Beschreibung der Operationen + * - : be eb.... im Detail

Stand: 12.08.2022

Hinter dem + Symbol verbirgt sich die größte Funktionalitätsvielfalt der obigen Operationen. Man kann damit nicht nur verschiedene Zahlentypen addieren, sondern auch zu einer Liste, einem Tupel oder einem beliebigen Tabment etwas hinzufügen. Ferner kann man Tupel gleicher Länge komponentenweise addieren sowie Texte verketten. Durch unsere 4 Zahlendarstellungen ZAHL, PZAHL, RATIO, BARl kann man vielfältigere Probleme lösen als mit Sprachen, die nur `int` und `float` besitzen. Allerdings wird es dadurch auch etwas komplizierter. Obige Operationen haben im Wesentlichen die Eigenschaft, dass sich das Schema und sogar der Tabmenttyp (TT9 der ersten Inputtabelle nicht ändern. Wir wollen diese Eigenschaft **TT-Invarianz** nennen. Daraus folgt beispielsweise, dass das Ergebnis von

```
1/3 +2.1      (73/30)
```

eine rationale Zahl (RATIO), aber

```
2.1 + 1/3     (2.43333333333)
```

eine PZAHL ist. Damit kann der Nutzer leicht steuern, welche Addition er wünscht. Nachfolgende Inputwerte werden vor Anwendung stets in den Typ des ersten Inputwertes transferiert. Um bestimmten Gewohnheiten des Nutzers nicht zu stark entgegen zu wirken, vertreten wir die TTD-Invarianz nicht 100-prozentig.

```
1+1.3
```

müsste demnach eine ganze Zahl und damit

```
2
```

ergeben. Wenn der erwachsene Nutzer, das in der Regel gewünschte Ergebnis wollte, hätte er bei einer konsequenten Realisierung obiger Invarianz

```
1. + 1.3
```

schreiben müssen. Damit könnten wir ihn vielleicht verärgern, da man so einen Punkt immer wieder vergessen könnte. Das wollen wir aber nicht, weshalb o++o bei `1+1.3` doch `2.3` berechnet.

Zahlenadditionen

ZAHL: Addition von big_int: 1+2=3
PZAHL: Addition von float: 1.2 + 2.1 = 3.3
RATIO: Addition rationaler Zahlen: 1/3 + 1/4 = 7/12

gemischte Zahlenadditionen:

ZAHL+ PZAHL = PZAHL+ZAHL = PZAHL
ZAHL + RATIO = RATIO + ZAHL = RATIO
PZAHL+ RATIO = PZAHL
RATIO + PZAHL = RATIO

Textverkettungen:

Auto + mobil = Automobil

gemischte Textverkettungen

WORT + TEXT = WORT
TEXT + WORT = TEXT

sonst bleibt der erste Inputwert stets unverändert:

damit gilt insbesondere:
Text + Zahl = Text
Zahl + Text = Zahl
```
Hallo + 1    = Hallo
```
aber
```
Hallo + "1" = Hallo1
```

1 Verarbeitung von Zahlen mit + - * : eb und be

o++o besitzt die elementaren Zahltypen ZAHL, PZAHL und RATIO sowie Listen von Strichen (BARl) für Schüler und Kindergartenkinder.
Eine Zahl aus **ZAHL** ist ganzzahlig und hat keine Einschränkung bzgl. der Größe. Wir stellen diesen Typ big_int auch für Anroid bereit, obwohl JavaScript diesen nicht besitzt.
Sehr große natürliche Zahlen sind kaum erfassbar. Daher können wir sie in 3er Blöcke einteilen. Ein Beispiel:
```
1'234'567
```
oder in Exponent-Mantisseform die Größenordnung noch größere Zahlen ebenfalls noch leicht zu erfassen:
27m`1.234'567'890'123`
oder in Mantisse-Exponentform
```
1.234'567'890'123e27
```

Eine (Punkt-)Zahl aus **PZAHL** besitzt doppelte Genauigkeit. Sie enthält eine Folge von bis 64 Dualziffern. In o++o tritt sie in verschiedenen Repräsentationen auf:
```
1'234'567'890.12
```
oder ein anderes Beispiel:
`9m12.34567890123` (Der Exponent 9 entspricht Milliarden) oder
```
12.34567890123e9
```
Zähler und Nenner einer rationalen Zahl aus **RATIO** können ebenfalls beliebig große Zahlen sein.
Beispiel:
```
123456789/57
```
Keine der Zahlen dieser 3 Typen darf in irgend einer Repräsentation ein Leerzeichen enthalten.

Die 6 obigen Operationen sind in o++o wesentlich allgemeiner als in anderen Programmiersystemen. Man kann nämlich beispielsweise eine Zahl nicht nur zu einer anderen Zahl sondern auch zu einem beliebigen Tabment hinzufügen. Das entstehende Tabment besitzt die gleiche Struktur (den gleichen Typ) (den gleichen TT =TabmenTyp) wie der erste der beiden Inputwerte.

2 Die + Operation

Beispiele:
Textverkettungen:
Auto + mobil = Automobil

Addition von Tabment und Einzelwert
```
<TAB!
X,Yl m
```

```
1 2
  3
4 5
!TAB>
+2
```
ergibt:
```
X, Yl m
3  4
   5
6  7

<TAB!
X, Yl m
1  2
   3
Ha 5
!TAB>
+ llo
```
ergibt:
```
X,    Yl  m
1     2
      3
Hallo 5
```

Addition von Tabelle und elementarem Tupel
```
<TAB!
X, Yl m
1  2
   3
Ha 5
!TAB>
+ (llo,3)
```
ergibt:
```
X,    Yl  m
1     5
      6
Hallo 8
```

Man erkennt, dass der erste Wert des Tupels zu jedem Wert der ersten Spalte und der zweite Wert zu jedem Wert der zweiten Spalte hinzugefügt wird.

Addition von Tabellen gleicher Struktur:
```
<TAB!
X, Yl m
1  2
   3
Ha 5
!TAB>
+ <TAB!
X, Yl m
1  2
```

```
   3
Ha 5
!TAB>
```
ergibt:
```
X,   Yl  m
2     4
      6
HaHa 10
```

Hierbei wird zu jedem Wert der i-ten Spalte und der j-ten Zeile der Wert der zweiten Tabelle der i-ten Spalte und j-ten Zeile addiert. Wenn an einer Stelle verschiedene Zeilen- oder Spaltenanzahlen existieren, ist die + Operation nicht definiert.

```
1,2+3
```
ergibt:
```
4,5
```
bzw. in web-Darstellung:

ZAHL	ZAHL
4	5

und in tab-Repräsentation:
```
ZAHL,ZAHL
   4    5
```
D.h., das Ergebnis ist ein Paar von 2 Zahlen.
Sogar für elementare Inputdaten ergeben sich aus diesem Prinzip interessante Aspekte:
```
SUMME1:=1/6+1.4
SUMME2:=1.4+1/6
```
ergibt in web-Darstellung:

SUMME1	SUMME2
47/30	1.56666666667

SUMME1 ist vom Typ RATIO, da 1/6 als rationale Zahl eingegeben wurde. Das ist auch die einzige exakte Repräsentationsmöglichkeit dieser Zahl. Der erste Inputwert von SUMME2 ist eine PZAHL, weshalb auch das Ergebnis ebenfalls eine Punktzahl ist. Offensichtlich sind beide Darstellungen sinnvoll. Die erste gibt das exakte Ergebnis wieder und die zweite eine Zahl, dessen Größenordnung man besser einschätzen kann. Die Operation + ist am vielfältigsten einsetzbar.

3 Die - Operation

Die Operation - unterscheidet sich von der + Operation nur in 2 Punkten. Erstens wird die Addition von Zahlen stets durch die entsprechende Subtraktion von Zahlen ersetzt und zweitens bedeutet das „Subtrahieren“ von Texten, dass der zweite Inputwert in jedem TEXT- oder WORT-Wert eliminiert wird.
```
<TAB!
1  Heute ist Monnntag.
2  Morgen ist Dienstag.
!TAB>
- "nn"
```
ergibt
```
X, Y              l
```

```
1  Heute ist Montag.
2  Morgen ist Dienstag.
```

4 Die * Operation

Gegenüber der + Operation muss man hier nur die Zahlenaddition durch die Zahlenmultiplikation ersetzen und die Textaddition durch die Operation ersetzen, die den ersten Inputwert unverändert lässt. Eine Tabelle vom Typ `A1,A2, ...An l` ist eine Matrix. Haben 2 Matrixtabellen die gleiche Anzahl von Spalten und Zeilen, so stimmt die + Operation mit der Matrixaddition überein. Das gilt nicht für die Multiplikation. Bekanntlich wird bei der Matrixmultiplikation nicht einfach zellenweise multipliziert. Daher stellt o++o eine weitere Operation `*mat` für die Matrixmultiplikation bereit.

5 Die : Operation

Hier gilt das für die * Operation Gesagte. Ferner bewirkt eine einzige Division durch Null nicht die Ungültigkeit der Gesamtoperation. Das Ergebnis enthält dann einen leeren optionalen Wert.

```
<TAB!
X, Y l
1  Heute ist Sonntag.
2  Morgen ist Montag.
!TAB>
Z:= 3 : (X-2)
```

ergibt:

```
X, Y,                 Z?  l
1  Heute ist Sonntag. -3.
2  Morgen ist Montag.
```

Die Division ist relativ allgemein.

```
Xl:= 3 .. 9
Y:= X : 4
```

ergibt

```
X,Y l
3 0.75
4 1.
5 1.25
6 1.5
7 1.75
8 2.
```

9 2.25

Mit diesem Ergebnis werden viele jüngere Schüler nicht zufrieden sein. Aber auch sie können häufig schon den Rest der Division ermitteln. o++o kann den Rest allein ermitteln bzw. das ganzzahlige Divisionsergebnis und den Rest.

```
Xl:= 3 .. 9
Y,REST:= X /rest 4
```

Ergebnis:

X	Y	REST
3	0	3

4	1	0
5	1	1
6	1	2
7	1	3
8	2	0
9	2	1

6 Textverarbeitung mit +, - und -+

Das + Symbol erlaubt auch das Verbinden und Manipulieren von Texten. Dabei wird wieder zwischen TEXT und WORT unterschieden. Beispiel:

```
WORTERGEBNIS:=otto + " o++o"
TEXTERGEBNIS:=otto text + " o++o"
```

Ergebnis:

WORTERGEBNIS	TEXTERGEBNIS
otto_o++o	otto o++o

Da der erste Inputwert von WORTERGEBNIS ein Wort ist, ist auch das Ergebnis vom Typ WORT. Analoges gilt im zweiten Fall, wo das Ergebnis ein Text ist. Wörter können keine Leerzeichen enthalten.

-+ ist eine Operation mit 3 Inputwerten.
Der TT des ersten Inputwerts bleibt bestehen. Jedes Vorkommen des zweiten Inputwerts wird durch den dritten ersetzt.

```
<TAB!
X, Y l
1  Heute ist Sonntag.
2  Gestern ist Samstag.
!TAB>
-+ "n ist" ! "n war"
```

ergibt:

```
X, Y                  l
1  Heute ist Sonntag.
2  Gestern war Samstag.
```

Für Textmanipulationen geben wir nur noch ein Beispiel an:

```
TEXTPLUS:="Heute ist ein schöner " + Tag
TEXTMINUS:=Donnerwetter - "n"
TEXTMINUSPLUS:="Heute ist ein schoener Tag." -+ oe ! ö
```

Ergebnis:

TEXTPLUS	TEXTMINUS	TEXTMINUSPLUS
Heute ist ein schöner Tag	Doerwetter	Heute ist ein schöner Tag.

7 Die be - und eb - Operationen

Potenzieren mit vertauschten Inputwerten

be kürzt bASISeXPONENT ab. D.h. man schreibt, wie beim Potenzieren üblich zunächst die Basis später den Exponenten. Bei unseren infixen Schreibweisen ergibt sich daraus

```
basis be exponent
```

Beispiel:

```
2 be 3
```

ergibt

```
8
```

```
2 be 1/3
```

ergibt

PZAHL
1.25992104989

Bei der Operation eb (ExponentBasis) wird zuerst der Exponent und dann die Basis geschrieben. Das hat den Vorteil, dass man auch Listen von Exponenten als erstes Argument eingeben kann.

```
0 .. 9 eb 10 '3
```

Ergebnis:

ZAHL
1
10
100
1'000
10'000
100'000
1'000'000
10'000'000
100'000'000
1'000'000'000

Die obigen Operationen haben folgende Typ-Eigenschaften:

Operation	Inputtyp1	Inputtyp2	Outputtyp
+ - *	PZAHL	z	PZAHL
	RATIO	z	RATIO
	z	z'	z
	ZAHL	z	z
+	WORT	WORT TEXT	WORT
	TEXT	WORT TEXT	TEXT

:	ZAHL	ZAHL	PZAHL
	RATIO	RATIO	RATIO
	z	z'	PZAHL
/rest	ZAHL	ZAHL	ZAHL*ZAHL
be eb	ZAHL	ZAHL	ZAHL
	x	x'	PZAHL

Die 4 Operationen + * - : haben darüber hinaus noch die Eigenschaft, dass in den nicht genannten Fällen das erste Argument nicht verändert wird.

8 Die gib - Anweisung

Für die Beschreibung von `gib` benutzen wir Tabmente mit den Schemen:

```
t1! s1 = X1,X2,X3,(Y1,Y2 m) m
```

und

```
t2! s2 = X1,X2,X3,X4,(Y1,Y2,Zl,(W1,W2 m) m) m
```

Mit der `gib`-Anweisung können folgende Aufgaben realisiert werden:
Sortierung, Umstrukturierung, Vereinigung, Aggregation, Duplikateelimination, Taggen

1. Sortierung

```
t1
gib X2,X3,X1 m
```

sortiert zuerst nach X2, dann nach X3 und dann nach X1

```
t1
gib X2,Y2m m
```

sortiert die äußere Kollektion nach `X2` und die innere Kollektion nach `Y2`; die innere Kollektion enthält keine Duplikate und die äußere Kollektion enthält jeden `X2` Wert nur einmal.

2. Umstrukturierung

```
t1
gib X2,X3m m
```

Die X-Segmente werden nacheinander zuerst in die `X2`-Ebene und dann in die `X3`-Ebene eingefügt. Dadurch enthält die äußere Menge jeden `X2`-Wert nur einmal und jede innere Menge jeden `X3`-Wert nur einmal. Die Y-Segmente bleiben bei dieser Umstrukturierung unberührt.

```
t1
gib Y2,X1m m
```

Da die `X`-Segmente `Y2` nicht enthalten, können sie nicht in die Zielstruktur eingefügt werden. Dagegen können die verlängerten `Y`-Segmente (`X1,X2,X3,Y1,Y2`) in beide Ebenen eingefügt werden. Die untergeordneten Y2-Werte werden somit den X1-Werte übergeordnet.

```
t2
gib Y2,Zm,W2m m
```

Da die `X`-Segmente nicht eingefügt werden können, erfolgt dieses erst durch die verlängerten `Y`-Segmente (`X1,X2,X3,Y1,Y2).` Für jedes solche `Y`-Segment werden `Z`-Elemente und anschließend die `W`-Segmente eingefügt. Keine Kollektion des Resultats enthält Duplikate.

```
t2
gib Y2,(Z,W2 m) m
```

Da die verlängerten Z-Segmente W2 nicht enthalten und die verlängerten W-Segmente nicht Z, bleibt jede innere (Z,W2 m) Kollektion leer. Alle Y2-Werte erscheinen aber im Ergebnis; jedoch ohne Duplikate.

3. Vereinigung

```
t1,t2
gib X2b
```

Hier werden zunächst alle X-Segmente von t1 eingefügt und anschließend alle X-Segmente von t2. Es werden hierbei keine Duplikate eliminiert. D.h., die „gewöhnliche“ mengentheoretische Vereinigung ergibt sich dann mit der Anweisung

```
t1,t2
gib X2m.
```

4. Taggen

Manchmal ist es sinnvoll zusätzliche Tags einzufügen oder Tags an völlig neuen Stellen zu verwenden. Z.B., wenn man Daten aus Tabellenkalkulationen bequem weiterverarbeiten will.

```
t: ID,LAND,BREITE,LAENGE,MEERESHOEHE?,JAN,FEB,MRZ,APR,MAI,JUN,JUL,AUG,SEP,OKT,NOV,DEZ l
aus t
gib  ID,LAND,BREITE,LAENGE,MEERESHOEHE?,MONATE l
     MONATE! JAN,FEB,MRZ,APR,MAI,JUN,JUL,AUG,SEP,OKT,NOV,DEZ
```

Hierbei werden die Monate jeweils durch den Tag MONATE eingeschlossen. Durch sie können bestimme Formulierungen verkürzt werden. Z.B.:

```
gib LAND,(ID, MONATE m) m
```

9 keys - effiziente Selektion

keys ist eine Selektionsoperation, d.h., der Typ der Inputtabelle wird durch keys nicht verändert. keys liefert gegenüber avec keine neue Ausdruckskraft. Die Selektion ist in der Regel nur effizienter. Das liegt daran, dass für eine Selektion nicht die gesamte Tabelle durchforstet werden muss. Es wird bei jeder Menge (Liste) direkt auf den gegebenen Schlüssel (das n-te Element) zugegriffen. Da für Selektion die Spaltennamen der gegebenen Werte bedeutungslos sind, reicht es nackte Werte vorzugeben. Weiterhin ist es nicht erforderlich, dass die Kollektionstypen der Schlüsseltabelle mit denen der Ausgangstabelle übereinstimmen.

```
t1
keys [k1 k2 k3]
```

angewandt auf die Tabelle t1 mit dem Schema s1=(X1,X2,X3,X4,(Y1,Y2 m)m) ergibt eine maximal 3-elementige Tabelle vom Typ X1,X2,X3,X4,(Y1,Y2 m)m mit den X1-Werten k1, k2 und k3. Ist X1,X2 der Schlüssel, so wird jeweils ein Element ausgewählt, dass den X1-Wert k1 besitzt. In einem solchen Fall sollte man für jedes Element den gesamten Schlüssel vorgeben:

```
keys <! [k11,k12    k21,k22   k31,k32] !>
```

Bei der komplexeren Tabelle t2 mit s2=(X1,X2,X3,X4,(Y1,Y2,Zl,(W1,W2 m) m) m), kann auch auf innere Elemente direkt zugegriffen werden. Bei Listen wird eine natürliche Zahl x vorgegeben und dann wird das x-te Element ausgegeben. Man kann auch auf innere Elemente direkt zugreifen, wenn der Schlüssel der vorangehenden Kollektion nicht gegeben ist:

```
t2
keys <! [kx1,kx2,[ky1,[“*“],[kw1]]] !>
```

Hier wird das Element der s2-Kollektion mit dem Schlüssel kx1,kx2 und dem untergeordneten Schlüssel ky1 selektiert. Das ky1-Element enthält die vollständige Zl-Kollektion, sowie die (W1,W2 m)-Kollektion mit dem (W1,W2)-Element mit dem Schlüssel kw1.

Es wird stets gefordert, dass die Schlüsselstruktur nach `keys` mit dem Anfang der Schlüsselstruktur der gegebenen Tabelle übereinstimmt. Den Werten aus dem 2. Argument von `keys` müssen einfache Felder der betrachteten Tabelle entsprechen. Damit wäre beispielsweise

```
keys 1234,Berlin
```

unzulässig, wenn die gegebene Tabelle den Typ `(STID, ORT?,… m)` besitzt. Die Überspezifizierung des Schlüssels `STID` ist dagegen erlaubt, wenn das Schema `(STID,ORT,… m)` lautet. Das Gleiche trifft auf `keyslike` (siehe unten) zu.

10 keyslike - effiziente Selektion mit unvollständigem Schlüssel

`keyslike` basiert auf `keys` und der `like` Funktion, unterscheidet sich deswegen auch in einigen Punkten von `keys`.

Wird ein Schlüssel, der aus mehr als einem Feld besteht, unvollständig angegeben, so werden alle Tupel bzw. Subtupel ausgegeben, die den Teilschlüssel besitzen.

Beispiel:

gegeben: `tab! K1,K2,X,Y m`

```
tab
keys k1
```

bestimmt **ein** 4-Tupel mit dem Teilschlüssel `K1`.

```
tab
keyslike k1
```

ermittelt alle 4-Tupel mit dem Teilschlüssel `K1`.

Ansonsten können die ausdrucksstärkeren aber ineffizienteren Möglichkeiten von `like` einbezogen werden.

```
tab
keyslike “?enec*“, “Achim“
```

bestimmt alle Personen mit Vornamen Achim und einem Namen, dessen 2,3,4 und 5. Buchstabe das Teilwort `enec` ergibt. Hier wurde `K1=NAME` und `K2=VORNAME` angenommen.

11 onrs für Stücklisten

Die Operation `onrs` wurde eingeführt, um o++o-Nummern für die Lösung von Stücklistenproblemen bereitzustellen. Das gegebene Tabment muss vom Typ

X1,…,Xn, (Y1, … Yk m) m

, wobei X1 und Y1 die Schlüssel der jeweiligen Kollektionen sind. Wir setzen voraus, dass beide Kollektionen Mengen sind. Dadurch haben wir im Arbeitsspeicher direkten Zugriff auf die entsprechenden Tupel bzw. Subtupel. Wir betrachten ein Beispiel:

```
<TAB!
OT,         EIGENSCHAFT,(UT,         ANZ m) m
Buchse      zylindrisch
Felge       glatt
Polo        modern       Rad         4
                         Motor       1
                         Karosse     1
Golf        schnell      Rad         4
                         Karosse     1
                         Klimaanl    1
```

```
                        Motor      1
Kolben      leicht      KolbRing   2
                        Buchse     1
Motor       schwer      Kolben     6
                        Schraube   8
Rad         rund        Schraube   5
                        Reifen     1
                        Felge      1
!TAB>
onrs OTTONR ! Golf
```

Ergebnis:

```
OT,  EIGENSCHAFT, (OTTONR, UT,       ANZ  m) l
Golf schnell       1       Rad      4
                   1.1     Schraube 5
                   1.2     Reifen   1
                   1.3     Felge    1
                   2       Motor    1
                   2.1     Schraube 8
                   2.2     Kolben   6
                   2.2.1   KolbRing 2
                   2.2.2   Buchse   1
                   3       Klimaanl 1
                   4       Karosse  1
```

Man erkennt, dass allen direkten Unterteilen vom Golf eine otto-Nummer, die nur aus einer Zahl besteht, zugeordnet wird. Der Motor ist ein solches Teil. Die direkten Unterteile vom Motor (Schraube und Kolben) erhalten otto-Nummern mit zwei Zahlen. Analog erhalten die direkten Unterteile vom Kolben otto-Nummern mit 3 Zahlen. Für den Golf wird also eine nichtrekursive Menge ohne Redundanz gebildet. Auf diese Menge könnte jetzt die otto-Rekursion angewandt werden, um die Vielfachheit des Enthaltenseins eines Unterteils zu berechnen.
Neben dem Inputtabment benötigt `onrs` noch den Namen der ONR-Spalte (hier OTTONR) und ein oder mehrere Teile (hier nur Golf) für die die ONR-Auflösung vorgenommen werden soll. Demnach ist auch die Programmzeile

```
onrs STRUK_NR ! [Polo Motor]
```

korrekt und sinnvoll.

12 verti - Tupel zu Liste

Mit der Operation `verti` können bestimmte Tupel in Listen von Paaren umgewandelt werden. Das hat den Vorteil, dass man auf diese Listen Listenoperationen, wie z.B. das Selektieren anwenden kann. Dabei werden die Metadaten in Primärdaten überführt und es wird ein neues Schema in das bestehende eingefügt. Die folgende Datei wurde aus einer EXCEL-Tabelle gewonnen.
`klimate_strahlung1.tab` hat das Schema:

```
ID,LAND,BREITE,LAENGE,MEERESHOEHE?,JAN,FEB,MRZ,APR,MAI,JUN,JUL,AUG,SEP,OKT,NOV,DEZ l
```

Sie enthält 17 Spalten. Mittels `verti` kann man die Spaltenzahl in folgender Weise auf 7 reduzieren:

```
aus klimate_strahlung1.tab
verti MON,STRAHLUNG l:= JAN ..DEZ
```

Es entsteht aus dieser flachen Tabelle eine strukturierte, in der die Strahlungen vertikal angeordnet sind und die Monate in einer zusätzlichen Spalte ausgegeben werden:

```
ID,              LAND,     BREITE,    LAENGE,       MEERESHOEHE?, (MON, STRAHLUNG  l) l
BG0001a-Varna    Bulgaria  43.21      27.91          44.           Jan   63.
                                                                   Feb   68.
                                                                   Mrz   81.
                                                                   Apr   87.
                                                                   Mai   88.
                                                                   Jun   81.
                                                                   Jul   86.
                                                                   Aug  100.
                                                                   Sep   95.
                                                                   Okt   88.
                                                                   Nov   66.
                                                                   Dez   59.
BG0002a-Shumen   Bulgaria   43.283    26.933        242.          Jan   59.
                                                                   Feb   68.
                                                                   Mrz   83.
                                                                   Apr   88.
                                                                   Mai   88.
                                                                   Jun   81.
                                                                   Jul   88.
                                                                   . . .
. . .
```

Jetzt sei eine Tabelle `enkel.tabh` gegeben:

```
NAME,   ORT,    GEBDAT, KLASSE?, (HOBBY,      STUNDEN  l),MAl, DEUl  l
Clara   Oehna   12.6.11 4         Reiten      5            1 2  3 1 1 1
                                   Schach      1            1
Claudia Dallgow 14.9.17            Chinesisch  5
                                   Essen       4
Sophia  Dallgow 7.9.13  2          Malen       5            1 2  1 2 1
                                   Radschlagen 4            1 1
                                   Chinesisch  6
```

Dann können diese Noten mit entsprechenden Fächern vertikal angeordnet werden:

```
enkel.tabh
verti FACH,NOTEl l:= MAl ..DEUl
```

ergibt bei tabh-Ausgabe:

```
NAME,   ORT,    GEBDAT, KLASSE?, (HOBBY,      STUNDEN  l),(FACH, NOTEl  l) l
Clara   Oehna   12.6.11 4         Reiten      5            Ma    1 2 1
                                   Schach      1            Deu   3 1 1 1
Claudia Dallgow 14.9.17            Chinesisch  5
                                   Essen       4
Sophia  Dallgow 7.9.13  2          Malen       5            Ma    1 2 1 1
                                   Radschlagen 4            Deu   1 2 1
                                   Chinesisch  6
```

`meinenoten2.tabh`

13 meta - Umkeroperation von verti

Die Umkehroperation zu `verti` ist `meta`. Hier ist als zweiter Inputwert der elementare Tag, dessen Werte Spaltennamen werden sollen, anzugeben.

```
meinenoten2.tabh
meta FACH
```

ergibt die folgende tabh-Tabelle:

```
NAME,    ORT,     KLASSE?, MAl, DEUl  l
Clara    Oehna    4        1 2  3 1 1 1
                           1
Claudia  Dallgow
Sophia   Dallgow  2        1 2  1 2 1
                           1 1
```

Die beiden Fächerlisten für Claudia sind leer.

```
klimate_strahlung1.tab
verti MONAT,STRAHLUNG l :=JAN .. DEZ
meta MONAT
```

ergibt die Ausgangstabelle mit folgender TTD:

```
<META!
TABMENT! KLIMATE_STRAHLUNG1
KLIMATE_STRAHLUNG1!
(ID,LAND,BREITE,LAENGE,MEERESHOEHE?,JAN,FEB,MRZ,APR,MAI,JUN,JUL,AUG,SEP,OKT,NOV,DEZ l)
ID LAND! TEXT
APR AUG BREITE DEZ FEB JAN JUL JUN LAENGE MAI MEERESHOEHE MRZ NOV OKT SEP! PZAHL
!META>
```