SUMMER TRIP

MSX COMPUTER MAGAZINE

N.8 MAG/GIU 1986

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

10 PROGRAMMI SU CASSETTA

★ TAPE SOFT

★ CORSO DI LINGUAGGIO MACCHINA

★ PIXEL COPIER

MILKO MRSEK

PER IL TUO COMPUTER
GIOCHI E UTILITY SU CASSETTA!

Se hai lo spectrum

in edicola

Se non trovassi le raccolte in edicola, chiedile direttamente inviando esclusivamente vaglia postale ordinario di Lire 10mila ad Arcadia srl, c.so V. Emanuele 15, Milano specificando ciò che vuoi ed i tuoi dati chiari e completi.

l'Hardware

Raccolta Speciale

commodore 64

UNA FANTASTICA COMPILATION

MSX Computer Magazine è edita da Arcadia srl,
C.so Vitt. Emanuele 15, Milano.
Tel. 02/706329 (solo giovedì h. 15-18).
Una copia L. 9.000.
Fotocomposizione: Composit.
Stampa: Garzanti,
Milano. Distribuzione: SO.DI.P. Angelo
Patuzzi srl, Via Zuretti 25, Milano.
Registrato Trib. Milano N. 52 del 2/2/85.
Resp. Sira Rocchi.
Sped. in abb. post. Gr. III/70.

Manoscritti, disegni, fotografie
e programmi inviati non si
restituiscono anche se non pubblicati.

MSX COMPUTER MAGAZINE 8

IN QUESTO NUMERO

W L'ESTATE!

★ IMPARA A PROGRAMMARE

★ PIXEL COPIER

★ SERENO VARIABILE

★ A CACCIA DI DATI

PIÙ 10 PROGRAMMI 10!!

- GOLF
- CASTLE
- RIBOT
- LA CITTÀ LABIR.
- ITALTOUR
- DUE SEDUTTORI A PARIGI
- PARK 1
- MUSICA
- CODICE MORSE C.W.
- ISTOGRAMMI

UNA MAGLIETTA IN REGALO!

a chi si abbona a

MSX COMPUTER MAGAZINE

✸

sei magnifiche cassette di programmi di gioco e di utilità, sempre più belle e ricche!

✸

il prezzo dell'abbonamento (Lire 50 mila) è bloccato per sei numeri e non ti verranno quindi richiesti aumenti (già subito intanto risparmi 4 mila lire)!

✸

avrai subito, direttamente a casa, un'elegante maglietta (realizzata con le riviste consorelle Elettronica 2000 e Load'n'Run) assolutamente gratis!

ABBONATI OGGI STESSO

Basta inviare un vaglia ordinario (quello rosa, da richiedere in un qualunque ufficio postale) di lire 50 mila.
Indica esattamente da quale fascicolo desideri l'abbonamento ed i tuoi dati chiari e precisi. Indirizza a MSX Computer Magazine, C.so Vitt. Emanuele, 15 - 20122 Milano.

LETTERE

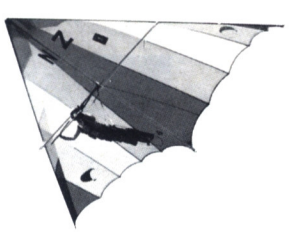
IL VIDEO IN MEMORIA

Sono un fortunato possessore di un computer MSX: il Sony Hit-Bit. Purtroppo i testi in circolazione riguardanti i sistemi MSX sono pochi. Conosco l'assembler Z80 e vi chiedo: come posso trasferire l'area del video alla memoria centrale e viceversa?

Attilio Milani - Ivrea

La soluzione al tuo problema è semplice perché esistono delle routine in ROM che ti permettono di trasferire blocchi di VRAM. Esse hanno indirizzo di CALL alla posizione 0059H e 005CH.

La prima routine (0059H) trasferisce n bytes, dove n è il contenuto del registro BC, dalla posizione i, dove i è il contenuto



del registro HL, alla posizione f, indicata da DE, in memoria utente. La seconda routine, con start all'indirizzo 005CH, permette di trasferire i dati dalla memoria utente alla memoria video (VRAM). I parametri di questa subroutine sono:

HL = start dati in memoria utente
DE = destinazione dati in VRAM
BC = lunghezza blocco dati da trasferire.

LA MEMORIA NON BASTA

Ho trovato da un amico un numero vecchio della rivista, con i programmi Slalom e Corri ma...

Maurizio Rettaroli - Perugia

Non riesci a caricare i due programmi. "SLALOM" e "CORRI". Semplicemente perché il sistema che possiedi (VG-8000) non dispone di sufficiente memoria per accogliere i programmi in questione. Praticamente hai solo 16K utente (in verità la memoria libera è meno, circa 12K) mentre "CORRI" e "SLALOM" girano

su sistemi MSX con almeno 32K utente. Quindi l'unico modo per vedere i due giochi è quello di espandere il computer con una memoria RAM aggiuntiva di 16K per raggiungere così i trentadue K!

CON LA LIGHT PEN

Dispongo di software con pagine grafiche molto belle. Vorrei farne una copia su carta. I normali sistemi di hard copy non funzionano...

Rino Lenzi - Arezzo

Forse le tue pagine video sono state preparate con una Light Pen.

I cinquantadue passi di programma qui riportati sono un esempio pratico di come è possibile richiamare da Basic una pagina video costruita con la Light Pen Unit Sanyo MLP-001. Per notizie tecniche più dettagliate contattare la Sanyo in via Vittorio Veneto 22, Milano.

```
10 '*** SCREEN LOAD ***
20 CLEAR 100,&HEAC0:DEF USR=&HEAD0
30 FOR I=&HEACF TO &HEC6D
40 READ A$:POKE I,VAL("&H"+A$)
50 NEXT I
60 POKE &HF3C8,&H38:POKE &HF3CA,&H20
70 POKE &HF3CC,&H0:POKE &HF3CE,&H1C
80 POKE &HF3D0,&H18
90 SCREEN 2,1:CLS:A=USR(X)
100 DATA 00,CD,40,EC,00,00,00,CD,FC,EB
110 DATA CD,59,EB,00,00,00,01,0D,0A,CD
120 DATA 69,EB,B9,20,F1,10,F8,21,B0,FF
130 DATA E5,06,06,CD,69,EB,77,23,10,F9
140 DATA E1,11,B6,FF,06,06,1A,13,FE,20
150 DATA 20,04,10,F8,18,0D,11,B6,FF,06
160 DATA 06,1A,BE,00,00,23,13,10,F8,00
170 DATA 00,00,00,00,00,18,0B,11,C7,EB
180 DATA CD,94,EB,CD,59,EB,18,B8,CD,59
190 DATA EB,CD,69,EB,00,00,00,CD,69,EB
200 DATA 32,D3,FF,CD,69,EB,32,D4,FF,CD
210 DATA 30,EC,21,00,00,11,FF,3A,CD,F1
220 DATA EB,CD,69,EB,D3,98,E7,30,03,23
230 DATA 18,F5,CD,E7,00,C9,00,00,F5,C5
240 DATA D5,E5,CD,E1,00,00,00,00,E1,D1
250 DATA C1,F1,C9,00,C5,D5,E5,CD,E4,00
260 DATA E1,D1,C1,C9,CD,E7,00,C3,F3,EB
270 DATA CD,52,EC,11,B6,EB,CD,9E,EB,3A
280 DATA B6,FF,00,00,00,01,05,00,11,06
290 DATA F4,21,B1,EB,ED,B0,C9,CD,9E,EB
300 DATA 3A,B0,FF,00,00,00,C9,06,01,21
310 DATA BC,FF,CB,6E,21,18,16,28,03,21
320 DATA 04,16,00,00,00,C9,25,2D,0E,16
330 DATA 1F,09,46,C9,6C,65,6E,61,6D,65
340 DATA 3A,06,46,6F,75,6E,64,3A,06,53
350 DATA 6B,69,70,20,3A,CD,84,44,CD,53
360 DATA 52,21,BC,FF,CB,6E,21,01,01,28
370 DATA 03,21,05,01,CD,E9,E1,CD,A1,44
380 DATA CD,40,45,CD,B3,55,31,00,F3,C9
390 DATA F5,7D,D3,99,7C,F6,40,D3,99,F1
400 DATA C9,3E,0E,D3,99,3E,82,D3,99,3E
410 DATA FF,D3,99,3E,83,D3,99,3E,03,D3
420 DATA 99,3E,84,D3,99,3E,38,D3,99,3E
430 DATA 85,D3,99,3E,03,D3,99,3E,86,D3
440 DATA 99,C9,00,00,00,00,00,00,00,00
450 DATA 00,00,00,F5,3A,D4,FF,D3,99,3E
460 DATA 87,D3,99,F1,C9,00,00,00,00,F5
470 DATA AF,D3,99,3E,60,D3,99,0E,30,CD
480 DATA 60,EC,0D,20,FA,00,F1,C9,00,00
490 DATA 00,00,00,00,00,00,00,00,00,00
500 DATA 00,F5,C5,06,80,AF,D3,98,00,00
510 DATA 10,FA,C1,F1,C9,00
520 GOTO 520
```

MSX TAPE SOFT

I programmi che vi presentiamo in questo ottavo numero di MSX Computer Magazine sono tutti compatibili con qualsiasi sistema MSX. Ecco per voi ben 10 programmi.

Ricordate di collegare la spina del controllo motore alla presa REM del vostro registratore, se quest'ultimo la possiede. Assicuratevi che la spina

rossa sia collegata alla presa MIC del registratore e la spina nera alla presa EAR.

Se il vostro mangiacassette non possiede la presa REM, fate particolare attenzione a quando un programma è stato caricato o deve essere caricato, affinché il nastro scorra per il giusto tempo. Appena vedete apparire sul video, dopo un comando di ca-

1

GOLF
("GOLF")
di R. Bruni

Il gioco consiste nel mandare in buca la palla cercando di evitare gli ostacoli disposti ogni volta in modo casuale.

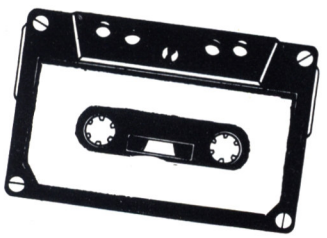
Ad ogni buca si cambia di livello (primo riquadro in alto a destra), aumentano gli ostacoli e, in proporzione, anche la quantità di tiri a disposizione (secondo riquadro in alto a destra) terminati i quali, si perde una vita (terzo riquadro in alto a destra). Anche mandando la palla fuori dal campo si perde una vita. Il punteggio che si guadagna ad ogni buca è proporzionale al numero di livello, pertanto più è alto e più punti ci sono a disposizione. Anche il numero dei tiri influisce sul punteggio, infatti meno tiri si fanno più punti si guadagnano. Ogni 10.000 punti si guadagna una vita.

Per giocare si può usare sia la tastiera che il joystick.

Il movimento a destra o a sinistra mette l'omino in posizione di tiro, mentre il movimento in alto o in basso modifica la lunghezza del tiro, il quale si effettua premendo il tasto fire o la barra spaziatrice. Il disegno della schermata iniziale è dalla linea 20 alla linea 260.

ricamento, la scritta OK, spegnete il registratore.

ATTENZIONE: tutti i programmi vanno caricati con il comando CLOAD (i caratteri racchiusi tra virgolette, nelle parentesi, rappresentano il nome con cui è stato registrato il programma) ed eseguiti con l'istruzione RUN. Per caricare il programma "Due seduttori a Parigi" seguite le istruzioni nell'apposito riquadro.



Nella cassetta allegata a questo fascicolo troverete sul lato A:

Golf, Castle, Ribot, La città labirinto, Italtour.

Sul lato B:

Due seduttori a Parigi, Park 1, Musica, C.W. codice Morse, Istogrammi.

RIBOT

("RIBOT")

di R. Policastro

Le fantastiche prestazioni grafiche dell'MSX, in particolare la gestione degli sprite, consentono di realizzare in Basic simpatici videogiochi che su altri computer necessiterebbero di routine in linguaggio macchina. In questo programma, infatti, la maggior parte della grafica è realizzata con gli sprite.

Il gioco si ispira al mondo dell'ippica, in particolare al famoso cavallo "Ribot". Vi trovate nel campo di gara a diversi ostacoli: paletti, paletti doppi, sedie, paletti con l'acqua.

I paletti doppi vengono saltati velocemente, gli altri più lentamente. Se la velocità è troppa il cavallo cade, se è poca si ferma. Quando ci si trova davanti a due ostacoli il segreto per superarli consiste nel cambiare velocità mentre state saltando il primo. Se superate i 20 falli dovete ritirarvi dalla gara, altrimenti, se superate tutti gli ostacoli (non è facile!) entrate in classifica.
Tasti: cursore su = aumenta velocità; cursore giù = diminuisce velocità

CASTLE

("CASTLE")

adatt. da G. Riccobono

Dopo aver dato il RUN attendete alcuni secondi dopodiché vedrete apparire sullo schermo un labirinto con quattro fantasmi rossi ed un PAC-MAN giallo che dovrete guidare nel labirinto. Scopo del gioco è riuscire a mangiare il massimo numero di palline evitando i fantasmi, il cui contatto è letale. Se riuscite a mangiare la grossa mela al centro del labirinto ciò vi dà la possibilità, per qualche istante, di mangiare anche i fantasmi. State attenti che appena mangiate un fantasma se ne genera subito un altro che non potete uccidere: i fantasmi che potete mangiare sono di colore blu gli altri di colore rosso.

La scritta "LEFT", in alto sullo schermo, indica il numero di vite ancora a vostra disposizione. Una volta che finisce il gioco (appare la scritta "GAME OVER") potete ricominciare di nuovo premendo il tasto "S".

La maggior parte del programma è in linguaggio macchina. Le locazioni di memoria occupate dai codici macchina vanno dall'indirizzo &HD807 all'indirizzo &HDEC7.

4

LA CITTÀ LABIRINTO

("CITLAB")

di D. Montresor

Quello dell'archeologo è senza dubbio un mestiere affascinante, quindi voi che siete cittadini dell'impero galattico del 3000 d.C. avete pensato bene di studiare le antiche civiltà scomparse e le loro opere.

Su un pianeta appena scoperto è stata trovata una grande città in cui è impossibile entrare tranne che da una delle porte, perché protetta dall'alto da una cupola di energia.

Sembrerebbe cosa facile esplorarla ma gli antichi costruttori per impedire ai curiosi di poter entrare facilmente hanno disseminato il percorso di trabocchetti mortali.

Dovete raggiungere il centro e bloccare tutte le trappole per rendere possibile lo studio delle antiche costruzioni e poter così ritirare il premio stanziato dall'associazione archeologi.

Alla domanda del programma si può rispondere con un comando (es. nord, sud, verbi, ecc.) oppure con una frase (salgo a nord, prendo il laser, ecc.), è necessario scrivere sempre in prima persona.

Sono disponibili i seguenti comandi speciali:

VERBI: visualizza tutti i verbi o comandi che si possono impiegare.

SALVO: registra su nastro il punto in cui si è arrivati.

RECUPERO: carica da nastro un'avventura già incominciata.

PROLOGO: spiega qual è la situazione.

ISTRUZIONI: indica il modo di operare.

INVENTARIO: elenca gli oggetti in proprio possesso (max. 8).

All'inizio si hanno tre oggetti: il laser, il diagnosticatore e il rivelatore (con la frase "uso il rivelatore" si possono conoscere tutti gli oggetti presenti dentro la città).

Per alcuni comandi si possono usare i tasti delle funzioni già completi di return. Si può terminare il gioco scrivendo "FINE" e ripartire con l'istruzione CONT.

Per poter registrare e far girare il programma con interfaccia per disco inserita occorre eliminare i caratteri speciali con il comando DELETE 760-1980.

Buona avventura!

5

ITALTOUR

("ITALT")

di E. Medvet

6

DUE SEDUTTORI A PARIGI

("SEDUT1")

di N. Paggin

Siete a bordo della vostra Torpedo blu per guidarla in lungo e in largo per tutte le strade d'Italia. Dovete fare benzina ogni volta che arrivate al centro di una città scrivendo la targa esatta della sua provincia (per Roma scrivere RM).

Entrando in città (come anche uscendo) potete incontrare vari semafori che interrompono il vostro cammino: sta alla vostra abilità incontrarne il meno possibile onde evitare di perdervi nelle strade di periferia di ogni abitato. Il semaforo rosso indica che dovete prendere un'altra direzione, mentre quello giallo che avete imboccato una strada senza uscita. Dopo ogni pieno di benzina viene visualizzato il numero di chilometri percorsi.

Dopo il disegno della cartina d'Italia, appare il menu con 5 diverse opzioni che si scelgono con la pressione dei tasti numerici da 1 a 5. L'opzione 5 (PERCorsi) è quella sopra descritta. Le altre sono:

SCRivi 1 = scrivete la provincia in maiuscolo (es. MILANO) e il computer ve la indicherà con una freccia.

SCRivi 2 = il computer indica una provincia a caso e voi dovete scriverne il nome.

INDica 3 = voi indicate con una freccetta una provincia ed il computer scriverà il suo nome.

INDica 4 = il computer vi scriverà una provincia a caso e voi dovrete indicarla sulla cartina.

Per la scrittura delle province si usano le lettere maiuscole (si può usare anche Black Space) con l'enter alla fine di ogni nome di provincia. Per tutte le altre scritte (targhe, subopzioni) si usano le maiuscole senza l'enter.

Per ogni situazione in cui è previsto lo spostamento manuale degli sprite, si può scegliere l'uso del joystick o dei tasti cursore. Per indicare una provincia si sposta una freccia nera facendo combaciare la sua punta con il puntino indicante il capoluogo, indi si preme il tasto fire o la barra spaziatrice.

Nell'opzione 5 (percorsi) la pressione del tasto fire o spazio azzera il chilometraggio e riporta la macchina in garage.

Buon viaggio!

I due giocatori, per l'occasione in vacanza a Parigi, scommettono su chi dei due abbia più fascino... e quale miglior modo per saperlo se non corteggiando entrambi la stessa ragazza e vedendo poi chi otterrà i risultati migliori?

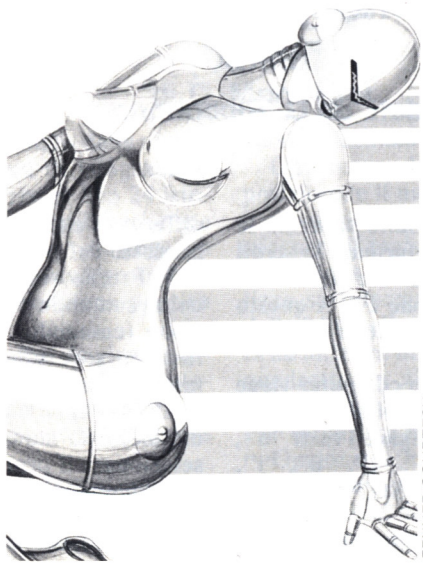
Il gioco si compone di 5 fasi:

Prima fase: i due giocatori decidono insieme quale delle quattro ragazze corteggiare comunicandolo al computer che provvederà a fornire loro una breve scheda della ragazza scelta.

Seconda fase: i due giocatori definiscono il proprio aspetto fisico. Importante ma affatto fondamentale per l'esito finale.

Terza fase: i due giocatori si preparano a portare fuori la ragazza. Scelgono cioè macchina, vestito, regalo, ecc. Attenzione a non spendere oltre le vostre possibilità. Al termine di questa fase uno dei due spasimanti viene scelto e ha la possibilità di portare fuori la ragazza. Ma questo non significa che l'altro sia tagliato fuori...

Quarta fase: questo è il nucleo centrale del gioco. Un seduttore porta la ragazza in 6 luoghi e in ciascuno di essi la ragazza fa dei commenti. Il seduttore ha a disposizione tre possibili risposte per ogni affermazione e deve cercare di scegliere la più adatta al carattere della ragazza. Se le risposte saranno quelle giuste allora tutto andrà bene ma se invece esse non si adattano al carattere della ragazza allora essa dimostrerà la sua disapprovazione e a questo seduttore si sostituirà l'altro, pronto comunque a subentrare nuovamente. In ogni posto c'è la possibilità per il seduttore in difficoltà di disturbare l'avversario scegliendo fra tre possibili azioni di disturbo.



FENNER COURTESY

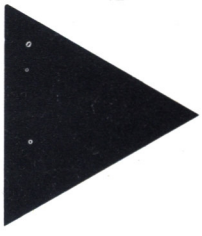
Quinta fase: un simpatico finalino mostrerà il punteggio raggiunto dal giocatore che è riuscito ad accompagnare la ragazza a casa e quindi vi saranno date le percentuali di gradimento per ognuno dei due seduttori. Non preoccupatevi per i bassi punteggi che otterrete all'inizio, è solo questione di conoscere meglio la ragazza!

ATTENZIONE:
Il programma è diviso per motivi di lunghezza in due parti. Per caricare la prima parte digitare CLOAD "SEDUT1". All'OK spegnere il registratore e dare RUN.

Successivamente il computer vi fornirà le istruzioni per proseguire il caricamento della seconda parte.

PARK 1

("PARK1")



di S. Loi

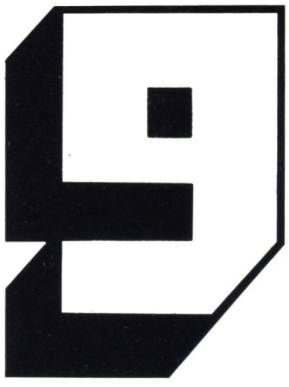
Il gioco si compone di tre livelli.
Primo livello: sotto un grande ombrellone verde un nastro trasporta delle anatre. Premendo la barra spaziatrice potrete ucciderle guadagnando, per ognuna di esse, 100 punti.

Attenzione all'omino giallo, se colpito vi toglierà 15 secondi dal tempo a vostra disposizione. Ogni 500 punti l'anatra centrale diventa rossa, in questo caso se si riesce a colpirla i punti guadagnati saranno 600. Per passare al livello successivo dovrete totalizzare 4000 punti.

Secondo livello: lanciate il martello sulla palla azzurra premendo la barra spaziatrice. Chiaramente per ottenere più punti dovrete muovere alternativamente i tasti cursore sinistra e destra in modo da aumentare l'energia. State attenti a non colpire l'anatra che passeggia. Per passare al terzo livello bisogna realizzare 3000 punti.

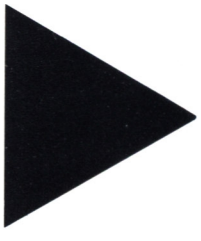
Terzo livello: un aquilone si solleva dal prato, aggiustando la mira dovrete colpirlo per 10 volte. Se ciò accade avrete un bonus di 1000 punti.

L'intero programma è stato realizzato con semplici istruzioni Basic senza usare funzioni o comandi di gestione avanzata della memoria (VPEEK, VPOKE, PEEK, POKE, USR, ecc.).

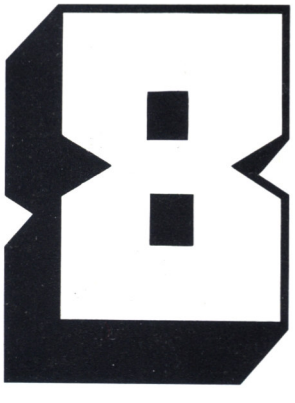


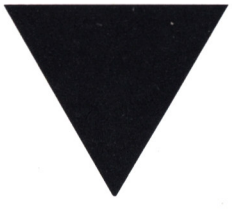
CODICE MORSE C.W.

("C.W.")



di J. Cordova

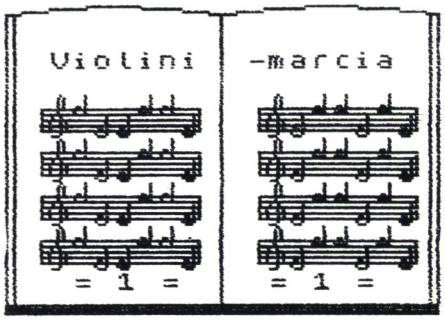




MUSICA

("MUSICA")

adatt. da G. Riccobono



Dopo aver dato il RUN appaiono sullo schermo le istruzioni in modo scorrevole. Per fermare lo scroll utilizzare il tasto "STOP". Finita la visualizzazione delle istruzioni premere il tasto funzione F1. A questo punto appare il menu con tre opzioni: 1) suono MSX, 2) piano, 3) profilo suono.

1) Suono MSX: la tastiera del vostro computer viene trasformata in una tastiera musicale a 3 ottave. Usare i tasti "." o "/" per cambiare ottava. Usare i tasti "d", "f", "g", "h", "j", "k", "l", ";", "r", "t", "u", "i", "o" per suonare. Premendo il tasto "ESC" si torna al menu.

2) Piano: è tutto uguale alla prima opzione con la sola differenza che il suono ottenuto è molto simile a quello prodotto dal pianoforte.

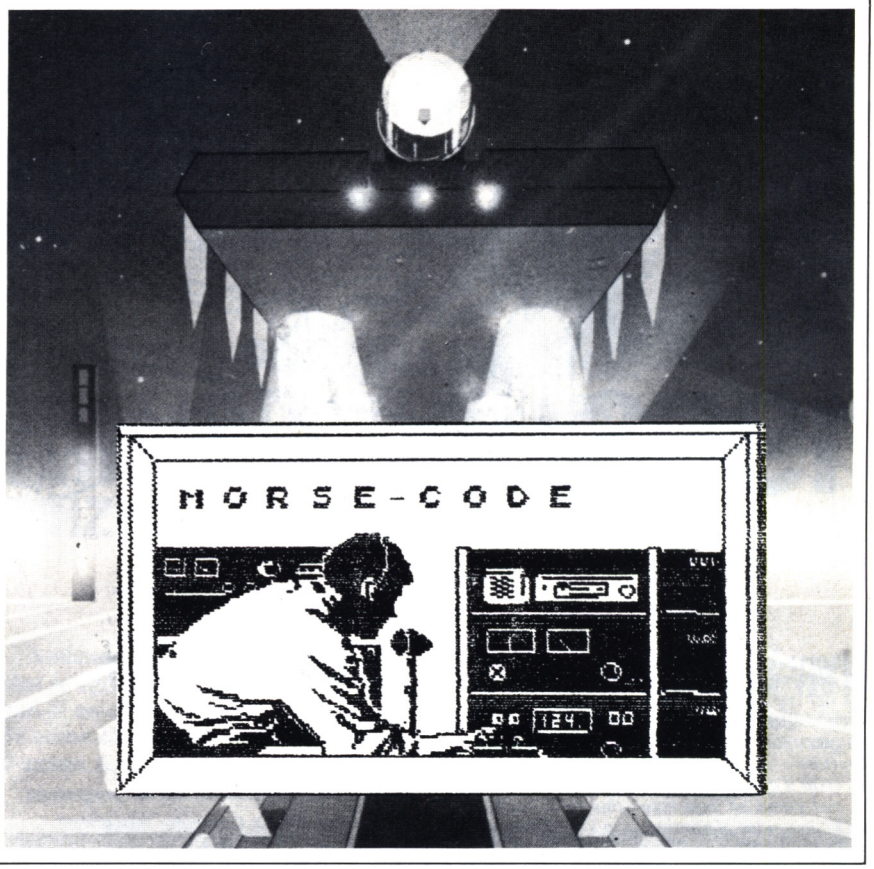
3) Profilo sonoro: è possibile cambiare il profilo sonoro (comando "S" dell'istruzione PLAY) e la modulazione (comando "M" dell'istruzione PLAY). Usare i tasti cursore alto/basso per aumentare/diminuire la modulazione per un fattore di M*1.

Usare i tasti cursore sinistra/destra per aumentare/diminuire la modulazione per un fattore di M*50. Per selezionare il profilo sonoro "S" basta premere il tasto con il numero relativo al profilo voluto. Infine premendo la barra spaziatrice potrete ascoltare la scala musicale adattata ai valori di "S" ed "M" da voi selezionati.

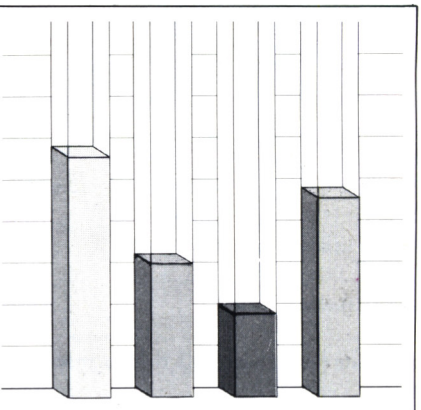
Dopo aver visualizzato il disegno di presentazione, una nave che comunica in telegrafia con una stazione costiera, il programma mostra cinque opzioni: tre per lo studio del codice morse, una per la fine del programma ed un'altra per rivedere il disegno di presentazione.

La prima opzione permette di vedere le lettere, numeri e segni in maniera singola, cioè visualizzando la sintassi di quel carattere ed il suo suono corretto in telegrafia.

Con la seconda opzione l'utente può scrivere un testo ed ascoltare contemporaneamente la sua trasmissione in codice morse. Infine con la terza opzione il computer emetterà il suono di un carattere e l'utente dovrà indovinare che cosa è stato trasmesso. Premendo il tasto "#" il computer emetterà nuovamente il suono del carattere e se l'utente non riuscirà ancora a capire di che cosa si tratti allora potrà premere il tasto ENTER per avere un aiuto dal computer.



10



ISTOGRAMMI

("ISTOGR")

di D. Montresor

Il modo migliore per mettere a confronto delle cifre è quello di rappresentarle sottoforma di istogrammi. Con questa utility è possibile memorizzare ben 24 istogrammi. Dopo aver dato il RUN appare sullo schermo una pagina di presentazione, dopo aver premuto un qualsiasi tasto apparirà sul video il menu composto da 9 opzioni. Diamo qui di seguito una descrizione relativa ad ogni opzione.

Opzione n. 1: consente di inserire i dati da visualizzare associandoli ad uno dei 24 istogrammi. Ogni istogramma è formato da 13 dati numerici e, se l'utente lo desidera, ogni dato può avere un commento lungo 8 caratteri.

Opzione n. 2: serve a scambiare i dati di un istogramma con un altro.

Opzione n. 3: inserendo un numero compreso tra 1 e 24 si cancellano i dati del relativo istogramma.

Opzione n. 4: esegue la funzione MOTOR ON per posizionare il nastro.

Opzione n. 5: esegue la funzione MOTOR OFF per spegnere il registratore.

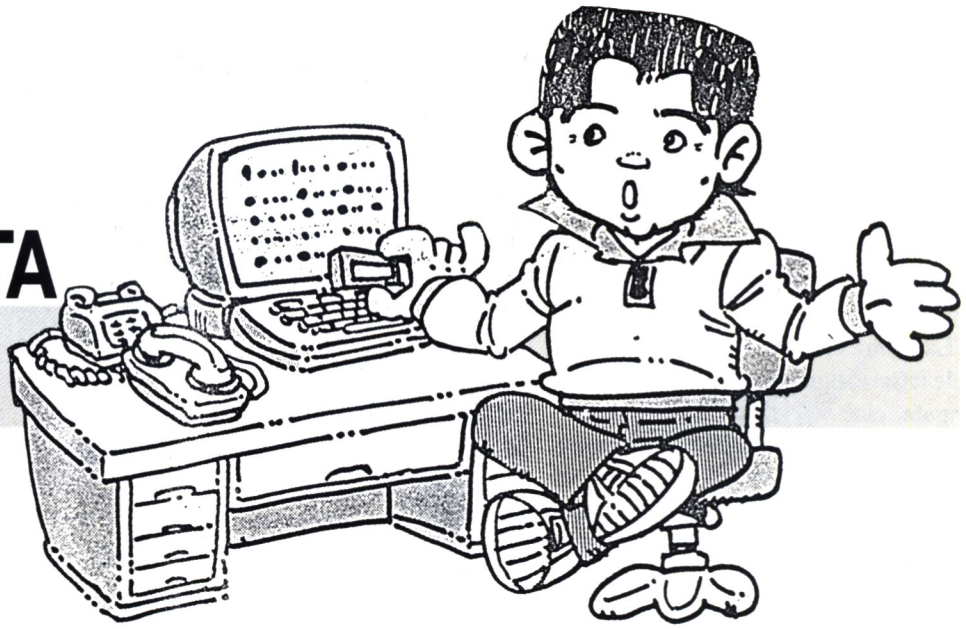
Opzione n. 6: è l'opzione più importante perché visualizza sottoforma di istogramma i dati relativi ad una delle 24 tabelle. Per cambiare istogramma da una tabella ad un'altra bisogna spostare, usando i tasti cursore, il quadretto, in alto allo schermo, nella casella relativa alla tabella da visualizzare e premere il tasto di spazio. Per tornare al menu premere il tasto SELECT.

Opzione n. 7: serve a caricare da nastro il file dati.

Opzione n. 8: salva il file dati su nastro.

Opzione n. 9: visualizza tutti i titoli associati alle 24 tabelle memorizzate.

GRANDE INCHIESTA MSX 1986



Abbiamo bisogno anche del tuo consiglio e del tuo parere per migliorare MSX Computer Magazine e seguitare a farne una rivista che corrisponda sempre e sempre di più alle tue aspettative ed ai tuoi desideri. Compila con sincerità il questionario e spediscilo (corredato del tuo nome ed indirizzo solo *se lo vuoi) a MSX Computer Magazine, c.so Vitt. Emanuele 15, 20122 Milano.*
Grazie!

Ciao, quanti anni hai?

.....

Dovi vivi?

☐ nord ☐ centro ☐ sud
☐ città ☐ provincia

Marca e tipo del tuo computer MSX:

.....

Sei uno studente o lavori?

.....

Da quanto tempo leggi MSX Computer Magazine?

.....

Lo compri sempre?

☐ sì ☐ no

Se no, perché?

.....

Quali altre riviste del settore leggi?

.....

.....

Secondo te, MSX Computer Magazine tratta in modo soddisfacente tutti i temi relativi agli MSX?

☐ sì ☐ no

Quali argomenti ti interessano di più?

.....

.....

.....

Quale giudizio complessivo dai dei programmi sulla cassetta di MSX Computer Magazine?

☐ discreto ☐ buono
☐ ottimo ☐ insufficiente

Qual è il genere di programma che preferiresti trovare sulla cassetta?

☐ giochi ☐ utility
☐ educational ☐ avventure

Hai mai provato ad "inventare" un programma tutto tuo?

☐ sì ☐ no

Indica tre articoli che vorresti veder presto pubblicati sulla rivista

.....

.....

.....

Indica tre programmi che vorresti veder pubblicati presto sulla nostra cassetta

.....

.....

.....

Quali sono gli articoli pubblicati che ti sono piaciuti di più?

.....

.....

.....

.....

.....

Quali sono i programmi già pubblicati che ti sono piaciuti di più?

.....

.....

.....

Cosa manca secondo te su MSX Computer Magazine?

.....

.....

.....

Ti piace partecipare ai concorsi con premi?

☐ sì ☐ no

Hai altri hobby oltre al computer? Quali?

.....

.....

.....

Qual è la tua critica più feroce ad MSX Computer Magazine?

.....

.....

.....

E il miglior complimento?

.....

.....

.....

Sei abbonato?

☐ sì ☐ no

Se no, come mai?

.....

.....

Se tu dovessi dare un voto di merito da 0 a 10 ad MSX Computer Magazine, che voto daresti?

.....

SCRIVI QUI IL TUO NOME
(solo se vuoi)

NOME..... COGNOME.....

VIA..... CITTÀ.....

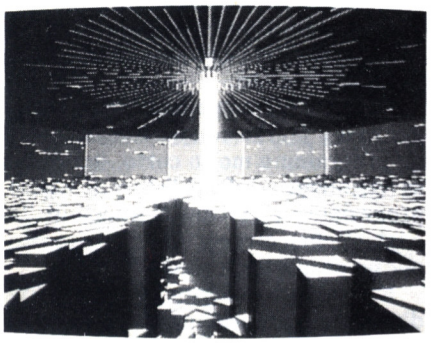
SERENO/VARIABILE

PER CHI IMPARA A PROGRAMMARE: COME OTTENERE LA LISTA DELLE VARIABILI UTILIZZATE CON SPECIFICATE LE FUNZIONI ESPLETATE

di G. RICCOBONO

Vi è mai capitato di dimenticare, durante la creazione di un programma, il nome delle variabili e la loro funzione?

Ecco a voi un programma che vi aiuta ad ottenere la lista delle variabili utilizzate e a risalire alla loro funzione, dando così un aiuto validissimo nella ricerca degli errori ed un articolo ricco di suggerimenti utili al programmatore esperto e di consigli indispensabili a chi si affaccia per la prima volta al mondo degli home-computers.



Sarà sicuramente capitato a molti, scrivendo un programma piuttosto lungo, di dimenticare la funzione delle variabili e, soprattutto, il loro nome.

Questo fatto, che non va assolutamente sottovalutato, può procurare dei seri problemi, vediamo quali.

Supponiamo di avere chiamato A una variabile numerica contenente una determinata informazione (un indirizzo, un valore etc.) e di esserci dimenticati di avere usato il nome A per assegnare tale variabile.

Andando avanti nello scrivere il nostro programma possiamo avere bisogno di un'altra variabile e, non sapendo come chiamarla, le diamo di nuovo il nome A, senza accorgerci di avere già utilizzato tale nome. Alteriamo così, durante l'esecuzione, il valore della nostra variabile perdendo, irrimediabilmente, l'informazione originaria in essa contenuta.

```
1000 REM ---- Programma prova ----
1010 SCREEN 0:COLOR 15,4,4:KEYOFF:CLS
1020 '                .....


1090 '                .....
2000 LET B=&HEA
2010 LET X=&HB000
2020 POKE X,B
2030 '                .....


2090 '                .....
2100 IF PEEK(X)=0 THEN GOSUB 20000
2110 '                .....


2990 '                .....
3000 POKE X,B
3010 '                .....


8990 '                .....
9000 LET X=30:LET Y=10
9010 LOCATE X,Y:PRINT A$
9020 '                .....


10000 GOTO 2100
```

Programma didattico di prova. I puntini sospensivi servono ad indicare la presenza eventuale di altre righe di elaborazione.

Questo può portare a un danno trascurabile nel caso che tale informazione originaria non serva più nella successiva parte del programma.

Ben diversa è però la situazione se invece abbiamo di nuovo bisogno della nostra informazione originale, ad esempio perché nel nostro programma c'è un'istruzione GOTO che rimanda alle prime righe per un'ulteriore elaborazione. Il nostro programma ritroverebbe infatti la variabile A da utilizzare secondo determinati schemi, senza sapere che il valore che è supposto essere presente in A è stato in verità cambiato con uno che non ha niente a che vedere con quello originale.

QUASI UN ESEMPIO

Rivediamo tutto questo con l'ausilio di un ipotetico programma.

Supponiamo di conservare nella variabile X un indirizzo di memoria e di riutilizzare poi questa variabile per altri scopi, in un programma del tipo in figura.

È chiaro che non abbiamo scritto un vero programma, ma solo alcune righe salienti, che servono a rendere chiari i concetti.

I puntini sospensivi, nel finto listato, servono a indicare la presenza eventuale di altre righe di elaborazione, che peraltro non interessano ai nostri scopi.

In un programma come quello precedente succederanno sicuramente cose assai strane.

È opportuno dire a questo punto che, anche se le variabili sono state usate in maniera propria (cioè se non è stato dato lo stesso nome a due variabili differenti), un altro tipo di errore può generarsi nel caso che siano stati assegnati valori errati a una delle nostre variabili.

Supponiamo per esempio di non aver scritto le righe 9000 e 9010 nel finto programma precedente e, quindi, di non aver fatto l'errore di confondere variabili; potremmo aver fatto però un errore di battitura alla riga 1000, avendo scritto qualcosa del tipo:

```
1000 LET X=&HB00
```

È da rimarcare che questo tipo di errore di battitura non provoca l'arresto del programma, con la generazione di un messaggio d'errore, cosa che, invece, sarebbe avvenuta nel caso che l'errore di battitura fosse stato:

```
1000 LT X=&HB000
```

oppure:

```
1000 LEET X=&HB000
```

GLI ERRORI INSIDIOSI

È quindi chiaro che errori del tipo in esame sono molto più insidiosi di questi ultimi, essendo impossibile la loro rivelazione durante l'esecuzione, tramite la generazione di messaggi d'errore.

E veniamo ora al commento del programma "RICERCA VARIABILI".

Il programma fornisce la lista di tutte le righe di programma contenenti assegnazioni di variabili, cioè righe contenenti istruzioni del tipo:

```
LET A$="CD"
A$="CD"
B=123
etc.
```

In questo modo potete venire in possesso:

1. di tutte le variabili, di qualunque tipo esse siano (numeriche, singola o doppia precisione, stringhe), cioè vi può aiutare, tra le altre cose, a trovare i nomi delle variabili già usate, aiutandovi così a decidere il nome delle nuove variabili, senza rischio di ripetere nomi già impiegati;
2. della elaborazione subita dalle variabili, cioè della successione di valori ad esse assegnate, accorgendovi, così, se avete usato la stessa variabile per due scopi differenti o se avete ad essa assegnato valori errati, ciò può esservi utile, a programma terminato, nella ricerca degli errori di battitura;
3. del numero di riga alle quali tali variabili vengono elaborate, dandovi la possibilità di un'immediata correzione degli errori.

Ovviamente un programma di questo tipo è inutile se dovete esaminare un programma molto corto, in tal caso vi conviene infatti listare il vostro programma e cercare le variabili "a occhio", ben diverso è però il discorso per un programma medio/lungo.

E veniamo adesso alle modalità d'uso del programma "RICERCA VARIABILI".

IL NOSTRO PROGRAMMA

Per prima cosa dovete salvare il programma da esaminare, cioè quello di cui desiderate le variabili, su cassetta, usando la istruzione:

```
SAVE "CAS:nome programma"
```

Controllate di aver ben digitato tale istruzione e poi date il "Return".

ECCO IL PROGRAMMA

```
100 '**********************
110 '*                      *
120 '* RICERCA VARIABILI    *
130 '*                      *
140 '**********************
150 ' ------ Inizializzazione ------
160 SCREEN0:COLOR 15,4,4:CLS:KEYOFF
170 ON ERROR GOTO 430
180 OPEN "CAS:" FOR INPUT AS#1
190 A$="":B$="":LINE INPUT#1,A$
200 LN=LEN(A$):IF LN<=3 THEN GOTO190
210 FOR I=1 TO LN-1
220 PA$=MID$(A$,I,1)
230 IF PA$="=" THEN GOTO 270
240 NEXT I
250 GOTO 190
260 ' ------ Ricerca 'PRINT' -------
270 IF LN<=8 THEN GOTO 400
280 FOR I=1 TO LN-7
290 T=I
300 PA$=MID$(A$,I,5)
310 IF PA$="PRINT" THEN GOTO 600
320 NEXT I
330 ' -------- Ricerca 'IF' --------
340 IF LN<=10 THEN GOTO 400
350 FOR I=1 TO LN-10
360 V=I
370 PA$=MID$(A$,I,2)
380 IF PA$="IF" THEN GOTO 740
390 NEXT I
400 '--Stampa linea dell'assegnaz.--
410 IF LEN(A$)<LEN(B$) THEN PRINT B$ ELSE PRINT A
420 GOTO 190
430 REM ----- Fine programma -----
440 IF ERR=55 THEN CLOSE#1:PRINT"Ho finito.":END
450 PRINT"Si è verificato l'errore numero";ERR;"al
la linea";ERL:CLOSE#1:END
600 ' ---- Ho trovato un print ----
610 ' ------- Ricerca ':' ---------
620 FOR J=T+5 TO LN-1
630 PB$=MID$(A$,J,1)
640 U=J
650 IF PB$=":" THEN GOTO 710
660 NEXT J
670 ' ---- Non ho trovato ':' -----
680 A$=LEFT$(A$,T-1)
690 GOTO 200
700 AUTO 710
710 '- Ho trovato ':' dopo 'PRINT' -
720 A$=LEFT$(A$,T-1)+RIGHT$(A$,LN-U)
730 GOTO 200
740 ' ----- Ho trovato un 'IF' ----
750 B$=A$
760 FOR J=V+3 TO LN-4
770 PB$=MID$(A$,J,4)
780 IF PB$="THEN" OR PB$="GOTO" THEN A$=LEFT$(A$,V
-1)+RIGHT$(A$,LN-J-4):GOTO 200
790 NEXT J
800 PRINT"NON CAPISCO LA SEGUENTE LINEA DI 'IF' :"
810 PRINT B$
820 GOTO 190
```

Una volta che il vostro programma è tutto salvato su cassetta, riavvolgete il nastro, date un RESET al calcolatore e caricate poi il programma RICERCA VARIABILI in memoria.

Quando quest'ultimo programma è tutto in memoria schiacciate il tasto "PLAY" del registratore e poi date il RUN. Al fine di un corretto funzionamento del programma è indispensabile che lo spinotto piccolo, cioè quello della presa REMOTE, sia inserito nel registratore.

Analizziamo ora il listato del programma "RICERCA VARIABILI".

— La riga 160 inizializza il sistema;

— la riga 170, allegata alle 430-480, fa in modo che, quando il programma da esaminare è stato tutto analizzato, il programma "RICERCA VARIABILI" non si interrompa con un'antipatica generazione dell'errore INPUT PAST END, cosa che altrimenti accadrebbe quando, terminato il programma su cassetta, viene data un'istruzione LINE INPUT#1;

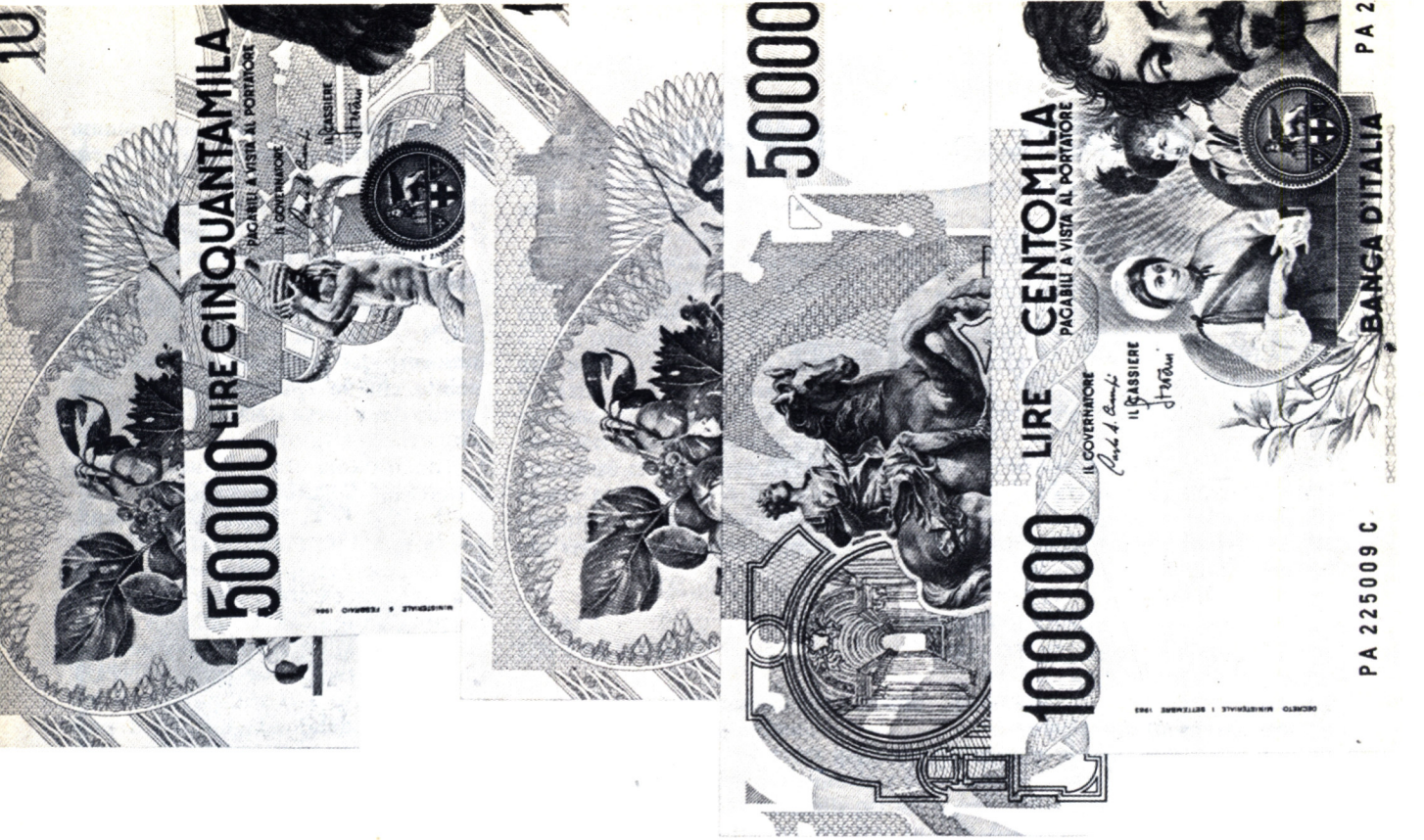
— le righe 190-250 leggono dalla cassetta, riga per riga, il programma da esaminare e cercano eventuali assegnazioni tramite la ricerca del carattere "=";

— le righe 260-320, unitamente alle righe 600-730, eliminano, nel caso sia stato trovato un carattere "=", le istruzioni "PRINT", in modo da evitare il listaggio indesiderato di righe del tipo PRINT "A=B", perché queste, pur contenendo un carattere "=", non sono ovviamente delle assegnazioni;

— le righe 330-420, unitamente alle 740-820, svolgono un compito analogo, evitando che venga scambiato per un'assegnazione il carattere "=" relativo all'istruzione: IF A=B GOTO...; attenzione però che il programma esegue il listaggio completo delle linee del tipo: IF A=B THEN C=..., contenendo queste ultime un'assegnazione della variabile C. Chi volesse evitare la stampa dell'istruzione IF, cioè nell'esempio appena citato evitare che compaia sullo schermo: IF A=B THEN, ottenendo solo la scrittura di: C=..., può eliminare la linea 750, cambiare la linea 410 in: 410 PRINT A$ e, se vuole essere proprio un sofista, inserire una routine di ricerca di ELSE e di ":", analoghe alle due già presenti, in modo da eliminare anche quanto sta dopo l'assegnazione contenuta nell'istruzione IF;

— le linee 400-420 provvedono infine alla stampa della linea dove è presente l'assegnazione di variabile.

■



MSX BANK

Hai programmi originali, esclusivamente pensati e fatti da te? Mandaceli in visione e, se verranno pubblicati, saranno certamente compensati a partire da lire 50 mila in su (a seconda del tipo di programma). Se sei fantasioso, bravo e veloce oltre che ordinato, puoi arrivare a guadagnare un bel gruzzoletto! Naturalmente il tuo nome apparirà stampato sulla rivista come valido collaboratore. Fruga nella tua fantasia e mandaci non solo arcade ma anche utility inedite, giochi di società particolari, interessanti routine in linguaggio macchina ed avventure con grafica (invia sempre anche la soluzione comando per comando in sequenza) allegando anche caratteristiche tecniche dei programmi, listati, stampate di screen etc. Se decidi di inviare un programma (indirizza a MSX Computer Magazine, c.so Vitt. Emanuele 15, 20122 Milano) segui queste regole:

1) salva il programma su entrambi i lati di una cassetta;
2) usa 1 cassetta per ogni programma;
3) scrivi il tuo nome, quello del programma e quello della macchina sulla cassetta;
4) accludi un foglio dattiloscritto dove spiegherai a cosa serve il programma, come si usa, quali tasti usare, lo scopo. Non proteggete i programmi! Dobbiamo guardarli dentro. Non mettete il vostro indirizzo all'interno del programma (basta «by... nome e cognome»).

Rispondiamo sempre a tutti e, soprattutto, manteniamo la parola data. Coraggio dunque, fatevi vivi!

ATTENZIONE: I programmi inviati debbono essere assolutamente inediti, a noi ceduti in esclusiva. L'editore si riserva di rifiutare programmi anche già accettati e di non compensare programmi già pubblicati qualora si verificasse che gli stessi programmi (anche quando con titoli diversi) siano stati ceduti o pubblicati da altri.

IL LINGUAGGIO MACCHINA

COME PROGRAMMARE IN LINGUAGGIO MACCHINA. I CODICI ISTRUZIONE DEL MICROPROCESSORE Z80 A

(3ª PUNTATA)

di EMANUELE DASSI

Abbiamo introdotto il concetto di flag e spiegato, in relazione alle operazioni aritmetiche, il loro significato (vedi numero 7/86). In questa puntata tratteremo le operazioni logiche per poi passare a quelle di manipolazione dei bit (shift, rotazione, test ecc.). Tutte queste istruzioni modificano lo stato di alcuni flag e quindi è in base anche a questi ultimi che si ottengono determinati risultati.

Le istruzioni logiche sono quattro: AND, OR, XOR e CP. Operano tutte con valori a 8 bit ed il loro primo operando è sempre A; anche il risultato è posto sempre nell'accumulatore. Ma vediamo, una per una, come lavorano.

L'istruzione AND viene usata nella seguente forma: AND s, dove s è un registro ad 8 bit (A, B, C, D, E, H, L), un valore numerico compreso tra 0 e 255 oppure una locazione di memoria puntata da (HL), (IY+d) o (IX+d). L'istruzione AND opera su ogni singolo bit dei due operandi (A ed s). Supponiamo di avere nell'accumulatore la rappresentazione binaria 11001011 e nel registro B il valore 10000110. Eseguendo l'istruzione AND B, il risultato, posto in A, sarà 10000010. L'AND confronta il primo bit del primo operando con il primo bit del secondo, il secondo bit del primo operando con il secondo bit del secondo operando e così via, fino al bit più significativo. Ogni confronto genera un bit che è 1 se i due bit confrontati sono entrambi a 1 viceversa 0.

```
4C 51 A8     JMP   $A85
4C 5C B1     JMP   $B15
59 FA BF     EOR   $BFF
9D 38 4C     STA   $4C3
B0 17        BCS   $040
4C 65 FF     JMP   $FF6
4C 65 FF     JMP   $FF6
65 FF        ADC   $FF
DD D0 D2     CMP   $D2D
CC CC AD     CPY   $ADC
A0 A0        LDY   #$A0
A0 A0        LDY   #$A0
A0 A0        LDY   #$A0
C6 A0        DEC   $
B9 C4 A0     LDA   $A0C
A0 A0        LDY   #$A0
A0 CA        LDY   #$CA
```

La tabella di verità della funzione AND è la seguente:

bit primo oper.	*bit secondo oper.*	*AND*
1	*1*	*1*
1	*0*	*0*
0	*1*	*0*
0	*0*	*0*

Nella tabella di verità sono elencate le diverse possibilità di confronto e i relativi risultati della funzione logica AND.

L'istruzione OR viene usata con gli stessi operandi di AND, cioè si utilizza come ORs. Anch'essa esegue un confronto sui singoli bit dei suoi operandi. La tabella di verità della funzione OR è:

bit primo oper.	*bit secondo oper.*	*OR*
1	*1*	*1*
1	*0*	*1*
0	*1*	*1*
0	*0*	*0*

Il risultato di OR è 0 solo e soltanto quando entrambi i bit del confronto sono 0. Per esempio se in A abbiamo il valore 11110000 ed eseguiamo l'istruzione OR 34, il risultato sarà il seguente:

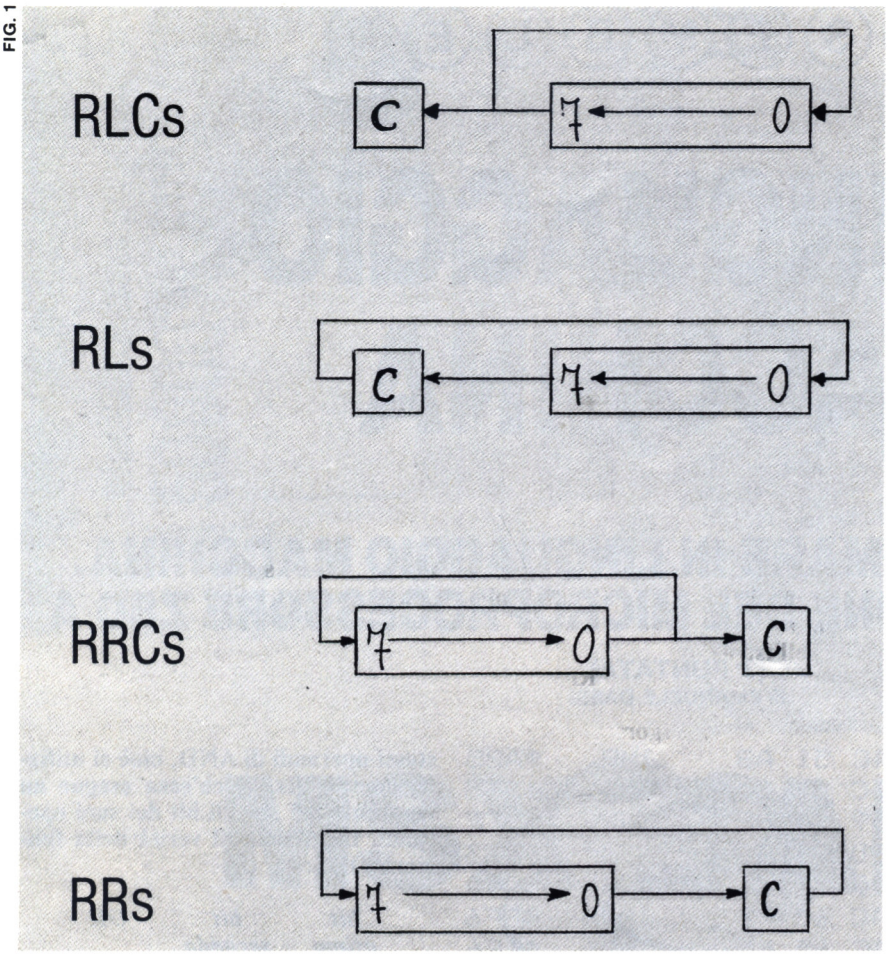
11110000 = A
00100010 = 34

11110010 = dopo l'istruzione OR 34

La terza istruzione logica è XORs. Anch'essa opera su singoli bit secondo la seguente tabella di verità:

bit primo oper.	*bit secondo oper.*	*XOR*
1	*1*	*0*
1	*0*	*1*
0	*1*	*1*
0	*0*	*0*

FIG. 1

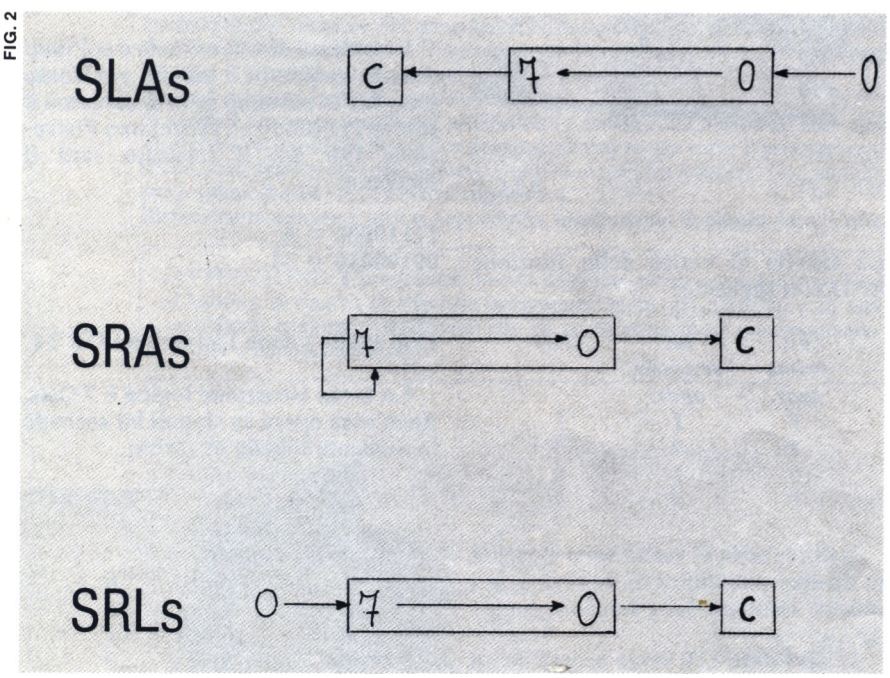


Il risultato di XOR è 1 quando i due bit del confronto sono diversi.

Quindi se in A c'è 11001101 e in E vi è 10111001 il risultato di XOR E sarà 01110100.

Le tre istruzioni logiche AND, OR e XOR alterano il flag di segno S e il flag di zero Z a seconda del loro risultato. Cioè S avrà il valore del bit più significativo e il flag Z sarà 1 se il risultato dell'operazione è 0. Il flag di carry C, invece, è posto sempre a 0. Infine il flag P/V indica la parità; sarà quindi 1 se il numero di bit a 1 sono pari altrimenti 0.

FIG. 2



L'ISTRUZIONE CP

L'ultima operazione logica è CPs, dove s può essere uno degli operandi descritti sopra. L'istruzione CP si differenzia dal modo di operare delle prime tre in quanto non esegue un confronto su ogni singolo bit, ma una sottrazione tra il valore dell'accumulatore e quello di s. Il risultato viene perso, quindi il valore di A non viene alterato, però lo stato dei flag determina se A è maggiore, uguale o minore di s. In particolare il flag C è posto a 1 se s è maggiore di A e a 0 se s è minore o uguale ad A. Facciamo subito un esempio: in A c'è il valore 30 e in H il valore 24. Dopo CP H in A vi è ancora 30 ed in H il valore 24, mentre il flag C è 0, così come il flag Z. Se invece in A vi fosse stato 24 e in H 30, CP H avrebbe posto il flag C ad 1 mentre il flag Z sarebbe rimasto a 0. In questa istruzione il flag P/V indica l'overflow per valori in complemento a due.

Insieme alle quattro istruzioni logiche ve ne sono da aggiungere altre due operanti esclusivamente sull'accumulatore; esse sono: CPL e NEG. La prima modifica i bit a 1 in 0 e viceversa. Avendo quindi in A il valore 00111011 dopo l'istruzione CPL in A avremo 11000100.

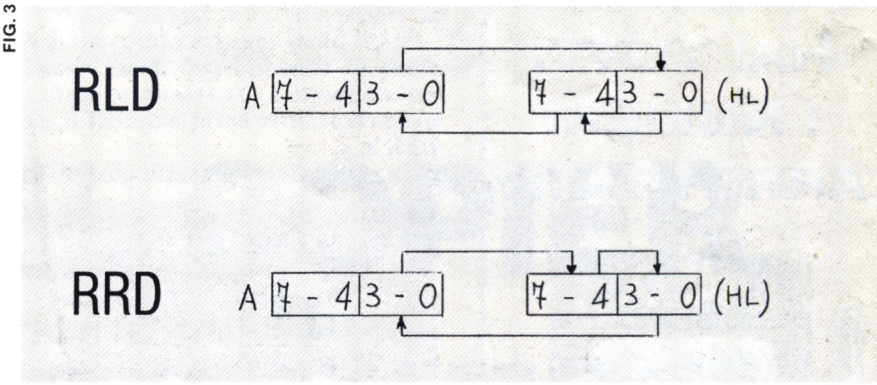
LE ISTRUZIONI DI ROTAZIONE

L'istruzione NEG invece esegue il complemento a due del valore posto in A.

CPL non modifica alcun flag mentre l'istruzione NEG agisce su S, Z e C secondo le regole già descritte più volte.

Anche le istruzioni di rotazione e shift operano su dati ad 8 bit ma prima di passare a descriverle dobbiamo precisare che i bit sono numerati da 0 a 7 da sinistra verso destra; in particolare avendo per esempio il valore

FIG. 3



10000000 il bit a 1 è quello di posizione 7 e il bit 0 sull'estrema destra della cifra è il bit di posizione 0. Le istruzioni di rotazione sono quattro: RLC, RL, RRC e RR. Il loro operando è s, cioè uno dei registri ad 8 bit, (HL), (IY+d) o (IX+d). Nella figura 1 è possibile vedere uno schema grafico del funzionamento delle quattro istruzioni. L'istruzione RLC ruota verso sinistra il contenuto di s e pone il bit 7 nel flag C e nella posizione 0 di s. Per esempio, avendo nel registro D il valore 01110011 dopo l'istruzione RLC D il registro D conterrà 11100110 e il flag C il valore 0. L'istruzione RL opera allo stesso modo di RLC, con la sola differenza che nel bit 0 avremo il valore del flag di carry. Supponiamo di avere in A il valore 01110010 e nel flag C 1. Dopo aver eseguito RL A avremo in C il valore 0 e in A il numero 11100101. Le istruzioni RRC e RR operano in modo complementare a RLC e RL; cioè la rotazione di s avviene verso destra e il bit entrante nel carry è quello di posizione 0.

UN ESEMPIO SUGLI SHIFT

Le istruzioni di shift sono tre: SLA, SRA e SRL. La differenza tra SLA e SRL risiede solo nella direzione di shift che per la prima è verso sinistra e per la seconda verso destra, come appare nella figura 2. L'operando delle istruzioni di shift è s come per le operazioni di rotazione. Vediamo ora un esempio per ogni istruzione di shift. Supponiamo, per tutti e tre gli esempi, di avere nel registro L il valore 10010000 e nel flag di carry 0. Segue ora una tabella dei risultati dopo le tre operazioni:

istruzione	*reg. L*	*flag C*
SLA L	*00100000*	*1*
SRA L	*11001000*	*0*
SRL L	*01001000*	*0*

GLI ALTRI FLAG

Abbiamo visto che il flag più importante coinvolto nelle operazioni di rotazione e shift è quello del carry ma non è l'unico ad essere modificato. Anche il flag S, Z e P/V vengono modificati, in particolare P/V indica la parità.

Vi sono infine altre due istruzioni di rotazioni assai particolari: RLD e RRD. Osservando la figura 3 esse agiscono esclusivamente sull'accumulatore e sulla locazione di memoria puntata da HL scambiando contemporaneamente il nibble (quattro bit) inferiore di A con quello superiore di (HL) nell'istruzione RLD, o con quello inferiore di (HL) nella RRD. Queste istruzioni sono dedicate alle operazioni in BCD, argomento che non tratteremo in questa sede.

Siamo giunti a presentare le operazioni di test, set e reset dei singoli bit. Sono istruzioni molto potenti, non riscontrabili su altri microprocessori a 8 e perfino a 16 bit. L'istruzione di test è BIT b, s dove b è un numero compreso tra 0 e 7, identificante il bit da testare ed s è, come al solito, un registro ad 8 bit, (HL), (IY+d) oppure (IX+d). Il risultato di BIT è quello di variare il contenuto del flag di zero. Precisamente Z è 1 se il bit testato è

```
                ORG 62000
3A4DF2          LD A,(LOC)
; DIVISIONE INT (A/2)
CB3F            SRL A
324DF2          LD (LOC),A
2A4EF2          LD HL,(LOC1)
; HL*2
CB25            SLA L
CB14            RL H
E5              PUSH HL
; HL*4
CB25            SLA H
CB14            RL H
; HL=HL+HL
29              ADD HL,HL
; DE=HL*8
54              LD D,H
5D              LD E,L
E1              POP HL
; HL=HL*10
19              ADD HL,DE
224EF2          LD (LOC1),HL
C9              RET
00              LOC: DEFB 0
0000            LOC1: DEFW 00
                END
```

Listato 1. Un piccolo programma in linguaggio macchina: in pratica un aiuto per capire i concetti esposti in questa puntata.

0, viceversa Z è 0 se il bit testato è 1.

Il bit di un registro o di una locazione di memoria può anche essere settato (posto a 1) o resettato (posto a 0) con le relative istruzioni: SET b, s e RES b, s.

In conclusione di questa puntata vi presentiamo un programmino Assembler (listato 1), e la codifica in codici macchina in MSX-Basic (listato 2), che utilizza istruzioni di shift e rotate per eseguire una divisione per 2 del valore dell'accumulatore e una moltiplicazione per 10 del valore di HL.

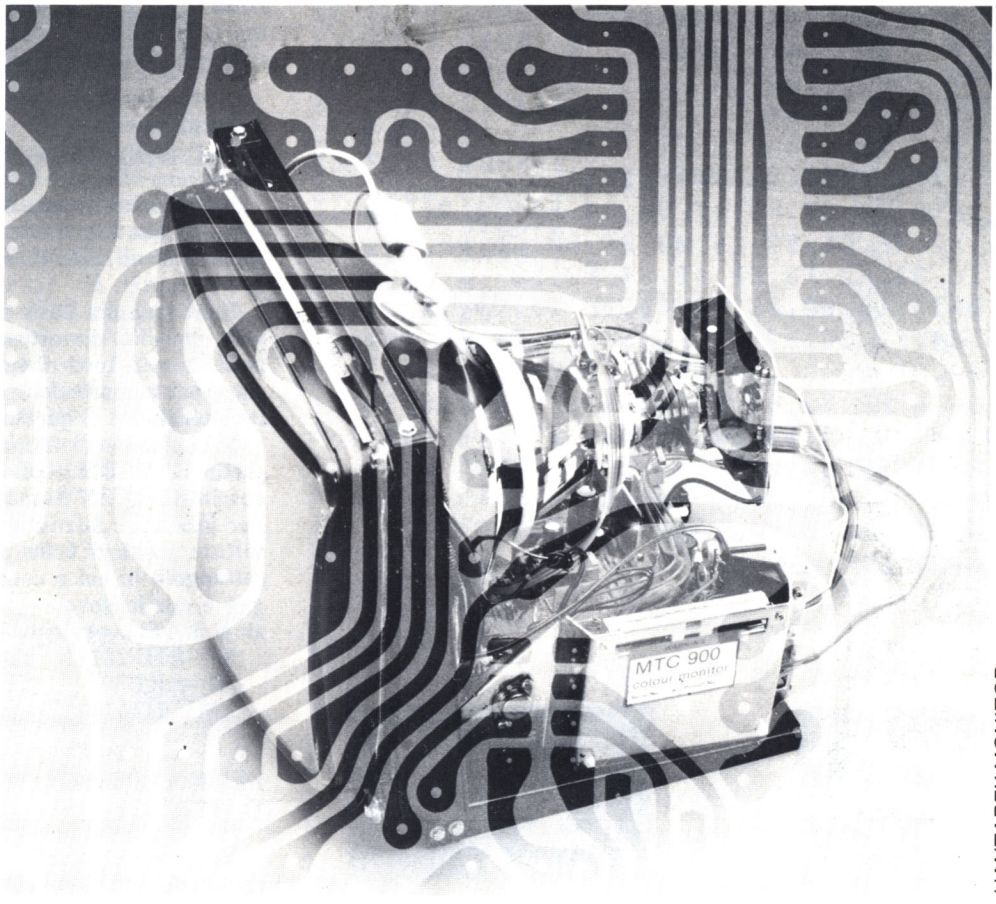
```
10 CLEAR 200,&HF22F
20 DATA 3A,4D,F2: '    LD A,(F24D)
30 DATA CB,3F: '       SRL A
40 DATA 32,4D,F2: '    LD (F24D),A
50 DATA 2A,4E,F2: '    LD HL,(F24E)
60 DATA CB,25: '       SLA L
70 DATA CB,14: '       RL H
80 DATA E5: '          PUSH HL
90 DATA CB,25: '       SLA L
100 DATA CB,14: '      RL H
110 DATA 29: '         ADD HL,HL
120 DATA 54: '         LD D,H
130 DATA 5D: '         LD E,L
140 DATA E1: '         POP HL
150 DATA 19: '         ADD HL,DE
160 DATA 22,4E,F2: '   LD (F24E),HL
170 DATA C9: '         RET
180 DATA 00: '         (F24D)
190 DATA 00,00: '      (F24E-F24F)
200 'CARICAMENTO DATI L/M IN MEMORIA
210 FOR N=&HF230 TO &HF24F
220 READ A$:POKE N,VAL("&H"+A$)
230 NEXT
240 DEFUSR=&HF230 'START L/M
250 CLS:INPUT "A <0-255>";A%
260 IF A%>255 OR A%<0 THEN 250
270 POKE &HF24D,A%
280 INPUT "HL <0-6535>";HL%
290 IF HL%>6535 OR HL%<0 THEN 280
300 POKE &HF24E,HL%-INT(HL%/256)*256
310 POKE &HF24F,INT(HL%/256)
320 A=USR(0) 'ESECUZIONE L/M
330 PRINT "IL RISULTATO DI SRL A"
340 PRINT " (CIOE' INT(A/2)) E':";
350 PRINT PEEK(&HF24D)
360 PRINT "IL RISULTATO DI SLA L,";
370 PRINT "RL H ECC.";
380 PRINT "(CIOE' HL*10)";
390 R=PEEK(&HF24E)+256*PEEK(&HF24F)
400 PRINT " E':";R
410 PRINT "VUOI ESEGUIRE NUOVAMENTE"
420 PRINT "IL PROGRAMMA <S/N>";
430 INPUT R$
440 IF R$="S" OR R$="s" THEN 250
450 END
```

Listato 2. È la codifica in codici macchina. Vengono utilizzate istruzioni di shift e rotate per dividere per 2 il valore dell'accumulatore (il valore di HL è moltiplicato per 10).

UTILITY

PIXEL COPIER

LEGGI IL COLORE DI TUTTI I PUNTI DELLO SCHERMO CON UNA ROUTINE IN LINGUAGGIO MACCHINA E TRASFERISCI L'IMMAGINE IN UN'ALTRA ZONA DEL VIDEO!



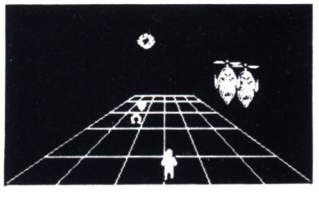
HANTAREX MONITOR

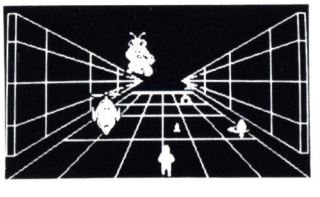
A voi tutti sarà certamente capitato di dover disegnare sullo schermo più volte la stessa figura e quindi di ripetere l'algoritmo di tracciatura spostando semplicemente le coordinate dell'origine del disegno. In questo modo, oltre a rendere più lenta l'esecuzione del programma, si è anche allungato il suo testo. L'utility che stiamo per presentarvi è una breve ma potente routine in linguaggio macchina che consente di copiare una qualsiasi immagine già presente sullo schermo in un'altra zona del video. Il programmino è assai veloce anche se poteva esserlo ancor di più ma abbiamo voluto adottare un algoritmo semplice e di facile comprensione.

La teoria che sta alla base di tutto è la seguente: per copiare un'immagine è sufficiente descrivere un rettangolo

IN BASIC

```
10 CLEAR 200,61999!
20 SCREEN 2
30 GOSUB 64000' CARICA L/M
40 DEFUSR=62000!' START L/M
50 CIRCLE (100,100),20
60 PAINT (100,100)
70 Y1=&HF280
80 POKE Y1,80:POKE Y1+1,80
90 POKE Y1+2,120:POKE Y1+3,120
100 POKE Y1+4,50:POKE Y1+5,30
110 A=USR(0)
120 POKE Y1+4,130
130 A=USR(0)
140 POKE Y1+5,150
150 A=USR(0)
999 BEEP:GOTO 999
64000 '************************
64001 '*   DISPLAY COPIER    *
64002 '*   ROUTINE IN L/M    *
64003 '*         BY          *
64004 '*   ORIGINAL SOFT.    *
64005 '************************
64006 DATA 3A,80,F2,47,3A,84,F2,32,86,F2
,3A,81,F2,4F,C5,58,16,00,06,00,CD,DF,15,
CD,47,16,32,F2,F3,ED,4B,84,F2,58
64007 DATA 16,00,06,00,CD,DF,15,CD,7E,16
,C1,3A,84,F2,3C,32,84,F2,3A,83,F2,0C,B9,
30,D3,3A,85,F2,3C,32,85,F2,3A,86
64008 DATA F2,32,84,F2,3A,82,F2,04,B8,30
,B5,C9,00,00,00,00,00,00,00
64009 RESTORE 64006
64010 FOR N=&HF230 TO &HF286
64011 READ A$:POKE N,VAL("&H"+A$)
64012 NEXT N
64013 RETURN
```





in cui è inscritta la figura e trasferire il suo contenuto in un'altra zona del video conosciute le coordinate del punto dove trasferire l'angolo superiore sinistro del rettangolo.

Supponiamo di aver definito una figura in un rettangolo le cui coordinate dell'angolo superiore sinistro siano X1, Y1 e quelle dell'angolo inferiore destro X2, Y2 e di trasferire il contenuto di questo rettangolo a partire dal punto di coordinate XM, YM. Il programma in Basic che esegue la copia dell'immagine sarà il seguente:

```
100 FOR Y=Y1 TO Y2
110 XA=XM
120 FOR X=X1 TO X2
130 C=POINT (X, Y)
140 PSET (XM, YM), C
150 XM=XM+1
160 NEXT X
170 YM=YM+1
180 XM=XA
190 NEXT Y
```

Il programma legge il colore di tutti i pixels che compongono il rettangolo (linea 130) e setta i punti con gli stessi colori nella parte del video dove ricopiare l'immagine (linea 140). È chiaro che questo algoritmo risulta essere lento nella sua esecuzione, soprattutto se il rettangolo ha una superficie grande, ma di facile comprensione. Il programma in Assembler Z80 sfrutta la stessa logica del programma Basic richiamando alcune routine presenti in ROM delle quali parleremo più avanti. La sua lunghezza è di soli 87 bytes e si colloca a partire dall'indirizzo &HF230 (62000 in decimale) ma è facilmente rilocabile visto che contiene istruzioni di salto relativo.

I LISTATI BASIC E ASSEMBLER

Per caricare la routine utilizzate il listato 1 che è in Basic oppure, se disponete di un assemblatore, il listato 2.

Il programma Basic contiene anche una piccola dimostrazione, raccomandiamo a tutti di prenderla in visione per comprendere maggiormente la potenzialità di questa routine.

Il copiatore di immagini funziona sia in SCREEN 2 (modo grafico 1) che in SCREEN 3 (modo grafico 2). Per utilizzare la routine è sufficiente settare i valori delle coordinate del rettangolo in cui è contenuto il disegno e quelle dove copiarlo. Le locazioni di tali coordinate sono:

X1=&HF281 Y1=&HF280
X2=&HF283 Y2=&HF282
XM=&HF284 YM=&HF285

PER UNA PORZIONE DI SCREEN

Supponiamo di aver già caricato in memoria la routine in linguaggio macchina e di aver definito il suo start con l'istruzione DEFUSR=&HF230; volendo copiare una porzione di screen, il cui rettangolo di definizione ha coordinate 10,15 (X1,Y1) e 105,120 (X2,Y2), a partire dal punto 110,30 (XM,YM), useremo le seguenti istruzioni Basic per attivare la routine:

```
 1                                      ORG   &2000
 2                                      LOAD  &2000
 3  F230 3A80F2                         LD    A,(Y1)
 4  F233 47                             LD    B,A
 5  F234 3A84F2    FORY:                LD    A,(XM)
 6  F237 3286F2                         LD    (XA),A
 7  F23A 3A81F2                         LD    A,(X1)
 8  F23D 4F                             LD    C,A
 9  F23E C5        FORX:                PUSH  BC
10  F23F 58                             LD    E,B
11  F240 1600                           LD    D,0
12  F242 0600                           LD    B,0
13  F244 CDDF15                         CALL  15DFH
14  F247 CD4716                         CALL  1647H
15  F24A 32F2F3                         LD    (0F3F2H),A
16  F24D ED4B84F2                       LD    BC,(XM)
17  F251 58                             LD    E,B
18  F252 1600                           LD    D,0
19  F254 0600                           LD    B,0
20  F256 CDDF15                         CALL  15DFH
21  F259 CD7E16                         CALL  167EH
22  F25C C1                             POP   BC
23  F25D 3A84F2    .                    LD    A,(XM)
24  F260 3C                             INC   A
25  F261 3284F2                         LD    (XM),A
26  F264 3A83F2                         LD    A,(X2)
27  F267 0C                             INC   C
28  F268 B9                             CP    C
29  F269 30D3                           JR    NC,FORX
30  F26B 3A85F2                         LD    A,(YM)
31  F26E 3C                             INC   A
32  F26F 3285F2                         LD    (YM),A
33  F272 3A86F2                         LD    A,(XA)
34  F275 3284F2                         LD    (XM),A
35  F278 3A82F2                         LD    A,(Y2)
36  F27B 04                             INC   B
37  F27C B8                             CP    B
38  F27D 30B5                           JR    NC,FORY
39  F27F C9                             RET
40  F280 00        Y1:                  DEFB  0
41  F281 00        X1:                  DEFB  0
42  F282 00        Y2:                  DEFB  0
43  F283 00        X2:                  DEFB  0
44  F284 00        XM:                  DEFB  0
45  F285 00        YM:                  DEFB  0
46  F286 00        XA:                  DEFB  0
47                                      END
```

POKE &HF281,10:POKE &HF280,15:'setta X1,Y1
POKE &HF283,105:POKE &HF282,120:'setta X2,Y2
POKE &HF284,110:POKE &HF285,30:'setta XM,YM
A=USR(0):'esegue la routine di copiatura

L'AREA DI DEFINIZIONE

È molto importante che il punto dove trasferire l'immagine sia al di fuori del rettangolo di definizione della figura. Se così non fosse non solo l'immagine copiata risulterebbe errata ma anche quella originale verrebbe alterata; quindi attenzione a settare i valori di XM,YM. Per esempio, con X1,Y1 pari a 10,20 e X2,Y2 a 100,40, il punto 50,30 come XM,YM non andrebbe bene perché appartiene al rettangolo di definizione immagine.

Passiamo ora ad analizzare il listato Assembler. Come è già stato detto in precedenza il programmino in Assembler esegue esattamente il programma Basic simulando le sue istruzioni. Il registro B funge da contatore del ciclo a conteggio FOR Y-NEXT (linea 4), mentre il registro C quello del ciclo FOR X-NEXT (linea 8). L'istruzione Basic di linea 110 (XA=XM) è simulata alle linee 5 e 6, così come l'istruzione NEXT X alle linee 26-29 e così via. Ma vediamo come è stato possibile simulare le istruzioni Basic POINT e PSET. Come già detto, abbiamo sfruttato delle routine presenti in ROM. Per simulare l'istruzione POINT si è fatto uso delle istruzioni CALL 15DFH e CALL 1647H (linee 13 e 14).

LA MAPPA DELLE COORDINATE

La prima chiamata consente di mappare le coordinate X,Y di un punto grafico al relativo indirizzo fisico di memoria. I valori d'ingresso sono il registro C, contenente la coordinata X, ed il registro E contenente la coordinata Y. La routine predispone le variabili di sistema CLOC e CMASK rispettivamente contenenti l'indirizzo in VRAM del punto e la sua mascheratura nel byte. La seconda chiamata, CALL 1647H, ritorna nell'accumulatore il colore del pixel specificato dalle variabili CLOC e CMASK, già settate con la CALL 15DFH.

Ecco realizzata in linguaggio macchina la funzione POINT.

Per quanto riguarda invece l'istruzione PSET, si è utilizzata ancora l'istruzione CALL 15DFH per predisporre le variabili CLOC e CMASK alla nuova posizione del disegno, e secondariamente la subroutine residente a partire dall'indirizzo 167EH (linea 21). Quest'ultima setta il corrente pixel (specificato da CLOC e CMASK) al colore il cui codice è contenuto all'indirizzo OF3F2H. Tale codice è quello letto con la CALL 1647H e restituito nel registro A, quindi immediatamente caricato all'indirizzo appena citato (linea 15). Così invece è stata realizzata l'istruzione PSET.

ROUTINE

DAI DATI DI STRINGA A QUELLI NUMERICI

PER I VOSTRI PROGRAMMI PUÒ ESSERE UTILISSIMO AVERE IL VALORE ALGEBRICO DI UNA STRINGA...

Ecco a voi una routine utilissima da inserire nei vostri programmi. La sua funzione è di fornire il valore algebrico di stringhe contenenti numeri e operazioni matematiche.

Ma lasciate che vi spieghi meglio. Come molti di voi sapranno, se assegnate una variabile con dei caratteri numerici, usando un'istruzione del tipo:

B$="123"

la variabile assegnata, in questo caso B$, viene a tutti gli effetti trattata come stringa, non potendo quindi su di essa effettuare nessuna operazione di tipo matematico, nemmeno le quattro operazioni fondamentali. Ad esempio il programma seguente:

```
10 A$="123"
20 B$="456"
30 C$=A$+B$
```

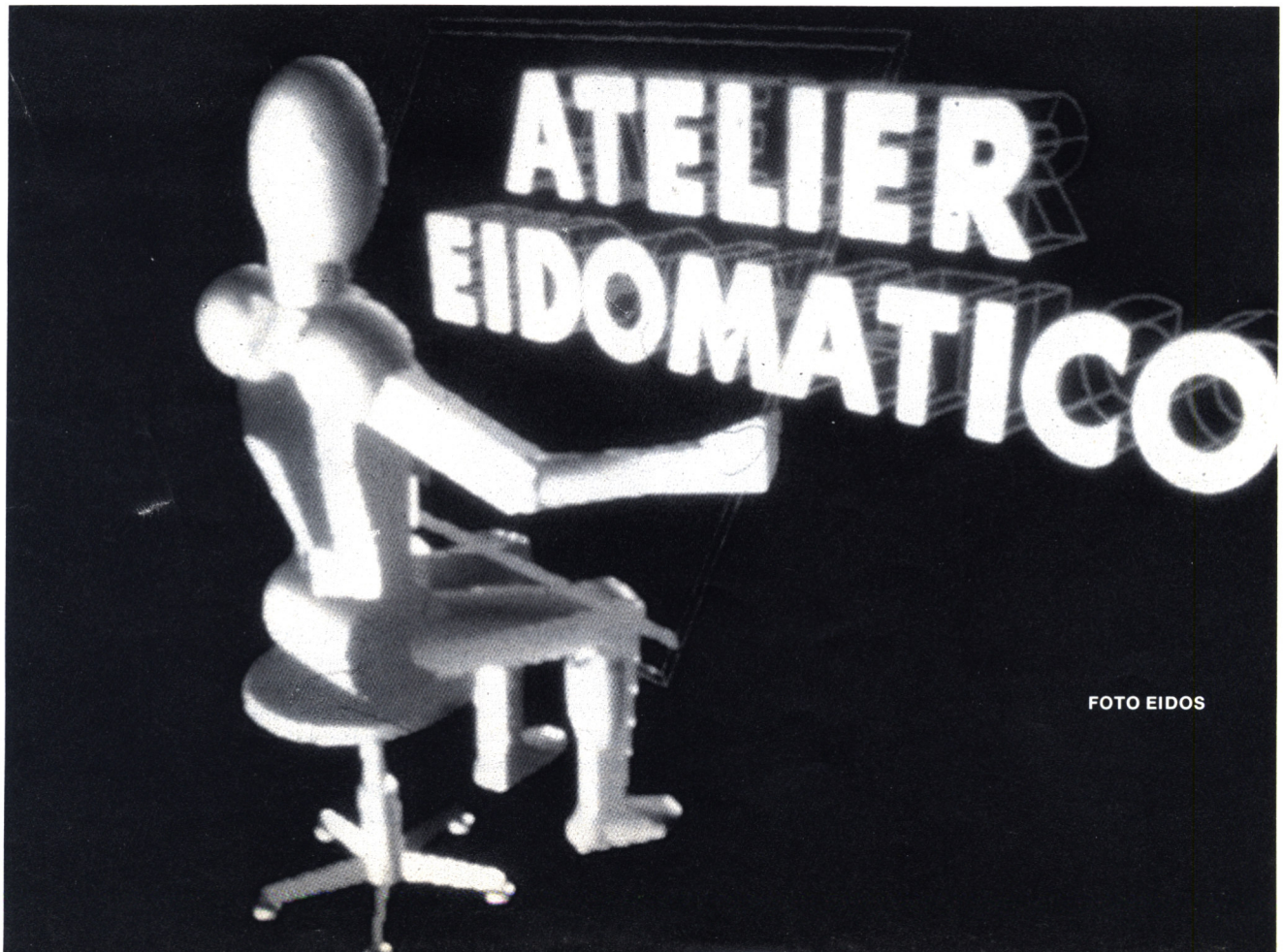


FOTO EIDOS

40 PRINT C$

darà come risultato 123456, cioè anziché sommare il "numero" 123 col "numero" 456, abbiamo unito le due stringhe A$ e B$ e il risultato ottenuto è anch'esso un dato di stringa.

Al fine di poter gestire una stessa variabile a volte come una stringa di caratteri numerici e a volte come un numero vero e proprio, il potente BASIC MSX ci mette a disposizione l'istruzione VAL( ), dove all'interno delle parentesi va posta la stringa di caratteri numerici in questione e come risultato otteniamo il valore numerico della stringa inserita.

VEDIAMO UN ESEMPIO

Ma lasciate anche qui semplificare la spiegazione attraverso l'ausilio di un esempio.

Consideriamo il seguente programma:

```
10 A$="123"
20 B$="456"
30 C=VAL (A$)+VAL(B$)
40 PRINT C
```

esso darà come risultato la variabile C=579, trattata stavolta come numerica.

Che cosa è mai successo rispetto al programma precedente? La risposta a questa domanda non risiede nei poteri occulti della magia nera, ma semplicemente nelle capacità del nostro BASIC MSX.

Alla linea 30, infatti, con l'istruzione VAL, abbiamo convertito in dati numerici i dati di stringa A$ e B$ e li abbiamo poi sommati, trattandoli adesso come i semplici e normali numeri a cui siamo tanto abituati.

Ed eccoci subito tentati di fare i furbi, abbreviando e semplificando il programmino precedente nel seguente modo:

```
10 A$="123+456"
20 PRINT VAL (A$)
```

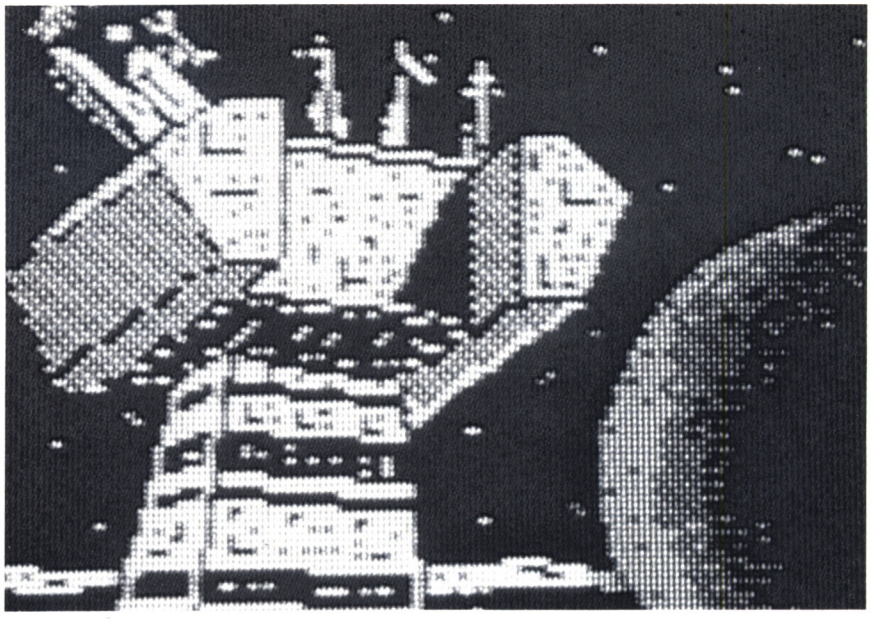
aspettandoci come risposta di nuovo il numero 579, cioè la somma numerica di 123 e 456.

QUASI UNO SHOCK

Provando però a far girare quest'ultimo programmino esemplificativo, subiamo uno shock tremendo: il valore stampato non è 579 ma è bensì 123!

Qualcuno di voi penserà che stavolta si tratti veramente di magia nera, anzi, nerissima!

In verità, anche stavolta, le cause di ciò che è successo risiedono nel tipo di



linguaggio parlato dal nostro microelaboratore. Infatti, anche se la funzione VAL implementata nel BASIC della Microsoft è molto utile e potente, essa non prevede l'esecuzione di eventuali operazioni algebriche all'interno delle stringhe in essa contenute. Ciò che quindi succede è che, se viene incontrato un carattere non numerico all'interno della stringa, viene restituito il valore della sola parte di stringa precedente il carattere in questione. Anche qui la cosa migliore è vedere alcuni esempi. L'istruzione:

C=VAL ("12/2")

fornisce come risultato il valore 12 e lo stesso vale per:

C=VAL ("12+2")
C=VAL ("12*2")
C=VAL ("12—2")

IL PROGRAMMA

```
64000 'Routine sostitutiva istr. VAL
64010 B$=A$
64020 LN=LEN(A$)
64030 '------ Ricerca parentesi ------
64040 FOR I=2 TO LN
64050 PA$=MID$(B$,I,1)
64055 U=I
64060 IF PA$=")" THEN 64240
64070 NEXT I
64080 ' Non ho trovato par. chiuse
64090 GOSUB 64120
64100 RETURN
64110 'Ricerca 4 operaz. fondam.
64120 FOR I=2 TO LEN(B$)-1
64130 W=I
64140 PA$=MID$(B$,I,1)
64150 IF PA$="+" GOTO 64340
64160 IF PA$="-" GOTO 64370
64170 IF PA$="/" GOTO 64400
64180 IF PA$="*" GOTO 64430
64190 NEXT I
64200 '-- Non ho trovato operazioni--
64210 B=VAL(B$)
64220 RETURN
```

segue

```
64230 ' Ho trovato parentesi chiusa
64240 FOR I=U-1 TO 1 STEP -1
64250 V=I
64260 PA$=MID$(B$,I,1)
64270 IF PA$="(" THEN B$=MID$(B$,V+1,U-I
-1):GOTO 64300
64280 NEXT I
64290 ' ----- Ho isolato B$ ------
64300 GOSUB 64120
64310 A$=LEFT$(A$,V-1)+STR$(B)+RIGHT$(A$
,LN-U)
64320 GOTO 64010
64330 ' ----- Ho trovato '+' ------
64340 B=VAL(LEFT$(B$,W-1))+VAL(RIGHT$(B$
,LEN(B$)-W))
64350 RETURN
64360 ' ----- Ho trovato '-' ------
64370 B=VAL(LEFT$(B$,W-1))-VAL(RIGHT$(B$
,LEN(B$)-W))
64380 RETURN
64390 ' ----- Ho trovato '/' ------
64400 B=VAL(LEFT$(B$,W-1))/VAL(RIGHT$(B$
,LEN(B$)-W))
64410 RETURN
64420 ' ----- Ho trovato '*' ------
64430 B=VAL(LEFT$(B$,W-1))*VAL(RIGHT$(B$
,LEN(B$)-W))
64440 RETURN
```

L'istruzione:

C=VAL ("LOG(10)")

oppure anche il programma:

10 A$="LOG(10)"
20 C=VAL(A$)
30 PRINT C

forniranno invece come risultato il valore 0.

Ecco dunque che può nascere un'esigenza: vogliamo gestire stringhe di caratteri numerici contenenti operazioni algebriche, cioè per intenderci stringhe del tipo:

A$="(12+3)/5"

ma vogliamo anche la possibilità di ottenere il risultato di tali operazioni, per esempio perché desideriamo far girare un programma contenente delle istruzioni del tipo:

10 A$="(16+4)/4"
20 PRINT A$;"=";
valore calcolato di A$

dove, per ottenere il valore calcolato di A$, non potendo usare l'istruzione VAL, dobbiamo per forza ricorrere a qualche artificio.

ED ECCO LA ROUTINE

Ecco dunque che vi presentiamo la routine annessa, che dà proprio la possibilità di calcolare il valore di stringhe contenenti operazioni algebriche. Analizziamone il listato:

come possiamo vedere dalle linee 64010-64020, la nostra routine utilizza le variabili A$ e B$ come stringhe contenenti l'espressione algebrica che vogliamo calcolare, è quindi essenziale, al fine di un corretto funzionamento del vostro programma e della stessa routine, che tali nomi non vengano assegnati ad altre variabili.

Ricordatevi inoltre, nel vostro programma, di mettere in A$ la stringa di cui vi interessa il valore, cioè, se per esempio avete:

500 T$="(16+4)/(3+2)"

dovete, prima dell'istruzione GOSUB 64000, inserire l'istruzione A$=T$, cosicché, in totale, il vostro programma contenga la seguente successione:

500 T$="(16+4)/(3+2)"
510 T$+A$
520 GOSUB 64000

È chiaro che non è obbligatorio che l'assegnazione di T$ sia immediatamente precedente all'istruzione A$=T$.

Le linee 64030-64070 ricercano eventuali parentesi presenti all'interno di A$, in modo da poter eseguire nel giusto ordine le operazioni presenti. Notiamo a tale proposito che la presenza delle parentesi per separare un'operazione da un'altra è fondamentale, cosicché il valore calcolato per la stringa A$="12+4/2" sarà sbagliato, mentre quello calcolato per A$="12+(4/2)" sarà esatto. Nel dubbio, quindi, abbondate con le parentesi piuttosto che rischiare calcoli sbagliati.

Le linee 64230-64280 isolano il contenuto della parentesi più interna. State sempre attenti a chiudere tante parentesi quante ne avete aperte.

Le linee 64080-64090 provvedono al calcolo della stringa nel caso non vengano trovate parentesi, cioè nel caso che sia presente una sola operazione matematica all'interno di essa.

La linea 64100 contiene un'istruzione RETURN che è quella che rimanda, a routine terminata, al vostro programma principale. Chi quindi chiamasse la nostra routine 64000 con un'istruzione GOTO deve togliere questo RETURN e sostituirlo semmai con un GOTO che mandi a un'istruzione desiderata.

Le linee 64110-64190 ricercano le quattro operazioni fondamentali e le eseguono. L'esecuzione vera e propria avviene dalla linea 64330 alla 64440. State molto attenti che il valore calcolato della stringa A$ viene posto nella variabile B. È questa che dovete pertanto usare nel resto del programma se volete su di essa eseguire altre operazioni.

Chi infine volesse cambiare qualcuno dei nomi di variabile da noi usato può ovviamente farlo, ecco a tal fine la lista delle variabili presenti nella Routine:

A$: contiene la stringa di partenza;
B$: contiene parti della stringa di partenza;
B: contiene il valore numerico calcolato di A$;
PA$: serve all'analisi del contenuto di A$ e B$;
U, V, W: puntatori atti al posizionamento di B$ all'interno di A$;
I: contatore dei cicli FOR-NEXT.

Chi volesse provare la routine può farlo utilizzando il seguente programma:

10 INPUT A$
20 GOSUB 64000
30 PRINT B
40 GOTO 10

Provare per credere! ■

Le linee 64030-64070 ricercano eventuali parentesi presenti all'interno di A$, in modo da poter eseguire nel giusto ordine le operazioni presenti. Notiamo a tale proposito che la presenza delle parentesi per separare un'operazione da un'altra è fondamentale, cosicché il valore calcolato per la stringa A$="12+4/2" sarà sbagliato, mentre quello calcolato per A$="12+(4/2)" sarà esatto. Nel dubbio, quindi, abbondate con le parentesi piuttosto che rischiare calcoli sbagliati.

Le linee 64230-64280 isolano il contenuto della parentesi più interna. State sempre attenti a chiudere tante parentesi quante ne avete aperte.

Le linee 64080-64090 provvedono al calcolo della stringa nel caso non vengano trovate parentesi, cioè nel caso che sia presente una sola operazione matematica all'interno di essa.

La linea 64100 contiene un'istruzione RETURN che è quella che rimanda, a routine terminata, al vostro programma principale. Chi quindi chiamasse la nostra routine 64000 con un'istruzione GOTO deve togliere questo RETURN e sostituirlo semmai con un GOTO che mandi a un'istruzione desiderata.

Le linee 64110-64190 ricercano le quattro operazioni fondamentali e le eseguono. L'esecuzione vera e propria avviene dalla linea 64330 alla 64440. State molto attenti che il valore calcolato della stringa A$ viene posto nella variabile B. È questa che dovete pertanto usare nel resto del programma se volete su di essa eseguire altre operazioni.

Chi infine volesse cambiare qualcuno dei nomi di variabile da noi usato può ovviamente farlo, ecco a tal fine la lista delle variabili presenti nella Routine:

A$: contiene la stringa di partenza;
B$: contiene parti della stringa di partenza;
B: contiene il valore numerico calcolato di A$;
PA$: serve all'analisi del contenuto di A$ e B$;
U, V, W: puntatori atti al posizionamento di B$ all'interno di A$;
I: contatore dei cicli FOR-NEXT.

Chi volesse provare la routine può farlo utilizzando il seguente programma:

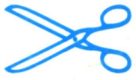
```
10 INPUT A$
20 GOSUB 64000
30 PRINT B
40 GOTO 10
```

Provare per credere!

■

SPLENDIDO

UN LOOK
COLORATO
PER LA TUA
CASSETTA



RITAGLIA
LUNGO
IL BORDO
SEGNATO
IN NERO
E
PIEGA
SEGUENDO
IL
TRATTEGGIO
INDICATO

PERSONALIZZA
LA
CASSETTA
CON IL
TUO NOME

MSX COMPUTER
MAGAZINE
PER LA TUA
SOFT-TECA