CROSS reference

in linguaggio macchina per CBM 3032

di Pierluigi Panunzi

Il programma che presentiamo in questo numero esegue una funzione particolarmente utile in molte applicazioni: il "cross reference" di un programma redatto in BASIC.

Per chi non sapesse di cosa si tratta, il cross reference non è altro che una tabella contenente, per ogni variabile utilizzata dal programma BASIC, i numeri delle linee in cui è presente tale variabile: l'utilità si apprezza quando si desidera effettuare una modifica ad un programma, per non duplicare involontariamente variabili già usate, e soprattutto per.... raccapezzarcisi un po' di più quando analizziamo un programma scritto da altri oppure da noi stessi, ma "tempo addietro", per cui non ricordiamo quasi nulla.

Come per il programma "Shell-Metzer Sort" pubblicato sullo scorso numero, anche in questo caso valgono le solite considerazioni: esisteranno senza dubbio versioni scritte in BASIC (un esempio, per l'Apple II, si trova nel n. 14 di MC), ma sono senz'altro molto più lente, e soprattutto, essendo "programmi in BASIC" a loro volta, devono essere agganciate alla fine del programma da analizzare, con tutte le scomodità del caso, dal momento che non tutti i personal computer (e tra essi il CBM 8032) sono dotati della funzione di "merge" tra programmi.

A favore del programma presentato va l'altissima velocità di esecuzione: per piccoli programmi BASIC il risultato si ha istantaneamente, mentre per i programmi più grandi in nostro possesso non si sono mai superati i tre-quattro secondi! Considerato che le linee di programma erano ben più di trecento (non le abbiamo contate....) e che le variabili erano... chi lo sa?;.... il risultato è notevole!

Il metodo usato per la creazione della "tabella dei riferimenti" è concettualmente semplice: si effettua l'analisi del testo del programma in BASIC alla ricerca di tutte le variabili, non appena se ne trova una, la si deve memorizzare da qualche parte, come pure si deve appuntare il numero della linea che ospita tale variabile: bisogna però accertarsi se tale variabile era già stata "chiamata" precedentemente oppure se nella stessa linea si hanno più richiami alla stessa variabile.

Il primo pensiero di un "programmatore" potrebbe essere:..... mi costruisco una matrice di tante righe e tante colonne, ad ogni riga associo una variabile e tutte le linee in cui compare....

Però basta pensarci un attimo per rendersi conto che non è possibile dimensionare tale matrice: infatti non è noto a priori il numero effettivo di variabili presenti in un programma (potremmo al limite farne una stima ad occhio, se ne siamo capaci), ma poi quante volte verrà chiamata una variabile? Cento volte, duecento volte? Non vale certo la pena mettersi a contare le variabili e le linee quando poi tale compito verrà eseguito dal programma.

Questo nei programmi lunghi, ovviamente....

Ecco che allora ci viene alla mente una struttura di dati, dalle caratteristiche notevoli: una lista multipla (vedi figura 1).

Perfetto: gli elementi della lista principale saranno le variabili incontrate via via nel programma, collegate da opportuni puntatori, in modo, tra l'altro, di averle sempre ordinate alfabeticamente; gli elementi delle sottoliste (una per ogni elemento della lista) saranno a questo punto i numeri di linea.

Qual è la "capacità" di una siffatta struttura? È presto detto: finché c'è spazio in memoria (la cosiddetta "lista libera") si potranno aggiungere elementi alla lista e alle sotto-liste.

Scendiamo un po' più in dettaglio: dei 32K di RAM presenti nel CBM 8032, il primo "k" è inutilizzabile in quanto riservato al sistema operativo (ma noi ne useremo alcune celle per i puntatori e per altri scopi, come vedremo più in dettaglio), mentre gli ultimi 2k sono occupati in parte dal nostro programma in assembler (che in realtà è più corto di 1k). Restano perciò 29k: una parte più o meno grossa sarà occupata dal programma in BASIC da analizzare mentre tutta la parte rimanente resta a disposizione della lista multipla.

Facendo riferimento alla figura 2, vediamo che ogni elemento della lista principale è lungo 8 byte e viceversa ogni elemento della sotto-lista ne occupa 4.

Degli 8 byte, quattro sono riservati al nome della variabile e quattro servono per i due puntatori: uno all'elemento successivo e l'altro alla sotto-lista.

I quattro byte per il nome della variabile derivano dalla massima lunghezza possibile, senza tener conto però della codifica "interna" usata dal Sistema Operativo; tutti i possibili casi di nomi di variabili sono i seguenti:

a a(aa aa(a$ a$(aa$ aa$(a% a%(aa% aa%(

dove la "(" si riferisce ad un vettore o ad una matrice.

Invece i 4 byte per gli elementi delle sotto-liste sono così divisi: una coppia per il valore, in esadecimale, del numero di linea e l'altra coppia per il puntatore all'elemento successivo della sotto-lista.

Inoltre vengono prese in considerazione le "funzioni definite nel programma" e cioè le "fnX(Y)", le quali richiedono una linea in cui vengono definite, tramite lo statement "def fnX".

Prima di proseguire, una considerazione: vengono trattati alla stessa stregua i vettori e le matrici, almeno per quanto riguarda il nome che li identifica.

Questo, lungi dall'essere un errore "premeditato", ricalca un po' la filosofia adottata dal Sistema Operativo, il quale sa distinguere, e co-

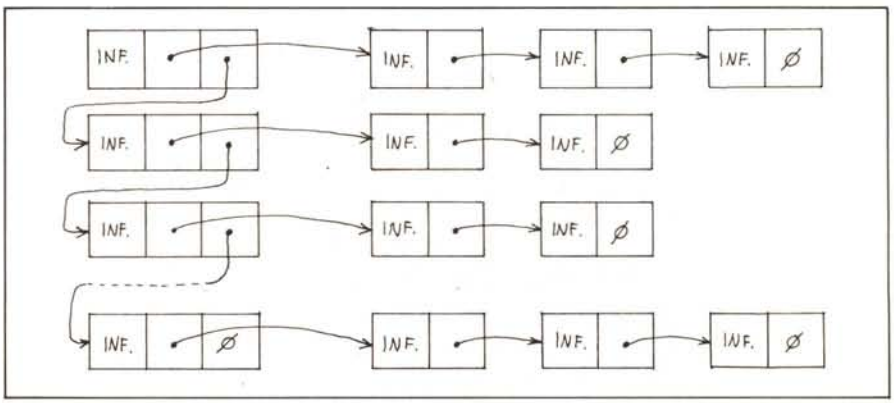

Figura 1 - Rappresentazione di una lista multipla. "INF." è la parte informazione dell'elemento; i puntatori realizzano la connessione con gli elementi successivi e con le sotto-liste.

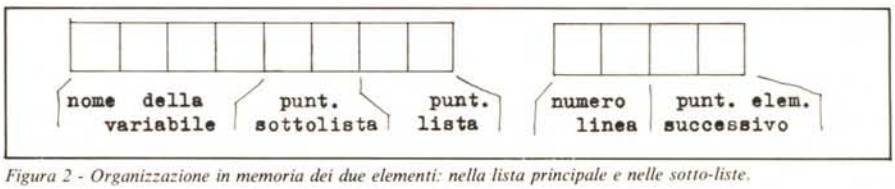

Figura 2 - Organizzazione in memoria dei due elementi: nella lista principale e nelle sotto-liste.

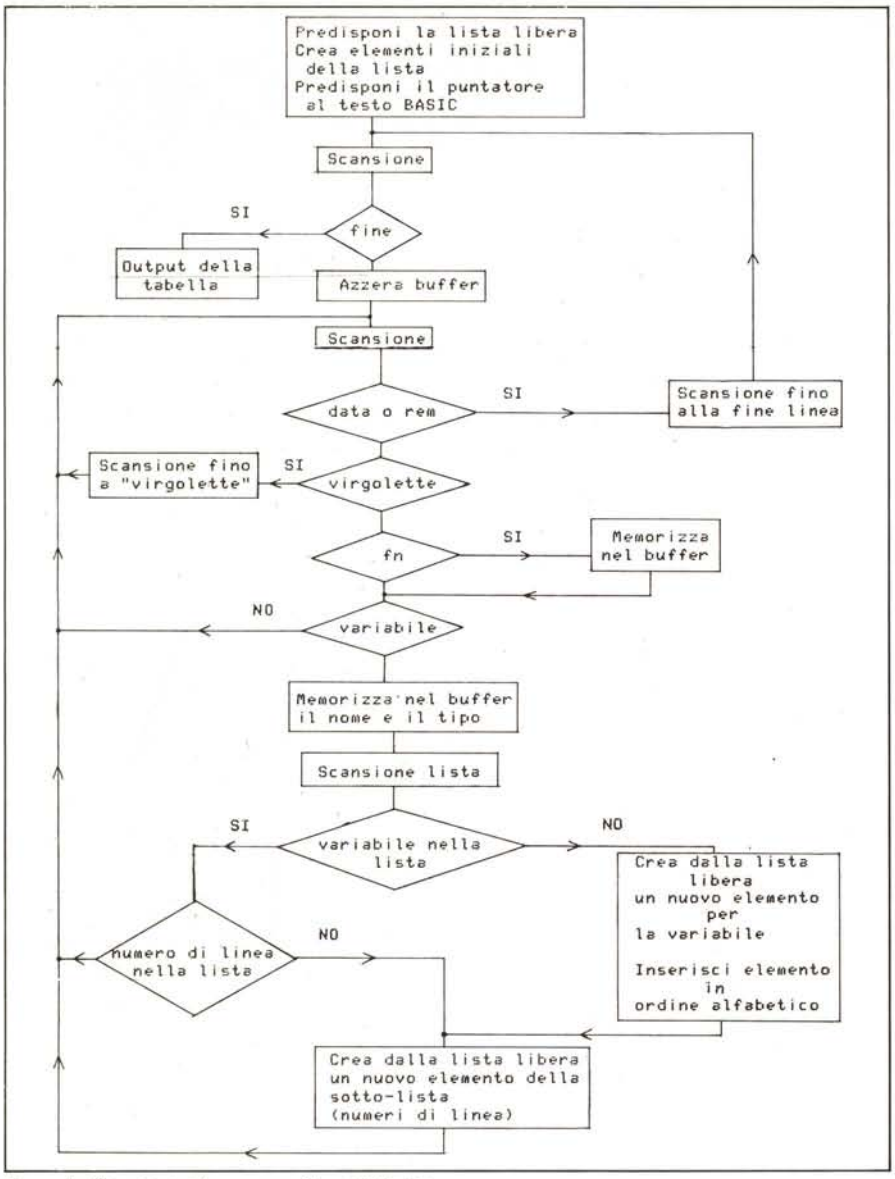

Figura 3 - Flow-chart del programma Cross Reference.

Routine di sistema operativo del CBM 8032

— Stampa del messaggio "out of memory error"
Si ottiene chiamando la routine avente indirizzo $b3cd con jmp $b3cd
Si ha il ritorno automatico al modo di comando contraddistinto dal prompt "ready".
— Stampa del carattere ASCII contenuto nell'accumulatore.
Supponendo di avere in accumulatore il carattere da stampare, si imposta
jsr $f266
Tale routine stampa il carattere nella posizione puntata dal cursore attuale ed inoltre consente l'invio di tutti i caratteri speciali di editing, quali ad esempio "home", "clr", "era begin", "rvs", "esc", ecc., nonché i controlli del cursore.
— Stampa automatica di caratteri di controllo
Le chiamate alla routine
jsr $d534, jsr $d531, jsr $d52e
permettono la stampa rispettivamente di: "RETURN" (e cioè fa saltare a linea nuova analogamente ad una "print" isolata in un programma BASIC), uno spazio e due spazi.
— Conversione esadecimale - ASCII e stampa
Dato un numero esadecimale a 16 bit, e perciò in due byte (LO e HI), si pone il byte LO nel registro x ed il byte HI nell'accumulatore. Si ottiene la stampa del corrispondente valore espresso in decimale con la chiamata
jsr $cf83
Dal momento che si tratta di una routine di stampa, valgono anche in questo caso le stesse considerazioni precedenti ed in particolare il valore decimale viene stampato a partire dalla posizione corrente del cursore.

difica in maniera differente, le variabili semplici (floating-point, intere o stringhe) dagli "array", intendendo con tale nome le matrici a qualsiasi dimensione: questi ultimi infatti presentano una parentesi subito dopo il nome.

Si badi bene però che la distinzione tra vettori e matrici a più dimensioni si ha solo in "run-time".

Nel nostro caso il programma, già alquanto "pesante", si sarebbe ulteriormente appesantito con una routine capace di analizzare se dopo la parentesi aperta sono presenti delle virgole: il numero di tali virgole più uno darebbe il numero di dimensioni, informazione questa che andrebbe aggiunta nell'elemento della lista principale. Abbiamo parlato al condizionale in quanto il tutto non è così semplice, come può sembrare a prima vista: in realtà il "contatore di virgole" dovrebbe tener conto anche di situazioni del tipo

.... a(b,a(6,d))....

cioè con indirizzamenti relativi di elementi di matrici; inoltre tale "contatore" dovrebbe pure tener conto delle variabili che mano a mano trova nella scansione e peggio è se, come in questo caso, incontra altre matrici. In definitiva tale contatore dovrebbe essere un vero e proprio analizzatore lessicale, il che è certo troppo per il nostro programma...

Comunque lettori coraggiosi potranno senz'altro affrontare questa estensione.

Il programma

Abbiamo già visto che il programma in assembler è posto negli ultimi 2k della memoria RAM: in particolare è posto a partire dall'indirizzo $7800 e termina alla locazione $7a70, peraltro con alcuni vuoti all'interno. Inoltre non è rilocabile direttamente in quanto presenta alcune chiamate a subroutine ed alcuni salti assoluti: se proprio fosse necessario spostarlo in memoria, bisogna cambiare "certosinamente" (o meglio con un buon assemblatore) tali riferimenti assoluti.

Osservando il flow-chart di figura 3, vediamo indicati i blocchi fondamentali nei quali è suddiviso l'algoritmo: alcuni di questi, e cioè quelli relativi alla gestione delle liste, contengono nascosti nel loro interno tutti quei problemi e quelle tematiche ben noti agli informatici e sui quali si sono scritte intere biblioteche. Il blocco contenente la frase "Inserisci l'elemento nella lista, in ordine alfabetico", tanto per fare un esempio, racchiude in sé il problema della scansione di una lista, del confronto di elementi e dell'inserimento di un elemento, operazione quest'ultima che richiede la presenza di una "lista libera". Niente paura, il tutto è realizzabile in maniera molto semplice.

Ma andiamo con ordine. Il primo blocco si riferisce all'inizializzazione: in particolare viene spostato l'indirizzo di "top of memory", in modo tale da proteggere il programma da "invasioni" da parte di stringhe, che come noto vengono memorizzate a ritroso a partire dall'indirizzo $7fff. In questo modo viene pure posto un limite alla zona di memoria utilizzabile dal programma come lista libera.

Quindi vengono inizializzati i puntatori principali: quello alla lista libera, quello al primo elemento, quello all'ultimo elemento (sempre della lista principale) e quello al testo BASIC, usato dalla routine di scansione.

Infine vengono creati due elementi della lista, il "primo" e l'"ultimo", contenenti, nella parte "informazione", rispettivamente i valori

OO OO OO OO ed ff ff ff ff

In questo modo, quando si effettua la scansione del testo BASIC, ogni variabile trovata verrà

```
7800 A9 78     LDA #$78
7802 85 35     STA $35
7804 A9 00     LDA #$00
7806 85 34     STA $34
7808 85 FB     STA $FB
780A A8        TAY
780B A2 0F     LDX #$0F
780D 91 2A     STA ($2A),Y
780F C8        INY
7810 CA        DEX
7811 10 FA     BPL $780D
7813 A0 08     LDY #$08
7815 8A        TXA
7816 A2 04     LDX #$04
7818 86 FC     STX $FC
781A 91 2A     STA ($2A),Y
781C C8        INY
781D CA        DEX
781E D0 FA     BNE $781A
7820 A0 06     LDY #$06
7822 A5 2B     LDA $2B
7824 85 F1     STA $F1
7826 18        CLC
7827 A5 2A     LDA $2A
7829 69 08     ADC #$08
782B 85 F0     STA $F0
782D 90 02     BCC $7831
782F E6 F1     INC $F1
7831 A5 F0     LDA $F0
7833 91 2A     STA ($2A),Y
7835 C8        INY
7836 A5 F1     LDA $F1
7838 91 2A     STA ($2A),Y
783A 18        CLC
783B A5 F0     LDA $F0
783D 69 08     ADC #$08
783F 85 F0     STA $F0
7841 90 02     BCC $7845
7843 E6 F1     INC $F1
7845 20 DD 79  JSR $79DD
7848 D0 03     BNE $784D
784A 4C 00 7A  JMP $7A00
784D 20 E0 79  JSR $79E0
7850 85 36     STA $36
7852 20 E0 79  JSR $79E0
7855 85 37     STA $37
7857 20 E0 79  JSR $79E0
785A F0 E9     BEQ $7845
785C A2 00     LDX #$00
785E 8E 7B 02  STX $027B
7861 8E 7C 02  STX $027C
7864 8E 7D 02  STX $027D
7867 C9 83     CMP #$83
7869 F0 04     BEQ $786F
786B C9 8F     CMP #$8F
786D D0 07     BNE $7876
786F 20 E0 79  JSR $79E0
7872 D0 FB     BNE $786F
7874 F0 CF     BEQ $7845
7876 C9 22     CMP #$22
7878 D0 0B     BNE $7885
787A 20 E0 79  JSR $79E0
787D F0 C6     BEQ $7845
787F C9 22     CMP #$22
7881 D0 F7     BNE $787A
7883 F0 D2     BEQ $7857
7885 C9 A5     CMP #$A5
7887 D0 07     BNE $7890
7889 8D 7A 02  STA $027A
788C E8        INX
788D 20 E0 79  JSR $79E0
7890 20 AC 79  JSR $79AC
7893 90 C2     BCC $7857
7895 9D 7A 02  STA $027A,X
7898 20 E0 79  JSR $79E0
789B 20 B8 79  JSR $79B8
789E EA        NOP
789F 90 07     BCC $78A8
78A1 E8        INX
78A2 9D 7A 02  STA $027A,X
78A5 20 E0 79  JSR $79E0
78A8 C9 24     CMP #$24
78AA F0 04     BEQ $78B0
78AC C9 25     CMP #$25
78AE D0 07     BNE $78B7
78B0 E8        INX
78B1 9D 7A 02  STA $027A,X
78B4 20 E0 79  JSR $79E0
78B7 C9 28     CMP #$28
78B9 D0 04     BNE $78BF
78BB E8        INX
78BC 9D 7A 02  STA $027A,X
78BF EA        NOP
78C0 A9 7A     LDA #$7A
78C2 85 66     STA $66
78C4 A9 02     LDA #$02
78C6 85 67     STA $67
78C8 A5 2A     LDA $2A
78CA 85 F2     STA $F2
78CC A5 2B     LDA $2B
78CE 85 F3     STA $F3
78D0 A2 04     LDX #$04
78D2 A0 00     LDY #$00
78D4 B1 66     LDA ($66),Y
78D6 D1 F2     CMP ($F2),Y
78D8 D0 3C     BNE $7916
78DA C8        INY
78DB CA        DEX
78DC D0 F6     BNE $78D4
78DE B1 F2     LDA ($F2),Y
78E0 85 68     STA $68
78E2 C8        INY
78E3 B1 F2     LDA ($F2),Y
78E5 85 69     STA $69
78E7 A0 00     LDY #$00
78E9 B1 68     LDA ($68),Y
78EB C5 36     CMP $36
78ED D0 07     BNE $78F6
78EF C8        INY
78F0 B1 68     LDA ($68),Y
78F2 C5 37     CMP $37
78F4 F0 38     BEQ $792E
78F6 A0 03     LDY #$03
78F8 B1 68     LDA ($68),Y
78FA F0 0C     BEQ $7908
78FC 48        PHA
78FD 88        DEY
78FE B1 68     LDA ($68),Y
7900 85 68     STA $68
7902 68        PLA
7903 85 69     STA $69
7905 18        CLC
7906 90 DF     BCC $78E7
7908 A5 F0     LDA $F0
790A A0 02     LDY #$02
790C 91 68     STA ($68),Y
790E C8        INY
790F A5 F1     LDA $F1
7911 91 68     STA ($68),Y
7913 4C 76 79  JMP $7976
7916 90 19     BCC $7931
7918 A5 F2     LDA $F2
791A 85 F6     STA $F6
791C A5 F3     LDA $F3
791E 85 F7     STA $F7
7920 A0 06     LDY #$06
7922 B1 F6     LDA ($F6),Y
7924 85 F2     STA $F2
7926 C8        INY
7927 B1 F6     LDA ($F6),Y
7929 85 F3     STA $F3
792B 18        CLC
792C 90 A2     BCC $78D0
792E 4C A3 79  JMP $79A3
7931 A5 F0     LDA $F0
7933 85 F4     STA $F4
7935 A5 F1     LDA $F1
7937 85 F5     STA $F5
7939 A0 06     LDY #$06
793B A5 F2     LDA $F2
793D 91 F4     STA ($F4),Y
793F C8        INY
7940 A5 F3     LDA $F3
7942 91 F4     STA ($F4),Y
7944 88        DEY
7945 A5 F4     LDA $F4
7947 91 F6     STA ($F6),Y
7949 C8        INY
794A A5 F5     LDA $F5
794C 91 F6     STA ($F6),Y
794E 18        CLC
794F A5 F0     LDA $F0
7951 69 08     ADC #$08
7953 85 F0     STA $F0
7955 90 02     BCC $7959
7957 E6 F1     INC $F1
7959 A5 F1     LDA $F1
795B C9 78     CMP #$78
795D D0 03     BNE $7962
795F 4C CD B3  JMP $B3CD
7962 A0 03     LDY #$03
7964 B1 66     LDA ($66),Y
7966 91 F4     STA ($F4),Y
7968 88        DEY
7969 10 F9     BPL $7964
796B A0 04     LDY #$04
796D A5 F0     LDA $F0
796F 91 F4     STA ($F4),Y
7971 C8        INY
7972 A5 F1     LDA $F1
7974 91 F4     STA ($F4),Y
7976 A5 F1     LDA $F1
7978 85 69     STA $69
797A A5 F0     LDA $F0
797C 85 68     STA $68
797E 18        CLC
797F 69 04     ADC #$04
7981 85 F0     STA $F0
7983 90 02     BCC $7987
7985 E6 F1     INC $F1
7987 A5 F1     LDA $F1
7989 C9 78     CMP #$78
798B D0 03     BNE $7990
798D 4C CD B3  JMP $B3CD
7990 A0 00     LDY #$00
7992 A5 36     LDA $36
7994 91 68     STA ($68),Y
7996 C8        INY
7997 A5 37     LDA $37
7999 91 68     STA ($68),Y
799B A9 00     LDA #$00
799D C8        INY
799E 91 68     STA ($68),Y
79A0 C8        INY
79A1 91 68     STA ($68),Y
79A3 20 E3 79  JSR $79E3
79A6 4C 5A 78  JMP $785A
79A9 EA        NOP
79AA EA        NOP
79AB EA        NOP
79AC C9 41     CMP #$41
79AE 90 04     BCC $79B4
79B0 C9 5B     CMP #$5B
79B2 90 02     BCC $79B6
79B4 18        CLC
79B5 60        RTS
79B6 38        SEC
79B7 60        RTS
79B8 20 AC 79  JSR $79AC
79BB B0 0A     BCS $79C7
79BD C9 30     CMP #$30
79BF 90 04     BCC $79C5
79C1 C9 3A     CMP #$3A
79C3 90 02     BCC $79C7
79C5 18        CLC
79C6 60        RTS
79C7 38        SEC
79C8 60        RTS
79C9 EA        NOP
79CA EA        NOP
79CB EA        NOP
79CC EA        NOP
79CD EA        NOP
79CE EA        NOP
79CF EA        NOP
79D0 EA        NOP
79D1 EA        NOP
79D2 EA        NOP
79D3 EA        NOP
79D4 EA        NOP
79D5 EA        NOP
79D6 EA        NOP
79D7 EA        NOP
79D8 EA        NOP
79D9 EA        NOP
79DA EA        NOP
79DB EA        NOP
79DC EA        NOP
79DD 20 E0 79  JSR $79E0
79E0 20 39 D5  JSR $D539
79E3 A0 00     LDY #$00
79E5 B1 FB     LDA ($FB),Y
79E7 60        RTS
79E8 AA        TAX
79E9 AA        TAX
79EA AA        TAX
79EB AA        TAX
79EC AA        TAX
79ED AA        TAX
79EE AA        TAX
79EF AA        TAX
79F0 AA        TAX
79F1 AA        TAX
79F2 AA        TAX
79F3 AA        TAX
79F4 AA        TAX
79F5 AA        TAX
79F6 AA        TAX
79F7 AA        TAX
79F8 AA        TAX
79F9 AA        TAX
79FA AA        TAX
79FB AA        TAX
79FC AA        TAX
79FD AA        TAX
79FE AA        TAX
79FF AA        TAX
7A00 A0 06     LDY #$06
7A02 B1 2A     LDA ($2A),Y
7A04 85 F2     STA $F2
7A06 C8        INY
7A07 B1 2A     LDA ($2A),Y
7A09 85 F3     STA $F3
7A0B 20 34 D5  JSR $D534
7A0E A9 12     LDA #$12
7A10 20 66 F2  JSR $F266
7A13 A0 00     LDY #$00
7A15 B1 F2     LDA ($F2),Y
7A17 C9 FF     CMP #$FF
7A19 D0 01     BNE $7A1C
7A1B 60        RTS
7A1C A2 04     LDX #$04
7A1E A0 00     LDY #$00
7A20 B1 F2     LDA ($F2),Y
7A22 20 66 F2  JSR $F266
7A25 C8        INY
7A26 CA        DEX
7A27 D0 F7     BNE $7A20
7A29 A9 92     LDA #$92
7A2B 20 66 F2  JSR $F266
7A2E 20 2E D5  JSR $D52E
7A31 20 2E D5  JSR $D52E
7A34 B1 F2     LDA ($F2),Y
7A36 85 F6     STA $F6
7A38 C8        INY
7A39 B1 F2     LDA ($F2),Y
7A3B 85 F7     STA $F7
7A3D C8        INY
7A3E B1 F2     LDA ($F2),Y
7A40 48        PHA
7A41 C8        INY
7A42 B1 F2     LDA ($F2),Y
7A44 85 F3     STA $F3
7A46 68        PLA
7A47 85 F2     STA $F2
7A49 A0 00     LDY #$00
7A4B B1 F6     LDA ($F6),Y
7A4D AA        TAX
7A4E C8        INY
7A4F B1 F6     LDA ($F6),Y
7A51 20 83 CF  JSR $CF83
7A54 20 31 D5  JSR $D531
7A57 A0 03     LDY #$03
7A59 B1 F6     LDA ($F6),Y
7A5B F0 AE     BEQ $7A0B
7A5D 48        PHA
7A5E 88        DEY
7A5F B1 F6     LDA ($F6),Y
7A61 85 F6     STA $F6
7A63 68        PLA
7A64 85 F7     STA $F7
7A66 18        CLC
7A67 90 E0     BCC $7A49
7A69 AA        TAX
7A6A AA        TAX
7A6B AA        TAX
7A6C AA        TAX
7A6D AA        TAX
7A6E AA        TAX
7A6F AA        TAX
7A70 AA        TAX
```

Disassemblato del programma Cross Reference.

```
VAR    7  85  125  140  147  6020
VAR2   85  125  147  6000  6010  6030
VAR3   10  11  6000  6030  9000
VAR4   1  6000
SEN    7  85  125  140  147
SEN2   24  400  1001  1010  1011  3015  3020  3025  3040  3045  3050  3500
SEN3   24  400  1320  1330  1340  1501  3505  3507  3510  3530  3535
```

Esempio dell'output del programma Cross Reference.

confrontata sempre con due elementi consecutivi della lista, ottenendo così ogni volta l'ordinamento alfabetico degli elementi.

Successivamente si ha la scansione vera e propria del testo BASIC alla ricerca di tutte le variabili presenti: se si incontrano istruzioni del tipo "data" o "rem", che possono contenere al loro interno lettere varie, erroneamente interpretabili come variabili, allora si effettua la scansione "cieca" fino alla fine della linea di programma in esame.

Analogamente se si incontrano delle virgolette (in questo caso da considerare "aperte"), deve essere trascurato tutto quello che si trova fino alle successive virgolette (da considerare ora come "chiuse"): è questo il caso di definizione di stringhe (ad es. a$ = "esempio") oppure di stampa (ad es. print "il risultato è" x).

Se invece si incontra una variabile oppure la funzione "fn" (di codice esadecimale $a5), si ha la memorizzazione del nome e dell'eventuale parentesi (nel caso della già citata "fn" e nel caso degli array) nel buffer, rappresentato da quattro locazioni di memoria a partire da $027a: in particolare se la variabile è intera o di tipo stringa, in tale buffer verrà anche correttamente memorizzato il carattere "%" o "$" al posto giusto.

Trovata dunque una variabile, si entra nella parte di programma riguardante appunto la gestione della lista multipla.

Si scandisce la lista, elemento per elemento, per cercare se la variabile era già stata memorizzata: il tutto però tenendo conto dell'ordine alfabetico.

Se la variabile già c'era, si andrà ora a scandire la sotto-lista per vedere se il numero della linea BASIC attuale (presente nelle locazioni $36,37) era già stato memorizzato.

In caso affermativo si prosegue oltre nella scansione del testo BASIC, altrimenti si "accoderà" il numero di linea corrente, senza preoccuparci di ordinarlo rispetto agli altri numeri già presenti, in quanto ... già ordinato, grazie al Sistema Operativo il quale di per sé ordina le linee di un programma BASIC.

Se invece la variabile non era già stata inserita nella lista principale, perché evidentemente non era ancora stata chiamata nel programma BASIC, il nostro programma provvederà, se ancora c'è spazio in memoria e cioè se c'è ancora lista libera, ad inserire l'elemento nuovo correggendo opportunamente il puntatore dell'elemento della lista che ora lo precederà; viceversa l'elemento "nuovo" dovrà ora essere collegato all'elemento successivo della lista.

Ancora una volta ciò che è alquanto nebuloso se spiegato a parole, risulta invece agevole e veloce nella realizzazione effettiva.

Nel caso alquanto remoto in cui non vi sia più spazio in memoria (con programmi enormi che utilizzano tantissime variabili) allora si avrà il salto ad una routine del Sistema Operativo che provvede a stampare il messaggio "out of memory error" per poi ritornare al "Command mode" segnalato dal prompt "ready".

Parlando di routine di Sistema, rimandiamo alla finestra di pagina 69 nella quale sono riproposte tutte quelle usate dal nostro programma.

Terminiamo questa descrizione segnalando l'utilizzazione di parecchie locazioni in pagina zero, per la precisione quelle tra $f0 e $f7, tra $66 e $69, il buffer già citato, nonché le locazioni $2a,2b (il puntatore all'"inizio della zona riservata alle variabili", nel nostro caso trasformato in "inizio della lista libera") e $36,37 usate dal S.O. per memorizzare il numero di linea corrente del programma BASIC. Inutile dire che queste variabili in pagina zero sono state scelte a ragion veduta e non certo a caso, in funzione del loro uso o meno da parte del S.O., almeno per le routine effettivamente operanti nel nostro caso.

L'uso

Dopo aver introdotto il programma ed averlo memorizzato ad esempio nel file denominato "crossreference", seguendo la prassi nota a tutti i possessori del CBM 8032, preferibilmente spegniamo e riaccendiamo il computer.

Dopo aver udito il consueto "cinguettio di approvazione" (= tutto è O.K.!), possiamo caricare il programma con

dload "crossreference"

fatto ciò impostiamo "new": no: non siamo ammattiti!. Il "new", ricordiamo, opera solo sui programmi redatti in BASIC, cancellandoli: nel nostro caso si ottiene una "furba" correzione di alcuni importanti puntatori, che ci permette di poter caricare il programma BASIC con il consueto

dload "programma BASIC"

Infatti se ci fossimo dimenticati del "new" di cui sopra, il computer ci avrebbe segnalato un altrimenti inspiegabile "out of memory error".

A questo punto impostiamo

.sys30720

il quale fa partire l'elaborazione: in men che non si dica si avrà il risultato, rappresentato dal nome della variabile in inverso, seguito dai numeri delle linee contenenti un riferimento a tale variabile.

Nel caso delle funzioni del tipo "fnX" (ad esempio con "fnxy(a)") si avrà, prima del nome "xy(", un apparente "blank", in inverso: in realtà si tratta del carattere grafico corrispondente al valore esadecimale $a5 (l'abbiamo già incontrato prima) e cioè una riga verticale posta tutta a sinistra. In inverso apparirà come un rettangolino bianco, dove però manca la prima colonna di puntini a sinistra.

Se vogliamo rivedere la tabella dall'inizio, perché ci è sfuggita una delle prime variabili, tanto vale far rielaborare il programma (dal momento che è velocissimo) con il comando "sys30720".

Ricordiamo a questo punto che un output lungo sullo schermo causa uno scorrimento verso l'alto di quanto scritto in precedenza: tale scorrimento avviene alla massima velocità consentita, dipendente dal numero di caratteri effettivamente stampati riga per riga. Se vogliamo rallentare questa corsa dobbiamo tener premuto il tasto "←", posto in alto a sinistra sulla tastiera.

Se vogliamo fermare l'output dobbiamo invece premere il tasto ":"; infine per far riprendere l'output basta "sfiorare" il tasto "9", proprio due tasti a sinistra di ":" in tal modo possiamo alternativamente far partire o arrestare l'output secondo i nostri desideri. MC

```
7800 A9 78 85 35 A9 00 85 34
7808 85 FB A8 A2 0F 91 2A C8
7810 CA 10 FA A0 08 8A A2 04
7818 86 FC 91 2A C8 CA D0 FA
7820 A0 06 A5 2B 85 F1 18 A5
7828 2A 69 08 85 F0 90 02 E6
7830 F1 A5 F0 91 2A C8 A5 F1
7838 91 2A 18 A5 F0 69 08 85
7840 F0 90 02 E6 F1 20 DD 79
7848 D0 03 4C 00 7A 20 E0 79
7850 85 36 20 E0 79 85 37 20
7858 E0 79 F0 E9 A2 00 8E 7B
7860 02 8E 7C 02 8E 7D 02 C9
7868 83 F0 04 C9 8F D0 07 20
7870 E0 79 D0 FB F0 CF C9 22
7878 D0 0B 20 E0 79 F0 C6 C9
7880 22 D0 F7 F0 D2 C9 A5 D0
7888 07 8D 7A 02 E8 20 E0 79
7890 20 AC 79 90 C2 9D 7A 02
7898 20 E0 79 20 B8 79 EA 90
78A0 07 E8 9D 7A 02 20 E0 79
78A8 C9 24 F0 04 C9 25 D0 07
78B0 E8 9D 7A 02 20 E0 79 C9
78B8 28 D0 04 E8 9D 7A 02 EA
78C0 A9 7A 85 66 A9 02 85 67
78C8 A5 2A 85 F2 A5 2B 85 F3
78D0 A2 04 A0 00 B1 66 D1 F2
78D8 D0 3C C8 CA D0 F6 B1 F2
78E0 85 68 C8 B1 F2 85 69 A0
78E8 00 B1 68 C5 36 D0 07 C8
78F0 B1 68 C5 37 F0 38 A0 03
78F8 B1 68 F0 0C 48 88 B1 68
7900 85 68 68 85 69 18 90 DF
7908 A5 F0 A0 02 91 68 C8 A5
7910 F1 91 68 4C 76 79 90 19
7918 A5 F2 85 F6 A5 F3 85 F7
7920 A0 06 B1 F6 85 F2 C8 B1
7928 F6 85 F3 18 90 A2 4C A3
7930 79 A5 F0 85 F4 A5 F1 85
7938 F5 A0 06 A5 F2 91 F4 C8
7940 A5 F3 91 F4 88 A5 F4 91
7948 F6 C8 A5 F5 91 F6 18 A5
7950 F0 69 08 85 F0 90 02 E6
7958 F1 A5 F1 C9 78 D0 03 4C
7960 CD B3 A0 03 B1 66 91 F4
7968 88 10 F9 A0 04 A5 F0 91
7970 F4 C8 A5 F1 91 F4 A5 F1
7978 85 69 A5 F0 85 68 18 69
7980 04 85 F0 90 02 E6 F1 A5
7988 F1 C9 78 D0 03 4C CD B3
7990 A0 00 A5 36 91 68 C8 A5
7998 37 91 68 A9 00 C8 91 68
79A0 C8 91 68 20 E3 79 4C 5A
79A8 78 EA EA EA C9 41 90 04
79B0 C9 5B 90 02 18 60 38 60
79B8 20 AC 79 B0 0A C9 30 90
79C0 04 C9 3A 90 02 18 60 38
79C8 60 EA EA EA EA EA EA EA
79D0 EA EA EA EA EA EA EA EA
79D8 EA EA EA EA EA 20 E0 79
79E0 20 39 D5 A0 00 B1 FB 60
79E8 AA AA AA AA AA AA AA AA
79F0 AA AA AA AA AA AA AA AA
79F8 AA AA AA AA AA AA AA AA
7A00 A0 06 B1 2A 85 F2 C8 B1
7A08 2A 85 F3 20 34 D5 A9 12
7A10 20 66 F2 A0 00 B1 F2 C9
7A18 FF D0 01 60 A2 04 A0 00
7A20 B1 F2 20 66 F2 C8 CA D0
7A28 F7 A9 92 20 66 F2 20 2E
7A30 D5 20 2E D5 B1 F2 85 F6
7A38 C8 B1 F2 85 F7 C8 B1 F2
7A40 48 C8 B1 F2 85 F3 68 85
7A48 F2 A0 00 B1 F6 AA C8 B1
7A50 F6 20 83 CF 20 31 D5 A0
7A58 03 B1 F6 F0 AE 48 88 B1
7A60 F6 85 F6 68 85 F7 18 90
7A68 E0 AA AA AA AA AA AA AA
```

Codice oggetto del programma Cross Reference.