Programmbeispiele in J

Im Folgenden die Aufzeichnung einer konkreten J-Sitzung: Am linken Rand steht die Antwort des Systems.

```
NB. --- Namen alphabetisch sortieren ---------------------------------

   words  =. ;:             NB. Woerter bilden / funktioanle Zuweisung
   sort   =. /: ~           NB. Sortieren
   under  =. &.             NB. Konjunktion: f&.g <=> (inv g) f g
   bywords=. under words    NB. etwas Wort-weise tun / adverb

   text=. 'Heinz Xenia Kurt Leopold'   NB. Liste mit Buchstaben

   sort bywords text        NB. sortiere Worte im Text
Heinz Kurt Leopold Xenia


   reverse=. |.             NB. umdrehen
   reverse bywords text     NB. Reihenfolge der Worte im Text umdrehen
Leopold Kurt Xenia Heinz
   text                     NB. urspruenglicher Text zum Vergleich
Heinz Xenia Kurt Leopold
   reverse text             NB. Text umdrehen
dlopoeL truK aineX znieH
```

```
NB. --------- Das Arithmetische Mittel ---------------------------------

   +/ 1 2 3 4 5           NB. Summe ueber Liste
15
                          NB. das entspricht der Anweisung:
  1+2+3+4+5
15

  */1 2 3 4 5             NB. Produkt ueber Liste, %/ , >./ Maximum
120

   sum=. +/               NB. funktionale Zuweisung
   sum 1 2 3 4 5          NB. +/ entspricht dem Summenzeichen Sigma
15
   count=. #              NB. Anzahl der Elemente in einem Array
   mean =. sum % count    NB. Das arithmetische Mittel
   mean 1 2 3 4 5         NB. ! das war soeben FUNKTIONALES PROGRAMMIEREN
3                         NB. ! = das Kombinieren von Funktionen (Verben)
                          NB. ! zu einem neuen Verb

   (sum 1 2 3 4 5) % (# 1 2 3 4 5)
3
   under=. &.             NB. Konjunktion zum Verbinden von Verben:
                          NB.      g &. f <=> (f invers) g f
   hmean=. mean under %   NB. Das Harmonische Mittel:
                          NB.      Der Kehrwert des Mittels der Kehrwerte

   hmean 1 2 3 4 5
2.18978

   % mean % 1 2 3 4 5
2.18978
```

```
NB. Gruppiere Verkaufszahlen nach Artikelnummer

   items=.  1  2  2  3  2  3  1  1  2
   sales=. 20 25 30 30 15 20 10 15 15
   box  =. <   [.   sum=. +/
   items box/. sales
+--------+-----------+-----+
|20 10 15|25 30 15 15|30 20|
+--------+-----------+-----+
   items sum/. sales
45 85 50
```

```
NB. ---------- hexadezimal Rechnen ---------------------------------

   h2d=. (16&#.)@('0123456789ABCDEF'&i.)  NB. hexadezimal nach dezimal
   under=. &.                             NB. f&.g <=> (inv g) f g
   HEX=. under h2d                        NB. HEX adverb

   'A' + HEX '1'
B
   'B' - HEX '1'
A
   'A' * HEX '5'
32
   'A' % HEX '5'
2
```

```
NB.  Fakultaet   1.) Schleifenprogramm
NB.               2.) Direkt Definiert
NB.                3.) Basisfunktion !

factorial=. 3 : 0             NB. Als Schleifen Programm
a=. 1                         NB. y. ist das rechte Argument
 while. y. > 1                NB.        des Verbs factorial
  do. a =. a  * y.
      y.=. y. - 1
 end.
a                             NB. a ist das Ergebnis
)


fac=. 1:`(] * fac@<:)  @. *   NB. direkt und rekursiv definiert

   factorial 9
362880
   fac 9
362880
   !9
362880
```

```
NB. --- Ueberfluessige Leerzeichen aus einem Text loeschen - 2 Versionen

lrb  =. (' '&,)@(,&' ')       NB. left right blank
fdb  =. '  '&E.               NB. find double blanks
DEB =. (#~ -.@fdb)&.lrb       NB. Delete Extraneous Blanks

   DEB '      The     Joy     of    J      '
The Joy of J
```

```
NB. ---  Das gleiche im Schleifenstil - zur Abschreckung ! --------------

DEB_LOOP=. 3 : 0
NB. Delete Extraneous Blanks - Looping Version
and=. *.  [. not=. -.      NB. functional assignment of logical and
blk=. ' ' [  res=. ''      NB. blank / result - empty
inx=. 0                    NB. index
nbr=. (# y.)-2             NB. number of loops

 while. inx <: nbr do.                    NB. compare adjacent elements
  if. not (blk=inx{y.) and (blk=(inx+1){y.)NB. if both are not blank then
    do. res=. res, inx { y.       NB. the first is part of the result
   end.
  inx=. inx + 1                   NB. counting index
 end.                             NB.

 if. 0 ~: # res do.               NB. check if result is empty

  if.   blk = {. res              NB. if first element is blank
   do.  res=. }. res              NB. delete it
  end.

  if.  blk = {: res               NB. if last element is blank
   do. res=. }: res               NB. delete it
  end.
 end.
res                               NB. result
)
```

Wie verschiedene Programmierer ihre Fahrräder bauen:

(Nachtrag zu PCNEWS edit-49, Seite 56)

J'uggler bauen etwas rundes und rollen damit überall hin.