

APL

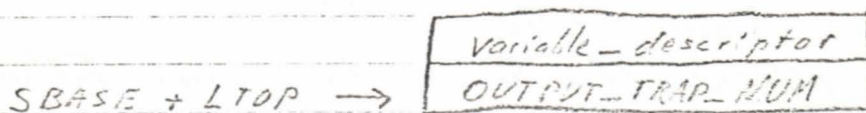
FREE FORMAT OUTPUT

646

The output routines should assume that the descriptors for WIDTH, DIGITS, and IO_BOXES are at APL data segment words +8, +9, and +10 respectively.

The output routines must be cognizant of the current horizontal position of the terminal cursor. This position must never be greater than the value of WIDTH. Wrap-around lines are indented six spaces. The significance of WIDTH and DIGITS may be different than what this paper outlines when IO_BOXES[2] is connected to something other than a terminal.

The APL stack looks like this after the APL output attention traps:



where OUTPUT-TRAP-NUM $\in \{ \text{QUAD-TRAP}, \text{QUOTEQUAD-TRAP} \}$

Output must begin on a new line whenever the variable-descriptor references a value of rank 2 or more. Output must end on a new line for QUAD-type output. Otherwise, QUAD and QUOTEQUAD are handled by common logic.

This common logic checks: that WIDTH is between 30 and the maximum linesize for the device; that DIGITS is between 1 and 7; that IO_BOXES[2] is attached to an output mailbox; that WIDTH, DIGITS & ^{IO_BOXES[2]} contain one element integers; and that IO_BOXES is a vector of length 2. If any of these conditions are violated, generate an IMPLICIT ERROR.

Conversion Rules

1. General:

- lines never break in the middle of a number
- the character element CRLF generates both a CR and a LF.
- integers & integral reals are treated alike.
- for arrays of dimension $d_1 d_2 d_3 \dots d_n$, insert a CRLF after every d_n elements, and a LF after every $d_{n-1} \times d_n$ elements, and another LF after every $d_{n-2} \times d_{n-1} \times d_n$ elements, ...

2. SCALARS & VECTORS:

- two blanks separate numbers from adjacent numbers; no blanks separate chars from numbers, and chars from chars.
- no blanks precede the first element nor follow the last element
- numbers are accurate to DIGITS significant figures.
- trailing zeros & decimal points are suppressed wherever possible.
- all numbers
 - 1) less than $1E^{-4}$ but greater than $-1E^{-4}$, except zero,
 - 2) not less than $1E<DIGITS>$, or
 - 3) not more than $-1E<DIGITS>$
 are expressed in E-notation. All other numbers are output in I or F-notation.
- numbers expressed in F-notation that are between -1 & 1 should have a zero to the left of the decimal point.

3. ARRAYS of rank > 1

- all numeric elements of an array are printed according to the same numeric format. No blanks separate numeric formats from themselves or chars.
- The numeric format is in E-notation if:
 - 1) any number satisfies (1), (2), or (3) above, or
 - 2) $\text{DIGITS} < 1 + \max_i \{ \text{EXP}_i \} - \min_i \{ \text{EXP}_i - \text{SIGDIG}_i \}$

where $i \in$ all numbers in array

$\text{EXP}_i =$ base 10 exponent of i^{th} element

$\text{SIGDIG}_i =$ # signif figures in i^{th} element

The E-notation specification takes up w print positions, where

$$w = 7 + \text{DIGITS} - \begin{cases} 1 & \text{if } \text{DIGITS} = 1 \\ 0 & \text{otherwise} \end{cases}$$

If, for example, $\text{DIGITS} = 4$, format = ' $\text{E}^{\pm}\text{X.XXXE}nnn$ ' where nnn is left justified. Trailing zeroes past the decimal point is allowed.

- if condition for E-notation not met, use Fw.d (d=0 $\Rightarrow$ don't print decimal point (Iv))

where $d = \max(0, 1 - \max_i \{ \text{EXP}_i - \text{SIGDIG}_i \})$

$$w = 3 + \max(1, \max_i \{ \text{EXP}_i \}) + \begin{cases} 0 & \text{if } d = 0 \\ d-1 & \text{if } d > 0 \end{cases}$$

So, if $w = 6$ & $d = 3$, format = ' $nn.ddd$ ' where nn is right justified. F-format #'s between 7 and 1 should have a zero to the left of the decimal point (except 0, which has no decimal point).

An error can also occur in the attempted execution of the APL expression represented by the argument of $\uparrow$; such an indirect error is reported by the error type prefaced by the symbol $\uparrow$ and followed by the character string and the caret marking the point of difficulty. For example:

```

 $\uparrow$  '4÷0'
 $\uparrow$  DOMAIN ERROR
    4÷0
    ^
 $\uparrow$  )WSID'
 $\uparrow$  VALUE ERROR
    )WSID
    ^

```

BOUND-FORMAT OUTPUT

FORMAT

The symbol ∇ (τ and $\circ$ overstruck) denotes two format functions which convert numerical arrays to character arrays. There are several significant uses of these functions in addition to the obvious one for composing tabular output. For example, the use of format is complementary to the use of execute in treating bulk input and output (via the shared variable facility), and in the management of combined alphabetic and numeric data.

The monadic format function produces a character array identical to the printing normally produced by its argument, but makes this result explicitly available. For example:

```

M←2=?4 4p2
R← $\nabla$ M
M
0 1 0 1
0 0 1 1
1 0 1 1
0 0 1 1
 $\rho$ M
4 4

R
0 1 0 1
0 0 1 1
1 0 1 1
0 0 1 1
 $\rho$ R
4 8

R[;2×14]
0101
0011
1011
0011

 $\rho$  $\nabla$ 2 5
3
 $\wedge$ /,R= $\nabla$ R
1
 $\nabla$ 'ABCD'
ABCD

```

The format function applied to a character array yields the array unchanged, as illustrated by the last two examples above. For a numerical array, the shape of the result is the same as the shape of the argument except for the required expansion along the last coordinate, each number going, in general, to several characters. The format of a scalar number is always a vector.

The dyadic format function accepts only numerical arrays as its right argument, and uses variations in the left argument to provide progressively more detailed control over the result. Thus, for $F \nabla A$, the argument F may be a single number, a pair of numbers, or a vector of length $2 \times 1 \div 1, \rho A$.

In general, a pair of numbers is used to control the result: the first determines the total width of a number field, and the second sets the precision. For decimal form the precision is specified as the number of digits to the right of the decimal point, and for scaled form it is specified as the number of digits in the multiplier. The form to be used is determined by the sign of the precision indicator, negative numbers indicating scaled form. Thus:

$\rho \square \leftarrow A$		$\rho \square \leftarrow 12 \ 3 \nabla A$	
12.34	-34.567	12.340	-34.567
0	12	0.000	12.000
-0.26	-123.45	-0.260	-123.450
3 2		3 24	
$R \leftarrow 9 \ 2 \nabla A$		$\rho \square \leftarrow 6 \ 0 \nabla A$	
$S \leftarrow 9 \ -2 \nabla A$		12 -35	
$\rho \square \leftarrow R$		0 12	
12.34 -34.57		0 -123	
0.00 12.00			
-0.26 -123.45			
3 18		3 12	
$\rho \square \leftarrow S$		$\rho \square \leftarrow 7 \ -1 \nabla A$	
1.2E01 -3.5E01		1E01 -3E01	
0.0E00 1.2E01		0E00 1E01	
-2.6E-01 -1.2E02		-3E-01 -1E02	
3 18		3 14	

If the width indicator of the control pair is zero, a field width is chosen such that at least one space will be left between adjacent numbers. If only a single control number is used, it is treated like a number pair with a width indicator of zero:

```

      p[+2vA
12.34 -34.57
   0.00  12.00
  -0.26 -123.45
3 16

```

```

      p[+^-2vA
1.2E01 -3.5E01
   0.0E00  1.2E01
  -2.6E-01 -1.2E02
3 18

```

```

      p[+0 2vA
12.34 -34.57
   0.00  12.00
  -0.26 -123.45
3 16

```

```

      p[+0 ^2vA
1.2E01 -3.5E01
   0.0E00  1.2E01
  -2.6E-01 -1.2E02
3 18

```

Each column of an array can be individually composed by a left argument that has a control pair for each:

```

      p[+0 2 0 2vA
12.34 -34.57
   0.00  12.00
  -0.26 -123.45
3 15

```

```

      p[+8 3 0 2vA
12.340 -34.57
   0.000  12.00
  -0.260 -123.45
3 16

```

```

      p[+6 2 12 ^3vA
12.34 -3.46E01
   0.00  1.20E01
  -0.26 -1.23E02
3 18

```

```

      p[+8 0 0 ^2vA
12 -3.5E01
   0  1.2E01
   0 -1.2E02
3 17

```

```

      6 2 8 3 3 0 4 0 5 0 12 4v,A
12.34 -34.567 0 12 0 -123.4500

```

The format function applied to an array of rank greater than two applies to each of the planes defined by the last two coordinates. For example:

```

      L+2=?2 2 5p2
      L
1 1 0 0 1
1 1 1 0 1

1 0 0 1 0
0 0 0 0 0

```

```

      4 1vL
1.0 1.0 0.0 0.0 1.0
1.0 1.0 1.0 0.0 1.0

1.0 0.0 0.0 1.0 0.0
0.0 0.0 0.0 0.0 0.0

```

Tabular displays incorporating row and column headings, or other information between columns or rows, are easily

configured using the format function together with extended catenation. For example:

```

ROWHEADS←4 3p'JANAPRJUL OCT'
YEARS←71+15
TABLE←.001×-4E5+?4 5.8E5
(' ',[1]ROWHEADS),(209 0 YEARS),[1]9 2TABLE

```

	72	73	74	75	76
JAN	318.13	351.55	3.62	144.77	4.82
APR	327.41	341.00	92.69	331.05	28.44
JUL	359.93	216.16	299.71	150.77	103.64
OCT	180.33	310.86	154.94	10.62	276.79

shift 2 blanks leftward

It is no longer necessary to use "heterogeneous output" in order to conveniently combine literal statements with numerical results. For example:

X←17.34	M←'THE VALUE OF X IS '
'THE VALUE OF X IS ';X	X←25.4
THE VALUE OF X IS 17.34	M;X
'THE VALUE OF X IS ',X	THE VALUE OF X IS 25.4
THE VALUE OF X IS 17.34	(M),(X)
	THE VALUE OF X IS 25.4

There are obvious restrictions on the left argument of format, since the width of a field must be large enough to hold the requested form, and if the specified width is inadequate the result will be a DOMAIN error. However, the width need not provide open spaces between adjacent numbers. For example, boolean arrays can be tightly packed:

```

1 0 2=24 4p2
1001
0000
1101
0111

```

The following formal characteristics of the format function need not concern the general user, but may be of interest in certain applications:

The least width required to represent a column of numbers C for an indicated precision P is determined as $W←(V/C<0)+(~P<0~1)+(|P|)+(4,1/0,1+[10@|C+C=0|1+P>0])$. If the width indicator is zero, the width used is $1+W$.

The expressions $(M~A),N~B$ and $(M,N)~A,B$ are equivalent if M and N are full control vectors, that is, if $((pM)=2×~1+pA)^(pN)=2×~1+pB$. If $2=pM$, then $(M~A),M~B$ and $M~A,B$ are equivalent.