APL Programming Language: Introduction

Alexey Veretennikov
alexey@veroveli.com

Veroveli AB

January 11, 2017

APL Invented by Kenneth E. Iverson (1920-2004)

APL Programming Language

- Invented by Ken Iverson in 1960s, first working version: 1966, by Larry Breed, IBM Research
- Stands for "A Programming Language" by the name of the book
- "Widely known" by its use of special charaters, for example[1]:

```
{(⌽∨\⌽⍵≠' ')/⍵}
```

- Ken Iverson received the Turing Award in 1979 for this language
- 51 years now!

[1] Remove trailing blanks from the string

Examples

APL phrase	Description
`(~∧\' '=S)/S`	Remove leading blanks from string *S*
`(S≠' ')⊂S`	Split string *S*
`A,A+(×B-A)×⍳|B-A`	List of integers from *A* to *B*
`(~R∈R∘.×R)/R←1↓⍳R`	Find all prime numbers from *1* to *R*
`(+/÷≢)R`	Average value of the numeric array R
`+/(V⍳V)=⍳⍴V`	Count unique elements in vector V
`+/(⊂W)∘.≡V`	Occurences of word W in list V
`+\(S='(')-0,¯1↓S=')'`	Depth of parenthesis in string S

Properties Of The Language

- Array-first programming language
- Discouraged use of explicit loops and control structures
- Interactive and workspace - oriented
- Modern dialects (Dyalog, J, IBM APL2) support OOP and FP
- Use special characters or combinations as a functions and operators
- Most APL characters have 1 or 2 meanings
- Extremely concise

Major Domain Fields

- Financial industry, bank sector
- Insurance industry
- Big data analysis
- Recreational programming and code contests

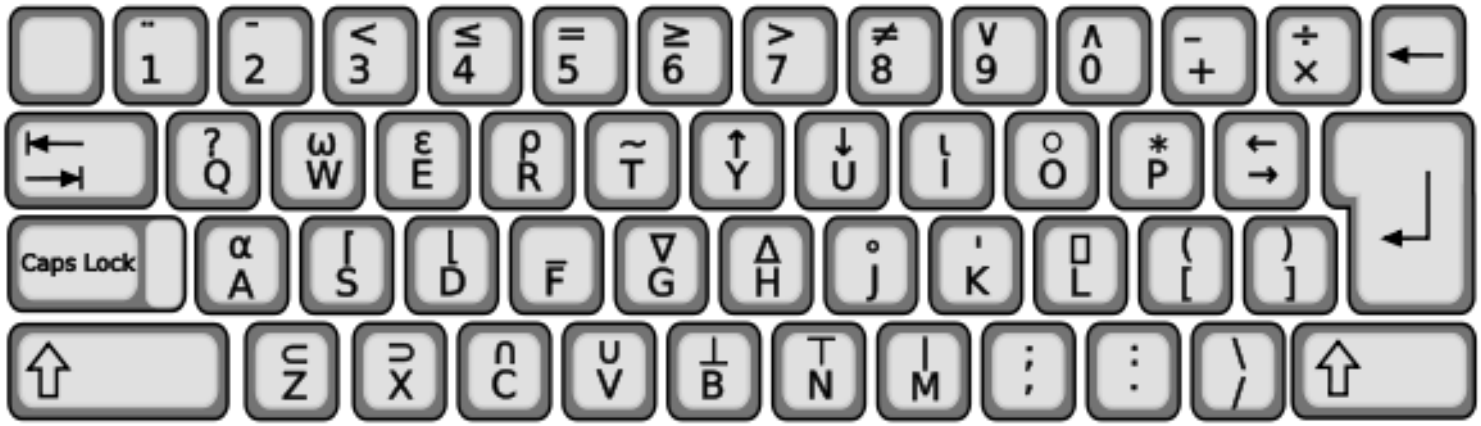
Array-Oriented Languages Family

- A+ - Free, APL extension
- J - Free, dual license, ASCII characters
- ELI - Free, educational, APL extension
- Q'Nial - Free, English words
- K and Q(English-words wrapper around K) - Commercial, used with KDB+
- MATLAB influenced by APL - Commercial, mathematics

Current Implementations Of APL

- Dyalog APL - commercial, free personal version available; actively developed, lots of features
- GNU APL - free, actively developed
- NARS2000 - free, Windows-only, actively developed
- APL.js - free, JavaScript-based
- APL2000 - commercial
- IBM APL2 still selling, no updates from 200x

Alphabet And Input Methods



- Windows:
  - Dyalog IME layout
  - Dyalog RIDE prefix keys
- MacOSX and Linux:
  - APL Keyboard Layout
  - GNU Emacs mode for GNU APL
  - Dyalog RIDE prefix keys
  - APL Keyboard Translator

Simple operations 1

Ordinary operations

```
      1 + 2
3
      10 × 1 2 3
10 20 30
      10 20 30 ÷ 2 3 4
5 6.666666667 7.5
```

Assignments

```
      a ← 10 20 30
      a+1
11 21 31
      a ← b ← c ← ¯1
      a b c + 2 4 6
1 3 5
```

Simple operations 2

Priority/Order of operations

```
      2×3+4
14
     (2×3)+4
10
```

Unary operations

```
      a ← 2
      ÷a
0.5
      ⌊3.5 0.5 2.1
3 0 2
      ~ 1 0 1 1
0 1 0 0
```

N-adic functions

Definition

- Function without arguments called *niladic* function
- Function with one argument (to the right) is *monadic* function
- Function with two argument (left and right) is *dyadic* function

Example (Niladic function - current time)

```
      ⎕TS
2017 1 9 21 22 16 865
```

Example (Monadic and dyadic functions: ÷)

```
      3÷2
1.5
      ÷2
0.5
```

Iota(ι)

Monadic Iota (ι) - Index Generator

```
      ⍳4
1 2 3 4
      ¯1+⍳4
0 1 2 3
```

Dyadic Iota (ι) - index of

```
      'aa' 'bb' 'cc' ⍳ 'bb' 'cc'
2 3
      A←'aa' 'bb' 'cc' 'aa'
      A⍳A
1 2 3 1
      (A⍳A)≠⍳4
0 0 0 1
```

Rho(ρ) - Shape of and Reshape

Monadic ρ - Shape Of

```
      ρ'aa' 'bb' 'cc'
3
      ρ⍳5
5
```

Dyadic ρ - Reshape

```
      2 2ρ1 2 3 4
1 2
3 4
      2 2ρ⍳4
1 2
3 4
      3ρ0
0 0 0
```

Functions and operators: Replicate

"/" Replicate (could be used as a "mask")

```
      1 0 1/1 2 3
1 3
      1 1 2/⍳3
1 2 3 3
```

Let's find unique elements!

```
      A←'aa' 'bb' 'cc' 'aa'
      A⍳A
1 2 3 1
      (A⍳A)=⍳4
1 1 1 0
      ((A⍳A)=⍳4)/A
aa bb cc
```

Functions and operators: Reduce

Definition

Operators take function and produce function

Boring code

```
      10+20+30
60
      10×20×30
6000
```

"/" as an operator: Reduce

```
      +/10 20 30
60
      ×/10 20 30
6000
```

Reduce and Scan: the fun part

"\" scan operator: partial sums

```
      +/⍳4
10
      +\⍳4
1 3 6 10
```

Let's count and remove leading spaces!

```
      S←'  hello world'
      S=' '
1 1 0 0 0 0 0 1 0 0 0 0 0
      ∧\S=' '
1 1 0 0 0 0 0 0 0 0 0 0 0
      +/∧\S=' '
2
      (~∧\S=' ')/S
hello world
```

Inner and outer product

“∘.fun“ Outer product

```
      (⍳3)∘.×⍳3
1 2 3
2 4 6
3 6 9
      (⍳3)∘.≤⍳3
1 1 1
0 1 1
0 0 1
```

“fun1.fun2“ Inner product

```
      (⍳3)+.×10 20 30
140
      (1 2 3)∧.=⍳3
1
```

“∈“ - Membership

X “∈“ Memerbers of Y

```
      'hello'∈'el'
0 1 1 1 0
```

(List ∈ Where) / ⍳⍴List

```
      ('hello'∈'el')/⍳⍴'hello'
2 3 4
```

Some notable examples

Average

```
      V←12 15 21
      (+/V)÷⍴V
16
```

Definition (Drop "↓")

```
      1↓10 20 30
20 30
      2↓10 20 30
30
```

Prime numbers

```
    (~R∈R∘.×R)/R←1↓⍳R
```

References and bibliography

- TryAPL.org try APL online
- APL Wiki
- Mastering Dyalog APL free E-book
- IBM APL2 Programming: Guide
- IBM APL2 Programming: Language Reference
- APL2 at a Glance By James A. Brown, Sandra Pakin, and Raymond P. Polivka, 1988
- APL An Interactive Approach By Leonard Gilman and Allen J. Rose

The end

Questions?