



TEXAS TI99/4A

## Sottoprogrammi in TI Extended BASIC

di Sergio Borsani

Quanto è "esteso" il nostro Extended BASIC? Non è nostra intenzione di parlare diffusamente del modulo di comando SSS, che permette di ampliare il linguaggio del TI99/4A, ma piuttosto di soffermarci su una delle sue caratteristiche più notevoli e forse meno conosciute.

Se infatti si chiedesse quali sono le istruzioni più apprezzate tra quelle fornite dal modulo di estensione, sicuramente la risposta riguarderebbe i comandi grafici per la creazione ed il controllo degli sprite. Mentre gran parte delle sue istruzioni si possono simulare anche con semplici programmi in TI BASIC (è il caso delle funzioni booleane AND e OR o di altre come DISPLAY AT, ACCEPT AT, ecc.), le funzioni grafiche per gli sprite sono invece esclusive dell'Extended BASIC. Esse inoltre risultano particolarmente agevoli per la programmazione, in quanto permettono di evitare l'uso delle POKE e la conoscenza di specifiche locazioni di memoria, come succede invece con altri computer.

Vista l'importanza e la notorietà dei comandi grafici, con molta probabilità passa inosservata un'altra caratteristica esclusiva dell'Extended-BASIC: la gestione dei sottoprogrammi. Normalmente per sottoprogrammi si intendono subroutine richiamate con un'istruzione GOSUB. I sottoprogrammi dell'Ext-BASIC sono invece qualcosa di più potente, anche se sostanzialmente analogo.

Un sottoprogramma è un programma distinto dal programma principale, non solo perché necessariamente deve essere posto al termine di esso, dopo l'istruzione END, ma anche perché le sue variabili, pur identiche a quelle del programma principale, si comportano in modo del tutto indipendente da esse. Ad esempio, se nel programma principale si è posto N=10, si può usare la variabile N anche nel sottoprogramma ed attribuire ad essa valori diversi, come 11, 12, 13, ecc. Quando si torna al programma principale, N assumerà di nuovo il valore precedente alla chiamata del sottoprogramma, cioè 10. Infatti, all'uscita di un sottoprogramma tutte le sue variabili vengono azzerate ed il programma principale riprende con le sue variabili poste al valore che avevano al momento della chiamata.

Un eventuale passaggio di valori tra il programma ed il sottoprogramma è possibile ed avviene solo sotto il controllo del programmatore.

A chi scrive non risulta che altri computer del livello medio-basso ed altre versioni di BASIC posseggano una simile capacità e al lettore attento non sfuggirà come ciò consenta una più agevole programmazione, sul tipo della "programmazione strutturata" ottenibile in Pascal.

In pratica è possibile costituire una libreria di routine da inserire, o più precisamente agganciare, ai normali programmi senza alcuna modifica e senza alcuna preoccupazione circa la compatibilità delle variabili.

A chi non è successo di "sporcare" il valore di una variabile perché inconsapevolmente aveva usato la stessa in una subroutine? Con l'uso dei sottoprogrammi ciò non succede! Quali istruzioni permettono la loro gestione?

Un sottoprogramma si apre con le istruzioni SUB "nome" (lista variabili). Essa, come è già stato detto, deve seguire il programma principale ed essere posta solo dopo un'istruzione END o una REM.

La lista delle variabili tra parentesi è il cordone ombelicale tra il programma madre ed il sottoprogramma e consente il passaggio di particolari valori nei due sensi.

Il sottoprogramma deve terminare solo con l'istruzione SUBEXIT o SUBEND, che pertanto hanno una funzione analoga al RETURN di una subroutine.

Per accedere ad un sottoprogramma si usa l'istruzione: CALL "nome" (lista variabili), che funziona come una GOSUB.

La linea delle variabili tra parentesi deve trovare una esatta corrispondenza con quella presente nell'istruzione SUB e serve per trasmettere e ricevere particolari valori. Alcuni esempi chiariranno meglio tutto il meccanismo. Battete il seguente programma.

```
100 REM SOTTOPROGRAMMA
110 REM *****
120 CALL CLEAR :: N=9999
130 PRINT "SCRIVI UNA PAROLA" :: INPUT A$
140 CALL REVERSE(A$)
150 PRINT " " : " " : A$
160 PRINT :: PRINT "N =" ; N :: END
1000 SUB REVERSE(X$)
1010 LN=LEN(X$) :: B$=""
1020 FOR N=LN TO 1 STEP -1
1030 B$=B$&SEG$(X$,N,1)
1040 NEXT N :: X$=B$
1050 SUBEND
```

La lista 140 chiama un particolare sottoprogramma, chiamato REVERSE, che accetta una stringa e la trasforma nella stringa inversa, cioè nella stessa, come risulterebbe leggendola da destra a sinistra. Nel programma principale A\$ contiene la parola specificata, mentre la variabile N è stata inserita solo a titolo dimostrativo.



## Sottoprogrammi in TI Extended BASIC

Il sottoprogramma riceve il contenuto di A\$ nella variabile X\$ e successivamente trasforma X\$ nella stringa inversa. All'uscita del sottoprogramma il contenuto di X\$ viene trasferito nuovamente in A\$. Lo scambio dei valori avviene solo tra le variabili poste tra parentesi nella CALL e nella SUB. Le altre variabili, pur con lo stesso nome, sono indipendenti le une dalle altre.

A conferma di ciò si è usata la N come variabile di lavoro nel sottoprogramma. Nonostante ciò, tornati nel programma principale, N mantiene il suo primitivo valore, cioè 9999.

A maggior riprova della potenza e della versatilità dei sottoprogrammi c'è un'opzione che permette di trasmettere un valore dal programma al sottoprogramma senza alterare nemmeno le variabili contenute tra parentesi nella CALL. Modificate nel modo seguente il programma ed eseguitelo.

```
100 REM SOTTOPROGRAMMA
110 REM *****
120 CALL CLEAR :: N=9999
130 PRINT "SCRIVI UNA PAROLA" :: INPUT A$
140 CALL REVERSE((A$))
150 PRINT :: PRINT "PROGRAMMA:":"A$ = "; A$
160 PRINT :: PRINT "N =";N :: END
1000 SUB REVERSE(X$)
1005 PRINT :: PRINT "SOTTOPROGRAMMA 1:":"X$ = ";X$
1010 LN=LEN(X$):: B$=""
1020 FOR N=LN TO 1 STEP -1
1030 B#=B$%SEG$(X$,N,1)
1040 NEXT N :: X#=B$
1045 PRINT "SOTTOPROGRAMMA 2:":"X$ = ";X$
1050 SUBEND
```

Questa volta il contenuto di X\$ non è tornato in A\$ e questo semplicemente perché A\$ è stata inserita nella CALL tra un'ulteriore coppia di parentesi.

Come ultimo esempio viene riportato un sottoprogramma che, a differenza dai primi, può rivestire qualche utilità nei programmi che richiedono l'introduzione del tempo nel sistema sessagesimale.

Esso inizia alla linea 1000 del listato 1 e porta il nome convenzionale di TIME. La sintassi per richiamarlo è: CALL TIME (n° riga, n° colonna, "00:00:00.00", variabile numerica).

Dopo la CALL appare sullo schermo la stringa 00:00:00.00 nella posizione specificata dal numero di riga e di colonna (il sottoprogramma non è protetto contro quei valori che porterebbero parzialmente o totalmente la stringa fuori dallo schermo).

Un cursore, con la forma di una lineetta bianca, si posiziona all'inizio della stringa e l'utente può inserire il tempo, specificando ore, minuti e secondi, ed

### Listato 1 - Il programma Time.

```
100 REM SOTTOPROGRAMMA
110 REM TIME ENTRY
120 REM *****
130 REM versione
140 REM TI EXTENDED BASIC
150 CALL CLEAR :: CALL CHAR(128,"00000000
0000FFFF")
160 CALL TIME(12,10,"00:00:00.00",S)
170 PRINT "SECONDI =";S
180 END
1000 SUB TIME(R,C,B$,TEMPO)
1010 DISPLAY AT(R,C)BEEP:B$
1020 Y=(R-1)*8+1 :: X=(C+1)*8+1 :: CALL
SPRITE(#1,128,16,Y,X)
1030 LN=LEN(B$):: C1=C :: C2=C+LN-1
1040 CALL KEY(S,K,S):: IF S=0 THEN 1040
1050 IF K=8 AND C>C1 THEN C=C-1 :: W=-1
:: GOTO 1140
1060 IF K=9 AND C<C2 THEN C=C+1 :: W=1
: GOTO 1140
1070 IF K=32 THEN W=1 :: GOTO 1130
1080 IF K=13 THEN 1160
1090 IF K<48 OR K>57 THEN 1150
1100 CALL HCHAR(R,C+2,K):: W=1 :: IF C=C
1 THEN B$=CHR$(K)&SEG$(B$,2,LN-1):: GOTO
1130
1110 IF C=C2 THEN B$=SEG$(B$,1,LN-1)&CHR
$(K):: GOTO 1130
1120 B$=SEG$(B$,1,C-C1)&CHR$(K)&SEG$(B$,
C-C1+2,C2-C)
1130 IF C<C2 THEN C=C+1
1140 IF C=C1+2 OR C=C1+5 OR C=C1+8 THEN
C=C+W
1150 X=(C+1)*8+1 :: CALL LOCATE(#1,Y,X):
: GOTO 1040
1160 TEMPO=VAL(SEG$(B$,1,2))*3600+VAL(SE
G$(B$,4,2))*60+VAL(SEG$(B$,7,5))
1170 CALL DELSPRITE(#1)
1180 SUBEND
```

usare i tasti di funzione FCTN (←) e FCTN (→) per editare ciò che ha scritto.

Dopo la scrittura delle ore il cursore passa automaticamente ai minuti e successivamente ai secondi, saltando il segno ":" o il punto che separa i secondi dai centesimi.

Il sottoprogramma trasforma il tempo tutto in secondi e restituisce il valore calcolato nell'ultima variabile presente tra parentesi nell'istruzione CALL che, in questo particolare esempio, è stata posta uguale a S. È inutile aggiungere che un simile sottoprogramma può essere vantaggiosamente agganciato a tutti i programmi di matematica e fisica che trattano il tempo. Chi poi possiede un sistema a dischi non dovrà nemmeno trascriverlo tutte le volte, poiché si limiterà semplicemente ad usare il comando MERGE.

Quando il sottoprogramma SUBTIME è agganciato al programma principale, l'istruzione CALL TIME si comporta come qualsiasi altra istruzione implementata nel modulo di comando, con una precisa sintassi da rispettare.

Il linguaggio Ext-BASIC risulterà arricchito ed ampliato in modo personale dall'utente.