Workshop: Programming With Dyalog APL

ECUST, July 5th 2018 Morten Kromberg

(based on work by Jay Foad, Roger Hui & John Scholes)



Agenda

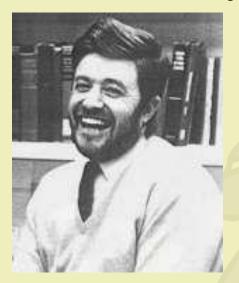
Morning

- Install APL
- Introduction to APL
- Exercises

Afternoon

- A quick look at tools for
 - Importing Data from Excel
 - Charts
- Review of MatLab code (SA_Cal) converted to APL
 Gitte + Morten depart for airport at 2PM

History of APL



Kenneth E. Iverson 1920-2004

- Canadian of Norwegian Descent
- Born on a small farm in Alberta (Canada)
- Finished one-room school after 9th grade and worked on the farm
- Army 1942, Flight Engineer in Air Force from 1943
 - Almost finished High School in the service
 - Promised his officers and mates that he would pursue an academic career after the war
- B.A. from Queens University, Kingston Ontario
 - Ken didn't know there was such a thing as University before he joined the army!



History of APL, continued



- Doctoral work at Harvard with Aiken and Leontief
- Taught at Harvard for 6 years,
 - frustrated with inadequacies of mathematical notation
 - Developed "Iverson Notation" in response
 - Published "A Programming Language" in 1962

ACM Turing award in 1979:

"For his pioneering effort in programming languages and mathematical notation resulting in what the computing field now knows as APL, for his contributions to the implementation of interactive systems, to educational uses of APL, and to programming language theory and practice."

Syntaxes of Mathematics

a b

 $Mat1 \cdot Mat2$

Problems:

- Wide variety of syntactical forms
- Strange and inconsistent precedence rules
- Things get worse when you deal with matrices

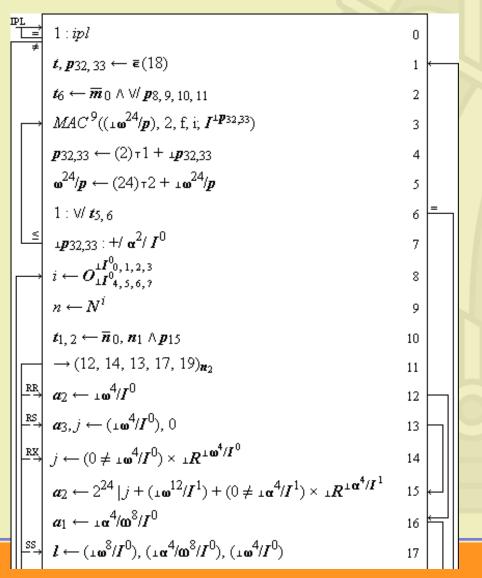
See http://www.jsoftware.com/papers/EvalOrder.htm

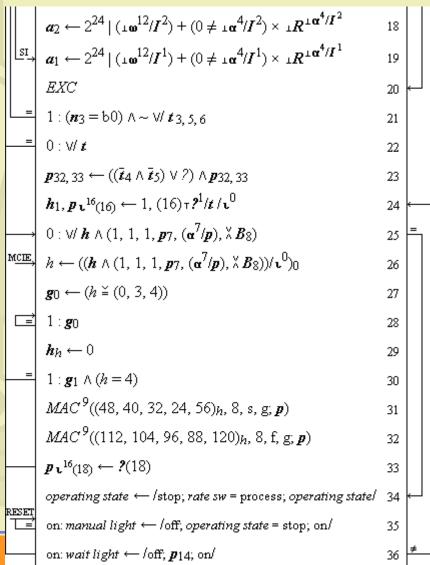
$$\sum_{n=1}^{\infty} 4n$$

$$\prod_{i=1}^{6} 4i$$



Iverson Notation: Description of IBM\360





A Programming Language

• The book, 1962

```
Quotient z \leftarrow x \div y z is the quotient of x and y

Absolute value z \leftarrow |x| z = x \times [(x > 0) - (x < 0)]

Floor k \leftarrow \lfloor x \rfloor k \le x < k + 1

Ceiling k \leftarrow \lceil x \rceil k \ge x > k - 1

j-Residue mod k \leftarrow k j i = kq + k; j \le k < j + k; and q is integral.
```

Linearization => APL\360

The 5: Ken Iverson, Adin Falkoff, Larry Breed,
 Dick Lathwell, Roger Moore.
 Operated by "Quaker Consensus".

```
Quotient z \leftarrow x \div y z is the quotient of x and y

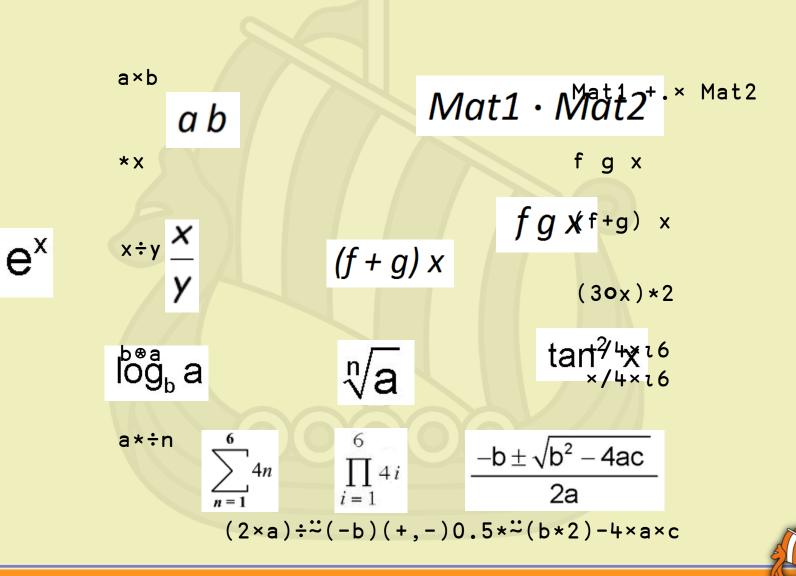
Absolute value z \leftarrow |x| z = x \times [(x > 0) - (x < 0)]

Floor k \leftarrow \lfloor x \rfloor k \le x < k + 1

Ceiling k \leftarrow \lceil x \rceil k \ge x > k - 1

j-Residue mod k \leftarrow k \mid_j i i = hq + k; j \le k < j + h; and q is integral.
```

(for Mathematics)



APL Was Born...

- The first running system started running in November of 1966.
- Let's do some exercises.



Exercises

- The folder "Exercises" contains numbered files, starting with "01-syntax.txt"
- Enter each expression (line)



Saving Your Work

- Historically, APL users have saved workspaces containing code and data as a single file
 - Similar to an Excel Spreadsheet
 - Takes a "snapshot of the VM"
 - Beware: also saves the execution stack if there is one
- Save your work using)save /path/mywsname[.dws]
- Load it again with) load /path/mywsname
- You can extract named objects from a workspace:
 -)copy /path/mywsname foo goo x y
- Saved workspaces can have a "latent expression"

 L X, which is executed when the workspace is loaded, unless you
 -)xload /path/mywsname



Saving Your Work, continued

- In the last few years, it has become more popular to use Unicode text files (and SVN/GIT), especially for source code
- You can save a fn/var, namespace or class using]save (]save is a "user command" - written in APL):

```
]save name /path/name[.dyalog]
(it is customary to use the same name for the file)
```

- You can bring it back into the active workspace using]load /path/name]load /path/*
- If you edit objects that were] loaded, the system will offer to update the file each time you make a change
- From version 15.0, the interpreter (editor) knows how to open and view source files without user commands.

Reading APL

$$CB \leftarrow \{\omega[1+(\rho\omega)|X\circ.+X\leftarrow(\iota\alpha)-1]\}$$

Imagine arguments:



Reading APL

What does this function do?

```
{(~R∈R∘.×R)/R←1↓ιω}
Or:
{{(~ω∈ω∘.×ω)/ω}1↓ιω}
```



Procedures / Tradfns

Monadic:

Dyadic:

```
∇ R←A MatMult B
[1] R←A+.×B
```

Niladic:



Procedures / Tradfns

"Ambi-valent" (+ use a control structure)

```
    ∇ R←{Window} Sum X
[1] :If O=□NC 'Window' ◇ R←+/X
[2] :Else ◇ R←Window +/ X
[3] :EndIf
```



Procedures / Tradfns

Name Elements of Right Argument

- + Local Variable
- + Class / DotNet declarations



Errors

```
1 2 3÷4 5
LENGTH ERROR
      1 2 3÷4 5
      □EN
5
      1÷0
DOMAIN ERROR: Divide by zero
      1÷0
      □EN (□EM 11)
    DOMAIN ERROR
 11
      DMX.Message
Divide by zero
```



Errors

- 1 WS FULL
- 2 SYNTAX ERROR
- 3 INDEX ERROR
- 4 RANK ERROR
- 5 LENGTH ERROR
- 6 VALUE ERROR
- 7 FORMAT ERROR
- 10 LIMIT ERROR
- 11 DOMAIN ERROR
- 12 HOLD ERROR
- 13 OPTION ERROR
- 15 LST FULL
- 16 NONCE ERROR
- 17 ACCESS ERROR

- 18 FILE TIE ERROR
- 19 FILE ACCESS ERROR
- 20 FILE INDEX ERROR
- 21 FILE FULL
- 22 FILE NAME ERROR
- 23 FILE DAMAGED
- 24 FILE TIED
- 25 FILE TIED REMOTELY
- 26 FILE SYSTEM ERROR
- 28 FILE SYSTEM NOT AVAILABLE
- 30 FILE SYSTEM TIES USED UP
- 31 FILE TIE QUOTA USED UP
- 32 FILE NAME QUOTA USED UP
- 34 FILE SYSTEM NO SPACE
- 35 FILE ACCESS ERROR CONVERTING FILE
- 36 INCOMPATIBLE ARRAY
- 38 FILE COMPONENT DAMAGED



Error Trapping: Dfns

```
div+{0::'Something Else is Wrong'
     11::0 A DOMAIN error: return 0
     α÷ω}
     3 div 0
0
     1 2 3 div 4 5
Something Else is Wrong
```



Error Trapping: Tradfns

Using: Trap

```
∇ R←A DIV B

[1] :Trap 0

[2] R←A÷B

[3] :Case 11 ♦ R←O A DOMAIN error

[4] :Else ♦ R←'Something Else is Wrong'

[5] :EndTrap

∇
```



Error Trapping: Tradfns

Using **TRAP**

