Basic modulare

di Tino Costantini - Pescara

Un linguaggio, per permettere una buona programmazione, deve necessariamente presentare un completo set di istruzioni strutturate e un alto livello di modularità, in modo da poter dividere i programmi in un programma principale e una serie di sottoprogrammi parametrici.

I sottoprogrammi rendono più facile e veloce la realizzazione di un programma; inoltre i programmi risultano più comprensibili e facili da modificare.

Come detto, un programma presenta una serie di sottoprogrammi parametrici. I parametri dichiarati in un sottoprogramma sono detti formali, quelli che compaiono nelle chiamate sono detti attuali (o effettivi); naturalmente ci deve essere corrispondenza tra di loro, sia nel numero che nel tipo.

Per quanto riguarda le altre variabili, quelle dichiarate nel programma principale sono dette globali mentre quelle dichiarate nel sottoprogramma sono dette locali.

Quindi in un sottoprogramma vi possono essere tre tipi di variabili: i parametri formali, le variabili locali e le variabili globali.

I parametri formali sono di due tipi: valore e variabile. Nel primo caso viene assegnato al parametro formale il valore del parametro attuale, che quindi deve essere un'espressione; nel secondo caso, invece, tramite il parametro formale si agisce direttamente sul parametro attuale, che quindi deve essere una variabile.

Un esempio per chiarire tutto quello detto finora: due matrici possono essere visualizzate con una stessa routine passando la matrice e le dimensioni. Bisogna però fare attenzione: in caso di parametro formale variabile, se si eseguono delle operazioni su di esso si perde il valore originale.

In pratica, in questo modo, si aggiungono comandi a quelli già esistenti; quindi si possono preparare biblioteche di routine, ad esempio per operazioni con matrici, che possono essere riutilizzate in qualsiasi momento. La modularità, unita alla possibilità da parte di una routine di richiamare se stessa, permette la ricorsività, molto utile per determinate applicazioni.

Questo programma è disponibile su cassetta presso la redazione. Vedere l'elenco dei programmi disponibili e le istruzioni per l'acquisto a pag. 120.

Il Basic

Visti questi vantaggi, si è cercato di portarli sul Basic ed infatti oggi sono presenti diversi Basic strutturati, che però devono essere divisi in due tipi ben distinti.

Da una parte i compilati, che sono completi sia per la strutturazione che per la modularità, però perdono il vantaggio essenziale del Basic, cioè la praticità; dall'altra i non compilati, che sono strutturati ma non risolvono il problema della modularità. Quindi, anche se questi ultimi migliorano nettamente il Basic, i sottoprogrammi non sono completamente indipendenti, con tutte le conseguenze negative che questo comporta.

Il metodo che descriviamo risolve tutti i casi di variabili locali e parametri formali valore e variabile, sia per le variabili semplici che per gli array; quindi è possibile avere un Basic strutturato e modulare senza bisogno di compilazione e, per giunta, sfruttando l'organizzazione attuale delle variabili.

La chiamata di una generica routine per il display di una matrice potrebbe avere questo aspetto:

```
 50 ....
 55 GOSUB MAT-DISPLAY(A,M,N)
 60 ....
300 ROUTINE   MAT-DISPLAY(VARAA();P,Q)
305 VAR I,J
310 FOR I=1 TO P
315 FOR J=1 TO Q
320 PRINT TAB(6*(J-1)+1);AA(I,J);
325 NEXT J
330 PRINT
335 NEXT I
340 RETURN
```

È evidente la maggior chiarezza del listato e la completa indipendenza di un sottoprogramma dal programma principale.

Prima di andare avanti, abbiamo ritenuto utile stabilire una convenzione per snellire il discorso. Le variabili locali e i parametri formali valore li indicheremo con la sigla LOC, mentre con la sigla VAR ci riferiremo ai parametri formali variabile. I primi sono trattati insieme perché consistono entrambi nella inizializzazione di nuove variabili, con la differenza che nei secondi viene ripreso un valore. Invece nel caso VAR non c'è nuova occupazione di memoria.

Metodo teorico

A questo punto un rapido richiamo alla gestione delle variabili in Applesoft. Ci sono due zone: una per le variabili semplici e un'altra per gli array. In pagina zero ci sono tre puntatori: lomem, che punta all'inizio della zona variabili semplici, il puntatore inizio array e il puntatore fine array. Quando viene inizializzata una variabile viene posta in coda a quelle del suo tipo.

Vediamo ora il metodo usato. Innanzitutto c'è bisogno di uno stack per il quale verrà utilizzata pag. 3; vedremo poi cosa vi verrà immagazzinato. Vediamo ora come realizzare i casi LOC e VAR visti prima.

Caso LOC

Per risolvere tutti i casi bisognava trovare un sistema per rendere dinamica l'allocazione statica delle variabili.

Ecco come si procede per le variabili semplici: alla partenza del programma viene spostato in avanti LOMEM, lasciando una certa zona libera (vedi fig. 1); non è necessario che questa zona sia molto grande, perché viene occupata e rilasciata ad ogni entrata e uscita di routine.

Se per chiamate successive dovesse occuparsi tutta, un controllo provvederà a spostare le variabili in avanti per aumentarla. Quando si inizializzano delle variabili tipo LOC, queste vengono poste davanti a LOMEM, questo spostato all'indietro e lo spostamento immagazzinato nello stack.

In questo modo, quando il Basic scorre le variabili, troverà prima le locali; se ve ne erano altre con lo stesso nome rimarranno temporaneamente nascoste. All'uscita dalla routine ci sarà un rilascio di memoria,

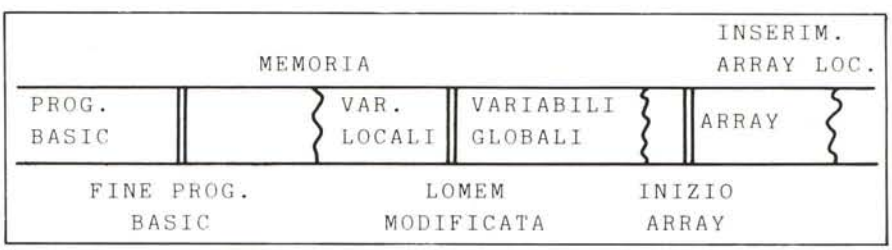

Figura 1 - Disposizione dei puntatori e contenuto delle aree dati.

```
10  REM  ********************
11  REM  * VARIABILI LOCALI *
12  REM  *                  *
13  REM  * TINO E GABRIELE  *
14  REM  *    COSTANTINI    *
15  REM  *                  *
18  REM  ********************
19  REM
40  GOSUB 1000: GOTO 2000
45  REM  ENTRATA
50 S3 = 1:S4 = 1
53 S$ = VL$:Y2 = 5:Y4 = 2: GOSUB
    500
59 S4 = Y3: IF  ASC (VL$) = 86 THEN
    76
71  GOSUB 570
74 ST = ST + 1: POKE ST,1: RETURN

76  IF S3 = 1 THEN SW = 1
77  IF S3 > 1 THEN SW = 2
78  IF SW = 1 THEN S3 = S4
80 S$ = PA$:Y2 = 1:Y4 = 1: GOSUB
    500
86  GOSUB 660
89  IF SW = 2 THEN S3 = S3 + 1: GOSUB
    570
92 ST = ST + 1: POKE ST,SW
95  RETURN : REM

500  REM  VAL
503 Y3 = 0:LU =  LEN (S$)
505  FOR Y1 = Y2 TO LU
510 Y3 = Y3 + 1:Y(Y4,Y3) =  ASC (
    MID$ (S$,Y1,1))
515 Y3 = Y3 + 1:Y(Y4,Y3) = 0
520  FOR Y2 = Y1 + 1 TO Y1 + 3
525  IF Y2 > LU THEN Y2 = LU + 5:
    Y1 = LU: GOTO 560
530 A =  ASC ( MID$ (S$,Y2,1))
535  IF A > 47 THEN Y(Y4,Y3) = A:
     NEXT Y2
540  IF A = 44 OR A = 45 THEN Y1 =
    Y2:Y2 = LU + 5: GOTO 560
545  IF A = 37 THEN Y(Y4,Y3 - 1) =
    Y(Y4,Y3 - 1) + 128
550  IF A < 38 THEN Y(Y4,Y3) = Y(
    Y4,Y3) + 128: NEXT Y2
555  IF A = 47 THEN S3 = Y3:Y1 =
    Y2 + 4:Y2 = LU + 5
560  NEXT Y2,Y1
562  RETURN : REM
570  REM  LOC-IN
573 BY = 7 * (S4 - S3 + 1) / 2
575  POKE ST + 1,BY: POKE ST + 2,
    1:ST = ST + 2
580 VL = LM - BY
585  FOR Y1 = S3 TO S4 STEP 2
590  POKE VL,Y(2,Y1)
595  POKE VL + 1,Y(2,Y1 + 1)
600  FOR Y2 = VL + 2 TO VL + 6
605  POKE Y2,0
610  NEXT
615 VL = VL + 7: NEXT Y1
620 LM = LM - BY: GOSUB 630
625  RETURN
630  REM  POKE-LM
635 Y2 =  INT (LM / 256):Y3 = LM -
    (Y2 * 256)
640  POKE 768,Y3: POKE 769,Y2
645  POKE 105, PEEK (768)
650  POKE 106, PEEK (769)
655  RETURN : REM

660  REM   VAR-IN
665 NU = S3 / 2:S2 = ST + (NU * 4
    ):S1 = ST:LV = LV + 1:NN = 0

670 PA =  PEEK (107) +  PEEK (108
    ) * 256:FA =  PEEK (109) +  PEEK
    (110) * 256
675 LA = PA:N1 =  PEEK (LA):N2 =
     PEEK (LA + 1)
680  FOR Y2 = 1 TO 2: FOR Y1 = 1 TO
    S3 STEP 2
685  IF N1 = Y(Y2,Y1) AND N2 = Y(
    Y2,Y1 + 1) THEN  ON Y2 GOSUB
    715,755:Y1 = S3:Y2 = 2
690  NEXT Y1,Y2
695 PA =  PEEK (LA + 2) +  PEEK (
    LA + 3) * 256 + LA
700  IF (PA <  > FA) THEN 675
705  POKE S2 + 1,NU: POKE S2 + 2,
    NN: POKE S2 + 3,2:ST = S2 +
    3
710  RETURN

715  REM  Y2=1
720  FOR Y3 = 0 TO 1
725  POKE LA + Y3,Y(2,Y1 + Y3)
730  POKE S1 + 1 + Y3,Y(2,Y1 + Y3
    )
735  POKE S1 + 3 + Y3,Y(1,Y1 + Y3
    )
740  NEXT
745 S1 = S1 + 4
750  RETURN
755  REM  Y2=2
760 S2 = S2 + 1: POKE S2,Y(2,Y1)
765  POKE LA,LV:NN = NN + 1
770  RETURN : REM

800  REM  USCITA
801 NU =  PEEK (ST):ST = ST - 1
802  IF  PEEK (ST) <  > 1 THEN 81
    2
804  REM  LOC-FI
806 BY =  PEEK (ST - 1):ST = ST -
    2
808 LM = LM + BY: GOSUB 630
810  IF NU = 1 THEN  RETURN
812  REM  VAR-FI
816 NU =  PEEK (ST - 2):NN =  PEEK
    (ST - 1):S2 = ST - 2 - NN:S1
     = S2 - (NU * 4):ST = S2
818 PA =  PEEK (107) +  PEEK (108
    ) * 256:FA =  PEEK (109) +  PEEK
    (110) * 256
820 LA = PA:N1 =  PEEK (LA):N2 =
     PEEK (LA + 1)
822  IF N1 =  PEEK (S1) THEN  IF
    N2 =  PEEK (S1 + 1) THEN  IF
    S1 < ST THEN  GOSUB 834
824  IF N1 = LV THEN  GOSUB 840
826 PA =  PEEK (LA + 2) +  PEEK (
    LA + 3) * 256 + LA
828  IF (PA <  > FA) THEN 820
830 LV = LV - 1:ST = ST - (NU * 4
    ) - 1
832  RETURN
834  REM  LA-S1
836  POKE LA, PEEK (S1 + 2): POKE
    LA + 1, PEEK (S1 + 3):S1 = S
    1 + 4
838  RETURN
840  REM  LA-S2
842  POKE LA, PEEK (S2):S2 = S2 +
    1
844  RETURN : REM

1000  REM  PROG-IN
1001  POKE 106, PEEK (106) + 1: POKE
    108, PEEK (106): POKE 110, PEEK
    (106)
1003 LM = 0:IA = 0:FA = 0:BY = 0:
    ST = 769
1004 LU = 0:LA = 0:PA = 0:N1 = 0:
    N2 = 0:LV = 0
1005 S1 = 0:S2 = 0:NU = 0:NN = 0:
    SW = 0:A = 0
1007 Y1 = 0:Y2 = 0:Y3 = 0:Y4 = 0
1008  DIM Y(2,20)
1010 LM =  PEEK (105) +  PEEK (10
    6) * 256:VL = LM
1050  RETURN : REM

1500  REM  FINE ROUTINE SISTEMA

2000  REM  PROGRAMMA PRINCIPALE
2020  HOME :F = 0
2025  INPUT " FATTORIALE DI ";N
2030  IF N > 10 THEN 2020
2032  PRINT
2035 K1 = N: GOSUB 3000
2040  PRINT : PRINT " IL FATTORIA
    LE E' ";F
2045  PRINT : PRINT " PREMI UN TA
    STO ";: GET V$: HOME
2047  REM
2050 M = 4:N = 3: DIM A(M,N),B(N,
    M)
2052  DATA  1,0,8,1,5,2,7,9,1,3,5
    ,6
2055  FOR I = 1 TO M: FOR J = 1 TO
    N
2057  READ A(I,J): NEXT J,I
2058  REM
2060  PRINT " MATRICE A": PRINT
2065 PA$ = "A":K1 = M:K2 = N: GOSUB
    3100
2067  REM
2070  PRINT : PRINT " STO ESEGUEN
    DO LA TRASPOSTA": PRINT
2075 PA$ = "A,B":K1 = M:K2 = N: GOSUB
    3200
2077  REM
2080  PRINT : PRINT " LA TRASPOST
    A E'": PRINT
2085 PA$ = "B":K1 = N:K2 = M: GOSUB
    3100
2095  END : REM

3000  REM  FATTORIALE
3005  REM
3010  REM  VARIABILI LOCALI
3015 VL$ = "LOC:N": GOSUB 45
3020 N = K1
3025  REM  INIZIO ROUTINE
3030  REM
3040  PRINT  TAB( 15 - N);"N=";N;
    "  F=";F
3045  IF N > 0 THEN K1 = N - 1: GOSUB
    3000:F = F * N
3050  IF N = 0 THEN F = 1
3055  PRINT  TAB( 15 - N);"N=";N;
    "  F=";F
3070  REM
3075  REM  FINE-ROUTINE
3080  GOSUB 800
3085  RETURN : REM

3100  REM  MAT-DISPLAY
3105  REM
3107  REM  VARIABILI LOCALI
3110 VL$ = "VAR:AA/LOC:M,P-I,J": GOSUB
    45
3115 M = K1:P = K2
3121  REM  INIZIO ROUTINE
3122  REM
3125  FOR I = 1 TO M
3127  FOR J = 1 TO P
3130  PRINT  TAB( 6 * (J));AA(I,J
    );
3140  NEXT J: PRINT : NEXT I: PRINT

3160  REM
3165  REM  FINE ROUTINE
3170  GOSUB 800
3180  RETURN : REM

3200  REM  MAT-TRASP
3205  REM
3207  REM  VARIABILI LOCALI
3210 VL$ = "VAR:XX,Z/LOC:M,P-I,J"
    : GOSUB 45
3215 M = K1:P = K2
3221  REM  INIZIO ROUTINE
3222  REM
3225  FOR I = 1 TO M
3227  FOR J = 1 TO P
3230 Z(J,I) = XX(I,J)
3240  NEXT J,I
3260  REM
3265  REM  FINE ROUTINE
3270  GOSUB 800
3280  RETURN : REM
```

Sistema

Il sistema ha due routine principali: ENTRATA e USCITA.

La prima utilizza le routine VAL, LOC-IN, VAR-IN. VAL valuta le stringhe VL$ e PA$ e mette i codici ASCII nella matrice Y. LOC-IN legge nella matrice i nomi delle variabili di tipo LOC e le inizializza. Inoltre mette nello stack il valore dello spostamento e il codice dell'operazione. VAR-IN utilizza sempre i codici nella matrice per cambiare i nomi degli array passati con quelli locali.

ENTRATA

53 richiama VAL per valutare VL$

59 se c'è solo LOC esegue LOC-IN ed esce

76 dato che c'è sicuramente VAR deve vedere se c'è anche LOC (S3 < > 1) in uscita numero dei casi in stack

USCITA

801 legge nello stack il numero dei casi che si è avuto in entrata

802 si vede quale caso eseguire; 1 = LOC, 2 = VAR

spostando LOMEM in avanti per quante sono state le variabili inizializzate.

Per quanto riguarda gli array il metodo è identico ma, ad ogni inizializzazione, bisogna prima spostare in avanti quelli già esistenti per fare spazio. Quindi il metodo potrebbe essere velocizzato se si modificasse all'indietro la lettura degli array da parte del Basic (vedi fig. 2), in modo che i nuovi array LOC sarebbero accatastati in coda a quelli già presenti, rendendo il tempo necessario indipendente dalla lunghezza degli array già dimensionati.

Caso VAR

Occupiamoci ora del caso VAR. Il passaggio di un parametro variabile si ottiene sostituendo il nome del parametro formale al nome del parametro attuale, cioè della variabile globale su cui si vuole agire.

In questo modo, di volta in volta, si potrà agire dall'interno di una routine sul parametro attuale desiderato. Ad ogni sostituzione viene immagazzinato nello stack prima il nome del parametro formale e poi quello del parametro attuale. Il numero di byte occupati nello stack per tutte le sostituzioni è uguale perciò al numero dei parametri passati per quattro.

Un ulteriore problema che bisognava risolvere era quello che si presentava se tra le variabili globali, davanti alla variabile da passare, ve ne era già qualcuna con lo stesso nome di un parametro formale, di cui non avrebbe permesso la lettura. Questo problema è stato risolto sostituendo al primo byte del nome della variabile che interferiva un codice (livello di chiamata) in modo tale che il Basic non la riconoscesse e quindi la ignorasse.

In questo modo si risolve il problema delle variabili che interferiscono e ci si può spingere a più livelli ritrovando poi in uscita tutte le informazioni nel modo più opportuno per ripristinare la situazione iniziale.

Tutte queste sostituzioni devono essere immagazzinate nello stack. I codici utili sono 64 (65 è il codice ASCII della A), raddoppiabili considerando anche i negativi; quindi la massima profondità di chiamata è più che sufficiente per qualsiasi tipo di programma.

Il procedimento è identico sia per le variabili semplici che per gli array.

Il programma

Innanzitutto alcune puntualizzazioni.

La realizzazione del metodo teorico è stata fatta con routine in BASIC, dato il carattere puramente dimostrativo del programma. Quindi le chiamate e le dichiarazioni dei sottoprogrammi non potevano essere come quelle definitive viste nell'introduzione, ma si è dovuto ricorrere a qualche espediente. Comunque ciò che si voleva dimostrare rimane invariato, in quanto si può vedere come le routine siano completamente indipendenti dal programma principale.

Per non appesantire troppo il programma non abbiamo preso in considerazione i controlli come ad esempio quello necessario nel caso VAR per evitare lo scorrimento di tutti gli array. Inoltre abbiamo tenuto conto di due casi dei quattro possibili; caso LOC per le variabili semplici, caso VAR per gli array. Di conseguenza nella routine fattoriale viene utilizzata una variabile globale, cosa non necessaria se l'implementazione fosse stata completa.

Il programma si divide in due parti: le routine del sistema, che devono essere sempre presenti e, da linea 2000, il programma dimostrativo con le sue subroutine.

Lo stack è in pagina 3; dato che i primi due byte sono riservati ad altro, in effetti inizia dalla locazione 770.

Alla partenza il sistema sposta in avanti LOMEM di 256 byte, in modo che ci sia spazio per accatastare fino a 36 variabili di tipo LOC, e inizializza le variabili del sistema; poi salta al programma principale.

Il programma principale richiama le routine passandogli i parametri; all'ingresso della routine c'è la dichiarazione dei parametri formali e delle variabili locali, nella stringa VL$. Poi si richiama il sistema, in particolare la routine entrata, che provvede a fare tutto quello che abbiamo visto prima in teoria. Al ritorno, se c'erano dei parametri valore, vengono ripresi. Prima di uscire dalla routine viene richiamato di nuovo il sistema che ripristina la situazione di ingresso.

Ci sono delle regole da rispettare per la chiamata e per la forma delle routine.

Per quanto riguarda la chiamata le regole sono:

1) la stringa PA$, che contiene i nomi degli array da passare, non deve contenere spazi e i nomi delle variabili devono essere separati da virgola.

2) I valori devono essere assegnati a variabili inizianti per K, che dovranno essere riutilizzate nella routine.

Per quanto riguarda le routine le regole sono:

1) la stringa VL$ deve contenere nell'ordine prima la dicitura VAR:, poi il nome delle variabili con le regole di PA$, la sbarra, la dicitura LOC:, i nomi delle variabili con le solite regole. Per separatore dei nomi delle variabili si può usare anche il trattino, che conviene usare per chiarezza per separare nel caso LOC i parametri formali dalle variabili locali. Se è presente solo uno dei casi LOC o VAR non bisogna usare la sbarra.

2) Subito dopo devono essere ripresi i valori passati mettendo i parametri valore formali uguali ai corrispettivi Kn.

Conclusioni

Il metodo proposto sfrutta al massimo il BASIC già presente, in modo che sia sufficiente un programma esterno per renderlo modulare. Infatti, come visto, il metodo agisce all'entrata e all'uscita di una routine mentre per il resto il BASIC agisce normalmente.

Potendo modificare alcune routine del BASIC (come avevamo visto prima per il trattamento degli array, ma ce ne sono delle altre) il metodo potrebbe essere ancora migliorato.

Comunque, anche per questo metodo, ci sono delle velocizzazioni, come ad esempio immagazzinare nello stack, per il caso VAR, la locazione del nome sostituito in modo che in uscita la risostituzione dei nomi delle variabili avverrà direttamente per indirizzo di memoria.

Non si è preso in considerazione il caso parametro formale funzione. Dato che le funzioni sono tenute tra le variabili (rappresentano l'ultima combinazione di ASCII positivi e negativi), il passaggio di una funzione come parametro è possibile ed equivale in pratica al caso VAR.

Concludendo, abbiamo visto come la modularità sia possibile su un linguaggio non compilato.

Questo programma, come detto prima, è stato realizzato principalmente come dimostrazione pratica del metodo teorico. Comunque può essere anche utilizzato per fare vostri programmi; basterà raggruppare tutte le variabili del sistema in modo che inizino per una stessa lettera, ad esempio S. In questo modo durante i programmi si dovrà solo fare attenzione a non usare variabili inizianti per S e K, che come abbiamo visto servono per passare valori. MC

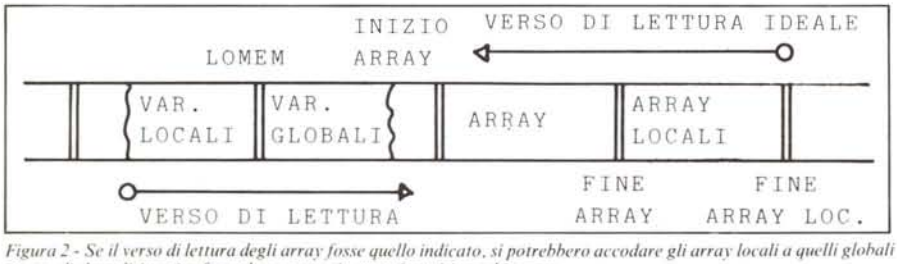

Figura 2 - Se il verso di lettura degli array fosse quello indicato, si potrebbero accodare gli array locali a quelli globali invece di doverli inserire facendo scorrere in avanti tutti i vecchi array.

QUOTAZIONI

Materiale nuovo imballato

CENTRO ASSISTENZA SPECTRUM

sumus

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	706.000
Lemon II 64K con Z-80	799.000
Lemon II 64K con Z-80 compatto, con unità a disco incorporata	1.299.000
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.	

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

Il NEGOZIO DI SUPER SUMUS!

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.