Programmare meglio..

SHARP PC-1211

di Fabio Marzocca

Questo articolo non vuole assolutamente essere una copia del manuale d'impiego della Sharp PC-1211, ma piuttosto un "tiriamo le somme" sulla programmazione di questi pochi programmi di Basic, molto diffusi ormai in Italia. Esploreremo le capacità più nascoste delle 1211 mentre, per chi si accingesse ad acquistarne un esemplare in questi giorni, daremo un'occhiata sommaria alle caratteristiche più peculiari della sua programmazione. Il manuale che viene consegnato insieme alla macchina purtroppo è in grado di descrivere solo una piccola frazione delle reali possibilità d'impiego di questo pocket computer, completamente tascabile e dalle infaticabili caratteristiche.

Le variabili

Nella PC-1211 la data memory è divisa in fixed-memory (26 variabili da A a Z), e in flexible-memory (da A(27) a A(204) in assenza di programma), e ciascuna variabile può accettare valori numerici o caratteri alfanumerici. Attenzione, però: se esiste la variabile A, in cui è stato registrato un valore numerico, non può esistere la variabile A$, e viceversa. La parte di memoria variabile identificata come fixed-memory può anche essere espressa da una variabile ad indice da A(1) a A(26). Cosicché A(6) corrisponde alla memoria F, A(26) alla memoria Z, ecc. Questa tecnica però va usata con molta accortezza, in special modo durante i cicli FOR-NEXT.

Vediamo un classico esempio di ciclo che provoca un loop infinito per la 1211:

```
10 FOR B = 1 TO 10
20 A(B) = 1
30 NEXT B
```

Ciò è dovuto al fatto che la memoria A(2) è la stessa di B, cosicché ogni volta che B diventa 2, viene resettato a 1, per cui il ciclo non ha mai termine. Per evitare questi errori, perciò, è importante determinare se, durante il ciclo, la variabile di controllo viene influenzata dalle frasi di assegnazione.

Un'operazione tra variabili consentita dalla PC-1211, ma non da computer di ben altra "stazza", è l'omissione del segno di moltiplicazione tra due variabili. L'esempio seguente calcola il volume di un parallelepipedo a base quadrata:

```
10 INPUT "SPIGOLO?" ; A
20 INPUT "ALTEZZA?"; B
30 C = AAB
40 PRINT "VOLUME="; C
```

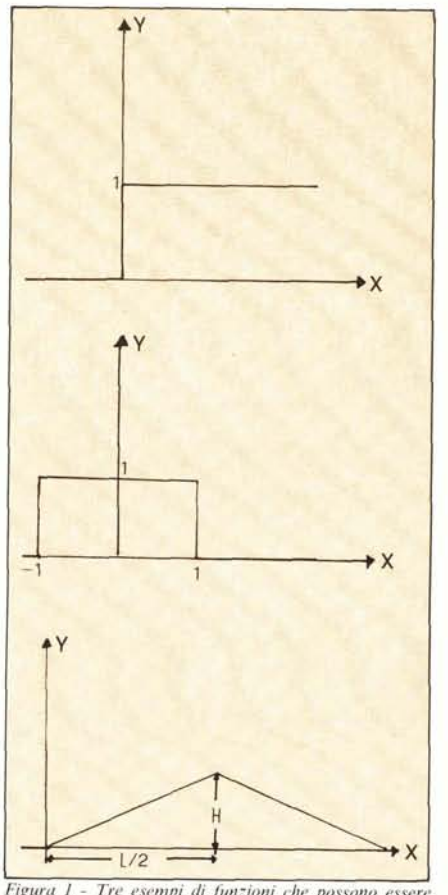

*Figura 1 - Tre esempi di funzioni che possono essere definite in una singola linea Basic dalla PC-1211, grazie alla possibilità di gestire in forma "mista" le variabili reali e quelle logiche. La frase Y = (X>0) definisce la funzione "gradino" (prima in alto); Y=(X>-1) * (X<1) definisce la funzione "unità" (al centro); mentre la funzione triangolare (in basso) è definita dall'espressione:*
Y=(X>0)(X<L/2)*2HX/L+*
+2H(1-X/L)*(X<L)*(X>L/2)*

La linea 30 mette in evidenza la proprietà appena descritta. Con i metodi tradizionali sarebbero stati impiegati almeno 2 byte in più:

```
30 C = B * A^2
```

e, soprattutto, con un tempo d'elaborazione doppio, per quanto riguarda la 1211. Inoltre il prodotto AB non è soltanto un'esatta sostituzione del prodotto A * B, in quanto il primo ha priorità su tutte le moltiplicazioni e divisioni con il segno esplicito. AB viene calcolato prima di A * B e di A/B ma non prima di A^B. Quindi spesso possono essere risparmiati anche 3 byte sostituendo AB al prodotto (A * B).

La memoria

Abbiamo visto che, in assenza di programmi in memoria, le variabili "flessibili" vanno da A(27) a A(204), quindi 178 più le 26 fisse da A(1) a A(26), comunque sempre disponibili. Ogni 8 "step", di programma, si sacrifica una cella di memoria flessibile, per cui è evidente come sia importante in questo caso scrivere programmi il più economicamente possibile. Una tecnica da tenere a mente è senz'altro quella descritta nel paragrafo precedente.

Ogni parola del Basic PC-1211 occupa un byte, mentre il line-number ne occupa 3: si vede quindi come può essere fruttuoso lo scrivere più frasi sulla stessa linea, separandole con i due punti. Ricordando inoltre che ogni carattere in una stringa occupa un byte, spesso possiamo evitare segnalazioni come questa:

```
10 INPUT "INSERIRE IL VALORE DI
X"; X
```

che occupa 31 byte, sostituendola con la frase:

```
10 INPUT "X="; X
```

che ne occupa solo 10.

Se ora si andasse ad indagare su come i dati vengono rappresentati in memoria, potremmo trovare ulteriori soluzioni ai nostri problemi di spazio. La PC-1211 rappresenta i numeri con una mantissa di 33 bit, e ciò significa che possiamo immagazzinare 33 singole informazioni in ogni cella. È chiaro che una precisione di una parte su 10^{10} occorre solo in casi di calcoli scientifici, mentre per ciò che riguarda giochi, controlli di magazzino, archivi, applicazioni commerciali, ci accontentiamo anche di molto meno.

Il seguente programma realizza un archivio di studenti, codificati da 1 a 156, e per ogni studente memorizza gli esami superati nel corso di laurea, con codice da 1 a 32:

```
10 INPUT "CODICE STUDENTE?"; A:
   A = A+26
20 INPUT "CODICE ESAME?"; B
30 GOSUB 500
40 IF F PRINT "ESAME GIÀ SUPERA-
   TO" : GOTO 10
50 A(A) = 2^B + A(A) : PRINT "REGI-
   STRATO ESAME": GOTO 10
500 G = A(A)*2^-(B+1)
510 F = INT (2* (G-INT(G)))
520 RETURN
```

Dopo aver ricevuto i dati, il programma passa alla subroutine 500 la quale caricherà in F il valore 1 se l'easame è stato già superato dallo studente A(A), altrimenti F sarà 0 e nella linea 50 si provvederà a registrare l'esame nella "scheda" dello studente. Perciò ogni memoria A(A) rappresenta uno studente, e può contenere fino a 32 informazioni. Se nella memoria della PC-1211 c'è solo il programma descritto, si hanno a disposizione 156 memorie per altrettanti studenti, riuscendo così a gestire circa 5000 informazioni!

L'istruzione IF

La semantica della frase

IF <espressione> THEN ...

nel Basic della PC-1211 è la seguente: se <espressione> è maggiore di 0, esegui *tutte* le istruzioni che seguono il THEN, anche se separate dai due punti, altrimenti se è uguale o minore di 0 salta alla linea successiva.

Quindi provando a digitare sul visualizzatore (A>C) e premendo enter, si avrà come risposta 1 se A è maggiore di C, altrimenti 0. Inoltre abbiamo la possibilità di risparmiare un paio di byte scrivendo

10 IF A THEN 100

invece di

10 IF A>0 THEN 100

Dato che per la PC-1211 non esistono distinzioni da dichiarare fra le variabili reali e quelle logiche, è possibile realizzare funzioni "miste" di grande utilità (v. fig. 1). La frase:

IF A*B*C*D ...*K THEN 100

sta ad indicare: "se *tutte* le variabili sono vere, allora vai a 100".

IF A + B + C + ... + K THEN 100

invece significa: "se *almeno una* delle variabili è vera, vai a 100". Si possono creare anche casi intermedi, come:

IF A+B+C+D ... +K>,3 THEN 100

che indica: "se *almeno 3* delle variabili sono vere, vai a 100", e così via.

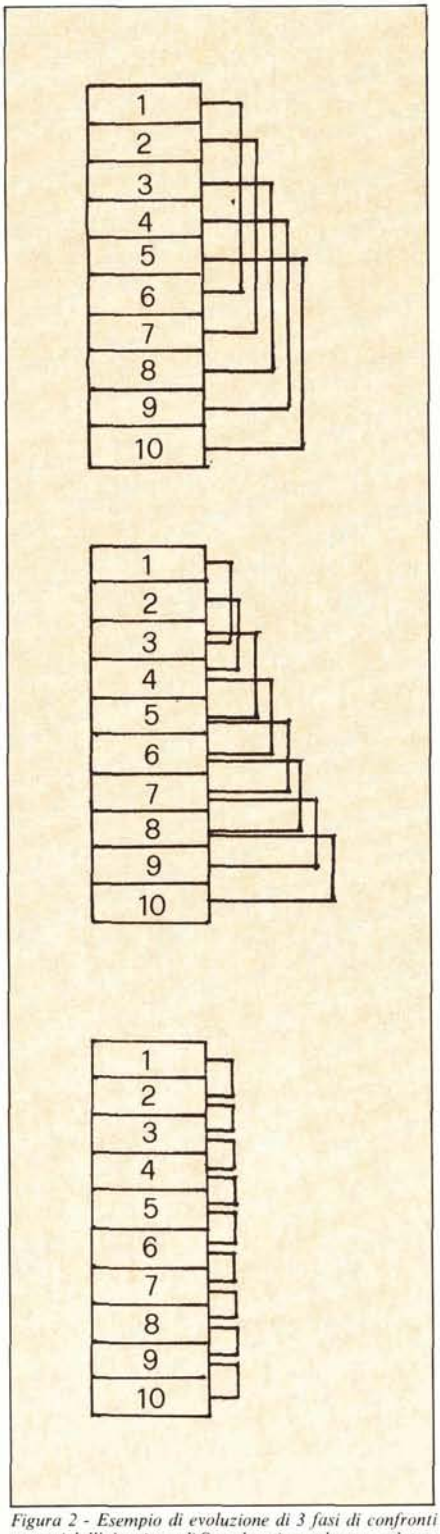

Figura 2 - Esempio di evoluzione di 3 fasi di confronti operati dall'algoritmo di Sort descritto nel testo, nel caso di un riordino su 10 elementi.
Le linee esterne al vettore indicano gli elementi confrontati fra loro.

Routine di utilità

Generatore numeri casuali

Il generatore descritto sul manuale della PC-1211 occupa inutilmente molta memoria e non è spesso utilizzabile nella totalità dei casi. Vi proponiamo così la seguente routine che può essere registrata in una cella RESERVE (es.: SHFT A), e restare sempre a disposizione dell'operatore:

X = (X+π) ^5 - INT ((X+π) ^5)

La routine va inizializzata con un valore decimale per X, e fornirà come output numeri compresi fra 0 e 1.

Fattoriale

Per il fattoriale vi presentiamo 2 routine diverse. La prima non è altro che un loop sulla variabile F, ed è utilizzabile al massimo fino a N = 100, dopodiché i tempi diventano inaccettabili.

```
10 INPUT N : F = 1
20 FOR W = 1 TO N: F=F*W : NEXT
   W
30 PRINT F
```

La seconda routine invece è molto veloce, ed ha un tempo di elaborazione costante ed indipendente da N; i risultati ottenuti però sono approssimati secondo l'algoritmo di Pearson:

```
10 INPUT N
20 J = SIN(1/N/√ 5)*√ 5/ 12+LN
(2πN) /2+ (LN N-1)*N
30 J = J/LN 10 : D = INT J
40 F = 10^(J - D)
50 Print F; "E"; D
(il programma gira in RADIAN mode)
```

Sort

Gli algoritmi di sort usati in Basic per il riordino di una lista sono ormai innumerevoli. Per la PC-1211 abbiamo scelto quello di shell-Metzner (v. fig. 2) in quanto garantisce una certa velocità rispetto ad altri più noti quali ad esempio il Bubble-sort. Le linee da 10 a 40 provvedono all'input: dopo l'ultimo dato immesso, inserire 9999 per dare il via al riordino. La routine di output parte dalla linea 150, mentre da 50 a 140 è contenuta la routine di sort.

```
10 CLEAR
20 FOR A = 8 TO 100 : INPUT A(A)
30 IF A(A) = 9999 THEN 50
40 NEXT A
50 B = A - 7
60 B = INT(B/2) : IF B=0 THEN 150
70 C = A-B-7 : D = 1
80 E = D+7
90 F = E + B
100 IF A(E)<= A(F) THEN 130
110 G = A(E):A(E) = A(F): A(F)=G
120 E=E-B : IF E>= 8 THEN 90
130 D = D+1 : IF D>C THEN 60
140 GOTO 80
150 FOR B = 8 TO A-1
160 PRINT A(B)
170 NEXT B
180 END
```

MC