pasadas como parámetros

- **Descripción**
  - **Paso de una función como parámetro de otra función *f***
    1. Función definida previamente
    2. Función creada con ***lambda***

3. Funciones pasadas como parámetros

- **Descripción**
  - **Paso de una función como parámetro de otra función *f***
    1. Función definida previamente

*(**define** (**h** ...)*

*<cuerpo de **h**>*

)

...

*(**f** pr_1 pr_2 ... pr_i **h** pr_{i+1} ... pr_n)*

donde pr_1, pr_2, ..., pr_i, **h,** pr_{i+1}, ..., pr_n

son los parámetros **reales** de ***f***

3. Funciones pasadas como parámetros

- **Descripción**
  - **Paso de una función como parámetro de otra función *f***

    2. Función creada con **lambda**

*(**f*** pr_1 pr_2 ... pr_i

(lambda (<parámetros) <cuerpo>)

pr_{i+1} ... pr_n

)

3. Funciones pasadas como parámetros

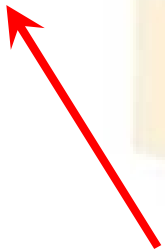
- **Ejemplo**
  - **Función que recibe otra función como parámetro**

```
(define (duplicar  g  x)
   (* 2 (g x))
)
```

3. Funciones pasadas como parámetros

- **Ejemplo**
  - **Función que recibe otra función como parámetro**

```
(define (duplicar  g  x)
   (* 2 (g x))
)
```



Uso de la función pasada como parámetro

3. Funciones pasadas como parámetros

- **Ejemplo**
  - Paso de una **función definida** como parámetro de la función *duplicar*

```
(define (cuadrado x)
   (* x x)
)
(duplicar cuadrado 7) ➔ 98
```

  - Paso de una **función creada** con ***lambda*** como parámetro de la función *duplicar*

```
(duplicar  (lambda (x) (* x x))  7) ➔ 98
```

3. Funciones pasadas como parámetros

- **Ejemplo**
  - **Paso de una función creada con lambda como parámetro de la función *duplicar***

(duplicar (lambda (x) (+ x 1)) 7) ➔ *16*

3. Funciones pasadas como parámetros

- **Ejemplo preparatorio**

$$\sum_{n=a}^{b} n^3 = a^3 + (a+1)^3 + (a+2)^3 + \ldots + (b-1)^3 + b^3$$

```
(define (suma-cubos inicial final)
        ;; función auxiliar
         (define (cubo x) (* x x x))
        ;; cuerpo de suma-cubos
        (if (> inicial final)
                0
                 (+
                        (cubo inicial)
                        (suma-cubos  (+ inicial 1) final)
                )
         )
)
```

3. Funciones pasadas como parámetros

- **Ejemplo preparatorio**

$$\sum_{n=a}^{b} n^3 = a^3 + (a+1)^3 + (a+2)^3 + \ldots + (b-1)^3 + b^3$$

```
(define (suma-cubos inicial final)
        ;; función auxiliar
         (define (cubo x) (* x x x))
        ;; cuerpo de suma-cubos
        (if (> inicial final)
                0
                 (+
                    (cubo inicial)
                    (suma-cubos  (+ inicial 1) final)
                 )
         )
)
```

Siguiente elemento de la serie

Término de la serie

3. Funciones pasadas como parámetros

- **Ejemplo preparatorio**

$$\sum_{\substack{n=1 \\ n=n+4}}^{\infty} \frac{1}{n\times(n+2)} = \frac{1}{1\times3}+\frac{1}{5\times7}+\frac{1}{9\times11}+\ldots=\frac{\pi}{8}$$

```
(define (suma-pi-octavos inicial final)
        (define (termino n)
            (/ 1.0  (* n (+ n 2.0)))
        )
        (if (> inicial final)
                0
                (+
                    (termino inicial)
                    (suma-pi-octavos  (+ inicial 4.0) final)
                )
        )
)
```

3. Funciones pasadas como parámetros

- **Ejemplo preparatorio**

$$\sum_{\substack{n=1 \\ n=n+4}}^{\infty} \frac{1}{n\times(n+2)} = \frac{1}{1\times3}+\frac{1}{5\times7}+\frac{1}{9\times11}+\ldots=\frac{\pi}{8}$$

```
(define (suma-pi-octavos inicial final)
        (define (termino n)
            (/ 1.0  (* n (+ n 2.0)))
        )
        (if (> inicial final)
                0
                (+
                    (termino inicial)
                    (suma-pi-octavos  (+ inicial 4.0) final)
                )
        )
)
```

Siguiente elemento de la serie

Término de la serie

3. Funciones pasadas como parámetros

- **Estructura general de los ejemplos preparatorios**

```
(define (<nombre> inicial final)
        (if (> inicial final)
            0
            (+
                (<término> inicial)
                 (<nombre> (<siguiente> inicial) final)
            )
        )
)
```

  - **Observación**
    - Las funciones *término* y siguiente son externas

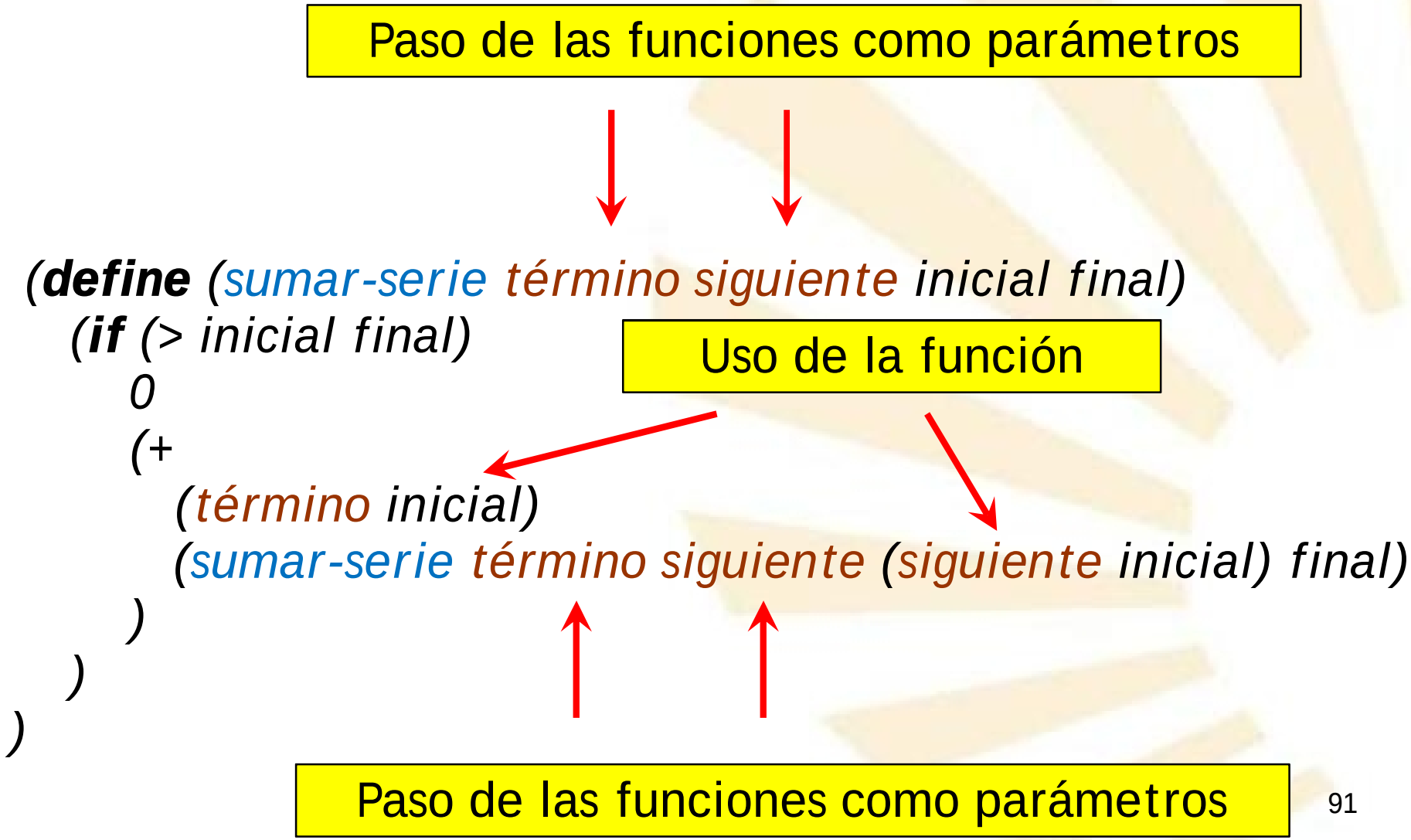
3. Funciones pasadas como parámetros

- **Estructura general de los ejemplos preparatorios**

```
(define (sumar-serie término siguiente inicial final)
   (if (> inicial final)
       0
       (+
          (término inicial)
          (sumar-serie término siguiente (siguiente inicial) final)
       )
   )
)
```

3. Funciones pasadas como parámetros

- **Estructura general de los ejemplos preparatorios**



3. Funciones pasadas como parámetros

- **Estructura general de los ejemplos preparatorios**

```
(define (suma-serie término siguiente inicial final)
  (if (> inicial final)
      0
      (+
        (término inicial)
        (suma-serie término siguiente (siguiente inicial) final)
      )
  )
)
```

- **Observación**
  - La función *suma-serie* permite sumar **cualquier serie**

3. Funciones pasadas como parámetros

- **Ejemplo: aplicaciones de la función *suma***

$$\sum_{n=a}^{b} n^3 = a^3 + (a+1)^3 + (a+2)^3 + \ldots + (b-1)^3 + b^3$$

```
;; función que calcula el término de la serie
(define (cubo x) (* x x x))

;; función para obtener el "siguiente" elemento
(define (suma-uno x)   (+  x  1.0))

;; llamada a sumar-serie
(sumar-serie   cubo  suma-uno  1 3) ➔ 36.0
```

3. Funciones pasadas como parámetros

- **Ejemplo: aplicaciones de la función *suma***

$$\sum_{n=a}^{b} n^3 = a^3 + (a+1)^3 + (a+2)^3 + \ldots + (b-1)^3 + b^3$$

;; llamada a sumar-serie usando funciones anónimas

```
(sumar-serie
        (lambda (x) (* x x x))
        (lambda (x) (+ x 1))
        1
        3
) ➔ 36.0
```

3. Funciones pasadas como parámetros

- **Ejemplo: aplicaciones de la función *suma***

$$\sum_{\substack{n=1 \\ n=n+4}}^{\infty} \frac{1}{n\times(n+2)} = \frac{1}{1\times3} + \frac{1}{5\times7} + \frac{1}{9\times11} + \ldots = \frac{\pi}{8}$$

```
;; función que calcula el término de la serie
    (define (termino-pi  x)
                (/ 1.0  (* x (+ x 2.0)))
    )
;; función para obtener el "siguiente" elemento
    (define (siguiente-pi  x)   (+ x 4))

;;llamada a la función sumar-serie
(* 8.0 (sumar-serie termino-pi siguiente-pi 1 10000)
    ➔ 3.14139265
```

3. Funciones pasadas como parámetros

- **Ejemplo: aplicaciones de la función *suma***

$$\sum_{n=1}^{\infty}\frac{1}{n^2} = \frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\ldots=\frac{1}{1}+\frac{1}{4}+\frac{1}{9}+\ldots=\frac{\pi^2}{6}$$

```
;; función que calcula el término de la serie
 (define (termino n)
                (/ 1.0  (* n n))
 )
;; función para obtener el "siguiente" elemento
(define (siguiente  n)   (+  n 1))

;; llamada a la función sumar-serie
(sumar-serie  termino siguiente 1 1000)
```

➔ ***1.64***39...

3. Funciones pasadas como parámetros

- **Ejemplo: aplicaciones de la función *suma***

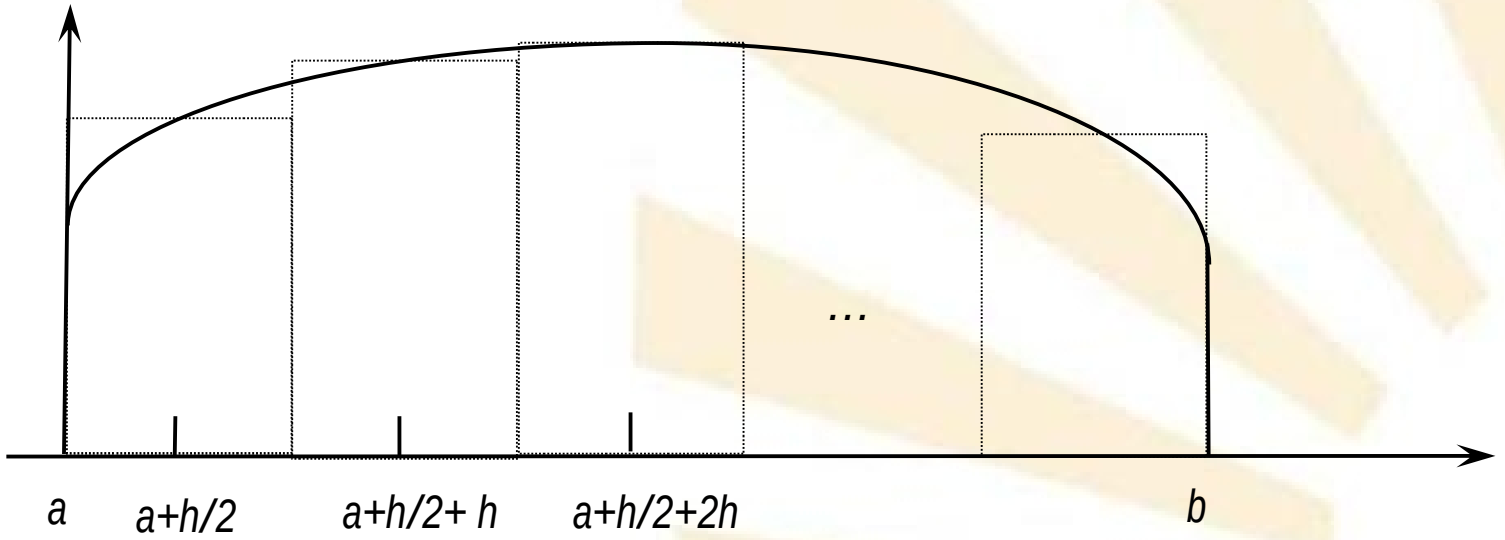
$$\sum_{n=1}^{\infty}\frac{1}{n!}=\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\ldots=\frac{1}{1}+\frac{1}{2}+\frac{1}{6}+\ldots=e=2{,}7182818284\ldots$$

```
;; función que calcula el término de la serie
   (define (termino n)
           (/ 1.0 (factorial n))
   )
;; función para obtener el "siguiente" elemento
   (define (siguiente  n )   (+  n 1))

;; llamada a la función sumar-serie
   (sumar-serie termino siguiente   1   10)
   ➔ 2.7182818284...
```

3. Funciones pasadas como parámetros

- **Ejemplo: aplicaciones de la función *suma***
  - **Integral definida**



$$\int_a^b f(x) = \left(f(a+h/2) + f(a+h/2+h) + f(a+h/2+2h) + \ldots \right) \times h$$

  - "*h*" debe ser suficientemente pequeño

3. Funciones pasadas como parámetros

- **Ejemplo: integral definida**

```
(define (integral f inicial final h)
  ;; función para obtener el siguiente elemento
  (define (sumar-h x) (+ x h))

  ;; cuerpo de integral
    (*
      (sumar-serie f sumar-h (+ inicial (/ h 2)) final)
      h
    )
)

(define (f x) x)
(integral f 0 1 0.001)                  ➔ 0.50000000
(integral (lambda (x) x) 0 1 0.001)     ➔ 0.50000000
```

3. Funciones pasadas como parámetros

- **Ejemplo**
  - **Series no paramétricas: versión iterativa**

```
(define (sumar-serie-v2  termino siguiente inicial final)
 (do
    (
     (i          inicial   (siguiente i))
     (resultado  0         (+ resultado (termino i)))
     )
  ;;
  ((> i final) resultado)
  ;; No hay cuerpo
  )
 )
```

3. Funciones pasadas como parámetros

- **Ejemplo**
  - **Series no paramétricas: versión iterativa**
    - Uso de la función *sumar-serie-v2*

$$\sum_{n=a}^{b} n^3 = a^3 + (a+1)^3 + (a+2)^3 + \ldots + (b-1)^3 + b^3$$

```
;; función que calcula el término de la serie
(define (cubo x) (* x x x))

;; función para obtener el "siguiente" elemento
(define (siguiente n)  (+ n 1))

;; llamada a la función
(sumar-serie-v2  cubo   siguiente 1 3) ➔ 36.0
```

3. Funciones pasadas como parámetros

- **Ejemplo**
  - **Series paramétricas: versión iterativa**
    - Función que suma **todas** las series paramétricas

```
(define (sumar-serie-paramétrica término siguiente inicial final x)
  (do
      (
       (i              inicial (siguiente i))
       (resultado 0.0     (+ resultado (término x i)))
       )
    ;;
    ((> i final) resultado)
    ;; No hay cuerpo
    )
  )
```

3. Funciones pasadas como parámetros

- **Ejemplo**
  - **Series paramétricas: versión iterativa**

```
;; término de la serie
(define (término-seno x n)
                (* (/
                     (expt -1 n)
                     (factorial (+ (* 2 n) 1))
                   )
                   (expt x (+ (* 2 n) 1))
                )
             )
 ;; función para obtener el "siguiente" elemento
(define (incrementar-uno  n)
                (+ n 1)
)
```

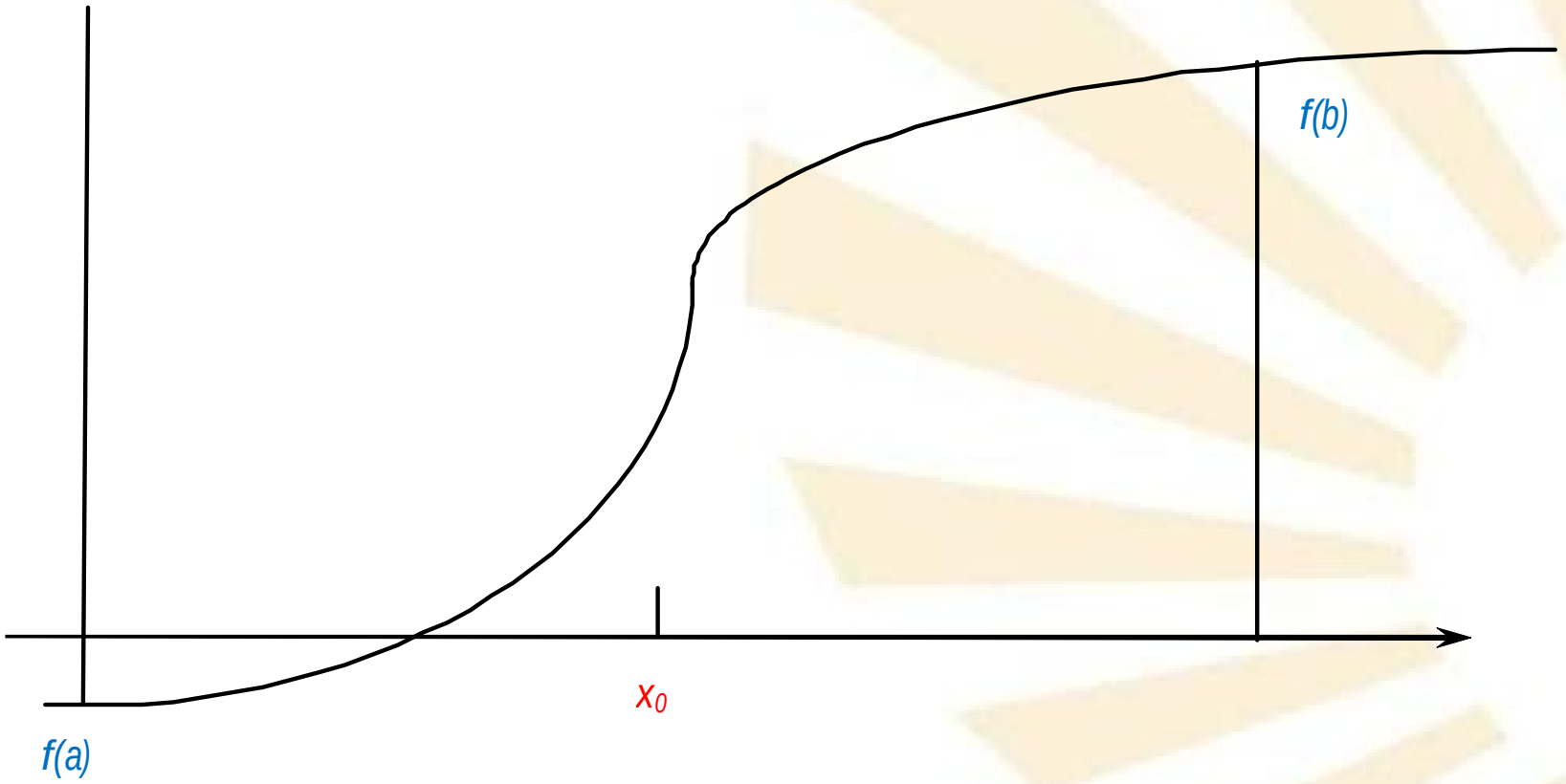
3. Funciones pasadas como parámetros

- **Ejemplo**
  - **Series paramétricas: versión iterativa**
    - *Uso de la función genérica para calcular seno(pi/2.0)*

```
(sumar-serie-paramétrica
        término-seno
        incrementar-uno
        0
        100
        (/ pi 2.0)
)
➔ 1.0000000000000002
```

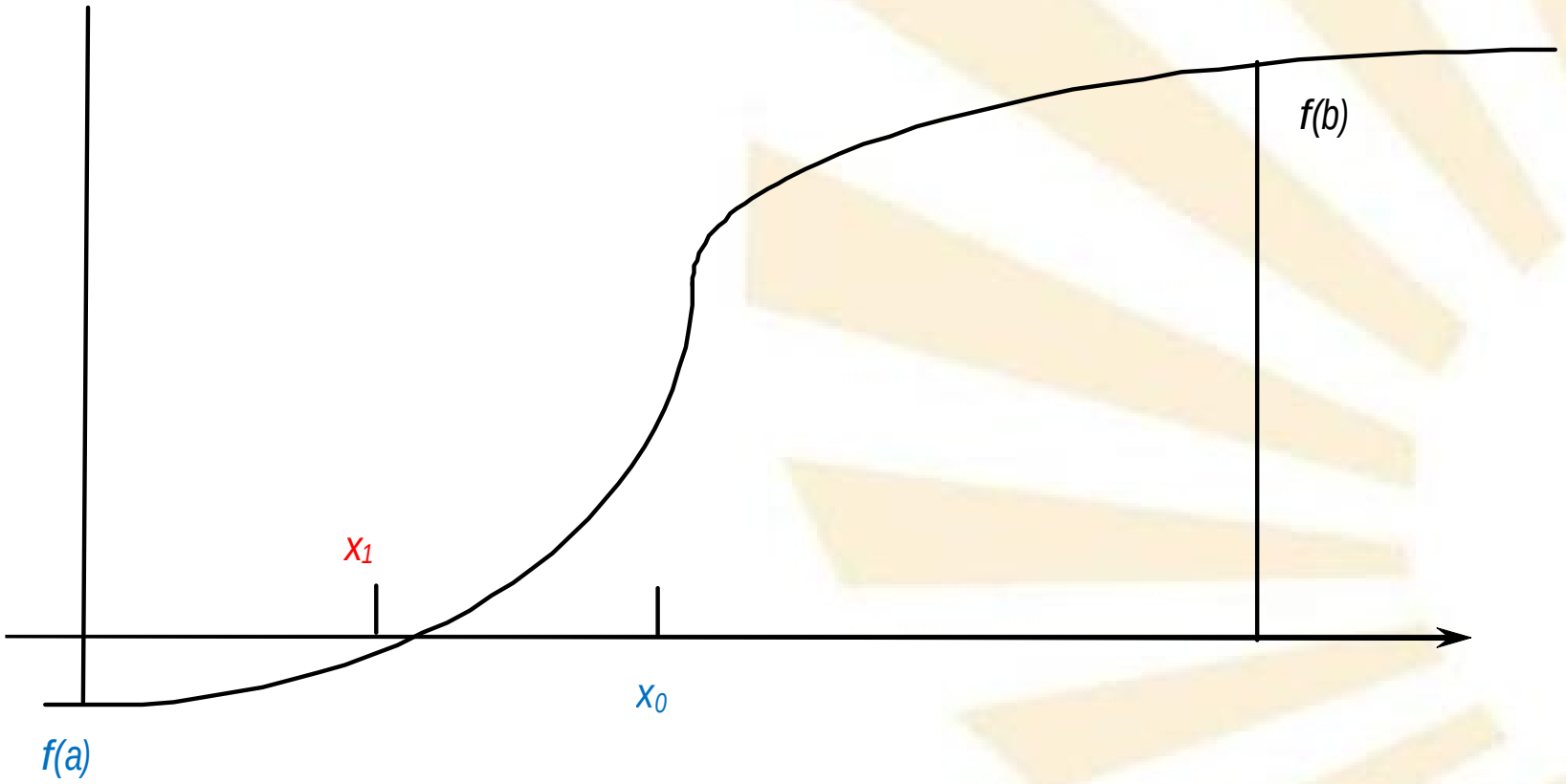
3. Funciones pasadas como parámetros

- **Ejemplo: raíz de una función mediante bisección**



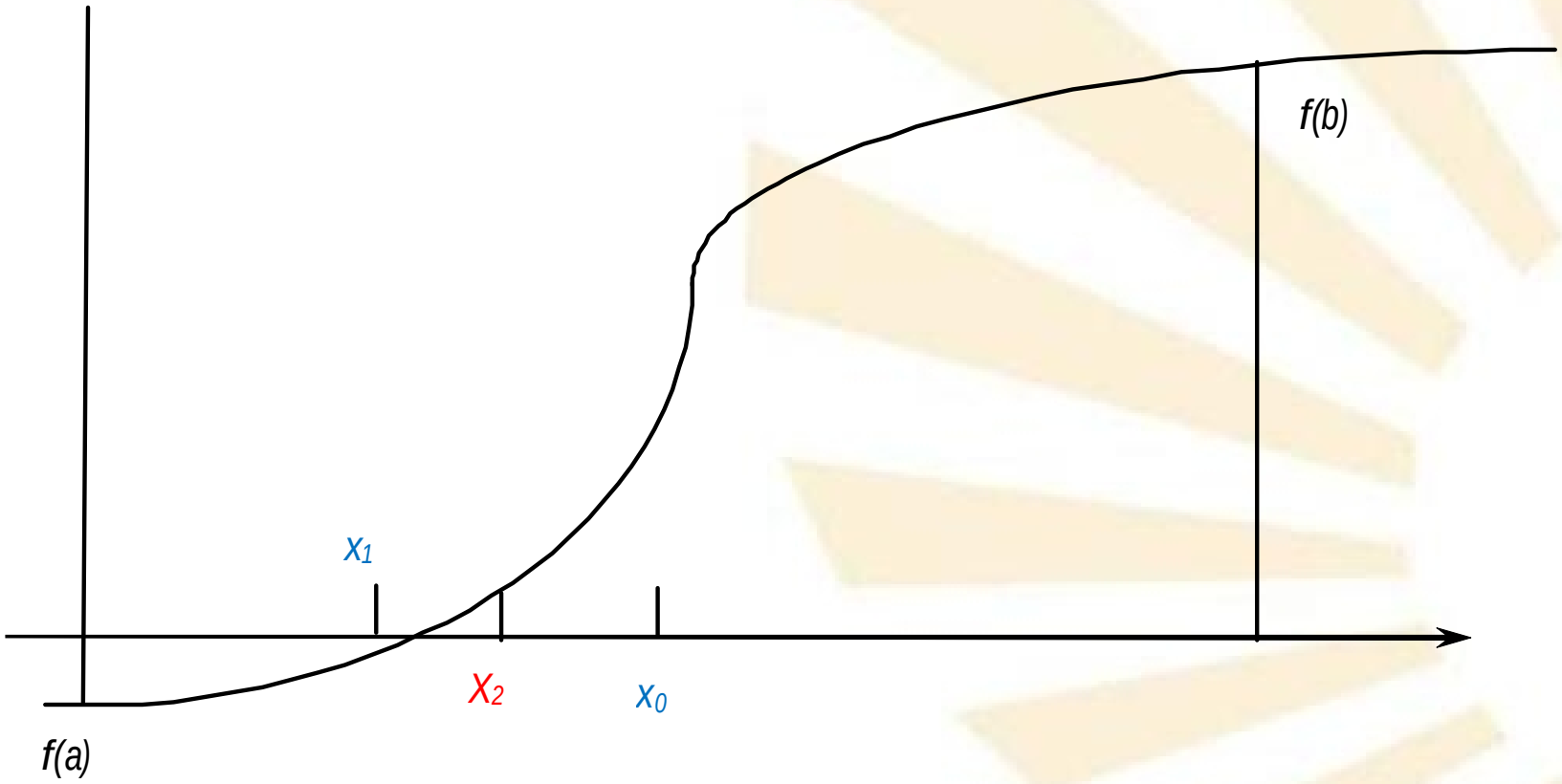
3. Funciones pasadas como parámetros

- **Ejemplo: raíz de una función mediante bisección**



3. Funciones pasadas como parámetros

- **Ejemplo: raíz de una función mediante bisección**



3. Funciones pasadas como parámetros

- **Ejemplo: raíz de una función mediante bisección**
  - **Restricción**
    - ***f(a)*** y ***f(b)*** deben tener signos diferentes

      f(a) y ***f(b) < 0***
    - Por ejemplo:
      - ***f(a) < 0***
      - ***f(b) > 0***

3. Funciones pasadas como parámetros

- **Ejemplo: raíz de una función mediante bisección**
  - **Funciones auxiliares**

```
(define (cercanos? x y)
        (< (abs (- x y)) 0.001)
)

(define (promedio x y)
        (/ (+ x y) 2.0)
)
```

- **Ejemplo: raíz de una función mediante bisección**

```
(define (buscar f negativo positivo)
   (define (cercanos? x y)
            (< (abs (- x y)) 0.001)  )
   (define (promedio x y)
            (/ (+ x y) 2.0) )
    ;; cuerpo de buscar
   (let      ;; Variable local
            ((mitad (promedio negativo positivo)))
              ;; Cuerpo de let
            (if (cercanos? negativo positivo)
                 mitad
                (let   ;; variable local
                        ((valor (f mitad)))
                    ;; cuerpo del let anidado
                    (cond
                          ((positive? valor)     (buscar f negativo mitad))
                          ((negative? valor)     (buscar f mitad positivo))
                          (else mitad)
                    )
                )
            )
       )
)
```

3. Funciones pasadas como parámetros

- **Ejemplo: raíz de una función mediante bisección**
  - **Llamada la función "buscar"**

```
(define (f x)
    (- (* x x) 2)
)
(buscar f 1 3) ➔ 1.41455078
;; Modo alternativo
(buscar   (lambda (x) (- (* x x) 2))   1   3)
➔ 1.41455078
```

3. Funciones pasadas como parámetros

- **Ejemplo: raíz de una función mediante bisección**
  - Se va a definir una nueva función denominada ***bisección*** para controlar el signo de ***f(a)*** y ***f(b)***

3. Funciones pasadas como parámetros

- **Ejemplo: raíz de una función mediante bisección**

```
(define (bisección f a b)
   (let (
            (valor-a   (f a))
            (valor-b    (f b))
        )
    (cond
    ( (and (negative? valor-a)  (positive? valor-b))
      (buscar f a b)
     )
    ( (and (negative? valor-b)   (positive? valor-a) )
     (buscar f b a)
     )
    (else  (display "los puntos iniciales tienen el mismo signo"))
     )
  )
)
```

3. Funciones pasadas como parámetros

- **Ejemplo: raíz de una función mediante bisección**

;; Ejemplo de llamada a la función bisección

(bisección sin 2 4) ➔ *3.14111328*

Índice

4. Funciones devueltas como resultados

- **Descripción**
- **Ejemplos**

4. Funciones devueltas como resultados

- **Descripción**
  - Se puede devolver una función como resultado de otra función de dos maneras
    1. Devolviendo el nombre de una función u operador
    2. Devolviendo una función creada con lambda

4. Funciones devueltas como resultados

- **Ejemplo**

1. Devolviendo el nombre de una función u operador

```
(define (elegir opcion f g)
  (if (even? opcion)
      f
      g
      )
  )
```

4. Funciones devueltas como resultados

- **Ejemplo**

1. Devolviendo el nombre de una función u operador

```
;; Ejemplos de llamadas a la función devuelta

((elegir 1 sqrt abs) 9) ➔ 9

((elegir 1 * + ) 9 2) ➔ 11

((elegir 2 (lambda (x) (+ x 1))
           (lambda (x) (* 2 x))
 )
           9
) ➔10
```

4. Funciones devueltas como resultados

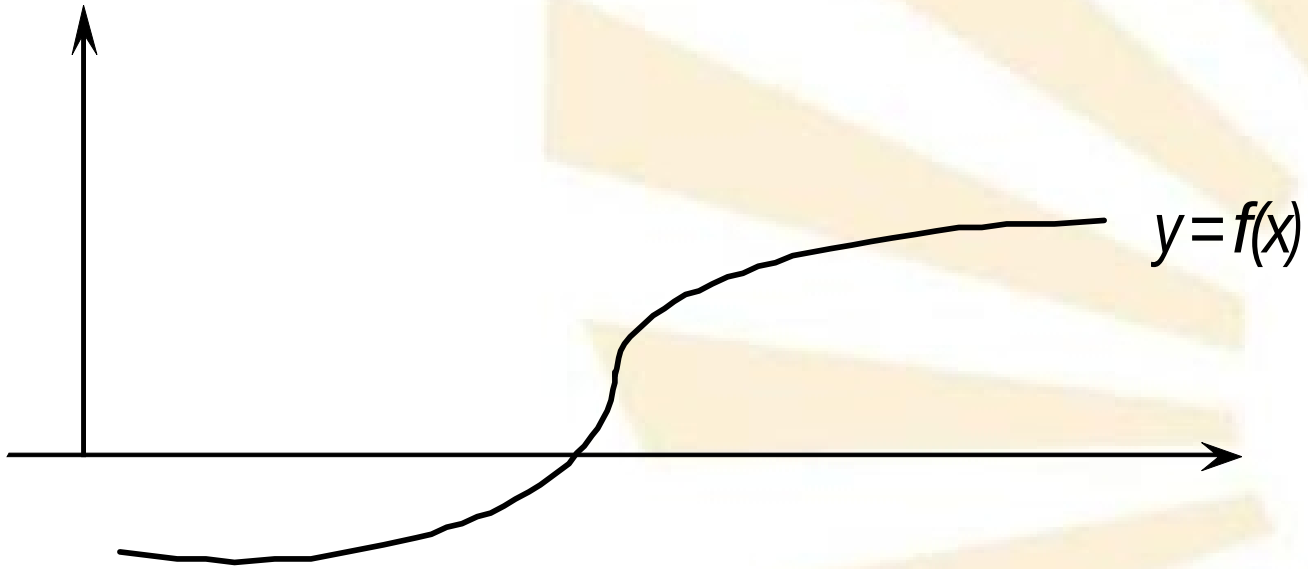
- **Ejemplo**

2. Devolviendo una función creada con lambda

```
(define (componer f g)
  (lambda (x)
    (f (g x))
    )
  )
;; Llamada a la función devuelta
( (componer sqrt sqrt)  81)  ➔ 3
```

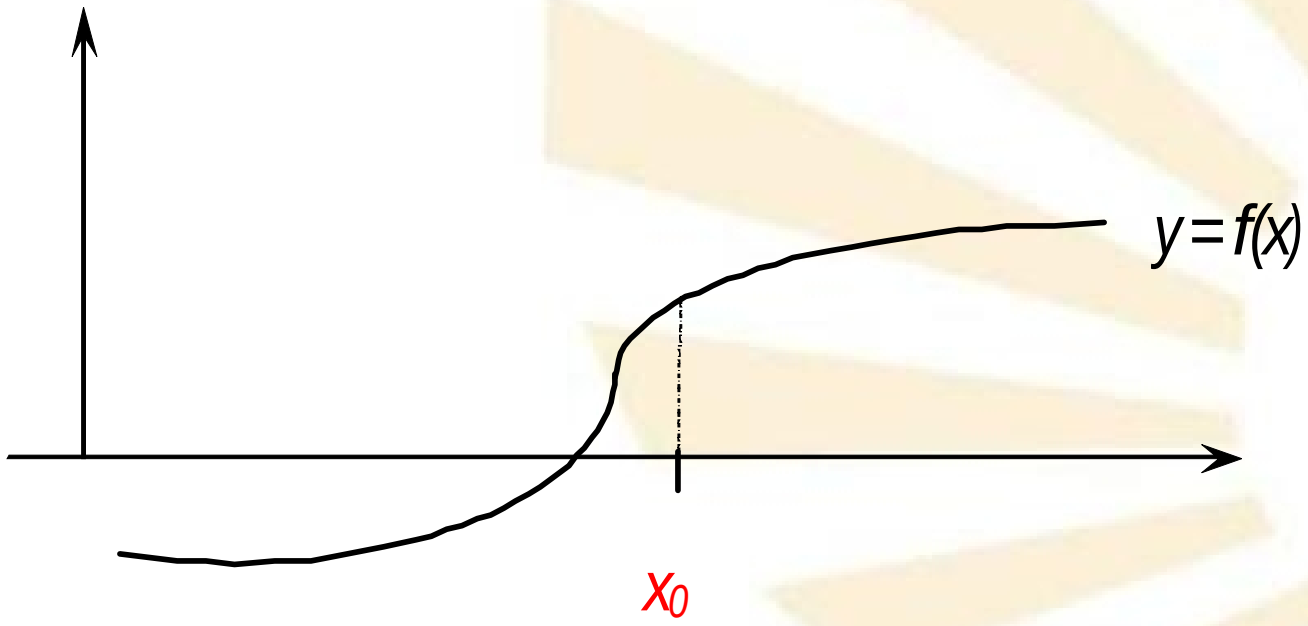
4. Funciones devueltas como resultados

- **Ejemplo**
  - **Método de Newton** para calcular la raíz de cualquier función



4. Funciones devueltas como resultados

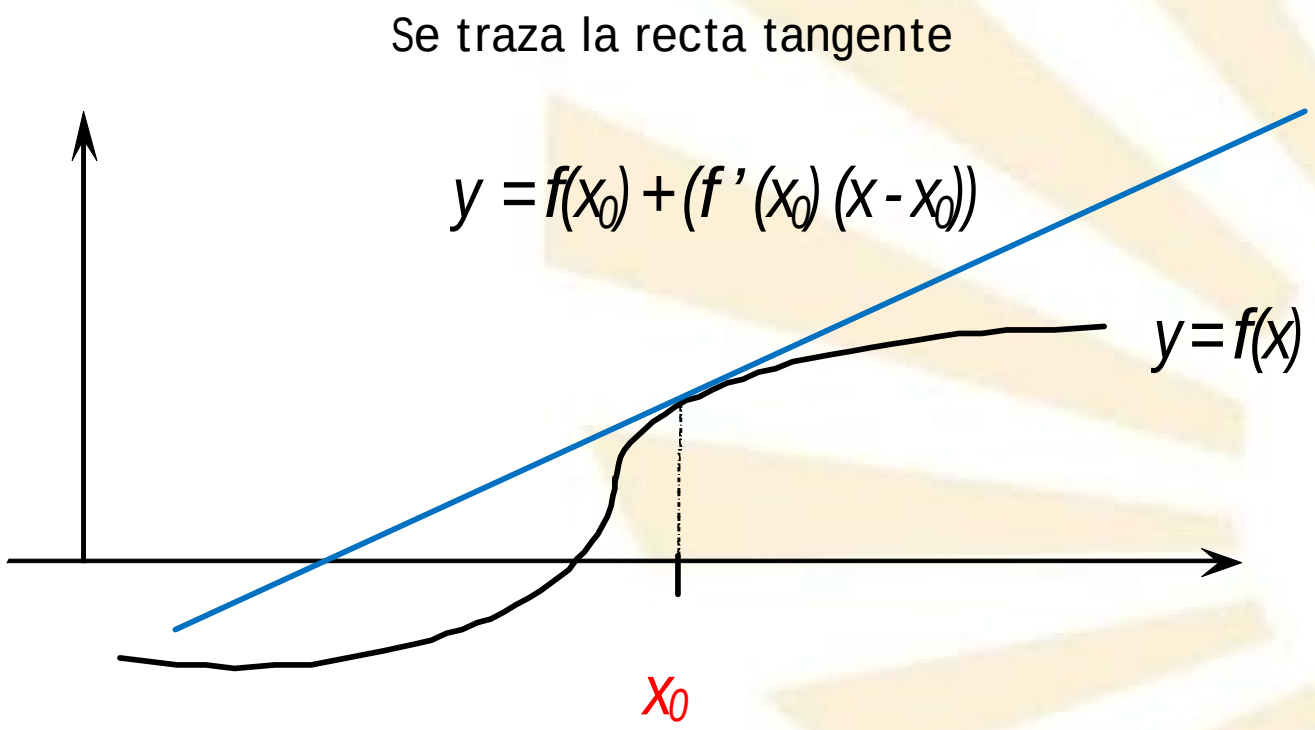
- **Ejemplo**
  - **Método de Newton** para calcular la raíz de cualquier función



Se elige un punto inicial

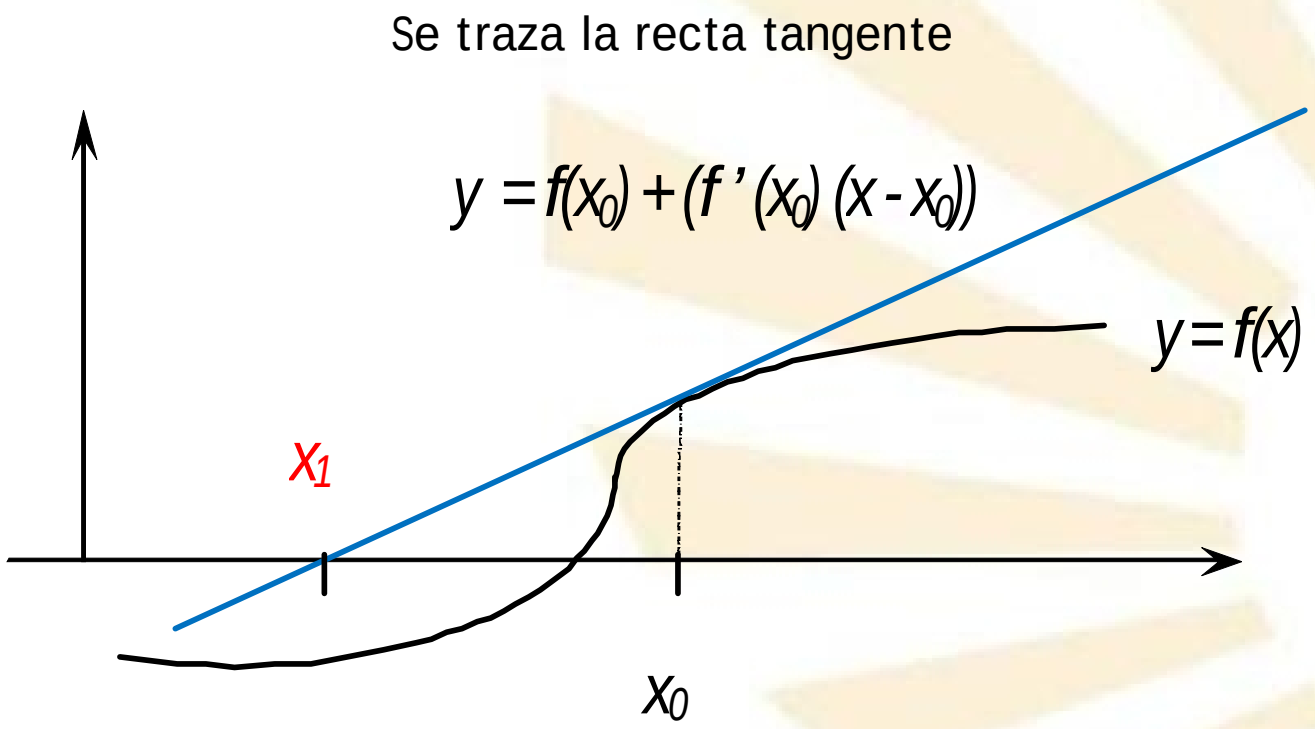
4. Funciones devueltas como resultados

- **Ejemplo**
  - **Método de Newton** para calcular la raíz de cualquier función



4. Funciones devueltas como resultados

- **Ejemplo**
  - **Método de Newton** para calcular la raíz de cualquier función



Se obtiene el punto de corte con el eje de abscisas

4. Funciones devueltas como resultados

- **Ejemplo**
  - **Método de Newton**
    - Sea x_0 una **aproximación** inicial a la raíz.
    - Si $\boldsymbol{f(x_0) \neq 0}$ entonces se obtiene el siguiente elemento de la sucesión en la intersección de
      - Eje de abscisas: $y = 0$
      - Recta tangente a la curva $y = f(x)$ en el punto $(x_0, f(x_0))$

$$y = f(x_0) + f'(x_0)\,(x - x_0)$$

    - El siguiente elemento es $x_1 = x_0 - \dfrac{f(x_0)}{f'(x_0)}$

4. Funciones devueltas como resultados

- **Ejemplo**
  - **Método de Newton**
    - En general, el término general de la sucesión es

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

4. Funciones devueltas como resultados

- **Ejemplo:**
  - **Método de Newton**
    - Aplicación a la función

      $y = f(x) = x^2 - a$

    - cuya solución es $y = \sqrt{a}$
    - obtendremos la siguiente sucesión:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - a}{2x_n} = \frac{x_n + \frac{a}{x_n}}{2}$$

4. Funciones devueltas como resultados

- **Ejemplo:**
  - **Método de Newton**
    - Funciones auxiliares:
      - ❑ Aproximación a la derivada de una función

$$f'(x) = Df(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

      - ❑ Cociente incremental

$$\frac{f(x+h) - f(x)}{h}$$

4. Funciones devueltas como resultados

- **Ejemplo:**
  - **Método de Newton**
    - Funciones auxiliares:
      - ❑ Aproximación a la derivada de una función

```
(define (cociente-incremental f h)
        (lambda (x)
                (/
                        (- (f (+ x h))  (f x))
                        h
                )
        )
)
```

4. Funciones devueltas como resultados

- **Ejemplo:**
  - **Método de Newton**
    - Funciones auxiliares:
      - ❑ Aproximación a la derivada de una función

*(**define** (cubo x) (* x x x))*

((cociente-incremental cubo 0.001) 5) ➔ *75.01500100*

o

((cociente-incremental (lambda (x) (x x x)) 0.001) 5)*
➔ *75.01500100*

4. Funciones devueltas como resultados

- **Ejemplo:**
  - **Método de Newton**
    - Funciones auxiliares:
      - ❑ Función para comprobar si la aproximación a la raíz es buena

```
(define (bueno? x f)
        (< (abs (f x)) 0.001))
```

4. Funciones devueltas como resultados

- **Ejemplo:**
  - **Método de Newton**
    - Funciones auxiliares:
      - ❑ Función que obtiene el siguiente elemento de la sucesión

```
(define (siguiente x f)
        (-      x
                (/      (f x)
                        ((cociente-incremental f 0.001)  x)
                        )
                )
        )
```

4. Funciones devueltas como resultados

- **Ejemplo:**
  - **Método de Newton**

```
(define (newton f inicial)
        (if (bueno? inicial f)
            inicial
            (newton f (siguiente inicial f))
        )
)
```

(newton (lambda (x) (- (x x) 2)) 1)* ➔ *1.41421657*

UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA

DEPARTAMENTO DE
INFORMÁTICA Y ANÁLISIS NUMÉRICO

PROGRAMACIÓN DECLARATIVA

INGENIERÍA INFORMÁTICA

ESPECIALIDAD DE COMPUTACIÓN

CUARTO CURSO

PRIMER CUATRIMESTRE

Tema 4.- Recursión e iteración