Datos y operadores

José A. Alonso y María J. Hidalgo

Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Sesión inicial

- Sesión

```
# scm
> 23
23
> (+ 2 3)
5
> (exit)
#
```

- Comentarios:
  - Expresiones: números, símbolos y listas.
  - Evaluación de expresiones.
  - Ciclo básico: Leer–Evaluar–Escribir.

Números

- Ejemplos

```
> 3
3
> -2.7
-2.7
> .4
0.4
> 333333333333333333333333333333333
333333333333333333333333333333333
```

- Comentarios:
  - Números enteros.
  - Números reales.
  - Números enteros de precisión arbitraria.

Variables

- Ejemplos

```
> pi
ERROR: unbound variable:  pi
> (define pi 3.14)
#<unspecified>
> pi
3.14
> (+ pi 2)
5.14
```

- Comentarios:
  - **Definición:** `(define <símbolo> <expresión>)`
  - **Mensaje de error**
  - **Variable ligada**

Literales

- Ejemplos

```
> pi
3.14
> (quote pi)
pi
> 'pi
pi
> diez
ERROR: unbound variable:  diez
> 'diez
diez
```

- Comentarios:
  - Tilde: `(quote <símbolo>)`
  - Abreviatura: `'<simbolo>`
  - Valor literal del símbolo

Variables y valores literales

- Ejemplos:

```
> pi
3.14
> (define pi/2 (/ pi 2))
#<unspecified>
> pi/2
1.57
> (define numero ´pi)
#<unspecified>
> numero
pi
```

Operaciones aritméticas

- Ejemplos:

```
(define pi 3.14)                    =>  #<unspecified>
(+ 3 4 pi)                          =>  10.14
(- 7 5 8 9)                         =>  -15
(* 3 (- 12 5))                      =>  21
(expt 2 3)                          =>  8
(sin 3)                             =>  0.141120008059867
(sin (/ pi 2))                      =>  0.999999682931835
(sqrt 9)                            =>  3.0
(+ (expt 2 (+ 6 3 )) (sqrt 25))     =>  517.0
(sqrt 2 5)                          =>  ERROR: Wrong number of args ...
(pi / 3)                            =>  ERROR: Wrong type to apply:  3.14 ...
(-1 3)                              =>  ERROR: Wrong type to apply:  -1 ...
```

- Comentarios:
  - **Expresión aritmética:** `(<operador> <operando-1> ... <operando-n>)`
  - **Evaluación de expresiones**

Construcción de listas

- Ejemplos

```
()                                  =>  ()
(cons 3 ())                         =>  (3)
(define lista-1 (cons 3 ()))        =>  #<unspecified>
lista-1                             =>  (3)
(cons 'a lista-1)                   =>  (a 3)
(define lista-2 (cons 'a lista-1))  =>  #<unspecified>
lista-2                             =>  (a 3)
```

- Comentarios:
  - Lista vacía: `()`
  - Constructor: `(cons <expresion> <lista>)`
  - Definición de lista

Listas literales

- Ejemplos:

```
(a b c)                  =>  ERROR: unbound variable:  a
(quote (a b c))          =>  (a b c)
´(a b c)                 =>  (a b c)
(define lista-1 ´(b c))  =>  #<unspecified>
(cons ´a lista-1)        =>  (a b c)
```

- Comentarios:
  - Tilde: `(quote <lista>)`
  - Abreviatura: `´<lista>`
  - Valor literal de una lista

Descomposición de listas

- Ejemplos

```
(car ´(b 3 a 5))                    =>  b
(car ´((b) (3 a) 5))                =>  (b)
(car ´(b3 a 5))                     =>  b3
(cdr ´(b 3 a 5))                    =>  (3 a 5)
(cdr ´((b) (3 a) 5))                =>  ((3 a) 5)
(cdr ´(b3 a 5))                     =>  (a 5)
(car (cdr ´((b) (3 a) 5)))          =>  (3 a)
(car (car (cdr ´((b) (3 a) 5))))    =>  3
(caadr ´((b) (3 a) 5))              =>  3
```

- Comentarios:
  - **Primero:** `(car <lista>)`
  - **Resto:** `(cdr <lista>)`
  - **Composición:** `(c...r <lista>)`

Pares punteados

- Ejemplos:

```
(cons 3 5)                 =>  (3 . 5)
(define par (cons 3 5))    =>  #<unspecified>
par                        =>  (3 . 5)
(car par)                  =>  3
(cdr par)                  =>  5
'(a . b)                   =>  (a . b)
(cdr '(a . b))             =>  b
'(a . ())                  =>  (a)
(cdr '(a . ()))            =>  ()
```

Predicados de tipo

- Ejemplos:

```
(number? 3.7)                    =>  #t
(number? (/ 1 2))                =>  #t
(number? 'dos)                   =>  #f
(symbol? 'dos)                   =>  #t
(symbol? ())                     =>  #f
(list?  ())                      =>  #t
(list? '(a b))                   =>  #t
(list? (cons 1 (cons 2 ())))     =>  #t
(list? (cons 'a 'b))             =>  #f
(pair? (cons 'a 'b))             =>  #t
(pair? '(a b))                   =>  #t
(pair? ())                       =>  #f
```

Predicados de tipo

- Ejemplos

```
(null? ())                  =>  #t
(boolean? #f)               =>  #t
(symbol? #f)                =>  #f
(procedure? cons)           =>  #t
(procedure? +)              =>  #t
```

- Comentarios:
  - Booleanos.
  - Definición de predicado
  - Sintaxis: `(<predicado-de-tipo> <expresión>)`
  - Predicados de tipo: `number?`, `symbol?`, `list?`, `pair?`, `null?`, `boolean?`, `procedure?`

Predicado de igualdad numérica

- Ejemplos:

```
(= 2 (/ 8 4))                          =>  #t
(= 1 1.0)                              =>  #t
(= (* 2 7) 8)                          =>  #f
(define dos 3)                         =>  #<unspecified>
(= dos 3)                              =>  #t
(= (car '(-1 si)) (/ -20 (* 4 5)))     =>  #t
(= 'a 'a)                              =>  ERROR: =: Wrong type in arg1 a
```

- Comentarios:
  - Igualdad numérica: `(= <expresión-1> <expresión-2>)`

Predicado de igualdad simbólica

- Ejemplos:

```
(eq? 'a 'a)           =>  #t
(eq? 'a 'b)           =>  #f
(define pepe 'a)      =>  #<unspecified>
(define juan 'b)      =>  #<unspecified>
(eq? pepe juan)       =>  #f
(eq? pepe 'A)         =>  #t
(eq? 1 (+ 1 0))       =>  #t
(eq? 1 1.0)           =>  #f
(eq? 1.0 1.0)         =>  #f
(eq? '(a b) '(a b))   =>  #f
```

- Comentarios:
  - Igualdad simbólica: `(eq? <expresión-1> <expresión-2>)`

Predicado de igualdad numérica–simbólica

- Ejemplos:

```
(eqv? 1.0 1.0)         =>  #t
(eqv? 1.0 1)           =>  #f
(= 1.0 1)              =>  #t
(eqv? '(a b) '(a b))   =>  #f
```

- Comentario:
  - Igualdad numérica–simbólica: `(eqv? <expresión-1> <expresión-2>)`

Predicado de igualdad de listas

- Ejemplos:

```
(equal? ´(a b) ´(a b))                    =>  #t
(equal? ´(a b) ´(b a))                    =>  #f
(equal? ´(a b) ´((a) b))                  =>  #f
(equal? ´((a) b) ´((a) b))                =>  #t
(equal? (cdr ´(a c d)) (cdr ´(b c d)))    =>  #t
```

- Comentario:
  - Igualdad: `(equal? <expresión-1> <expresión-2>)`

Bibliografía

- [Abelson–96]
  Cap. 1: "Building abstractions with procedures".
- [Springer–94]
  Cap. 1: "Data and operators".