Input senza errori

di Aurelio Mascheroni — Legnano

Spesso, nei programmi, occorre tenere sotto controllo le risposte dell'operatore per evitare che il sistema operativo o lo stesso programma vadano in errore.

Di solito si usa effettuare questo controllo dopo che il programma ha accettato già l'input, ma a volte è indispensabile farlo prima; ovvero carattere per carattere mentre questi vengono battuti.

Lasciamo ora la parola all'Autore.

In figura 1 trovate il listato di un programmino dimostrativo che impiega due subroutine che permettono: una di controllare la validità dei dati numerici (righe 1250-1440), l'altra quella dei dati alfanumerici (righe 1000-1240).

Le subroutine risolvono alcuni problemi di controllo della correttezza dell'input. In particolare, la prima (numerica) accetta solo le dieci cifre, il punto decimale e il segno "-" (solo come primo carattere digitato). Simula inoltre l'uso della freccia a sinistra come segno di cancellazione e usa il tasto di <RETURN> come nel normale INPUT dell'Applesoft. L'uso di questi tasti è inibito se non è stato introdotto ancora nessun carattere.

L'eventuale impiego di un flag di segno e di uno per gli interi (FS% ed FI% rispettivamente), nonché del massimo numero di caratteri accettabili (MC), consente un più completo controllo sulla correttezza dell'input.

La seconda routine, leggermente più complessa, accetta solo caratteri stampabili più lo spazio, ma non come primo carattere, ed elide automaticamente eventuali spazi finali. Permette un uso identico al precedente dei tasti ← e RETURN. Inoltre l'uso di un eventuale flag di file (FF%) inibisce la digitazione dei caratteri non leciti in un nome di file: per esempio la virgola o una cifra come primo carattere; e limita automaticamente a trenta il numero massimo di caratteri. La routine stampa, inoltre, una parentesi quadra per delimitare anche visivamente la massima lunghezza della stringa. La massima lunghezza per un input numerico è, da default, di undici caratteri mentre per quello alfanumerico è di 255.

```
10  REM  PROGRAMMA DI ESEMPIO
20  HOME : INVERSE : PRINT "INTRO
    DUCI UN NUMERO INTERO E POSI
    TIVO   MINORE DI 10000": NORMAL
    : PRINT
30 FS% = 1:FI% = 1:MC = 4: GOSUB
    1250:NR =  VAL (IN$): PRINT
    : PRINT NR
40  PRINT : INVERSE : PRINT "BATT
    I GENERALITA' E INDIRIZZO CO
    MPLETI  (MAX. 60 CARATTERI):
    ": NORMAL : PRINT
50 MC = 60: GOSUB 1000: PRINT : PRINT
    IA$
60  PRINT : INVERSE : PRINT "ASSE
    GNA UN NOME AD UN FILE:": NORMAL
    : PRINT
70 MC = 30:FF% = 1: GOSUB 1000: PRINT
    : PRINT "LOAD "IA$: PRINT
80  END
90 :
100 :
1000  REM  **********************
      CONTROLLO SINGOLI CARATTERI
       PER INPUT ALFANUMERICO
1010 :
1020  GOSUB 1440: IF MC = 0 AND F
    F% = 0 THEN MC = 255: GOTO 1
    050
1030  IF FF% THEN MC = 30
1040  GOSUB 1220
1050 IA$ = ""
1060  GET A$:CC =  ASC (A$): IF  LEN
    (IA$) = MC AND CC <  > 8 AND
    CC <  > 13 THEN  CALL  - 198
    : GOTO 1060
1070  IF F% = 0 THEN 1090
1080  IF CC = 8 THEN  VTAB RX + 1
    : HTAB CX + 1:CX =  PEEK (36
    ) - 1:RX =  PEEK (37): GOTO
    1150
1090  IF F% = 0 AND (CC = 32 OR C
    C = 13) THEN 1190
1100  IF FF% AND F% = 0 AND (CC <
    58 AND CC > 47) THEN 1190
1110  IF FF% AND CC = 44 THEN 119
    0
1120  IF F% AND CC = 13 THEN 1200

1130  IF CC < 32 THEN 1190
1140 CX =  PEEK (36):RX =  PEEK (
    37): PRINT A$;:F% = 1:IA$ =
    IA$ + A$: GOTO 1060
1150  IF RX = RO AND CX < CO THEN
    CX = CO:F% = 0: GOTO 1170
1160  IF CX =  - 1 THEN RX = RX -
    1:CX = 39
1170 LI =  LEN (IA$): IF LI = 1 THEN
    1050
1180 IA$ =  LEFT$ (IA$,LI - 1): GOTO
    1060
1190  CALL  - 198: GOTO 1060
1200  IF  RIGHT$ (IA$,1) =  CHR$
    (32) THEN IA$ =  LEFT$ (IA$,
    LI - 1): GOTO 1200
1210  HTAB CX + 2: VTAB RX + 1: CALL
     - 958:FF% = 0:MC = 0: RETURN

1220 PL = 40 - CO: IF MC <  = PL THEN
    RI = 0:NC = MC: GOTO 1240
1230 M = MC - PL:RT = M / 40:RI =
     INT (RT):NC = (RT - RI) * 4
    0:RI = RI + 1
1240  VTAB RO + RI + 1: HTAB CO +
    NC + 1: PRINT  CHR$ (93): HTAB
    CO + 1: VTAB RO + 1: RETURN

1250  REM  **********************
      CONTROLLO SINGOLI CARATTERI
       PER INPUT NUMERICO
1260 :
1270  GOSUB 1440: IF MC = 0 THEN
    MC = 11
1280 IN$ = ""
1290  GET A$:CC =  ASC (A$): IF  LEN
    (IN$) = MC AND CC <  > 8 AND
    CC <  > 13 THEN  CALL  - 198
    : GOTO 1290
1300  IF F% = 0 THEN 1320
1310  IF CC = 8 THEN  VTAB RX + 1
    : HTAB CX + 1:CX =  PEEK (36
    ) - 1:RX =  PEEK (37): GOTO
    1390
1320  IF F% = 0 AND (CC = 32 OR C
    C = 13) THEN 1420
1330  IF F% AND CC = 13 THEN 1430

1340  IF CC = 45 AND F% THEN 1420

1350  IF CC = 47 OR CC < 45 OR CC
    > 57 THEN 1420
1360  IF FI% AND CC = 46 THEN 142
    0
1370  IF FS% AND CC = 45 THEN 142
    0
1380 CX =  PEEK (36):RX =  PEEK (
    37): PRINT A$;:F% = 1:IN$ =
    IN$ + A$: GOTO 1290
1390  IF CX < CO THEN CX = CO:F% =
    0
1400 LI =  LEN (IN$): IF LI = 1 THEN
    1280
1410 IN$ =  LEFT$ (IN$,LI - 1): GOTO
    1290
1420  CALL  - 198: GOTO 1290
1430  HTAB CX + 2: VTAB RX + 1: CALL
     - 868:MC = 0:FI% = 0:FS% =
    0: RETURN
1440 CO =  PEEK (36):RO =  PEEK (
    37):F% = 0: RETURN
1500  REM  **********************

1510  REM   DI AURELIO MASCHERONI

1520  REM  **********************
```

Figura 1

CO	CC	CX
LI	M	MC
NC	NR	PL
RO	RI	RT
RX	F%	FF%
FI%	FS%	A$
IA$	IN$	

Figura 2 - Lista delle variabili usate dalla routine di controllo della validità di un input.

Commenti

La routine funziona bene anche se naturalmente si nota una certa lentezza nell'accettazione dei caratteri. A causa dell'elevato numero di variabili usate abbiamo aggiunto la lista di figura 2, ma considerate che comunque si possono anche riusare in altre parti del programma senza particolari problemi.

Un grave bug è invece il fatto che non bisogna assolutamente far girare la routine di input alfanumerica se il cursore ha già raggiunto la ventiduesima riga, in tal caso si interrompe il programma con un ILLEGALL QUANTITY ERROR.

Un DOS amico

Se volete evitare di cancellare un programma in basic che si trova sul disco solo perché vi eravate dimenticati che quel no-

me era già stato usato, senza ricorrere al pesante uso della lock/unlock, il modo c'è. Basta una piccola modifica al vostro DOS.

Il DOS, Disk Operative System, è il programma che gestisce tutte le operazioni di scrittura e di lettura sul dischetto. Per nostra fortuna il DOS non si trova nelle ROM di sistema (come il Basic o il Monitor) ma si autocarica in memoria RAM all'accensione della macchina. La zona "preferita" dal DOS è la parte alta della memoria, infatti in un Apple con 48K di RAM il DOS va ad occupare la memoria che va da $9600 a $BFFF.

A questo punto ci sono due modi per modificare il DOS. Si può attendere che si sia già caricato in memoria e poi lanciare un programma che lo modifichi: questo naturalmente tutte le volte che la macchina viene riaccesa. Oppure dopo aver fatto la modifica si può inizializzare un dischetto nuovo con il nostro DOS modificato.

Vediamo ora come funziona il programma che ci avverte dell'esistenza di un file con lo stesso nome di quello da salvare. Il DOS quando esegue una SAVE cerca per prima cosa se esiste già un file con lo stesso nome, se non lo trova ordina al FILE MANAGER (un sottoprogramma del DOS) di aprire un file con quel nome, altrimenti, trovato sulla directory l'indirizzo del file già esistente lo passa al FILE MANAGER. Per trovare se il nome esiste scandisce uno per uno tutti i file esistenti sul catalogo. La nostra routine intercetta il passaggio dei dati al FILE MANAGER sfruttando l'uscita file-trovato. A questo punto però si è presentato un piccolo problema: dopo ogni SAVE il DOS esegue automaticamente una VERIFY e la nostra routine si ostinava a dirci di nuovo che il file esisteva già. Quindi per prima cosa occorre controllare se l'operazione richiesta al F.M. è una OPEN o una VERIFY; dato che il codice operativo si trova nella locazione $AA63 basta controllare se questa contiene 1 (codice della OPEN). Anche così la nostra routine continuava ad avvertirci dell'esistenza del file con lo stesso nome anche su tutte le altre operazioni che coinvolgevano una OPEN, per esempio con la LOAD.

Ci sono due modi per sapere se la OPEN è stata chiamata da una SAVE, il primo è di cercarsi sullo STACK l'indirizzo di ritorno, il secondo è di usare una locazione in cui leggere se la SAVE è attiva. Dal momento che questa locazione non esiste, ce la siamo costruita.

Nelle prime righe della SAVE viene controllato il Flag $D6 che se è maggiore di $7F impedisce all'Applesoft di riconoscere i comandi e lo forza ad eseguire il solo RUN qualsiasi cosa si batta sulla tastiera. Questo però non vale per i comandi DOS che restano tutti attivi tranne, appunto, la SAVE che viene inibita: il tentativo di salvare un programma così protetto provoca l'errore di PROGRAM TOO LARGE. Tutto questo controllo occupava sette byte da $A39C a $A3A2; visto che nessuno ormai protegge i programmi con metodi così banali, abbiamo usato quelle sette locazioni per settare in $D7 un Flag di SAVE. La locazione $D7 è una normalmente libera ma si trova in mezzo a quelle destinate al DOS.

A questo punto il più è fatto. Sappiamo riconoscere una OPEN chiamata dalla SAVE e sappiamo che se è stata chiamata la nostra routine è perché il nome usato esiste già. Ci basta ora stampare un messaggio di avvertimento e attendere risposta dall'operatore. Purtroppo tutto ciò impegna memoria, e, se vogliamo che sia caricato insieme al DOS, deve trovarsi per forza "dentro" al DOS. Ci sono sparsi per il DOS vari byte liberi ma molto sparpagliati, le uniche zone possibili si sono rivelate quelle da $BCDF e $BCFF e da $BA69 a $BA95. In tutto 78 byte! La prima routine che funzionava correttamente era lunga più di 128 byte, ma a furia di spremere l'abbiamo portata a 77; sì da permetterci il lusso di inserire un CTRL G nel testo.

```
50  TEXT : HOME : PRINT
60  PRINT "SE HAI LA EPROM DELLE MINUSCOLE"
70  PRINT "CANCELLA LA RIGA 445: ALTRIMENTI"
80  PRINT "CANCELLA LA RIGA 440. POI RUN 100
    "
90  END
100 HOME : PRINT "  PROGRAMMA PER MODIFICAR
    E UN DOS"
110 PRINT "SLAVE 48K, IN MODO DA AVVERTIRE
    SE"
120 PRINT "SI TENTA DI SALVARE UN PROGRAMMA
    "
130 PRINT "CON UN NOME CHE GIA' ESISTE."
140 PRINT : PRINT "INSERISCI IL DISCO DA MO
    DIFICARE"
150 PRINT " E PREMI LO SPAZIO"
160 GET T$: IF T$ <  > " " THEN 160
200 READ A1,A2
210 FOR I = A1 TO A2
220 READ A: POKE I,A
230 NEXT
240 READ A: IF A < 0 THEN 200
250 POKE 45574,32: POKE 45575,223: POKE 455
    76,188
300 D$ =  CHR$ (4)
310 HOME : PRINT
320 PRINT "NOME DEL FILE DI HELLO: HELLO"
330 VTAB 2: HTAB 24: INPUT "";A$
340 PRINT D$"INIT ";A$
400 DATA  48351,48383
410 DATA  173,99,170,201,1,208,22,166,215,2
    08,18,133,215,32,165,186,32,12,253,162,
    255,201,206,240,7,201,211,208,243,174,1
    56,179,96
420 DATA  -1,47721,47765
430 DATA  32,142,253,160,0,177,66,32,237,25
    3,200,192,30,208,246,162,18,189,131,186
    ,32,237,253,202,208,247,96
440 DATA 191,239,228,229,227,239,242,240,16
    0,135,172,229,244,243,233,243,197,141
445 DATA  191,207,196,197,195,207,210,208,1
    60,135,172,197,212,211,201,211,197,141
450 DATA  -1,41884,41890
460 DATA  169,0,133,215,234,234,234
470 DATA  0
```

Figura 3 - Listato Basic del programma che modifica il DOS e inizializza un nuovo dischetto.

```
A39C-   A9 00       LDA   #$00
A39E-   85 D7       STA   $D7
A3A0-   EA          NOP
A3A1-   EA          NOP
A3A2-   EA          NOP


B206-   20 DF BC    JSR   $BCDF


BCDF-   AD 63 AA    LDA   $AA63
BCE2-   C9 01       CMP   #$01
BCE4-   D0 16       BNE   $BCFC
BCE6-   A6 D7       LDX   $D7
BCE8-   D0 12       BNE   $BCFC
BCEA-   85 D7       STA   $D7
BCEC-   20 69 BA    JSR   $BA69
BCEF-   20 0C FD    JSR   $FD0C
BCF2-   A2 FF       LDX   #$FF
BCF4-   C9 CE       CMP   #$CE
BCF6-   F0 07       BEQ   $BCFF
BCF8-   C9 D3       CMP   #$D3
BCFA-   D0 F3       BNE   $BCEF
BCFC-   AE 9C B3    LDX   $B39C
BCFF-   60          RTS


BA69-   20 8E FD    JSR   $FD8E
BA6C-   A0 00       LDY   #$00
BA6E-   B1 42       LDA   ($42),Y
BA70-   20 ED FD    JSR   $FDED
BA73-   C8          INY
BA74-   C0 1E       CPY   #$1E
BA76-   D0 F6       BNE   $BA6E
BA78-   A2 12       LDX   #$12
BA7A-   BD 83 BA    LDA   $BA83,X
BA7D-   20 ED FD    JSR   $FDED
BA80-   CA          DEX
BA81-   D0 F7       BNE   $BA7A
BA83-   60          RTS

BA84- BF EF E4 E5
BA88- E3 EF F2 F0 A0 87 AC E5
BA90- F4 F3 E9 F3 C5 8D
```

Figura 4 - Disassemblato di tutte le modifiche apportate al DOS in modo che ci avverta prima di aggiornare un file Basic. Il testo del messaggio corrisponde al dump da BA84 a BA90 in ordine inverso.

```
A39C:A9 00 85 D7 EA EA EA

B206:20 DF BC

BA69:20 8E FD A0 00 B1 42 20 ED FD
     C8 C0 1E D0 F6 A2 12 BD 83 BA
     20 ED FD CA D0 F7 60

BA84:BF EF E4 E5 E3 EF F2 F0 A0 87
     AC E5 F4 F3 E9 F3 C5 8D

BCDF:AD 63 AA C9 01 D0 16 A6 D7 D0
     12 85 D7 20 69 BA 20 0C FD A2
     FF C9 CE F0 07 C9 D3 D0 F3 AE
     9C B3 60
```

Figura 5 - Dump delle modifiche: da inserire in memoria prima di inizializzare un nuovo dischetto.

Come si fa

Il modo più semplice è di copiarsi il programma in Basic di figura 3 che fa tutto da solo; proprio tutto!

Altrimenti passate al Monitor con CALL -151 e inserite i Dump di figura 5; disassemblate il tutto e confrontateli con quelli di figura 4. Tornate al Basic, scrivete un programmino di HELLO e inizializzate un nuovo dischetto. Provate ora a salvare nuovamente il programma di HELLO e vedrete apparire la scritta:

HELLO
Esiste, proseguo?

rispondete "S" per proseguire il salvataggio (cancellando quindi la vecchia versione) o "N" se si è cambiata idea. Nel caso si risponda "N" alla domanda "proseguo?" si ottiene il messaggio FILE TYPE MISMATCH e il salvataggio viene interrotto.

V. D. D.

Nota: A causa di differenze hardware, in alcuni Apple la locazione D7 all'accensione contiene zero invece di FF.
Se il vostro Apple è uno di questi vi verrà stampato il messaggio di avvertimento durante il Boot del disco. Per evitarlo cambiate lo 0 della DATA 460 in 1, il secondo 208 della DATA 410 in 240 e il 133, sempre alla riga 410, in 198.