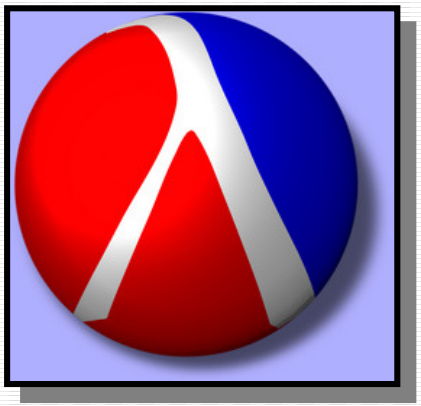
Fundamentos de Programación

Condicionales



Profesor: Daniel Wilches Maradei
Diapositivas Originales: Jesús A. Aranda

Universidad del Valle

Qué es un condicional ?

- Es la forma de decirle al lenguaje de programación como tomar decisiones sobre lo que debe ejecutar.
- Sin ellas no podríamos crear funciones que calculen el máximo de varios números, por ejemplo.
- Es el equivalente a la palabra SI del español

Tipos de condicionales

- Si deseamos indicar que si una condición se cumple se ejecute una instrucción, y si no se cumple, se ejecute otra: usaremos el if

Sintaxis:

(if condición si-si si-no)

Ejemplo

- Calcular el máximo de dos números:

```
(define (máximo A B)
   (if (> A B) A B))
```

- Literalmente dice: Si A > B, entonces retorne A, sino, retorne B

- Es decir:

```
(if (> A B) A B))     →     si A>B entonces A sino B
```

Ejemplo

- Dados 3 números calcule el mayor de ellos:

```
(define (máximoTresNumeros A B C)
   (if  (> A B)
        (if (> A C) A C)    ; si se cumple el primer if
        (if (> B C) B C)    ; si no se cumple el primer if
   ))
```

Ejemplo

- Dado un número devolver “N” si el número es negativo, o “P” si el número es positivo.

```
(define (negativo-positivo A)
  (if (> A 0) "P" "N"))
```

Ejercicio

- Pero y qué pasa si el número era cero ?
  Modifique la función para que devuelva “C” si el número es 0.

Condiciones múltiples: cond

- Hay otra construcción del Scheme que nos permite evaluar condicionales, se llama cond, y es usada cuando son muchas las condiciones que queremos evaluar.
- Por ejemplo, realicemos el ejercicio anterior con cond:

Ejemplo

```
(define (positivo-negativo-cero A)
  (cond
      ( (= A 0) "C")
      ( (> A 0) "P")
      ( else "N")
  ))
```

Condiciones múltiples: cond

- El cond es la manera abreviada de escribir múltiples condicionales, es equivalente a usar muchos if anidados.
- La cláusula else (sino en español) es una forma de decirle al cond que si ninguna otra condición se cumple, entonces ejecute la última instrucción.
- La cláusula else siempre debe ir al final

Operadores booleanos

- En Scheme podemos hacer uso de los operadores lógicos (y,o,no) en inglés. Estos sirve para unir condiciones:

```
(cond
      ((< edad 15)                          “niño”)
      ((and (> edad 15) (< edad 40)) “joven”)
      (else                                 “mayor”)
)
```

Operadores booleanos

and	Y lógico: la condición se cumple si todos sus argumentos se cumplen
or	O lógico: la condición se cumple si alguno de sus argumentos se cumple
not	Negación: La condición solo se cumple si su argumento no se cumple.

Ejemplos

```
(and (> 5 3) (< 8 9))                    -> false
(or (> 5 3) (< 8 9))                     -> true
(not (> 5 3))                            -> false
(and (or (> 2 1) (> 3 2)) (> 4 PI))      -> true
```

Ejercicio

- Escriba una función que reciba 3 notas de un estudiante, y calcule su promedio.

Ejercicio

- Escriba una función que reciba 3 notas de un estudiante, y calcule su promedio.
- Escriba una función que reciba 3 notas de un estudiante y diga si su promedio es "mayor a 3", "es 3" o si es "menor que 3"

Evaluación por corto circuito

- Scheme no tiene que evaluar siempre todos los componentes de una condición para determinar si es cierta (true) o falsa (false).
- A él método inteligente que permite conocer el resultado de una expresión booleana sin evaluar todas las cláusulas se le conoce como: Corto Circuito.

Evaluación por corto circuito

- Ejemplo:

  ```
  (and (> 4 5) (< 5 6))
  ```

  En cuanto Scheme evalúa la primera cláusula `(> 4 5)` se da cuenta de que el resultado de la evaluación será false, sin importar el resultado de las demás cláusulas. Porqué ?

Evaluación por corto circuito

- Igual sucede con el OR:

```
(or (> 7 1) (> 10 90))
```

  En cuanto Scheme evalúa la primera cláusula `(> 7 1)` se da cuenta de que el resultado de la evaluación será true.

Evaluación por corto circuito

- Ya que conocemos algo más de cómo Scheme evalúa las expresiones booleanas podemos concluir que el orden de las cláusulas en un los or y los and SI importa ! Pero no para el resultado, sino para la velocidad de ejecución.

Predicados

- Los predicados son funciones de Scheme que reciben un valor y retornan un valor booleano (true o false), y su objetivo es determinar la satisfacción o no de cierta propiedad en el valor que reciben.
- Los predicados se componen de un nombre bastante explicativo, seguidos de un signo de interrogación: ?
- Ejemplo:

number? string? odd?

Predicados definidos por el usuario

- Como un predicado no es más que una función, definirlos es también igual:

```
;; Determina si la edad ingresada es menor que
;; 18
(define (menor-de-edad? edad)
    (< edad 18))
```

Ejercicio

- Diseñemos una función para una tienda, que reciba la cantidad de DVD's que un cliente va a comprar y calcule el precio que se le debe cobrar. Teniendo en cuenta claro, las promociones de la tienda:
  - Los DVD's individuales valen $700 pesos
  - Si compra entre 10 y 50 se le cobra $600 por cada uno
  - Si compra más de 50, se le cobran $500 por cada uno

Datos adicionales sobre los condicionales

- En el cond, pueden usarse paréntesis o corchetes para encerrar las condiciones. Es solo por legibilidad, no hay ninguna diferencia en cuanto a funcionamiento.

```
(cond
  [ (> A B) "es mayor" ]
  [ (< A B) "es menor" ]
  [ else "son iguales" ]
)
```

Datos adicionales sobre los condicionales

- Si con el cond vamos a comparar una variable contra muchos valores diferentes, puede usarse una forma abreviada, el case:

```
(case variable
    ((1) "A")
    ((2) "B")
    (else "ni A, ni B")
 )
```

- El case no funciona en el Lenguaje "Beginning Student" en el que estamos programando, pero ya saben que existe

Ejercicio

- Ahora, supongamos que queremos validar que el dato ingresado a la función sea un número, con el fin de evitar este tipo de problemas:

  (valor-dvds "cincuenta") -> Error !!

- Podemos hacer uso del predicado "number?"

Bonus: Interacción con el usuario

- No funciona en el lenguaje "Beginning Student". De aquí hasta el final de esta presentación deberán colocar el lenguaje "Advanced Student"
- Para mostrar mensajes en el área de interacciones de Scheme usaremos la función display:

```
(display "Cómo te llamas ?")
```

- Para pedirle al usuario que digite un string por teclado, usaremos la función read:

```
(define nombre-usuario (read))
```

Bonus: Interacción con el usuario

- Ejemplo:

```
(display "Cómo te llamas ?")
(define nombre-usuario (read))
(display (string-append
                "Hola "
                (symbol->string nombre-usuario)
                " !!" ))
```

Ejemplo útil de interacción con el usuario

- Ejemplo:

```
(display "Digite 2 números:")
(define numeroA (read))
(define numeroB (read))
(display "Desea sumar o restar (S o R)?")
(define operación (symbol->string (read)))
(display
  (cond
        ((string=? operación "S") (+ numeroA numeroB))
        ((string=? operación "R") (- numeroA numeroB))
        (else "Operación inválida")
))
```