ON ERROR GOTO

di Adriano Vertua - Roma

Questo programma serve a supplire alla mancanza dell'istruzione ON ERROR ... GOTO sul VIC-20.

A volte può essere utile poter gestire correttamente gli errori senza far fermare il programma. Per esempio, provate il seguente programma:

```
10 INPUT N
20PRINT N
```

Dando RUN e rispondendo all'input numerico con (ad es.) 1024 il calcolatore passa alla riga 20 e stampa N; rispondendo invece con 1E90 (ovvero 10^{90} troppo grande per l'aritmetica del VIC) lui emette subito un messaggio d'errore ("OVERFLOW ERROR IN 10"). A questo punto bisogna notare che l'esecuzione del programma non è arrivata alla linea 20 (dal momento che non ha stampato il numero) quindi non è possibile controllare il contenuto della variabile N con un IF dopo l'input. Questo è solo uno dei tanti esempi che si potrebbero fare con i messaggi d'errore standard.

```
B*
   PC  SR AC XR YR SP
.;603E 33 00 63 00 F6
.
.
.; 1DDA PHA   SALVATAGGIO NELLO STACK
.; 1DDB TXA   DEL REGISTRO X E
.; 1DDC PHA   DELL'ACCUMULATORE
.; 1DDD LDX #$08  IMMAGAZZINA IN 198
.; 1DDF STX $C6   LA QUANTITÀ DEI CARATTERI
.;                (IN QUESTO CASO 8)
.; 1DE1 LDA $1DD1,X  CICLO PER IMMAGAZZINARE
.; 1DE4 STA $0276,X  I CARATTERI
.; 1DE7 DEX          NEL BUFFER DI TASTIERA
.; 1DE8 BNE $1DE1
.; 1DEA PLA   RECUPERO DEL REGISTRO X E
.; 1DEB TAX   DELL'ACCUMULATORE
.; 1DEC PLA
.; 1DED JMP $C43A  SALTO ALLA NORMALE ROUTINE
.                  DI ERRORE (50234)
.
-
.;1DD2 47 CF 31 30 30  GOTO 19000 [RETURN]
.;1DD7 30 30 0D 48 8A  (esclusi 48 e 8A)
.;1DDC 48 A2 08 86 C6
.;1DE1 BD D1 1D 9D 76
.;1DE6 02 CA D0 F7 68
.;1DEB AA 68 4C 3A C4
.
```

Vediamo adesso cosa succede dando il RUN al programma ON ERROR, listato a parte. Questo ci chiederà un numero in ingresso da tastiera e, in caso d'errore, prima eseguirà le sue funzioni, poi si sostituirà al sistema operativo del personal, stampando anche un nostro messaggio d'errore (quindi non incluso tra quelli standard) e, cosa importante, continuando l'esecuzione della routine. Se rispondiamo alla richiesta numerica con 1024 il programma risponde con OK 1024, e si ferma senza alcun messaggio d'errore. Se invece rispondiamo con 1E90 apparirà, oltre al regolare messaggio "OVERFLOW ERROR IN 260 READY", anche la scritta G ⌜10000, corrispondente ad un GOTO 10000 in forma abbreviata, e sotto un messaggio d'errore più esplicito (e anche più confidenziale), "HO DETTO MINORE DI 1E38"; dopo una breve pausa apparirà nuovamente la richiesta del numero.

Va specificato che il listato del programma non realizza strettamente le funzioni descritte: questo perché la linea 10000 comprende un comando di pulizia schermo, dimodoché il messaggio di overflow e il successivo G ⌜10000 rimarranno sullo schermo per un tempo brevissimo, comunque sufficiente perché siano visibili. Abbiamo scelto di pubblicare questa forma del listato in quanto è certamente la più utile in fase di applicazione.

A voler essere precisi il programma si è fermato a causa dell'errore sull'input numerico, ma immediatamente dopo è ripartito per eseguire la nostra routine, e questo *senza alcun intervento dell'operatore!* senza cioè digitare RUN (RETURN) dopo l'apparizione del famoso messaggio d'errore di overflow. Magia? No di certo! Entriamo dunque nei dettagli.

Il concetto fondamentale del programma si basa su due caratteristiche del sistema operativo: andiamo a vederle da vicino, per comprenderne il funzionamento e di conseguenza regolarci per le modifiche che ci servono.

Prima caratteristica

Quando in un programma BASIC è presente un errore di qualsiasi tipo, il sistema salta ad una routine che invia al video il messaggio d'errore. Il sistema sa dove trovare questa routine perché è indicata da due puntatori residenti nelle locazioni di memoria 768 (low byte) e 769 (high byte), ovviamente indicate in notazione decimale.

Normalmente queste locazioni contengono rispettivamente i numeri 58 e 196; ciò vuol dire che la routine parte dalla locazione:

$$58 + 196*256 = 50234$$

(ovviamente decimale)

Precisando, in caso di errore il sottoprogramma che parte dalla locazione 50234 controlla il contenuto del registro X del microprocessore 6502 (residente nella locazione di memoria 781), lo interpreta come numero di codice dell'errore e invia al video la corrispondente scritta. Per verifica provate il seguente programma:

```
10 INPUT N
20 POKE 781,N
30 SYS 50234;
```

dopo aver dato il RUN rispondere all'INPUT con un numero compreso tra 1 e 30: si otterrà come uscita il messaggio d'errore corrispondente al codice dato in ingresso: per esempio 1 = TOO MANY FILES, oppure 9 = ILLEGAL DEVICE NUMBER e così via.

Bisogna notare ora che le locazioni 768 e 769 risiedono in RAM e sono quindi accessibili all'utente: questo vuol dire che possiamo cambiare i contenuti delle citate locazioni (cioè i puntatori della routine di sistema) indirizzando così il sistema stesso ad una diversa gestione dei messaggi d'errore. Per capire meglio si può dire che facciamo credere al computer che la routine non risiede più in 50234 (ROM) bensì in 7642 (RAM), ove trovasi un programma in linguaggio macchina creato dall'utente che sfrutta le caratteristiche del buffer di tastiera (realizzando in LM quello che abbiamo già visto in BASIC, ovvero la simulazione della digitazione d'una o più linee di comandi diretti) che risiede dalla locazione 631 alla 640.

Seconda caratteristica

Durante l'esecuzione d'un programma nel buffer di tastiera vengono accumulati i caratteri corrispondenti ai tasti premuti.

Le istruzioni INPUT e GET prelevano i caratteri da questo buffer, se ce ne sono: la massima capacità del buffer è di 10 caratteri (ma può esser diminuita mettendo un valore inferiore in 649) e la quantità di caratteri presenti nel buffer è indicata dal contenuto della locazione 198.

Se le istruzioni INPUT e GET non vengono usate, il buffer accumula i caratteri (per l'appunto fino ad un massimo di 10) e si scarica solo quando il programma si ar-

```
10 REM **********************
20 REM **  ON ERROR GOTO  **
30 REM **DI ADRIANO VERTUA**
40 REM **********************
100 POKE51,206:POKE52,29
110 POKE55,206:POKE56,29
120 FORA=0TO29
130 READB:POKE7634+A,B
140 NEXT
150 REM * GOTO *
160 DATA71,207
170 REM* 10000*
180 DATA49,48,48,48,48
190 REM * BUFFER *
200 DATA13,72,138,72,162,8
201 DATA134,198,189,209,29
202 DATA157,118,2,202,208,247
203 DATA104,170,104,76,58,196
210 REM * ERROR PNTR *
220 POKE768,218:POKE769,29
230 REM * PROVA *
240 PRINT"□";
250 PRINT"INSERISCI UN NUMERO";
251 PRINT"   MINORE DI 1E38"
260 INPUT N
270 PRINT"OK ";N
280 POKE768,58:POKE769,196
290 END
10000 PRINT"□HO DETTO MINORE DI"
10001 PRINT"1E38!"
10010 FORT=1TO3000:NEXT
10020 GOTO240
```

resta. Provare (per credere!) il seguente programma:

10 PRINT "[HOME]"PEEK(198):IF PEEK (198) <10 THEN 10

Dare il RUN e premere 10 tasti. Notare che ogni volta che si preme un tasto il contenuto della locazione 198 viene incrementato di 1: quando questo contatore arriva a 10 il programma si ferma e il contenuto del buffer si scarica sul video, visualizzando tutti i caratteri corrispondenti ai tasti premuti.

Dalla combinazione di queste due caratteristiche nasce l'idea dell'istruzione, ON ERROR, o meglio della sua simulazione: vediamo di schematizzarne la procedura.

1) Il sistema verifica la presenza d'un errore in un programma BASIC.

2) Il sistema cerca l'indirizzo della routine di gestione degli errori, che normalmente parte da 50234 (in esadecimale C43A), nelle locazioni 768 e 769.

3) Queste locazioni (che normalmente contengono i valori 58 e 196, ovvero il numero decimale 50234 nella forma 58 + 196*256) contengono, nel caso della nostra routine, i valori 218 e 29, che corrispondono all'indirizzo 7642, locazione ove abbiamo posto la routine che ci serve.

4) Il compito di questo sottoprogramma è caricare nel buffer di tastiera l'istruzione GOTO 10000 (scritta in forma abbreviata) perché nel programma BASIC la linea 10000 è quella che chiarisce l'errore tramite un'opportuna diagnostica. Il caricamento avviene sotto forma di codici ASCII.

La routine deve inoltre caricare nella locazione di memoria 198 (contatore dei caratteri nel buffer di tastiera) il numero 8, appunto la quantità di caratteri che ci serve per simulare il G⌐ 10000 + CR (carriage return, codice ASCII 13). Dopo queste operazioni la routine salterà alla normale gestione degli errori (che avviene con la JMP $C43A), stampando il messaggio standard del VIC e fermando l'esecuzione solita.

5) A questo punto il buffer è pronto ad essere scaricato e vengono eseguiti i comandi citati, ovvero stampa dell'istruzione GOTO 10000 più (simulazione del) RETURN, in modo da eseguire i nostri comandi (memorizzati in BASIC dalla linea 10000 in poi).

Vic-Maze - n. 19

```
100 01=1:I=17:J=1:POKE37154,127
110 GOSUB430:FORL=1TO10:A$=A$+"↖":C$=C$+"/":B$=B$+"/":D$=D$+"↖":NEXT
120 FORL=1TO20:E$=E$+" ":F$=F$+"|":G$=G$+"_":H$=H$+"":NEXT
130 FORL=1TO20:L$=L$+"":I$=I$+"":X$=X$+"":Y$=Y$+"":NEXT
140 IX=0:IY=1
150 GOSUB740
160 A=A%2:B=B%4:X=X%2:Y=Y%4:X1=X-2*(X>0):Y1=Y-1*4*(Y>0)
170 PRINT"":V=X1/2-X/2:V1=-(21-X)*3*(X>0):V2=-(21-X1)*3*(X1<>0):K=Y1/4-Y/4
180 PRINT"";LEFT$(A$,X)"";LEFT$(E$,V1)LEFT$(I$,V1/3)"";LEFT$(Y$,V);
190 PRINTLEFT$(X$,-1*(X>0))LEFT$(I$,-1*(X=0))LEFT$(G$,V)"";LEFT$(E$,V2)LEFT$(I$
,V2/3);
200 PRINT"";LEFT$(I$,-1*(X1=0))LEFT$(A$,A-X1)"";LEFT$(E$,-(21-A)*3*(A>0))
210 PRINT"";LEFT$(B$,X)LEFT$(I$,V)"";LEFT$(H$,V)"";
220 PRINTLEFT$(B$,A-X1)
230 PRINT"";TAB(19)LEFT$(C$,Y)"";LEFT$(F$,-(21-Y/2)*3*(Y>0))LEFT$(I$,21-Y/2)"";
240 PRINTLEFT$(L$,K)LEFT$(Y$,K)"";LEFT$(F$,-(21-Y1/2)*3*(Y1>0))LEFT$(I$,21-Y1/2)
;
250 PRINT"";LEFT$(I$,-1*(Y1<=4))LEFT$(G$,K)LEFT$(L$,K)"";LEFT$(I$,-1*(Y1=0))LEF
T$(C$,B-Y1);
260 PRINT"";LEFT$(F$,-(22-B/2)*3*(B>0))
270 PRINT"";TAB(19)LEFT$(D$,Y)LEFT$(L$,K)LEFT$(I$,K);
280 PRINT"";LEFT$(H$,K)LEFT$(L$,K)"";LEFT$(D$,B-Y1)
290 F=A/2+B/4:O=A/2:IFB/4>A/2THENO=B/4
300 IFR=1THENO=O+1
310 PRINT"";LEFT$(Y$,O)LEFT$(X$,A/2)LEFT$(H$,20-F)LEFT$(L$,20-F)LEFT$(Y$,20-2*O)
"";LEFT$(H$,20-F)
320 IFPEEK(197)=39ORPEEK(37137)=122THEN380
330 IFPEEK(197)=55ORPEEK(37137)=118ANDFL<>1THENIY=-IY:IX=-IX:GOTO150
340 IFPEEK(197)=18ORPEEK(37152)=119ANDFL<>1THENIY=-T*(IY=0):IX=T*(IX=0):GOTO150
350 IFPEEK(197)=41ORPEEK(37137)=110ANDFL<>1THENIY=T*(IY=0):IX=-T*(IX=0):GOTO150
360 IFPEEK(197)=13THENPRINT"   POSIZIONE ATTUALE";K3=K3+10:GOSUB650
370 GOTO320
380 A=A+2*(A>0):X=X+2*(X>0):X1=X1+2*(X1>0):B=B+4*(B>0):Y=Y+4*(Y>0):Y1=Y1+4*(Y1>0
)
390 IFA=0ANDB=0ANDR<>1THENPOKE36878,15:FORZ=1TO255STEP8:POKE36876,Z:NEXT:POKE368
78,0
400 IFA=0ANDB=0ANDFL=1THENO1=0:POKE36879,25:PRINT"":GOSUB660:GOSUB860:POKE37154
,255:END
410 IFA>0ORB>0ORR=1THENI=I+IX:J=J+IY:R=-R*(A>0)
420 GOTO170
430 DIMX%(18):POKE36879,8:PRINT"":TI$="000000"
440 X%(0)=0
450 X%(1)=16126
460 X%(2)=10882
470 X%(3)=10938
480 X%(4)=11144
490 X%(5)=10558+256*(RND(1)>.5)
500 X%(6)=12256
510 X%(7)=8270
520 X%(8)=16248
530 X%(9)=2370+64*(RND(1)>.7)
540 X%(10)=14842
550 X%(11)=10286
560 X%(12)=12264+8*(RND(1)>.5)
570 X%(13)=8510
580 X%(14)=12192+32*(RND(1)>.5)
590 X%(15)=10430
600 X%(16)=10888
610 X%(17)=16126
620 X%(18)=0
630 Z=INT(RND(0)*8+1)*2-1:X%(Z)=X%(Z)+1
640 :PRINT"      LABIRINTO":PRINT"   ? ? ? ? ? ? ? ?"
650 PRINT"  "
660 FORL=01TO14:PRINTTAB(2);
670 FORL1=0TO18:IFL=JANDL1=ITHENPRINT"";:NEXT
680 IF(2↑LANDX%(L1))=0THENPRINT"";:NEXT:GOTO700
690 PRINT" ";:NEXT
700 PRINT:NEXT:PRINT:IFFL=1THENRETURN
710 PRINT"    PREMI UN TASTO "
720 IFPEEK(197)=64ANDPEEK(37137)=126THEN720
730 RETURN
740 X=0:Y=0:R=0:T=IX+IY:XX=I:YY=J
750 IFJ>14ORI>18ORI<=0ORJ<=0THENFORBI=1TO10:H$="":NEXT:I=I+1:J=J+1:B=U:FL=1:A
=U:GOTO810
760 IF(2↑(J+IY)ANDX%(I+IX))=0THENA=U:B=U:GOTO810
770 K=1-ABS(IY):K1=1-ABS(IX):IF(2↑(J-T*K)ANDX%(I+T*K1))<>0THENX=U
780 IF(2↑(J+T*K)ANDX%(I-T*K1))<>0THENY=U
790 J=J+IY:I=I+IX:U=U+1
800 GOTO750
810 IFA=0THEN850
820 A=A-1*((2↑(J+T*K)ANDX%(I-T*K1))<>0)
830 B=B-1*((2↑(J-T*K)ANDX%(I+T*K1))<>0)
840 IFA=BANDU<>ATHENR=1:A=A-1:B=B-1
850 U=0:J=YY:I=XX:RETURN
860 PRINT"SEI USCITO IN",INT(TI/60)"SECONDI"
870 PRINT"VUOI RIGIOCARE? (S/N)"
880 IFPEEK(197)=64THEN880
890 IFPEEK(197)=41THENRUN
900 RETURN
```

Il listato del VIC-MAZE pubblicato nel n. 19 conteneva alcuni errori che ne pregiudicano il funzionamento (per una svista non è andata in stampa la versione definitiva). Pubblichiamo qui il listing corretto, chiedendo scusa e ringraziando i numerosi lettori che ci hanno scritto per segnalarci il problema. Tranquillizzatevi comunque: il VIC-MAZE funziona perfettamente ed è divertentissimo, parola di quello zuccone di Leo Sorge!

Osservazioni

Questo programma può essere adattato a qualsiasi applicazione che richieda il salto ad una linea di programma in caso d'errore. Per modificare il numero di linea ove si rimanda è sufficiente cambiare i valori del blocco DATA in linea 180 (indicata dalla REM in 170) usando i codici ASCII del nuovo numero di linea, che dovrà essere sempre di cinque cifre (a meno di cambiare altre parti del programma: chi è in grado lo può fare senza problemi).

Per esempio se si vuole eseguire un ON ERROR GOTO 5675 la linea 180 dovrà essere riscritta come segue:

180 DATA 48,53,54,55,53

Un consiglio pratico: prima dell'istruzione END posta al termine di questo programma principale (linea 290) occorre ripristinare i normali valori delle locazioni 768 e 769, come descritto (ed attuato) in linea 280. Infatti se dopo aver eseguito il programma commettete un errore (anche in modo diretto) digitando ad esempio TUN anziché RUN il sistema salterà nuovamente alla linea da voi scelta per il vostro messaggio (nel programma listato la linea è la 10000). Provate ad eliminare la linea 280 e date RUN al programma ON ERROR: rispondete all'input con 1 e premete RETURN. Il programma si fermerà dopo aver risposto OK 1. Provate ora a digitare A e RETURN, e vedrete che il programma andrà, voi volenti o nolenti, alla linea 10000. MC