APL
A Programming Language

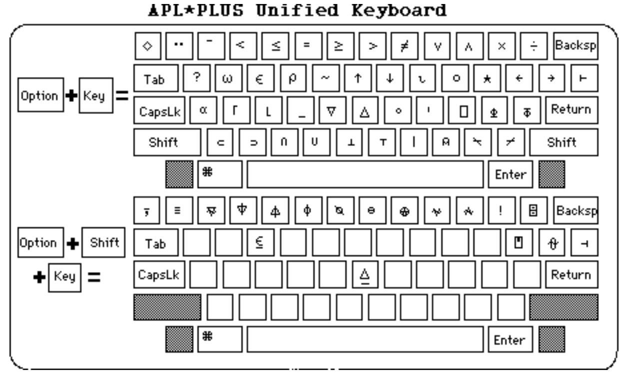
APL is

- a concise mathematical notation
- oriented towards manipulation of arrays
- an interactive computer language

APL symbols include many not found on ordinary keyboards.

These are typed by use of the "option" or "option-shift" combination, together with one of the regular keys:
For example, option + H = ∆

APL*PLUS Unified Keyboard



This keyboard diagram may be displayed at any time by selecting "APL keyboard" from the menu:

(Unfortunately, the command-shift-4 key combination, which usually prints the screen, will not print this diagram!)

At any time, the system is either in

- immediate execution mode
  (expressions entered will be evaluated and displayed)
- edit mode
  (used when entering or revising a function, for example)

Immediate execution mode

Example:

```
      36 × 5    ← You type this
180 ←────────── Computer displays this
```

(Note that what you type is automatically indented, while the computer displays the result at the left margin.)

Right-to-left evaluation:

Contrary to usual practice, in APL there is no hierarchy of operations;
Instead, all expressions are evaluated from right to left within parentheses.

```
      2 × 5 + 3    ←── same as 2 x (5+3)
16
      9 - 5 - 2    ←── same as 9 - (5-2)
6
      (9 - 5) - 2
2
```

APL statements are of 2 types:

- assignment statements

```
        Y ← 8
```

(note that "equals" is used for comparison, not for assignment!)

- branching statements

```
        → 5
```

(Here, 5 is a line number within a user-defined function)

Functions, whether primitive or user-defined, are either:

- *monadic* (single argument)

```
      ⌈2.47
3
```

(the "ceiling" function)

- *dyadic* (two arguments)

```
      3.15 ⌈ 2.47
3.15
```

(the "maximum" function)

(Note that two functions, one monadic and one dyadic, are represented by the same symbol, ⌈)

User-defined functions may also be defined to have NO arguments ("niladic")

Note that an argument may be a vector, i.e., $F(x_1, x_2, x_3, \ldots x_n)$ may be defined as a function of a *single* vector argument $X = (x_1, x_2, x_3, \ldots x_n)$

Reduction of vectors

The "slash" (/) preceded by a dyadic function, has the effect of inserting the dyadic function between each pair of consecutive entries of the vector which follows:

```
      + / 3 4 5
12
```

(equivalent to 3 + 4 + 5 *)*

That is, +/X is APL notation for $\sum_i X_i$

(Note that the "divide" function is ÷)

Other examples of reduction:

```
      X ← 5  2  4
      × / X
40
      ⌈ / X
5
      - / X
7
      ∨ / 0 0 0 1 0 1
1
```

(logical "or")

Further examples of reduction of a vector X:

- mean value of X
  `(+/X) ÷ ⍴X`
- variance of X
  `(+/(X-(+/X)÷⍴X)*2) ÷ ⍴X`
- number of elements which are even numbers
  `+/ X = 2×⌊X÷2`
- number of times that the largest element appears
  `+/ X = ⌈/X`

Compression of vectors:

The "slash" also is used to "compress" vectors, with a logical (zero-one) vector of the same length on the left and the vector on the right:

```
      X ← 5  2  8  4
      1 0 1 0 / X
5  8
      X < 5          ⟵ an expression with a logical value
0 1 0 1
      (X < 5)/X      ⟵ selects elements of X less than 5
2  4
```

Selection of elements of a vector, using compression:

```
V ← ?10⍴100            randomly generate vector
(V ≤ (+/V)÷10 )/V      elements ≤ average
(V = 2×⌊V÷2)/V         even-valued elements
X ← ?20⍴100            randomly generate X
(X ∊ V)/X              elements of X which are in V
```

Expansion of vectors

Similar to compression, but lengthens the vector, inserting zeroes (if numeric vector) or blanks (if character vector) where indicated:

```
      X ← 5  2  8  4
      1 1 1 0 1 \ X
5  2  8  0  4
```

↖ "backward slash"

(Number of 1's in the logical vector must equal the length of the vector on the right!)

Generalized Inner Products

The usual inner product of 2 vectors combines the operations of multiplication and addition:

$U \bullet V = U_1 x V_1 + U_2 x V_2 + \ldots + U_n x V_n$

This could be expressed in APL as the "plus" reduction of the element-by-element product of U and V: `+ / U × V`

An alternate, equivalent notation is

```
U +.× V
```

Instead of "+" and "x", any two dyadic functions which operate on scalars will do.
For example,

```
      X ← 8  5  2
      X +.* 1  2  3
41
      Y ← 4  6  3
      X ^.≥ Y
0
      X ⌈.+ Y
12
```

i.e., $8^1 + 5^2 + 2^3$
(is the exponention function)*

← *(checking whether X dominates Y)*
(^ is logical "and")

Generalized Outer Product

If U is a vector of length m,
V is a vector of length n,
and ● is any scalar dyadic function,
then the OUTER PRODUCT U ∘.● V is an mxn matrix, whose element in row i, column j is U[i]●V[j]

(∘ is the "null" symbol, obtained by pressing simultaneously the keys [option] + [J] *)*

Example of generalized outer product:

```
      U ←  5  2  8
      V ←  1  0  3  7
      U ∘.× V
5   0  15  35
2   0   6  14
8   0  24  56
```

← *for example, U[2]xV[4]*

Creating identity & lower-triangular matrices:

```
      (⍳3) ∘.= (⍳3)
1  0  0
0  1  0
0  0  1
      (⍳3) ∘.≥ (⍳3)
1  0  0
1  1  0
1  1  1
```

Computing a Euclidean distance matrix:
Let X[i] and Y[i] be the (x,y) coordinates of point #i

```
      X ← 10  30  50
      Y ← 40  50  20
      ( ((X∘.-X)*2) + (Y∘.-Y)*2 )*0.5
0        22.361  44.721
22.361   0       36.056
44.721   36.056  0
```

i.e., $\sqrt{(X_i-X_j)^2+(Y_i-Y_j)^2}$ is in row #i, column #j

Creating Arrays

Arrays may be created by using the "reshape" function, ⍴ .
The left argument of ⍴ is the shape desired, and the right argument is scalar or array to be reshaped. (The result is created, row by row, from the right argument:

```
      A ← 2 4 ⍴ 1 2 3 4 5 6
      A
1  2  3  4
5  6  1  2
```

↖ *The character ⍴ is obtained by* [option] + [R]

Subscripting Arrays

Subscripts are written within square brackes [ ]
Row subscripts and column subscripts are separated by a semicolon

- A[2;3] is the element in row #2, column #3
- A[1 ;] is row #1 of the array
- A[; 3] is column #3 of the array
- A[1 3;] are rows #1&3 of the array
- A[1 ; 2 3] are the elements in row 1 and columns 2&3, i.e., the 1x2 array [A_{12} A_{13}]

Examples of subscripting:

```
      A ← 2 4 ρ 1 2 3 4 5 6 7 8
      A
1  2  3  4
5  6  7  8
      A[1;2 4]
2  4
      A[ ; 3 4]
3  4
7  8
```

Indexing a vector by an vector:

Let V be a vector, and K a vector of indices. Then
V[K] is the vector
V[K[1]], V[K[2]], etc.
For example:

```
    V ← 111  222  333  444
    K ← 3  1  1  2  1  3
    V[K]
333  111  111  222  111  333
```

Indexing a vector by a matrix:

Let V be a vector and A a matrix of subscripts.
Then V[A] is a matrix whose size is that of A, and whose entry in location [I;J] is V[A[I;J]]
For example:

```
   V ← 111  222  333  444
   ⎕ ← A ← 2 3 ρ 2 3 3 1 3 2
2  3  3
1  3  2
  V[A]
222  333  333
111  333  222
```

Example: "Picture" of a matrix

```
      A
2  0  1  0
0  1  0  0
0  5  0  3
      1+A ≠ 0
2  1  2  1
1  2  1  1
1  2  1  2
      (' ★')[1+A ≠ 0]
★ ★
 ★
 ★ ★
```

Reduction of matrices:

```
     ⎕← A←2 3 ρ ⍳6
1 2 3
4 5 6
     +/[1]A                 ← axis operator
5  7  9          equivalent to  +⌿X
     +/[2]A
6  15            equivalent to  +/X
```

Example Given a matrix A of numbers, delete the rows which are all zeroes.

```
    A
3   2   1   0
4   0   0   1
0   0   0   0
2   0   1   3
    A∨.≠0
1   1   0   1
    (A ∨.≠ 0)/[1] A
```

Example Given a character matrix with a word in each row, locate the row containing a given word.

```
   NAMES
JACK
JOHN
MARY
   NAMES ∧.= 'JOHN'
0  1  0
   (NAMES ∧.= 'JOHN')⍳1      ← index
2
```

Defining functions

Type the symbol ∇ ("del") followed by the function header (specifying number of arguments and whether result is returned)

Example: A function to evaluate the mean of a vector

```
∇z←MEAN X
```

Both z and X are "dummy" variable names which will be "locally" defined within the function.

When you press the "return" key after typing the function header, you will enter "edit" mode, and you may then type the function definition:

```
    ∇z←MEAN X
 [1] z←(+/X)÷ρX
```

After the function definition is complete, select "exit editor" from the "Edit" menu.
You may then use the function as you would a primitive function:

```
   MEAN  2 4 1 0
1.75
```

The variable V is a vector of integers.
Write an expression for:

a) the maximum value
b) the minimum value
c) the range of values
d) the mean value
e) the median value
f) the number of elements exceeding the mean
g) the variance around the mean
h) the standard deviation
i) the number of times the largest number appears
j) the number of components which are odd numbers

CS is a character string.

Write an expression for determining how many times each of the vowels (A,E,I,O,U,Y) appear in the character string.

Write an expression for the number of double letters which appear in the character string. (For example, "NO BOOKKEEPERS ALLOWED" contains 4 instances of double letters.)

```
     T ← 1 0 2
      U
 0  1  2
 1  2  0
 2  0  1
      V
 0  ¯1  ¯1
 1   0  ¯1
 2   1   0
```

What are the values of:

a) T +.x T
b) T ∧.= V
c) T ∧.> V
d) U +.x T
e) V ∨.= U

DeMorgan has shown that π can be computed by the following alternating series:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \ \ldots$$

$$\frac{\pi - 3}{4} = \frac{1}{2x3x4} - \frac{1}{4x5x6} + \frac{1}{6x7x8} - \ \ldots$$

$$\frac{\pi}{6} = \sqrt{1/3}\left(1 - \frac{1}{3^1 x3} + \frac{1}{3^2 x5} - \frac{1}{3^3 x7} + \frac{1}{3^4 x9} - \ldots\right)$$

Write APL expressions for estimating π using the first N terms of the series.

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \ \ldots$$

Approximation of π by first N terms of series:

```
PI ← 4 x -/÷ ¯1 + 2xιN
```

$$\frac{\pi - 3}{4} = \frac{1}{2x3x4} - \frac{1}{4x5x6} + \frac{1}{6x7x8} - \ \ldots$$

$$\pi = 3 + 4\left(\frac{1}{2x3x4} - \frac{1}{4x5x6} + \frac{1}{6x7x8} - \ \ldots\right)$$

Approximation of π by first N terms of series:

```
PI ←3+4x -/÷(2 x ιN) x (1 + 2 x ιN) x 2 + 2 x ιN
```

$$\frac{\pi}{6} = \sqrt{1/3}\left(1 - \frac{1}{3^1 \times 3} + \frac{1}{3^2 \times 5} - \frac{1}{3^3 \times 7} + \frac{1}{3^4 \times 9} - \dots\right)$$

$$\pi = 6\sqrt{1/3}\left(1 - \frac{1}{3^1 \times 3} + \frac{1}{3^2 \times 5} - \frac{1}{3^3 \times 7} + \frac{1}{3^4 \times 9} - \dots\right)$$

Approximation of π by first N terms of series:

```
PI ← 6 × (÷3*.5)× -/÷(3*¯1+⍳N)×-1+2×⍳N
```

Write functions C_to_F *and* F_to_C *which will convert Celsius temperatures to Fahrenheit temperatures, and vice versa, respectively.*

The vector V contains a list of a student's homework scores.

Write an expression for

a) the scores sorted into descending order

b) the average score, after dropping the 4 lowest scores