The most important Keywords of o++o

(16.05.2018)

KEYWORD	CATEGORY	DESCRIPTION
+	binary algebraic operation	**Addition**: 3+6=9 (2,3,Paul)+4=(6,7,Paul)
*	binary algebraic operation	**Multiplication**: (2,3,Paul)*4=(8,12,Paul); tab*z1: every number of tab is multiplied by z1; the rest remains unchanged; tab1 * tab2: both tabs must be of the same type
-	binary algebraic operation	**Subtraction**: 7-3 = 4:
:	binary algebraic operation	**Division**: 6 : 4 = 1.5
++	unary algebraic operation (aggregation)	**Sum**: [1 2 3] ++ = 6 1, Wladimir, 8 ++ = 9
**	unary algebraic operation (aggregation)	**Product**: [2 3 7]** = 42
--	unary algebraic operation	**multiple subtraction**: [55 3 4] -- = 48
::	unary algebraic operation (aggregation)	**multiple divisions**: [64 2 2 2] :: = 8

++1	unary algebraic operation (aggregation)	**Count** (statistics): `[5 6] ++1 =2`;
		`[2 3 5 7 8 9], {rr tt zz uu},99`
		`++1`
		`= (6,4,1)`

+m, +b, +l	binary operation	set theoretic **union,** but also elements can be added to the collection, ..., the collection symbol indicates the target type
-m, -b, -l	binary operation	**Set difference**:

		<TAB! X,Y l 1 2 3 4 !TAB> -l ## <TAB! X,Y l 3 4 1 5 !TAB> = X, Y l 1 2
:m, :b, :l	binary operation	set theoretic **Intersection**,….
*m, *b, *l	binary operation	Cartesisches product
,	binary operation	**Pair:** 1,6,(otto,9) = 1,6,otto,9 (4-tuple) 1,otto =(1,otto) is of type (ZAHL,WORT) 1,2 unequal 1.2
;	base symbol	**Semicolon:** can be in front of a binary operator; the second argument then extends to the next semicolon or to the end of the line: 1+2 ;* 3+4 ;*2+3 = 105
blank	base symbol	[1 2 4] = List of 3 numbers also separates values and operations; An indentation (four spaces) joins new line with the previous one to a logical unit
=	relation symbol	LOC = Hadmersleben for example, selects all persons who live in Hadmersleben
:=	base symbol	**Zuweisung**: GROSS: = NET + TARA newname : = expression newname is assigned the result of the calculation of expression
::=	base symbol	New values are assigned to an existing column. n ::= e: replace the contents of the "column" n with the values of the expression e
=:	base symbol	**Assignment after**: 1,2 =: $XX
&	binary Boolean operation	**Konjunction (and):** si & no = no
|	binary Boolean operation	**Disjunction** (or): si | no = si

->	binary Boolean operation	**locical implication**: si `-> no = no`
<->	binary Boolean operation	**logical equivalence:** no `<-> no = si`
>, <, <=, >=	relation symbols	`3>4` is wrong (no)
“	quotation marks	used to merge texts with spaces into a single entity Otto = "Otto", "It's raining today" is a text; (It's raining today) is a list of words.
+text	binary text operation	**concatenation**: `Hallo +text ” “ +text Otto = “Hallo Otto”` The type of the first table remains unchanged.
{}	Brackets for sets of values	{1 1}={1}
{{ }}	Brackets for bags (multi-sets) of values	{{1 1}} unequal {{1}}
[ ]	Brackets for lists of values	[1 3 4]
#	base symbol	**Comment**: the following text of the line is comment
$	base symbol	**denotes a (tabment) variable**: $X:=(8,9)
(#, #)	limiter	**Comment: Begin/End** of multi-line comment
|	base symbol	**Alternative**: X|Y l; (bar): [| | |]

+tup	binary algebraic operation	`<TAB!`
		`X,Y l`
		`1 2`
		`3 4`
		`!TAB>`
		`+tup (4,5)`
		`=`
		`X,Y l`
		`5 7`
		`7 9`
		to each tuple of a tabment, a second tuple is added

*tup	binary algebraic operation	**Tuplemultiplication:** (2,3) *tup (4,5)= (8,15)
-tup	binary algebraic operation	**Tupledifference**:
:tup	binary algebraic operation	**Tupledivision**:

*mat	binary operation	**matrix multiplication:** (1,2) *mat [2 3]=8
		`<TAB!`
		`X1,X2,X3 l`

		1 0 2 0 2 0 0 0 8 !TAB> *mat ## <TAB! X1, X2, X3 l 1. -0. -0.25 -0. 0.5 -0. 0. -0. 0.125 !TAB> = X1, X2, X3 l 1. 0. 0. 0. 1. 0. 0. 0. 1.
-1mat	unary operation	<TAB! X1,X2,X3 l 1 0 2 0 2 0 0 0 8 !TAB> -1mat = X1, X2, X3 l 1. -0. -0.25 -0. 0.5 -0. 0. -0. 0.125 **inverse matrix**
<, >	limiter	**Begin/End of Tags:** <HSQ! ; !TAB>
/	base symbol	AUTHOR/NAME: in the XML representation the tag NAME is contained directly in AUTHOR; this is e.g. needed if the following two types are given: AUTHOR! NAME, FIRSTNAME EDITOR! NAME, FIRSTNAME, FIRMA
//	base symbol	AUTHOR//VOR: between AUTHOR and FIRST multiple tags are allowed; AUTHOR! NAME, COMPANY NAME! SUR, FIRST, MID ?, but AUTHOR/NAME/FIRST is more efficient
!	base symbol	**limiter**: [X! 77 88] is used for single line tabments
&&	unary operation (aggregation)	Statistics: **for all** aggregation: si,66,si && = si

||	unary operation aggregation	Statistics: **exist** aggregation**:** WEIBLICHl || 1=2,7=8 || = no
..	binary operation	1 .. 4 ** = 24 (corresponds here to the faculty function)
...	operation	1 ... 4,2 = 1 3 x ... y,z: all numbers x, x+z, x+2z,..., x+n*z: x+n*z<=y
1in	relation symbol	[1 2] onein “1 3 4” = si one word or number of the left is included in the right
abs	unary operation	: 7~ abs = 7 absolute amount of a number (ZAHL or PZAHL) 6 abs = 6
add	binary algebraic operation	t1 add t2: insert t2 in t1 with the same column names; the type of t1 is the result type; similar to the gib clause <TAB! X,Yl m 1 2 3 4 5 !TAB> add ## <TAB! Y,X 6 1 !TAB> = X, Yl m 1 2 3 6 4 5
add1	binary algebraic operation	[1 7] add1 6 =[1 7 6] $x add1 $y: add an element $y to the collection $x
at	keyword	**extent the tabment at a column** ext n:=e at n2 expand to the right of n2
atom	gib-clause	atomic subtabment atom! HOBBYl HOBBYl will be transferred as a whole during the restructuring
aus	keyword	aus studenten.tab

avec	keyword	**Selection**
		`avec SALARY > 5000`
		only the specified (complex) lines remain in the result
		`avec DEPT!  SALARY >5000`
		all departments where one employee earns more than 5000

BAR	data type	contains only one element (|); therefore only BARl useful
begin, end	limiter	**Begin/End** of a subprogram
BOOL	data type	contains 2 truth values (si, no)

comp	binary operation	`NAME,   VORNAME,LOC`
		`Miller  Paul    Magdeburg`
		`comp LOC`
		results in `Magdeburg;` see also `nth`

cos	trigonometric operation	**Cosine**: 0 `cos=1.`

crosstab	binary algebraic operation	**cross table**:
		`<TAB!`
		`NAME, (SUBJECT,MARK m)m`
		`Paul   Mathe   1`
		`       Deutsch 2`
		`Sophia Mathe   2`
		`       Bio     1`
		`       Deutsch 1`
		`!TAB>`
		`crosstab`
		`=`
		`NAME,  BIO?, DEUTSCH?, MATHE? m`
		`Paul         2         1`
		`Sophia 1     1         2`
		the values of the penultimate column become column names and the values of the last column become the associated values

csv	Suffix	**csv-file** can serve as input or output to o++oPS programs

dann	keyword	` 1=1 dann 2,3 = 2`
		` 1<1 dann 2,3 = 3`
		`condition2 dann expr1, expr2`

dann1	keyword	`XX := LOC=Halle dann1 gut` XX receives only one entry if the person is from Halle, that means it is extended by XX?.

det	Matrix operation	`<TAB!`
		`X1,X2,X3 l`
		`1 0  2`

		0 2 0 0 0 8 !TAB> det = 16. **determinant**
div	binary algebraic operation	**integer Division** 7 div 3 = 2 8.55 div 2.1 = 4
divrest	binary algebraic operation	**integer division with rest** 7 divrest 3 = 2,1 7.1 divrest 3.5 = 7,0.1
ext	base operation	**extension (Erweiterung)**: ext n1:=e at n2 erweitere das gegebene Tabment um (komplexe) n1-Werte rechts neben jedem n2-Wert
gib	base operation	restructure, sort, aggregate, union, eliminate duplicates, ...: aus students.tab gib FAC,(LOC,NAMEm)m)m transform a Tabment into a Tabment with given scheme or given TTD
gib+	base operation	restructure several tabments; before the restructuring, the "join" ("intersection") of the given tables is realized. aus studenten1.tab,examen1.tab gib+ FAK,(NAME,(KURS,NOTE m)m)m
giball	base operation	giball X | Y l List of all X and Y elements (arbitrary depth); corresponds to the double-slash ... // X | Y of XPath
gibtop	base operation	gibtop Xl corresponds to the slash: t/X: List of all X-subtabments of t that occur in the top level of t.
hoch	binary algebraic operation	**to the power of** 2 hoch 3 = 8 e hoch 2 = 7.38905609893
hsq	suffix	Input and output file; each line corresponds to one segment; the fields of a segment are separated by 2 or more spaces
in	relation symbol	are left and right side collections, so in is the "set theoretic" inclusion, [1 3] in [1 4 3] = si if the left side of the element type is the right one, then in is in the element relation 2 in {6 7 2} = si otherwise “in” is not defined.
inm, inb, inl	relation symbol	the left hand side and the right hand side are transferred to

		word and number collections; then it is tested whether the left word and numbers are contained in the right ones
leftat	keyword	`ext n:=e leftat n2`: see above **at**

like	Boolean predicate	`Hadmersleben like "?admers*"`
		`= si`
		`'?'`: represents one character
		`'*'`: zero or more characters

linreg	unary algebraic operation Aggregation	`<TAB!`
		`FLASCHENPREIS, VERKAUFTEMENGE l`
		`20               0`
		`16               3`
		`15               7`
		`16               4`
		`13               6`
		`10              10`
		`!TAB>`
		`linreg`
		`=`
		`Y0,           ANSTIEG`
		`19.7321428571 -0.982142857143`

lists	unary base operation	**List of Lists:**
		`[X! 1 2] lists 2`
		`=`
		`Xl l`
		`1 1`
		`1 2`
		`2 1`
		`2 2`

ln	unary algebraic operation	**natural Logarithm**; e ln =1.
m, mv, b, bv, l, lv, a, ?	collection symbols	m: set (Menge), mv: set vice versa, b bag, l: list, a: collection of ANY-Typ, ?: optional values the collection symbols are noted postfix and can be appended to a tag without spaces.
mal	Tabment operation	times 5 mal "Ich liebe Dich!" = Ich liebe Dich! Ich liebe Dich! Ich liebe Dich! Ich liebe Dich! Ich liebe Dich!

max	unary operation Aggregation	**maximum (statistics)**
		`1.1,2,Hallo`
		`max = 2.`

median	unary operation Aggregation	`(2 6 3 2),7,8`
		`median`
		results in
		`4.5`

ment	Suffix	document representation of a tabment; differs actually only in a simpler representation of meta data

min	unary operation
	Aggregation
		minimum (statistics)
		`1.1,2,Hallo`
		`min`
		results in `1.1`

no	Booloperation	**Boolean constant** (false)

not	unary Boolean operation	`si not = no`
		negation
		`X not not = X`

nth	binary operation	n-th component of a tuple resp. n-th element of a collection `emperors.tab nth 2 comp NAME`
polygon	unary algebraic operation	`[X,Y! 0 0  1 1  0 1]` draws the 2 lines (0,0) to (1,1) and (1,1) to (0,1)

polynom	binary alebraic operation	`2 polynom [3 1 4]`
		computes the value of 3*X*X+X+4
		for the argument 2

pos	unary position operation	`MARK pos < 5` the first four marks
pos-	unary position operation	`MARK pos- < 5` the last 4 marks
pred	unary position operation	predecessor 1. within ext: NAME pred: NAME of the predecessor 2. within rec: AMOUNT pred: AMOUT value of the predecessor
pred_n	binary position operation	`MARK pred_n 3` third **predecessor** within a collection
PZAHL	data type	**number with point (f**loat) 2.34
pzahl	conversion operation	convert a number or a text into a **PZAHL**
pzahlen	unary Tabmentoperationen	all floats (PZAHL) of a tabment are output (no type conversions)
rename to	base operation	`rename X to Y` **replace** the column name X by Y
rest	binary algebraic operation	rest of the integer division `7 rest 3 = 1`

rnd	binary algebraic operation	`[2.1436 5.88] rnd 1`
		`=`
		`2.1 5.9`
		`z rnd n`: **round** z to n digest after point

sans	keyword	**selection** sans LOC=Magdeburg sans: without the specified (complex) tuples
saetze	text operation	VITA saetze**: List of all sentences** the resultat is of type SATZl
si	Boolean constant	truth value true (corresponds to the answer **yes**)
sin	unary trigonometric operation	3.14159 sin =2.65358979335e-06 **sine function**
sqrt	unary algebraic operation	**square root** 4 sqrt =2.
streuung	unary Aggregation	[1 2 5 3 5 1] streuung (mad) = 1.5
strip	unary base operation	<TAB! X,Y?,Zl,Wm m 1 2 3 4 !TAB> strip = (X, Y?, Z?, Wm)? 1 2 3 4 All collection symbols, each collection containing at most 1 item, will be replaced by ? .
subtext	text operation	aBCdE subtext 2,3 = BCd
subtext2	text operation	aBCdEfgh subtext2 "B", fg =CdE
subtextend	binary text operation	asdfgh subtextend 4 =fgh **rest of the text** from the specified position
subtextend-	binary text operation	asdfgh subtextend- 4 =dfgh **rest of the text** from the specified position from the back side
succ	unary position operation	MARK succ **Successor** within one collection
succ_n	binary position operation	MARK succ_n 3 third **succsessor** within one collection
tab	Suffix	a **Tabment** in table view

TABMENT	Tag	**virtual tag** around the current tabment
tag	tabment operation with parameter	t1 tag ROOT around t1 a ROOT-tag is placed
tags	Tabmentoperation with parameter	1 bis 3 tags X = Xl 1 2 3
TEXT	data type	**text data type** (string)
text	unary text operation	13.2,[ab cc],Bc text =“13.2 ab cc Bc“ all values are converted into text and concatenated
textsep	binary text operation	1 bis 10 textsep "," = 1,2,3,4,5,6,7,8,9,10
textindex	binary text operation	“Heute wird schoenes Wetter“ textindex wir =7
textcut	binary text operation	
time	algebraic operation	ext X:= time =1449251939.91 **system time** usually needs to be applied twice to make up the difference
untag	unary base operation	1 tag X untag =1
upper	unary text operation	1.2,aW upper = 1.2 AW (hsq output) every lowercase letter is converted to a capital letter; the rest remains unchanged.
variance	Aggregation	[1 2 4 6] variance = 4.91666666667
vertical	base operation	<TAB! NAME, BIO?,GERMAN?, MATHS? l Paul 2 1 Sophia 1 1 2 !TAB> vertical ## SUBJECT,MARK l:=BIO,GERMAN,MATHS = NAME, (SUBJECT, MARK m)m Paul DEUTSCH 2

		`        MATHE    1`
		`Sophia  BIO      1`
		`        DEUTSCH  1`
		`        MATHE    2`
		vertical X,Y m :=C,D,E
		the column names C, D, E appear in column X and the corresponding values in column Y

vlists	unary base operation	variable long lists; the operation is nearly the same as `lists`, except that all shorter lists are still included in the result
weg	base operation	`weg XX Y` omit the columns (tags) XX and Y

wege	base operation	a given table
		tab: `SUP,XX,…,(SUB,YY,…l)m`
		is interpreted as a directed, weighted, cycle-free graph with edges from SUP to SUB.
		`tab`
		`wege sup0`
		is the list of all paths from sup0 to the "end node". She is from the type
		` ((SUB,YY,…)l)l.`
		`<TAB!`
		`SUP, (SUB,ANZ l)m`
		`t0`
		`t1`
		`t2`
		`t3     t1   5`
		`       t4   6`
		`t4     t2   3`
		`       t0   2`
		`!TAB>`
		`wege t3`
		`=`
		`(SUB, ANZ l)l`
		` t4    6`
		` t0    2`
		
		` t4    6`
		` t2    3`
		
		` t4    6`
		
		`    t1    5`
		`(the empty rows were inserted)`

worte	unary text operation	`"We are 6."`
		`worte`
		={6 are we}

		all **words** of a tabment
xml	suffix	studenten.xml: **XML-file**
ZAHL	data type	arbitrarily large integers(bigInt)
zahl	unary conversion operation	**konvertiere** TEXT oder PZAHL **in ZAHL** "12" zahl =12
zahl1	unary conversion operation	"24:5:33" zahl1 =24 **first number** in text;the text has to start with a digit
zahl2	unary conversion operation	"24.05" zahl2 =5 **second number** in text
zahl3	unary conversion operation	"24:AA:5::087" zahl3 =87 **third number** in text
zahlen	unary tabment operation	all numbers of an tabments (no type convertsions)
zufall	operation with list output	5 zufall 1,6 = 1 4 2 6 1