Parla più FORTH

di Raffaello De Masi

Settima parte

Variabili, costanti ed array

Finora abbiamo sempre utilizzato, per maneggiare ed utilizzare valori, lo stack; la filosofia e la grande potenza del FORTH si basa appunto sull'uso di questa struttura il cui accesso è oltremodo rapido. Talora, però, anche utilizzando tali strutture, risulta difficile risolvere complesse situazioni numeriche. Uno per tutti, valga l'esempio di utilizzo di un certo valore diverse volte in un programma od in una definizione, per conservare il quale dovremmo ricorrere a complicate manipolazioni dello stack. Appare allora conveniente potersi riferire a certi valori chiamandoli con un nome, cosa che ci consente di evitare lo sforzo di rammentare il loro effettivo valore. È il caso classico dello statement LET del Basic, solo che, in Forth, la cosa è leggermente diversa.

Premesso che le modalità operative interne sono, in Basic come in Forth, praticamente le medesime (vale a dire, viene scelta una locazione, ad essa va assegnato un nome, ed in essa viene immagazzinato un valore) nel nostro linguaggio esiste una differenziazione che nel Basic non c'è. In Forth, infatti, è possibile utilizzare per l'incasellamento dei dati due tipi strutturali diversi: le costanti e le variabili. La differenza tra le due è che la prima rappresenta un valore, inserito in una locazione, che rimane generalmente sempre eguale a se stesso (sebbene possa essere, con qualche difficoltà, cambiato) mentre la seconda è costituita da una o più locazioni consecutive, in possesso di un nome, che possono accogliere, volta per volta, valori diversi, con contenuto, appunto, variabile.

In queste puntate parleremo delle modalità di assegnazione e di manipolazione dei valori con le costanti e le variabili, nonché possibilità di preparare array mono e pluridimensionali.

Le Costanti

Identificare una costante significa praticamente assegnare un nome od un numero, secondo la forma

```
n CONSTANT nome
```

in cui nome è una sequenza di caratteri alfanumerici (senza le limitazioni del Basic e del Pascal) che, generalmente, non può essere lunga più di 31 caratteri.

Ricordiamo, per inciso, come sia sempre possibile ridefinire una word, per cui, a chi non piacesse utilizzare il termine inglese, è sempre possibile eseguire

```
: CONSTANT COSTANTE ;
```

per poter utilizzare la forma

```
n COSTANTE nome
```

ad esempio la sequenza

```
141 CONSTANT MCMICRO
```

consente di conservare il CAP della redazione nella variabile nominata.

A tutti gli effetti si tratta di una vera e propria definizione; infatti

```
: MCMICRO 141 (si sottintende 00) ;
```

consente lo stesso risultato, ma impiega più memoria o più tempo ad essere eseguita.

Una volta assegnata, una costante mantiene generalmente sempre lo stesso valore o significato. È la regola fondamentale, che differenzia queste dalle variabili (si ricordi che, con le costanti, non esistono coinvolgimenti relativi od indirizzi di memoria, ma solo un rapporto diretto definizione-valore, il che giustifica la loro maggiore rapidità); potrebbe però succedere che per un qualsiasi motivo occorra variare tale valore. Ciò richiede la seguente procedura

```
n1 ' nome !
```

Per esempio, per inserire il numero civico della redazione al posto del primo avremo

```
135 ' MCMICRO !
```

Questo determina tre passaggi: l'inserimento del numero dello stack, la ricerca dell'indirizzo della parola MCMICRO e l'inserimento del valore 135 al suo posto.

Le Variabili

Per definire una variabile in FORTH-79 basta scrivere la word VARIABLE seguita dal nome della variabile stessa.

Rifacendosi all'esempio precedente

```
VARIABLE    MCMICRO
```

registra il nome MCMICRO nel dizionario e riserva 2 byte destinati ad accogliere l'eventuale numero da associare ad esso. È importante ricordare che, in questo momento, tale spazio contiene un valore indefinito (e non zero). Per assegnare di nuovo il numero di CAP alla definizione avremo

```
141  MCMICRO  !
```

In FIG-FORTH la cosa è più semplice. In effetti inizializzazione di variabile ed assegnazione di valore sono contemporanei, con la forma

```
n  VARIABLE  nome
```

e nel nostro caso

```
141  VARIABLE  !
```

che abbina gli effetti delle due precedenti definizioni in FORTH-79. Come già evidenziato precedentemente, esiste una grande differenza tra il modo operativo delle costanti e delle variabili. Chiamando le prime, si ottiene direttamente il valore, trattandosi, per certi versi, di sequenze di operazioni simili alle word. Vale a dire che mentre il richiamo ad una costante fornisce direttamente il risultato mettendolo in TOS

```
MCMICRO     (MCMICRO definito come CONSTANT)
            (l'esecuzione lascia in TOS il valore 141)
MCMICRO     (MCMICRO definito come VARIABLE)
```

lascerà in TOS l'indirizzo del valore 141. Per sapere effettivamente il valore di MCMICRO occorrerà battere

```
MCMICRO     @ (lascia il valore in TOS)
```

oppure

```
MCMICRO     @ DUP .
```

oppure

```
MCMICRO     ?
```

qualora il valore voglia essere contemporaneamente visualizzato e conservato in TOS.

Una utile applicazione delle variabili è quella, come succede spesso nel controllo di strumentazione, di funzionare da contatori numerici. Ad esempio, una macchina contatrice potrebbe essere così regolata (in FORTH-79)

```
VARIABLE  CONTATORE     (definisce la variabile)
                        (evidentemente solo la prima volta)
0         CONTATORE !   (inizializza CONTATORE ad ogni valore od oggetto da contare fornito da un input; lo vedremo la prossima puntata).
DUP       BEGIN       1
WILE      CONTATORE     (inserisce in stack l'indirizzo della variabile)
+!                      (aggiunge 1 al valore presente all'indirizzo)
REPEAT
DROP      CONTATORE c ."ho contato". ."oggetti"
```

Le Array

Ad onor del vero non so se l'articolo femminile plurale va bene o no! Le buone regole della grammatica italiana, che la mia maestra cercava di impormi a bacchettate sulle ginocchia inorridiscono, come inorridiscono di fronte alle finezze di far terminare il plurale delle parole inglesi con la s, dimostrando buona conoscenza delle lingue straniere e scarsa di quella italiana.

Comunque le array, come anche il più bolso dei programmatori sa, sono un utile mezzo di raccolta, in maniera ordinata, di dati che hanno, per così dire, una matrice od un riferimento comune e conviene pertanto incasellare, anzi, schierare in un unico gruppo. Ad esempio immaginiamo che Marinacci, il despota, voglia tenere conto di tutti i compensi (magri, anzi magrissimi), elargiti, durante l'anno, agli instancabili collaboratori. Secondo quanto visto finora dovremmo dimensionare tante variabili quanti sono i collaboratori; è possibile, però, utilizzare un gruppo di variabili, facenti capo allo stesso capoverso cui far riferimento in maniera più rapida e, comunque, che occupi meno spazio in memoria.

Una array, quindi, può essere definita come un raggruppamento logico di variabili identicamente conformate. Ad esempio, immaginando che ogni articolo venga compensato con una cifra variabile tra le 500 e le mille lire (non ridete), sarà sufficiente che ogni variabile sia formata di due byte (senza segno, ovviamente) con un valore massimo immagazzinabile annuo di 65535 (pura illusione). Per 10 collaboratori sarà pertanto necessario disporre di 20 byte riservati, due byte per ciascun collaboratore.

Cominciamo col definire la variabile con un nome. COMPENSI (o magari MISERIE) va bene come un altro.

Una volta definita

```
VARIABLE COMPENSI
```

esisterà in memoria il nome COMPENSI con, a disposizione, due byte di spazio riservati a tal nome (figura 1).

La variabile COMPENSI ha, però, necessità, come abbiamo visto, di 20 byte. È possibile espandere, per così dire, la variabile COMPENSI aggiungendovi altri 18 byte. La procedura per eseguire ciò, piuttosto semplice, è

```
18 ALLOT
```

eventualmente eseguita tutt'insieme con la fase precedente.

A questo punto occorre spezzettare tale grossa variabile in 10 più piccole di cui fare uso. È possibile, pertanto far riferimento ad ogni casella utilizzando l'indirizzo di partenza ed aggiungendo ad essa il numero di byte necessario.

Poiché questo numero è funzione del nome del collaboratore cominciamo ad associarli fra di loro nel modo

```
0 CONSTANT ANGELETTI
2 CONSTANT BERGAMI
4 CONSTANT DE MASI
6 CONSTANT DE PRISCO
```

e così via, utilizzando le definizioni di costanti visto che si tratta di valori non destinati a cambiare e vista la notevole rapidità d'intervento di tale definizione.

A questo punto la cosa diviene molto semplice. Immaginiamo di voler inserire il numero 100 nella casella riservata a Giustozzi; avremo

```
100 COMPENSI GIUSTOZZI + !
```

che sarà seguito dall'imperturbabile OK.

Se, in qualsiasi momento, vorremmo addizionarvi un altro valore, ad esempio 235 basterà battere

```
135 COMPENSI GIUSTOZZI + +!
```

COMPENSI (nome)	COMPENSI (nome)	COMPENSI (nome)
(ind) 2 byte	(ind) 2 byte	ANGELETTI (loc. all'ind.)
	30 byte aggiunti da ALLOT	BERGAMI (ind + 2) DE MASI (ind + 4) DE PRISCO (... ecc) DI DIO GALASSETTI GIUSTOZZI MARZOCCA MORANDO PANTUSO PANUNZI PETRONI PRINCIPI SCHIATTARELLA SORGE TASSO
FINE ARRAY	FINE ARRAY	

Figura 1 - Processo di costruzione della array "COMPENSI".

e per leggere gli emolumenti ricevuti

```
COMPENSI GIUSTOZZI + ?
```

Per array molto estese (per esempio quando la Technimedia diventerà una multinazionale ed avrà 500 collaboratori) diviene impratico definire una costante per ogni posto (sia questo uno o più byte), anche perché ciò va a scapito, comunque della quantità di memoria a disposizione. In questi casi è molto meglio, anche se un po' meno pratico, far riferimento ad un determinato numero indicandolo con il suo valore di scartamento rispetto al valore iniziale di indirizzo.

Per esempio, utilizzando il caso precedente avremo

```
VARIABLE COMPENSI  (definisce la variabile compensi)
499 2 * ALLOT  (e costruisce una array di 500 posti di due byte ognuno).
```

Questa nuova versione di COMPENSI contiene una array di 500 posti di due byte ciascuno e, insieme, ogni numero (due byte) o, per meglio dire, ogni allocazione di esso ha un indirizzo che è due volte più grande del numero che lo precede. Così Angeletti è allocato all'indirizzo iniziale, Bergami all'indirizzo +2, ecc. e l'improbabile collaboratore Zuzzerelloni all'indirizzo +1000. La forma generale del valore di scartamento all'indirizzo iniziale è:

Valore = 2 × n - 2

dove n è il numero di matricola progressivo assegnato al collaboratore.

A questo punto, per inserire il valore 50 al collaboratore n. 375 batteremo

```
50 COMPENSI 375 2 * 2- + !
```

e per leggere il valore corrispondente al collaboratore 175

```
COMPENSI 175 2 * 2- + ?
```

Alcune array, usate generalmente, completamente o in parte, come strutture di dati iniziali hanno, ad esempio, bisogno, all'inizio di un programma, di essere inizializzate ad un determinato valore. In Basic veniva per lo più usata la struttura READ... DATA, non scevra di possibilità d'errore. In Forth, oltre che la semplice assegnazione uno ad uno dei valori di array (assurdo se si pensa ad array di alcune centinaia di posti) è possibile utilizzare una struttura diversa, che fa capo alla word [,] (le parentesi quadre non c'entrano, sono state qui messe solo per non confondere la word con un segno di interpunzione).

Ad esempio, se il solito Marinacci volesse tenere conto, per lesinare la lira ed evidenziare al terrorizzato e sprovveduto collaboratore, che ingenuamente chiede un aumento, i compensi già percepiti, all'inizio dell'85 avrebbe bisogno di immagazzinare, nelle caselle dell'array COMPENSI, i denari già percepiti immeritatamente dagli schiavi. La word [,] va così utilizzata

```
VARIABLE COMPENSI !
1500 ANGELETTI ! (inizializza la variabile al compenso già percepito da Angeletti).
1200, 600, 400, 1300,
350, 1750, 720, 1485, (e così via)
```

Per essere più precisi, la word [,] è una combinazione delle word ALLOT e !. Essa riserva due byte nel dizionario così come 2 ALLOT e conserva il numero in TOS (Top Of Stack) in questi due byte, come fa la word !. Fin qui tutto chiaro, per quanto possa essere chiaro un divulgatore di mezza tacca quale il buon De Masi, in barba al detto del mio vecchio professore di fisica terrestre che asseriva che quando uno parla e l'altro non capisce, chi parla, la cosa o non la sa o non te la vuol dire!

E fin qui, fintanto che le array da creare od inizializzare sono due o tre, la sequenza VARIABLE - ALLOT va pure bene. Ma, quando le array cominciano a diventare più di una, potrebbe essere più conveniente automatizzare la procedura.

Sto scrivendo questo articolo il giorno della vigilia di Natale e le montagne di Avellino non promettono niente di buono, pioggia sicura, neve probabile. E non promette niente di buono neppure mia moglie che sta decidendo di cambiare l'arredamento in camera da letto. Come potrò fare a conservare l'elenco di tutte le spese di questo santo giorno (santo anche, prevedo, per l'arredatore con cui discute mia moglie) ordinandole secondo voci diverse?

Occorrerebbe creare diverse array e, visto come si sta esaurendo, precipitosamente, il libretto degli assegni sarebbero veramente molte. Con VARIABLE la cosa è certo possibile, ma sarebbe molto più con-

veniente avere una word che creasse automaticamente una array.

Occorrerebbe avere una word, ad esempio proprio ARRAY o magari MATRICE per dirla in italiano, che accetti due parametri, un valore ed un nome ed utilizzi questi per aggiungere appunto una array al dizionario. Per esempio

```
25 ARRAY SPESEDINATALE
```

dovrebbe creare una schiera, anzi un vettore di 50 byte dal nome SPESEDINATALE. In Forth esiste una sequenza, costituita da due word, CREATE e DOES> (in Fig Forth <BUILDS e DOES>) che consente l'operazione (tanto per intenderci come il DIM del Basic). Usato da solo, CREATE inserisce un nome nel dizionario, senza però allocare spazio di memoria; sarebbe come dire che crea una array indeterminata (in pratica funziona come VARIABLE ma senza lo spazio dei due byte successivi). Al contrario la combinazione di CREATE e DOES> 1 in una definizione del tipo

```
: nome CREATE ... DOES> ... ;
```

crea una nuova struttura di determinato nome. Facciamo un esempio: si voglia creare una array di 24 numeri. Definiremo inizialmente la word MATRICE che crea appunto una array di dimensioni prestabilite con un determinato nome. Definiremo inizialmente

```
: ARRAY
  (definizione di una array avente spazio per
  n numeri (n = TOS) usando la forma generale : n ARRAY nome  in aggiunte, la word
  cerca l'indirizzo di un elemento inserendone il numero nello stack)
  CREATE (inserisce il nome nel dizionario)
  2 * ALLOT (effettua la relativa allocazione
          di 2n byte)
  DOES>
  SWAP (scambia il numero degli elementi
       e l'indirizzo di base)
  2 *
  + . (indirizzo = n°elementi × 2 + indirizzo base)
  ; (fine definizione)
```

In questo modo la sequenza

```
25 ARRAY SPESEDINATALE
```

crea la nostra array e

```
1200 8 SPESEDINATALE ! OK   (inizializza l'elemento 8 è valore 1200)
8 SPESEDINATALE ? 1200 OK   (mostra il valore contenuto nell'elemento 8)
2 8 SPESEDINATALE +! OK     (aggiunge 2 all'elemento 8)
8 SPESEDINATALE ? 1202 OK   (mostra il nuovo valore dell'elemento 8)
```

Un'ultima forma di array può essere considerata una tabella. Questa è utilizzata per sostituire operazioni che impiegano lunghi periodi di tempo, come il calcolo dei valori dei logaritmi od i seni e coseni di una sequenza di angoli. Le tabelle sono molto utili ed efficienti quando la funzione è limitata ad un ridotto valore di argomenti, ed elimina la necessità di calcoli ripetitivi ogni volta che si ha bisogno della funzione. Il risvolto della medaglia è dato dalla grande quantità di memoria utilizzata.

TABLE è una word che va così definita:

```
: TABLE (o TABELLA se si preferisce)
        (la word crea una tabella in cui i valori vanno aggiunti con la virgola inoltre la ricerca del valore la si ottiene con la sequenza n ° elemento nome)
  CREATE
  DOES>
  SWAP (scambia l'elemento e l'indirizzo di base)
  2 * (slitta di 2 × n ° elemento)
  + (il nuovo indirizzo = indirizzo di base + slittamento)
  @ (ricerca il valore all'indirizzo)
```

Quale applicazione di quanto finora detto, Leo Scanlon presenta un suo programma, in FORTH PROGRAMMING, per la ricerca dei valori del seno degli angoli compresi tra 0 e 360 o già inseriti in una tabella. Il procedimento utilizzato è estremamente semplice e pulito ed è veramente un gioiello di programmazione. Noi ne riportiamo qui la sola definizione

```
: SIN
  (fornisce il valore del seno di un angolo compreso tra 0 e 360 gradi. Il risultato va diviso per 10.000 per la ben nota assenza, nel FORTH classico, dei numeri decimali)
  (angolo — seno)
  DUP 270 >
  IF
  360 SWAP - (per valori tra 271 e 360 il seno
             = -seno (360-x))
  SINE-TABLE NEGATE
  ELSE
  DUP 180 >
  IF
  180 - (per valori tra 181 e 270 seno = - seno
       (x - 180))
  SINE - TABLE NEGATE
  ELSE
  DUP 90 >
  IF
  180 SWAP - (per valori tra 91 e 180)
           (seno = -seno (180-x))
  THEN
  SINE-TABLE (tra 0 e 90, calcola)
  THEN
  THEN ;
```

La definizione va preceduta, una volta per tutte, dalla creazione della tabella SINE-TABLE, nel modo

```
TABLE SINE-TABLE
0, 175, 349, 523 (seni tra 0 e 3/10.000)
698, 872, 1045,
....
....
9986, 9994, 9998, 10000 (seni tra 87 e 90/10000)
```

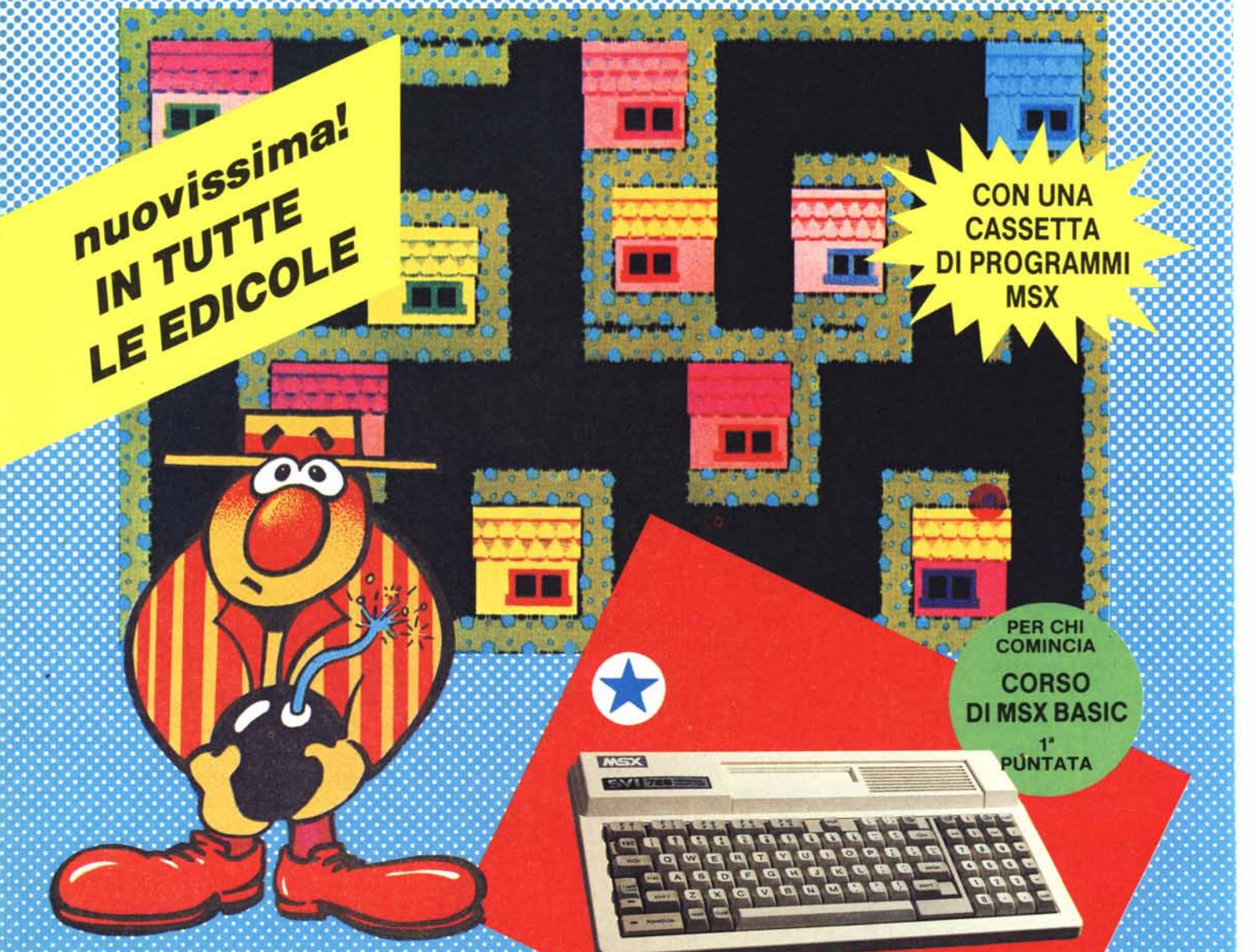
Se state leggendo queste parole e non avete pensato che è più semplice battere in Basic PRINT SIN (30), beh potete anche dire a vostra moglie che De Masi non vi considera più programmatori della domenica. Per chi, invece, lo è proprio della domenica a mezzogiorno e non capisce perché uno debba così avvelenarsi la vita, provi a calcolare o cercare in una array già inizializzata un certo valore in Basic, in Fortran od in Pascal, e poi lo faccia in FORTH ed il tutto lo si faccia 10 o 20.000 volte. Per valutare la differenza non serve un cronometro, basta la cipolla del nonno. Provare per credere. MC

SPECIALE

MSX COMPUTER MAGAZINE

N. 1/1985

Sped. in abb. post. Gr. III L. 9.000



nuovissima!
IN TUTTE
LE EDICOLE

CON UNA
CASSETTA
DI PROGRAMMI
MSX

PER CHI
COMINCIA
CORSO
DI MSX BASIC
1ª
PUNTATA

Spectravideo
SVI 728

MSX SPRITE STORY

PHILIPS UNA MACCHINA TUTTA EUROPEA

IL PRIMO ARCHIVIO ELETTRONICO

ISTRUZIONI: TUTTE LE EQUIVALENZE