|  |  |
| --- | --- |
| Dialecto de lisp. | Scheme |
| Extensión | .scm o .ss |
| Definir variables | Variables globales:  (define var1 value)  Variables con let:  (let ((var1 value))  ...  ámbito de var1  ...) |
| Comentarios | ;;Este es un comentario en scheme |
| Operadores matemáticos | + \* - /  abs, exp, expt, max, min, sqrt |
| Funciones | (define (nombre\_funcion argumentos)  sentencia  …  Sentencia  )  ó  (define nombre\_funcion  (lambda (argumentos)  (expresión))) |
| Operadores booleanos | and, or, not  (and #t #t) -> #t  (or #t #t) -> #t  (not #t) -> #f |
| Tipos de datos | Átomo: Elemento básico -> Identificador, letras, dígitos.  Boléanos: #t(verdadero), #f(falso)  Números: 42, 2+3i, 3.16, 4/6  Listas: (1,2,3)  Carácter: #\a, etc.  String: “Hola” |
| Input / output | current-input-port, current-output-port y current-error-port. |
| Arreglos / listas | (list 1 2 3 4)  ‘(1 2 3 4)  (car ‘(1 2 3 4)) ;primer elemento de la lista  (cdr '(1 2 3 4)) ; resto de la lista  '() ;lista vacía  (cdr '(1)) ; devuelve la lista vacía  (null? (cdr '(1))) ;comprueba si una lista es vacía  (append '(1) '(2 3 4) '(5 6)) ; construye una lista nueva concatenando los argumentos |
| Recursión | (define (factorial n)  (cond ((= n 0) 1)  (else (\* n (factorial (- n 1))))))  (factorial 5)  ;; => 120 |
| Estructuras de control | (if (condición)  (expresión verdadera)  (expresión falsa))  ó  (cond  (condición 1 expresion {expresion})  (condición 2 expresion {expresion})  (else expresión {expresion})  )    \*Solo evalúa la primera condición valida. |