

A AAAAAAAAAAAAAAAAAAAAAAAAAA \subseteq

A Monadic form: $\subseteq \omega$

A Frequently occurring pattern: `enclose-if-simple`

`process 'red' 'green' 'blue'` A process several items

`process 'purple'` A or single item

∇ process items;item

```

[1]
[2] :If 1 $\equiv$ items
[3]     items $\leftarrow$ items
[4] :EndIf
[5]
[6] :For item :In items
[7]     AAA do stuff
[8] :EndFor

```

∇

becomes:

∇ process items;item

```

[1] :For item :In  $\subseteq$ items
[2]     AAA do stuff
[3] :EndFor

```

∇

A definition: $\subseteq \leftrightarrow \{c \times (1 \equiv \omega) \vdash \omega\}$

A Ex: think of an example from your own work.

A Dyadic form: $\alpha \subseteq \omega$ AAAAAAAAAAAAAAAAAAAAAAAAAA

A definition: $\subseteq \leftrightarrow \{\lceil ml \leftarrow 3 \diamond \alpha \subset \omega \}$ A APL2-style partition
A (may be applied under axis)

A \subset Dyalog default ($\lceil ml \leftarrow 1$) partitioned enclose:
A left argument must be boolean vector or scalar
A new partition at `_each_ 1`
A leading 0s omit items

1 0 1 0 \subset 3 4 5 6

3	4	5	6
---	---	---	---

0 1 0 1 \subseteq 3 4 5 6

4	5	6
---	---	---

text

some of your purple berries

A \subseteq Partition:

A left argument must be non-negative integer vector

A new partition when item > previous one (1,2</math>)

A all 0s omit items

1 1 2 2 \subseteq 3 4 5 6

3	4	5	6
---	---	---	---

1 0 2 2 \subseteq 3 4 5 6

3	5	6
---	---	---

split $\leftarrow\{(\alpha\neq\omega)\subseteq\omega\}$ A α -split of ω

' ' split text

some	of	your	purple	berries
------	----	------	--------	---------

split $\leftarrow \neq\subseteq$ A coded as a fork

' ' split text

some	of	your	purple	berries
------	----	------	--------	---------

A Ex: Write \subseteq in terms of \subseteq

A Ex: Discover something interesting to share

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA @

A {val} (mod @ sel) array

A Sub-array selection:

A numeric indexing

A boolean mask

A ...

A Right operand: [sel]ection:

A array:

A simple : major cell selection

A nested : choose / reach

A function: returns boolean mask

A Items of [array] are replaced with {val} mod sub,
A where [sub] is the sub-array of [sel]ected items.

A One last thing: this commonly occurring form:
A val (\rightarrow @ sel) array
A can be shortened to:
A (val @ sel) array

A Ex: write expressions for:
A vector ω with 0s at alternate positions
A Sentence ω with first letter of each word capital
A Exchanging rows p and q in matrix ω
A FizzBuzz

A Ex: Discover something interesting to share

A Implementation:
A 0. Monitor usage in "real" apps.

A Performance improvements:
A Phase 1 (done): split single MAGIC fn into sub-cases so
A that case-selection occurs in C rather than APL.
A Phase 2 (done): process `_function_` right operand in C-
A code to produce index array.
A Phase 3: Recode popular sub-cases into C to avoid inter-
A pretative overhead, and Unsharing() where possible.
A Phase 4: Further reduce Unsharing() by overwriting
A "garbage" right argument in-situ.
A Identifying the right argument as garbage might require
A some help from the compiler.

A In-situ substitution:

A 0@1 $\tau\omega$ A OK, because $\tau\omega$ is not shared

A 0@1 $\leftarrow\omega$ A OK, iff ω has low refcount

A $\omega+0@1\leftarrow\omega$ A NOT OK

A $v\leftarrow0@1\leftarrow\omega$ A NOT OK if ω is referenced downstream.

A 1+0@1 $\leftarrow\omega$ A NOT OK if a local error-guard references ω

A ... etc