Funktionale Sprachen: LISP

- Lisp heute:
  - pure Lisp
  - plus viele vordefinierte Funktionen („leider“ auch mit Seiteneffekten!)
  - plus weitere bedingte Ausdrücke
  - plus Iterationen
  - plus I/O-Funktionen

- → *impure Lisp*

Funktionale Sprachen: LISP

- Beispiele für Funktionen mit Seiteneffekten:
  - (rplaca x y) ;ersetzt in Cons cell x das a-Feld durch y
  - (rplacd x y) ;ersetzt in Cons cell x das d-Feld durch y
  - (setf a wert) ;Variable a erhält Wert wert
  - (setf a 4 b 2) ;a erhält Wert 4, b Wert 2

- Funktionen mit Schlüsselwort-Argumenten:
  - (fill liste wert :start pos1 :end pos2)
    (fill ´(1 2 3 4) 9) ;liefert (9 9 9 9)
    (fill ´(1 2 3 4) 9 :start 1 :end 3) ;liefert (1 9 9 4)

Funktionale Sprachen: LISP

- Funktionen auf Listen:
  - (first liste) ; wie car
  - (rest liste) ; wie cdr
  - (endp liste) ;Test auf leere Liste, liefert T oder NIL
  - (list-length liste) ; liefert Anzahl Elemente
  - (nth pos liste) ; liefert Element an Position pos
  - (last liste) ; liefert Liste mit letztem Listenelement
  - (list a1 … an) ; bildet Liste mit den Elementen a1, …, an
  - (append liste1 liste2) ;verbindet Listen zu einer
  - (mapcar function liste1 … liste_n) ;wendet function auf alle Elemente von liste1 bis liste_n an

    (mapcar #´numberp ´(a 1 b 2)) → (NIL T NIL T)

    (mapcar #´+ ´(1 3 5) ´(10 20 30)) → (11 23 35)

Funktionale Sprachen: LISP

- Welche Ergebnisse liefern folgende Ausdrücke?
  - (cons 5 (list 6 7))
  - (cons 5 ´(list 6 7))
  - (list 3 ´from 9 ´gives (- 9 3))
  - (+ (length ´(1 foo 2 moo)) (third ´(1 foo 2 moo)))
  - (rest ´(cons is short for construct))

- Folgende Ausdrücke liefern Fehler. Welche? Was ist falsch?
  - (third (the quick brown fox))
  - (list 2 and 2 is 4)
  - (+ 1 ´(length (list t t t t)))
  - (cons ´harry (ron hermine))
  - (cons ´harry (list ron hermine))

Funktionale Sprachen: LISP

- Bedingte Ausdrücke: neben (cond …..) gibt es
  - (if test thenteil [elseteil]) ; ist test nicht NIL, wird thenteil ausgeführt, sonst elseteil
  - (when test ausdruck1 ausdruck2 …) ;ist test nicht NIL, alle nachfolgenden Ausdrücke ausgeführen
  - (unless test ausdruck1 ausdruck2 …) ; wie when, nur hier muss test NIL liefern

Funktionale Sprachen: LISP

- Iteration:
  - (loop ausdruck1 ausduck2 …) ; führt wiederholt ausdruck1,ausdruck2,… aus

    bis Befehl (return wert) ausgeführt wird
  - (do ((var1 wert1)

    …

    (var_n wert_n))

    (end-test resultat)

    ausdruck1

    …

    ausdruck_m)

    ausdruck1,..,ausdruck_m werden solange ausgeführt, bis end-test ungleich NIL ist; Ergebnis ist Wert von resultat

Funktionale Sprachen: LISP

– (dolist (x liste result)
  ausdruck1
  …
  ausdruck_m)

x werden nacheinander alle Werten aus liste zugewiesen, immer wird ausdruck1 bis ausdruck_m ausgeführt; das Ergebnis ist result

– (dotimes (var n result)
  ausdruck1
  …
  ausdruck_m)

var wird nacheinander der Wert 0, 1, …, n-1 zugewiesen, jedes mal wird ausdruck1 bis ausdruck_m ausgeführt; Ergebnis ist result

Funktionale Sprachen: LISP

- Lokale Variablen:

```
(let  ( (var1  wert1)
                …
        (var_n   wert_n))
   ausdruck1
   …
   ausdruck_m)
```

var1 bis var_n stehen nur für ausdruck1 bis ausdruck_m zur Verfügung

Lokale Variablen in der Deklaration weiterer benutzen: let*