# Reference to Joy of Postfix

from 2024-10-21

Subset of Joy Programming Language
with some Modifications

Original:
https://www.kevinalbrecht.com/code/joy-mirror/html-manual.html

## Definition of Identifiers

*identifier1* **==** *word1 word2 word3* …
*identifier2* **==** *word4 word5 word6* …

**Example:**

makelist == [ ] swap [cons] times <CALC>
10 20 30 40 50 3 makelist .s <CALC>
… 10 20 [30 40 50]

quote '
notation -- erklären

## Words for the Stack

The parameter stack is a linked list.

**stack** -- *list*
Pushes the stack as a list onto the stack.

*list* **unstack**
The *list* becomes the new stack.

**clear** -- *(null)*
Clears the stack.

*x* **dup** -- *x x*
Pushes an extra copy of x onto the stack.

*x* **pop** --
Removes *x* from the top of the stack.

*x y* **swap** -- *y x*
Swaps *x* and *y* at the top of the stack.

*x y* **over** -- *x y x*
Gets the second value from stack.

*x y z* **rotate** -- *z y x*
Swap *x* and *z*.

*x y z* **rollup** -- *z x y*

*x y z* **rolldown** -- *y z x*

dupd --

popd --

swapd --

rotated --

rollupd --

rolldownd --

... *n* **index** -- *n*th_stack_value
Picks a copy of the stack value with position num relative to the
stack top from the stack and pushes it onto the stack;
with *n* = 1 -> first value, *n* = 2 -> second value, ...

*x [program]* **dip** -- ... *x*
Stores the *x*, executes the *program*, pushes *x* back onto the stack.

*x y [program]* **dip2** -- ... *x y*
Stores the *x* and *y*, executes the *program*, pushes *x* and *y* back onto the stack.

**id**
Identity function, does nothing; as a placeholder for a function.

## Words for Input/Output

*value* **.** --
Prints the top value from the stack. (monad behavior)

**.s** --
Prints the contents of the stack. (monad behavior)

*list* **print** --
*string* **print** --
Outputs the *list* without square brackets. (monad behavior)
Outputs the *string* without quotation marks. (monad behavior)

*fname* **load** --

*fname* **save** --

*fname* **loadtext** -- *string*
Loads the contents of a text file and pushes it
as a *string* on the stack. (monad behavior)

*fname string* **savetext** --
Saves the *string* as text in a text file. (monad behavior)

**files** -- *list* (monad behavior)

*fname* **fremove** -- *bool* (monad behavior)

**timestamp** -- *num* (monad behavior)

**date** -- *string* (monad behavior)

**words** -- identlist print (monad behavior)

**dump** -- identdump print (monad behavior)

**help** -- helpinfo print (monad behavior)

## Words for List Processing

**[***value1 value2 value3 ...***]**

| | | |
|---|---|---|
| *list* **first** | -- | *value* |
| *list1* **rest** | -- | *list* |
| *value1 list1* **cons** | -- | *list* |
| *list1 value1* **swons** | -- | *list* |
| *list1* **uncons** | -- | *value list* |
| *list1* **unswons** | -- | *list value* |
| *list1* **reverse** | -- | *list* |
| *list* **size** | -- | *num* |

make

| | | |
|---|---|---|
| *list1 num* **take** | -- | *list* |
| *list1 num* **drop** | -- | *list* |
| *list1 list2* **concat** | -- | *list* |
| *list1 list2* **swoncat** | -- | *list* |

enconcat

| | | |
|---|---|---|
| *list1* **last** | -- | element |
| *list1* **init** | -- | *list* |

*num* **iota** -- *list*

fromto

*list num* **at** -- elementvnum

of

.

find

count

*value1 value2* **pair** -- *[value1 value2]*

*[value1 value2]* **unpair** -- *value1 value2*

*x y* **pair** -- *[x y]*

*[x y]* **unpair** -- *x y*

## Words for Processing Dict Lists

**[***key1 value1 key2 value2 ... ...***]**

*dict key* **get** -- *value*

*dict1 key value* **put** -- *dict*

.

.

.

## Mathematical Functions

*num1 num2* **+** -- *num*

*num1 num2* **-** -- *num*

*num1 num2* ***** -- *num*
*num1 num2* **×** -- *num*

*num1 num2* **/** -- *num*
*num1 num2* **÷** -- *num*

*num1 num2* **mod** -- *num*
*num1 num2* **rem** -- *num*

*num1* **reci** -- *num*

*num1 num2* **pow** -- *num*

*num1 num2* **root** -- *num*

*num1* **pred** -- *num*

*num1* **succ** -- *num*

*num1* **sign** -- *num*

*num1* **abs** -- *num*

*num1* **neg** -- *num*

*num1* **floor** -- *num*

*num1* **ceil** -- *num*

*num1* **trunc** -- *num*

*num1* **int** -- *num*

*num1* **frac** -- num

*num1* **round** -- *num*

*num1 fix* **roundto** -- *num*

*num1* **exp** -- *num*

*num1* **log** -- *num*

*num1* **log10** -- *num*

*num1* **log2** -- *num*

*num1* **fact** -- *num*

**pi** --

*num1* **sin** -- *num*

*num1* **cos** -- *num*

*num1* **tan** -- *num*

*num1* **asin** -- *num*

*num1* **acos** -- *num*

*num1* **atan** -- *num*

*numy numx* **atan2** -- *num*

*num1* **sinh** -- *num*

*num1* **cosh** -- *num*

*num1* **tanh** -- *num*

*num1* **sq** -- *num*

*num1* **sqrt** -- *num*

*num1* **cbrt** -- *num*

*num1* **deg** -- *num*

*num1* **rad** -- *num*

*[num1 num2 ... numn]* **sum** -- *num*
Sum of all elements of the list.

*[num1 num2 ... numn]* **prod** -- *num*
Product of all elements of the list.

## Logical Functions

true and false are of type bool

**true**

**false**

*bool1* **not** -- *bool*

*bool1 bool2* **and** -- *bool*

*bool1 bool2* **or** -- *bool*

*bool1 bool2* **xor** -- *bool*

*data1 data2* **=** -- *bool*

*data1 data2* **<>** -- *bool*
*data1 data2* **!=** -- *bool*

*data1 data2* **<** -- *bool*

*data1 data2* **>** -- *bool*

*data1 data2* **<=** -- *bool*

*data1 data2* **>=** -- *bool*

*num* **small** -- *bool*
*list* **small** -- *bool*

*data1* **null** -- *bool*

*data1* **list** -- *bool*

*data1* **leaf** -- *bool*

*data1* **consp** -- *bool*

*data1* **bool** -- *bool*

*data1* **ident** -- *bool*

*data1* **float** -- *bool*

*data1* **string** -- *bool*

*data1* **undef** -- *bool*

*ident1* **user** -- *bool*

*data1* **type** -- *ident*
?

in

has

*data1 data2* **min** -- *data*

*data1 data2* **max** -- *data*

*list* **qsort** -- *list*

## String Functions

*string1 num1 num2* **substring** -- *string*

.

.

*string1 sub* **indexof** -- *num*

*string1* **upper** -- *string*

*string1* **lower** -- *string*

*string1* **capitalize** -- *string*

*string1* **trim** -- *string*

*string1* **triml** -- *string*

*string1* **trimr** -- *string*

*string1 pre* **trimpre** -- *string*

*num* **chr** -- *string*

*string* **ord** -- *num*

*string1 old new* **replace** -- *string*

*string1 old new* **replace1** -- *string*

*string sep* **split** -- *list*

*list sep* **join** -- *string*

*string* **unpack** -- *list*

*list* **pack** -- *string*

*string* **parse** -- *list*

*data* **tostr** -- *string*

*string* **toval** -- *data1*

*string* **trytoval**

*string* **strtod** -- *num*

*num* **timeformat** -- *string*

## Words for Flow Control and Combinators

i
dip
dip2
nullary
do
return
if
branch
ifte
choice
case
cond
times
while
loop
break
step
map
fold
filter
split2
cleave
primrec
tailrec
genrec
linrec
binrec

*[program]* **Y** -- ...
Y-Combinator in Joy

try

*num [program]* **' !** (monad behavior)
[*monad] [program]* **' !**
*program* can also be a monad. The monad is placed before
the output and triggers side effects and continues with the *program*.

*x [then] [else]* **ifnull** -- ...
*x [then] [else]* **iflist** -- ...
*x [then] [else]* **ifcons** -- ...
*x [then] [else]* **ifbool** -- ...
*x [then] [else]* **ifident** -- ...
*x [then] [else]* **iffloat** -- ...
*x [then] [else]* **ifstring** -- ...
*x [then] [else]* **ifundef** -- ...

## Misc Functions

| | | |
|---|---|---|
| *ident* **name** | -- | *string* |
| *ident* **body** | -- | *num* \| *list* \| undef |
| *ident* **info** | -- | *string* |
| **intern** | - | |
| *ident* **user** | - | |
| *ident* **bound** | - | |
| *data1* **type** ? | -- | *ident* |
| **identlist** | -- | *list* |
| **identdump** | -- | *string* |
| **helpinfo** | -- | *string* |
| **gc** | -- | |
| **abort** | >>> | exception |
| *string* **error** | >>> | exception |
| **undefined** | >>> | exception |

## eof