Lisp

Standardfunktionen

`(QUOTE expr)`	Liefert den Ausdruck ohne weitere Auswertung zurück
`(SETQ var value)`	Setzt eine Variable und liefert den neuen Wert
`(EVAL var)`	Liefert den Wert der Variablen
`(TYPEP expr type)`	Typecast, type u.a. NUMBER, SYMBOL, INTEGER, ...
`(LIST elem elem ...)`	Fügt die Elemente zu einer Liste zusammen
`(CONS elem list)`	Fügt ein Element vor einer Liste an
`(APPEND list list ...)`	Fügt die Listenelemente zu einer Liste zusammen
`(REVERSE list)`	Liste mit umgekehrter Reihenfolge der Elemente
`(CAR list)`	Liefert das erste Element der Liste
`(CDR list)`	Liefert die Liste ab dessen zweitem Element
`(CabcdR list)`	entspricht `(CaR (CbR (...)))`, auch weniger als 4
`(LAST list)`	Liefert letztes Element der Liste
`(LENGTH list)`	Liefert die Länge der Liste
`(MEMBER expr list)`	Liefert Liste ab (einschl.) expr oder NIL
`(MEMBER e l: Test 'EQUAL)`	Erzwingt EQUAL für den Vergleichstest, Standard: EQ
`(op expr ...)`	Grundrechenarten (op ist +, -, * oder /)
`(+ expr) (- expr)`	Vorzeichenwahl
`(EXPT base exp)`	Exponentialfunktion für beliebige Basen
`(MAX expr ...) (MIN expr ...)`	Maximum bzw. Minimum
`(GCD expr ...)`	Größter gemeinsamer Teiler

Prädikate

`(NULL expr)`	Liefert T, wenn expr den Wert NIL hat
`(SYMBOLP expr)`	Liefert T, wenn expr ein Symbol (Variable) ist
`(ATOM expr)`	Liefert T, wenn expr ein Atom ist
`(NUMBERP expr)`	Liefert T, wenn expr einen numerischen Wert hat
`(STRINGP expr)`	Liefert T, wenn expr eine Zeichenkette ist
`(LISTP expr)`	Liefert T, wenn expr eine Liste oder NIL ist
`(CONSP expr)`	Liefert T, wenn expr eine Liste ist
`(= expr ...) (\= )`	T, wenn Ausdrücke paarweise gleich/ungleich sind
`(< expr ...) (<= ) (> ) (>= )`	T, wenn die Ausdrücke richtig geordnet sind
`(EQUAL expr expr)`	Liefert T, wenn beide Ausdrücke gleich sind
`(EQ expr expr)`	Liefert T bei gleicher Speicherrepräsentation
`(AND expr ...) (OR ) (NOT expr)`	Logische Funktionen, Kurzschlußauswertung
`(COND (expr fkt) ...)`	If-Anweisung; Wert: alle expr=NIL: NIL, sonst fkt; keine fkt angegeben: expr.

Funktionen

`(DEFUN fktname paramlist fkt)`	Definiert eine Funktion
`(fktname param1 ...)`	Funktionsaufruf
`(SETQ fktsymb fktname)`	Verbindet ein Symbol mit einer Funktion
`(FUNCALL fktsymb param ...)`	Ruft die entsprechende Funktion auf
`(APPLY fktsymb param/list ...)`	FUNCALL, Listenelemente statt der Listen
`(SYMBOL-FUNCTION fktsymb)`	Liefert den Funktionsnamen der gelinkten Funktion
`(LAMBDA paramlist fkt)`	Führt fkt mit angegebenen Parametern aus
`(LET ((par expr) ...) fkt)`	LAMBDA mit vorheriger Parameterbindung
`(FUNCTION lambdaexpr)`	Führt eine Lambda-Funktion aus