Lisp

Lisp steht für "LISt Processing" und ist nach Fortran die zweitälteste, heute noch verwendete Computersprache der Welt. Die ersten Versionen stammen aus dem Jahre 1960. Trotzdem erfreut sich Lisp heute immer noch großer Beliebtheit, insbesondere im Bereich künstlicher Intelligenz. Doch nicht nur da. So wurden dem CAD-Paket AutoCAD und dem UNIX-Editor Emacs Lisp als Makrosprache mitgegeben.

Der erfolg von Lisp liegt in der einfachen Syntax ind im Interpreter, der es erlaubt, schnelle Prototypen, die später verfeinert werden, zu erstellen. Die Entwicklung durh John McCarthy begann 1956 mit den ersten Überlegungen zur Sprache Lisp. Zwei Jahre darauf startete er die Entwicklung von Lisp 1, gefolgt von Lisp 1.5, dem Lisp der 60er Jahre. Zum Ende der 60er folgten zwei wichtige Implementationen: Maclisp,welche mehr Wert auf Kompaktheit und Effizienz legte, ha und Interlisp, bei der die Benutzerfreundlichkeit im Mittelpunkt stand.

1975 begann die Entwicklung von Scheme, das heute zur Erforschung von Programmierkonzepten und zu Schulung von Programmierkenntnissen eingesetz wird. In den 80ern war die Blütezeit der Lispmaschinen bei denen die effektive Abarbeitung von Lisp-Befehlen im Vordergrund stand. Das Lisp dieser Maschinen enthielt assemblerartige Grundoperatinonen, so daß dasBetriebssystem in Lisp geschrieben werden konnte.

Wichtige Hersteller: Symbolics, Xerox (Dandelion), TI (Explorer). 1984 wurde durch Guy Steele Jr. In seinem Werk ?Common Lisp the Language 1st Ed. (Cltl1)? Lisp zum ersten mal als Standart beschrieben. Danach folgten verschiden freie und kommerzielle Implementationen von Common Lisp. Die letzte Aktualisierung von Common Lisp geschah 1990 wieder durch Guy Steele Jr. mit ? Common Lisp the Language 2nd Edition.?.

Entwicklung von Lisp

1956 John McCarthy stellt erste Überlegungen zur Sprache Lisp an
1958 McCarthy (MIT) formuliert die Eigenschaften von LISP
1959 McCarthy schafft die theoretische Grundlage (?Recursive Functions of Symbolic Expressions and their Computation by Machine?)
1960/62 LISP/LISP 1.5 (MIT), lge ein Quasistandard
1966 BBN LISP (Bolt,Beranek & Newman),MAC LISP (Projeckt Mac am MIT);
1973 BBN LISP in InterLISP-10 umbenant
1975 Beginnt die Entwicklung von Scheme, das heute zur Erforschung von Programmierkonzepten und in der Lehre eingesetzt wird
1977 LISP-Maschinen-Entwicklung beginnt am Mit
1978 NIL für kommerzielle timesharing-Systeme
1980 ZetalLISP (Symbolics),LM LISP (LMI),InterLISP-D (Xerox), SPICE LISP (CMU)
1981 LISP-Maschinen kommen auf den Markt,LeLISP

1982 S1 LISP für Supercomputer mit MultiProcessing (CMU,MIT,Lawrence Livermore Lab) CommonLISP (viele Firmen und Institutionen)
1984 In Guy Steele Jr: Common Lisp the Language 1st Ed. (CltL1) wird zum erstenmal ein Sprachstandard beschrieben. In der Folgezeit gibt es verschiedene kommerzielle und freie Implementationen von Common Lisp.

??
Anfang 90er: EuLISP,ISO LISP AutoLISP für AutoCAD-Systeme.....

Wie der Name schon andeutet, ist Lisp auf Listenverarbeitung spezialisiert.

Anders gesagt: Lisp verarbeitet eiist nach Fortran die zweitälteste, heute noch verwendete Computersprache der Welt. Die ersten Versionen stammen aus dem Jahregentlich nur Listen.

Beispiele für Listen in Lisp:

```
( 1 2 3 4 ) ; Liste mit Elementen 1, 2, 3, 4
```

(Name 2) ; Liste Student, Name, 2
(Name 2 (/ 2 3)) ; Liste Name, 2, Unterliste /, 2, 3
Name, 2 ... Symbole
Beachte, daß Name nicht in Anführungszeichen steht, ist also kein String!
In Listen können Elemente verschiedenen Typs vorkommen:

Lisp ist eine ungetypte Sprache!
(In Lisp haben Objekte einen Typ, Variablen aber nicht. Also kein 'int x;'

Formen in Lisp

LispProgramme bestehen aus Formen, die ausgewertet werden.
Form: Entweder ein Atom oder eine Liste.
Atom:
Zahl: 123
Variable: x, 4h//34+, ...
Konstante: (defconst konstante 4)
Liste:
(list 'Listenelement2'Elem3'x)
(+ 4 5)
Auswertung von Listen:
Das erste Element wird als Funktion gewertet und auf die anderen Elemente angewendet.

```
> (+ 4 5)                 ; Numerische Operationen in PräfixNotation
9
>(list 'x 3 'Name)        ; Bildet Liste mit Elementen x, 3, Name
(x 3 Name)
```

Listen in Lisp

Listen:
(element1 element2 element3 ...)
Listenelemente werden durch ' ' getrennt.
Listenkonstruktoren:
(cons h r): Bildet eine Liste mit Kopf h und Rest r
(cons 1 nil) -> (1) ; nil steht für leere Liste
(cons 1 (cons 2 nil)) -> (1 2)
(cons 1 '(2)) -> (1 2)
': quote Verhindert, daß Lisp versucht, '2' als Funktion auszuwerten.
(cons 1 (quote (2))) äquivalent zu (cons 1 '(2))
Listen zerlegen:
(first '(1 2 3)) -> 1 ; liefert das erste Element einer Liste
(rest '(1 2 3)) -> (2 3) ; liefert den 'Rest' einer Liste
(Man beachte wieder die quotes)
(rest (list 1 2 3)) -> (2 3) ; liefert den 'Rest' einer Liste

Einfache Sprachelemente (Spezielle Formen)

ist nach Fortran die zweitälteste, heute noch verwendete Computersprache der Welt. Die ersten Versionen stammen aus dem Jahre
Variablenzuweisung:
(setq x wert) Weist x den Wert 'wert' zu
Dabei wird das erste Argument 'x' nicht ausgewertet
1)Lisp in der

```
(setq x 123) ; Weist x die Zahl 123 zu
(setq x '(1 2 3)) ; Weist x die Liste ( 1 2 3 ) zu
```

Bedingte Verzweigung:
(if test w1 w2)
Wertet zu w1 aus, wenn 'test' erfüllt ist, sonst zu w2.
(if (= 1 1) (+ 3 4) "String")

Conditional:
(cond (t1 v1) (t2 v2) ...)
Wertet zu vi aus, wenn Test ti erfüllt ist. (Erster erfüllter Test)
(cond ((= 1 1) "String") ((= 2 3) Symbol) (t 123))

Funktionen

Evaluation
Die Auswertung eines symbolischen Ausdrucks im Interpreter erfolgt nach 3 Regeln:
1.Identität
2 .Listen
3.Sybole

1.Identität:
Eine Zahl ,eine Zeichenfolge, oder die Symbole t und nil evaluiern zu sich selbst.

? 11.5 --->11.5
?"String"--->"String"
? t--->T
?nil ---->NIL

2.Listen
Jede Liste stellt einen Funktionsaufruf dar.
-Erstes Element:Funktor (Funktiosname)
-Alle weiteren Elemente :Argumente
-Die Argumente werden vor dem Funktionsaufruf evaluiert
-Dei leere Liste () hat den Wert NIL

3.Symbole
Die Auswertung eines Symbols liefert dan mit dem Symbol assoziierten Wer zurück.
-Ausnahme:t und nil.

Funktionsdefinition:
(defun funktion (p1 p2 ...) (Funktionsrumpf))
Definiert Funktion 'funktion' mit Parametern p1 p2 ....

?Hello World? :
> (write ?Hello World!?)
Hello World!

Fakultätsfunktion:
> (defun factorial (n)
(cond ((= n 0) 1)

```
(t (* n (factorial (- n 1))))))
```

FACTORIAL

Addieren der Elemente einer Liste:

```
(defun add (l) (cond ( (eq l () ) 0)
( t (+ (first l) (add (rest l))))))
```

Eingabe:

(read)	Liest Eingabe von der Standardeingabe
(setq x (read))	Liest eine Eingabe und weist sie der Variabel x zu

Ausgabe:

```
(write ?Hello World!?)
```

Besonderheiten:
'read' liefert einen LispAusdruck zurück, keine Zeichenkette.

```
> (read)
Dies ist eine Zeichenkette!
*** - EVAL: variable IST has no value
```

Nützlich in Verbindung mit 'eval':
(eval w) wertet den Ausdruck 'w' aus.
(eval '(setq x 10)) weist x den Wert 10 zu
Betrachte:

```
> (eval (read))
(setq x 100)
```

Liest Eingabe und wertet sie aus, d.h. weist x den Wert 100 zu.
Man spart sich also den Parser!

Ein Beispiel

Datenbank:

```
(setq datenbank '((father Homer Bart)
    (father Homer Lisa)
    (son Bart Homer)))
```

Programm:

```
> (eval (read))
(cons '(daughter Lisa Homer) datenbank)
```

Fügt den Eintrag '(daugther Lisa Homer)' der Datenbank hinzu.
Werden jetzt noch Funktionen wie 'remove' und 'lookup' definiert, entsteht in wenigen Zeilen ein Programm zur Verwaltung einer einfachen Abstammungsdatenbank!

Man vergleiche dies mit einem äquivalenten Program in C oder Java!

Bisher liest das Programm allerdings nur eine einzelne Eingabe.
Es fehlt ein Wiederholungsmechanismus!

Schleifen

LoopSchleife:
(loop (body)) Wiederholt unendlich oft den Schleifenrumpf '(body)'
Abbruch der Schleife:
(return wert) bricht die Schleife mit Wert 'wert' ab.
Weitere Schleifen:
(dotimes (n Zahl) (body1) (body2) ...) Äquivalent einer 'for'Schleife

```
> (dotimes (n 11) (print n) (print (* n n)))
0 0
```

1 1
2 4
3 9
4 16
...
Es gibt noch andere Schleifenkonstrukte.

Ein Beispiel

Datenbank:

```
(setq datenbank '((father Homer Bart)
     (father Homer Lisa)
     (son Bart Homer)))
```

Programm:

```
> (loop (eval (read)))
(cons '(daughter Lisa Homer) datenbank)
```

Fügt den Eintrag '(daugther Lisa Homer)' der Datenbank hinzu.

Mit der Schleife ist sind nun beliebige viele Datenbankoperationen möglich.

Die LispTopLoop:

```
(loop (write (eval (read))))
```

Liest endlos einen LispAusdruck von der Standardeingabe, wertet in aus
und gibt das Ergebnis auf dem Bildschirm aus.

Basistypen in Lisp

Zahlen:
BIGNUM: Ganze Zahlen beliebiger Stelligkeit (beschr. durch Speicher)
RATIO: Rationale Zahlen
(* 3 (/ 1 2)) -> 3/2
FIXNUM: Integers (Größe maschinenabhängig. Entspr. int)
FLOAT: Gleitkommezahlen (maschinenabhängig)

Lisp benutzt Präfixnotation (da in Listen erstes Symbol Funktion sein muß)

Zeichenketten:
STRING: Zeichenketten
CHAR: Einzelne Zeichen (#\a steht für Buchstaben 'a')

- Lisp ist ungetypt, d.h. Variablen haben keinen Typ (also kein BIGNUM x).
-Natürlich kann auch Lisp keine Zahlen zu Wörtern addieren.

```
> (defun f (n) (+ n "Hallo")) ; Def. Funktion die n zu ?Hallo? add.
F                             ; Lisp akzeptiert Definition.
> (f 3)
*** - +: "Hallo" is not a NUMBER
```

Zusammenfassung

Lisp ist eine kleineprogrammiersprache wie pascal,fortran.Lisp ist nach Fortran die zweitälteste, heute noch verwendete Computersprache der Welt. Um heutzutage leichte programme vom durcheinander zu retten ist Lisp die beste Version.Die neue entwicklung vom Lisp ist Interlisp.Kurz gefasst obwohl nach Lisp mehrere programme entwickelt sind brauchen wir Lisp noch immer.

Quellen

1)Lisp in der Abteilung Intelligente Systeme der Uni Stutgart.
2)Modellierung und Programmierung Ipke Wachsmuth von UNi Bielefeld