LRN

Programming my TI – Programmer in LMS
TI-58 / TI-58C / TI-59
Pierre Houbert

LRN

Programmare in Soa

TI-58 / TI-58C / TI-59

Claudio Larini

collection
savoir faire

Versione italiana di Claudio Larini – 7 Giugno 2023

Introduzione

Le calcolatrici programmabili Texas Instruments TI-58 e TI-59 sono apparse nel 1977, seguite nel 1979 dalla TI-58C.
Basate sul sistema A.O.S. (notazione algebrica diretta), erano programmabili utilizzando un linguaggio specifico chiamato, in Francese, LMS (*langage machine spécialisé* - linguaggio macchina specializzato).

Alcuni utenti hanno visto in queste macchine più il lato "calcolatrice scientifica" (o matematica) per via delle loro numerose funzioni matematiche e statistiche, altri hanno adottato queste calcolatrici come "computer tascabili" e addirittura inventato, in quegli anni di nascente microinformatica, il termine "pico-informatica".

Ci avvicineremo qui al lato “calcolatrice programmabile” e cercheremo di scoprire questo linguaggio, apparentemente rudimentale e semplice, che però ha saputo affascinare molti adepti.

In effetti, questo linguaggio si è rivelato davvero attraente perché sufficientemente completo per sviluppare programmi complessi.

La gamma delle possibili applicazioni ha persino permesso un uso professionale in alcune aree.

I moduli di programma commercializzati coprivano matematica, navigazione, ingegneria elettrica, agricoltura, investimenti finanziari, gestione dell'inventario e molte altre attività, per non parlare dei giochi.

Gli unici limiti erano dovuti ai vincoli fisici di queste macchine: nessun display alfanumerico (ma stampa

di testi su carta), dimensioni della memoria centrale, supporto di "memoria di massa" (schede magnetiche solo per **TI-59**).

Allora perché usare oggi, nell'era di "smartphone" e "tablet", queste macchine ancestrali e questo linguaggio d'altri tempi?

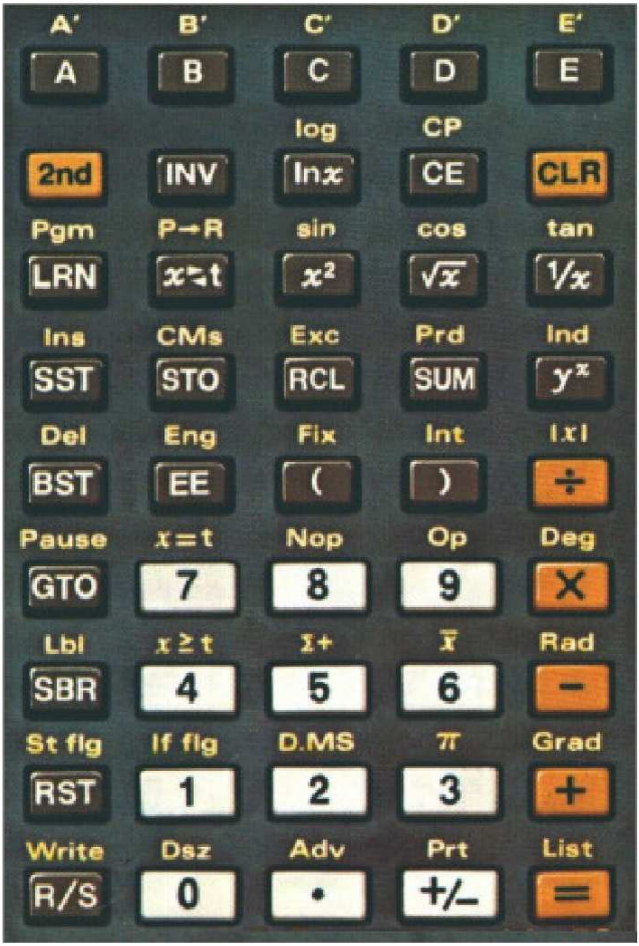
Per lo stesso motivo per cui nell'era delle navette spaziali, dei treni ad alta velocità e degli altri veicoli veloci, i nostri figli, i nostri nipoti continuano a voler imparare a guidare un velocipede: per divertimento!

Oggi gli emulatori delle TI esistono su vari sistemi operativi (MS-DOS, Windows, Android, Pocket PC) e permettono di riscoprire questo particolare piacere di programmare con un linguaggio del genere.

Il primo programma

Primi passi

Per cominciare, diamo un'occhiata alla tastiera della nostra calcolatrice.



Il primo tasto di cui parleremo è il tasto [2nd].
Ci permetterà di accedere alla "seconda" funzione di un tasto, quindi per ottenere π (pi) dobbiamo usare la seconda funzione del tasto [3].

Quindi la sequenza di tasti [2nd][π] darà 3.14159265359
Per calcolare il perimetro di un cerchio con raggio di 4 cm, bisogna fare 4 x 2 x π = e quindi digitare:

[4][x][2][x][π][=]

Possiamo realizzare un primo programma che permetta di calcola il perimetro di un cerchio quale che sia il suo raggio...

Questo programma assomiglierà a qualcosa del genere:

- Inserire un numero
- Moltiplicare per 2
- Moltiplicare per Pi
- Visualizzare il risultato

L'immissione del numero verrà eseguita sulla tastiera e quindi si dovrà avviare l'esecuzione del programma che si arresterà visualizzando il risultato.

Per avviare il programma (e fermarlo) utilizzeremo il tasto [R/S] (e l'istruzione) che significa *Run/Stop*.

Il nostro programma sarà quindi nella forma:

[x][2][x][π][=][R/S]

Scrittura del programma

Per scrivere un programma, è necessario passare alla modalità programmazione tramite il tasto [LRN] (Learn).

Quando premiamo questo tasto [LRN], il visualizzatore cambia per mostrare due gruppi di numeri separati da uno spazio.

Il primo gruppo, composto da 3 cifre, rappresenta l'indirizzo dell'istruzione (si parlerà piuttosto di un **passo di programma**), ed il secondo gruppo, composto da 2 cifre, rappresenta il codice dell'istruzione.
Ad ogni istruzione visualizzata sulla tastiera corrisponde un codice di due cifre (da 00 a 99) per cui il nostro programma

[x][2][x][2nd][π][=][R/S]

si potrà scrivere

65 02 65 89 95 91

poiché i rispettivi codici sono

- 65 per [x]
- 02 per [2]
- 65 per [x]
- 89 per [2nd][π]
- 95 per [=]
- 91 per [R/S]

Per tornare al modo "calcolatrice" o per uscire dal modo programmazione, premiamo [LRN].

Prima di introdurre il nostro programma, ci assicureremo che nessun altro programma sia in memoria cancellando la memoria di programma con **CP** ottenuto con [2nd][CP] (Clear Program).

Se torniamo in modalità programmazione premendo [LRN], siamo al passo 000 con 00 come codice di istruzione.

Premendo [x], viene visualizzato il passo 001 con 00 come codice di istruzione.

Quindi premere su [2], poi [x], poi [2nd][π], poi [=], poi [R/S].

Proseguendo con l'impostazione, possiamo vedere l'incremento dei passi del programma per il posizionamento sul passo della prossima istruzione da introdurre.

Per controllare il nostro input, abbiamo due soluzioni: o "camminare" nel nostro programma per visualizzare i passaggi successivi, oppure uscire dalla modalità di programmazione (con [LRN]) e stampare il nostro programma.

Facciamo un giro ...

Per controllare il nostro input, possiamo "risalire" nel nostro programma usando [BST].

Ogni pressione su [BST] ci porta un passo indietro e noi vediamo l'indirizzo del passo e il codice dell'istruzione: [BST] visualizza 005 91 poi [BST] 004 95 poi [BST] 003 89 poi [BST] 002 65 poi [BST] 001 02 poi [BST] 000 65.

Possiamo anche "scendere" nel nostro programma con l'aiuto di [SST].

[SST] visualizza 001 02, poi [SST] 002 65, poi [SST] 003 89 poi [SST] 004 95 poi [SST] 005 91.

Premere [LRN] per tornare al modo "calcolatrice".

Abbiamo lasciato la modalità di programmazione mentre il contatore di programma era al passo 005.

Se premiamo di nuovo per tornare al modo programmazione, viene visualizzato il passo 005.

Se provassimo ad avviare l'esecuzione del programma, non accadrebbe nulla perché il contatore di programma è posizionato sull'istruzione di arresto.

Per tornare, in modo "calcolatrice", premere [LRN] quindi [RST] per riportare il contatore di programma al passo 000.

Per controllare il programma, lo stamperemo usando **LST** ([2nd][List])
Sulla stampante otteniamo:

```
000 65  ×
001 02  2
002 65  ×
003 89  π
004 95  =
005 91 R/S
```

La stampa ci fornisce l'indirizzo (passo) dell'istruzione, il suo codice e anche la sua traduzione.

Primo test

Ora possiamo testare il nostro programma.

Bisogna :

- riportare il contatore di programma a **000**: [RST]
- inserire un raggio: per esempio[2][5]
- avviare il programma: [R/S]

e otteniamo 157.0796327

L'usabilità potrebbe essere migliorata evitando di dover utilizzare tasti come [RST] e [R/S].

Infatti la calcolatrice ha dei tasti "funzione" (**A, B, C, D, E**) che potrebbero esserci utili.

Useremo quindi la nozione di "label" (o etichetta).

Per modificare il nostro programma, riportiamo il contatore di programma all'indirizzo 000 con [RST], quindi passiamo al modo programmazione con [LRN].

Siamo al passo 000 davanti al quale inseriremo 2 righe utilizzando **INS** due volte: [2nd][Ins][2nd][Ins]

Ora possiamo mettere la nostra dichiarazione di etichetta con [2nd][Lbl](**LBL**), quindi [A].

Torniamo al modo "calcolatrice" per stampare: [LRN], poi [RST][2nd][List].

sulla stampante otteniamo:

```
000 76  LBL
001 11   A
002 65   x
003 02   2
004 65   ×
005 89   π
006 95   =
007 91  R/S
```

Ora possiamo testare nuovamente il nostro programma.

Bisogna :
- inserire un raggio: ad esempio [2][5]
- avviare il programma: [A]

e otteniamo 157.0796327

Se dopo l'esecuzione premiamo per passare alla modalità programmazione, vediamo che il contatore di programma è posto al passo **008**.

Aggiungeremo una seconda parte che permette il calcolo dell'area del cerchio:

[LBL][B][x^2][x][2nd][π][=][R/S]

cioè: **LBL B X^2 x π = R/S**

quindi per tornare al modo "calcolatrice".

[RST][2nd][List] per stampare.

```
000 76 LBL
001 11  A
002 65  x
003 02  2
004 65  x
005 89  π
006 95  =
007 91 R/S
008 76 LBL
009 12  B
010 33 X^2
011 65  x
012 89  π
013 95  =
014 91 R/S
```

Ora un numero n seguito da [A] mostra il perimetro e un numero n seguito da [B] visualizza l'area di un cerchio.

Ora possiamo modificare questo programma in modo da non inserire il raggio che solo una volta ed eseguendo i nostri due calcoli in successione durante digitando:

raggio [A] [B]

Per fare ciò, dovremo mettere il raggio in memoria nella procedura [A] e richiamare l'informazione memorizzata nella procedura [B].

Archiviazione in memoria

La TI contiene diverse "zone" di archiviazione per conservare i dati utilizzati. Queste aree sono chiamate ***Registri***.

Il primo registro è quello che corrisponde al visualizzatore numerico: questo è il registro "x".
Il secondo registro è il registro di prova denominato "t".

Il comando [x⇆t] consente, come suggerisce il nome "scambia x con t", di scambiare i valori di **x** e **t**.

Esempio:
123 [x⇆t] 456 pone il valore 123 in t e 456 in x
[x⇆t] scambio: 123 in x e 456 in t

Altri registri sono utilizzati per la memorizzazione, sono numerati da 00 a 99 (1).

Per gestire questi registri sono utilizzabili differenti istruzioni:

- [STO] nn copia registro x nel registro nn
- [RCL] nn copia registro nn nel registro x
- [SUM] nn aggiunge il registro x al registro nn
- [INV][SUM] nn sottrae il registro x dal registro nn
- [2nd][Prd] nn moltiplica registro nn per registro x
- [INV][2nd][Prd] nn divide registro nn per registro x
- [2nd][Exc] nn scambia il registro nn con il registro x

(1) Differisce in base al modello di TI e alle opzioni selezionate - Vedi OP 16 e OP 17

Per il nostro programma "Cerchio", memorizzeremo il raggio nel registro 01 per poi recuperarlo.

Dietro [2nd][Lbl][A] inseriremo [STO][0][1].

Inserendo l'indirizzo del registro (01), il visualizzatore non avanza di due passi: infatti, dopo [STO], sono attesi 2 caratteri che però occupano un solo passo di programma.

Dopo [2nd][Lbl][B] inseriamo [RCL][0][1].

Otteniamo :

```
000 76 LBL
001 11  A
002 42 STO
003 01  01
004 65  ×
005 02  2
006 65  x
007 89  π
008 95  =
009 91 R/S
010 76 LBL
011 12  B
012 43 RCL
013 01  01
014 33 X^2
015 65  ×
016 89  π
017 95  =
018 91 R/S
```

Per testare, inseriamo semplicemente il raggio, premiamo su [A] per ottenere il perimetro e poi [B] per ottenere l'area.

Se in modalità "calcolatrice" premiamo [RCL][0][1], viene visualizzato il raggio.

Stampa

Useremo la stampante per migliorare la presentazione dei risultati.
Per l'utilizzo della stampante, abbiamo già visto LST (2nd List) che permette di elencare un programma.
Possiamo anche usare:

- [INV][2nd][List] per stampare il contenuto dei registri (**INV LST**)
- [2nd][Prt] per stampare registro x (**PRT**)
- [2nd][Adv] per avanzare di una riga (**ADV**)

Inoltre, tramite l'istruzione **OP** ([2nd][Op]) è possibile accedere a funzioni "speciali":

- **OP 01, OP 02, OP 03, OP 04** e **OP 05** consentono di stampare testo alfanumerico fino a 20 caratteri, una riga di stampa è larga 20 caratteri.
- **OP 06** stampa il registro x seguito da 4 caratteri alfanumerici
- **OP 07** stampa una curva con il carattere "✶"
- **OP 08** stampa l'elenco delle etichette utilizzate dal programma in memoria.

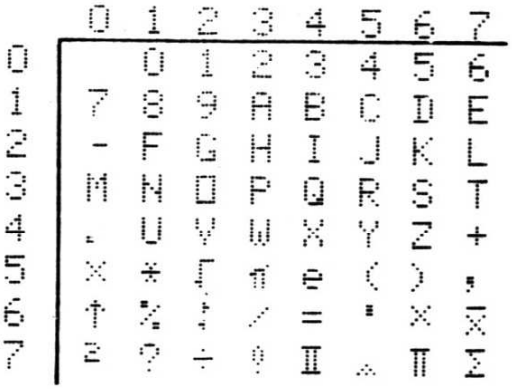
La stampa di un testo alfanumerico avviene su una riga di 20 caratteri divisi in 4 gruppi di 5 caratteri.

- **OP 01** interessa il gruppo 1 (esterno sinistro)
- **OP 02** interessa il gruppo 2 (all'interno a sinistra)
- **OP 03** interessa il gruppo 3 (interno a destra)
- **OP 04** interessa il gruppo 4 (esterno destro)

```
| 1  2  3  4  5| 6  7  8  9 10|11 12 13 14 15|16 17 18 19 20|
|      OP 0 1  |      OP 02   |      OP 03   |      OP 04   |
```

- **OP 05** stampa la riga
- **OP 00** cancella il contenuto dei 4 gruppi (zero)

Per assegnare valori ai gruppi, la TI utilizza una tabella di corrispondenza dei caratteri:



Quindi, il carattere "**A**" si ottiene con il codice **13**, il carattere "**=**" con il codice **64**...

Quindi per stampare:

RAYON =

bisogna scrivere:

OP 00	Cancella il buffer
3 5	R (carattere 1)
1 3	A (carattere 2)
4 5	Y (carattere 3)
3 2	O (carattere 4)
3 1	N (carattere 5)
OP 01	vanno nel gruppo 1
0 0	Spazio (carattere 6)
6 4	= (carattere 7)
0 0	Spazio (carattere 8)
0 0	Spazio (carattere 9)
0 0	Spazio (carattere 10)
OP 02	vanno nel gruppo 2
OP 05	stampa la linea

Possiamo anche stampare un testo di soli 4 caratteri dopo al numero presente sul visualizzatore (registro x) utilizzando **OP 04** (gruppo 4) e **OP 06**.

Per stampare:

12 cm²

bisogna scrivere:

OP 00
1 5
30
7 0
0 0
OP 04
12
OP 06

Programma completo

000 76 LBL	037 42 STO	074 00 00	111 00 0
001 11 A	038 03 03	075 69 OP	112 00 0
002 42 STO	039 71 SBR	076 02 02	113 00 0
003 01 01	040 30 TAN	077 69 OP	114 00 0
004 32 X:T	041 43 RCL	078 05 05	115 00 0
005 01 01	042 03 03	079 92 RTN	116 00 0
006 32 X:T	043 71 SBR	080 76 LBL	117 00 0
007 22 INV	044 28 LOG	081 38 SIN	118 69 OP
008 77 GE	045 71 SBR	082 69 OP	119 03 03
009 96 WRT	046 23 LNX	083 00 00	120 69 OP
010 71 SBR	047 25 CLR	084 03 3	121 05 05
011 23 LNX	048 91 R/S	085 03 3	122 92 RTN
012 71 SBR	049 76 LBL	086 01 1	123 76 LBL
013 39 COS	050 39 COS	087 07 7	124 30 TAN
014 43 RCL	051 69 OP	088 03 3	125 69 OP
015 01 01	052 00 00	089 05 5	126 00 00
016 71 SBR	053 03 3	090 02 2	127 03 3
017 28 LOG	054 05 5	091 04 4	128 06 6
018 65 ×	055 01 1	092 03 3	129 04 4
019 02 2	056 03 3	093 00 0	130 01 1
020 65 ×	057 04 4	094 69 OP	131 03 3
021 89 π	058 05 5	095 01 01	132 05 5
022 95 =	059 03 3	096 01 1	133 02 2
023 42 STO	060 02 2	097 07 7	134 01 1
024 02 02	061 03 3	098 03 3	135 01 1
025 71 SBR	062 01 1	099 07 7	136 03 3
026 38 SIN	063 69 OP	100 03 3	137 69 OP
027 43 RCL	064 01 01	101 05 5	138 01 01
028 02 02	065 00 0	102 01 1	139 01 1
029 71 SBR	066 00 0	103 07 7	140 05 5
030 28 LOG	067 06 6	104 00 0	141 01 1
031 43 RCL	068 04 4	105 00 0	142 07 7
032 01 01	069 00 0	106 69 OP	143 00 0
033 33 X^2	070 00 0	107 02 02	144 00 0
034 65 ×	071 00 0	108 06 6	145 06 6
035 89 π	072 00 0	109 04 4	146 04 4
036 95 =	073 00 0	110 00 0	147 00 0

```
| 148 00  0   | 179 02  2   | 210 02  2   | 241 69 OP   |
| 149 69 OP   | 180 06 06   | 211 99 PRT  | 242 02  02  |
| 150 02  02  | 181 04 04   | 212 22 INV  | 243 00  0   |
| 151 69 OP   | 182 06 06   | 213 58 FIX  | 244 00  0   |
| 152 05  05  | 183 04 04   | 214 92 RTN  | 245 03  3   |
| 153 92 RTN  | 184 06 06   | 215 76 LBL  | 246 01  1   |
| 154 76 LBL  | 185 04 04   | 216 96 WRT  | 247 03  3   |
| 155 23 LNX  | 186 06 06   | 217 69 OP   | 248 02  2   |
| 156 06  6   | 187 04 04   | 218 00  00  | 249 03  3   |
| 157 04  4   | 188 06 06   | 219 00  0   | 250 00  0   |
| 158 06  6   | 189 04 04   | 220 00  0   | 251 01  1   |
| 159 04  4   | 190 69 OP   | 221 03  3   | 252 04  4   |
| 160 06  6   | 191 03  03  | 222 06  6   | 253 69 OP   |
| 161 04  4   | 192 06  6   | 223 01  1   | 254 03  03  |
| 162 06  6   | 193 04  4   | 224 03  3   | 255 03  3   |
| 163 04  4   | 194 06  6   | 225 02  2   | 256 05  5   |
| 164 06  6   | 195 04  4   | 226 04  4   | 257 01  1   |
| 165 04  4   | 196 06  6   | 227 03  3   | 258 07  7   |
| 166 69 OP   | 197 04  4   | 228 06  6   | 259 00  0   |
| 167 01  01  | 198 06  6   | 229 69 OP   | 260 00  0   |
| 168 06  6   | 199 04  4   | 230 01  01  | 261 07  7   |
| 169 04  4   | 200 06  6   | 231 02  2   | 262 03  3   |
| 170 06  6   | 201 04  4   | 232 04  4   | 263 00  0   |
| 171 04  4   | 202 69 OP   | 233 03  3   | 264 00  0   |
| 172 06  6   | 203 04  04  | 234 05  5   | 265 69 OP   |
| 173 04  4   | 204 69 OP   | 235 00  0   | 266 04  04  |
| 174 06  6   | 205 05  05  | 236 00  0   | 267 69 OP   |
| 175 04  4   | 206 92 RTN  | 237 04  4   | 268 05  05  |
| 176 06  6   | 207 76 LBL  | 238 01  1   | 269 25 CLR  |
| 177 04  4   | 208 28 LOG  | 239 03  3   | 270 35 1/X  |
| 178 69 OP   | 209 58 FIX  | 240 01  1   | 271 91 R/S  |
```

Utilizzazione del programma:

raggio [A]

Il risultato viene inviato alla stampante.

Esempio:
Inserisci:[1][5][A]

Risultato:

```
====================
RAYON =
                 15.00
PERIMETRE =
                 94.25
SURFACE =
                706.86
====================
```

In questo programma, vediamo prima che funzioni possono essere usate come etichette (labels):

LBL COS, LBL LNX, LBL WRT ...

e che le etichette possono essere "chiamate" da SBR ([SBR]) con un ritorno dopo la chiamata grazie a **RTN** ([INV] [SBR]), **SBR** e **RTN** significano rispettivamente Subroutine e Return.

Un'altra osservazione, vediamo che la stampa del testo alfanumerico è costosa in termini di "passi" di programma:

- "RAYON ="routine **COS**, passi da 49 a 79 = 31 passi
- "PERIMETRE ="routine **SIN**, passi da 80 a 122 = 43 passi
- "SURFACE ="routine **TAN**, passi da 123 a 153 = 31 passi
- "========...=" routine **LNX**, passi da 154 a 206 = 53 passi
- "SAISIR UN NOMBRE!" Routine **WRT** da 215 a 271 = 57 passi

Sono in tutto 215 passi per un programma di 272 passi!

Il programma include un test che consente di gestire una routine di trattamento errori (LBL WRI) se il valore del raggio inserito è minore di 1.

```
004 32 X:T
005 01  1
006 32 X:T
007 22 INV
008 77 GE
009 96 WRI
```

che si sarebbe potuto scrivere:

```
004 32 X:T
005 00  0
006 77 GE
007 96 WRT
008 32 X:T
```

GE e **INV GE** consentono il salto condizionale secondo un confronto tra i registri x e t, **GE** significa *Greater or Equal* (Maggiore o Uguale).

La soluzione 1 (6 passaggi) inserisce il raggio in t scambiando x e t, inserisce il valore "1" in x, scambia nuovamente x e t per avere "1" in t e il raggio in x quindi verifica se x è strettamente minore (INV GE) a t:

"Il raggio è strettamente inferiore a 1?"

La soluzione 2 (5 passaggi) inserisce il raggio in t scambiando x e t, inserisce il valore "0" in x quindi verifica se x è maggiore o uguale a t:

"Zero è maggiore o uguale al raggio?"

(uno scambio di x e t, dopo il test, riporta il raggio in x per il resto dei calcoli, essendo **RCL 01** più costoso di un passo)

Vanno chiarite altre due particolarità:

1. La routine **LOG** permette la stampa del contenuto del registro x formattandolo con due cifre decimali.

```
207 76 LBL
208 28 LOG
209 58 FIX
210 02  02
211 99 PRT
212 22 INV
213 58 FIX
214 92 RTN
```

FIX 2 fissa il display a due cifre decimali, **PRT** stampa il registra x e **INV FIX** annulla la formattazione.

2. La routine **WRT**, che stampa il messaggio di errore, finisce con:

```
269 25 CLR
270 35 1/X
271 91 R/S
```

CLR 1/X imposta il registro x su zero e divide 1 per x che causa un errore (divisione per zero!) e attiva il lampeggio

del display per segnalare l'errore, **R/S** arresta il programma. (Questo "trucco" viene spesso utilizzato per avvisare l'utente di un errore di immissione.)

Alla luce delle osservazioni precedenti, possiamo considerare di modificare questo programma per migliorarlo, infatti per i calcolatori interessati (**TI-58, TI-58C** e **TI-59**) che hanno un memoria "programma" limitata nel numero di passi, una delle principali preoccupazioni di programmazione è il risparmio di passi, un eccessivo approccio "economico" potrebbe pregiudicare la leggibilità, quindi la manutenibilità, di un programma...

Ecco una versione "ottimizzata" di questo programma:

000 69 OP	037 07 7	074 01 01	111 00 00
001 00 00	038 00 0	075 69 OP	112 69 OP
002 03 3	039 00 0	076 01 01	113 01 01
003 06 6	040 07 7	077 06 6	114 01 1
004 01 1	041 03 3	078 04 4	115 07 7
005 03 3	042 00 0	079 00 0	116 03 3
006 02 2	043 00 0	080 00 0	117 07 7
007 04 4	044 69 OP	081 00 0	118 03 3
008 03 3	045 04 04	082 00 0	119 05 5
009 06 6	046 69 OP	083 00 0	120 01 1
010 69 OP	047 05 05	084 00 0	121 07 7
011 01 01	048 25 CLR	085 69 OP	122 00 0
012 02 2	049 35 1/X	086 02 02	123 00 0
013 04 4	050 91 R/S	087 69 OP	124 69 OP
014 03 3	051 76 LBL	088 05 05	125 02 02
015 05 5	052 11 A	089 43 RCL	126 06 6
016 00 0	053 42 STO	090 01 01	127 04 4
017 00 0	054 01 01	091 71 SBR	128 65 ×
018 04 4	055 32 X:T	092 28 LOG	129 06 6
019 01 1	056 00 0	093 65 *	130 22 INV
020 03 3	057 77 GE	094 02 2	131 28 LOG
021 01 1	058 00 00	095 65 ×	132 95 =
022 69 OP	059 00 00	096 89 π	133 69 OP
023 02 02	060 32 X:T	097 95 =	134 03 03
024 03 3	061 71 SBR	098 42 STO	135 69 OP
025 01 1	062 23 LNX	099 02 02	136 05 05
026 03 3	063 69 OP	100 69 OP	137 43 RCL
027 02 2	064 00 00	101 00 00	138 02 02
028 03 3	065 03 3	102 03 3	139 71 SBR
029 00 0	066 05 5	103 03 3	140 28 LOG
030 01 1	067 01 1	104 01 1	141 43 RCL
031 04 4	068 03 3	105 07 7	142 01 01
032 69 OP	069 04 4	106 03 3	143 33 X^2
033 03 03	070 05 5	107 05 5	144 65 ×
034 03 3	071 03 3	108 02 2	145 89 π
035 05 5	072 02 2	109 04 4	146 95 =
036 01 1	073 03 3	110 03 3	147 42 STO

```
| 148 03  03 | 165 01  1  | 182 23 LNX | 199 69 OP  |
| 149 69 OP  | 166 07  7  | 183 25 CLR | 200 02  02 |
| 150 00  00 | 167 00  0  | 184 91 R/S | 201 69 OP  |
| 151 03  3  | 168 00  0  | 185 76 LBL | 202 03  03 |
| 152 06  6  | 169 06  6  | 186 23 LNX | 203 69 OP  |
| 153 04  4  | 170 04  4  | 187 06  6  | 204 04  04 |
| 154 01  1  | 171 00  0  | 188 04  4  | 205 69 OP  |
| 155 03  3  | 172 00  0  | 189 06  6  | 206 05  05 |
| 156 05  5  | 173 69 OP  | 190 04  4  | 207 92 RTN |
| 157 02  2  | 174 02  02 | 191 06  6  | 208 76 LBL |
| 158 01  1  | 175 69 OP  | 192 04  4  | 209 28 LOG |
| 159 01  1  | 176 05  05 | 193 06  6  | 210 58 FIX |
| 160 03  3  | 177 43 RCL | 194 04  4  | 211 02 02  |
| 161 69 OP  | 178 03  03 | 195 06  6  | 212 99 PRT |
| 162 01  01 | 179 71 SBR | 196 04  4  | 213 22 INV |
| 163 01  1  | 180 28 LOG | 197 69 OP  | 214 58 FIX |
| 164 05  5  | 181 71 SBR | 198 01  01 | 215 92 RTN |
```

216 passi di programma anziché 272, sono un risparmio di 56 passi!

Il linguaggio

Ora possiamo avvicinarci ai "verbi" suddivisi per argomento per rendere la panoramica la più completa possibile:

- Programmazione
- Tasti aggiuntivi
- Inserimento dati
- Operazioni aritmetiche
- Cancellazione
- Radici e potenze
- Funzioni matematiche
- Trigonometria
- Stampa
- Opzioni di visualizzazione
- Gestione delle memorie
- Istruzioni di salto
- Statistiche
- Tasti funzione
- Lettura / Scrittura di schede magnetiche
- Moduli di libreria
- Operazioni speciali
- Altre funzioni
- L'istruzione nascosta

Programmazione

● **CP** ([2nd][CP]) in modalità "calcolatrice", cancella tutta la memoria di programma (mette a zero tutti i passi), cancella gli indirizzi di ritorno delle subroutine, riposiziona il contatore di programma al passo 000 e cancella il registro t.

● **LRN** ([LRN]) viene utilizzato per accedere al modo "programmazione" o uscirne (tornando al modo "calcolatrice").

● **SST** ([SST]) in modo "programmazione", avanza di un passo.

● **BST** ([BST]) in modo "programmazione", torna indietro di un passo.

● **INS** ([2nd][INS]) in modo "programmazione", inserisce un passo prima del passo corrente.

● **DEL** ([2nd][Del]) in modalità "programmazione", cancella il passo corrente.

In modo programmazione, la pressione di un tasto sostituisce l'istruzione del passo corrente.

Tasti aggiuntivi

● **2nd** ([2nd]) consente di utilizzare la seconda funzione di un tasto corrispondente all'istruzione sopra il tasto stesso.
Esempio: [2nd][Lbl]dà **LBL**

● **INV**([INV]) per alcune funzioni (**EE, ENG, FIX, LOG, LNX, Y^X, INT, SIN, COS, TAN, PRD, SUM, DMS, P/R, STA, AVG, LST, SBR, EQ, GE, IFF, STF, DSZ, WRT**), attivano la funzione inversa.

In alcuni casi, i due tasti [2nd] e [INV] possono essere entrambi usati.
Esempio: il logaritmo decimale si ottiene facendo [2nd] [lnx] e l'antilog del logaritmo decimale si ottiene facendo [INV][2nd][lnx], che scriveremo rispettivamente **LOG** e **INV LOG**.

Il modo "calcolatrice" ci consente di digitare anche [2nd] [INV][lnx] piuttosto che[INV][2nd][lnx], il modo programmazione non ammette che questa seconda notazione ([INV] prima di [2nd]): sconsigliamo di prendere l'abitudine di usare l'opposto ([2nd] davanti a [INV]).

● **IND** ([2nd][Ind]) consente l'indirizzamento indiretto delle istruzioni di gestione dei registri, delle istruzioni di salto e di qualche altra istruzione specifica.

Sono interessati da questo uso:

● Le istruzioni di gestione per i registri **STO, RCL, EXC, SUM, INV SUM, PRD, INV PRD**
● Le istruzioni di salto **GTO, SBR, EQ, INV EQ,GE, INV GE, DSZ, INV DSZ, IFF, INV IFF**
● Le istruzioni specifiche **PGM, OP, FIX, STF**.

L'indirizzamento indiretto consente di utilizzare il contenuto di un registro come contenitore dell'indirizzo da utilizzare.

Esempio:
[5][STO][0][1] inserisce il valore 5 nel registro 01,
[5][STO][2nd][Ind][0][1] inserisce il valore 5 nel registro il cui l'indirizzo è nel registro 01. (Se il registro 01 contiene 4, il valore 5 sarà memorizzato nel registro 04)

Le istruzioni **DSZ** e **IFF** possono utilizzare il doppio indirizzamento indiretto perché gestiscono sia un registro che un indirizzo di salto.

[INV][IFF][IND][0][1][IND][0][2] significa che se il segnalatore, il cui numero è contenuto nel registro 01, si abbassa (flag = 0) il programma andrà all'indirizzo specificato nel registro 02.

La scrittura di istruzioni con indirizzamento indiretto può essere diversa da "istruzione" seguito da **IND** secondo la tabella seguente:

Sequenze di tasti	Istruzioni	Codici
[STO][2nd][Ind]	ST*	72
[RCL][2nd][Ind]	RC*	73
[2nd][RCL][2nd][Ind]	EX*	63
[SUM][2nd][Ind]	SM*	74
[INV][SUM][2nd][Ind]	INV SM*	22 74
[2nd][SUM][2nd][Ind]	PD*	64
[INV][2nd][SUM][2nd][Ind]	INV PD*	22 64
[GTO][2nd][Ind]	GT*	83
[SBR][2nd][Ind]	SBR IND	71 40
[2nd][x=t][2nd][Ind]	EQ IND	67 40
[INV][2nd][x=t][2nd][Ind]	INV EQ IND	22 67 40
[2nd][x≥t][2nd][Ind]	GE IND	77 40
[INV][2nd][x≥t][2nd][Ind]	INV GE IND	22 70 40
[2nd][Dsz][2nd][Ind]	DSZ IND	97 40
[INV][2nd][Dsz][2nd][Ind]	INV DSZ IND	22 97 40
[2nd][Iff][2nd][Ind]	IFF IND	87 40
[INV][2nd][Iff][2nd][Ind]	INV IFF IND	22 87 40
[2nd][LRN][2nd][Ind]	PG*	62
[2nd][Op][2nd][Ind]	OP*	84
[2nd][Fix][2nd][Ind]	FIX IND	58 40
[2nd][RST][2nd][Ind]	STF IND	86 40
[INV][2nd][RST][2nd][Ind]	INV STF IND	22 86 40

Inserimento dati

● **Cifre**([0][1][2][3][4][5][6][7][8][9]) introduzione delle cifre nel registro del visualizzatore x.

● **Punto** ([●]) introduzione del punto decimale.

● **Segno** ([+/-]) cambia il segno del registro del visualizzatore x.

● **PI** ([2nd][π]) introduce 3.14159265359 nel registro del visualizzatore x.

● **|X|** ([2nd][|x|) restituisce il valore assoluto del registro del visualizzatore x.

● **OP 10** ([2nd][Op][1][0]) indica il segno del valore del registro del visualizzatore x, restituendo 1 se x> 0, 0 se x = 0, -1 se x <0

● **INT**([2nd][Int]) restituisce la parte intera del registro x.

● **INV INT**([INV][2nd][Int]) restituisce la parte decimale del registro x.

Le Operazioni aritmetiche

- / ([÷]) divisione.
- * ([x]) moltiplicazione.
- - ([-]) sottrazione.
- + ([+]) aggiunta.
- = ([=]) visualizza e "congela" il risultato.
- (([( ]) parentesi aperta.
-) ([ )]) parentesi chiusa.

Le calcolatrici TI-58/TI-58C/TI-59 usano la notazione algebrica diretta (sistema **AOS**).

Le operazioni seguono quindi la regola della priorità degli operatori.

2 + 3 * 4 = darà come risultato 14 come pure 2 + (3 * 4) =, le parentesi non sono, in questo caso, necessarie.

D'altra parte (2 + 3) * 4 = darà come risultato 20.

È possibile utilizzare diversi livelli di parentesi:

2 + 3 * 4/5 = darà 2.8
((2 + 3) * 4) / 5 = darà 4

Cancellazione

● **CE**([CE]) cancella la voce corrente senza interferire con le operazioni in corso e interrompe il lampeggio del visualizzatore.

● **CLR** ([CLR]) cancella il registro x e tutti i calcoli in corso. Inoltre interrompe il lampeggio del visualizzatore.

● **CMS** ([2nd][CMs]) cancella tutti i registri di dati secondo la partizione definita (vedere **OP 16** e **OP 17**)

● **CP** ([2nd][CP]) in modo programmazione, cancella solo il registro t.

Radici e potenze

● **X^2** ([x^2]) eleva al quadrato il contenuto del registro del visualizzatore x.

● **SQR** ([$\sqrt{x}$]) restituisce la radice quadrata del registro del visualizzatore x. (se il registro x contiene un valore negativo provoca il lampeggio del visualizzatore)

● **Y^X** ([y^x]) eleva il numero nel registro del visualizzatore alla potenza immessa: [5][y^x][9][=] darà 1953125

● **INV Y^X** ([INV][y^x]) calcola la radice x-esima del numero contenuto nel registro del visualizzatore: [5][INV][y^x] [9][=] darà 5

Funzioni matematiche

● **1/X**([1/x]) calcola l'inverso del contenuto del registro del visualizzatore X.

● **LNX**([lnx]) calcola il logaritmo naturale (base e) del registro del visualizzatore x. (x<0 fa lampeggiare il visualizzatore)

● **INV LNX**([INV][lnx]) calcola l'esponenziale (e^x) del registro del visualizzatore x.

● **LOG**([2nd][log]) calcola il logaritmo decimale (base 10) del visualizzare il registro x. (x <0 fa lampeggiare il visualizzatore)

● **INV LOG**([INV][2nd][log]) calcola l'antilogaritmo del registro del visualizzatore x. (elevamento di 10 alla potenza x)
Spesso utilizzato nei programmi per moltiplicare per un
multiplo di 10 maggiore di 100:

RCL 01 * 1 0 0 0 0 0 = costa 10 passi
RCL 01 * 5 INV LOG = costa 7 passi!

● **P/R**([2nd][P/R]) converte le coordinate polari in coordinate cartesiane dai registri x (θ - angolo) e t (R -raggio) e restituisce l'ordinata (y) nel registro x e l'ascissa (x) nel registro t.

Esempio:
10 x:t inserisce il raggio R nel registro t
35 P/R inserisce l'angolo θ nel registro x e restituisce l'ordinata 5.73576436351
x:t restituisce l'ascissa 8.19152044289

● **INV P/R**([INV][2nd][P/R]) converte le coordinate cartesiane in coordinate polari a partire dall'ordinata (y) nel registro x e ascissa (x) nel registro t, restituisce l'angolo θ nel registro x e il raggio R nel registro t.

Bisogna fare attenzione a scegliere il modo angolare DEG, RAD o GRD prima di procedere con il calcolo.

Il modo angolare definisce i limiti dell'angolo:

Modo angolare	Limite inferiore	Limite Superiore
DEG	-90°	270°
RAD	-π/2	3π/2
GRD	-100^g	300^g

Trigonometria

- **DEG** ([2nd][Deg]) selezione della modalità angolare "gradi sessagesimali".

- **RAD** ([2nd][Rad]) selezione della modalità angolare "radianti".

- **GRD** ([2nd][Grad]) selezione della modalità angolare "gradi centesimali".

- **SIN**([2nd][sin]) seno del contenuto del registro del visualizzatore x.

- **INV SIN** ([INV][2nd][sin]) Arcoseno del contenuto del registro del visualizzatore x.

- **COS**([2nd][cos]) coseno del contenuto del registro del visualizzatore x.

- **INV COS** ([INV][2nd][cos]) Arcocoseno del contenuto del registro del visualizzatore x.

- **TAN** ([2nd][tan]) tangente del contenuto del registro del visualizzatore x.

- **INV TAN** ([INV][2nd][tan]) Arco tangente del contenuto del registro del visualizzatore x.

Arco cosecante = **1/X INV SIN**
Arco secante = **1/X INV COS**
Arco cotangente = **1/X INV TAN**

● **DMS** ([2nd][D.MS]) converte un angolo misurato in gradi, minuti, secondi in gradi decimali.
Il formato di input è DD.MMSSsss, il punto decimale deve separare i gradi dai minuti.

● **INV DMS** ([INV][2nd][D.MS]) converte un angolo misurato in gradi decimali in gradi, minuti, secondi.

Stampa

- **ADV** ([2nd][Adv]) fa avanzare la carta di una riga.

- **PRT** ([2nd][Prt]) stampa il registro x.

- **LST** ([2nd][List])lista il programma.

- **INV LST** ([INV][2nd][List]) stampa il contenuto dei registri dal registro nn all'ultimo disponibile, essendo nn il valore del registro x.

- **OP 00** ([2nd][Op][0][0]) cancella il buffer di stampa alfanumerico.

```
| 1  2  3  4  5| 6  7  8  9 10|11 12 13 14 15|16 17 18 19 20|
|      OP 01   |      OP 02   |      OP 03   |      OP 04   |
```

- **OP 01** ([2nd][Op][0][1]) interessa il gruppo 1 (estrema sinistra) del buffer di stampa alfanumerico.

- **OP 02** ([2nd][Op][0][2]) interessa il gruppo 2 (interno sinistro) del buffer di stampa alfanumerico.

- **OP 03** ([2nd][Op][0][3]) interessa il gruppo 3 (interno destro) del buffer di stampa alfanumerico.

● **OP 04**([2nd][Op][0][4]) interessa il gruppo 4 (estrema destra) del buffer di stampa alfanumerico.

● **OP 05**([2nd][Op][0][5]) stampa il buffer di stampa.

● **OP 06**([2nd][Op][0][6]) stampa sulla stessa riga il contenuto del registro del visualizzatore x e gli ultimi 4 caratteri del gruppo 4 (esterno destro) del buffer di stampa alfanumerico.

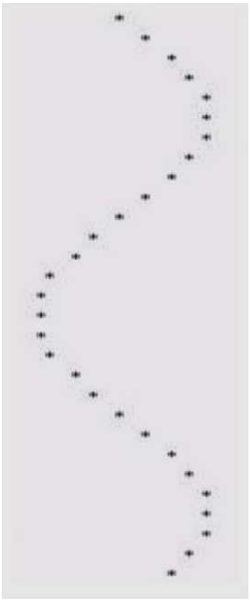
La codifica del buffer di stampa viene mostrata dalla tabella seguente:

● **OP 07**([2nd][Op][0][7]) consente di disegnare una curva stampando un asterisco in una colonna da 0 a 19.

Un singolo asterisco è stampato su ogni riga nella colonna corrispondente alla parte intera del registro di visualizzatore x nell'intervallo di valore -1 < x < 20.

Esempio:
Sinusoide da 18 gradi in 18 gradi.

```
000 76  LBL
001 11   A
002 43  RCL
003 01   01
004 38  SIN
005 85   +
006 01   1
007 95   =
008 65   x
009 09   9
010 93   .
011 09   9
012 95  =
013 69  OP
014 07   07
015 01   1
016 08   8
017 44  SUM
018 01   01
019 61  GTO
020 11   A
```



+1 = * 9.9 = permette di ridurre il valore in un intervallo da 0 a 19.8 per determinare la colonna dell'asterisco.

- **OP 08**([2nd][Op][0][8]) elenco delle etichette (labels) del programma.

```
001 25  CLR
015 15   E
108 11   A
191 12   B
267 35  1/X
322 24  CE
354 45  Y^X
423 33  X^2
```

Opzioni di visualizzazione

Il display TI standard è di 10 cifre mentre la gestione interna è a 13 cifre per una maggiore precisione nei calcoli.
La visualizzazione è quindi limitata a numeri compresi tra .00000000001 e 9999999999 (in valore assoluto, il segno non occupa una posizione sui 10 caratteri).

I numeri che superano questi limiti devono essere inseriti in notazione scientifica.

Quindi il numero

-0.00000000000000000000000000000012345678

si può scrivere

$-1.2345678 * 10^{-31}$

e sarà impostato come segue

1.2345678[+/-][EE]31[+/-]

e verrà visualizzato

-1.2345678 è la **mantissa** e -31 è l'**esponente**.

La mantissa è quindi limitata a 8 caratteri e l'esponente a 2.

● **EE** ([EE]) passa alla notazione scientifica.

● **INV EE** ([INV][EE]) consente la cancellazione della notazione scientifica.

● **ENG** ([2nd][Eng]) consente di passare alla notazione "ingegneristica" [1].
La notazione "ingegneristica" è una variante della notazione scientifica ed è caratterizzata da un aggiustamento della mantissa e dell'esponente al fine di avere un esponente multiplo di tre.
Quindi -1.2345678-31 darà -123.45678-33 in notazione "ingegneristica".

● **INV ENG** ([INV][2nd][Eng]) consente la cancellazione della notazione "ingegneristica".

---

(1) Un termine equivalente per notazione “ingegneristica” è notazione “tecnica”.

La notazione "ingegneristica" permette di rappresentare i numeri nelle consuete unità di misura:

10^n	Prefisso	Numero decimale
10^24	yotta	1 000 000 000 000 000 000 000 000
10^21	zetta	1 000 000 000 000 000 000 000
10^18	exa	1 000 000 000 000 000 000
10^15	peta	1 000 000 000 000 000
10^12	tera	1 000 000 000 000
10^9	giga	1 000 000 000
10^6	mega	1 000 000
10^3	kilo	1 000
10^2	etto	100
10^1	déca	10
10^0	unità	1
10^-1	deci	0,1
10^-2	centi	0,01
10^-3	milli	0,001
10^-6	micro	0,000 001
10^-9	nano	0,000 000 001
10^-12	pico	0,000 000 000 001
10^-15	femto	0,000 000 000 000 001
10^-18	atto	0,000 000 000 000 000 001
10^-21	zepto	0,000 000 000 000 000 000 001
10^-24	yocto	0,000 000 000 000 000 000 000 001

● **FIX** ([2nd][Fix]) **m** permette di scegliere la decimalizzazione. Il numero **m** che segue il tasto FIX indica il numero di posizioni decimali fisso (da 0 a 8).

● **FIX IND** ([2nd][Fix][2nd][Ind]) **nn** consente di scegliere, o annullare, la decimalizzazione indiretta tramute il registro nn.

Il numero che segue il tasto **FIX** indica il numero di registro contenente il numero di cifre decimali fisse (da 0 a 8), o il valore 9 per tornare alla virgola mobile.

● **INV FIX**([INV][2nd][Fix]) annulla la decimalizzazione e ritorna alla virgola mobile. (**FIX 9** ha lo stesso effetto)

Gestione delle memorie

● **X:T**([x⇆t]) scambia il contenuto dei registri x e t.

● **STO** ([STO]) **nn** memorizza il contenuto del registro x nel registro nn.

● **ST*** ([STO][2nd][Ind]) **nn** memorizza il contenuto del registro x nel registro il cui indirizzo è contenuto nel registro nn.

[1][5][STO][2nd][Ind][0][1]
1 5 ST * 01 inserisce il valore 15 nel registro il cui indirizzo è memorizzato nel registro 01.
Se il registro 01 contiene 20, mette 15 nel registro 20,
Se il registro 01 contiene 7 mette 15 nel registro 7 ...

● **RCL** ([RCL]) **nn** inserisce il contenuto del registro nn nel registro x.

● **RC*** ([RCL][2nd][Ind]) **nn** mette il contenuto del registro il cui indirizzo è contenuto nel registro nn nel registro x.

● **SUM** ([SUM]) **nn** aggiunge il contenuto del registro x al contenuto del registro nn.

● **SM*** (([SUM][2nd][Ind])) **nn** aggiunge il contenuto del registro x al contenuto del registro il cui indirizzo è contenuto nel registro nn.

• **INV SUM**([INV][SUM]) **nn** sottrae il contenuto del registro x dal contenuto del registro nn.

• **INV SM*** ([INV][SUM][2nd][Ind]) **nn** sottrae il contenuto del registro x il contenuto del registro, il cui indirizzo è contenuto nella registro nn.

• **PRD**([2nd][Prd]) **nn** moltiplica il contenuto del registro nn per il contenuto del registro x.

• **PD***([2nd][Prd][2nd][Ind])) **nn** moltiplica il contenuto del registro il cui l'indirizzo è contenuto nel registro nn per il contenuto del registro x.

• **INV PRD**([INV][2nd][Prd]) **nn** divide il contenuto del registro nn per il contenuto del registro x.

• **INV PD***([INV][2nd][Prd][2nd][Ind]) **nn** divide il contenuto del registro il cui indirizzo è contenuto nel registro nn dal contenuto del registro x.

• **EXC**([2nd][Exc]) **nn** scambia il contenuto del registro nn con il contenuto del registro x.

● **EX*** ([2nd][Exc][2nd][Ind]) **nn** scambiare il contenuto del registro il cui l'indirizzo è contenuto nel registro nn con il contenuto del registro x.

● **OP 2n**([2nd][Op][2]**n**)incrementa la memoria del registro n di 1. Si applica ai registri da 0 a 9.
OP 21 è equivalente a **1 SUM 01**

● **OP 3n**([2nd][Op][3]**n**)decrementa la memoria del registro n di 1. Si applica ai registri da 0 a 9.
OP 31 è equivalente a **1 INV SUM 01**

Istruzioni di salto

● **LBL**([2nd][Lbl]) consente di definire le etichette del programma(o labels).

È possibile utilizzare 2 tipi di etichette:

- le etichette "utente" (o tasti funzione):[A],[B], [C]...
- etichette ordinarie: tutti i tasti possono quindi essere utilizzati come etichette ad eccezione dei tasti numerici ([0],[1],[2], ...) e dei tasti [2nd], [LRN], [SST],[BST],[2nd][Ins],[2nd][Del],[2nd][Ind] e il tasto [R/S] (autorizzato ma fortemente sconsigliato).

Naturalmente, nel caso di utilizzo di un tasto come etichetta, quest'ultima non sarà trattata come istruzione nel corso del programma ma solo come etichetta.

● **GTO** ([GTO]) consente il salto ad un indirizzo specifico.
Sposta il contatore di programma all'indirizzo specificato e, in modo programmazione, continua l'esecuzione del programma da quell'indirizzo.

Sono possibili due indirizzamenti:

- Indirizzamento logico: GTO è quindi seguito da un nome definito altrove come etichetta.
  Esempio:
  [GTO][x^2]... e altrove nel programma [2nd][Lbl][x^2]...

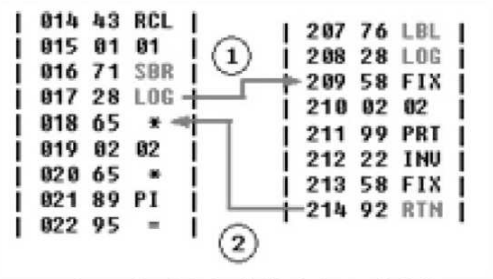
- Indirizzamento assoluto: GTO è quindi seguito da un numero di passo.
  Esempio: [GTO][1][2][3]che si salta al passo 123

Il vantaggio dell'indirizzamento logico risiede nella chiarezza e leggibilità del programma, e l'aggiunta, o la cancellazione, di passi nel programma non modifica il collegamento logico. (Questo metodo costa almeno 4 passi.)
L'indirizzamento assoluto permette una economia di passi (3 passi) ma richiede attenzione per quanto riguarda la manutenzione perché integrazioni, o cancellazioni, di passi nel programma spostano l'indirizzo di riferimento dei GTO se questi aggiornamenti si verificano prima di questo indirizzo.

● **GO***([GTO][2nd][INV]) **nn** consente l'indirizzamento relativo in un programma utilizzando un registro dati che contiene l'indirizzo del passo a cui effettuare il salto.
Esempio: [GTO[[2nd][Ind][0][1]
significa che l'indirizzo del salto è contenuto nel registro 01.

● **SBR**([SBR]) consente il salto all'indirizzo specificato come per GTO ma la prima istruzione di ritorno RTN([INV] [SBR])restituirà il puntatore al passo seguente la chiamata SBR. **SBR** utilizza, come **GTO**, l'indirizzamento logico e l'indirizzamento assoluto.

Esempio:



1) chiamata alla procedura che inizia dall'etichetta LOG,
2) ritorno dopo la chiamata.

● **SBR IND**([SBR][2nd][Ind]) **nn** consente la relativa chiamata in un programma utilizzando un registro dati che contiene l'indirizzo del passo a cui si rivolge la chiamata di procedura.
La prima istruzione di ritorno **RTN** ([INV][SBR]) restituirà il contatore di programma dopo la chiamata alla **SBR**.

● **RTN**([INV][SBR]) ritorna dalla procedura richiamata da SBR (Return).

Nel caso in cui l'esecuzione incontri un'istruzione **RTN** mentre nessuna istruzione **SBR** è in attesa di ritorno, allora **RTN** si comporta come **R/S** e termina il programma.

Nel caso di chiamate nidificate, il ritorno viene effettuato dopo l'ultima chiamata effettuata e così via fino all'esaurimento dello stack degli indirizzi di ritorno.

● **RST** ([RST]) riporta il contatore di programma al passo 000, azzera gli indirizzi di ritorno della subroutine e disattiva i segnalatori ("posizione bassa").

● **R/S** ([R/S]) in modo "calcolatrice" avvia il programma dalla posizione corrente del contatore di programma o lo interrompe, in modo "programma" interrompe il programma.

● **EQ**([2nd][x=t]) test condizionale, va all'indirizzo specificato se il registro x è uguale al registro t, altrimenti il programma continua in sequenza.
EQ può utilizzare, come **GTO**, sia l'indirizzamento logico che l'indirizzamento assoluto.
Esempio:
[2nd][x=t][lnx] vai all'etichetta LNX se x=t
[2nd][x=t][1][2][3] vai all'indirizzo 123 se x=t

● **EQ IND** ([2nd][x=t][2nd][Ind]) **nn** test condizionale, utilizzando un registro dati che contiene l'indirizzo del passo di destinazione se il registro x è uguale al registro t, altrimenti il programma continua in sequenza.

● **INV EQ** ([INV][2nd][x=t]) test condizionale, vai all'indirizzo specificato se il registro x è diverso dal registro t, altrimenti il programma continua in sequenza.

● **INV EQ IND** ([INV][2nd][x=t][2nd][Ind]) **nn** test condizionale, utilizzando un registro dati che contiene l'indirizzo del passo di destinazione se il registro x è diverso dal registro t, altrimenti il programma prosegue in sequenza.

● **GE** ([2nd][x≥t]) test condizionale, vai all'indirizzo specificato se il file il registro x è maggiore o uguale al registro t, altrimenti il programma continua in sequenza.
GE può utilizzare, come **GTO**, sia l'indirizzamento logico che l'indirizzamento assoluto.

● **GE IND** ([2nd][x≥t][2nd][Ind]) **nn** test condizionale, utilizza un registro dati che contiene l'indirizzo del passo di destinazione se il registro x è maggiore o uguale al registro t, altrimenti il programma continua in sequenza.

● **INV GE** ([INV][2nd][x≥t]) test condizionale, vai all'indirizzo specificato se il registro **x** è minore del registro **t**, altrimenti il programma continua in sequenza.

● **INV GE IND** ([INV][2nd][x≥t][2nd][Ind]) **nn** test condizionale, utilizza un registro dati che contiene l'indirizzo del passo di destinazione se il registro **x** è minore del registro **t**, altrimenti il programma continua in sequenza.

Salti condizionali	
Uguale	**EQ**
Diverso	**INV EQ**
Maggiore o uguale a	**GE**
Inferiore a	**INV GE**

Oltre ai test condizionali confrontando i registri x e t, la TI può gestire fino a 10 segnalatori (flag) il cui stato (alzato o abbassato) può essere testato per il salto. I segnalatori sono numerati da 0 a 9.

• **STF**([2nd][Stf]) **m** attiva il segnalatore specificato (Set Flag).
Esempio:
[2nd][Stf][1] attiva il segnalatore 1

• **INV STF**([INV][2nd][Stf]) **m** disattiva il segnalatore specificato.
Esempio:
[INV][2nd][Stf][1] disattiva il segnalatore 1

• **STF IND**([2nd][Stf][2nd][Ind]) **nn** utilizza un registro dati che contiene il numero del segnalatore da attivare.

• **INV STF IND** ([INV][2nd][Stf][2nd][Ind]) **nn** utilizza un registro dati che contiene il numero del segnalatore da disattivare.

• **IFF** ([2nd][Iff]) **m** test condizionale, vai all'indirizzo specificato se è attivato il segnalatore specificato, altrimenti il programma continua in sequenza.
IFF utilizza, come **GTO**, l'indirizzamento logico e l'indirizzamento assoluto.

• **IFF IND** ([2nd][Iff][2nd][Ind]) **nn** test condizionale, utilizzando un registro dati che contiene l'indirizzo del passo di destinazione se il segnalatore specificato è disattivato, altrimenti il programma continua in sequenza.

• **INV IFF** ([INV][2nd][Iff]) **m** test condizionale, vai all'indirizzo specificato se il segnalatore specificato viene abbassato, altrimenti il programma prosegue in sequenza.

---

(1) I flag 7, 8 e 9 hanno un significato speciale e precisamente il flag 7 viene utilizzato in abbinamento con Op 18 e Op 19 per gestire le condizioni di errore, il flag 8 interrompe l'esecuzione del programma in caso di errore e il flag 9 permette di tracciare il programma sulla stampante durante l'esecuzione. Il tasto TRACE sulla stampante ha la stessa funzione.

● **INV IFF IND**([INV][2nd][Iff][2nd][Ind]) **nn** test condizionale, utilizzando un registro dati che contiene l'indirizzo del passo di destinazione se il segnalatore specificato viene abbassato, altrimenti il programma continua in sequenza.

● **DSZ** ([2nd][Dsz])) **m** test condizionale che consente di gestire degli anelli (loop), manipola un registro dati (solo da 0 a 9) [1] e utilizza, come GTO, l'indirizzamento logico e assoluto.
DSZ procede in due fasi:

- Decremento di uno del registro testato se valore positivo (o incremento di uno se valore negativo)
- Test se il registro è a zero: se NO va all'indirizzo specificato, se SI continua in sequenza.

● **DSZ IND** ([2nd][Dsz][2nd][Ind]) **nn** test condizionale che consente di gestire degli anelli. Gestisce un registro dati (da 0 a 9 soltanto) [1] e utilizza un registro dati che contiene l'indirizzo del passo di destinazione se il test è soddisfatto.

● **INV DSZ** ([INV][2nd][Dsz]) **m** test condizionale utilizzato per gestire anelli (loop), manipola un registro dati (solo da 0 a 9) [1] e utilizza, come GTO, indirizzamento logico e assoluto.
INV DSZ procede in due fasi:

---

(1) È possibile utilizzare ogni registro per DSZ (eccetto il registro 40 che implica indirizzamento indiretto) nel limite della partizione corrente con il trucco spiegato più avanti per l'istruzione HIR.

- Decremento del registro testato se valore positivo (o incremento se valore negativo)
- Test se registro a zero: se SI va all'indirizzo specificato, se NO continua in sequenza.

● **INV DSZ IND** ([INV][2nd][Dsz][2nd][Ind]) **nn** test condizionale che permette di gestire anelli, manipola un registro dati (solo da 0 a 9) [(1)] e utilizza un registro dati che contiene l'indirizzo della fase di destinazione se il test è soddisfatto.

Statistiche

La TI gestisce le statistiche per un campione a due valori che rappresentano un punto su un piano di assi x e y. Sulla popolazione di punti, possiamo determinare la media, la varianza, la deviazione standard...

● Inizializzazione dei dati statistici: le statistiche utilizzano 6 registri dati, e il registro **t**, che deve essere azzerato prima di ogni nuova immissione.

Registri	01	02	03	04	05	06
Contenuto	Σy	Σy^2	N	Σx	Σx^2	Σxy

Questa inizializzazione può essere eseguita:

- o manualmente: **[2nd][Cms]** che cancella tutti i registri,
- o manualmente: **[CLR], [STO] [0][1], [STO] [0][2], [STO] [0][3], [STO] [0][4], [STO] [0][5], [STO] [0][6]**,
- Utilizzando la routine di inizializzazione del modulo 01 della libreria di base (ML-01): **[2nd][Pgm][0][1], [SBR][CLR]** (1)

(1) La sequenza nn [2nd][Pgm][0][1],[SBR][0][1][2] azzera i registri da **01** a **nn**, partizione di memoria permettendo..

● **STA** ([2nd][Σ+]) inserimento dati. È:
  • x [x⇆t] y [2nd][Σ+] per introdurre x e y
  • y [2nd][Σ+] per introdurre y da solo

il rango i viene visualizzato per ogni coppia (x_i, y_i) introdotta.

● **INV STA** ([INV][2nd][Σ+]) cancellazione dei dati. È:
  • x [x⇆t] y [INV][2nd][Σ+] per annullare x e y
  • y [INV][2nd][Σ+] per annullare y da solo

● **AVG** ([2nd][$\bar{x}$]) calcola e visualizza la media dei diversi valori di y ([x⇆t] per visualizzare la media dei diversi valori di x).

● **INV AVG** ([INV][2nd][$\bar{x}$]) calcola e visualizza la deviazione standard dei diversi valori di y ([x⇆t] per visualizzare la deviazione standard di diversi valori di x).

● **OP 11** ([2nd][Op][1][1]) calcola e visualizza la varianza dei diversi valori di y ([x⇆t] per visualizzare la varianza dei diversi valori di x).

● **OP 12** ([2nd][Op][1][2]) *Regressione lineare* - calcola e visualizza l'intercetta (punto di intersezione della linea di regressione con l'asse y per x = 0) e visualizza la pendenza.

● **OP 13** ([2nd][Op][1][3]) *Regressione lineare*: calcola e visualizza il coefficiente di correlazione, cioè quanto la retta di regressione si adatta ai dati x, y.

● **OP 14** ([2nd][Op][1][4]) *Regressione lineare*: calcola e visualizza il valore di y (y') per un valore immesso di x.

● **OP 15** ([2nd][Op][1][5]) *Regressione lineare*: calcola e visualizza il valore di x (x') per un valore immesso di y.

Tasti funzione

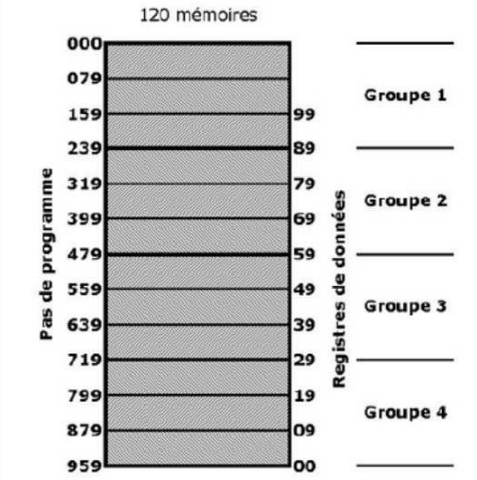
I tasti funzione (o tasti utente) sono 10. Possono essere utilizzati nei programmi come etichetta e possono essere richiamati dalle istruzioni di connessione (GTO, GE, EQ, ecc.).
L'utilizzo di un tasto da solo equivale a SBR. (**Esempio:** [A]=[SBR][A])
In modalità "calcolatrice", consentono l'avvio da un punto specifico del programma.

- **A** ([A])
- **B** ([B])
- **C** ([C])
- **D** ([D])
- **E** ([E])

- **A'** ([2nd][A])
- **B'** ([2nd][B])
- **C'** ([2nd][C])
- **D'** ([2nd][D])
- **E'** ([2nd][E])

Lettura/scrittura di schedee magnetiche

Le istruzioni di lettura/scrittura possono essere utilizzate solo su **TI-59** perché è l'unico modello dotato di lettore di schede magnetiche.
La TI-59 ha fino a 120 memorie dati divise in 4 gruppi.



Una scheda magnetica della TI-59 contiene 2 tracce che possono registrare ciascuna un gruppo.
Sono quindi necessarie due schede per immagazzinare tutta la memoria di una TI-59.

● **WRT** ([2nd][Write]) scrivi sulla scheda (deve essere preceduto dal numero del gruppo da registrare 1, 2, 3 o 4) [(1)].

(1) Se il numero del gruppo è negativo, il programma salvato risulterà "protetto" e, dopo il suo caricamento, potrà essere solo eseguito essendo disabilitate le istruzioni [2nd][List], [INV][2nd][List] e [2nd][Op] 08. La protezione sarà rimossa spegnendo e riaccendendo la calcolatrice, perdendo così il programma ed i dati. Analogo meccanismo di protezione è possibile per i moduli di libreria (soprattutto quelli prodotti da terze parti).

● **INV WRT**([INV][2nd][Write])) legge la scheda (se preceduta dal numero di gruppo con segno negativo -n, forza la lettura nel gruppo n)

Moduli di libreria

Con la calcolatrice viene fornito di serie un modulo inseribile chiamato "Biblioteca di Base (Master Library)" che contiene venticinque programmi di utilità.

Può essere sostituito da un altro dei moduli commercializzati dalla Texas Instruments o da terze parti.

#	Cod.	Nome
01	ML	Master Library
02	STA	Applied Statistics
03	RE	Real Estate / Investment
04	SV	Surveying
05	NG	Marine Navigation
06	AV	Aviation
07	LE	Leisure Library
08	SA	Securities Analysis
09	BD	Business Decisions
10	MU	Math / Utilities
11	EE	Electrical Engineering
12	FM	Agriculture
13	RPN	RPN Simulator

● **PGM**([2nd][Pgm]) consente di attivare o disattivare un programma della biblioteca.

- [2nd][Pgm] nn attiva il programma nn,
- [2nd][Pgm][0][0] disattiva il programma corrente.

[2nd][Pgm][0][1][SBR][2nd][Write] visualizza il numero del modulo inserito e stampa il suo nome se la stampante è collegata.

Esempio:
Il programma 24 della "Biblioteca di Base" converte le misure decimali (cm, m, km) in misure imperiali (pollici, piedi, yarde, miglia)
Quindi per sapere quanti metri è 1 iarda, devi inserire la sequenza:

[2nd][Pgm][2][4][1][C]

ML-24 UNIT CONVERSIONS (I)				
cm -> in	m -> ft	m -> yd	km -> mi	n.mi -> mi
in -> cm	ft -> m	yd -> m	mi -> km	mi -> n.mi

● **PG*** ([2nd][Pgm][2nd][Ind]) **nn** utilizza un registro dati che contiene il numero del programma da attivare.

● **OP 09** ([2nd][Op][0][9]) carica il programma attivato nella memoria di programma della TI. (*Nota:* Il programma in memoria viene cancellato!)

Operazioni speciali

● da **OP 01** a **OP 08** vedere *Stampa*

● **OP 09** vedere *Moduli libreria*

● **OP 10** vedere *Inserimento dati*

● **OP 11** a **OP 15** vedere *Statistiche*

● **OP 16** ([2nd][Op][1][6]) visualizza la partizione di memoria: ripartizione tra passi di programma e registri dati.

● **OP 17** ([2nd][Op][1][7]) effettua la partizione di memoria: ripartisce tra passi di programma e registri dati per gruppo di 10 registri.

TI59	OP 17	TI58/TI58C
959-00	0	479-00
879-09	1	399-09
799-19	2	319-19
719-29	3	239-29
639-39	4	159-39
559-49	5	079-49
479-59	6	000-59
399-69	7	
319-79	8	
239-89	9	
159-99	10	

Esempio:
Sulla TI-58, [3] [2nd] [Op][1] [7] darà 239.29 che significa 240 passi (da 000 a 239) e 30 registri (da 00 a 29)

● **OP 18** ([2nd][Op][1][8])attiva il segnalatore 7 se nessun errore di esecuzione viene incontrato.

● **OP 19** ([2nd][Op][1][9]) attiva il segnalatore 7 se un errore di esecuzione viene incontrato.

● da **OP 20** a **OP 39** vedere *Gestione delle memorie*

● **OP 40** ([2nd][Op][4][0]),solo su TI-58C, attiva il flag 7 se la stampante è collegata.

● **OP IND**([2nd][Op][2nd][Ind]) **nn** utilizza il contenuto di un registro nn per determinare quale operazione è applicabile.
Esempio:
[2nd][Op][2nd][Ind][0][1]utilizza il contenuto del registro 01.

- Se il registro 01 contiene 16, visualizza la partizione (come OP 16),
- Se il registro 01 contiene 0, cancella il buffer di stampa (uguale a OP 00).

Altre funzioni

● **PAU** ([2nd][Pause]) mantiene la visualizzazione del registro x per mezzo secondo durante l'esecuzione del programma. Diverse Pause possono susseguirsi per prolungare la visualizzazione.

● **NOP** ([2nd][Nop]) nessuna operazione. Istruzione senza alcun effetto per l'esecuzione. Utilizzato per inserire un grado per lasciare uno spazio tra due sequenze di programma o per sostituire una istruzione senza cambiare la numerazione dei passi invece di usare **Del**.

L'istruzione nascosta

● **HIR** (nessun tasto). Le TI-59/58/58C "nascondono" 8 registri interni utilizzati dal sistema per le proprie funzioni.

Il sistema basato sulla notazione algebrica diretta gestisce uno stack AOS in questi registri per mettere in attesa i numeri nei calcoli con più operatori in modo da rispettare la precedenza di questi operatori.

Quindi funzioni complesse (**STA, AVG, P/R, DMS**) memorizzano i risultati intermedi in questi registri e le funzioni statistiche (**OP 11, OP 12, OP 13, OP 14, OP 15**) e le funzioni di stampa alfanumerica (**OP 00, OP 01, OP 02, OP 03, OP 04**).

C'è un'istruzione speciale per la gestione di questi registri.

Ufficialmente, questa istruzione non esiste:

- Nessuna menzione nella documentazione della TI,
- Nessun tasto per inserirla in un programma.

E tuttavia ...
È quindi necessario essere astuti per introdurre questa istruzione con manipolazioni più simili alla giocoleria che alla programmazione.

Allora prendiamo un programmino...
Passiamo in modo programmazione ([LRN]) dopo aver cancellato il contenuto della memoria di programma ([2nd][CP]).

Introduciamo le seguenti istruzioni:

[2nd][LBL][A][STO][8][2][STO][1][1][R/S]

che ci dà con:[2nd]List]

```
000 76  LBL
001 11   A
002 42  STO
003 82   82
004 42  STO
005 11   11
006 91  R/S
```

Ora modifichiamo il nostro programma rimuovendo il passo 004 e quindi il passo 002:

- [RST],[LRN] quindi [SST] [SST] [SST] [SST] per andare al passaggio 004
- [2nd][Del] per eliminare il passaggio 004
- [BST],[BST] per andare al passo 002
- [2nd][Del] per cancellare il passo 002

Otteniamo :

```
000 76  LBL
001 11   A
002 82  HIR
003 11   11
004 91  R/S
```

Vediamo che il codice 82 è stato tradotto in HIR dalla stampante.
Quindi ecco la nostra istruzione nascosta che appare.

In modo "calcolatrice", inseriamo il seguente piccolo calcolo:

[7] [+] [3] [x] [4] [=]

che ci dà 19 perché la moltiplicazione ha la priorità sull'addizione.

Ora eseguiamo il nostro programmino premendo il tasto funzione [A].
Sul visualizzatore compare il numero 7.

Questa è la prima cifra del nostro calcolo che è stata messa in attesa (memorizzata nello stack AOS) in modo che la moltiplicazione possa essere eseguita per prima.

HIR 12 mostrerebbe 3 sul display, mostrandoci che anche il secondo numero della nostra operazione è stata memorizzato nello stack AOS.

- **HIR 0n** ($1 \le n \le 8$) trasferisce il contenuto del registro del visualizzatore x al registro interno n. (= STO)
- **HIR 1n** ($1 \le n \le 8$) trasferisce il contenuto del registro interno n al registro del visualizzatore x. (= RCL)
- **HIR 2n** ($1 \le n \le 8$) usato solo nel firmware della TI come HIR 20.
- **HIR 3n** ($1 \le n \le 8$) aggiunge il contenuto del registro del visualizzatore x al registro interno n. (= SUM)

- **HIR 4n** (1 ≤ n ≤ 8) moltiplica il contenuto del registro interno n dal contenuto del registro del visualizzatore x. (= PRD)
- **HIR 5n** (1 ≤ n ≤ 8) sottrae il contenuto del registro del visualizzatore x del registro interno n. (=INV SUM)
- **HIR 6n** (1 ≤ n ≤ 8) divide il contenuto del registro interno n per il contenuto del registro del visualizzatore x. (= INV PRD)

Tabella riassuntiva delle istruzioni

Cod.	Istr.	Tasti	Cod.	Istr.	Tasti
00	0	[0]	31	LRN	[LRN]
01	1	[1]	32	X:T	[x⇆t]
02	2	[2]	33	X^2	[x^2]
03	3	[3]	34	SQR	[√x]
04	4	[4]	35	1/X	[1/x]
05	5	[5]	36	PGM	[2nd][Pgm]
06	6	[6]	37	P/R	[2nd][P/R]
07	7	[7]	38	SIN	[2nd][sin]
08	8	[8]	39	COS	[2nd][cos]
09	9	[9]	40	IND	[2nd][Ind]
10	E'	[2nd][E]	41	SST	[SST]
11	A	[A]	42	STO	[STO]
12	B	[B]	43	RCL	[RCL]
13	C	[C]	44	SUM	[SUM]
14	D	[D]	45	Y^X	[Y^X]
15	E	[2nd]	46	INS	[2nd][Ins]
16	A'	[2nd]	47	CMS	[2nd][CMs]
17	B'	[2nd]	48	EXC	[2nd][Exc]
18	C'	[2nd]	49	PRD	[2nd][Prd]
19	D'	[2nd]	50	IXI	[2nd][|x|]
20	CLR	[2nd][CLR]	51	BST	[BST]
21	2nd	[2nd]	52	EE	[EE]
22	INV	[INV]	53	(	[( ]
23	LNX	[lnx]	54	)	[ )]
24	CE	[CE]	55	/	[÷]
25	CLR	[CLR]	56	DEL	[2nd][Del]
26	2nd	[2nd][2nd]	57	ENG	[2nd][Eng]
27	INV	[2nd][INV]	58	FIX	[2nd][Fix]
28	LOG	[2nd][log]	59	INT	[2nd][Int]
29	CP	[2nd][CP]	60	DEG	[2nd][Deg]
30	TAN	[2nd][tan]	61	GTO	[GTO]

Cod.	Istr.	Tasti	Cod.	Istr.	Tasti
62	PG*	[2nd][Pgm] [2nd][Ind]	81	RST	[RST]
63	EX*	[2nd][Exc] [2nd][Ind]	82	HIR	-------
64	PD*	[2nd][Prd] [2nd][Ind]	83	GO*	[GTO][2nd][Ind]
65	*	[×]	84	OP*	[2nd][Op][2nd][Ind]
66	PAU	[2nd][Pause]	85	+	[+]
67	EQ	[2nd][x=t]	86	STF	[2nd][Stf]
68	NOP	[2nd][Nop]	87	IFF	[2nd][Iff]
69	OP	[2nd][Op]	88	DMS	[2nd][DMS]
70	RAD	[2nd][Rad]	89	PI	[2nd][π]
71	SBR	[SBR]	90	LST	[2nd][List]
72	ST*	[STO][2nd][Ind]	91	R/S	[R/S]
73	RC*	[RCL][2nd][Ind]	92	RTN	[INV][SBR]
74	SM*	[SUM][2nd][Ind]	93	.	[.]
75	-	[-]	94	+/-	[+/-]
76	LBL	[2nd][Lbl]	95	=	[=]
77	GE	[2nd][x≥t]	96	WRT	[2nd][Write]
78	STA	[2nd][Σ+]	97	DSZ	[2nd][Dsz]
79	AVG	[2nd][x̄]	98	ADV	[2nd][Adv]
80	GRD	[2nd][Grad]	99	PRT	[2nd][Prt]

Test Comparativi

Per la stessa funzionalità, si possono presentare diverse soluzioni di programmazione.

Il costo nel numero di passi o nel tempo di esecuzione può influenzare la scelta della programmazione secondo il caso studiato.

A volte l'economia dei passi può essere cruciale poiché la memoria è relativamente limitata.
Occasionalmente il tempo di esecuzione sarà privilegiato come criterio di ottimizzazione.

Fortunatamente, data la natura stessa dei programmi sviluppati per questo tipo di macchina, queste preoccupazioni saranno molto spesso superflue.

Tuttavia, lo studio delle diverse ipotesi per la risoluzione di casi di programmazione rimane utile per comprendere i meccanismi del linguaggio.

Azzeramento dei registri

Un grande classico della programmazione con questo tipo di macchina è l'azzeramento di alcuni registri.

Infatti, per azzerare tutti i registri contemporaneamente abbiamo l'istruzione [2nd][CMs] che soddisfa tutti i criteri possibili: velocità e solo un passo di programma.

Ma per azzerare un insieme di registri avremo tre scelte di programmazione:

- Programmazione per decremento,
- Gestione delle partizioni,
- Utilizzo di programmi di libreria.

I 3 metodi discussi si basano sul ripristino dei registri da 00 a 09 e dei registri da 00 a 29.

1° metodo: Programmazione per decremento

```
... n
... n     nn = registro massimo (9 o 29)
... STO
...  00
... CLR
xxx ST*    xxx = indirizzo del salto
...  00
... DSZ
...  00
...  0x
... xxx
```

2° metodo: manipolazione delle partizioni

```
... n
... OP   n = numero di gruppi di memoria
...  17  (1 da 00 a 09, 3 da 00 a 29 [*])
... CMS
... m
... OP   m = torna alla partizione originale
...  17  (5 ad esempio per 159-39 [*])
```

[*] *riguarda le TI-58 e TI-58C*

3° metodo: utilizzo di programmi di libreria

```
... n
... n       nn = registro massimo (9 o 29)
... PGM
...  01
... SBR
...  00
...  12      Usando  il modulo base ML-01
```

oppure

```
... n
... n        nn = registro massimo (9 o 29)
... PGM
...  01
... SBR
...  00
...  04      Usando il modulo Math/Utilities MU-10
```

Ovviamente questi tre metodi non danno lo stesso risultato in termini di numero di passaggi e tempo di esecuzione:

	Metodo 1		Metodo 2		Metodo 3	
memorie	n.passi	tempi	n.passo	tempi	n.passi	tempi
00 a 09	10	3,5s	7	0,4s	6	2,4s
00 a 29	11	10,5s	7	0,4s	7	7s

Il 1° metodo che sembra il più giudizioso in termini di programmazione è comunque il più costoso in termini di passi oltre che in termini di tempo. Tuttavia, rimane il più utilizzato.

Il secondo metodo è il vincitore come tempo di esecuzione ma non offre compatibilità tra la **TI-58/58C** e la **TI-59** perché le definizioni dei gruppi di memoria non sono le stesse (vedi **OP 17**).

Il 3° metodo, poco utilizzato, è un buon compromesso e merita più attenzione.

Sequenza ripetitiva

In un programma, la presenza di una serie di istruzioni simili in più punti del codice è abbastanza comune.
La domanda che si pone allora è se abbia senso o meno trasformare questa sequenza ripetitiva in una procedura richiamabile ogniqualvolta sia necessario.
Sebbene alcuni metodi si basino su un'eccessiva modularità, il punto non è sistematizzare il processo ma piuttosto stimare quando può essere vantaggioso.

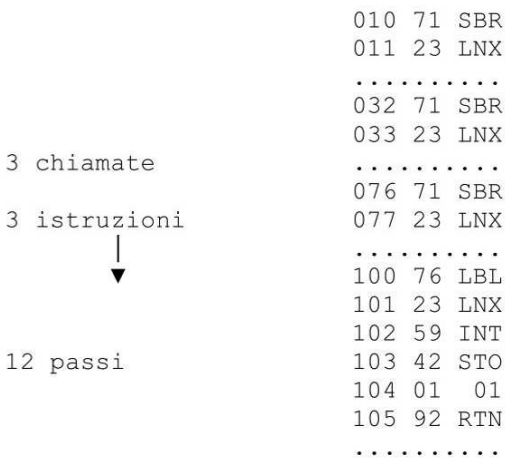
Gli esempi seguenti si basano sul principio di 3 istruzioni ripetute in 3 punti dello stesso programma. ([2nd][Int][STO][0][1])

Soluzione 1:
Scrivere la sequenza di istruzioni tante volte quanto necessario.

```
3 sequenze              010 59 INT
                        011 42 STO
                        012 01  01
                        ..........
                        032 59 INT
3 istruzioni            033 42 STO
      |                 034 01  01
      ▼                 ..........
                        076 59 INT
9 passi                 077 42 STO
                        078 01  01
```

n° passi = n° sequenze * n° istruzioni

Soluzione 2:
Richiamo di una procedura per indirizzamento relativo (etichetta).



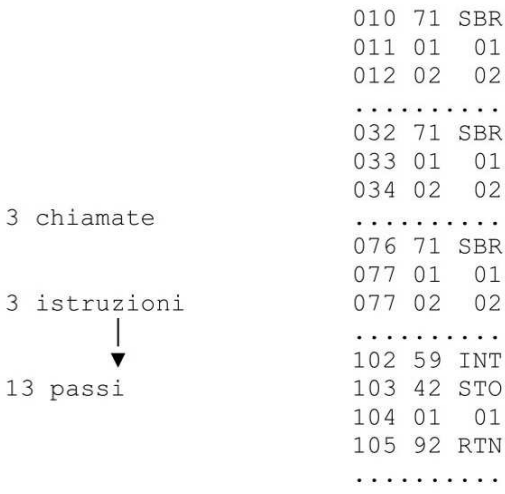
n° passi = (n° chiamate * 2) + (n° istruzioni + 3)

La seguente tabella riepilogativa ci permette di determinare, a partire da quante istruzioni e da quante chiamate, possa essere fatto un sostanziale risparmio di passi.

Chiamate	1		2		3		4		5		6		7		8	
Soluzione	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
I 1	1	6	2	8	3	10	4	12	5	14	6	16	7	18	8	20
s 2	2	7	4	9	6	11	8	13	10	15	12	17	14	19	16	21
t 3	3	8	6	10	9	12	12	14	15	16	18	18	21	20	24	22
r 4	4	9	8	11	12	13	16	15	20	17	24	19	28	21	32	23
u 5	5	10	10	12	15	14	20	16	25	18	30	20	35	22	40	24
z 6	6	11	12	13	18	15	24	17	30	19	36	21	42	23	48	25
i 7	7	12	14	14	21	16	28	18	35	20	42	22	49	24	56	26
o 8	8	13	16	15	24	17	32	19	40	21	48	23	56	25	64	27
n 9	9	14	18	16	27	18	36	20	45	22	54	24	63	26	72	28
i 10	10	15	20	17	30	19	40	21	50	23	60	25	70	27	80	29

Avremmo potuto studiare anche una terza soluzione...

Soluzione 3:
Richiamo di una procedura per indirizzamento assoluto (indirizzo di istruzione)



n° di passi = (n. di chiamate * 3) + (n. di istruzioni + 1)

Prova del ciclo

[RST] è il comando che mette il contatore di programma all'inizio della memoria di programma.

Di fatto, 3 possibilità permettono di tornare all'inizio della memoria di programma:

- [RST], ovviamente, ma questa istruzione azzera anche i flag e gli indirizzi di ritorno dei sottoprogrammi,
- [GTO] [0][0][0],
- [GTO] etichetta.

Tre semplici programmini possono aiutare a confrontare prestazioni di ogni caso.

Caso n° 1	Caso N° 2	Caso N° 3
000 85 +	000 85 +	000 76 LBL
001 01 1	001 01 1	001 23 LNX
002 81 RST	002 61 GTO	002 85 +
	003 00 0	003 01 1
	004 00 0	004 61 GTO
	005 00 0	005 23 LNX

Ogni esecuzione viene avviata per [RST][R/S] poi essere interrotta da [R/S] dopo 60 secondi.

	Conteggio per 1 mn		
	Risultato (+1)	N° passi	Rapporto
Caso N° 1	538	3	179.33
Caso N° 2	299	5	59.80
Caso N° 3	350	6	58.33

Il test sembra dimostrare, a parte [RST] (Caso N° 1), che l'indirizzamento relativo (caso N°3) sarebbe significativamente più efficiente dell'indirizzamento assoluto (caso N°2).

D'altra parte, [RST] sembra interessante, anche se poco utilizzato, perché economica in termini di passo questa istruzione è delle più veloce e merita una certa attenzione.

Chiamata di procedura

Il tipo di indirizzamento, assoluto (indirizzo) o relativo (etichetta), può essere utilizzato con tutte le istruzioni di salto condizionale o diretto.
Il test del ciclo eseguito in precedenza tenderebbe a dimostrare che l'indirizzamento relativo sia significativamente più efficiente dell'indirizzamento assoluto, ma ulteriori confronti portano al perfezionamento questo giudizio.

Per ogni tipo di indirizzamento, 3 casi dovrebbero permetterci di saperne di più:

- N° 1: Richiamo di una procedura all'inizio della memoria di programma,
- N° 2: Chiamare una procedura a metà della memoria di programma,
- N° 3: Richiamo di una procedura alla fine della memoria di programma.

1) indirizzamento relativo:

Caso n° 1	Caso N° 2	Caso N° 3
000 71 SBR	000 71 SBR	000 71 SBR
001 45 Y^X	001 45 Y^X	001 45 Y^X
002 81 RST	002 81 RST	002 81 RST
003 76 LBL	.....	.....
004 45 Y^X	235 76 LBL	475 76 LBL
005 81 +	236 45 Y^X	476 45 Y^X
006 01 1	237 81 +	477 81 +
007 92 RTN	238 01 1	478 01 1
	239 92 RTN	479 92 RTN

2) indirizzamento assoluto:

Caso n° 1	Caso N° 2	Caso N° 3
000 71 SBR	000 71 SBR	000 71 SBR
001 00 00	001 02 02	001 04 04
002 04 04	002 37 37	002 77 77
003 81 RST	003 81 RST	003 81 RST
004 81 +	.....	.....
005 01 1	237 81 +	477 81 +
006 92 RTN	238 01 1	478 01 1
	239 92 RTN	479 92 RTN

Ogni programma viene eseguito ([RST][R/S]) quindi interrotto da [R/S] dopo 60 secondi.

Conteggio per 1 mn

	Caso N°1	Caso N°2	Caso N°3
Relativo	224	66	32
Indirizzamento assoluto	208	196	186

Questi programmi ci dimostrano che i due tipi di indirizzamento sono competitivi per indirizzi bassi ma che l'indirizzamento assoluto è più veloce per indirizzi alti.

La calcolatrice e le sue funzioni statistiche possono essere utilizzate per eseguire l'analisi di tendenza:

1) Per l'indirizzamento assoluto, introduciamo il nostro esempio:

Indirizzo_del_passo **X:T** conteggio **STA**

[4] [x⇆t] [2][0][8] [2nd] [Σ+]
[2][3][7] [x⇆t] [1][9][6] [2nd][Σ+]
[4][7][7] [x⇆t] [1][8][6] [2nd][Σ+]

Possiamo calcolare diversi valori di y (*conteggio*) della retta di regressione inserendo diversi valori di x (*indirizzo_del_passo*) seguiti da **OP 14**.

[9] [2nd][Op][1][4] dà 207, ...
[5][9] [2nd][Op][1][4] dà 204, ...
[1][0][9] [2nd][Op][1][4] dà 202, ...

e così via fino a 459.

2) Per l'indirizzamento relativo, introduciamo il nostro esempio:

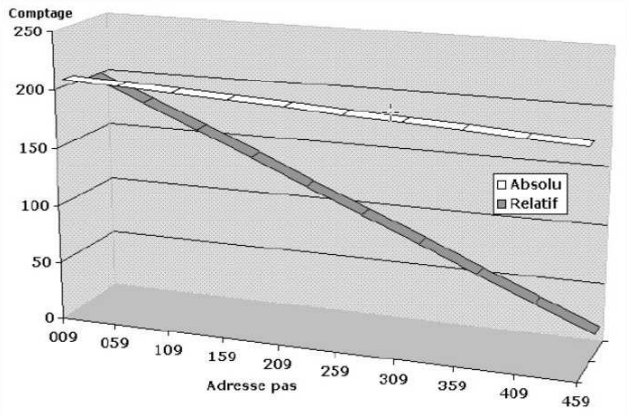
Indirizzo_del_passo **X:T** conteggio **STA**

[4] [x⇆t] [2][0][8] [2nd][Σ+]
[2][3][7] [x⇆t] [1][9][6] [2nd][Σ+]
[4][7][7] [x⇆t] [1][8][6] [2nd][Σ+]

Possiamo calcolare diversi valori di y (conteggio) della linea di regressione inserendo diversi valori di x (indirizzo del passo) seguiti da **OP 14**.

[9] [2nd] [Op][1][4] dà 200, ...
[5][9] [2nd] [Op][1][4] dà 180, ...
[1][0][9] [2nd] [Op][1][4] dà 160, ...
e così via fino a 459.

Otteniamo i seguenti dati:



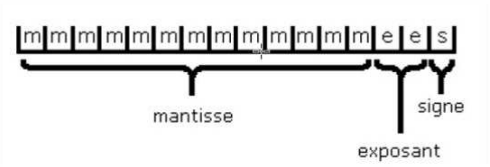
Questo grafico conferma le prestazioni dell'indirizzamento assoluto.

I dati

Struttura dei dati

I dati vengono visualizzati su10 cifre, possibilmente con un segno meno.
Nel caso di visualizzazione in notazione scientifica ([EE]) o “ingegneristica” ([Eng]) la mantissa viene visualizzata su 8 cifre e l'esponente su 2 cifre con eventuale visualizzazione dei segni meno (mantissa e/o esponente).

In tutti i casi, la gestione interna dei registri rimane la stessa: la mantissa su 13 caratteri, l'esponente su 2 caratteri e 1 carattere per esprimere i segni.
Cioè un totale di 16 caratteri [1] (o 2 byte)



Valore Segno

segno	Mantissa	Esponente
0	+	+
2	-	+
4	+	-
6	-	-

(1) Chiamati anche “nibble” in inglese.

Analisi dei dati

La memoria della calcolatrice è condivisa tra programma e dati. Questa divisione può essere modificata ([2nd][Op][1][7]) per dividere la memoria tra passi di programma e registri dati

OP 17	TI-58/TI-58C
0	479-00
1	399-09
2	319-19
3	239-29
4	159-39
5	079-49
6	000-59

Prendendo come riferimento la **TI-58**, vediamo che 480 passi di programma corrispondono a 60 registri.
Un registro prende quindi il posto di 8 passi.

Abbiamo 60 registri da 16 caratteri, oppure 480 passi da 2 caratteri: la **TI-58** ha quindi 960 caratteri di memoria utilizzabile. (1920 per la **TI-59**)

Nel caso di una partizione TI-58 "239-29" ([3][2nd][Op][1][7]) abbiamo 480 byte disponibili per il programma (240 passi) e 480 byte disponibili per i dati (30 registri).

Questa ripartizione tra programma e dati ci permette di fare un'equivalenza tra passi e registri (per la **TI-58**):

- per registrare 00 corrispondono ai passi 479, 478, 477,476, 475, 474, 473, 472.
- al registro 59 corrispondono i passi 007, 006, 005, 004, 003, 002, 001, 000.
- eccetera ...

Reg	Passo da	a	Reg	Passo da	a	Reg	Passo da	a	Reg	Passo da	a
00	479	472	15	359	352	30	239	232	45	119	112
01	471	464	16	351	344	31	231	224	46	111	104
02	463	456	17	343	336	32	223	216	47	103	096
03	455	448	18	335	328	33	215	208	48	095	088
04	447	440	19	327	320	34	207	200	49	087	080
05	439	432	20	319	312	35	199	192	50	079	072
06	431	424	21	311	304	36	191	184	51	071	064
07	423	416	22	303	296	37	183	176	52	063	056
08	415	408	23	295	288	38	175	168	53	055	048
09	407	400	24	287	280	39	167	160	54	047	040
10	399	392	25	279	272	40	159	152	55	039	032
11	391	384	26	271	264	41	151	144	56	031	024
12	383	376	27	263	256	42	143	136	57	023	016
13	375	368	28	255	248	43	135	128	58	015	008
14	367	360	29	247	240	44	127	120	59	007	000

Possiamo verificare questa logica di corrispondenza.

In modo "calcolatrice", inseriamo:

```
Tasti                Visualizzatore
[ 4 ][2nd][Op ][1][7]     159.39 Cambio partizione
[2nd][ π ]             3.14159265 PI
[STO][3][0]                       memorizza nel registro 30
[ 4 ][2nd][Op ][1][7]     239.29 Cambio partizione
[GTO][2][3][9][LRN]       239 31 modo programmazione
```

Analizziamo i passi, a ritroso:
`Il display ci dà 239 31 quindi ...`

- `[BST] ci dà 238 41`
- `[BST] ci dà 237 59`
- `[BST] ci dà 236 26`
- `[BST] ci dà 235 53`
- `[BST] ci dà 234 59`
- `[BST] ci dà 233 00`
- `[BST] ci dà 232 00`

cioè:

Mantissa									**Esp.S.**	
Reg. 30	3	1	4	1	5	9	2	6	0	0	0	0
Pas	239		238		237		236		235		234		233		232	

Registri interni

I registri interni che possono essere manipolati con l'istruzione nascosta **HIR** sono utilizzati dallo stack AOS, da cui la necessità di comprenderne il funzionamento al fine di evitare conflitti tra l'uso personale di tali registri e la gestione effettuata dalla calcolatrice degli stessi registri.

La seguente operazione utilizza l'intero stack, quindi tutti i registri interni:

2 x (8 - (90 / (3 * (9 - (1 + (45 / (3 * 5)))))) =

Un'analisi di questi registri tramite un programma (vedi pagina successiva) ci dà:

```
 2.  HIR1
 8.  HIR2
90.  HIR3
 3.  HIR4
 9.  HIR5
 1.  HIR6
45.  HIR7
 3.  HIR8
```

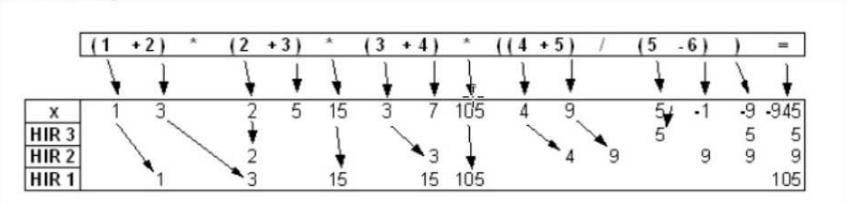
Cioè, tutti gli operandi inseriti fino alla prima parentesi chiusa.

```
000 76 LBL | 041 42 STO | 082 69  OP | 124 00  0
001 11  A  | 042 01  01 | 083 00  0  | 125 07  7
002 02  2  | 043 82 HIR | 084 03  3  | 126 69 OP
003 65  x  | 044 12  12 | 085 69 OP  | 127 04  04
004 53  (  | 045 42 STO | 086 04  04 | 128 43 RCL
005 08  8  | 046 02  02 | 087 43 RCL | 129 06  06
006 75  -  | 047 82 HIR | 088 02  02 | 130 69  OP
007 53  (  | 048 13  13 | 089 69 OP  | 131 06  06
008 09  9  | 049 42 STO | 090 06  06 | 132 71 SBR
009 00  0  | 050 03  03 | 091 71 SBR | 133 69  OP
010 55  /  | 051 82 HIR | 092 69  OP | 134 01  1
011 53  (  | 052 14  14 | 093 00  0  | 135 00  0
012 03  3  | 053 42 STO | 094 04  4  | 136 69 OP
013 65  x  | 054 04  04 | 095 69 OP  | 137 01  01
014 53  (  | 055 82 HIR | 096 01  01 | 138 43 RCL
015 09  9  | 056 15  15 | 097 43 RCL | 139 07  07
016 75  -  | 057 42 STO | 098 03  03 | 140 69 OP
017 53  (  | 058 05  05 | 099 69 OP  | 141 06  06
018 01  1  | 059 82 HIR | 100 06  06 | 142 71 SBR
019 85  +  | 060 13  13 | 101 71 SBR | 143 69 OP
020 53  (  | 061 42 STO | 102 69  OP | 144 01  1
021 04  4  | 062 06  06 | 103 00  0  | 145 01  1
022 05  5  | 063 82 HIR | 104 05  5  | 146 69 OP
023 55  ÷  | 064 17  17 | 105 69 OP  | 147 04  04
024 53  (  | 065 42 STO | 106 04  04 | 148 43 RCL
025 03  3  | 066 07  07 | 107 43 RCL | 149 08  08
026 65  x  | 067 82 HIR | 108 04  04 | 150 69 OP
027 05  5  | 068 18  18 | 109 69 OP  | 151 06  06
028 54  )  | 069 42 STO | 110 06  06 | 152 91 R/S
029 54  )  | 070 08  08 | 111 71 SBR | 153 76 LBL
030 54  )  | 071 71 SBR | 112 69  OP | 154 69 OP
031 54  )  | 072 69  OP | 113 06  0  | 155 02  2
032 54  )  | 073 00  0  | 114 00  6  | 156 03  3
033 54  )  | 074 02  2  | 115 69 OP  | 157 02  2
034 54  )  | 075 69 OP  | 116 04  04 | 158 04  4
035 95  =  | 076 04  04 | 117 43 RCL | 159 03  3
036 91 R/S | 077 43 RCL | 118 05  05 | 160 05  5
037 76 LBL | 078 01  01 | 119 69  OP | 161 92 RTN
038 12  D  | 079 69 OP  | 120 06  06 |
039 82 HIR | 080 06  06 | 121 71 SBR |
040 11  11 | 081 71 SBR | 122 69  OP |
```

Lo stack AOS funziona come segue:

- Un numero seguito da un operatore memorizza il registro di visualizzazione x (numero precedente o risultato intermedio) nel registro **HIR** di rango r,
- Un operatore o una parentesi aperta aggiunge 1 al rango r,
- Una parentesi chiusa esegue l'ultimo operatore tra il registro di visualizzazione x e il registro **HIR** di rango r, inserisce il risultato nel registro di visualizzazione x e quindi sottrae 1 dal rango r.

Esempio:



Qualsiasi risultato intermedio viene visualizzato sul display prima di essere memorizzato nello stack AOS.

I registri interni **HIR** sono utilizzati anche dalle funzioni di stampa alfanumerica. Se, in modo "calcolatrice", inseriamo:

6 4 6 4 6 4 6 4 6 4 **OP 01**
3 6 3 6 3 6 3 6 3 6 **OP 02**
5 2 5 2 5 2 5 2 5 2 **OP 03**
7 7 7 7 7 7 7 7 7 7 **OP 04**

OP 05 ci dà:

=====SSSSS√√√√ΣΣΣΣΣ

Si noti il contenuto dei registri interni da 5 a 8:

- **HIR 15** fornisce .0064646465 (6464646464000034 internamente)
- **HIR 16** fornisce .0036363636 (3636363636000034 internamente)
- **HIR 17** fornisce .0052525253 (5252525252000034 internamente)
- **HIR 18** fornisce .0077777778 (7777777777000034 internamente)

Ecco perché il programma a pagina 106 recupera i registri **HIR** per memorizzarli nei registri dati da 00 a 08 prima di utilizzare le funzioni di stampa OP: per evitare conflitti tra l'uso personale di questi registri e la gestione fatta dal calcolatore degli stessi registri.

Come esercitarsi?

Le calcolatrici TI-59/58/ 58C necessarie per la pratica del linguaggio LMS non sono più commercializzate da molti anni.
Sebbene a volte sia possibile trovare l'usato nei mercatini o nei siti di aste, queste opportunità sono piuttosto rare e le condizioni delle macchine così rinvenute non sono realmente garantite, la tastiera tende a "rimbalzare" e le batterie sono spesso difettose.

Fortunatamente la passione degli "appassionati" è continuata negli anni e internet offre vari siti che offrono informazioni interessanti, manuali e altra documentazione ma soprattutto soluzioni sostitutive: emulatori in esecuzione su PC o tablet (MS-DOS, Windows, Android, Pocket PC).

Un emulatore per TI-59/58/58C, su piattaforma Windows, è offerto su un sito interamente dedicato a questi calcolatrici, ed è ora possibile abbandonarsi ai piaceri di questa linguaggio scaricando questo software gratuito da

http://TI58C.phweb.me

Questo sito elenca la maggior parte dei software di emulazione disponibili e fornisce anche i collegamenti principali ad altri siti dedicati a questi calcolatori.

Indice

Indice analitico