# Overview of the Operations of o++o

(09/09/2021)

Most of the known operations have an arity.

For example, the square root only needs one input value, or argument, – this is usually a number. In the *o++o* data model, it can also be a list of numbers. Then the square root of each of the numbers is taken. The list is then considered an input value, although it can contain ten or even ten-thousand numbers. That means, `sqrt` remains unary in this case as well.

In *o++o* syntax the `sqrt` must follow the argument (postfix). This means that no additional brackets are required. It is permissible in *o++o* to write `sqrt([2 4 7])` instead as

```
[2 4 7] sqrt
```

or even

```
2 4 7 sqrt
```

.

The same result is obtained in both cases. You can apply `sqrt` to any tabment.

Another example is addition. The operator `+` is even better known than the root operation. It has arity 2, meaning it requires two input values. Applying the wrong number of arguments will result in a syntactic error and a corresponding error message. In the term

```
3 + 4
```

3 is the first argument and 4 is the second argument. Here, too, a list or another tabment can be used as the first argument. The operation and second argument are then applied to all elements of the list/tabment.

```
1 3 7 +4
```

results in

```
5 7 11
```

Here and in many other operations, the type of the result matches the type of the first input tab. The above result is therefore also a list of numbers. Binary operations are always written between the two input tabs in *o++o*. You can also see it as them being written after the first input tabment like the unary operations. The same applies to many three-digit operations in *o++o*. "!" is used as a separator between the second and third input value.

```
Hadmersleben subtext 4!5
```

has the result, for example

```
mersl
```

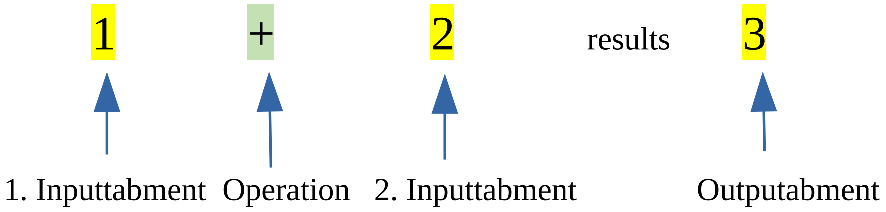
The first input value is “`Hadmersleben`”. The second input value (4) indicates the position of the initial letter of the partial word and the third input value (5) indicates the desired length.

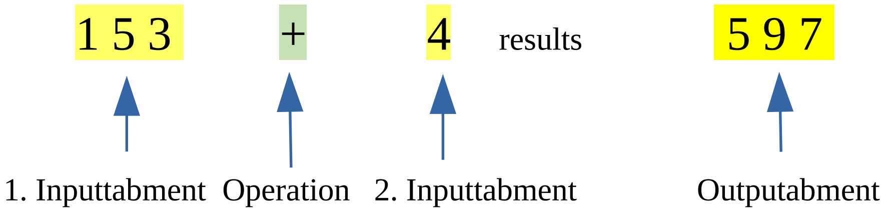
For some operations, familiar spellings are used.

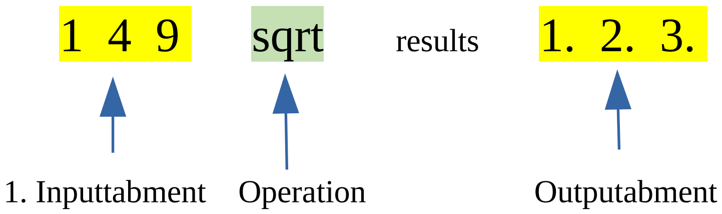
```
 if X>3 then 5 else 6
```

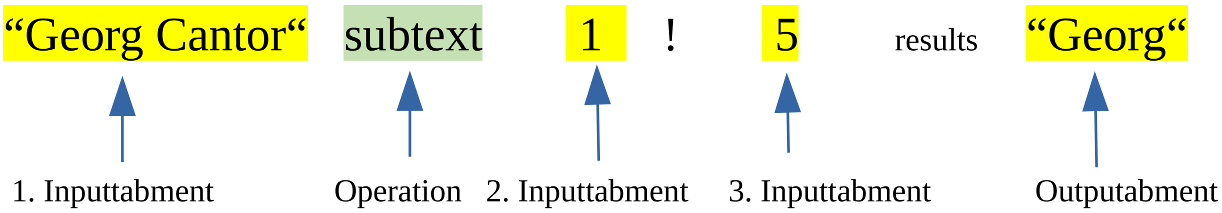
also requires 3 input values (here: a truth value, the 5 and the 6). If we substitute 10 for X, the condition is fulfilled and the if-then-else operation returns 5.

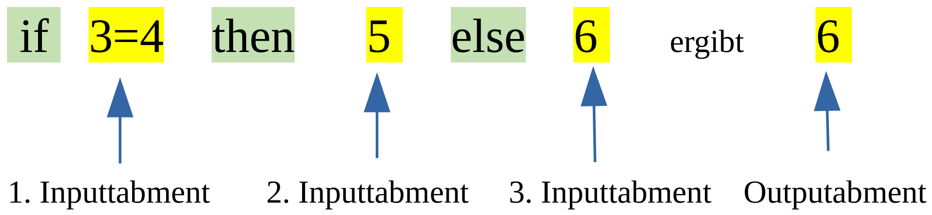
In the following, input and output data are illustrated again using 5 examples.











It should be noted at this point that in many cases the result of the previous line counts as the first input tabment of an operation:

```
marks.tab
++:
```

gives the average of all numbers that appear in the first input `marks.tab`. The program

```
xx.tab
+2
```

adds `2` to each number in table `xx.tab`. `xx.tab` is the first input tabment and `2` is the second. In the same way

```
names.tab
subtext 3! 4
```

extracts a text of length 4 beginning at third position from each textual value (TEXT or WORT) of `names.tab`. Here the ternary subtext operation has the input tabments `names.tab`, 3 and 4.

In the table below we use the following abbreviations.

TT = Tabment Type

TT1 = Tabment Type of the first inputtabment

Zahl = ZAHL or PZAHL or RATIO

Text = TEXT or WORT

| **Operation** | **Arity** | **Output-TT** | **Meaning** | **Examples** |
|---|---|---|---|---|
| `+` | 2 | TT1 | Adding numbers and connecting texts | `1 3 + 2.1 = 3.1 5.1`<br>`xy ab + de = xyde abde` |
| `*` | 2 | TT1 | multiplication | `2 3 5 * 2 = 4 6 10` |
| `-` | 2 | TT1 | subtraction | `3-2=1` |
| `:` | 2 | TT1 | division | `3:4=0.75` |
| `++` | 1 | Zahl | sum | `2 3 6 ++ = 11` |
| `**` | 1 | Zahl | product | `1 3 5 ** = 15` |
| `--` | 1 | Zahl | multiple subtraction | `20 5 4 -- = 11` |

| Operation | Arity | Output-TT | Meaning | Examples |
|---|---|---|---|---|
| :: | 1 | Zahl | multiple division | 64 2 2 :: = 16 |
| ++: | 1 | PZAHL | average | 1 2 3 2 ++: = 2.0 |
| ++1 | 1 | (tuple of) ZAHL | count | 3 4 7 9 ++1 = 4 |
| ++ratio | 1 | RATIO | exact sum | 3 5 5.2 1/3 ++ratio<br>results<br>203/15 |
| **ratio | 1 | RATIO | exact product | 3 5 5.2 1/3 **ratio<br>results<br>26/1 |
| ++text | 1 | TEXT | connect to Text | ab cde fg ++text = abcdefg |
| ++textsep | 2 | TEXT | connect to Text with Separator | ab cde fg ++textsep “;“ = “ab;cde;fg“ |
| +m | 2 | X1,..Xn m | union to flat set | {1 2 3} +m {6 3} = {1 2 3 6} |
| +b | 2 | X1,..Xn b | union to flat bag | {1 2 3} +b {6 3} = {{1 2 3 3 6}} |
| +l | 2 | X1,..Xn l | union to flat list | 3 5 5 +l  6 3 = 3 5 5 6 3 |
| -m | 2 | X1,..Xn m | difference to flat set | { 2 4 5} -m {6 2} = {4 5} |
| -b | 2 | X1,..Xn b | difference to flat bag | {{1 2 4 4 4}} -b {{2 4}}<br>= {{1 4 4}} |
| -l | 2 | X1,..Xn l | difference to flat list | 1 3 4  -l  5 3 = 1 4 |
| :m | 2 | X1,..Xn m | intersection to flat set | {1 3 7} :m {3 9} = {3} |
| :b | 2 | X1,..Xn b | intersection to flat bag | {{2 2 4 }} :b {{2 7 7}} = {{2}} |
| :l | 2 | X1,..Xn l | intersection to flat list | 2 2 4  :l  3 2 4 = 2 4 |
| *m | 2 | X1,..Xn m | Cartesian product | {1 2} *m {3 4 6}<br>= {1,3  1,4 1,6 2,3 2,4 2,6} |
| *b | 2 | X1,..Xn b | Cartesian product to bag | {{1 2}} *b  {{ 2 2 }}<br>= {{1,2 1,2 2,2 2,2}} |
| *l | 2 | X1,..Xn l | Cartesian product to list | 1 2 *l  2 3 = 1,2 1,2 2,2 2,2 |
| , | 2 | TT1,TT2 | pair formation | 1 2,3<br>results<br>ZAHLl,ZAHL<br>1 2   3 |

| Operation | Arity | Output-TT | Meaning | Examples |
|---|---|---|---|---|
| = | 2 | BOOL | equality | 1 = 2<br>results<br>si |
| := [at] | 1 | TT1+ new column | assignment | X:=1<br>Y:= X+2<br>results<br>X,Y<br>1 3 |
| := first next at | 1 | TT1+ new column | recursive assignment | AMOUNT:=first 100<br>next AMOUNT pred*<br>1.03 at YEAR |
| := firstonr nextonr [at] | 1 | TT1 +new column | recursive assignment with otto numbers | CNT2:=firstonr CNT<br>nextonr CNT2 pred<br>*CNT leftat CNT |
| ::= | 1 | TT1 | overwrite | X::= X+3*Y |
| =: $Name | 1 | TT1 | assignment for a variable | 2,3<br>=: $X |
| & | 2 | TT1 | conjunction (and) | 1=1 & 2=3 =no |
| \| | 2 | TT1 | disjunction (or) | 1=1 \| 2=3 = si |
| -> | 2 | TT1 | logical implication | si → si = si |
| <-> | 2 | TT1 | logical equivalence | si <-> no = no |
| < | 2 | BOOL | smaller | 3<4 = si |
| > | 2 | BOOL | greater | 3>4 = no |
| <= | 2 | BOOL | smaller or equal | 2 <= 2 results si |
| >= | 2 | BOOL | greater or equal | 2 >= 4 results no |
| *mat | 2 | X1, ..Xn l | matrix-multiplication | (1,2) *mat [2 3] = 8 |
| -1mat | 1 | TT1 | inverse matrix | <TAB!<br>X1,X2,X3 l<br>1 0 2<br>0 2 0<br>0 0 8<br>!TAB><br>-1mat<br>=<br>X1, X2, X3 l<br>1. -0. -0.25<br>-0. 0.5 -0.<br>0. -0. 0.125 |
| && | 1 | BOOL | for all | si,66,si && = si |
| \|\| | 1 | BOOL | existence aggregation | 1=2,no \|\|<br>= no |

| Operation | Arity | Output-TT | Meaning | Examples |
|---|---|---|---|---|
| .. | 2 | Zahl l | from to | 1 .. 4 = 1 2 3 4 |
| ... | 3 | Zahl l | from to ! step | 0 … 6!2 = 0 2 4 6 |
| ..x | 3 | Zahl l | random numbers<br>from to ! count | 1 ..x 6!3= 5 3 2 |
| 1in | 2 | BOOL | a word on the left is contained in the right | [1 2] 1in “1 3 4“ = si |
| add | 2 | TT1 | add the second table to the first, where the column names have to agree | <TAB!<br>X1,X2 l<br>1  0<br>0  2<br>!TAB><br>add <TAB!<br>X2,X1,X3 m<br>4  5  6<br>7  8  9<br>!TAB><br>results<br>X1, X2  l<br>1   0<br>0   2<br>5   4<br>8   7 |
| abs | 1 | Zahl | absolute amount | -3 abs = 3<br>7 abs =7 |
| at | - |  | to the right of | Z:= Y+3 at X |
| aus | - |  | (new) beginning in the program | aus rivers.tabh |
| avec | 1 | TT1 | selection (whith, where) | rivers.tabh<br>avec LAENGE >800 |
| comp | 1+name | Name | component | NAME, VORNAME,LOC<br>Mill  Paul    Halle<br>comp LOC<br>results Halle; siehe auch nth |
| cos | 1 | TT1 | cosine | pi cos = -1. |
| det | 1 | Zahl | determinant | <TAB!<br>X1,X2,X3 l<br>1  0  2<br>0  2  0<br>0  0  8<br>!TAB><br>det<br>= 16. |
| div | 2 | ZAHL | integer division | 11 div 5 =2 |

| Operation | Arity | Output-TT | Meaning | Examples |
|---|---|---|---|---|
| `divrest` | 2 | ZAHL,ZAHL | integer division with remainder | `11 divrest 5 = 2,1 (not 2.1)` |
| `e` | 0 | PZAHL | Euler's constant | `e hoch 3 ln`<br>results<br>`3.` |
| `empty_t` | 0 | empty_s | Empty table with empty head | `aus empty_t`<br>`X:=1`<br>results<br>`X`<br>`1` |
| `gib` | 1+scheme+.. | S2 | restructure,<br>transform a tabment into a tabment with a given scheme or given TTD | `aus students.tab`<br>`gib FAC,(LOC,NAMEm m)m` |
| `giball` | 1+scheme | S2 | all values | `giball X \| Y l`<br>list of all X- and Y-subtabments (arbitrary depth); corresponds to doubleslash ...//X\|Y of XPath |
| `gibtop` | 1+scheme | S2 | only the top values | `gibtop Xl`<br>corresponds to slash: t/X: list of all X-subtabmente of t, occurring in the to level of t. |
| `hoch` | 2 | TT1 | to the power of | `4 hoch 1/2 = 2.` |
| `if then else` | 3 | TT2=TT3 | if then else | `if 3=4 then 5 else 6`<br>results `6` |
| `if then` | 2 | TT2 | if then | `if 3=4 then 5` results `empty` |
| `igib` | 1+scheme | S2 | join and restructuring | `studenten.tab,faks.tab igib FAC,DEAN,NAMEm m` |
| `in` | 2 | BOOL | containment of words and numbers | `“1 2 1“ in “1 2“ = si`<br>`“1 2 3“ in “1 1 2“ = no` |
| `inmath` | 2 | BOOL | mathematical containment | `[1 3] inmath [1 4 3] =si`<br>`2 inmath {6 7 2} =si` |

| Operation | Arity | Output-TT | Meaning | Examples |
|---|---|---|---|---|
| keys | 2 | TT1 | efficient selection in sets or lists | Xl:= 1 ..40<br>Y:=X*X<br>gib X,Y m<br>keys [7 34]<br>results in tab-Format:<br>X,  Y m<br> 7   49<br>34 1156<br>or<br>keys <![yy,[y2] zz]!> |
| keyslike | 2 | TT1 | efficient selection in sets or lists | <TAB!<br>NAME,   LOC m<br>Clara   Oehna<br>Claudia Dallgow<br>Sophia  Dallgow<br>!TAB><br>keyslike ["*ia"]<br>results<br>NAME,   LOC m<br>Claudia Dallgow<br>Sophia  Dallgow |
| leftat | - | - | left at | GROSS:=NET*1.19<br>leftat NET |
| letterb | 1 | WORTb | build a bag of letters | "Heute ist ."<br>letterb<br>results im tabh-Format:<br>WORTb<br>. e e H i s t t u |
| letterm | 1 | WORTm | build a set of letters | "Heute ist Dienstag."<br>letterm<br>results im tabh-Format:<br>WORTm<br>. a D e g H i n s t u |

| Operation | Arity | Output-TT | Meaning | Examples |
|---|---|---|---|---|
| `letterl` | 1 | WORTl | build a list of letters | `"Heute ist Dienstag." letterl`<br>results im tab-Format:<br>`WORTl`<br>`H`<br>`e`<br>`u`<br>`t`<br>`e`<br> <br>`i`<br>`s`<br>`t`<br> <br>`D`<br>`i`<br>`e`<br>`n`<br>`s`<br>`t`<br>`a`<br>`g`<br>`.` |
| `like` | 2 | BOOL | similar | `Hadmersleben like "?admers*"`<br>`= si`<br>'?': represents one letter<br>'*': zero or more letters |
| `linreg` | 1 | Y0,ANSTIEG | linear regression | `<TAB!`<br>`PRICE,         SOLD l`<br>`20               0`<br>`16               3`<br>`15               7`<br>`16               4`<br>`13               6`<br>`10              10`<br>`!TAB>`<br>`linreg`<br>`=`<br>`Y0,      ANSTIEG`<br>`19.73214 -0.98214` |

| Operation | Arity | Output-TT | Meaning | Examples |
|---|---|---|---|---|
| `lists` | 1 | TT1 l | list of lists | `Xl:= 1 2`<br>`lists 2`<br>results (tabh-Format)<br>`Xl l`<br>`1 1`<br>`1 2`<br>`2 1`<br>`2 2` |
| `ln` | 1 | TT1 | natural logarithm | `e ln = 1.` |
| `log` | 2 | TT1 | general logarithm | `100 log 10 = 2.` |
| `lower` | 1 | TT1 | to small letters | `AsdRRGee34 lower =`<br>`asdrrgee34` |
| `mal` | 2 | TT1 l | make | `Auto mal 3`<br>`= Auto Auto Auto`<br>`or`<br>`xx.tab mal 3` |
| `max` | 1 | Zahl | maximum of all numbers | `12.21,2,Hallo`<br>`max`<br>results<br>`12.21` |
| `median` | 1 | Zahl | middle number | `1 2 4,9.9`<br>`median`<br>results<br>`3.0` |
| `meta` | 1+Name | Modification of TT1 | transform data into meta data | `<TAB!`<br>`SUBJ,NOTE m`<br>`Deu  1`<br>`Phy  2`<br>`Ma   1`<br>`!TAB>`<br>`meta SUBJ`<br>results<br>`DEU, MA, PHY`<br>`1    1   2` |
| `min` | 1 | Zahl | minimum of all numbers | `12.21,2,Hallo`<br>`min`<br>results<br>`2` |
| `next` | - |  | begin of the second expression | `X:=first 100 next X`<br>`pred`<br>`     *1.03 at Y` |
| `nextonr` | - |  | next for onr-recursion | `X:=firstonr 100`<br>`nextonr X pred`<br>`     *1.03 at Y` |

| Operation | Arity | Output-TT | Meaning | Examples |
|---|---|---|---|---|
| `no` | 0 | BOOL | Boolean constant: false; corresponds to the answer no (Spanish) | `no \| si = si` |
| `not` | 1 | TT1 | negation | `si not = no` |
| `nth` | 2 | TT1 ohne m | n-th component | `1 3 5 nth 2 = 3` |
| `onr` | 1 | TT1 | conversion into otto-numbers | `1 3 5.2 "4.5.5" onr`<br>results in tabh:<br>`1 3 5.2 4.5.5` |
| `onrs` | 1+Name +1 | TT1 extended by Name | generate otto-numbers in a table | `<TAB!`<br>`X,Ym m`<br>`k y`<br>`  z`<br>`y w`<br>`!TAB>`<br>`onrs OTTO!k`<br>results<br>`X, (OTTO, Y  m) l`<br>`k   1     z`<br>`    2     y`<br>`    2.1   w` |
| `pi` | 0 | PZAHL | circle constant | `CIRCLEAREA:=R*R*pi` |
| `poly`<br>`polynom` | 2 | TT1 | polynomial | `3 poly [1 2 3]`<br>results<br>`18` |
| `pos` | Name | ZAHL | position | `avec X pos  < 10` |
| `pos-` | Name | ZAHL | position from back | `avec X pos- > 5` |
| `pred` | Name | Name | predecessor | `X:= first 100`<br>`    next X pred *1.03` |
| `pred_n` | Name+1 | Name | n-th predecessor | `X pred_n 3` |
| `pzahl` | 1 | TT1 with PZAHL | conversion | `1/5 6 9.7 pzahl`<br>results als tabh-format<br>`0.2 6. 9.7` |
| `pzahl1de` | 1 | TT1 with PZAHL | extraction | `"Heute bekomme ich 356,88 Euro und nicht 66,8 ."`<br>`pzahl1de`<br>results<br>`356.88` |

| Operation | Arity | Output-TT | Meaning | Examples |
|---|---|---|---|---|
| `rat` | 2 | RATIO | conversion into RATIO | `<TAB!`<br>`X,Yl l`<br>`1 2`<br>`  3`<br>`!TAB>`<br>`Z:= X rat Y`<br>results<br>`X, (Y, Z  l) l`<br>`1   2  1/2`<br>`    3  1/3` |
| `ratio` | 1 | TT1 mit RATIO | conversion | `1/5 6 9.7 ratio`<br>results (tabh-format)<br>`1/5 6/1 97/10` |
| `rename` | 1+Name +Name | TT1 up to N2 | column name renaming | `rename X!Y` |
| `rest` | 2 | TT1 | rest of integer division | `13 rest 5`<br>results<br>`3` |
| `rnd` | 2 | TT1 | round | `17.678 3.45 zz 8 rnd 1`<br>results<br>`17.7 3.5 zz 8` |
| `route` | 1 | TT1 | fill the route sequence | `<TAB!`<br>`X,Y m`<br>`0 0`<br>`1 1`<br>`0 1`<br>`!TAB>`<br>`route`<br>plots the 2 lines from (0,0) to (1,1) and (1,1) to (0,1) |
| `sans` | 1+Bed | TT1 | selection (without) | `sans LOC=Magdeburg`<br>`sans Magdeburg`<br>`sans:` without the specified (complex) tuples |
| `satzl` | 1 | SATZl SATZ! TEXT | list of all sentences | `"Es ist prima. Toll. Morgen feiern wir." satzl`<br>results (tabh-format):<br>`SATZl`<br>`Es ist prima.`<br>`Toll.`<br>`Morgen feiern wir.` |

| Operation | Arity | Output-TT | Meaning | Examples |
|---|---|---|---|---|
| seg | Name | elementary tuple | segment | enkel.tabh<br>avec Oehna in NAME seg<br>or<br>X  seg ++:<br>average of all numbers of the segments, containing X |
| si | 0 | BOOL | truth value true (corresponds to the answer yes) | si & no = no |
| sin | 1 | TT1 | sine function | 3.14159 sin<br>=2.65358979335e-06 |
| split | 2 | S1 l | split text | LOCl:= "Brati,Novi Sad, Belgrad" split ","<br>result (ment):<br><TABM><br>  <LOC>Brati</LOC><br>  <LOC>Novi Sad</LOC><br>  <LOC>Belgrad</LOC><br></TABM> |
| sqrt | 1 | TT1 | square root | 4 sqrt<br>results<br>2. |
| streuung | 1 | PZAHL | mean variation | [1 2 5 3 5 1] streuung<br>results<br>1.5 |
| subtext | 3 | TT1 | subtext (substring) | aBCdE subtext 2 ! 3<br>= BCd |
| subtext2 | 3 | TT1 | subtext | aBCdEfgh subtext2 "B" ! fg<br>=CdE |
| succ | Name | | successor | X succ |
| succ_n | Name+1 | | n-th successor | X succ_n 3 |
| tag0 | 1+Name | Name | put a out-most tag | 11 13  tag0 XX<br>results (ment-format)<br><XX><br>11<br>13<br></XX> |

| Operation | Arity | Output-TT | Meaning | Examples |
|---|---|---|---|---|
| `tags` | 1+<br>Name | Name collection | give each element of a collection a name | `{ 1 3 } tags XY`<br>results (ment)<br>`<TABM>`<br>`  <XY>1</XY>`<br>`  <XY>3</XY>`<br>`</TABM>` |
| `tagtup` | 1+<br>NameTup | Name-tuple | give each component of a tuple's name | `1,4 tagtup X,Y`<br>results (ment)<br>`<TABM>`<br>`  <X>1</X>`<br>`  <Y>4</Y>`<br>`</TABM>` |
| `tan` | 1 | TT1 | tangent function | `3.14 tan`<br>results<br>`-0.00159265493641` |
| `text` | 1 | TT1 | conversion | `3.14  ttt 8 text`<br>results<br>`TEXTl`<br>`3.14 ttt 8` |
| `textend` | 2 | TT1 | subtext | `asdfgh textend 4 =fgh`<br>tail of the text from the specified position |
| `textend-` | 2 | TT1 | subtext | `asdfgh textend- 4 =dfgh`<br>tail of the text from the specified position counted from back |
| `textindex` | 2 | ZAHL | position | `"Heute ist Dienstag." textindex Di`<br>results<br>`ZAHL`<br>`11` |
| `time` | 0 | PZAHL | system time | `time`<br>may result<br>`PZAHL`<br>`1.557021` |
| `trim` | 1 |  | remove spaces at the back and front | `"  Hi o++o   " trim`<br>results im ment-format<br>`<TABM>Hi o++o</TABM>` |
| `tup` | Name | whole tuple | whole tuple | `enkel.tabh`<br>`avec Deu in NAME tup` |

| Operation | Arity | Output-TT | Meaning | Examples |
|---|---|---|---|---|
| `untag0` | 1 | | omit the out-most tag | `X:=1`<br>`untag0`<br>results<br>`ZAHL`<br>`1` |
| `upper` | 1 | TT1 | to capital letters | `1.2,aW upper`<br>results (tab-format)<br>`PZAHL, WORT`<br>`1.2    AW`<br>each small letter is transformed into a capital letter, the other digits remain unchanged |
| `variance` | 1 | PZAHL | variance | `[1 2 4 6] variance`<br>`=`<br>`4.91666666667` |
| `verti` | 1+scheme+scheme | TT1-S2+S1 | arrange data vertical | `verti MON,XX l:=JAN ..DEC`<br>`verti SUBJ,MARKl l:= PHYl ..MAl` |
| `vlists` | 1 | TT1 l | list of lists | variable long lists; the operation is the same as lists, only that all shorter lists are still in result included. |
| `weg` | 1+Namen | TT1 without the names | omit column | `<TABH!`<br>`X,Ym m`<br>`1 2 3`<br>`4 5`<br>`!TABH>`<br>`weg Y`<br>results (tab-format)<br>`Xm`<br>`1`<br>`4` |
| `WORT` | 1 | TT1 with WORT | conversion | `"Ich bin gut.Du auch."`<br>`WORT`<br>results<br>`WORT`<br>`Ich_bin_gut.Du_auch.` |
| `WORTb` | 1 | WORTb | bag of all words | `"Ich bin. Ich auch."`<br>`WORTb`<br>results (tabh-format):<br>`WORTb`<br>`auch bin Ich Ich` |
| `WORTm` | 1 | WORTm | set of all words | `"We are 6."`<br>`WORTm`<br>`= {6 are we}` |

| Operation | Arity | Output-TT | Meaning | Examples |
|---|---|---|---|---|
| `WORTl` | 1 | WORTl | list of all words | `"We are 6."`<br>`WORTl`<br>results (tabh-format):<br>`WORTl`<br>`We are 6` |
| `zahl` | 1 | TT1 with ZAHL converts PZAHL and certain texts into ZAHL | conversion | `"12" zahl`<br>results<br>`12`<br><br>`3.14 zahl`<br>results<br>`3` |
| `zahl1` | 1 | TT1 with ZAHL | first number in text | `"24:5:33" zahl1`<br>`=24` |
| `zahl1de` | 1 | TT1 with ZAHL | first German number in text | `"Heute bekomme ich 66.356,88 Euro"`<br>`zahl1de`<br>`results`<br>`66356` |
| `zahl2` | 1 | TT1 with ZAHL | second number in text | `"24.05" zahl2`<br>`=5` |
| `zahl3` | 1 | TT1 with ZAHL | third number in text | `"24:AA:5::087" zahl3`<br>`=87` |

Finally, the operations for schemes should be specified. They are used, for example, in `gib` and `igib` statements.

| **`Operation`** | Arity | Meaning | Examples |
|---|---|---|---|
| `,` | 2 | pair formation at the schema level | `NAME,LOC`<br>`NAME,HOBBYl` |
| `b` | 1 | scheme for bag | `NAMEb` |
| `m` | 1 | scheme for set | `LOCm`<br>`NAME,LOC m` |
| `l` | 1 | scheme for list | `MARKl` |
| `b-` | 1 | scheme to sLOC multisets downwards | `NAMEb-` |
| `m-` | 1 | scheme to sLOC sets downwards | `SALARY,NAME,LOC m-` |
| `l-` | 1 | scheme to reverse the order | `MARKl-` |
| `\|` | 2 | alternative | `MARK \| EXAM l`<br>`NAME,(MARK \| EXAM l) m` |