# software RPN

*a cura di Paolo Galassetti*

## Riordino Registri 41C

*di Giulio Croci - Medicina (BO)*

*Spett.le Redazione*

*vi invio per l'esame e l'eventuale pubblicazione (quale presunzione mostruosa!!) un programmino in R.P.N. per HP 41C.*

*Ha il vantaggio di essere compatto (21 linee) e, soprattutto, di utilizzare esclusivamente la catasta operativa; uso: ordinamento di qualsiasi gruppo di dati numerici contenuti in registri consecutivi (in teoria; in pratica, al di là di poche decine i tempi diventano eccessivi).*

```
01*LBL "ORD"      12 X<=Y?
02 STO T          13 X<>Y
03*LBL 08         14 STO IND Z
04 RCL T          15 ISG Z
05 ENTER↑         16 GTO 09
06 ENTER↑         17 X<>Y
07 RCL IND X      18 STO IND T
08 ENTER↑         19 ISG T
09*LBL 09         20 GTO 08
10 CLX            21 RTN
11 RCL IND Z      22 END
```

**Riordino Registri 41C**

*Vado di fretta e non ho il tempo di fermarmi a fare quattro chiacchiere: ripasserò. Complimenti per la rivista.*

*Uso del programma:*

*Scopo - ordinare il contenuto dei registri da $N_1$ a $N_n$ in ordine crescente.*

*Input - introdurre nel reg. X il primo e l'ultimo registro nella consueta forma $N_1 \cdot N_n$, con $N_n$ = 001; 010; ecc.*

*così ad esempio inserendo 10.025 verranno ordinati i registri dal 10 al 25*

- *avviare il programma o utilizzarlo come subroutine.*

*Accetta qualsiasi serie di registri, reg. 00 compreso.*

*Schema - ricerca il valore più basso e lo mette nel primo registro, poi il più basso dei restanti e lo mette nel secondo e così via.*

Il programma funziona in modo perfetto e oltretutto ha il pregio non indifferente di lavorare esclusivamente con la catasta. La rapidità di esecuzione è buona, ovviamente con serie molto lunghe di registri occorre un certo tempo per il riordino.

Il programma usa come contatori i registri Z e T della catasta; il registro Z è utilizzato dal loop costituito dai passi 09 ÷ 16, che serve a ricercare il valore più basso di tutta la serie di registri esaminata; una volta trovatolo, l'istruzione "18 STO IND T" lo pone nel primo registro, poi l'operazione si ripete ma ripartendo dal registro successivo. Questo avanzamento dal punto di partenza ad ogni iterazione viene operato dal registro T con l'istruzione "19 ISG T".

**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.

## Fattoriali 33E/33C

*Andrea Colasanti - Roma*

*Gentile Redazione,*

*partecipo con gioia alla Vostra brillante e simpatica iniziativa di pubblicare programmi dei lettori per i lettori, anche perché spero che non sia riservata ai soli, fortunati, possessori di una 41C.*

*La mia fedele collaboratrice si chiama infatti HP-33C. Non ha grandi pretese di potenza e versatilità, ma è molto pratica per apprendere i primi rudimenti della programmazione, ed è sempre pronta ad elargire non poche soddisfazioni.*

*Una delle principali mancanze ovviabili, di questa calcolatrice, penso che sia l'assenza della funzione fattoriale: n!*

*Quello che segue è il programma più breve che son riuscito a realizzare a tal scopo. Penso possa rimanere molto utile non solo come programma a se stante ma anche come "subroutine" in altri programmi più complessi con le opportune piccole modifiche che indicherò in seguito.*

*Vorrei ora ricordare, brevemente, come si definisce la funzione fattoriale.*

$n! = 1 \times 2 \times 3 \times \ldots \times (n-2) \times (n-1) \times n$

*Il mio programma calcola questa funzione per* $1 < n \leq 69$

*Ricordo che 1! = 1; il che è banale e di*

**Calcolo del Fattoriale**

```
 1 STO 0
 2 1
 3 -
 4 STO 1
 5 1
 6 RCL 1
 7 X≠Y
 8 GTO 11
 9 RCL 0
10 R/S
11 RCL 0
12 RCL 1
13 x
14 STO 0
15 1
16 STO-1
17 GTO 05
```

**Tempi:**

| n | n! | sec. |
|---|---|---|
| 6! | 720 | 3 |
| 55! | $1{,}27 \cdot 10^{73}$ | 28 |
| 69! | $1{,}71 \cdot 10^{98}$ | 35 |

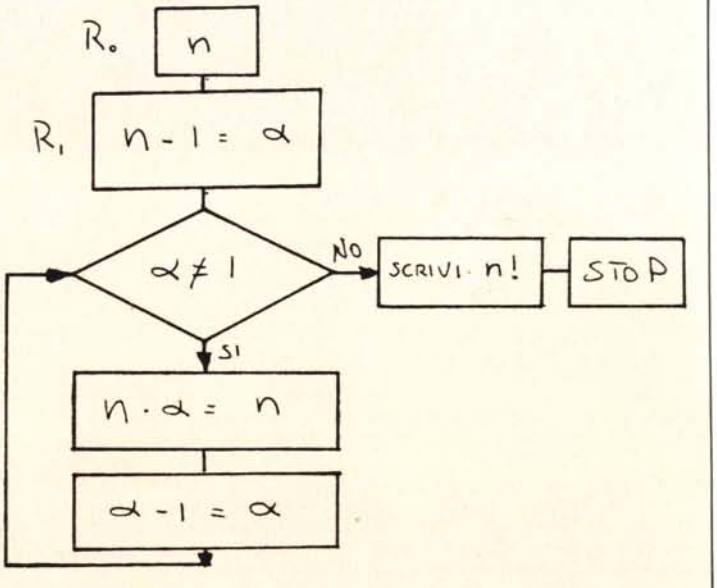



Figura 1

*scarsa importanza pratica, il programma comunque non lo calcola come si può notare dal diagramma di scorrimento.*

*Come si vede anche dal diagramma di flusso, utilizzo due memorie; una conta il numero dei cicli da compiere (α) e l'altra conserva i prodotti. Una volta programmata la macchina, basta impostare il numero n sul visore premere R/S ed attendere.*

*Per utilizzare questo programma come "subroutine" di un altro programma consiglio di togliere completamente il passo 9 e cambiare i passi 8 e 10 scrivendo rispettivamente GTO 10 e RTN. Il nuovo programma avrà allora 16 passi: n! rimarrà in R.*

*Allego i tempi per il calcolo di alcuni valori di n tra cui n=69 che è il massimo valore calcolabile.*

Non è un programma eccezionale, ma con esso sembra che la 33C voglia dire: "Ehi, ci sono anch'io".

Sicuramente sarà apprezzato dai possessori di una 33C i quali lamentano la pubblicazione di programmi RPN che sono utilizzati solo dalla 41C. La routine proposta da Colasanti supplisce alla mancanza della funzione fattoriale sulla HP 33C, e gira senza problemi a parte il fatto che, così com'è, ad ogni elaborazione richiede il riposizionamento del puntatore sul passo Ø1 per iniziare una nuova elaborazione; questo perché il passo "1Ø R/S", su cui si ferma il programma, non è seguito da alcuna istruzione di indirizzamento verso l'inizio della routine, per esempio un "GTO Ø1".

La routine per il calcolo dei fattoriali, anche se il sig. Colasanti ha fatto un buon lavoro, può essere realizzata secondo il listato riportato in figura 2, più breve e di più rapida esecuzione.

**"Calcolo del Fattoriale" (versione modificata)**

| | |
|---|---|
| 1 STO Ø | 7 GTO 1Ø |
| 2 RCL Ø | 8 X |
| 3 1 | 9 GTO Ø2 |
| 4 - | 10 R↓ |
| 5 STO Ø | 11 RTN |
| 6 X = Ø | |

Figura 2

# PROGRAMMAZIONE SINTETICA HP-41

Sul numero 6 di MC abbiamo visto come ottenere dalla 41C una strana funzione: il "Byte Jumper"; sul numero successivo ho dato un esempio della sua possibile utilizzazione quale "strumento" per creare funzioni non comprese tra quelle proprie della 41C, tali erano le due funzioni "STO d" e "RCL d". Questa volta, "Byte Jumper" su una mano e "Byte Table" (numero 2 di MC) sull'altra, vediamo come è possibile "sintetizzare" (da cui la denominazione "synthetic programming") molte funzioni non comprese tra quelle standard della 41C.

Il principio usato per la generazione di tali istruzioni si basa sulla possibilità di modificare quelle standard, separandone i vari byte per mezzo del "Byte Jumper" e operando su di essi.

Per esempio, la linea "STO 97", richiede due byte per essere memorizzata, precisamente 91 per indicare l'operazione "STO" e 61 per determinare l'indirizzo del registro al quale è diretta l'operazione "STO", ciò è facilmente verificabile sulla byte table.

Attenzione! Il byte 61 indica l'indirizzo "97" ma può anche indicare la funzione "ABS" o il carattere alpha "a". Come fa la 41C a capire che 61 è un indirizzo e non un "ABS" o una "a"? Come già dissi sul N° 2 di MC, la 41C in questi casi "va a vedere" il byte precedente, nel nostro caso è uno "STO", per cui la macchina sa di dover interpretare quel byte come un indirizzo; se il byte precedente fosse stato una funzione a sé stante o parte di un'altra istruzione precedente, senza dubbio 61 sarebbe stato un "ABS", infine la 41 sa di dover leggere quel byte come una "a" qualora fosse stato compreso tra gli n byte seguenti un byte Fn (TEXTn).

Tornando al nostro "STO 97", se volessimo sostituire l'indirizzo "97" con un altro, per esempio "d" dovremmo compiere le seguenti operazioni:

1) Ingannare la 41C, e fargli "credere" che il byte 61 sia una istruzione a sé, cioè "ABS".
2) Sostituire l'istruzione "ABS" con un "AVIE" (byte 7E, corrispondente anche all'indirizzo "d").
3) lasciare che di nuovo la macchina interpreti il byte come un indirizzo.

A questo punto abbiamo creato l'istruzione "STO d", operazione che del resto abbiamo già visto sul N° 7. Con analogo sistema è possibile sostituire un carattere in una stringa alpha, e vedremo fra poco come. Ingannare la 41C è possibile proprio grazie al "Byte Jumper" e a una istruzione "STO NN" che chiameremo "controllo del salto"; senza dilungarmi in spiegazioni teoriche do subito un esempio generico di "Byte Jumping" su una stringa.

Supponiamo di avere la stringa "ABCD", essa risulta composta dai seguenti byte F4-41-42-43-44; per accedere ai singoli byte faremo uso del byte jumper, ma innanzitutto dobbiamo inserire prima del passo "ABCD" il passo "STO NN" dove NN indica il byte al quale si desidera saltare, indica il byte immediatamente successivo al "controllo di salto" STO NN, ma tale istruzione non ha senso poiché, trovando un F4, la 41C va subito a leggersi i successivi 4 byte e quindi non siamo riusciti a ingannarla; con 1, 2, 3 (e così via) invece la 41C va a leggere rispettivamente il primo, il secondo o il terzo byte dopo F4 ed è così che avremo possibilità di poter esaminare i singoli byte, semplicemente facendo uso del tasto SST.

Vediamo allora di esaminare uno per uno i byte della stringa "ABCD":

1) Impostare il "Byte Jumper" (come indicato sul numero 6 di MC) nel nostro caso tale funzione (XROM Ø5, Ø1) era stata assegnata al tasto "ENTER↑"
2) Impostare prima del passo "ABCD" una istruzione Ø1 STO Ø1 (controllo del salto) e compattare il tutto con un "PACK" per evitare che rimangano byte vuoti.
3) Posizionare il puntatore sul passo che si vuole scomporre, nel nostro caso Ø2 "ABCD".
4) Portare la macchina nel modo "normal" premere il tasto "ENTER↑" (XROM Ø5, Ø1) e quindi riportarla in PRGM.

A questo punto vedrete un passo 02-. Bene, quella è la lettera "A" della nostra stringa cioè il byte 41, che è stato interpretato dalla calcolatrice come istruzione a sé stante, poiché questa non ha letto il byte F4. Premendo SST apparirà il passo 03 * (B), premendola ancora apparirà il passo 04/(C); potreste premere ancora SST e vedere anche il byte relativo alla lettera "D", ma non fatelo e fermatevi al passo 04/ visualizzato sul display. Come ben sapete, la 41C sposta automaticamente in avanti tutti i passi (e quindi i byte) successivi, quando viene inserito un nuovo passo in un programma; il nostro caso non fa eccezione e quindi, se col passo 04/ visualizzato sul display, inseriamo una istruzione, per esempio "LBL 11", il byte successivo verrà spinto in avanti di un posto dal nuovo byte introdotto ØC. L'operazione compiuta può essere schematizzata come in figura 3.

| prima dell'inserimento di "LBL 11" | dopo l'inserimento di "LBL 11" | |
|---|---|---|
| F4 TEXT 4 | F4 TEXT 4 | stringa |
| 41 — (A) | 41 — (A) | |
| 42 * (B) | 42 * (B) | |
| 43 / (C) | 43 / (C) | |
| 44 X<Y? (D) | ØC LBL 11 (μ) | |
| | 44 X<Y? (D) | |

Figura 3

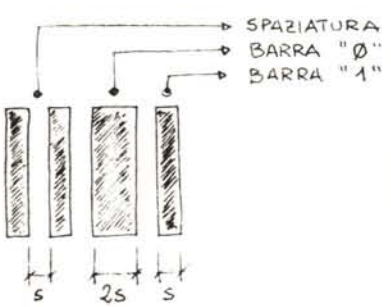
## PER CHI HA IL LETTORE OTTICO...

Un accessorio della 41C, forse poco menzionato in queste pagine, è il lettore ottico di codici a barre 82153A. Con esso è possibile impostare istruzioni anche complesse con una semplice "spazzolata" su un foglio di carta, permettendo così a chi lo usa di risparmiare parecchio tempo. Purtroppo però, il lettore ottico non sa leggere il nostro alfabeto e accetta solo messaggi scritti in codice barre, costringendoci ad usare unicamente dati e programmi stampati dal costruttore o dalla nuova stampante, peraltro assai costosa. Impariamo allora noi a scrivere nel linguaggio usato dal lettore ottico. Sembrerebbe una cosa impossibile, ma non lo è affatto, tant'è vero che attualmente uso con successo il mio lettore ottico insieme ad un pennarello nero; questo grazie al fatto che il lettore è veramente poco sensibile a eventuali imprecisioni di stampa dei codici a barre, e quindi riesce benissimo a leggere codici scritti a mano libera con un minimo di attenzione. La forma usata per codificare le informazioni binarie nella scrittura a barre è la seguente: se chiamiamo S la spaziatura tra due barre, una barra di valore binario 0 deve essere larga S e una barra di valore binario 1 deve essere larga 2S (vedi figura). Il bello è che S non è assolutamente necessario che sia di valore ben definito, il lettore accetta senza problemi valori di S che vanno da mezzo millimetro a tre o quattro millimetri; unica cosa di fondamentale importanza è rispettare entro limiti assai stretti il rapporto tra la larghezza delle barre nere e gli spazi bianchi che ad esse sono vicini, perché per valutare lo spessore di una barra il lettore usa, con termine di paragone, lo spazio bianco ad essa adiacente. Altra cosa assai importante è usare un foglio di carta che sia bianco e scrivere su di esso con un pennarello nerissimo. Con un po' di esercizio, per scrivere a penna una istruzione in codice barre, si impiega poco più che a scriverla in caratteri alfanumerici, basta farci la mano...

Attualmente scrivo le mie istruzioni in codice barre semplicemente tracciando righe verticali singole o doppie (passando due volte il pennarello) e spaziandole a occhio in modo quasi infallibile. Unica limitazione è il fatto che non è facile comporre con il codice a barre, righe contenenti più istruzioni, e bisogna accontentarsi di scriverle una per una, un po' come suggerisce di fare l'HP usando le etichette adesive fornite con il lettore. Il "vocabolario" da usare per ricavare (e magari imparare) i codici delle istruzioni, è la "Wand paper keyboard" fornita insieme al lettore. Ovviamente potrete sembre sbizzarrirvi a cercare nuovi codici per ottenere strane istruzioni, tenendo presente che ogni riga di codice barre deve sempre iniziare con due barre sottili (00) e finire con una barra grande e una sottile (01); le barre comprese tra queste due coppie costituiranno i byte relativi all'istruzione o messaggio, e quindi dovranno essere 8 o 16 (o comunque multipli di 8).

Buon lavoro, e non scoraggiatevi se ai primi tentativi non riuscite, in fondo si tratta di riprendere la mano e tracciare i bastoncelli, come facevamo all'asilo...

*P.G.*




In pratica il byte 44 è stato "spinto fuori" dalla serie di byte interpretati dalla 41C come caratteri ALPHA ed è diventata l'istruzione a sé stante "X<Y?", mentre la nostra stringa alpha è diventata "ABCμ". Avremmo potuto utilizzare qualsiasi carattere tra quelli indicati sulla Byte Table, creando così stringhe di ogni genere. Vediamo invece come creare una istruzione diversa da quelle ottenibili da tastiera (per esempio "RCL d", "RCL a", "TONE d", eccetera). Come esempio proviamo a impostare l'istruzione "TONE e". Innanzi tutto bisogna "preparare" due istruzioni necessarie per la costruzione come STO 1, la seconda è una stringa di due caratteri "AB" che useremo come "banco da lavoro" per costruire il nostro "TONE e". Eseguendo il Byte Jumping sul passo "AB", (posizionare la macchina sul passo "AB", passare in "NORMAL", premere "ENTER" cioè XROM 05,01, e tornare in "PRGM") ci troveremo col puntatore posizionato sul byte corrispondente alla lettera "A" della stringa, visualizzato come "-". In questa posizione, inseriamo una istruzione "TONE 9". Quello che succede è riportato in figura 4.

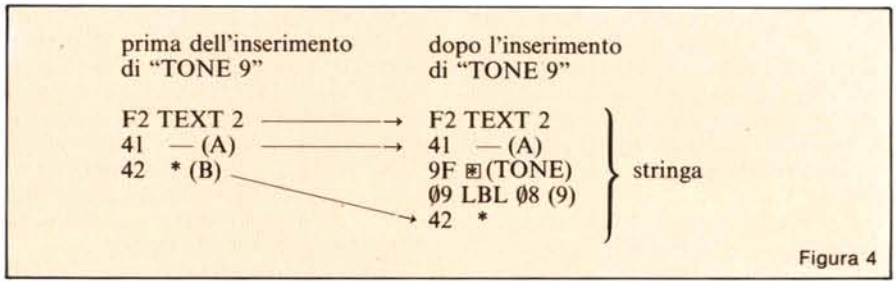



Figura 4

"TONE 9" è una istruzione a due byte; il primo viene interpretato come simbolo alpha trovandosi nel campo indicato da "TEXT 2", il secondo, quello relativo all'indice "9", viene interpretato come "LBL 08" poiché la macchina non si accorge dell'esistenza del byte 9F (TONE) che in questo momento è per lei un carattere alpha. A questo punto sostituiamo l'istruzione "LBL 08" con l'istruzione "CLD" (corrispondente al byte 7F ed anche all'indice "e") operando normalmente, cioè impostando "CLD" con la macchina posizionata sul passo "A"; la nuova situazione è la seguente:

stringa
F2 TEXT 2
41 — (A)
9F ▣ (TONE)

7F CLD (e)
09 LBL 08
42 *

Per finire, bisogna "spingere" fuori dalla stringa il byte 9F, e per fare ciò si deve eseguire di nuovo il Byte Jumping sulla stringa alpha "A*" e inserire una istruzione "*"; questo byte rimette al suo posto la lettera "B" e i byte componenti il programma diventano:

stringa
F2 TEXT 2
41 — (A)
42 * (B)

TONE
9F TONE
7F e
09 LBL 08
42 *

Il programma finale, risultante da tali operazioni è il seguente:

01 STO 01
02 TAB
03 TONE e
04 LBL 08
05

I passi 01 e 02 sono di nuovo quelli usati per la generazione, e se non occorrono per creare altre funzioni possono essere eliminati, i passi 04 e 05 sono "scarti di lavorazione" e vanno cancellati, infine, il passo 03 è il nostro prodotto ultimato "TONE e"; per provarlo, posizionate il puntatore su tale linea, passate in NORMAL e premete SST, il suono che udite è proprio il "TONE e".

Se durante le operazioni commettete qualche errore, un "PACK" non guasta mai.

Questo procedimento può essere usato genericamente per ottenere molte istruzioni procedendo in modo perfettamente analogo; basta impostare anziché il "TONE 9", la funzione desiderata della quale utilizzeremo poi soltanto la parte indicante l'operazione (TONE, STO, RCL, eccetera) e, al posto di "CLD", l'istruzione relativa al byte corrispondente all'indirizzo desiderato. Ovviamente tutto ciò va fatto con le dovute cautele, perché alcune istruzioni potrebbero trarre in inganno; per esempio, mentre uno "STO 25" si presta perfettamente alla sostituzione dell'indirizzo, in quanto funzione a due byte, altrettanto non può dirsi della funzione "STO 07", essendo questa una istruzione ad un solo byte e quindi inscindibile.

Facendo uso della Byte Table, è possibile ottenere, con le tecniche che abbiamo visto, istruzioni assai interessanti per l'accesso agli angoli "segreti" della memoria della 41C. Vedremo poi come usare un buon numero di funzioni sintetiche per incrementare le capacità del nostro "mostro a batterie".

# Scambiando «mele» con «limoni» i programmi non cambiano

proprio così tutti i programmi che voi desiderate, gli stessi che «girano» su quello che forse è il più famoso personal americano vanno bene anche per Lemon II, questo significa **compatibilità totale.**
Ma c'è di più, la Selcom Elettronica è un'azienda italiana, che fabbrica in Italia e ciò consente prezzi ancora più competitivi e una reale assistenza tecnica, data dal produttore medesimo.
Un'assistenza che viene data anche a coloro, e sono molti, che desiderano montare da soli l'insieme attraverso poche, chiare e semplici, istruzioni realizzando risparmi ancora più consistenti.

| | |
|---|---|
| Microprocessore | 6502 |
| Memoria RAM | 48 K |
| Memoria EP ROMS | 12 K |

PERIFERICHE OPZIONALI
- Unità disco mobile 5" - 8"
- Stampanti seriali - parallele
- Plotter - Tavola grafica

- Bus di espansione periferiche
- I/O analogici/digitali
- Uscita video compatibile

Linguaggio residente BASIC ESTESO Opzionale - Fortan - Pascal - Cobol - Possibilità grafica a colori e sintesi musicale.

Disponibilità di SOFTWARE indirizzato
- Didattico
- Applicativo
- Hobbystico
- Scientifico
- Gestionale
- Statistico

**LEMON II**

e fabbricato dalla SELCOM elettronica
via lametta, 9 - 48100 ravenna - tel. 0544-35365