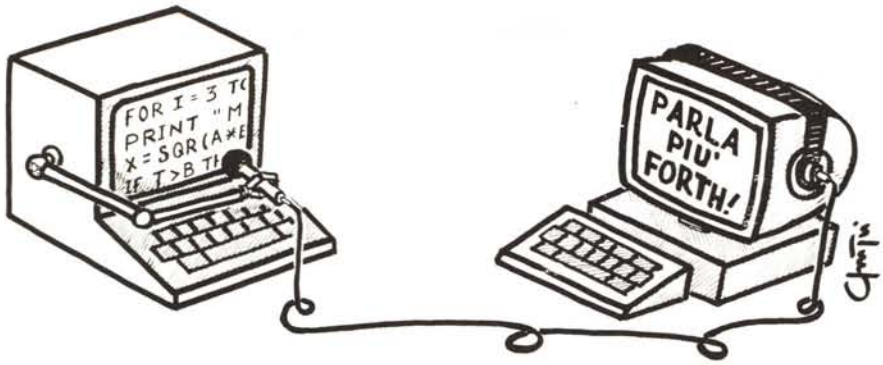


Parla più FORTH

Quinta parte **di Raffaello De Masi**

Le strutture decisionali

Tutto quello che è stato finora scritto ci ha consentito di preparare programmi, o word (in Forth, se ci pensate un attimo, sono la stessa cosa), che vengono eseguiti sequenzialmente. Vale a dire che il sistema operativo, al momento di eseguire un programma od una definizione, esegue la prima word, poi la seconda, la terza ecc. fino a pervenire alla word. Cioè, come pittorescamente riferisce Reynamm nel suo "Understanding Forth", il programma si comporta come uno chef durante un pranzo, portando in tavola un piatto alla volta senza mai ripresentarlo ritornando indietro nel menu.

In altre parole, finora abbiamo avuto programmi che non avevano potere decisionale, eseguendo la stessa sequenza di ordini senza tener conto del genere di input inserito. Purtroppo le cose, nella realtà, non sono così semplici, in quanto, molto spesso, è lo stesso input che impone certe scelte o decisioni particolari, in funzione del genere di input medesimo o dello stato globale delle variabili utilizzate nel programma in quel particolare momento.

Un tipo decisionale può essere quello che un certo tipo di operazioni debba essere eseguito in determinate circostanze, mentre è bypassato in altre. O, ancora, un gruppo di istruzioni vada eseguito un determinato numero di volte prima di passare ad altre, o vada ripetutamente eseguito finchè una certa variabile non assuma un valore particolare.

Il computer, in effetti, basa le sue capacità decisionali sul controllo dell'ordine con cui le linee sono eseguite nell'ambito del programma. Il linguaggio Forth contiene uno speciale set di parole che consentono tali controlli. Queste word non sono mai usate individualmente ma sono sempre combinate, sia in coppia, sia in gruppi di tre, a formare una struttura di controllo.

L'utilità di tali strutture è evidente, specie per chi conosce altri linguaggi di programmazione. Ma, proprio per questo, appare evidente immediatamente che l'uso del solo stack (anche in combinazione col return stack) non è più sufficiente. Non è più possibile, cioè, utilizzare solo strutture temporanee di deposito ma anche aree più stabili e sicure.

Queste non possono essere altro che aree di RAM, eventualmente interessanti, talvolta, le memoria di massa.

Per chi, come dicevamo, conosce le strutture decisionali, appare evidente la complessità di poter utilizzare il solo stack per eseguire forme e tipi decisionali o sequenze di controllo. Introdurremo qui alcune chiarificazioni circa word che operano direttamente nella memoria.

Nel forth (come anche in altri linguaggi) gli indirizzi sono rappresentati da numeri assoluti compresi tra 0 e 65535. È questo il generico campo di dominio di memoria di un microprocessore ad 8 bit, che possieda bus di indirizzi di 16 linee.

Inoltre, generalmente, le memorie sono organizzate in byte (8 bit di dati), la più piccola unità di memoria indirizzabile da un microprocessore. Questa può rappresentare un range di valori tra 0 e 225 e tra -128 e 127, a seconda che il MSB (Most Significant Bit, l'ultimo bit a sinistra) sia riservato al segno, o come più comunemente si dice, se si tratti di numeri senza segno o no.

È opportuno, inoltre, che ogni forthista abbia una almeno generica conoscenza della mappa di memoria del proprio computer. Dietro tale terrificante parola si nasconde una cosa piuttosto semplice: la memoria di un computer, infatti, è suddivisa in aree diverse, in zone in cui sono contenute distinte cose. C'è l'area di ROM, dove è compresa la Read Only Memory, contenente la struttura di sistema, non volatile e non modificabile; c'è l'area di RAM destinata ad accogliere il programma e le variabili; esistono ancora aree particolari destinate al pilotaggio delle periferiche, alla ROM video, a programmi binari, siano essi rilocabili o non.

È necessaria, dicevamo, una conoscenza, anche solo superficiale, delle aree specifiche della mappa di memoria. Le ragioni appariranno più evidenti in seguito. Ma per fare un esempio pratico ed immediato, la conoscenza delle frontiere e delle aree riservate in mappa è importante perchè, potendo direttamente intervenire il Forth su specifiche locazioni di memoria, non si invadano (o lo si faccia a ragion veduta) certe aree particolari. Comunque non vi spaventate, è più semplice da fare che da dire.

Le operazioni sulla memoria

Esistono 4 word fondamentali che trasferiscono numeri e byte da e alla memoria. Le word @ e C@ leggono un numero od un byte all'indirizzo di memoria specificato e lo pongono in TOS (Top Of Stack). Le word ! e C! eseguono l'operazione inversa estraendo l'indirizzo ed il valore dello stack.

Ad esempio

```
50     200 !
```

deposita il numero 50 all'indirizzo 200

L'operazione

```
200 @ . darà
50 OK
```

leggendo il valore 50 all'indirizzo 200, depositandolo in TOS e visualizzandolo in ossequio alla word.

Analogamente con

```
50 200 @ ! e
200 C@ .
50 OK
```

viene eseguita la stessa funzione manipolando byte e non numeri.

È importante chiarire che la word finora definita (e generalmente, quelle che operano in memoria) allorchè estraggono numeri dalla memoria, eseguono solo operazioni di copiatura, vale a dire non cancellano il dato già presente nella locazione richiamata. Occorre inoltre ricordare che un numero semplice occupa *due* byte in memoria, ed un numero in doppia precisione occupa *quattro* byte in memoria.

Esistono in alcuni sistemi le word 2@ e 2! che manipolano (lettura e scrittura in memoria) numeri doppi. Queste due word, piuttosto utili, possono essere, nei sistemi mancanti, sostituite secondo la procedura indicata in figura 1 e figura 2. Nella prima viene effettuata una ricerca di un numero in doppia precisione alla locazione il cui valore è presente nello stack, nell'altra viene deposta nella locazione adr il numero doppio d.

Lettura e manipolazione del contenuto in memoria

Spesso, specie durante il debug di un programma, è necessario conoscere, in un dato momento, che cosa è contenuto in

```
0 ( definizione della word 2@)
1 ( destinata alla ricerca (fetch) di un numero doppio in memoria)
2
3 : 2@          ( definisce la word )
4 DUP 2+        ( lascia in TOS l'indirizzo + 2 )
5 @             ( legge i 16 bit meno significativi )
6 SWAP          ( forza l'indirizzo iniziale in TOS )
7 @             ( legge i 16 bit piu' significativi )
8 ;             ( fine definizione )
9
10              ( utilizzare la definizione nel seguente modo )
11              ( indirizzo   ---   numero doppio )
12
13
14
15
```

Figura 1 - Definizione della word 2@.

```
0               ( definizione della word 2! )
1               ( destinata a depositare all'indirizzo in TOS )
2               ( il doppio numero contenuto al secondo posto )
                ( dello stack )
3
4 : 2!
5 ROT           ( ruota i 16 bit di ordine basso in TOS )
6 OVER          ( posta l'indirizzo in TOS )
7 2+            ( e in modifica il valore di 2 )
8
9 !             ( immagazzina i 16 bit di ordine basso )
10 !            ( immagazzina i 16 bit di ordine alto )
11 ;            ( fine definizione )
12              ( valore indirizzo  --- )
13
14
15
```

Figura 2

una certa locazione di memoria, senza coinvolgere in questa operazione lo stack.

La word ? mostra direttamente sul display, senza alcun altro coinvolgimento implicito od esplicito di altre strutture di calcolo, il numero immagazzinato all'indirizzo specificato. Ad esempio, la sequenza

```
200 ? 50 OK
```

mostra il contenuto (50) della locazione di memoria 200.

Una ancora più utile word è DUMP che mostra un determinato numero di locazioni di memoria, a partire dall'indirizzo specificato. Molti linguaggi FORTH, incluso il FIG-FORTH mostrano (ricordate il dump in linguaggio macchina) diversi valori su una linea, preceduti dall'indirizzo di inizio di lettura.

Ad esempio la procedura

```
220 10 DUMP
```

produrrà

```
220 15 0 30 2 0 0 0
228 3 125
```

È possibile, altresì, aggiungere un valore ad un altro già allocato in memoria, con la word +!. Per esempio

```
5 200 +!
```

aggiunge il valore 5 al contenuto della locazione 200. Una sottrazione, evidentemente, è altrettanto semplice

```
-5 200 +!
```

Inoltre è possibile, pur non essendoci una vera e propria word dedicata, eseguire moltiplicazioni e divisioni su numeri in memoria. L'algoritmo consiste nel ricercare il numero, operare su di esso tenendolo in stack, riportandolo in memoria al termine delle operazioni. La sequenza in figura 3 è pertanto generalizzata con le righe 8, 9, 10 contenenti commenti e che vanno sostituite con le operazioni. È evidente che queste possono essere di qualunque tipo e quantità, tenendo solo conto che, alla fine, i primi due posti dello stack devono essere occupati dall'indirizzo e dal risultato.

A tal uopo può essere molto utile l'uso di una particolare word, DEPTH, che lascia in TOS il valore di numeri doppi (16 bit) presenti nello Stack. Si tratta di una word molto utile che, ahimé, non esiste nello standard FIG FORTH. Esistono altresì due altre word SP@ e S0 che puntano all'indirizzo del TOS e del fondo dello stack. Con esse è possibile conoscere l'indirizzo di qualunque numero presente in stack; ad esempio, se SP@ lascia l'indirizzo del primo numero, SP@ 2+ lascerà l'indirizzo del secondo numero, SP@ 4+ quello del terzo (non dimenticate che stanno lavorando su numeri doppi) e così via.

Spostare un blocco di dati da un punto ad un altro della memoria è altrettanto molto agevole e rapido con le due word MOVE e CMOVE. Ambedue le word accettano tre argomenti; un indirizzo sorgente (ind. 1) un indirizzo di destinazione (ind.2) ed un numero che individua i valori da muovere (n). Funzionano allo stesso modo con la sola differenza che la prima opera su numeri e la seconda su byte. Ad esempio la sequenza

```
1200 1300 8 MOVE
```

copierà dall'indirizzo 1200 al 1300 in sequenza otto numeri.

È interessante notare che è anche possibile spostare numeri nella parte bassa della memoria. Basta in questo caso investire i valori di partenza ed arrivo. L'esempio precedente diverrà:

```
1300 1200 8 MOVE
```

Il Forth 79 include un'altra word molto interessante, <CMOVE, che consente una copia di blocchi di memoria (byte) invertendo l'ordine di copiatura, vale a dire che il primo valore da copiare sarà sistemato all'ultimo indirizzo del blocco di destinazione. La necessità di una tale word non sembra evidente; essa però consente di copiare un blocco di memoria in un altro attiguo ma più corto, vale a dire che è possibile copiare ad esempio un blocco di 8 byte in uno attiguo lungo 3 byte (gli altri 5 occuperanno una porzione del blocco di partenza) senza cancellare alcun valore.

È infine possibile fare pulizia di quanto esiste in un blocco di memoria (vale a dire è possibile inizializzarne una porzione) inserendo in esso un valore conosciuto (per lo più zero) con la word FILL. Essa possiede la seguente sintassi

```
ind n n FILL ---
```

Ad esempio la sequenza

```
75 50 0 FILL
```

inserisce alle locazioni comprese tra 75 e 124 (75+25 ricordando di contare anche la prima locazione) il valore 0.

Spesso, però, in sistemi Forth orientati al word processing o comunque alla manipolazione di dati alfanumerici più che lo 0, necessario in operazioni a base numerica, è opportuno ripulire le aree di memoria inserendo il blank (ASCII 32 o SP mnemonico). La word BLANKS esegue la stessa operazione di FILL, eliminando, in pratica, il testo scritto nelle locazioni desiderate.

BLANKS, assieme a CMOVE, è molto utile appunto nel word processing. Ad esempio, la sequenza

```
200 400 20 C MOVE 200 20 ERASE
```

sposta una frase di 20 lettere da un punto all'altro della memoria facendo pulizia nel blocco di partenza.

La struttura DO-LOOP

Chi possiede un minimo di conoscenza di programmazione conosce od ha già intuito questa insostituibile sequenza operativa. Ad esempio immaginiamo di voler leggere i primi 10 numeri presenti nello stack e di volerli incolonnati.

Costruiremo la word

```
: ELENCO 10
CR.CR.CR.CR.CR.
(stampa i primi 5 numeri partendo dal TOS)
CR.CR.CR.CR.CR.
(stampa il secondo gruppo)
(fine definizione) ;
```

Utilizzando la struttura DO-LOOP avremo

```
: ELENCO 10
11 1 DO CR . LOOP ;
```

che darà lo stesso risultato.

La sintesi della word è la seguente:

```
: n1 n2 DO istruzioni LOOP ;
```

dove

n1 è il valore d'arrivo aumentato di 1 ;

n2 è quello di partenza ;

```
0               ( operazioni numeriche su memoria )
1               ( il programma consente di eseguire operazioni )
2               ( su un numero presente in memoria )
3               ( e di sostituire ad esso il risultato )
4               ( lo stack contiene gia' l'operando e l'indirizzo )
5 DUP @         ( duplica l'indirizzo e ricerca il numero in memoria )
6 ROT           ( ruota l'operando in TOS )
7
8               ( righe libere per eventuali operazioni )
9               (              "                    )
10              (              "                    )
11
12 SWAP         ( riporta l'indirizzo in TOS )
13 !            ( riporta il risultato all'indirizzo )
14              ( fine )
15
```

Figura 3

istruzioni sono le operazioni da eseguire.

Il sistema operativo legge n2 ed n1 ed esegue (n1 - n2) volte l'operazione rappresentata dalla parola istruzioni.

La sintassi è leggermente diversa dall'analoga struttura presente in altri linguaggi. Il valore d'arrivo va sempre aumentato di 1; questo è dovuto al particolare modus operandi del sistema operativo del FORTH che, incontrando tale struttura, inizializza un contatore al valore n2 ed aggiunge 1 all'indice ogni volta che incontra la parola LOOP. Quando il valore del contatore è eguale o più grande di n1 il ciclo è completo ed il controllo passa alla word successiva (nel nostro caso ;). Il diagramma di flusso in figura 4 esemplifica il processo.

È interessante notare come, poiché per l'escape dalla struttura è necessario incontrare la word LOOP (che è sempre posta alla fine della definizione), il ciclo DO-LOOP è sempre eseguito almeno una volta. Inoltre occorre tener presente che la sequenza n1 = 10 ed n2 = 0 oppure n1 = -15 ed n2 = -4 daranno lo stesso risultato vale a dire qualunque combinazione di numeri validi è efficace. Facciamo infine notare che n1 ed n2 che vengono, nel momento della loro stessa definizione, messi in Stack, vengono immediatamente recuperati dalla word DO, che li preleva da esso. Ciò è importante in quanto i due valori possono benissimo già essere presenti in esso.

Tanto per complicare le cose, il Forth consente ad ogni ciclo di copiare nello stack l'indice (il valore di conto del loop). Ciò è possibile inserendo nella struttura la word I nel seguente modo

```
n1 n2 DO ... I ... LOOP
```

Una volta nello stack il valore dell'indice può essere usato come un qualsiasi valore di stack. Esso può essere stampato, duplicato, sdoppiato, ecc. in quanto non è più vincolato in alcun modo alla struttura del loop nè è collegato con l'indice di DO-LOOP. Ad esempio la word QUADRATI

```
: QUADRATI 9 1 DO CR !
DUP DUP . "il quadrato di" . * . "è"
. LOOP ;
```

stamperà tutti i quadrati dei numeri dall'1 all'8.

Analogamente a molti altri linguaggi il Forth consente però di eseguire cicli di DO-LOOP non solo con indice 1 ma con qualsiasi indice (sempre intero, come al solito) si voglia. La struttura della word corrispondente è

```
n1 n DO ..... n2 +LOOP
```

usa normalmente i valori di partenza e di arrivo della struttura ma l'indice di loop viene incrementato di n2.

Se n2 è positivo n1 deve essere più grande di n ed il loop prosegue finchè il valore del contatore è eguale o supera n1. Se n2 è negativo accade il contrario.

Ad esempio

```
10 0 DO .... 2 +LOOP
```

causerà l'esecuzione della operazione rappresentata da .... 5 volte (corrispondente ai valori 0, 2, 4, 6, 8 assunti dall'indice).

Allo stesso modo

```
0 10 DO .... -2 +LOOP
```

determinerà anche esso l'esecuzione di.... 5 volte (valori assunti dal contatore 10,8,6,4,2).

Come in molti altri linguaggi i cicli DO-LOOP possono essere nidificati. Vale a dire che una serie di operazioni viene ripetuta nel loop più interno per ogni ciclo del loop più esterno. Generalmente per prassi comune, i loop più interni vengono sistemati sulle righe spostati una colonna più a destra di quelli più esterni (memento Pascal).

A questo punto facciamo una piccola digressione. Ricordate quel bug, presente su diversi Basic anche abbastanza recenti che si manifestava allorché si usciva prematuramente da un loop? Controllate la seguente routine Basic

```
490  ....
500  ....
510  FOR  A = 1 TO 30
520  B = A*C
530  IF C>  200  THEN  550
540  NEXT  A
550  ....
```

Se la variabile C assumeva ad esempio il valore 10 al raggiungimento di A del valore 20 si sarebbe avuto il prematuro abbandono del loop. Fin qui tutto giusto ma se l'episodio si ripeteva varie volte alcuni interpreti Basic, dopo un certo numero di accadenze, si piantavano. La spiegazione era abbastanza ovvia in quanto (per usare una espressione molto indovinata che ricordo di aver letto su una rivista) la serie continuamente interessata da loop lasciava, per così dire l'interprete col fiato sospeso fino a che il continuo ripetersi delle sospensioni mandava in tilt il sistema (in pratica esauriva la disponibilità di puntatori). Occorreva pertanto modificare la riga 530 in

```
530  IF C>200 Then A = 30
```

Il Forth evita l'artificio e utilizza la word LEAVE che setta il puntatore di LOOP al limite finale. Ma, poichè tale word è implicitamente collegata a test condizionali, ne rimandiamo la discussione al prossimo numero.

Anche stavolta pare che abbiamo finito. La prossima volta parleremo di strutture decisionali dopo le quali passeremo alla vera e propria programmazione. MC

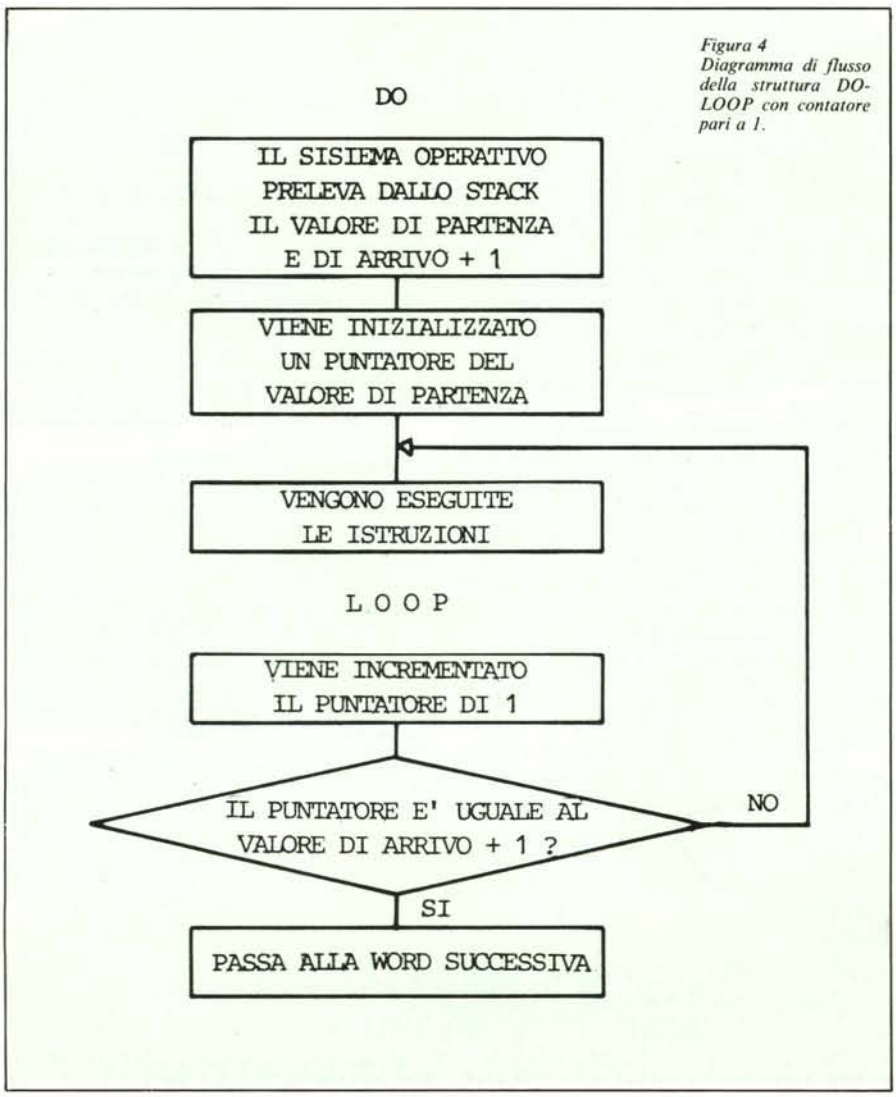


Figura 4
Diagramma di flusso della struttura DO-LOOP con contatore pari a 1.

"THE MAGIC BOX"

L. 299.000 IVA esclusa

XL 80* Arricchisce l'intelligenza del vostro Commodore

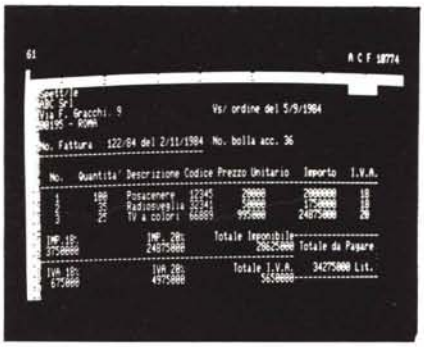
L'XL 80 è una unità di espansione per il computer Commodore che offre una serie di caratteristiche professionali solo riscontrabili in computer più costosi.
Una volta inserito nella « Cartridge port » del Vs/Commodore, L'XL 80 Vi fornirà un'uscita video ad alta definizione di 25 linee a 80 colonne oppure di 25 linee a 40 colonne.
Il « Firmware » dell'XL 80 rende il Commodore di un 20% più veloce liberandolo da molte funzioni di « house-keeping ».
Inoltre, può fungere da « terminal emulator », il che significa che il Vs/Commodore può essere utilizzato come terminale ad 80 colonne di un « Mainframe computer » oppure come servizio di « time-sharing ».
In più, il sistema Vi viene fornito con un consistente pacchetto il sofware completo di « auto-start menu »

« Word Manager »

Il « Word Manager » è veramente amico Vostro. Ha una caratteristica speciale; una striscetta da posizionare sulla tastiera del Vs/Commodore per facilitarVi la scelta delle varie funzioni.
Il « Word Manager » raffigura sullo schermo il documento esattamente come sarà stampato, così non occorre fare prove di stampa per vedere come verrà il documento.
Per la compilazione dei testi, c'è una caratteristica professionale che Vi permette di inserire parole o frasi — una caratteristica che taglia la riga al punto che volete per inserire quel che volete.
È disponibile anche un programma « Mailing List » (indirizzario) su disco e questo può operare in combinazione con « Word Manager ».
Alcune caratteristiche importanti del « Word Manager » sono:

- — File di 4 pagine
- — Editing completo
- — Block Move e Copy (spostamento o copia di blocchi di testo)
- — Ricerca e sostituzione di parole e di blocchi di parole
- — Margini e tabulati
- — Centramento automatico
- — Allineamento del margine sia a sinistra che a destra
- — Selezione stampante
- — Compatibile con Seria! e RS-232

Il « Plan Manager » è uno spreadsheet professionale completo di un « Help screen ».
Lo spreadsheet Vi permette fino a 63 colonne e 254 righe per i calcoli ed ha un programma completo « What If ».



« Plan Manager »

Sono disponibili funzioni matematiche complete come addizioni, sottrazioni, moltiplicazioni, divisioni ed operazioni negative, ammontare di riga o di colonna, minimi, massimi, conteggi, calcoli esponenziali, logaritmi, percentuali, integrazioni ed approssimazioni.
Il « Plan Manager » permette l'allineamento dei titoli, permette di variare la larghezza della colonna; dispone della funzione di replicare, copiare e cancellare.
I rapporti finali possono essere stampati sia su stampante di serie che su stampante RS-232, ed è disponibile anche la funzione grafici a barre.
Insieme all'XL 80 è provveduto un pacchetto di software con programmi « Utility » per RS-232 e duplicazione dischi.

* *Funziona con Commodore 64, con disk-drive 1541 e con qualsiasi monitor b/n.*

TRANSIMAGE INTERNATIONAL

L'XL 80 è prodotto negli Stati Uniti ed è importato e distribuito esclusivamente dalla
TRANSIMAGE INTERNATIONAL Srl - Computer Division.
V.le Umberto Tupini 103, 00144 Roma (Eur) - Italia
Tel. (06) 59.18.846 TLX 612619 TI ROMA I

QUOTAZIONI sumus

Materiale nuovo imballato

CENTRO ASSISTENZA SPECTRUM

SUMUS s.r.l.
Via S. Gallo 16/r
50129 Firenze
tel. 055/29.53.61
tlx. 57.10.34

Computers Apple compatibili

Lemon II 64K	
Lemon II 64K con Z-80	
Lemon II 64K con Z-80 compatto, con unità a disco incorporata	
Mouse IA 64K	649.000
Mouse IC 64K con Z-80	754.000
Mouse IIA 64K tastiera separata	845.000
Mouse IIC 64K con Z-80 tastiera separata	972.000

Altri computers

Sharp MZ-721 con registratore e programmi in omaggio	529.000
Spectrum 16K	276.000
Spectrum 48K	369.000
Dragon 32K	419.000
Dragon 64K	589.000
Commodore	telefonare
Atari 800XL con tavoletta grafica	telefonare
Sanyo MBC 550 128K, hires, colore, drive da 160K, 16 bit, MS-DOS, ecc. (è la cosa più bella e conveniente che potete trovare alla SUMUS!)	2.099.000
Aquarius	126.000
ZX-81	84.000
Oric 1 48K	338.000
Spectravideo	telefonare
Olivetti M10 24K	1.399.000

Accessori Apple II o compatibili

Sistema grafico a colori per penna ottica, corredato di un completo programma applicativo	335.000
Modem/accoppiatore acustico	259.000
Joystick professionale metallico	37.000
Modem per linea telefonica con auto/answer	126.000
Disk drive standard 5'' 1/4	338.000
Disk drive slim	388.000
Base a snodo per monitor 12''	35.000
Programmatore di eprom (2716/32/64)	99.000
Inaterfaccia RS-232 con cavo	79.000
Buffer di stampa 16K	209.000
Scheda di espansione +128K	350.000
Scheda A/D	125.000
Scheda PAL (per TV)	99.000
Scheda RGB (per monitor a colori)	99.000
Music card	109.000
Sinterizzatore vocale	69.000
Scheda orologio/calendario	99.000
Floppy disk controller	75.000
Scheda 80 colonne	165.000
Scheda CP/M	99.000
Scheda interfaccia Centronics	79.000
Idem tipo Grappler	99.000
Sistema grafico plotter Strobe	1.100.000
Altre schede speciali a richiesta.	

Il NEGOZIO DI SUPER SUMUS!

Stampati

Alphacom 32 per Spectrum	169.000
Stampante Mannesmann Tally MT-80	telefonare
Stampante colori 120 cps, 136 colonne, carta larga, letter quality, grafica	1.937.000
Ampio assortimento - aghi - margherita - macchine per scrivere già interfacciate	

Altre novità e varie

Monitors Hantarex colori e monocromatici	telefonare
Espansioni RAM e 48K per Spectrum	67.000
Portadischi da 10	5.084
Portadischi da 100	33.050
Registratore compatibile Commodore	50.000
Registratore originale Commodore	99.000
Floppy disk 5'' doppia faccia doppia dens	3.389
Joysticks - ampio assortimento	
Sconto 33% su libri inglesi per Spectrum!	
Interface 1	151.000
Microdrive	151.000
Floppy A5'' 1/4 con i/f per Spec trum, interfacciato	542.000
Interfaccia joystick Protek	26.000

Software

Cassette «Ultimate» originali titoli vari	10.170

NUOVA SUCCURSALE
Via Statuto, 17/R - Firenze

MERAVIGLIOSO ASSORTIMENTO DI COMPUTERS (BASI E CARTUCCE DI TUTTE LE MARCHE) - LIBRI - PROGRAMMI ACCESSORI - NON POSSIAMO ELENCARE TUTTO - VENITE A VISITARCI!

Condizioni:
Tutti i prezzi non comprendono l'IVA.
Disponibilità e prezzi variano frequentemente. Telefonateci prima dell'ordine o prima di venire.
La merce è resa franco ns. negozio. Imballo gratis. Lunedì mattina chiuso.
Pagamento anticipato a mezzo di vaglia o assegno. Le spese di spedizione sono addebitate in contrassegno.