

Dyalog APL/W Conference 2011 Unicode Edition
Serial No : 000000
Mon Feb 20 20:24:29 2012
clear ws

)load wg28
c:\demos\wg28 saved Mon Feb 20 15:19:54 2012

A John Scholes - Dyalog APL

10 × 2 3 4 ρ ι 24 A This is a live APL session.
0 10 20 30
40 50 60 70
80 90 100 110

120 130 140 150
160 170 180 190
200 210 220 230

AA
A Array: rectangular collection of items arranged along 0 or more axes,
A where an item is a number, a character or an array.
A Everything is an array. There is no access to underlying scalars!
A NB: APL recycles the word "scalar" to mean rank-0 array.
A
A Functions: infix, associate RIGHT with equal precedence.
A Functions take array "arguments" and return array results.
A A "monadic" function takes only a right argument.
A
A Operators: infix, associate LEFT with equal precedence.
A Operators curry array or function "operands" to form functions.
A A monadic operator takes only a left operand.
A
A Operator-operand binding is stronger than function-argument binding.
A APL has only a single level of high-order function.
A
A There is a rich set of primitive functions and operators.
AA

42[2 3ρ<θ] A indexing a rank-0 array
42 42 42
42 42 42

2×3+4 A right-associative, equal precedence

```

2+÷-4          A "monadic" use of functions
1.75
A Function definition: {α ∇ ω}

  {[ω+0.5] 3.4 ^5.6          A round to nearest
3 ^6
3 {ω*÷α} 64 125          A αth root
4 5
30 {ω=0:|α ∘ ω ∇ ω|α} 105  A GCD (Euclid)
15
A Operator definition: {α (αα ∇∇ ωω) ω}
A          ┌───∇───┘

÷{αα αα ω} 4          A twice (monadic)
4
0≠{αα ω:ωω ω ∘ 0}÷ 4          A sequential test (cf C's &&)
0.25
A We name things using ← (right-pointing finger :-)

vec ← 1 2 3          A naming an array
vec = 3 2 1          A cf: item-wise comparison
0 1 0
sum ← +/          A naming a "derived" (curried) function.
avg ← {(+/ω)÷ρω}  A naming a "defined" function ("D-fn").

sum avg" vec(4 5)  A sum of avg mapped over a 2-vector.
6.5
A A number is a number is a number ...

ι 5          A first five nats
0 1 2 3 4

ι 6+*o^-1*÷2          A ditto - (c) Euler
0 1 2 3 4

1=49x÷49          A tolerant comparison
1
A Many functions are shape-invariant:

)copy dfns.dws easter
c:\Home\dfns saved Thu Jan 26 09:52:17 2012

```

```

    cr'easter'      A display source of easter function:
easter←{
    G←1+19|ω      A year golden number in 19-year Metonic cycle.
    C←1+⌊ω÷100    A Century: for example 1984 → 20th century.

    X←-12+⌊C×3÷4  A number of years in which leap yr omitted.
    Z←-5+⌊(5+8×C)÷25 A synchronises Easter with moon's orbit.

    S←(⌊(5×ω)÷4)-X+10 A find Sunday.
    E←30|(11×G)+20+Z-X A Epact.
    F←E+(E=24)∨(E=25)∧G>11 A (when full moon occurs).

    N←(30×F>23)+44-F A find full moon.
    N←N+7-7|S+N      A advance to Sunday.

    M←3+N>31        A month: March or April.
    D←N-31×N>31     A day within month.
    ↑10000 100 1+.×ω M D A yyyyymmdd.
}

    easter 2012      A Easter this year April 8th
20120408

```

```

    ⍵ ← vec ← c[1 2]2000+2 2 3p12      A vector of matrices
2000 2001 2002 2006 2007 2008
2003 2004 2005 2009 2010 2011

```

```

    easter vec      A vector of easter matrices
20000423 20010415 20020331 20060416 20070408 20080323
20030420 20040411 20050327 20090412 20100404 20110424

```

A Some more examples:

```

pow ← {(αα*α)ω}      A Primitive operator * is power

```

```

0 1 ϕpow'' 'hello' 'world' A conditional application
hello dlrow

```

```

f ← (32÷+ )◦(×◦1.8)  A Operator ◦ is compose / curry
f ^273.15 ^40 0 100  A Fahrenheit from Celsius
^-459.67 ^40 32 212

```

```

c ← f*^-1            A *^-1 is inverse
c ^-459.67 ^40 32 212 A Celsius from Fahrenheit
^-273.15 ^40 0 100

```

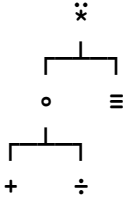
1 +∘÷×≡ 1
1.618033989

A ×≡ is fixpoint

)copy dfns dft
c:\Home\dfns saved Thu Jan 26 09:52:17 2012

+∘÷×≡ dft 3

A display function tree



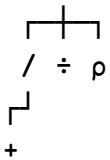
A NB:This section experimental - won't work in TryAPL.org
 A Iverson struggled for years to express calculus: f+g
 A And then, waking from a nap, he realised:
 A "Forms f+g and f×g were not used in APL
 A and could be introduced without conflict"
 A
 A (f g h)ω → (f ω)g(h ω) A monadic "fork"
 A α(f g h)ω → (α f ω)g(α h ω) A dyadic "fork"

mean ← +/ ÷ ρ A mean as fork
 mean 1 2 3 4

2.5

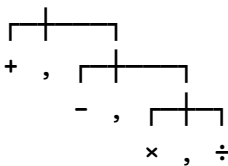
mean dft 1

A display of function train



(+,-,×,÷)dft 1

A display of train (forks of forks)



6(+,-,×,÷)2

A vector of fns! (comma not special)

8 4 12 3

A Forks (cf S) together with primitive functions:

2→3 A "right" (cf I)

3

2→3 A "left" (cf K)

2

A and the primitive composition operator f∘g,
 A provide a combinator base for function/array expressions.

A This means APL {...} "lambda" functions may be converted
A mechanically into tacit "combinator" form. Cool or what!

A Implementation

A APL is usually implemented as an interpreter.
A Interpretative overhead amortized over items of large array.

A←0.1× 1e6?1e6 A million-item 64-bit float vector.
B←0.1×1[1e6?1e6

)copy dfns cmpx

c:\Home\dfns saved Thu Jan 26 09:52:17 2012

cmpx'A÷B' A timing in seconds (may use multiple cores).
3.4E-2

A Dyalog "compiler" project.
A Pre-evaluation of (small) constant expressions.
A Elimination of local names.
A Byte code.
A General parse-tree topiary.
A ...

A Semantics: Arrays are passed by value.
A Implementation: (no they're not)
A APLers like to mutate their arrays:

A[100?1000]←0 A replace items of A with zeros

A Uses ref-counting to defer copy until/if mutation.

A More modern array-languages (J) have removed this mutation
A syntax in favour of a pure "merge" operator:
A new ← old (selection merge) vals
A but the performance considerations remain.

A Traditionally, APL has (over-) indulged its users
A by allowing them to:
A - interrupt the evaluation of an expression,
A - save the heap ("workspace") to a binary file,
A - update the interpreter executable,
A - reload the saved workspace,
A - resume execution.

(8↑''),<display demo.links A What's next?

```
→
| http://TryAPL.org:8080  A Play APL + more links.
| http://dfns.dyalog.com  A Lots of sample code.
|
| www.jsoftware.com     A J: Iverson's successor to APL
| www.kx.com             A K: Arthur Whitney's .. ..
| www.nars2000.org      A open-source APL (Bob Smith)
| www.smartarrays.com   A commercial API for APL-style arrays
|
| Dyalog conference Helsingør.DK, Oct 2012
| Minnowbrook.NY Array-language conference, autumn 2013?
|
| At a future WG2.8 meeting, please consider inviting:
|
|         Roger Hui (J)           Arthur Whitney (K)
```

A That's all folks! Questions?