MC software SOA

a cura di Pierluigi Panunzi

In questo numero "accontentiamo" quei lettori che ci hanno scritto sollecitandoci a dedicare spazio, nell'ambito del software per calcolatrici Texas Instruments, alla sorella minore della serie la TI-57.

L'aggettivo "minore" è in pratica riferito alle dimensioni hardware-software della 57 rispetto alle 58-59: infatti questa calcolatrice presenta, quasi come in embrione, tutte le caratteristiche di una programmabile che si rispetti anche se per certi motivi si distacca un po' dalle sorelle maggiori. Per i dettagli rimandiamo senz'altro alla rubrica "L'angolo delle TI", mentre lasciamo la parola ad un lettore di Milano, Tommaso Berardinelli, il quale ha inviato alla redazione qualcosa come 13 programmi, che spaziano da argomenti "didattici" (esercitazioni sulle tabelline) alla matematica (fattori primi, interesse, equazioni di terzo grado) e ai giochi (l'onnipresente Master-Mind, la Morra cinese, ecc.). Abbiamo scelto tre programmi, in quanto le ovvie esigenze di spazio ci impediscono di considerarne altri, che ci sembrano rappresentativi delle capacità, solo apparentemente limitate della TI-57.

Scomposizione di un numero in fattori primi

di Tommaso Berardinelli (Milano)

Questo primo programma, in appena 39 istruzioni (confessiamo: ne abbiamo eliminate due in quanto non indispensabili e che provocavano in alcuni casi una certa confusione nel funzionamento) permette il calcolo dei fattori primi di un qualsiasi numero intero.

Come si può vedere dal flow-chart di fig. 1 il procedimento è molto semplice: si divide il numero di partenza N prima per 2 e poi per 3, 5, 7, ecc. finché si trova il primo fattore F1, cioè il numero che è divisore di N, e che viene mostrato sul display.

Fatto ciò si sostituisce ad N il valore N/F1 e su questo si ricomincia il procedimento di ricerca di (eventuali) F2, F3 ecc.

Ciò viene ripetuto per tutto il tempo che il divisore (l'eventuale fattore primo) rimane minore o uguale alla radice del numero N, sul quale si sta effettuando la ricerca.

Si può vedere sempre dal flow-chart che il divisore, contenuto in R2, vale al principio 2 e tale rimane finché il valore N è pari (e perciò verrà mostrato tante volte il fattore 2), dopodiché R2 viene ogni volta incrementato del contenuto di R3. All'inizio questo contenuto vale 1 (dato che il divisore deve passare da 2 a 3) e successivamente viene portato a 2 in quanto il divisore, una volta assunto il valore 3, dovrà passare a 5, poi a 7 e cioè tutti valori dispari.

Infatti è perfettamente inutile dividere il numero N per 4, 6, 8 in quanto sicuramente non sarà più pari.

Stesso discorso si dovrebbe fare per i valori di R2, per 9, 15, 21, 27, ecc. e cioè i multipli dispari di 3 e successivamente quelli di 5, poi quelli di 7, ecc.

Per evitare cioè di compiere divisioni inutili si dovrebbe in realtà far sì che il divisore R2 sia ogni volta un numero primo. Ciò però ci porta inevitabilmente molto lontano, ben al di là dei 50 passi disponibili.

L'unico inconveniente di questa "scelta forzata" è perciò un allungamento dei tempi di elaborazione, che solo per valori altissimi di N può risultare inaccettabile.

Prima di passare all'esempio del programma, segnaliamo un piccolo "bug" nella parte di programma che segue la Lbl 6: in questo frammento deve essere effettuato il test se N2 è minore o uguale alla radice quadrata di N.

I "puristi" avranno senz'altro notato la presenza di ben due test: uno per il "minore" e l'altro per l'"uguale", avendo posto nel registro t (che nella 57 è anche R7) la radice ed in x il divisore.

Era senz'altro più semplice scambiare i ruoli della x e della t ed usare semplicemente il test $x \leq t$ che letto alla rovescia suona così: t minore o uguale ad x!!

Tra l'altro in questo modo i passi di programma si riducono ulteriormente di due unità mentre la velocità di elaborazione risulta teoricamente più alta per la presenza di un solo test invece di due: ma ricordiamoci che queste sono finezze...

Un rapido sguardo all'utilizzazione del programma: si inizializza il tutto premendo RST ed R/S, dopodiché si imposta il numero N e si preme R/S. Dopo un tempo variabile a seconda del valore di N e del suo primo fattore, l'elaborazione si fermerà mostrando quest'ultimo sul display.

```
DECOMPOSIZIONE  IN
 FATTORI PRIMI

00  48 0        FIX 0
01  01            1
02  32 3        STO 3
03  85            =
04  81          R/S
05  32 1        STO 1
06  02            2
07  32 2        STO 2
08  86 4        LBL 4
09  00            0
10  32 7        STO 7
11  33 1        RCL 1
12  45            ÷
13  33 2        RCL 2
14  85            =
15 -49        I INT
16  66          EQ
17  51 7        GTO 7
18  33 3        RCL 3
19  34 2        SUM 2
20  02            2
21  32 3        STO 3
22  86 6        LBL 6
23  33 1        RCL 1
24  24          √X
25  32 7        STO 7
26  33 2        RCL 2
27 -76        I GE
28  51 4        GTO 4
29  66          EQ
30  51 4        GTO 4
31  33 1        RCL 1
32  81          R/S
33  71          RST
34  86 7        LBL 7
35  33 2        RCL 2
36  81          R/S
37 -39 1      I PRD 1
38  51 6        GTO 6
```

```
   EQUAZIONI DI
  TERZO  GRADO

00  15          CLR
01  32 4        STO 4
02  81          R/S
03  32 1        STO 1
04  81          R/S
05  32 2        STO 2
06  81          R/S
07  32 3        STO 3
08  86 4        LBL 4
09  33 1        RCL 1
10  32 6        STO 6
11  33 4        RCL 4
12  34 6        SUM 6
13  39 6        PRD 6
14  33 6        RCL 6
15  32 5        STO 5
16  34 5        SUM 5
17  33 2        RCL 2
18  34 6        SUM 6
19  34 5        SUM 5
20  33 4        RCL 4
21  39 6        PRD 6
22  23          X²
23  34 5        SUM 5
24  33 3        RCL 3
25  34 6        SUM 6
26  33 5        RCL 5
27  40          |X|
28 -39 6      I PRD 6
29  33 6        RCL 6
30 -34 4      I SUM 4
31  40          |X|
32  76          GE
33  51 4        GTO 4
34  33 4        RCL 4
35  81          R/S
36  34 1        SUM 1
37 -39 3      I PRD 3
38  02            2
39  84          +/-
40 -39 1      I PRD 1
41  33 1        RCL 1
42  81          R/S
43  23          X²
44  34 3        SUM 3
45  33 3        RCL 3
46  24          √X
47  81          R/S
48  71          RST
```

```
CALENDARIO PERPETUO

00  48 0        FIX 0
01  81          R/S
02  32 6        STO 6
03  05            5
04  34 6        SUM 6
05  81          R/S
06  32 1        STO 1
07  55            ×
08  33 4        RCL 4
09  85            =
10  49          INT
11  34 6        SUM 6
12  81          R/S
13  32 0        STO 0
14  55            ×
15  33 3        RCL 3
16  75            +
17  33 1        RCL 1
18  55            ×
19  33 2        RCL 2
20 -34 6      I SUM 6
21  85            =
22  34 6        SUM 6
23  56          DSZ
24  04            4
25 -39 0      I PRD 0
26  33 0        RCL 0
27  49          INT
28  34 6        SUM 6
29  33 1        RCL 1
30 -76        I GE
31  51 6        GTO 6
32  33 0        RCL 0
33 -49        I INT
34  38 5        EXC 5
35  34 5        SUM 5
36  38 5        EXC 5
37  49          INT
38  34 6        SUM 6
39  86 6        LBL 6
40  33 6        RCL 6
41  49          INT
42  45            ÷
43  07            7
44  85            =
45 -49        I INT
46  55            ×
47  07            7
48  85            =
49  71          RST
```

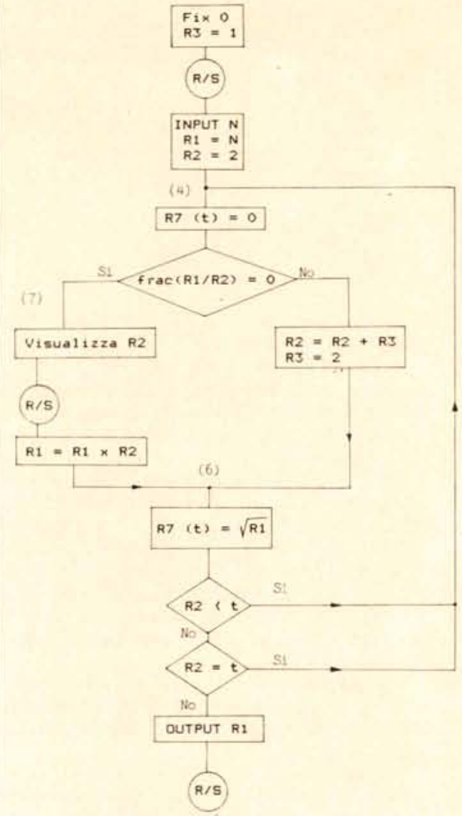

fig.1
Flow-chart di
"Scomposizione in fattori primi"

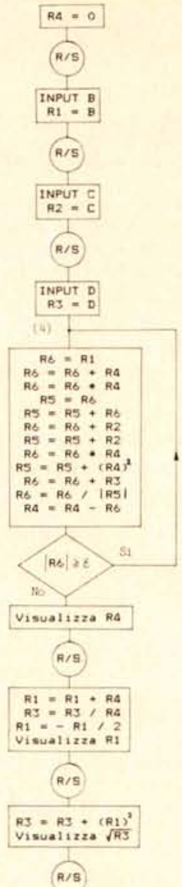

fig.2
Flow-chart di
"Equazione di terzo grado"

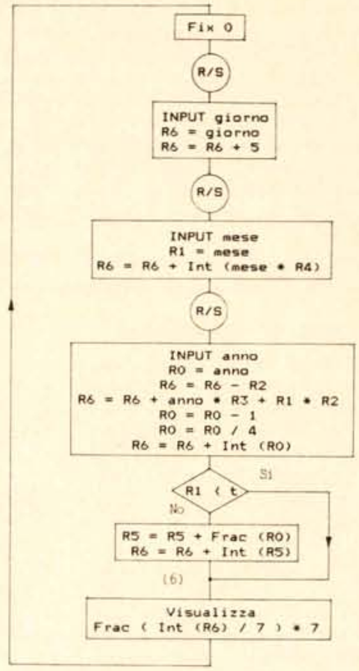

fig.3
Flow-chart di
"Calendario Perpetuo"

Il tempo di annotare questo valore e premiamo ancora R/S per avere il secondo fattore e così via. Quando sul display vedremo il valore "1" vuol dire che la decomposizione è terminata.

Equazioni di terzo grado

di Tommaso Berardinelli (Milano)

Ecco un programma che può risultare molto utile in alcuni casi e che fa piacere veder risolto su una "piccola" TI-57 nonostante l'algoritmo usato risulti "importante".

Si tratta appunto della ricerca delle tre radici di un'equazione di terzo grado ($x^3 + Bx^2 + Cx + D$), ricerca che avviene, per la prima radice, per approssimazioni successive, con il metodo della derivata o metodo di Newton-Raphson di primo grado.

Calcolata poi la prima radice (x_1) si effettua la divisione dell'equazione di partenza per il fattore ($x-x_1$) ottenendosi un'equazione di secondo grado che viene risolta con il metodo che tutti conosciamo.

Torniamo alla prima radice: abbiamo detto che viene calcolata per approssimazioni successive a partire da un valore iniziale nullo (contenuto in R4) e andando volta per volta a sommare a questa quantità un termine correttivo fino a che non si ottiene la precisione richiesta cioè fino a che Δ non diventa, in valore assoluto, minore o uguale ad un valore ε prefissabile a piacere.

Il valore di Δ, che come si vede è funzione di x, è dato da:

$$\Delta (x) = -f(x)/f'(x)$$

e nel nostro caso vale:

$$\Delta(x) = -(x^3 + Bx^2 + Cx + D)/(3x^2 + 2Bx + C)$$

mentre per il valore della radice si ha la formula ricorrente

$$x = x_0 + \Delta(x)$$

dove all'inizio x_0 vale 0 e successivamente assume il valore volta per volta trovato di x.

In definitiva una volta ottenuto il valore della radice x_1, sicuramente reale, le altre due radici x_2 e x_3 potranno essere o entrambe reali e distinte, o reali e coincidenti oppure complesse coniugate, secondo la ben nota regola per le equazioni di secondo grado.

La calcolatrice, per x_2 e x_3 mostrerà successivamente due quantità m e k (con k eventualmente lampeggiante): se k non lampeggia siamo nei primi due casi e le radici saranno date da

$$x_2 = m + k$$
$$x_3 = m - k$$

che coincideranno se $k = 0$.

Viceversa se k appare lampeggiante siamo in presenza di radici complesse coniugate di cui m rappresenta la parte reale e k quella immaginaria e cioè

$$x_2 = m + ik$$
$$x_3 = m - ik$$

Rapidamente vediamo il funzionamento: si inizializza il tutto impostando il valore di ε (precisione) in STO 7, ad esempio .0000001 STO 7 e si preme RST e R/S.

Quindi si impostano, premendo ogni volta R/S, i tre coefficienti dell'equazione (B, C, D) ovviamente impostando 0 se uno o più di questi coefficienti risultano nulli. Quando si preme l'ultimo R/S (per impostare il valore di D) si ha l'avviamento dell'elaborazione che consiste in un ciclo controllato, come visto, dal valore assoluto di Δ: quando si esce da questo loop il display mostrerà x_1.

Premendo successivamente R/S comparirà il valore m e con R/S il valore k eventualmente lampeggiante.

Ad esempio calcoliamo le radici di

$$x^3 + 12x^2 + 4x + 103 = 0$$

Si troveranno i valori seguenti:

x_1 = -12.351314
x_2 = .1756572 + i 2.8824188
x_3 = .1756572 - i 2.8824188

cioè k ci appare lampeggiante (deriva infatti dalla radice quadrata di un numero negativo).

Un'ultima nota riguarda il metodo usato: può

INVIATECI I VOSTRI PROGRAMMI!

Se, qualunque sia la vostra macchina, avete realizzato programmi o routine che ritenete possano interessare altri lettori, inviateceli. Saranno esaminati e, se pubblicati, ricompensati con valutazioni approssimativamente fra le 30 e le 100.000 lire, secondo la complessità, la genialità, l'originalità e la presentazione del materiale e della documentazione (listati, diagrammi, commenti ecc.). Per ragioni organizzative non possiamo impegnarci, salvo eventuali accordi presi prima dell'invio, alla restituzione dei materiali, che resteranno di proprietà della redazione che si impegna a non divulgarli (se non tramite la rivista) senza l'autorizzazione dei rispettivi autori.

capitare che nel corso dei calcoli il valore approssimato di x cada proprio in un punto in cui la derivata f'(x) si annulla, nel qual caso per calcolare il (x) si ha una divisione per zero.

In questo caso l'elaborazione si ferma mostrando uno zero lampeggiante: si deve perciò premere CE e RCL 4 che ci mostra il valore di x "incriminato". Si sceglie a questo punto un valore più a sinistra (o minore che dir si voglia) e lo si pone in STO 4 dopodiché si fa riprendere l'elaborazione con SBR 4. Se anche in questo caso si otterrà uno 0 lampeggiante si dovrà procedere per tentativi scegliendo valori per la x ancora più a sinistra e ripetere le operazioni segnalate.

Comunque a parte questo caso, il programma funziona perfettamente, mentre viceversa, per il solito motivo dei 50 passi, non era possibile includere un'autocorrezione in caso di errore.

Calendario perpetuo

di Tommaso Berardinelli (Milano)

Quest'ultimo programma consente di trovare il giorno della settimana corrispondente ad una qualsiasi data compresa tra il 1582 ed il 2500. La risposta della macchina è un numero compreso tra 0 e 6 la cui corrispondenza con i giorni della settimana è la seguente: 0 = domenica, 1 = lunedì, ..., 6 = sabato.

Praticamente però il risultato ottenuto è esatto solo per questo ed il prossimo secolo, mentre per date anteriori e posteriori si devono apportare (mentalmente!!!) le seguenti correzioni al risultato ottenuto:

da	a	correzione
15-10-1582	28-2-1700	+3
1-3-1700	28-2-1800	+2
1-3-1800	28-2-1900	+1
1-3-1900	28-2-2100	0
1-3-2100	28-2-2200	-1
1-3-2200	28-2-2300	-2
1-3-2300	28-2-2500	-3

correzioni che tengono conto che gli anni 1600, 2000 e 2400 sono bisestili.

Anche in questo caso il programma è molto semplice, in quanto si tratta di effettuare una serie di operazioni elementari (somme, moltiplicazioni e divisioni) per il calcolo del coefficiente contenuto in R6.

Però, al solito, per far entrare il tutto in 50 passi, l'autore ha ben sfruttato le capacità della TI-57, tramite le operazioni "in" memoria (si veda anche il secondo programma presentato) e tramite dei "trucchi" programmativi. In particolare al passo 23 compare l'istruzione Dsz il cui compito, nel nostro caso, è semplicemente di decrementare di un'unità il contenuto del registro R0 (anno).

I possessori di TI-58 e TI-59, che non conoscano il funzionamento della TI-57 troveranno ancora una volta nell'"Angolo delle TI" dei ragguagli in merito.

Inoltre ai passi 34, 35, 36 compare una sequenza (Exc 5, SUM 5, Exc 5) che permette di sommare al contenuto del visualizzatore il contenuto di R5, senza alterare quest'ultimo registro.

È questo senza dubbio un metodo alternativo alla banale (!) sequenza + RCL 5 = che tra l'altro occupa lo stesso numero di passi...

Effettuata sulle 58 o 59, la sequenza proposta occuperebbe sei passi di memoria (Exc, 05, SUM, 05, Exc, 05) invece dei quattro di + RCL 05 =.

Allargando un po' il discorso e ragionando in termini cari a chi si occupa di microprocessori, si può notare che la sequenza usata richiede tre accessi alla memoria contro l'unico dell'altra sequenza e ciò può non essere desiderabile in certi casi.

Comunque le due sequenze funzionano entrambe correttamente, considerato che a questi livelli non si creano problemi se una sequenza magari dura qualche micro-secondo in più o in meno: ma bisogna pur sempre ricordare che dentro alla TI-57 c'è un microprocessore...

Tra l'altro, a parte una resistenza ed un condensatore, è anche l'unico componente presente in questa calcolatrice.

Torniamo al programma e vediamone il funzionamento: innanzitutto bisogna impostare delle costanti nelle memorie:

30 STO 2
365 STO 3
0.5625 STO 4
-2.5 STO 5
3 STO 7 (registro t)

poi si preme RST ed R/S.

A questo punto si introducono i valori numerici del giorno, del mese e dell'anno, premendo ogni volta R/S. Al terzo R/S (dopo l'impostazione dell'anno) parte l'elaborazione e dopo un po' si avrà sul display il risultato, eventualmente da correggere come dalla tabellina vista prima e quindi da convertire in giorno della settimana.

Più semplice di così!

MC

L'angolo delle TI

Dopo aver parlato negli scorsi numeri di alcune caratteristiche più o meno sconosciute delle calcolatrici TI-58 e TI-59, questa volta ci occuperemo della TI-57 e cercheremo di mostrarne il funzionamento a chi non la conosce, indicandone le differenze con i modelli maggiori.

Innanzitutto cominciamo con la memoria, che consente l'uso di 50 passi di programma e 8 registri dati, senza la possibilità di effettuare una ripartizione diversa. Mentre però nei modelli superiori la lunghezza di un'istruzione può variare da 1 byte (Sin, Int, +, ecc.) a ben 6 byte (INV Dsz Ind 00 Ind 01, INV Iffg Ind 00 Ind 01), nella TI-57 viene usato un codice compattato in modo tale che anche un'istruzione, che dovrebbe essere da tre byte (INV SUM 0) viene riunita in un solo byte.

In particolare il codice con cui viene indicata una certa istruzione in un programma è del tipo "S CC N", dove S (Segno) vale "-" se è stato premuto il tasto INV, CC (Codice Codice) è l'usuale codifica della tastiera degli altri modelli (che tiene ovviamente conto che i tasti e le funzioni sono molte di meno) ed N è il numero della memoria o della label a seconda che l'istruzione lo richieda. Ecco che INV SUM 0 diventa "-34 0", il tutto in un unico byte.

In questo modo si può già vedere che i 50 passi equivalgono grosso modo ad un centinaio di passi delle "maggiori".

Per quanto riguarda le memorie, si risente della semplicità della 57, in quanto gli 8 registri hanno anche altre funzioni; in particolare:

R0 è utilizzato dall'istruzione Dsz per il controllo dei loop;

R5 e R6 sono usati dal Sistema Operativo Algebrico in caso di operazioni in sospeso (non esiste qui uno stack, né tantomeno una HIR);

R7 infine non è altro che il registro "t" che si usa nei confronti.

Inoltre tutti i registri, eccetto R6, vengono utilizzati nel caso di funzioni statistiche.

È ovvio che questa sovrapposizione di compiti può costringere a volte a dei piccoli salti mortali da parte dell'utente, comunque ci si può accontentare.

Altra differenza riguarda le label che, come già visto, sono numeriche (tutto l'opposto delle altre TI) con valori da 0 a 9, che possono essere chiamate o come subroutine (es. SBR 5) o con salti incondizionati (es. GTO 6).

L'argomento dei salti ci porta dunque ad un'altra differenza "logica" della TI-57 nella gestione appunto dei salti condizionati (x = t, INV x = t, x ≤ t, INV x ≤ t) e dei cicli (Dsz). Mentre nelle 58 e 59 subito dopo l'istruzione di salto condizionato deve comparire il nome di un'etichetta oppure un indirizzo numerico, dove il programma salta solo se è verificata la condizione del test, nel caso della 57, una volta effettuato il test, se questo è verificato si esegue l'istruzione successiva, altrimenti questa viene saltata per proseguire in sequenza.

Perciò se si vuole un salto ad una certa etichetta a seguito del verificarsi di una certa condizione, bisogna proprio porre GTO n.

Analogamente avviene per l'istruzione Dsz, controllata dal contenuto di R0: quando il contenuto di questo registro diventa 0, viene saltata l'istruzione successiva al Dsz. Se perciò si vuole creare un loop bisogna impostare ad esempio:

Lbl 0 ... (istruzioni) ... Dsz GTO 0 ...

Torniamo un attimo al terzo programma presentato, dove vediamo appunto al passo 23 un Dsz che, come visto, decrementa di un'unità il contenuto di R0: ora dato che in R0 è posto l'anno, senz'altro maggiore di 1582 e perciò maggiore di 0, si ha che l'istruzione successiva viene comunque effettuata, cosa che era ovviamente nelle intenzioni dell'autore.

Se per divertimento si imposta come anno il valore 1 si ha che, arrivati al Dsz, questo 1 diventa 0, viene saltata l'istruzione successiva ed il programma cade in errore.

Altre differenze sostanziali, oltre quelle citate, non ve ne sono: in pratica, come si può vedere già dalla tastiera, mancano molte funzioni (le Op, i flag, le operazioni con la stampante, ecc.) ma di certo era troppo pretenderle.

Un'ultima caratteristica è il fatto che la TI-57, durante l'esecuzione di un programma, mostra sul display tutti i risultati parziali, però a velocità molto elevata e con il display stesso debolmente illuminato, così come succede nelle 58 e 59 quando si preme il tasto GTO durante l'elaborazione. Per chi è abituato alla fioca "C" che appare sulla sinistra quando le "sorelle maggiori" sono all'opera, il vedere quella sarabanda di cifre mutevoli è alquanto sconcertante.

Ciò può comportare degli inconvenienti quando si imposta un programma di giochi in cui un eventuale numero segreto può essere mostrato più volte sul display...

Con questo abbiamo chiuso la panoramica sulla TI-57: se qualche lettore avesse delle notizie inedite su questa calcolatrice (qualche istruzione nascosta, chissà!) ce le comunichi e le pubblicheremo in un prossimo numero della rubrica.

P.P.