Aggiungiamo un'istruzione RESTORE numero di riga

Il Basic dell'Apple, l'Applesoft, pur essendo molto esteso e potente, manca purtroppo di alcune istruzioni a volte utili. È questo il caso del RESTORE parziale. La restore è una di quelle istruzioni che fanno parte integrante di un gruppo di altre, come la ELSE per la IF o la USING per la PRINT.

Infatti queste istruzioni non servono da sole ma usate a dovere permettono di potenziare notevolmente le capacità, di un dato linguaggio. La RESTORE fa parte del gruppo READ ... DATA, consente di riinizializzare la lista dei dati dopo la lettura dell'ultimo.

```
302_   A9 01        LDA   #01
304_   85 06        STA   06
306_   A9 08        LDA   #08
308_   85 07        STA   07
30A_   A0 00        LDY   00
30C_   B1 06        LDA   (06),Y
30E_   85 08        STA   08
310_   C8           INY
311_   B1 06        LDA   (06),Y
313_   85 09        STA   09
315_   C8           INY
316_   B1 06        LDA   (06),Y
318_   CD 00 03     CMP   300
31B_   D0 13        BNE   330
31D_   C8           INY
31E_   B1 06        LDA   (06),Y
320_   CD 01 03     CMP   301
323_   D0 0B        BNE   330
325_   C6 06        DEC   06
327_   A5 06        LDA   06
329_   85 7D        STA   7D
32B_   A5 07        LDA   07
32D_   85 7E        STA   7E
32F_   60           RTS
330_   A5 08        LDA   08
332_   85 06        STA   06
334_   A5 09        LDA   09
336_   85 07        STA   07
338_   4C 0A 03     JMP   30A
```

Figura 1 - List in assembler della routine RESTORE nn.

```
10 FOR I = 770 TO 826
20 READ J: POKE I,J: NEXT
30 DATA  169,1,133,6,169,8,133,7,
     160,0,177,6,133,8,200,177,6,
     133,9,200,177,6,205,0,3,208,
     19,200,177,6,205,1,3,208,11,
     198,6,165,6,133,125,165,7,13
     3,126,96,165,8,133,6,165,9,1
     33,7,76,10,3
```

Figura 2 - Programma in Basic per introdurre il list di figura 1.

Come i più sapranno, la READ permette di leggere un elemento di una lista di costanti che si trova internamente al programma.

Per esempio un programma Calendario conterrà la lista dei mesi quella dei giorni della settimana e, perché no, la lista dei Santi (sempre comoda quando un programma non vuol girare). Queste liste sono contenute in una o più istruzioni DATA. All'inizio del RUN, un puntatore del BASIC, punta il primo elemento di queste liste; per l'Applesoft non fa differenza il fatto che un elemento appartenga a una o all'altra DATA. Per lui i dati sono tutti uguali e si susseguono solo per l'ordine naturale con cui sono stati inseriti nel programma. Per cui, se gli elementi della prima DATA si esauriscono, la READ successiva leggerà quelli, se ci sono, della DATA che segue, altrimenti darà un OUT OF DATA ERROR.

Quando a noi interessa rileggere la lista, un comando RESTORE riporta il puntatore all'inizio del programma da dove ripartirà in cerca della prima DATA che trova. Tornando all'esempio del calendario, sarebbe possibile determinare quale sia il sesto mese leggendo sei volte la lista dei mesi e mettendo quello che si legge sempre sulla medesima stringa: dopo sei letture il contenuto della stringa sarà ovviamente il sesto elemento. Per usare questo sistema, occorre però riinizializzare ogni volta la lista, o la prossima lettura sarà falsata. Ma se vogliamo usare lo stesso metodo anche per i giorni dobbiamo poter puntare, di volta in volta, al primo elemento dei mesi e dei giorni e dei Santi o quel che sia.

Il sistema c'è ed è anche usato da altri personal, come il PET nei quali esiste l'istruzione RESTORE nn, dove nn è il numero di riga relativo alla DATA da cui vogliamo sia letto il prossimo elemento. Quindi RESTORE 2000 posizionerà il puntatore della READ sulla istruzione il cui numero di riga è 2000.

Si potrebbe, una volta finita la stesura del programma, segnarsi le posizioni di inizio delle DATA ma così facendo, ogni modifica del programma richiederebbe una nuova ricerca degli indirizzi.

La routine in linguaggio macchina che presentiamo, cerca invece, a partire dall'inizio del programma, un certo numero di riga e lì posiziona il puntatore della READ. È sufficiente mettere in una specifica locazione il numero di riga, e chiamare la routine, perché il prossimo dato letto sia il primo del DATA con quel numero di riga.

Per capire come funziona questo programma bisogna vedere prima come l'Applesoft memorizza le righe dei programmi in Basic.

La zona di memoria destinata ai programmi parte dalla locazione Hex 800 e prosegue verso la parte alta della RAM.

Un puntatore in pagina-zero ricorda la locazione iniziale, e un altro (LOMEM) punta la fine del programma.

Quando noi scriviamo la prima istruzione (per es. 10 HOME) viene creata, a partire dalla locazione 801, una stringa che contiene:

- nelle prime due posizioni l'indirizzo assoluto della prima locazione successi-

```
801- 12 08 0A 00 81 49 4B D0 31 32 C1 33 35 30 3A 82 00
     (punt.) 10   FOR I K  =  1  2 TO  3  5  0 : NEXT (fine)
      0812

812- 23 08 14 00 BA 22 41 50 50 4C 45 20 5D 5B 22 2C 00
     (punt.) 20 PRINT " A  P  P  L  E     I  I  "  , (fine)
      0823

823- 00 00
     (chiusura)
```

Figura 3 - Esempio di memorizzazione di righe Applesoft

```
100 POKE 769,N / 256: POKE 768,N - PEEK (769) * 256
110 CALL 770
```

Figura 4 - Uso del RESTORE. N è il numero di riga da resettare.

va; ovvero della prima posizione della stringa seguente.
- nei due byte successivi è scritto il numero di riga, che essendo compreso tra zero e 63999, occupa appunto due byte.

Sia il puntatore alla stringa successiva che il numero di riga sono scritti, secondo l'uso dei microprocessori, con la parte alta dopo la parte bassa: sicché il numero 10 (Hex ØØØA) si ritroverà come ØA ØØ.

Dopo questi quattro byte viene scritta l'istruzione vera e propria. Viene usato il normale codice ASCII per le variabili e per i numeri, mentre le parole-chiave sono sostituite dal corrispondente codice. Il codice delle parole riservate è riportato nella tabella sia in decimale che in esadecimale.

Più istruzioni su una stessa riga sono separate dai ":", naturalmente in ASCII; mentre uno zero (Ø) segna la fine della riga (vedi fig. 3).

Per spostarsi da un numero di riga ad un altro si può quindi usare il puntatore alla istruzione successiva, senza dover scorrere tutta la riga. È appunto questo che il restore fa: si posiziona all'inizio del programma e, mediante i puntatori, scorre tutti i numeri di riga finché non trova quello uguale al numero che si trova in 300, 301. A questo punto trascrive nel registro della DATA (loc. 7D, 7E) la locazione iniziale della riga.

Allora, nel momento in cui dobbiamo RESTORare una certa riga, non dobbiamo far altro che scriverne il numero (tradotto in esadecimale) nelle locazioni dec. 768 e 769, e chiamare la routine 770. La riga di figura 4 effettua direttamente la conversione del numero N, lo deposita nelle apposite locazioni e chiama la routine di RESTORE.

Valter Di Dio

APPLE-minus: un solo comando per le minuscole

Chi ha un Apple e possiede la Eprom Apple-minus di MCmicrocomputer può usare la routine in linguaggio macchina presentata nel n. 4, da includere nell'hello del dischetto, per accedere tramite lo shift al set minuscolo. Vogliamo tuttavia presentarvi un interessante comando "alternativo" che, da solo, permette di avere le minuscole sia sul video sia sulla stampante. Il funzionamento è diverso e, ovviamente, più limitato rispetto alla routine di cui sopra. Si comporta esattamente come i noti INVERSE e FLASH: infatti occorre settarlo per far sì che tutti i caratteri in uscita, da quel momento in poi, sul video o su stampante vengano convertiti in minuscolo; un comando NORMAL ripristinerà, poi, il normale funzionamento. Il comando magico per ottenere tutto ciò è semplicemente un POKE. Infatti in pagina zero esiste una locazione che l'Applesoft usa per convertire i caratteri in inverso e in flash; se noi ora poniamo in questa locazione un particolare valore questo verrà ogni volta OR-ato col carattere in uscita.

La locazione miracolosa è la 243. Quale sarà il valore da metterci dentro? Provate a indovinare; la soluzione è qui sotto capovolta ...

Soluzione: il valore misterioso è 32, dato che per passare dai caratteri maiuscoli a quelli minuscoli basta sommare al codice maiuscolo appunto 32 (decimale). Il comando completo è quindi: POKE 243,32. Per ripristinare le maiuscole, NORMAL.

		128 Ø8O	144 Ø9O	160 ØAO	176 ØBO	192 ØCO	208 ØDO	224 ØEO
O	ØO	END	HGR2	COLOR=	GOSUB	TAB(	=	TAN
1	Ø1	FOR	HGR	POP	PETURN	TO	<	ATN
2	Ø2	NEXT	HCOLOR=	VTAB	REM	FN	SGN	PEEK
3	Ø3	DATA	HPLOT	HIMEM:	STOP	SPC(	INT	LEN
4	Ø4	INPUT	DRAW	LOMEM:	ON	THEN	ABS	STRØ
5	Ø5	DEL	XDRAW	ONERR	WAIT	AT	USR	VAL
6	Ø6	DIM	HTAB	RESUME	LOAD	NOT	FRE	ASC
7	Ø7	READ	HOME	RECALL	SAVE	STEP	SCRN(	CHRØ
8	Ø8	GR	ROT=	STORE	DEF	+	PDL	LEFTØ
9	Ø9	TEXT	SCALE=	SPEED=	POKE	-	POS	RIGHTØ
1O	ØA	PR#	SHLOAD	LET	PRINT	*	SQR	MIDØ
11	ØB	IN#	TRACE	GOTO	CONT	/	RND	
12	ØC	CALL	NOTRACE	RUN	LIST	^	LOG	
13	ØD	PLOT	NORMAL	IF	CLEAR	AND	EXP	
14	ØE	HLIN	INVERSE	RESTORE	GET	OR	COS	
15	ØF	VLIN	FLASH	&	NEW	>	SIN	

Tabella dei codici decimali ed esadecimali delle istruzioni Applesoft