Università degli Studi di Pisa

Introduzione

APL / 360

CNUCE

A cura di V. Casarosa
del Centro Scientifico I. B. M. di Pisa

Pubblicazione a circolazione interna

Introduzione

all' APL / 360

IL SISTEMA APL/360

L'APL/360 è un sistema conversazionale introdotto inizialmente in via sperimentale all'interno dell'IBM. Dopo molti mesi di uso regolare esso aveva profondamente influenzato le abitudini dei Centri di ricerca. Forti utenti della programmazione in batch erano diventati utenti del sistema APL per sviluppare in linea i loro algoritmi. L'elaborazione di molti dati di laboratorio, fatta in precedenza con programmi in batch o con calcolatori da tavolo, veniva regolarmente eseguita presso terminali in loco utilizzando programmi scritti in APL. La preparazione di articoli e rapporti tecnici veniva spesso fatta da terminale, utilizzando una libreria di programmi APL per il trattamento di testi. Molte persone, in precedenza indifferenti o addirittura contrarie all'uso dei calcolatori erano diventate degli utenti regolari del sistema APL/360.

L'uso del sistema APL/360 è stato quindi esteso ad altre organizzazioni e presso università e scuole secondarie. I risultati ottenuti possono riassumersi come segue:

- sebbene l'APL sia molto facile da imparare, spesso sviluppa nel programmatore (anche non esperto) un interesse per un uso assai raffinato del linguaggio, a causa del suo potere analitico e della sua struttura matematica;
- le funzioni primitive sulle matrici lo rendono molto adatto per applicazioni scientifiche, trattamento di testi ed in generale per l'elaborazione di dati dal momento che matrici (di qualunque rango) e vettori sono molto usati in tutti questi campi;
- l'uso di uno strumento computazionale potente e facilmente accessibile può materialmente cambiare la qualità e l'orientamento di un corso accademico.

Caratteristiche del sistema

Il sistema APL/360 è basato sul linguaggio definito da K. E. Iverson nel libro "A Programming Language" (John Wiley, 1962). E' un sistema interpretativo in time-sharing costruito attorno al linguaggio di Iverson in modo da preservare l'integrità del linguaggio e in modo da funzionare con le seguenti caratteristiche principali:

- regole di sintassi semplici ed uniformi;
- uso dei simboli usuali per le comuni operazioni aritmetiche;
- ingresso decimale in formato libero;
- conversione automatica alla rappresentazione interna utilizzando, se necessario, una doppia precisione (16 cifre decimali);
- un vasto insieme di funzioni primitive;
- funzioni definite dall'utente che si possono usare con la stessa facilità di quelle primitive;
- tempi di risposta molto brevi;
- un modo di esecuzione immediata completamente libero da parole chiave;
- un vasto repertorio di comandi di sistema;
- concisi, ma chiari messaggi di errore.

Applicazioni

Poichè il sistema APL/360 è organizzato attorno al concetto di spazio di lavoro, esso si presta molto bene a suddividere i programmi in vari spazi di lavoro a seconda del loro campo di applicazione, dal momento che programmi e dati si muovono come un insieme unico quando lo spazio di lavoro viene memorizzato o attivato. Alcuni dei programmi applicativi che sono stati finora realizzati sono i seguenti:

- un insieme di funzioni per calcoli statistici;
- un insieme di programmi per la geometria a coordinate, compreso un programma per disegnare grafici;

- un sistema per il controllo della produttività;
- un insieme di programmi per comporre e scrivere testi;
- un manipolatore algebrico per la semplificazione di polinomi e matrici di espressioni algebriche;
- un insieme di programmi per applicazioni contabili;
- moltissimi programmi per l'analisi numerica di funzioni reali e complesse.

Attualmente il sistema APL/360 trova applicazione in moltissimi campi: da semplici calcoli numerici (tipo calcolatrice da tavolo) a studi sperimentali su problemi di fisica teorica e algebra astratta, dall'insegnamento nelle scuole elementari all'uso in corsi universitari.

IL LINGUAGGIO APL

Il sistema APL/360 esegue comandi di sistema oppure statement matematici, che vengono comunicati al sistema per mezzo di un terminale tipo telescrivente. Gli statement matematici accettati possono utilizzare funzioni primitive (ad. es. +,-) che sono già definite nel sistema oppure funzioni definite che vengono fornite dall'utente introducendo la loro definizione da terminale.

Gli statement possono essere di due tipi: statement di salto (che vedremo in seguito) e statement di assegnamento. Un tipico statement di assegnamento è della forma

```
X←3×4
```

Questo statement assegna alla variabile X il valore dell'espressione che è alla destra della freccia di assegnamento. Se si omettono il nome della variabile e la freccia, il sistema stampa il valore dell'espressione. Come si può vedere nell'appendice, la quale riporta alcuni esempi sull'uso del sistema APL/360, tutto quello che viene stampato dal sistema inizia dal margine sinistro, mentre quello che è

scritto dall'utente viene automaticamente spostato a destra di sei spazi.

Tutti i numeri introdotti dall'utente, oppure scritti dal sistema, sono in base 10, sia nella forma convenzionale (compreso il punto decimale, se necessario) che in quella esponenziale.

Un vettore di costanti può essere introdotto scrivendo le sue componenti nell'ordine, separate da uno o più spazi. Una costante alfanumerica viene introdotta battendo il carattere tra apostrofi, ed una sequenza di caratteri racchiusa da apostrofi viene considerata come un vettore alfanumerico le cui componenti sono i caratteri della sequenza.

Il tasto di 'backspace' serve soltanto a posizionare il carrello e non provoca alcuna cancellazione di caratteri. Esso può essere usato per tre scopi:

a) inserire caratteri mancanti, se era stato lasciato lo spazio per essi;
b) formare caratteri costituiti dalla sovrapposizione di due o più simboli;
c) posizionare il carrello per una cancellazione, la quale viene effettuata battendo il tasto 'linefeed'. Il tasto 'linefeed' cancella tutti i caratteri che si trovano a destra della posizione attuale del carrello.

La fine di uno statement viene indicata battendo il tasto 'ritorno carrello' e lo statement viene interpretato dal sistema esattamente come esso appare sul foglio, senza alcun riferimento all'ordine con cui i caratteri sono stati battuti.

In una espressione composta, come ad es. 3×4+6÷2 , le funzioni vengono eseguite da destra verso sinistra, senza alcuna priorità predefinita. Ogni funzione cioè prende come argomento destro il valore di tutta l'espressione che si trova alla sua destra (il valore dell'espressione precedente viene calcolato come 21). Usando la parentesi viene applica

ta la stessa regola, ma, come al solito, vengono prima calcolate le espressioni in parentesi.

Se, per una ragione qualsiasi, uno statement non può essere eseguito, il sistema scrive un messaggio di errore e, sulla riga successiva, lo statement sbagliato con una indicazione approssimativa del punto in cui è stato scoperto l'errore.

Funzioni scalari

Le funzioni primitive sono divise in funzioni scalari e funzioni miste. Le funzioni scalari sono definite su elementi singoli, cioè scalari, e vedremo tra breve come possono essere estese anche a matrici.

La tabella I mostra le funzioni scalari che esistono nel sistema. Ciascuna è definita su numeri reali, o cu un loro sottoinsieme, come nel caso delle funzioni logiche and, or, nand e nor. Non viene fatta alcuna distinzione fra numeri in virgola fissa o in virgola mobile, in quanto questa è principalmente una questione di rappresentazione interna. La precisione a cui arriva il sistema è di circa 16 cifre decimali.

Ciascuna delle funzioni della tabella I si può usare allo stesso modo delle note funzioni aritmetiche.
La maggior parte dei simboli può indicare sia una funzione monadica (che ha un solo argomento, a destra) che una funzione diadica (che ha due argomenti). Ogni volta che è presente un argomento a sinistra la funzione viene interpretata dal sistema come diadica.

Vettori e matrici

Il numero di indici necessari ad individuare un elemento di una matrice rappresenta il rango della matri

ce. Così ad es. un vettore è di rango 1, una matrice (nel suo significato più comune) è di rango 2 ed uno scalare è di rango 0.

Gli esempi che vedremo in seguito sono limitati a matrici di rango 1 e 2, ma il sistema accetta matrici di qualunque rango (compatibilmente con la memoria centrale a disposizione dell'utente).

Se X è un vettore e I è uno scalare, *X [I]* indica l'i-esimo elemento di X. Se M è una matrice di rango 2 e I e J sono scalari, *M[I;J]* indica l'elemento di M che si trova nella i-esima riga e j-esima colonna.

Più in generale, gli indici possono essere una qualsiasi espressione il cui valore sia uno scalare o una matrice. Ad es., se X è un vettore, *X [3 15]* rappresenta un vettore di 3 elementi che sono rispettivamente il terzo, il primo e il quinto elemento di X. Ancora, se T, P e Q sono matrici di rango 2, e se *R←T[P;Q]* allora R è una matrice di rango 4 e *R[I;J;K;L]* è uguale a *T[P[I;J];Q[K;L]]* L'espressione *M[I;]* rappresenta tutta la riga i-esima della matrice M di rango 2 e l'espressione *M[;J]* rappresenta tutta la colonna j-esima.

Le funzioni scalari della tabella I possono essere estese alle matrici in quattro modi: elemento per elemento, riduzione, prodotto interno e prodotto esterno.

Estensione elemento per elemento. Tutte le funzioni scalari della tabella I possono essere applicate a matrici aventi lo stesso rango e le stesse dimensioni, elemento per elemento. Se M e N sono matrici, ⊙ indica una qualunque funzione scalare, e *P←M⊙N* allora *P[I;J]* è uguale a *M[I;J]⊙N[I;J]*. Se i due argomenti non hanno lo stesso rango e le stesse dimensioni, viene dato un messaggio di errore, a meno che uno dei due non sia uno

scalare o una matrice di un solo elemento, nel qual caso l'elemento singolo viene applicato a tutti gli elementi dell'altro argomento.

Riduzione Se ⍟ indica una qualunque funzione diadica e X un vettore, la forma ⍟/X rappresenta la riduzione di X secondo ⍟ ed il suo valore è uno scalare dato da X[1]⍟X[2]⍟X[3]⍟X[4]..X[N] , dove N è il numero delle componenti di X. Più in generale, se M è una matrice di rango qualsiasi, ⍟/[I]M rappresenta la riduzione di M secondo ⍟ lungo l'i-esima coordinata, ed il suo valore è una matrice il cui rango è minore di uno di quello di M. Ad es., se M è una matrice di rango 3 e T←⍟/[2]M , allora T è una matrice di rango 2 e T[I;J] è uguale a M[I;1;J]⍟M[I;2;J]..M[I;N;J] , dove N e la lunghezza della seconda coordinata.

Poichè in sostanza la riduzione è una funzione monadica, essa può essere applicata ad argomenti di qualunque rango e dimensione.

Prodotto interno. Il prodotto interno di due matrici A e B di qualsiasi rango viene indicato con A⍟.⊠B dove ⍟ e ⊠ indicano due qualunque funzioni diadiche. Perchè il prodotto interno sia definito la dimensione dell'ultima coordinata di A deve essere uguale a quella della prima coordinata di B, a meno che uno dei due argomenti non sia uno scalare oppure una matrice di un solo elemento. La tabella II mostra la definizione del prodotto interno per matrici fino al rango 2. L'estensione a matrici di rango maggiore è immediata. Ad es., se A e B sono matrici di rango 3 e T←A⍟.⊠B , allora T è una matrice di rango 4 e T[I;J;K;L] è uguale a ⍟/A[I;J;]⊠B[;K;L]. L'ordinario prodotto di matrici di rango 2 (righe per colonne) viene indicato con A+.×B.

Prodotto esterno . Il prodotto esterno di due matrici A e B, di rango qualsiasi, rispetto a una funzione scalare diadica ⍟ viene indicato con A∘.⍟B e produce una matrice il cui rango è pari alla somma dei ranghi di A e di B, ed i cui elementi sono formati applicando ⍟ ad o-

gni coppia di elementi di A e di B. La tabella III mostra la definizione del prodotto esterno per matrici fino al rango 2. Anche in questo caso è immediata l'estensione a matrici di rango maggiore.

Funzioni miste

Al contrario delle funzioni scalari, definite soltanto per argomenti e per risultati scalari (eventualmente estendibili anche a matrici, come già visto) le funzioni miste illustrate nella tabella IV possono avere come argomenti vettori e dare come risultato scalari o vettori oppure, viceversa, possono avere argomenti scalari e dare risultati vettoriali.

Nell'estendere le definizioni della tabella IV a matrici di rango maggiore di 1, può talvolta risultare necessario indicare su quale coordinata della matrice si vuole applicare la funzione mista. L'espressione [*I*] che segue il simbolo di funzione indica che la funzione mista deve essere applicata lungo l'i-esima coordinata della matrice. Se questa indicazione viene omessa, la funzione viene sempre applicata all'ultima coordinata della matrice.

Definizione di funzioni

Una funzione L la quale ad es. calcoli la lunghezza della ipotenusa di un triangolo rettangolo i cui cateti hanno lunghezze X e Y, viene definita come segue:

```
      ∇ Z←X L Y
[1]   Z←((X*2)+Y*2)*.5
[2]   ∇
```

Il primo ∇(chiamato del) indica l'inizio della definizione ed il secondo ∇ indica la fine. I numeri di statement (in parentesi quadra) vengono scritti automaticamente

dal sistema. Durante la definizione gli statement non vengono eseguiti, nè viene controllata la loro validità. Una volta che una funzione è stata definita essa può venire usata come le altre funzioni standard ed in particolare può essere usata nella definizione di altre funzioni. E' possibile anche definire una funzione in modo ricorsivo utilizzando la funzione stessa nella sua definizione.

La prima riga di una funzione (header) stabilisce se la funzione deve avere un valore esplicito oppure no (presenza della freccia di assegnamento) e stabilisce inoltre il numero dei parametri. Questi possono essere due, come nell'esempio dato, oppure uno, oppure nessuno.

I nomi di variabili che compaiono nella prima riga di una definizione vengono considerati locali al programma definito.

Salti. In una funzione gli statement vengono normalmente eseguiti nell'ordine indicato dal numero di statement e l'esecuzione termina alla fine dell'ultimo statement. L'ordine di esecuzione può essere modificato con statement di salto.

L'espressione →4 indica ad es. un salto allo statement numero 4. In generale la freccia di salto può essere seguita da una qualsiasi espressione il cui valore sia un numero intero. Questo valore rappresenta il numero di statement che verrà eseguito dopo quello attuale. Un salto ad un numero di statement che non esiste determina la fine dell'esecuzione di una funzione.

Revisione di funzioni. Il numero di statement scritto automaticamente dal sistema durante la definizione di una funzione può essere sostituito battendo [*N*] dove N è un qualunque numero (intero o decimale) che può andare da 1 a 9999. Quando viene chiusa la definizione di una funzione

(battendo ∇) il sistema esegue le seguenti operazioni:

a) se un numero di statement è ripetuto viene considerata valida solo l'ultima versione;
b) se uno statement è vuoto, cioè se il numero di statement scritto dal sistema era stato seguito da un 'linefeed' ed un ritorno di carrello, lo statement viene cancellato;
c) gli statement vengono ordinati in base al loro numero e vengono numerati di nuovo con gli interi 1, 2, 3 ecc.

Risulta quindi molto facile eseguire sostituzioni, cancellazioni od inserimenti durante la definizione di una funzione. Durante la definizione di una funzione si può far scrivere al sistema uno degli statement già definiti battendo [*N*□]. Dopo aver scritto lo statement N il sistema va a capo, scrive [*N*] ed aspetta la nuova definizione. Battendo [□] il sistema scrive tutti gli statement della funzione, compreso il carattere ∇ all'inizio ed alla fine, e resta in attesa dello statement successivo.

In un qualunque momento dopo la fine della definizione di una funzione R, la definizione di R può essere riaperta battendo ∇*R*. Il sistema risponde scrivendo [*N*] essendo N-1 il numero totale di statement nella definizione attuale di R. A questo punto si possono eseguire sostituzioni, cancellazioni, inserimenti o aggiunte nel modo già visto.

Debugging di funzioni. L'esecuzione di una funzione può essere seguita per mezzo della traccia. Quando una funzione viene tracciata in modo completo, per ogni statement eseguito il sistema scrive il nome della funzione, il numero di statement ed il valore che è stato calcolato nello statement. La traccia può essere fatta anche in modo selettivo, in modo da tracciare soltanto gli statement desiderati. Il tracciamento di una funzione viene controllato dal vettore di traccia. Gli elementi del

vettore sono i numeri degli statement che si vogliono tracciare. L'assegnamento dei valori al vettore di traccia può essere fatto anche in modo dinamico, dalla funzione stessa che si desidera tracciare.

Ad ogni funzione è associato anche un vettore di arresto analogo a quello di traccia. Il sistema arresta l'esecuzione di una funzione immediatamente prima di ciascuno degli statement il cui numero compare nel vettore di arresto, e scrive il nome della funzione ed il numero di statement. Quando è in questo stato la funzione si dice sospesa. Durante la sospensione sono possibili tutte le normali attività ed è possibile anche, con certe restrizioni, riaprire la definizione di una qualunque funzione, compresa quella sospesa.

IL SISTEMA

Spazio di lavoro

L'unità base su cui opera il sistema APL/360 è lo spazio di lavoro. Quando è in uso uno spazio di lavoro si dice attivo ed è costituito da una zona della memoria centrale del calcolatore. Durante una seduta al terminale uno spazio di lavoro attivo è sempre associato al terminale e tutte le operazioni avvengono al suo interno. I nomi di variabili e di funzioni si riferiscono sempre ad oggetti esistenti nello spazio di lavoro attivo ed hanno validità globale nello spazio di lavoro (eccetto per quelle variabili dichiarate locali ad una funzione).

Questa organizzazione si presta molto bene alla suddivisione dei programmi in vari spazi di lavoro a seconda del loro campo di applicazione. E' poi possibile suddividere ulteriormente gli oggetti (funzioni e valori globali) all'interno di uno spazio di lavoro, riunendoli in gruppi.

Librerie

Gli spazi di lavoro che non sono attivi vengono conservati nella memoria esterna del calcolatore (dischi) e identificati con nomi arbitrari definiti dall'utente. A richiesta, uno spazio di lavoro può essere reso attivo copiandolo nella memoria centrale, oppure solo alcune informazioni possono essere copiate da uno spazio di lavoro che sta in libreria in uno spazio di lavoro attivo.

Le librerie private sono associate ad un utente e vengono identificate dal suo numero di codice. L'accesso ad una libreria privata da parte di altri utenti è limitato alla sola lettura in quanto essi non possono memorizzarvi spazi di lavoro, nè possono ottenere una lista degli spazi di lavoro che vi sono memorizzati. E' comunque possibile da parte di un utente attivare una copia di uno spazio di lavoro di un altro utente, se egli conosce il numero di libreria in cui è memorizzato ed il nome dello spazio di lavoro.

Le librerie pubbliche sono a disposizione di tutti gli utenti e vengono identificate da numeri inferiori a 1000. Ognuno può memorizzare uno spazio di lavoro in una libreria pubblica ed è possibile avere la lista degli spazi di lavoro in essa memorizzati. Comunque uno spazio di lavoro memorizzato in una libreria pubblica rimane sempre sotto il controllo dell'utente che ve lo ha messo e non può essere modificato dagli altri utenti.

I comandi del sistema

I comandi del sistema APL/360 vengono riconosciuti dal fatto che cominciano sempre per parentesi chiusa (costruzione sintatticamente non valida nel linguaggio APL) e, a seconda del loro effetto sullo stato del sistema, possono essere raggruppati in cinque classi:

1. controllo di terminale; agiscono sulla connessione tra il terminale ed il sistema;

2. controllo dello spazio di lavoro; influenzano lo stato dello spazio di lavoro attivo;
3. controllo delle librerie; influenzano lo stato delle librerie;
4. domande; forniscono informazioni senza alterare lo stato del sistema;
5. comunicazioni; trasmettono messaggi da un terminale all'altro.

Ogni statement che comincia con una parentesi chiusa viene intepretato dal sistema APL/360 come un tentativo di eseguire un comando. Quando il comando è eseguito con successo viene data la risposta normale. Se per una qualsiasi ragione il comando non può essere eseguito viene data una risposta di errore.

Vengono qui di seguito illustrati i principali comandi del sistema APL/360.

Controllo di terminale

) numero codice

Una volta che il terminale è stato connesso con il calcolatore, questo comando stabilisce il primo contatto fra l'utente e il sistema APL/360. Viene attivato uno spazio di lavoro pulito ed inizia il conteggio del tempo per l'addebito.

) *OFF*

Questo comando indica la fine di una seduta. Lo spazio di lavoro attivo viene distrutto, il terminale viene disconnesso e viene addebitato il tempo di occupazione (l'addebito distingue tra tempo di occupazione del terminale e tempo di occupazione dell'unità centrale).

Controllo dello spazio di lavoro

) *GROUP* lista di nomi

Il primo nome della lista diventa il nome di un gruppo i cui membri sono gli altri nomi della lista. I membri di un gruppo possono essere variabili globali, funzioni definite oppure altri gruppi.

) *ERASE* lista di nomi

I nomi che compaiono nella lista vengono cancellati dallo spazio attivo, rendendo così disponibile della memoria di lavoro. Se il nome è quello di un gruppo, tutti i membri del gruppo vengono cancellati.

) *LOAD* identificatore

Uno spazio di lavoro non attivo viene identificato da un numero di libreria e dal nome dello spazio di lavoro. Questo comando rende attivo (copia cioè nella memoria centrale) lo spazio di lavoro individuato dall'identificatore. Lo spazio di lavoro che era attivo prima di questo comando viene distrutto.

) *COPY* identificatore nome

L'oggetto individuato da nome (può essere una variabile globale, una funzione o un gruppo) viene preso dallo spazio di lavoro individuato dall'identificatore e copiato nello spazio di lavoro attivo. Se viene omesso il nome, tutti gli oggetti dello spazio di lavoro indicato vengono copiati nello spazio di lavoro attivo. Gli eventuali omonimi vengono rimpiazzati da quelli provenienti dallo spazio di lavoro non attivo.

Controllo delle librerie

) *SAVE* identificatore

Una copia dello spazio di lavoro attivo viene memorizzato nella libreria e col nome che sono indicati nell'identificatore.

) *DROP* identificatore

Lo spazio di lavoro non attivo individuato dall'identificatore viene cancellato dalla libreria e lo spazio libero viene reso disponibile. Questo comando non altera lo spazio di lavoro attivo.

Domande

) *FNS*

Vengono listati in ordine alfabetico i nomi di tutte le funzioni definite nello spazio di lavoro attivo.

) *VARS*

Vengono listati in ordine alfabetico i nomi di tutte le variabili globali che esistono nello spazio di lavoro attivo.

) *GRPS*

Vengono listati in ordine alfabetico i nomi di tutti i gruppi che esistono nello spazio di lavoro attivo.

) *GRP* nome

Vengono listati i membri del gruppo il cui nome è stato specificato nel comando.

) *LIB* numero

Vengono listati i nomi degli spazi di lavoro memorizzati nella libreria pubblica indicata dal numero.

) *PORTS*

Viene data una lista dei terminali attivi (ognuno identificato da un numero) insieme al numero di codice degli utenti che vi stanno lavorando.

Comunicazioni

) *MSGN* numero testo

Il testo scritto nel comando viene inviato e stampato al terminale indicato dal numero, insieme al numero del terminale del mittente.

) *OPRN* testo

Viene inviato un messaggio al terminale dell'operatore del sistema APL/360.

o - o - o - o - o - o - o - o - o - o - o

Queste brevi note costituiscono semplicemente un'introduzione al sistema APL/360.
Chi fosse interessato ad approfondire l'argomento può riferirsi alle seguenti pubblicazioni:

P.C. Berry , APL/360 Primer , IBM Corporation, 1968

L.M. Breed e R.H. Lathwell, APL/360 , IBM, Contributed Program Library, N' 360D-03.3.007, 1968

S. Pakin, APL/360 Reference Manual, Science Research Associates, 1968.

Monadic form fB		f	Dyadic form AfB	
Definition or example	Name		Name	Definition or example
$+B \leftrightarrow 0+B$	Plus	+	Plus	$2+3.2 \leftrightarrow 5.2$
$-B \leftrightarrow 0-B$	Negative	-	Minus	$2-3.2 \leftrightarrow {}^{-}1.2$
$\times B \leftrightarrow (B>0)-(B<0)$	Signum	×	Times	$2\times 3.2 \leftrightarrow 6.4$
$\div B \leftrightarrow 1\div B$	Reciprocal	÷	Divide	$2\div 3.2 \leftrightarrow 0.625$
B | $\lceil B$ | $\lfloor B$ 3.14 | 4 | 3 ${}^{-}3.14$ | ${}^{-}3$ | ${}^{-}4$	Ceiling Floor	⌈	Maximum	$3\lceil 7 \leftrightarrow 7$
		⌊	Minimum	$3\lfloor 7 \leftrightarrow 3$
$\star B \leftrightarrow (2.71828..)\star B$	Exponential	⋆	Power	$2\star 3 \leftrightarrow 8$
$\circledast\star N \leftrightarrow N \leftrightarrow \star\circledast N$	Natural logarithm	⍟	Logarithm	$A\circledast B \leftrightarrow$ Log B base A $A\circledast B \leftrightarrow (\circledast B)\div\circledast A$
$\lvert {}^{-}3.14 \leftrightarrow 3.14$	Magnitude	|	Residue	Case | $A\lvert B$ $A\neq 0$ | $B-(\lvert A)\times\lfloor B\div\lvert A$ $A=0, B\geq 0$ | B $A=0, B<0$ | Domain error
$!0 \leftrightarrow 1$ $!B \leftrightarrow B\times !B-1$ or $!B \leftrightarrow$ Gamma$(B+1)$	Factorial	!	Binomial coefficient	$A!B \leftrightarrow (!B)\div(!A)\times !B-A$ $2!5 \leftrightarrow 10 \quad 3!5 \leftrightarrow 10$
$?B \leftrightarrow$ Random choice from ιB	Roll	?	Deal	A Mixed Function (See Table 3.8)
$\circ B \leftrightarrow B\times 3.14159...$	Pi times	○	Circular	See Table at left
$\sim 1 \leftrightarrow 0 \quad \sim 0 \leftrightarrow 1$	Not	~		
		∧	And	
		∨	Or	
		⍲	Nand	
		⍱	Nor	
		<	Less	
		≤	Not greater	
		=	Equal	
		≥	Not less	
		>	Greater	
		≠	Not Equal	

A	B	$A\wedge B$	$A\vee B$	$A\barwedge B$	$A\barvee B$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	1	0	0

Relations
Result is 1 if the relation holds, 0 if it does not:
$3\leq 7 \leftrightarrow 1$
$7\leq 3 \leftrightarrow 0$

$(-A)\circ B$	A	$A\circ B$
$(1-B\star 2)\star .5$	0	$(1-B\star 2)\star .5$
Arcsin B	1	Sine B
Arccos B	2	Cosine B
Arctan B	3	Tangent B
$({}^{-}1+B\star 2)\star .5$	4	$(1+B\star 2)\star .5$
Arcsinh B	5	Sinh B
Arccosh B	6	Cosh B
Arctanh B	7	Tanh B

Table of Dyadic ○ Functions

Tabella I - Funzioni scalari primitive

Rango di A	Rango di B	Rango di $A \circledcirc.\boxtimes B$	Definizione $Z \leftarrow A \circledcirc.\boxtimes B$
0	0	0	$Z \leftarrow \circledcirc/A \boxtimes B$
0	1	0	$Z \leftarrow \circledcirc/A \boxtimes B$
1	0	0	$Z \leftarrow \circledcirc/A \boxtimes B$
1	1	0	$Z \leftarrow \circledcirc/A \boxtimes B$
0	2	1	$Z[I] \leftarrow \circledcirc/A \boxtimes B[;I]$
2	0	1	$Z[I] \leftarrow \circledcirc/A[I;] \boxtimes B$
1	2	1	$Z[I] \leftarrow \circledcirc/A \boxtimes B[;I]$
2	1	1	$Z[I] \leftarrow \circledcirc/A[I;] \boxtimes B$
2	2	2	$Z[I;J] \leftarrow \circledcirc/A[I;] \boxtimes B[;J]$
.			

Tabella II - Prodotto interno

Rango di A	Rango di B	Rango di $A\circ.\circledcirc B$	Definizione $Z\leftarrow A\circ.\circledcirc B$
0	0	0	$Z\leftarrow A\circledcirc B$
0	1	1	$Z[I]\leftarrow A\circledcirc B[I]$
1	0	1	$Z[I]\leftarrow A[I]\circledcirc B$
1	1	2	$Z[I;J]\leftarrow A[I]\circledcirc B[J]$
0	2	2	$Z[I;J]\leftarrow A\circledcirc B[I;J]$
2	0	2	$Z[I;J]\leftarrow A[I;J]\circledcirc B$
1	2	3	$Z[I;J;K]\leftarrow A[I]\circledcirc B[J;K]$
2	1	3	$Z[I;J;K]\leftarrow A[I;J]\circledcirc B[K]$
2	2	4	$Z[I;J;K;L]\leftarrow A[I;J]\circledcirc B[K;L]$

Tabella III - Prodotto esterno

Name	Sign[1]	Definition or example[2]
Size	ρA	$\rho P \leftrightarrow 4$ $\quad \rho E \leftrightarrow 3\ 4$ $\quad \rho 5 \leftrightarrow \iota 0$
Reshape	$V\rho A$	Reshape A to dimension V $\quad 3\ 4\rho\iota 12 \leftrightarrow E$
Ravel	$,A$	$12\rho E \leftrightarrow \iota 12$ $\quad 0\rho E \leftrightarrow \iota 0$ $,A \leftrightarrow (\times/\rho A)\rho A$ $\quad ,E \leftrightarrow \iota 12$ $\quad \rho,5 \leftrightarrow 1$
Catenate	V,V	$P,\iota 2 \leftrightarrow 2\ 3\ 5\ 7\ 1\ 2$ $\quad$ 'T','HIS' $\leftrightarrow$ 'THIS'
Index[3 4]	$V[A]$ $M[A;A]$ $A[A;..;A]$	$P[2] \leftrightarrow 3$ $\quad P[4\ 3\ 2\ 1] \leftrightarrow 7\ 5\ 3\ 2$ $E[1\ 3;3\ 2\ 1] \leftrightarrow \begin{matrix}3 & 2 & 1\\ 11 & 10 & 9\end{matrix}$ $E[1;] \leftrightarrow 1\ 2\ 3\ 4$ $E[;1] \leftrightarrow 1\ 5\ 9$ $\quad$ 'ABCDEFGHIJKL'[E] $\leftrightarrow \begin{matrix}ABCD\\ EFGH\\ IJKL\end{matrix}$
Index generator[3]	ιS	First S integers $\quad \iota 4 \leftrightarrow 1\ 2\ 3\ 4$ $\iota 0 \leftrightarrow$ an empty vector
Index of[3]	$V\iota A$	Least index of A in V, or $1+\rho V$ $\quad P\iota 3 \leftrightarrow 2$ $\quad 4\ 4\iota 4 \leftrightarrow 1$ $\quad P\iota E \leftrightarrow \begin{matrix}5 & 1 & 2 & 5\\ 3 & 5 & 4 & 5\\ 5 & 5 & 5 & 5\end{matrix}$
Take Drop	$V\uparrow A$ $V\downarrow A$	Take (drop) $\lvert V[I]$ first elements on coordinate I. (Last if $V[I]<0$) $\quad 2\ 3\uparrow X \leftrightarrow \begin{matrix}ABC\\ EFG\end{matrix}$ $\quad {}^{-}2\uparrow P \leftrightarrow 5\ 7$
Grade up[5] Grade down[5]	$\grave{a}A$ $\forall A$	The permutation which would order A (ascending or descending) $\quad \grave{a}3\ 5\ 3\ 2 \leftrightarrow 4\ 1\ 3\ 2$ $\quad \forall 3\ 5\ 3\ 2 \leftrightarrow 2\ 1\ 3\ 4$
Compress[5]	V/A	$1\ 0\ 1\ 0/P \leftrightarrow 2\ 5$ $\quad 1\ 0\ 1\ 0/E \leftrightarrow \begin{matrix}1 & 3\\ 5 & 7\\ 9 & 11\end{matrix}$ $1\ 0\ 1/[1]E \leftrightarrow \begin{matrix}1 & 2 & 3 & 4\\ 9 & 10 & 11 & 12\end{matrix} \leftrightarrow 1\ 0\ 1\not E$
Expand[5]	$V\backslash A$	$1\ 0\ 1\backslash\iota 2 \leftrightarrow 1\ 0\ 2$ $\quad 1\ 0\ 1\ 1\ 1\backslash X \leftrightarrow \begin{matrix}A & BCD\\ E & FGH\\ I & JKL\end{matrix}$
Reverse[5]	ϕA	$\phi X \leftrightarrow \begin{matrix}DCBA\\ HGFE\\ LKJI\end{matrix}$ $\quad \phi[1]X \leftrightarrow \ominus X \leftrightarrow \begin{matrix}IJKL\\ EFGH\\ ABCD\end{matrix}$ $\quad \phi P \leftrightarrow 7\ 5\ 3\ 2$
Rotate[5]	$A\phi A$	$3\phi P \leftrightarrow 7\ 2\ 3\ 5 \leftrightarrow {}^{-}1\phi P$ $\quad 1\ 0\ {}^{-}1\phi X \leftrightarrow \begin{matrix}BCDA\\ EFGH\\ LIJK\end{matrix}$
Transpose	$V\circledcirc A$	Coordinate I of A becomes coordinate $V[I]$ of result $\quad 1\ 1\circledcirc E \leftrightarrow 1\ 6\ 11.$ $\quad 2\ 1\circledcirc X \leftrightarrow \begin{matrix}AEI\\ BFJ\\ CGK\\ DHL\end{matrix}$
	$\circledcirc A$	Transpose last two coordinates $\quad \circledcirc E \leftrightarrow 2\ 1\circledcirc E$
Membership	$A\in A$	$\rho W\in Y \leftrightarrow \rho W$ $\quad P\in\iota 4 \leftrightarrow 1\ 1\ 0\ 0$ $\quad E\in P \leftrightarrow \begin{matrix}0 & 1 & 1 & 0\\ 1 & 0 & 1 & 0\\ 0 & 0 & 0 & 0\end{matrix}$
Decode	$V\perp V$	$10\perp 1\ 7\ 7\ 6 \leftrightarrow 1776$ $\quad 24\ 60\ 60\perp 1\ 2\ 3 \leftrightarrow 3723$
Encode	$V\top S$	$24\ 60\ 60\top 3723 \leftrightarrow 1\ 2\ 3$ $\quad 60\ 60\top 3723 \leftrightarrow 2\ 3$
Deal[3]	$S?S$	$W?Y \leftrightarrow$ Random deal of W elements from ιY

Tabella IV - Funzioni miste primitive (vedi note)

Note alla tabella IV

1. Le restrizioni sul rango degli argomenti sono indicate come segue: S per scalare, V per vettore, M per matrice di rango 2, A per matrice di rango qualsiasi. Eccetto che come argomento di *V⍳A* o *V[A]* uno scalare può essere sempre usato al posto di un vettore.
   Una matrice di un elemento può rimpiazzare qualunque scalare.

2. Il simbolo ↔ indica equivalenza.
   Le matrici usate negli esempi sono:

```
                                  1  2  3  4
P↔2 3 5 7                  E↔ 5  6  7  8
                                  9 10 11 12

     ABCD
X↔ EFGH
     IJKL
```

3. Poichè l'origine degli indici può essere posta a 0 oppure a 1 (con un comando di sistema), il valore della funzione dipende dall'origine.

4. L'omissione di un indice seleziona tutte le componenti relative all'indice mancante.

5. La funzione è applicata lungo l'ultima coordinata. I simboli ⌿, ⍀ e ⊖ sono rispettivamente equivalenti a /, \ e ○ eccetto che la funzione è applicata lungo la prima coordinata. Se [*S*] segue uno di questi simboli, la coordinata su cui si opera viene determinata dallo scalare S.

Appendice
Esempio di seduta al terminale

```
)1776
010 )  19.32.36 07/03/68  JANET

   A P L \ 3 6 0
```

FUNDAMENTALS

```
      3×4
12
      X←3×4
      X
12
      Y←¯5
      X+Y
7
      144E¯2
1.44
      P←1 2 3 4
      P×P
1  4  9  16
      P×Y
¯5  ¯10  ¯15  ¯20
      Q←'CATS'
      Q
CATS
      YZ←5
      YZ1←5
      YZ+YZ1
10
      3+4×5+6
         ∨
         +5+6
18
      X←3
      Y←4
      (X×Y)+4
16
      X×Y+4
24
```

Entry automatically indented
Response not indented
X is assigned value of
the expression

Value of X typed out
Negative sign for negative
constants

Exponential form of constant

Four-element vector
Functions apply element by element

Scalar applies to all elements

Character constant (4-element
vector)

Multi-character names

Correction by backspace
and linefeed

Executed from right to left

```
      X Y
SYNTAX ERROR
      X Y
        ∧
      XY
VALUE ERROR
      XY
      ∧
```

Entry of invalid expression
Shows type of error committed
Retypes invalid statement with
caret where execution stopped
Multi-character name (not X×Y)

XY had not been assigned a value

SCALAR FUNCTIONS

```
      4×3⌈5.1
20.4
      (4×3)⌈5.1
12
      4×⌈5.1
24
      X←⍳5
      X
1  2  3  4  5
      ⍳0

      Y←5-X
      Y
4  3  2  1  0
      X⌈Y
4  3  3  4  5
      X≤Y
1  1  0  0  0
      ○1
3.141592654
      ○÷1 2
3.141592654  1.570796327
      X←45 90
      ○X÷180
0.7853981634  1.570796327
      1○1
0.8418709848
      2○1 2
0.5403023059  ¯0.4161468365
      3○1
1.557407725
      ¯3○1
0.7853981634
      3○¯3○⍳7
1  2  3  4  5  6  7
      Y←1 2
      4○Y
1.414213562  2.236067977
      0○÷Y
0  0.8660254038
      7○1 2
0.761594156  0.9640275801
      ¯7○7○1 2
1  2
```

Dyadic maximum

Monadic ceiling

Index generator function

Empty vector
prints as a blank line
All scalar functions extend
to vectors

Relations produce
logical (0 1) results

Pi×1

Pi÷1 2

Conversion of X to radians

Sin 1

Cos 1 2

Tan 1

Arctan 1

Tan Arctan 1 2 3 4 5 6 7

(1+Y*2)*.5

(1-÷Y*2)*.5

Tanh 1 2

Arctanh Tanh 1 2

DEFINED FUNCTIONS

```
      ∇Z←X F Y
[1]   Z←((X*2)+Y*2)*.5
[2]   ∇
      3 F 4
5
      P←7
      Q←(P+1)F P-1
      Q
10
      4×3 F 4
20
      ∇B←G A
[1]   B←(A>0)-A<0
[2]   ∇
      G 4
1
      G ¯6
¯1
      X←¯6
      G X
¯1
      ∇H A
[1]   P←(A>0)-A<0
[2]   ∇
      H¯6
      P
¯1
      Y←H¯6
VALUE ERROR
      Y←H ¯6
        ^
      ∇Z←FAC N;I
[1]   Z←1
[2]   I←0
[3]   L1:I←I+1
[4]   →0×ι I>N
[5]   Z←Z×I
[6]   →L1
[7]   ∇
      FAC 3
6
      FAC 5
120
      TΔFAC←3 5
      X←FAC 3
FAC[3] 1
FAC[5] 1
FAC[3] 2
FAC[5] 2
FAC[3] 3
FAC[5] 6
FAC[3] 4
      TΔFAC←0
```

Header (2 args and result)
Function body
Close of definition
Execution of dyadic function F

Use of F with expressions
as arguments

G is the signum function
A and B are local variables

Like G but has no explicit result
P is a global variable

H has no explicit result
and hence produces a value
error when used to right
of assignment
FAC is the factorial function

L1 becomes 3 at close of def
Branch to 0 (out) or to next

Branch to L1 (that is, 3)

Set trace on lines 3 and 5 of FAC

Trace of FAC

Reset trace control

```
      ∇G←M GCD N
[1]   G←N
[2]   M←M|N
[3]   →4×M≠0
[4]   [1]G←M
[2]   [4]N←G
[5]   [1□]
[1]   G←M
[1]   [□]
    ∇ G←M GCD N
[1]   G←M
[2]   M←M|N
[3]   →4×M≠0
[4]   N←G
    ∇
[5]   →1
[6]   ∇
      36 GCD 44
4
      ∇GCD
[6]   [4.1]M,N
[4.2] [□]
    ∇ G←M GCD N
[1]   G←M
[2]   M←M|N
[3]   →4×M≠0
[4]   N←G
[4.1] M,N
[5]   →1
    ∇
[6]   ∇
      36 GCD 44
8  36
4  8
4
      ∇GCD[□]∇
    ∇ G←M GCD N
[1]   G←M
[2]   M←M|N
[3]   →4×M≠0
[4]   N←G
[5]   M,N
[6]   →1
    ∇
      ∇GCD
[7]   [5]
        ^
      ∇
```

MECHANICS OF FUNCTION DEFINITION

Greatest common divisor
function based on the
Euclidean algorithm

Correction of line 1
Resume with line 4
Display line 1

Display entire GCD Function

Close of display, not close of def
Enter line 5
Close of definition
Use of GCD
4 is GCD of 36 and 44
Reopen def (Use ∇ and name only)
Insert between 4 and 5
Display entire function

Fraction stays until close of def

End of display
Close of definition

Iterations printed by
line 5 (was line 4.1)
Final result
Reopen, display, and close GCD

Line numbers have been
reassigned as integers
Close (Even number of ∇'s in all)
Reopen definition of GCD
Delete line 5 by linefeed

Close definition

```
      ∇Z←ABC X
[1]   Z←(33×Q+(R×5)-6
[2]   [1⎕9]
[1]   Z←(33×Q+(R×5)-6
         /   1 /1
[1]   Z←(3×Q)+(T×5)-6
[2]   ∇
      FAC 5
120
      )ERASE FAC
      FAC 5
SYNTAX ERROR
      FAC 5
      ^
      ∇Z←BIN N
[1]   LA:Z←(Z,0)+0,Z
[2]   →LA×N≥ρZ∇
      BIN 3
VALUE ERROR
BIN[1] LA:Z←(Z,0)+0,Z
                  ^
      Z←1
      →1
1  3  3  1
      BIN 4
VALUE ERROR
BIN[1] L1:Z←(Z,0)+0,Z
                  ^
      ∇BIN[.1]Z←1∇
      )SI
BIN[1] *
      →1
1  4  6  4  1
      ∇BIN[⎕]∇
    ∇ Z←BIN N
[1]   Z←1
[2]   LA:Z←(Z,0)+0,Z
[3]   →LA×N≥ρZ
    ∇
      SΔBIN←2
      Q←BIN 3

BIN[2]
      Z
1
      →2

BIN[2]
      →2

BIN[2]
      →0
```

A function to show line editing
A line to be corrected
Initiate edit of line 1
Types line, stops ball under 9
Slash deletes, digit inserts spaces
Ball stops at first new
space. Then enter) T
FAC still defined

Erase function FAC
Function FAC no longer exists

An (erroneous) function for
binomial coefficients

Suspended execution

Assign value to Z
Resume execution
Binomial coefficients of order 3

Same error (local variable Z
does not retain its value)

Insert line to initialize Z
Display state indicator
Suspended on line 1 of BIN
Resume execution (BIN now correct)

Display revised function
and close definition

Set stop on line 2
Execute BIN

Stop due to stop control
Display current value of Z

Resume execution

Stop again on next iteration
Resume

Stop again
Branch to 0 (terminate)

```
      ∇MULTDRILL N;Y;X
[1]   Y←?N
[2]   Y
[3]   X←⎕
[4]   →0×⍳X='S'
[5]   →⍳X=×/Y
[6]   'WRONG, TRY AGAIN'
[7]   →3∇
      MULTDRILL 12 12
2  10
⎕:
      37
WRONG, TRY AGAIN
⎕:
      20
6  7
⎕:
      'S'
      ∇Z←ENTERTEXT
[1]   Z←''
[2]   D←ρZ
[3]   Z←Z,⍞
[4]   →2×D≠ρZ
[5]   ∇
      Q←ENTERTEXT
THIS IS ALL
 CHARACTER INPUT

      Q
THIS IS ALL CHARACTER INPUT
      N←5
      'NOTE: ⍳';N;' IS ';⍳N
NOTE:⍳5 IS 1  2  3  4  5


      P←2 3 5 7
      ρP
4
      T←'OH MY'
      ρT
5
      P,P
2  3  5  7  2  3  5  7
      T,T
OH MYOH MY
      T,P
DOMAIN ERROR
      T,P
      ^
```

INPUT AND OUTPUT

A multiplication drill
ρN random integers
Print the random factors
Keyboard input
Stop if entry is the letter S
Repeat if entry is correct product
Prints if preceding branch fails
Branch to 3 for retry
Drill for pairs in range 1 to 12

Indicates that keyboard entry
is awaited

Entry of letter S stops drill
Example of character (⍞) input
Make Z an empty vector
D is the length of Z
Append character keyboard entry
Branch to 2 if length increased
(i.e., entry was not empty)

Keyboard
entries
Empty input to terminate
Display Q

Mixed output statement

RECTANGULAR ARRAYS

Dimension of P

Character vector

Catenation

Characters cannot be catenated
with numbers

```
      M←2 3ρ2 3 5 7 11 13
    ⋄ M

2   3   5
7  11  13
      2 4ρT

OH M
YOH
      6ρM
2  3  5  7  11  13
      ,M
2  3  5  7  11  13
      P←,M
      P[3]
5
      P[1 3 5]
2  5  11
      P[ι3]
2  3  5
      P[ρP]
13
      M[1;2]
3
      M[1;]
2  3  5
      M[1 1;3 2]

  5  3
  5  3
      A←'ABCDEFGHIJKLMNOPQ'
      A[M]

BCE
GKM
      A[M[1 1;3 2]]

EC
EC
      M[1;]←15 3 12
      M

 15   3  12
  7  11  13
```

Reshape to produce a 2×3 matrix
Display of an array of rank >1
is preceded by a blank line

A 2×4 matrix of characters

A matrix reshaped to a vector

Elements in row-major order

Indexing (third element of P)

A vector index

The first three elements of P

Last element of P

Element in row 1, column 2 of M

Row 1 of M

Rows 1 and 1, columns 3 2

The alphabet to Q
A matrix index produces
a matrix result

Respecifying the first row of M

```
      Q←3 1 5 2 4 6
      P[Q]
5  2  11  3  7  13
      Q[Q]
5  3  4  1  2  6
      P[3]
5
      )ORIGIN 0
WAS 1
      P[3]
7
      P[0 1 2]
2  3  5
      ι5
0  1  2  3  4
      )ORIGIN 1
WAS 0
      ι5
1  2  3  4  5
```

A permutation vector
Permutation of P

A new permutation

Present index origin is 1

Set index origin to 0

First three elements of P

Result of index generator
begins at origin

FUNCTIONS ON ARRAYS

```
      V←?3ρ9
      M←?3 3ρ9
      N←?3 3ρ9
      V
2  1  7
      M

 7  9  4
 5  8  1
 1  5  7
      N
 1  4  1
 4  7  6
 9  8  5
      M+N

 8  13   5
 9  15   7
10  13  12
```

Vector of 3 random integers (1-9)
Random 3 by 3 matrix
Random 3 by 3 matrix

Sum (element-by-element)

```
      M⌈N
  7  9  4
  5  8  6
  9  8  7
```
Maximum

```
      M≤N
 0 0 0
 0 0 1
 1 1 0
```
Comparison

```
      +/V
10
```
Sum-reduction of V

```
      ×/V
14
```
Product-reduction

```
      +/[1]M
13  22  12
```
Sum over first coordinate of M (down columns)

```
      +/[2]M
20  14  13
```
Sum over second coordinate of M (over rows)

```
      +/M
20  14  13
```
Sum over last coordinate

```
      ⌈/M
9  8  7
```
Maximum over last coordinate

```
      X←1.5
      +/(1 2○X)*2
1
```
Sin squared plus Cos squared

```
      ○/1 2,X
0.07067822453
```
Sin Cos X

```
      Y←○/0 2,X
      Y
0.9974949866
```
(1-(*COS X*)*2)*.5

```
      Y=1○X
1
```
An identity

```
      M+.×N
  79  123   81
  46   84   58
  84   95   66
```
Ordinary matrix (+ × inner) product

```
      M+.≤N
 1  1  1
 1  1  1
 2  3  2
```
An inner product

```
      M+.×V
51  25  56
```
+ × inner product with vector right argument

```
      V
2  1  7
      V∘.×15
  2   4   6   8  10
  1   2   3   4   5
  7  14  21  28  35
```
Outer product (times)

```
      V∘.≤19
0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 1
```
Outer product

```
      V∘.×M
 14  18   8
 10  16   2
  2  10  14

  7   9   4
  5   8   1
  1   5   7

 49  63  28
 35  56   7
  7  35  49
```
An outer product of rank 3

A blank line between planes

MIXED FUNCTIONS

```
      Q←?10⍴5
      Q
1  4  3  4  5  4  2  1  4  2
      +/[1]Q∘.=15
2  2  1  4  1
```
A random 10 element vector (range 1 to 5)

Ith element of result is number of occurences of the value I in Q

```
      2 1⍉M
 7  5  1
 9  8  5
 4  1  7
```
Ordinary transpose of *M*

```
      ⍉M
 7  5  1
 9  8  5
 4  1  7
```
Ordinary transpose of M (monadic)

```
      T←2 3 4⍴⍳24
      T
```

An array of rank 3

```
 1   2   3   4
 5   6   7   8
 9  10  11  12

13  14  15  16
17  18  19  20
21  22  23  24
```

```
      3 1 2⍉T
```

Transpose of T (dimension of result is 3 4 2)

```
 1  13
 2  14
 3  15
 4  16

 5  17
 6  18
 7  19
 8  20

 9  21
10  22
11  23
12  24
```

```
      1 1⌽M
7  8  7
      1 1 2⍉T
```

Diagonal of M

Diagonal section in first two coordinates of T

```
 1   2   3   4
17  18  19  20
      X←○(0,⍳5)÷6
      )DIGITS 4
WAS 10
      ⍉1  2  3○.×X
```

Set number of output digits to 4

```
 0.000E0     1.000E0     0.000E0
 5.000E¯1    8.660E¯1    5.774E¯1
 8.660E¯1    5.000E¯1    1.732E0
 1.000E0     1.744E¯16   5.734E15
 8.660E¯1   ¯5.000E¯1   ¯1.732E0
 5.000E¯1   ¯8.660E¯1   ¯5.774E¯1
```

Table of sines, cosines, and tangents in intervals of 30 degrees

```
      Q
1  4  3  4  5  4  2  1  4  2
      3⌽Q
4  5  4  2  1  4  2  1  4  3
     ¯3⌽Q
1  4  2  1  4  3  4  5  4  2
      0 1 2⌽[1]M
```

Rotate to left by 3 places

Rotate to right by 3 places

Rotate columns by different amounts

```
 7  8  7
 5  5  4
 1  9  1
     ¯2⌽[2]M
```

Rotation of rows all by 2 to right

```
 9  4  7
 8  1  5
 5  7  1
      1 2 3⌽M
```

Rotation of rows

```
 9  4  7
 1  5  8
 1  5  7
      ⌽Q
2  4  1  2  4  5  4  3  4  1
```

Reversal of Q

```
      ⌽[1]M
 1  5  7
 5  8  1
 7  9  4
      ⌽M
 4  9  7
 1  8  5
 7  5  1
```

Reversal of M along first coordinate

Reversal along last coordinate

```
      U←Q>4
      U
0 0 0 0 1 0 0 0 0 0
      U/Q
5
      (~U)/Q
1 4 3 4 4 2 1 4 2
      +/U/Q
5
      1 0 1/[1]M
  7 9 4
  1 5 7
      1 0 1/M
  7 4
  5 1
  1 7
      (,M>5)/,M
7 9 8 7
      V←1 0 1 0 1
      V\⍳3
1 0 2 0 3
      V\M

  7 0 9 0 4
  5 0 8 0 1
  1 0 5 0 7
      V\'ABC'
A B C
      10⊥1 7 7 6
1776
      8⊥1 7 7 6
1022
      (4⍴10)⊤1776
1 7 7 6
      (3⍴10)⊤1776
7 7 6
      10 10⊤1776
7 6
      10⊤1776
6
      24 60 60⊥1 3 25
3805
      24 60 60⊤3805
1 3 25
      2⊥1 0 1 1 0
22
```

Compression of Q by logical vector U

Compression by not U

Compression along first coordinate of M

Compression along last coordinate

,M is 7 9 4 5 8 1 1 5 7
All elements of M which exceed 5

Expansion of iota 3

Expansion of rows of M

Expansion of literal vector inserts spaces

Base 10 value of vector 1 7 7 6

Base 8 value of 1 7 7 6

4 digit base 10 representation of number 1776

3 digit base 10 representation of 1776

Mixed base value of 1 3 25 (time radix)

Representation of number 3805 in time radix

Base 2 value

```
      M
  7 9 4
  5 8 1
  1 5 7
      )ORIGIN 0
WAS 1
      M[2;0]
1
      (,M)[(⍴M)⊥2,0]
1
      )ORIGIN 1
WAS 0
      P
2 3 5 7 11 13
      P⍳7
4
      P⍳6
7
      P⍳4 5 6 7
7 3 7 4
      Q←5 1 3 2 4
      R←Q⍳⍳⍴Q
      R
2 4 3 5 1
      Q[R]
1 2 3 4 5
      A←'ABCDEFGHIJKLMNOPQ'
      A←A,'RSTUVWXYZ'
      A
ABCDEFGHIJKLMNOPQRSTUVWXYZ
      A⍳'C'
3
      J←A⍳'CAT'
      J
3 1 20
A[J]
CAT
```

Indexing of matrix in 0-origin. Note relation to indexing of ravel of M

Restore 1-origin

Index of 7 in vector P
7 is 4th element of P
6 does not occur in P, hence result is 1+⍴P

A permutation vector
R is the permutation inverse to Q

A is the alphabet

Rank of letter C in alphabet is 3

```
      M←3 5⍴'THREESHORTWORDS'
      M
```

A matrix of characters

```
THREE
SHORT
WORDS
```

```
      J←A⍋M
      J
  20   8  18   5   5
  19   8  15  18  20
  23  15  18   4  19
      A[J]
```

Ranking of M produces a matrix

Indexing by a matrix produces a matrix

```
THREE
SHORT
WORDS
```

```
      3?5
5  1  2
      6?5
DOMAIN ERROR
      6?5
       ^
      X←8?8
      X
4  6  7  2  5  1  8  3
      ⍋X
6  4  8  1  5  2  3  7
      X[⍋X]
1  2  3  4  5  6  7  8
      X[⍒X]
8  7  6  5  4  3  2  1
      U←A∈'NOW IS THE TIME'
      '01'[1+U]
0000100110001110001100100 0
      U/A
EHIMNOSTW
      (⍳8)∈3 7 5
0  0  1  0  1  0  1  0
```

Random choice of 3 out of 5 without replacement

A random permutation vector

Grading of X

Arrange in ascending order

Arrange in descending order

Membership