## TABELLA N. 1 - COMANDI DOS

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | APPEND | 6 | DATE | 11 | FREE | 16 | PRINT |
| 2 | ATTRIB | 7 | DEBUG | 12 | KILL | 17 | PROT |
| 3 | AUTO | 8 | DIR | 13 | LIB | 18 | RENAME |
| 4 | BASIC | 9 | DUMP | 14 | LIST | 19 | TIME |
| 5 | COPY | 10 | FORMAT | 15 | LOAD | 20 | VERIFY |

## TABELLA N. 2 - COMANDI E ISTRUZIONI BASIC

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | AND | 20 | EOF | 39 | LLIST | 58 | PRINTUSING (?USING) |
| 2 | AUTO | 21 | ERL | 40 | LOAD | 59 | PUT |
| 3 | CLEAR | 22 | ERR | 41 | LOC | 60 | READ |
| 4 | CLOSE | 23 | ERROR | 42 | LOF | 61 | REF |
| 5 | CMD"D" | 24 | FIELD | 43 | LPRINT | 62 | REM (') |
| 6 | CMD"funzione DOS" | 25 | FOR...TO...STEP...NEXT | 44 | LSET | 63 | RENUM |
| 7 | CMD"S" | 26 | FRE | 45 | MEM | 64 | RESTORE |
| 8 | CONT | 27 | GET | 46 | MERGE | 65 | RESUME |
| 9 | DATA | 28 | GOSUB | 47 | NEW | 66 | RETURN |
| 10 | DEFDBL | 29 | GOTO | 48 | NOT | 67 | REVOFF |
| 11 | DEFFN | 30 | IF...THEN...ELSE | 49 | ON...GOSUB | 68 | REVON |
| 12 | DEFINT | 31 | INKEY $ | 50 | ON...GOTO | 69 | RSET |
| 13 | DEFSNG | 32 | INPUT | 51 | ONERRORGOTO | 70 | RUN |
| 14 | DEFSTR | 33 | INPUT # | 52 | OPEN | 71 | SAVE |
| 15 | DEFUSR | 34 | KILL | 53 | OR | 72 | STOP |
| 16 | DELETE (D) | 35 | LET | 54 | PRINT (?) | 73 | SYSTEM |
| 17 | DIM | 36 | LINEINPUT | 55 | PRINT # (?#) | 74 | TROFF |
| 18 | EDIT (E) | 37 | LINEINPUT # | 56 | PINT @ (?@) | 75 | TRON |
| 19 | END | 38 | LIST (L) | 57 | PRINTTAB (?TAB) | | |

## TABELLA N. 3 - FUNZIONI DI STRINGA

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | ASC | 5 | CVS | 9 | MID $ | 13 | RIGHT $ |
| 2 | CHR $ | 6 | INSTR | 10 | MKD $ | 14 | STR $ |
| 3 | CVD | 7 | LEFT $ | 11 | MKI $ | 15 | STRING $ |
| 4 | CVI | 8 | LEN | 12 | MKS $ | 16 | VAL |

## TABELLA N. 4 - FUNZIONI ARITMETICHE

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | &H | 6 | CINT | 11 | FIX | 16 | SGN |
| 2 | &O | 7 | COS | 12 | INT | 17 | SIN |
| 3 | ABS | 8 | CSNG | 13 | LOG | 18 | SQR |
| 4 | ATN | 9 | E | 14 | RANDOM | 19 | TAN |
| 5 | CDBL | 10 | EXP | 15 | RND | 20 | ↑ |

## TABELLA N. 5 - FUNZIONI GRAFICHE

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | CLS | 2 | POINT | 3 | RESET | 4 | SET |

## TABELLA N. 6 - FUNZIONI SPECIALI

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | INP | 3 | PEEK | 5 | POS | 7 | USR |
| 2 | OUT | 4 | POKE | 6 | TIME $ | 8 | VARPTR |

## TABELLA N. 7 - SIMBOLI CHIAVE

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | ! | 7 | " | 13 | . | 19 | <> |
| 2 | # | 8 | ( ) | 14 | / | 20 | = |
| 3 | $ | 9 | * | 15 | : | 21 | > |
| 4 | % | 10 | + | 16 | ; | 22 | >= |
| 5 | & | 11 | , | 17 | < | 23 | ? |
| 6 | ' | 12 | — | 18 | <= | 24 | @ |
| | | | | | | 25 | ↑ |

## TABELLA n. 8 - COMANDI SPECIALI DA TASTIERA

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | (II) | 4 | (CTRL + I) | 7 | (CTRL + R) | 10 | (CTRL + Y) |
| 2 | (CTRL + A) | 5 | (CTRL + L) | 8 | (CTRL + T) | 11 | (DEL) |
| 3 | (CTRL + H) | 6 | (CTRL + P) | 9 | (CTRL + X) | 12 | (SPAZIO) |

## TABELLA N. 9 - COMANDI DI EDITING

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | (←) | 6 | (C) | 11 | (S) | 16 | (CTRL + J) |
| 2 | (↑) | 7 | (D) | 12 | (H) | 17 | (ESC) |
| 3 | (.) | 8 | (I) | 13 | (E) | 18 | (RETURN) |
| 4 | (,) | 9 | (K) | 14 | (Q) | | |
| 5 | (L) | 10 | (X) | 15 | (A) | | |

# ISTRUZIONI E COMANDI DEL BASIC E DEL DOS

**Riprendiamo in questo numero la descrizione delle istruzioni e dei comandi del Basic che dovrete conoscere per poter programmare qualsiasi computer che utilizzi tale linguaggio. Ovviamente essendo l'elenco di tali istruzioni molto lungo ci limiteremo per ora a brevi cenni riguardo ognuna di esse riservandoci comunque di spiegarvi in seguito più dettagliatamente come utilizzarle in pratica nei vostri programmi sia singolarmente che in combinazione tra di loro.**

Riprendendo la pubblicazione sulla rivista di articoli inerenti all'uso dei computer, sappiamo benissimo di suscitare due opposte reazioni tra i nostri lettori; infatti se da una parte facciamo contenti coloro che ormai da tempo si dedicano con passione a questa nuova branca dell'elettronica, dall'altra ci «inimichiamo» coloro che trovano questo argomento troppo ostico e non interessante.

Fra questi ultimi vi sono soprattutto i principianti i quali preferirebbero senz'altro veder pubblicato un semplice progetto da montare in una sera piuttosto che sentir parlare di istruzioni «Basic» in quanto ritengono a torto l'argomento computer troppo al di fuori delle proprie possibilità.

In realtà questo non è vero; infatti oggigiorno montarsi un computer in casa propria e farlo funzionare è forse più facile che realizzare un amplificatore di BF. Non solo, ma rinunciare a priori ad interessarsi a questa nuova branca dell'elettronica, potrebbe significare un domani ritrovarsi ad essere dei cittadini di serie «B».

Basti pensare che oramai non esiste azienda che non possegga un computer, grande o piccolo che sia, ed a breve scadenza non troveremo più il computer solo nelle banche o negli aeroporti, bensì lo troveremo dal macellaio, dal droghiere, in ferramenta, in libreria e forse anche in case private per tenere la contabilità domestica, per fare da segreteria telefonica o per gestire l'impianto di riscaldamento.

È assurdo quindi non voler neppure provare di capirci qualcosa, nè può essere una scusante il fatto che quando si parla di computer si tirano sempre in ballo dei termini astrusi e apparentemente incomprensibili come DEBUG, EDITING, MONITOR, DOS, DIRECTORY, CPU, SQR o altre cose di questo genere.

**Non fatevi spaventare** da questi vocaboli perché nessuno di essi nasconde dietro di sè cose incomprensibili o tanto difficili da non potersi capire: in realtà l'unica cosa «difficile» e l'unico vero ostacolo sono proprio questi termini anglosassoni che ricorrono con tanta frequenza e che noi italiani purtroppo, con la scarsa conoscenza delle lingue estere che abbiamo, non riusciamo proprio a digerire.

Una volta tradotti questi termini e spiegato in parole povere che cosa significano, tutto diviene così semplice da far perdere al computer quell'alone di mistero che tuttora lo circonda e che d'altronde è tipico di tutte le cose nuove anche se alle cose nuove, con i tempi che corrono, oramai dovremmo essere abituati e nulla dovrebbe più spaventarci.

Pensate che anche cinquant'anni fa, quando sulle riviste dell'epoca si iniziava a parlare di «supereterodina», di trasmettitori e ricevitori in modulazione di frequenza, di Hi-Fi ecc., sembrava di trovarsi di fronte a cose per soli «ingegneri» e nessun principiante osava sognare di poter capire e realizzare a sua volta tali circuiti.

Oggi invece, se un principiante leggesse tali articoli, non solo non avrebbe più nessun timore reverenziale, ma li considererebbe addirittura troppo semplici poiché qualsiasi persona che conosca i componenti elettronici e sappia stagnare, riesce oggigiorno a realizzare con estrema facilità trasmettitori, amplificatori, ricevitori e automatismi di vario genere e questo, consentiteci di dirlo senza falsa modestia, è merito anche di riviste come la

nostra che da tempo si sforzano di sfatare questi miti e di pubblicizzare il più possibile qualsiasi novità si presenti nel campo dell'elettronica.

Lo stesso fenomeno si verificherà tra qualche anno anche per i computers e quando questi, come da più parti si prevede, avranno invaso il mondo e saranno così familiari come lo sono oggi la radio e la televisione, coloro che adesso si lagnano perché dedichiamo troppe pagine della rivista a tale argomento andranno forse a ricercarsi in cantina o da un amico queste riviste per cercare di recuperare il terreno perduto.

L'unico inconveniente che presenta un computer rispetto ad un amplificatore o a un ricevitore è che non è sufficiente collegargli un giradischi o l'antenna per vederlo funzionare, bensì occorre sempre programmarlo per ottenere ciò che si desidera; al computer infatti, a dispetto di quanto si crede, occorre sempre insegnargli che cosa vogliamo che faccia utilizzando il solo linguaggio che esso è in grado di comprendere, linguaggio che per la maggior parte dei computers personal è appunto il Basic. Proprio per questo, se inizierete fin d'ora a leggere questi articoli, anche se non capirete tutto, un domani quando vi capiterà tra le mani un computer vi sentirete meno handicappati e forse, grazie alle vostre conoscenze, riuscirete a rimediare un posto di lavoro migliore dell'attuale.

Tenete presente che la maggior parte di queste istruzioni e comandi sono comuni a tutti i computers, quindi anche se non avete realizzato il nostro nè avete intenzione di realizzarlo in futuro, potranno sempre servirvi per programmare.

Per esempio sapere che per eseguire una moltiplicazione è necessario battere l'asterisco e non il 'x'; è una cosa basilare. Lo stesso dicasi per la divisione per la quale è necessario pigiare il tasto '/' e non il ':' come ci hanno insegnato tanto tempo fa alle elementari.

L'istruzione SQR non è una sigla da noi inventata o ricavata dagli antichi romani, bensì non è altro che l'abbreviazione della parola inglese «Square Root» utilizzata su tutti i computers per indicare la «radice quadrata», infatti scrivendo SQR(49) il computer esegue la radice quadrata di 49 e ci fornisce come risultato 7.

Allo stesso modo altre istruzioni come ABS, INT, PRINT, IF ecc. sono comuni a tutti i tipi di computers, non solo ma oramai talune di esse sono così familiari nel linguaggio di chi utilizza giornalmente tali macchine che non conoscerne il significato potrebbe voler dire rimediarci una magra figura anche in conversazioni da salotto, se non addirittura con la propria moglie o fidanzata.

Se volete rimanere al passo con i tempi queste sigle che ora ritenete così astruse dovranno diventare per voi familiari come lo sono oggi le sigle mA-V-pF-KW-mH ecc. quindi non snobbatele, bensì date loro il peso che meritano. Pensate che un domani il fatto di non sapere che ABS significa «valore assoluto» potrebbe mettervi in condizione di inferiorità rispetto ad altri che invece lo sanno, proprio come oggi è in condizione di inferiorità rispetto a voi chi non conosce l'elettronica e non sa che 1 W significa 1 Watt, oppure che 100 mH è una misura di impedenza e si legge 100 millihenry.

Voi oggi, se trovate scritto 100 pF, sapete già che non può trattarsi di una resistenza nè di una tensione, ma solo ed esclusivamente di un condensatore da 100 picofarad, tuttavia le prime volte avrete avuto qualche difficoltà ad imparare queste sigle, anche se ora tutto vi sembra così facile e vi sembra addirittura strano che qualcuno possa sbagliare. Lo stesso dicasi anche per il computer: quando a poco a poco avrete imparato tutte le sigle e fatto un po' di pratica, arriverete al punto di dire: «ma questa rivista non la smette mai di scrivere cose così ovvie? tutti ormai lo sanno che l'istruzione PRINT serve per stampare oppure che l'istruzione NEXT serve per chiudere un LOOP», e forse vi farete grandi con gli amici dicendo che per voi queste cose sono una «bazzecola».

## 1 - SUNTO DELLE ISTRUZIONI BASIC

(CONTINUAZIONE DAL NUMERO PRECEDENTE)

Nel numero precedente di NUOVA ELETTRONICA abbiamo pubblicato 7 tabelle con gli elenchi delle parole chiave, comandi, istruzioni e simboli facenti parte o del DOS o del BASIC.

Proseguendo nella descrizione delle varie voci, ci è sembrato più logico cambiare il contenuto delle tabelle 6 e 7, aggiungendone poi altre due. Così facendo, infatti, si ottiene il risultato di una maggiore chiarezza, in quanto è stata effetuata una migliore suddivisione per categorie e sono state aggiunte due tabelle. La prima (quella n. 8) contiene i COMANDI SPECIALI DA TASTIERA; la seconda (quella n. 9) elenca i COMANDI DI EDITING.

Per vostra comodità riportiamo comunque ancora tutte le tabelle, in modo da evitare ogni eventuale possibilità di confusione.

Detto questo, proseguiamo nella descrizione delle parole chiave del BASIC. Nel numero scorso avevamo completato la TABELLA 2; questa volta esamineremo il contenuto di quelle rimanenti, proseguendo dapprima con le tabelle che si riferiscono al BASIC, ed esaminando successivamente la TABELLA 1 del DOS.

Prima di cominciare, vi rammentiamo che il simbolo > sta ad indicare che si tratta di un COMANDO, mentre i numeri di linea (che sono solo d'esempio) precedono le ISTRUZIONI.

Oramai dovreste avere ben chiari questi concetti; in caso contrario vi consigliamo di riprendere in mano l'ultimo numero di NUOVA ELETTRONICA e di andarvi a rileggere quelle nozioni.

Ricordate anche che eventuali parentesi accanto alle parole chiave reappresentano le abbreviazioni possibili delle medesime.

## A - FUNZIONI DI STRINGA (TABELLA n. 3)

### ASC

Fornisce il valore decimale in CODICE ASCII del primo carattere della stringa argomento. (Vedere anche l'istruzione CHR $).

**30 PRINT ASC ("G")** — Visualizza il CODICE ASCII della lettera G, ossia il numero 71.

**70 PRINT ASC (F$)** — Visualizza il CODICE ASCII della prima lettera della stringa F$.

### CHR $

È l'inverso dell'istruzione ASC. Visualizza sul monitor il carattere che corrisponde al CODICE ASCII assegnato tra parentesi. (Vedere anche ASC).

**230 PRINT CHR $ (70)** — Visualizza la lettera F (il codice ASCII della lettera F maiuscola corrisponde al numero 70).

**710 LPRINT CHR $ (A + D*3)** — Stampa il carattere corrispondente al CODICE ASCII risultante dall'espressione entro parentesi; il valore di questa deve essere compreso tra zero e 255.

### CVD

Nel trattamento dei FILES RANDOM serve per ritrasformare dei dati in forma numerica in doppia precisione dopo che sono stati letti dal disco.

È l'inverso di MKDS. (Vedere anche CVI e CVS).

**3500 H # = CVD (E $)** — Trasforma la stringa E $ letta da disco nel numero H # in doppia precisione.

### CVI

È analoga alla CVD, ma riguarda i numeri interi: ripristina un numero intero in forma numerica dopo che la relativa stringa è stata letta da disco con GET.

È l'inverso di MKI $. (Vedere anche CVD e CVS).

**475 A% = CVI (A $)** — Trasforma la stringa A $ letta da disco nel numero intero A%.

### CVS

E simile alle due precedenti, ma si riferisce ai numeri in singola precisione: ripristina un numero in singola precisione in forma numerica dopo che la relativa stringa è stata letta da disco.

È l'inverso di MKS $. (Vedere anche CVD e CVI).

**1200 T! = CVS (DE $)** — Trasforma la stringa DE $ letta da disco nel numero a singola precisione T!.

### INSTR

Questa funzione effettua la ricerca di una STRINGA all'interno di un'altra.

Il numero corrispondente a questa variabile speciale dà la posizione di partenza della stringa da ricercare in quella assegnata; se il numero è zero significa che la stringa non è contenuta in quella principale.

**200 PRINT INSTR ("NUOVA ELETTRONICA", "ELETTRONI")** — Visualizza il numero 7, posizione d'inizio di ELETTRONI in NUOVA ELETTRONICA.

**210 PRINT INSTR ("NUOVA ELETTRONICA", "uova")** — Visualizza zero, perché la stringa minuscola 'uova' non è contenuta in NUOVA ELETTRONICA.

**925 POSIZIONE = INSTR (4,B$ ,HG $)** — Nella variabile numerica POSIZIONE è inserito il numero d'inizio della stringa HG $ all'interno di B$ ; la ricerca ha inizio a partire dal quarto carattere. In ogni caso il numero riportato fa riferimento all'inizio di B$ .

## LEFT $

Preleva un certo numero di caratteri alla sinistra della stringa indicata. (Vedere anche RIGHT$ e MID$).

```
45 CR $ = LEFT $ ("NUOVA ELETTRONICA", 5)
70 PRINT LEFT$(S1$ + S2$, C*5-D+2)
```

Pone nella variabile di stringa CR$ la stringa NUOVA.
Visualizza sul monitor la parte sinistra della stringa somma di S1$ e S2$, prendendo un numero di caratteri uguale al valore dell'espressione che figura dopo la virgola.

## LEN

Dà la lunghezza della stringa specificata.

```
20 PRINT LEN "NUOVA ELETTRONICA")
```

Visualizza il numero 17 (NUOVA = 5 + SPAZIO = 1 + ELETTRONICA = 11).

```
100 IF LEN (A$ ) < 23 THEN 1300
```

Se la lunghezza della stringa A$ è minore di 23, il programma prosegue alla linea 1300.

## MID $

Preleva un certo numero di caratteri in mezzo alla stringa indicata. (Vedere anche LEFT$ e RIGHT$).

```
60 F $ = MID $ ("NUOVA ELETTRONICA", 2, 4)
```

Pone nella variabile F$ la stringa UOVA, che si trova ad iniziare dal secondo carattere e prendendone 4.

```
90 MID $ ("NUORA ELETTTRONICA", 4, 1) = "V"
```

Corregge la quarta lettera, cambiando NUORA in NUOVA.

## MKD $

Nel trattamento dei FILES RANDOM converte un numero in doppia precisione in stringa prima di effettuare la registrazione sul disco con PUT.
(Vedere anche MKI $ e MKS $).

```
2000 LSET B $ = MKD $ (E#)
```

La variabile di stringa B$ conterrà la rappresentazione del numero in doppia precisione E# (ingombro di 8 bytes).

## MKI $

Analoga alla MKD$, ma riguarda i numeri interi: converte i numeri interi in stringa prima di inciderli su disco. (Vedere anche MKD$ e MKS$).

```
3270 C $ = MKI $ ((3*G-7)/R)
```

Nella variabile C$ và la parte intera del numero risultante dall'espressione tra parentesi (ingombro di 2 bytes).

## MKS $

Simile alle due precedenti: converte numeri in singola precisione in stringa prima della registrazione su disco. (Vedere anche MKD$ e MKI $).

```
550 IVA $ = MKS $ (IVA!)
```

La variabile IVA $ conterrà il numero in singola precisione IVA! (ingombro 4 bytes).

## RIGHT $

Simile a LEFT$, con la differenza che lavora alla destra della stringa indicata. (Vedere anche LEFT$ e MID$).

```
172 W $ = RIGHT $ ("MARIO ROSSI RAGIONIERE", 10)
```

Preleva alla destra della stringa indicata 10 caratteri; quindi la variabile W$ contiene la parola RAGIONIERE.

## STR $

Serve per convertire una costante numerica o una espressione in una stringa. (Vedere anche VAL).

```
815 NE $ = "NUOVA ELETTRONICA": NU$ = "NUMERO"
816 S $ = "      ": N = 17: RIMANENZA $ = ": "ESAURITO"
817 LPRINT NE $+S$ + NU $ + S $ + STR $ (N) + RIMANENZA $
```

Il programma dà sulla stampante la scritta NUOVA ELETTRONICA NUMERO 17: ESAURITO.

## STRING $

Serve per ottenere una stringa composta da un certo numero di caratteri tutti uguali.

```
10 Z $ = STRING $ (21, "*")
```

La stringa Z$ risulta composta da 21 asterischi.

### VAL

È l'inverso di STR $ : serve per tramutare in numero il contenuto di una stringa. (Vedere anche STR $).

```
390 A $ = "12" : B $ = "56" : PRINT VAL (A $) + VAL (B $) ; VAL (A $ + B $)
```

L'istruzione di stampa visualizza due numeri: 68 e 1256.

## B - FUNZIONI ARITMETICHE (TABELLA n. 4)

### &H

Utile per esprimere numeri in CODICE ESADECIMALE. (Vedere anche &O).

```
50 B = &H1A
```

Il valore 1A viene interpretato come espresso in codice esadecimale; nella variabile B viene messo il valore decimale 26.

### &O

Serve per esprimere costanti in CODICE OTTALE. (Vedere anche &H).

```
320 AB = &O35
```

Il valore 35 viene interpretato come espresso in codice ottale; nella variabile AB viene messo il numero decimale 29.
La lettera O può anche essere omessa.

### ABS

Fornisce il VALORE ASSOLUTO della variabile specificata nell'argomento.

```
40 EA = ABS (E)
```

Nella variabile EA viene depositato il valore assoluto di E.

### ATN

Fornisce il valore in RADIANTI dell'ARCOTANGENTE del parametro tra parentesi.

```
560 PRINT ATN(A)
```

Si ha la stampa del valore dell'arco (in radianti) la cui tangente vale A.
Per ottenere il valore in gradi moltiplicare per 57.29578.

### CDBL

Fornisce la rappresentazione in DOPPIA PRECISIONE dell'argomento specificato.
Il valore ottenuto contiene 17 cifre (solo 16 visualizzate), ma solo quelle contenute nell'argomento sono significative.
(Vedere anche CINT e CSNG).

```
140 A = 3.2 : A # = CDBL(A) : ?A #
```

La prima istruzione pone A = 3.2; la seconda chiede la rappresentazione di A in doppia precisione, tramite la variabile A #; la terza istruzione dà la stampa di A # (risultato 3.200000047683716: solo le 6 cifre di A sono esatte).

### CINT

Fornisce il NUMERO INTERO immediatamente inferiore al parametro indicato.
Il parametro deve essere compreso tra —32767 e + 32767.
(Vedere anche CDBL, CSNG, INT e FIX).

```
420 I = CINT(H)
770 PRINT CINT (12.4) ; CINT (—356.9801)
```

La variabile I assume il valore della parte intera di H.
Visualizza i seguenti numeri: 12 e —357.

### COS

Fornisce il COSENO dell'argomento, considerato espresso in RADIANTI.

```
610 CW = COS(W)
```

La variabile CW assume il valore del coseno dell'angolo in radianti W.

```
170 PRINT COS(2*3.1415926)
```

Si ottiene il valore del coseno di 2 pigreco (360 gradi), cioè 1.
Per ottenere il valore in gradi, moltiplicare l'argomento per 0.0174523.

### CSNG

Fornisce la rappresentazione in SINGOLA PRECISIONE del parametro indicato.

```
90 B! = CSNG (B#)
```

La variabile B! assume il valore in singola precisione di B #

## E

Fornisce il valore dell'ESPONENZIALE di 10; deve essere preceduto e seguito da numeri.

```
310 PRINT 3.2E4
```

Si ottiene il numero 32000.

## EXP

Fornisce la POTENZA DI 'e' (numero di Nepero = 2.7182818) con esponente uguale all'argomento.

```
80 A = EXP(4.2)
```

Nella variabile A viene depositato il numero 66.6863, cioè il numero di Nepero elevato all'indice 4.2.

## FIX

Fornisce la rappresentazione dell'argomento con tutte le cifre a destra della virgola eliminate. (Vedere anche CINT e INT).

```
610 PRINT FIX(28.71) ; FIX(—13.0073)
100 I = FIX(A)
```

Si ottengono i numeri 28 e —13.
Nella variabile I si ha un numero uguale ad A, ma senza le cifre dopo la virgola.

## INT

Fornisce il NUMERO INTERO immediatamente inferiore al parametro indicato.
Non è limitato come CINT. (Vedere anche CINT e FIX).

```
690 A% = INT(A)
```

La variabile A% viene ad assumere il valore del numero intero immediatamente inferiore al valore della variabile A.

```
150 PRINT INT (123456789123456789.123456789)
```

Si ottiene il numero 1.234567891234568 D + 17 (ossia il numero assegnato viene considerato in doppia precisione, arrotondato a 16 cifre e rappresentato con notazione esponenziale).

## LOG

Calcola il LOGARITMO NATURALE (ossia in base 'e') dell'argomento indicato.

```
20 PRINT LOG(342)
```

Stampa il numero 5.83481.

## RANDOM

Questa funzione serve per rinnovare il set di NUMERI CASUALI che sono utilizzati da una istruzione RND. (Vedere anche RND).

```
490 RANDOM
```

## RND

Fornisce un NUMERO PSEUDOCASUALE. (Vedere anche RANDOM).

```
30 A = RND(0)
```

Nella variabile A viene messo un numero decimale pseudorandom compreso tra zero e uno

```
10 PRINT RND(120)
```

Si ottiene la stampa di un numero intero pseudocasuale compreso tra 1 e 120.

## SGN

È una variabile speciale che vale —1 se il SEGNO dell'argomento è negativo, vale 0 se l'argomento è nullo, vale + 1 se l'argomento è positivo.

```
290 A = SGN(—28)
```

Assegna ad A il valore —1.

## SIN

Fornisce il valore del SENO dell'argomento, considerato in RADIANTI.

```
100 K = SIN(2)
```

Deposita in K il numero 0.909298 (valore del seno dell'angolo di 2 radianti).
Per calcolare in gradi, moltiplicare l'argomento per 0.0174533

## SQR

Fornisce la RADICE QUADRATA dell'argomento indicato.

```
1930 F3 = SQR(Z)
```

Assegna alla variabile F3 il valore della radice quadrata di Z.

```
1100 A = SQR(625)
```

Assegna alla variabile A il valore della radice quadrata del numero 625, cioè 25.

### TAN

Fornisce il valore della TANGENTE dell'argomento, considerato in RADIANTI.

**270 PRINT TAN(1)** — Visualizzata il numero 1.55741 (tangente dell'angolo di un radiante). Per calcolare in gradi, moltiplicare l'argomento per 0.0174533.

### ↑

Esegue gli elevamenti a potenza; deve essere preceduto e seguito da numeri.

**210 PRINT 3↑5** — Eleva alla quinta potenza il numero 3, cioè scrive il numero 243.

## C — FUNZIONI GRAFICHE (TABELLA n. 5)

### CLS

Cancella completamente lo schermo e posiziona il cursore in alto a sinistra.

**200 CLS**

### POINT

Serve per esaminare la CONDIZIONE GRAFICA (sul video) del punto specificato.

**380 T = POINT(7,12)** — Nella variabile T si ha il numero zero se il punto di coordinate 7 e 12 è in condizione RESET (spento); si ha il numero —1 se il punto è in condizione SET (acceso).

### RESET

Serve per SPEGNERE sul monitor il punto di coordinate specificate.
I limiti delle coordinate sono 0-31 per le X e 0-15 per le Y. (Vedi anche SET).

**910 RESET(31,11)** — Spegne l'ultimo punto della dodicesima riga ad iniziare dall'alto.

### SET

Utilizzato per ACCENDERE PUNTI sul video. Valgono le stesse considerazioni fatte in RESET. (Vedere anche RESET).

**10 SET(0,0)** — Accende il primo punto in alto a sinistra.

**NOTA** — Sia nel SET che nel RESET il punto è definito per lo spazio di un intero carattere. Punti più piccoli possono essere accesi o spenti con l'istruzione POKE.

## D - FUNZIONI SPECIALI (TABELLA n. 6)

### INP

Abilita il computer a ricevere dati dal canale specificato.
Serve per comunicare con l'esterno attraverso porte di INPUT-OUTPUT.
Il numero massimo dei canali è di 256. (Vedere anche OUT).

**50 MO = INP(73)** — Il valore introitato dal canale 73 viene posto nella variabile MO. Si può scrivere anche in esadecimale (&H49).

### OUT

Manda un valore nel canale specificato. Valgono considerazioni analoghe a quelle fatte per INP. (Vedere anche INP).

**70 OUT 37, 158** — Invia il valore 158 al canale 37. Entrambi i numeri devono essere compresi tra zero e 255. Si può usare anche la notazione esadecimale &H.

### PEEK

Preleva il valore decimale relativo al contenuto della locazione di memoria specificata. (Vedere anche POKE).

**320 A = PEEK(1024)**
**390 A = PEEK(&H400)** — Entrambe assegnano alla variabile A il contenuto (espresso in decimale) della locazione di memoria 1024, corrispondente all'esadecimale 400.

## POKE

Svolge la funzione inversa alla PEEK: deposita un valore alla locazione di memoria specificata. (Vedere anche PEEK).

**3810 POKE 11000, F**

Il valore numerico della variabile F viene memorizzato nella locazione di memoria 11000 (decimale). L'indirizzo e il valore possono essere espressi anche in esadecimale con la notazione &H (vedere PEEK).

## POS

Questa funzione fornisce un numero che rappresenta la posizione del cursore all'interno della riga video. POS deve essere seguito da un numero tra parentesi, di valore a piacere; normalmente si usa 0.

**380 PRINT "COMPUTER Z80" ;:**
**A = POS(0): PRINT A**

In A viene messo il numero 12, che è la posizione del cursore dopo che è stata scritta la frase indicata (notare che senza il punto e virgola dopo le virgolette il cursore sarebbe posizionato a capo della riga seguente, quindi A in tal caso varrebbe zero).

## TIME $

Fornisce una stringa che esprime la data e l'ora corrente.

**340 PRINT TIME $**

Stampa la data e l'ora nel seguente formato: M/GG/AA HH:MM:SS, vale a dire mese, giorno, anno, ora, minuti, secondi.

**NOTA** — Questa funzione attualmente restituisce sempre 00/00/00 00:00:00 per il motivo che manca nel computer il segnale di clock.
Tuttavia possono essere assegnate sia la data che l'ora con i comandi DOS: DATE e TIME.

## USR

Richiama una SUBROUTINE scritta in linguaggio macchina, precedentemente definita con una istruzione DEFUSR. (Vedere anche DEFUSR).

**2010 A = USR 3(2836)**

Il programma passa il controllo alla routine 3, che deve essere definita in precedenza; l'argomento viene passato alla routine. Il numero di routine può variare da zero a 9.

## VARPTR

Viene fornito un indirizzo (in decimale) corrispondente alla posizione in memoria della variabile tra parentesi.

**> PRINT VARPTR(A)**

Si ottiene un numero che è l'indirizzo a cui è depositato il valore di A.

## E - SIMBOLI CHIAVE (TABELLA n. 7)

Qui di seguito riportiamo la descrizione delle funzioni svolte dai vari simboli in qualche modo riconosciuti dal BASIC. Quando il programma incontra uno di questi simboli, è in grado di riconoscerli e di comportarsi di conseguenza: essi rappresentano perciò delle vere e proprie parole chiave. Alcuni di essi svolgono addirittura funzioni diverse a seconda del contesto in cui sono usati (vedere ad esempio i casi del segno = e delle parentesi).

## !

Quando un identificatore di variabile è seguito dal simbolo ! essa viene trattata in singola precisione (6 cifre significative).

**100 K! = 123456789 : PRINT K!**

Si ottiene per K! il valore 1.23457 E + 08: le 9 cifre assegnate sono state ridotte a 6 cifre significative; il numero 123456789 è memorizzato come 123457000.

## #

Una variabile seguita dal simbolo # viene considerata in doppia precisione (16 cifre significative).

**410 A # = F# / R #**

In A # viene posto il valore della divisione di F # per R#. Notare che se le variabili al secondo membro non fossero state in doppia precisione, le 10 cifre di A # successive alla sesta non avrebbero alcun senso. Per sincerarsene basta digitare A # = 3/5*5:PRINTA #. Si ottiene il numero 3.000000238418579.

## $

Una variabile seguita da $ viene trattata come stringa. Una stringa può contenere al massimo 255 caratteri.

```
770 A $ = "RICEVUTA BANCARIA"
```

Quando si chiede la stampa di A $ si ha la scritta indicata tra virgolette.

## %

Una variabile seguita dal simbolo % viene considerata come numero intero.
Il suo valore deve essere compreso tra —32767 e + 32767.

```
20 Z% = 2391.108
```

Nella variabile Z% viene messo il valore 2391, eliminando la parte decimale.

**NOTA** — Da quanto detto dovrebbe essere chiaro che, ad esempio, le variabili D! D # D% D $ sono per il computer 4 entità ben distinte.

## &

I numeri preceduti da questo segno sono interpretati come esadecimali (H) od ottali (D) a seconda della lettera che segue &.

```
300 N = &H30EF
```

Alla variabile N viene assegnato il numero decimale 12527 equivalente all'esadecimale 30EF.

```
830 J = &O345
```

Alla variabile J viene assegnato il numero decimale 229 equivalente al numero ottale 345.

## '

Quando un numero di linea è immediatamente seguito da questo segno, la linea stessa viene ignorata dal programma; utile per fare annotazione (è l'abbreviazione di REM).

```
60 'CALCOLO INTERESSE PASSIVO
```

La linea 60 ci ricorda, ad una lettura del listato, che le righe che seguono servono per calcolare l'interesse passivo.

## ""

Le virgolette indicano che ciò che le segue è il contenuto di una stringa.

```
150 C $ = "CARICO DI PUNTA"
```

Tutto ciò che è compreso tra le virgolette entra a far parte di C $ .

## ()

A seconda del contesto in cui compaiono, svolgono funzioni diverse.

```
10 G = (7 + A)/2
20 DIMK(34)
30 F = X(I,J)
40 M $ = MID$ (C $,3,5)
50 Y = COS(A*2)
```

Come si vede, le parentesi possono assegnare priorità di calcolo, oppure dimensionare vettori e matrici, o definire operandi e argomenti di funzioni.

## *

È uno degli operatori aritmetici; serve per effettuare le moltiplicazioni.
Come gli altri operatori aritmetici, può essere usato anche a livello di comandi diretti, per effettuare calcoli fuori dal programma.

```
490 A = B*C
```

Nella variabile A viene depositato il valore del prodotto di B per C.

```
> PRINT 4*12
```

Dà direttamente sul monitor il numero 48 (valore della moltiplicazione di 4 per 12).

## +

È uno degli operatori aritmetici e serve per fare le somme.

```
40 RE = A + 2
```

Nella variabile RE viene messo il valore della variabile A aumentato di 2.

```
> PRINT 234 + 18
```

Si ha sul monitor il valore della somma di 234 e 18, cioè il numero 252.

## ,

Serve per effettuare tabulazioni di stampa o per separare variabili.

```
30 PRINT A , B
```

I valori di A e di B sono spaziati tra di loro di 16 caratteri.

```
60 READ A , B
```

A e B sono letti uno dopo l'altro.

## —

É uno degli operatori aritmetici e serve per fare le sottrazioni.

```
930 D = Q — 3
```
Nella variabile D viene depositato il valore della variabile Q diminuito di 3.

```
> PRINT 10 — 2
```
Si ottiene la visualizzazione del numero 8 (differenza tra 10 e 2).

## .

Segna la separazione tra la parte intera e quella decimale di un numero.

```
40 A = 23.011
```
Alla variabile A viene assegnato il valore decimale 23 virgola 011.

## /

É uno degli operatori aritmetici e serve per fare le divisioni.

```
170 S = Q/4
```
Alla variabile S viene assegnato il valore della divisione di Q per 4.

```
> PRINT 10/3
```
Stampa il numero 3.33333 risultato di 10 diviso 3.

## :

Questo segno viene usato per separare le istruzioni le une dalle altre quando compaiono nella stessa linea.

```
300 A = 7 : B = F/4 : GOSUB 10000
```
La linea 300 contiene istruzioni multiple, separate tra di loro dai due punti.

## ;

Serve per stampare consecutivamente. Si usa anche nelle istruzioni di INPUT.

```
660 ?" IVA = " ; I ; "%"
```
Stampa la parola IVA seguita dal valore di I e dal simbolo %.

```
290 INPUT "PERCENTUALE IVA"; I
```
Aspetta un INPUT da tastiera e lo assegna ad I (dopo aver stampato la domanda PERCENTUALE IVA ?).

## < <= <> > >=

Sono operatori relazionali che servono per fare test di confronto.

```
230 IF A>B OR C<>6 THEN 1000
```
Se una delle due condizioni è verificata il programma prosegue alla linea 1000.

## =

Serve sia per l'assegnazione dei valori alle variabili che per effettuare test di confronto.

```
590 D3 = G + E*7.92
```
Assegna alla variabile D3 il valore dell'espressione posta a destra del segno = .

```
100 IF AA = 2 THEN B = AA
```
Se la variabile AA è uguale a 2, allora alla variabile B viene assegnato il valore di AA, cioè 2.

## ?

Fuori di stringa sostituisce la parola chiave PRINT.

```
10 ? "STAMPA CODICI"
```
Stampa il messaggio posto tra virgolette. Le virgolette a destra possono essere omesse se si trovano alla fine della linea di programma.

## @

In unione a PRINT serve per stampare in un determinato punto del monitor. Il numero specificato deve essere compreso tra zero e 511.

```
170 PRINT @ 300, "PROVA"
```
Scrive la parola PROVA iniziando dalla posizione 300.

## ↑

É uno degli operatori aritmetici ed effettua gli elevamenti a potenza.

```
400 WS = L ↑ F
```
Assegna alla variabile WS il risultato di L elevata a F.

```
> PRINT 5↑8
```
Stampa il numero 390625, cioè il valore di 5 elevato all'ottava potenza.

## F - COMANDI SPECIALI DA TASTIERA (TABELLA n. 8)

Ora esaminiamo il contenuto della TABELLA n. 8. Vi ricordiamo che quando una voce è posta tra parentesi significa semplicemente che bisogna premere i tasti indicati, vale a dire che non occorre digitare alcuna parola, ma è sufficiente premere il tasto o i tasti specificati per ottenere la funzione descritta.

**NOTA** — Il segno + posto tra due tasti, esempio CTRL + A, significa che occorre premere contemporaneamente il tasto CTRL e il tasto A.

### (II)

Quando si sta scrivendo alla tastiera, la pressione del tasto in alto a destra (recante il simbolo II), provoca uno spostamento verso sinistra del cursore; nello stesso tempo l'ultimo carattere che era stato digitato viene cancellato.

Per questa sua funzione il tasto viene chiamato BACKSPACE ( = spazio all'indietro). (Vedere anche CTRL + H).

### (CTRL + A)

Questa combinazione di tasti è equivalente a quella di premere i due tasti di BREAK contemporaneamente. Si usa quindi per interrompere l'esecuzione di un programma in corso.

### (CTRL + H)

La pressione contemporanea dei due tasti CTRL e H dà un risultato identico a quello ottenuto premendo il tasto BACKSPACE: si ha l'arretramento del cursore di una posizione. (Vedere anche il tasto II).

### (CTRL + I)

La pressione simultanea dei due tasti CTRL e I provoca l'avanzamento del cursore di 8 spazi. Alla prima pressione il cursore si sposta sul nono carattere alla seconda al diciassettesimo carattere alla terza pressione si sposta all'inizio della riga successiva. Tale funzione è utile per lasciare spazi bianchi o per effettuare dei tabulati.

### (CTRL + L)

Premendo questi due tasti si ha l'accensione pressoché istantanea di tutti i punti del monitor (16 righe per 32 colonne).

### (CTRL + P)

Con questo comando si ottiene una HARD COPY, vale a dire che tutto ciò che compare sullo schermo del monitor viene trascritto dalla stampante. Il comando opera anche a livello DOS.

### (CTRL + R)

Il comando ha l'effetto di passare dalla scrittura normale in scrittura REVERSE cioè in negativo. (Vedere anche CTRL + T).

### (CTRL + T)

Questo comando annulla quello di REVERSE, quindi serve per tornare dalla scrittura negativa a quella normale. (Vedere anche CTRL + R).

### (CTRL + X)

Quando si sta scrivendo qualcosa alla tastiera, la pressione contemporanea di questi due tasti provoca il ritorno istantaneo del cursore all'inizio della riga, con la cancellazione di tutto quello che si era scritto in precedenza.

### (CTRL + Y)

Con questa combinazione di tasti si passa alla scrittura allargata: ad ogni carattere battuto sulla tastiera automaticamente segue uno spazio libero.

(Vedere anche DEL).

### (SHIFT + ⊜ )

Si può usare per arrestare la lista di un programma sul video. Per continuare pigiare un tasto qualsiasi.

### (DEL)

Questo comando svolge due funzioni diverse e contemporanee: cancella lo schermo del monitor ed annulla nel contempo una eventuale condizione di scrittura allargata. (Vedere anche CTRL + Y).

### (SPAZIO)

Scrivendo alla tastiera, la pressione della barra dello spazio fa avanzare il cursore di una posizione, lasciando un carattere in bianco.

## G - COMANDI DI EDITING (TABELLA n. 9)

Con la descrizione delle voci della TABELLA 9 si chiude il sommario delle parole chiave del BASIC. Resterà da vedere solo il contenuto della TABELLA 1, relativa al DOS.

Delle TABELLE 2-3-4-5-6-7-8 abbiamo dato un veloce sunto, voce per voce; ciò per mettere in grado un buon numero di lettori di poter cominciare subito ad utilizzare il BASIC. Ci comporteremo invece in modo diverso per la TABELLA 9, in quanto un breve cenno delle varie voci non sarebbe sufficiente per capirne le funzioni svolte.

Per i comandi di EDITING daremo quindi subito una descrizione molto dettagliata.

Vi rammentiamo che la parola EDITING significa 'correzione dei programmi'.

Questa tabella contiene la spiegazione dei comandi concepiti appunto per facilitare la correzione di un programma già introdotto nel computer e che per qualche motivo necessita di essere cambiato in una sua parte. Per fare questo non è necessario ribattere il contenuto dell'intera linea, poiché l'applicazione dei comandi appositi permette di intervenire con rapidità solo nel punto desiderato e nel modo voluto.

Si tratta quindi di comandi molto interessanti e la loro conoscenza permette interventi rapidi di correzione.

Come è già accaduto per la tabella 8, anche per la 9 abbiamo a che fare con comandi resi operativi dalla semplice pressione di uno o più tasti: per questo e come già saprete i comandi che seguono sono chiusi tra parentesi.

Per spiegare la funzione dei vari tasti che costituiscono l'argomento di questa tabella, faremo riferimento costante al seguente programma esemplificativo nel quale volutamente abbiamo inserito degli «errori» per poterli poi correggere utilizzando i comandi che stiamo per esporre.

```
200 REM ———————— PROGRAMMA ESEMPLIFICATIVO
210 A = C*2,4
220 PRINT "IL PRODOTTO DI C PER IL COEfficIENTE 2.4VALE:" ; A
230 FOR I = 1 TO 23 : PRINT CF(I); : FOR J = 1 TO 2000 : NEXT J : NEXT I
240 CA = 328
250 LPRINT "CARICO                3280 kg."
260 LPRINT "LUNGHEZZA             3.5 m."
270 ENND
```

Poiché le spiegazioni che seguono fanno capo a questo esempio, vi conviene caricarlo nel computer così come lo vedete scritto, e provare ad editarlo seguendo gli esempi che troverete alle varie voci. Ribadiamo l'importanza di scriverlo esattamente come lo vedete. Ad esempio nella riga 250 ci sono 8 spazi tra la parola CARICO ed il numero 328; nella riga 260 tra la parola LUNGHEZZA e 3,5 m. vi sono 5 spazi: se ne mettete in numero diverso non vi torneranno poi i conti nelle spiegazioni successive.

Prima di proseguire diamo ancora due nozioni importanti. Innanzitutto rammentiamo che IL MONITOR VISUALIZZA ESCLUSIVAMENTE LETTERE MAIUSCOLE.

Ricordatelo sempre e sappiate che in effetti il computer memorizza le lettere sia in **minuscolo** che in **maiuscolo**; la cosa è evidente se listate con la stampante il programma che abbiamo proposto come esempio. Fate attenzione però a scriverlo esattamente come lo vedete stampato qui sopra: digitate maiuscole le lettere che tra virgolette sono maiuscole, ed altrettanto per le minuscole. In caso contrario molte delle spiegazioni che seguono perderanno il loro significato, vedi ad esempio alla riga 220 la parola COEfficIENTE.

Quando avremo pronta la scheda grafica, lo schermo sarà in grado di presentare anche le lettere minuscole; a quel punto tutto sarà più semplice e immediato. Rammentate anche che LA DISTINZIONE TRA MAIUSCOLE E MINUSCOLE HA IMPORTANZA SOLO PER TUTTO QUELLO CHE È SCRITTO TRA VIRGOLETTE. Tutto il resto, indipendentemente dalla posizione del tasto SHIFT, viene memorizzato maiuscolo. Fate qualche prova, utilizzando la stampante per evidenziare la differenza tra maiuscolo e minuscolo. Per iniziare è sufficiente che seguiate bene le istruzioni che vi daremo, eseguendole fedelmente passo passo.

L'altra cosa che volevamo dirvi è questa: quasi tutti i comandi che seguono potranno mettervi in imbarazzo quando li usate per la prima volta perché premendo i tasti che vi diremo, spesso lì per lì non succede nulla. Potremo dirvi, ad esempio, di premere i tasti 3 e poi 4 e poi C (vedere la voce (C) che tratta proprio questo caso); rimarrete sicuramente interdetti nel constatare che fino a quel punto le vostre battute non avranno sortito alcun effetto. In realtà non è affatto così, in quanto l'esecuzione di quelle istruzioni ha predisposto il computer a compiere correttamente i passi che seguono. Non vi resta allora che eseguire fedelmente le istruzioni che daremo: vedrete che dopo i primi, comprensibili imbarazzi, tutto andrà per il meglio.

Per farvi capire bene le voci della TABELLA n. 9 spieghiamo in dettaglio anche i comandi LIST, EDIT, (SPAZIO) e (ll), già visti a suo tempo nel sunto delle tabelle precedenti.

## LIST (L)

Ricorderete che serve per listare una parte del programma che si trova nella memoria del computer. Diciamo una parte perché normalmente un programma è abbastanza lungo da non trovare posto nelle 16 righe del monitor. Nel caso del nostro esempio, invece, potremo tranquillamente digitare la parola LIST e poi premere RETURN per vedere l'intero listato sul video. Il caso di scrivere LIST senza farlo seguire da alcun numero di linea è l'unico che non si può abbreviare: se provate a scrivere solo L e a premere RETURN il computer vi segnala errore di sintassi. Se invece provate a scrivere L230 e poi premete RETURN avrete il contenuto della linea di programma 230, e solo quello. Se provate a scrivere L50, non succede niente: infatti (supponendo ovviamente che abbiate introdotto il programma indicato) la linea 50 non esiste.

Andate poi a rivedervi i vari casi possibili d'uso del comando LIST, e già che ci siete leggete anche la voce LLIST che ha le stesse funzioni, ma con la differenza di visualizzare il listato del programma sulla stampante invece che sul monitor.

## ([)

Dopo aver chiesto il listato di tutto il programma o di una sua parte, se si preme questo tasto (il primo in alto a sinistra nella tastiera) si ha la visualizzazione della linea di programma precedente all'ultima scritta sul monitor. Ogni volta che si preme questo tasto (attenzione a non farlo col tasto SHIFT premuto, nel qual caso non si ha l'effetto desiderato ma solo la scrittura della parentesi quadra), si ha il listato della linea precedente; si procede cioè a ritroso nell'esame delle varie linee di programma.

Cosa succede se premiamo tante volte questo tasto fino ad arrivare alla prima linea del programma? Continuando a premerlo, non potendo arretrare ulteriormente, il computer continuerà a farci vedere il contenuto di quella linea, e questo tutto sommato è un comportamento abbastanza logico; come vedete, l'uso di questi comandi è semplice e spesso non conduce ad errori anche se vengono usati in un modo improprio.

Ricordate una cosa importante: questo comando è possibile solo se il tasto in questione è premuto per primo dopo una precedente pressione di RETURN. Vi accorgerete infatti che se premete prima qualche altro tasto e poi quello marcato [ non avrete la visualizzazione di una linea di programma, ma scriverete semplicemente il carattere [. A questo punto, se premete tante volte il tasto in alto a destra (quello marcato II) quante ne bastano per cancellare tutta la linea e lasciare solo il segno >, non riuscirete ancora a rendere efficace la funzione del tasto [. Per farlo dovete premere prima RETURN e poi, dopo che sarà apparso un nuovo segno >, premere il tasto [.

QUESTE ULTIME CONSIDERAZIONI VALGONO ANCHE PER I COMANDI ASSOCIATI AI TRE TASTI CHE SEGUONO.

## (■)

Questo tasto si trova a fianco di quello del RETURN; in alcune tastiere non reca alcun simbolo inciso, in altre è grigio, In altre porta il simbolo ^ oppure la scritta LF (LINE FEED). Naturalmente il colore o la dicitura messa sul tasto non fanno alcuna differenza per quello che diremo.

La funzione di questo comando è quella inversa al precedente: ad ogni pressione del tasto viene presentata la linea dl programma successiva all'ultima listata od editata. Per questo motivo il tasto viene anche chiamato LINE FEED (= avanzamento di linea), e noi d'ora in poi lo chiameremo con questo nome, per intenderci facilmente.

Per l'uso di questo comando valgono tutte le considerazioni fatte per quello precedente.

## (.)

Il comando associato alla pressione del tasto del punto è utile ed interessante. Se premete questo tasto dopo aver premuto RETURN, vedrete sul monitor il contenuto della linea appena editata o listata.

Pressioni ripetute di questo tasto sortiscono sempre lo stesso effetto, quindi la linea visualizzata ad ogni pressione è unica: quella precedentemente esaminata.

Vedremo tra poco che questa funzione risulta molto comoda in sede di correzione di un programma.

## (,)

Dopo aver editato o listato una linea di programma, provate a premere, sempre subito dopo aver premuto RETURN, il tasto della virgola. Vedrete ripetersi il numero di linea precedente, col cursore fermo subito dopo di esso. Ciò significa che siete passati a livello di EDITING nella linea in questione.

Alla voce seguente troverete chiarimenti a questo riguardo.

Due parole rivolte a chi deve imparare tutto partendo da zero: purtroppo gran parte delle nozioni in oggetto sono interlacciate tra di loro, quindi risulta praticamente impossibile affrontare un argomento e chiuderlo subito. Bisogna quindi che abbiate pazienza e che proseguiate anche se non tutto vi è chiaro al cento per cento. Le nozioni che troverete andando avanti serviranno anche per schiarirvi le idee su quelle già date; provando e riprovando arriverete al momento, sempre molto esaltante, in cui vi renderete conto di aver capito bene il funzionamento di una o più istruzioni. State certi che da quell'istante non dimenticherete più quei concetti.

## EDIT

Abbiamo già parlato della parola chiave EDIT nel sunto della tabella 2.

Ricorderete che serve per passare a livello di EDITING nella linea specificata; ricorderete anche che può essere abbreviata scrivendo solo la lettera E.

Provate allora a digitare E220 e premete poi il tasto RETURN: vedrete apparire il numero 220 seguito dal quadratino del cursore.

Con questa manovra vi siete portati a livello di EDITING nella linea 220.

Vedremo già alla voce successiva cosa significa in effetti editare una linea di programmazione preesistente.

## (SPAZIO) e (II)

Dopo aver fatto come detto sopra, provate a premere 5 volte il tasto SPAZIO: vedrete che ad ogni pressione il cursore scorre verso destra; alla prima battuta appare la lettera P, poi la R, poi la R e così via fino a che vedrete tutta la parola PRINT (primi cinque caratteri del contenuto della linea 220).

Il tasto marcato II è quello in alto a destra nella tastiera, e viene chiamato 'BACKSPACE' ( = spazio all'indietro), come già sapete.

A livello di EDITING, sposta il cursore verso sinistra di una posizione ogni qualvolta viene premuto. Il suo effetto è quindi l'inverso di quello che si ottiene premendo il tasto dello spazio. In entrambi i casi si vedono apparire e scomparire caratteri della riga che è in editing, ma il contenuto della medesima non viene cambiato.

Provate infatti a premere più volte il tasto BACKSPACE: il cursore va verso sinistra rimangiandosi, per così dire, i caratteri che trova sulla sua strada.

Quindi la pressione dei tasti SPAZIO e BACKSPACE serve solo per spaziare all'interno di una linea di programma, visualizzando il contenuto della linea stessa dall'inizio fino alla posizione del cursore.

Ora, per uscire dalla condizione di editing sulla linea 220, premete RETURN: vedrete riconfermato tutto il suo contenuto, in quanto avete fatto solo degli spostamenti coi tasti SPAZIO e BACKSPACE e non delle correzioni.

Supponiamo adesso di voler correggere la riga 210 del nostro esempio, dove nel numero decimale 2.4 è stata erroneamente scritta la **virgola** al posto del punto (come ricorderete, nella notazione anglosassone i numeri decimali hanno il punto come separatore della parte intera da quella decimale).

Scrivete allora sulla tastiera E210 e poi premete RETURN: sul monitor vedrete il numero 210 seguito, al solito, dal cursore.

Se adesso premiamo per sei volte consecutive il tasto SPAZIO, vedremo apparire in successione i vari caratteri che compongono la linea 210. Alla sesta battuta vedremo il carattere ',' che è proprio quello che dobbiamo correggere.

Vedremo più avanti come si effettua la correzione; ora esaminiamo più a fondo la funzione dei tasti SPAZIO e BACKSPACE.

Provate allora a premere più volte il tasto BACKSPACE: il cursore si sposta verso sinistra, cancellando via via i caratteri che incontra sulla sua strada. Se provate a premere poi nuovamente il tasto SPAZIO vi renderete conto che in effetti questi caratteri ricompaiono quando il cursore viene mosso verso destra. Portate ora il cursore completamente a sinistra, premendo più volte il tasto BACKSPACE fino a che il cursore stesso non ha cancellato tutti i caratteri tranne che il numero di linea 210; vi accorgerete che più a sinistra di così non si riesce a mantenerlo. Premete ora il tasto del numero 6, poi quello dello SPAZIO: il cursore fa istantaneamente un balzo in avanti di 6 caratteri e il monitor visualizza il contenuto della linea 210 fino alla virgola.

Provate ora a battere il numero 6 e il tasto BACKSPACE: il cursore si porta nuovamente all'inizio della linea.

Avrete visto che dopo la pressione del numero 6, in entrambi i casi non succede niente. La cosa è normale, perché il comando relativo diventa operante solo premendo anche il tasto seguente.

Avete allora sicuramente capito il funzionamento dei tasti SPAZIO e BACKSPACE: premendoli una volta si ha lo spostamento del cursore di un carattere; se prima di premerli si imposta un numero, il cursore si sposta avanti o indietro nella linea di un numero di caratteri uguale a quello indicato. Ovviamente se il numero che si specifica è tale da far superare la fine o l'inizio della linea, il cursore si porterà rispettivamente alla fine o all'inizio di essa; importante notare che in tali casi non si incorre in errore.

Se provate allora a scrivere 120 e poi premete la barra dello spazio, vedrete l'intero contenuto della linea 210, col cursore posizionato subito dopo il numero 4. Se adesso digitate il numero 21 e premete il tasto BACKSPACE, il cursore si riporta all'inizio della linea, cancellandone tutto il contenuto dal monitor. I numeri 120 e 21 sono solo esempi: se ne digitate di diversi il risultato non cambia, a patto che il numero introdotto sia superiore a 7, che è il numero di caratteri che compongono la linea in EDITING.

Per esercizio potete provare ad editare un'altra linea del nostro programma scelto come esempio; vi familiarizzerete così con le funzioni svolte dai tasti suddetti. Per poterlo fare ricordate però una cosa importante: fino a che non premete RETURN non uscite dalla condizione di editing; vedremo (ai tasti E e Q) due eccezioni a questa regola. Quindi per cambiare la linea che volete editare dovete premere RETURN, poi scrivere E seguito dal nuovo numero di linea da correggere ed infine dovete premere nuovamente RETURN per rendere operante l'ordine appena scritto di entrare in editing.

## (L)

Vi sarete resi conto che, dopo essere entrati in editing in una certa linea di programma, sarebbe utile poter vedere il suo intero contenuto prima di iniziare ad apportare le correzioni necessarie: vedendo tutta la riga, infatti, è molto facile rendersi conto di quello che si deve fare per correggere gli errori.

Allora, anziché agire sui tasti SPAZIO e BACKSPACE andando sù e giù per la linea allo scopo di vederne i caratteri che la compongono, è assai più comodo sfruttare il comando associato alla pressione del tasto L.

Provate infatti, dopo essere entrati in editing nella linea di programma 210, a premere innanzitutto il tasto L: vedrete che viene visualizzata l'intera linea. Successivamente il computer torna automaticamente in stato di editing sulla

medesima linea. Ora risulta più agevole fare spostamenti e correzioni al suo interno, poiché immediatamente sopra possiamo vederla tutta. Premete nuovamente RETURN per uscire dall'editing.

La pressione del tasto L risulta efficace anche se sono già stati premuti i tasti SPAZIO e BACKSPACE. Ciò significa che se dopo essere entrati in editing avete premuto un certo numero di volte sul tasto SPAZIO e sul BACKSPACE, la pressione del tasto L ottiene sempre l'effetto che abbiamo spiegato or ora.

## (C)

Fin qui abbiamo visto qualche comando che serve per entrare in editing o per spostare il cursore all'interno della riga chiamata. Vediamo ora il primo comando che ci permette di fare correzioni: il tasto C.

Supponiamo di aver richiamato la linea 210 (digitando nuovamente E210 e premendo RETURN) per correggere finalmente la virgola tra la cifra 2 e la cifra 4, sostituendola con un punto.

Dopo aver fatto come abbiamo appena detto, vedrete la scritta 210 seguita dal quadratino del cursore. Premete ora il tasto 5 e successivamente SPAZIO: il cursore si sposta verso destra di 5 caratteri fino a visualizzare la cifra 2. (Al solito la pressione del tasto del numero 5 non dà alcun effetto immediato, come abbiamo detto poco fa).

A questo punto fate così: premete il tasto C e poi il tasto del punto; vedrete che il cursore avanza di un passo, ed il punto avrà preso il posto della virgola. Premete RETURN per rendere definitiva la modifica appena fatta ed uscire dall'editing.

Siamo così riusciti a fare la correzione desiderata, intervenendo solo alla posizione interessata.

Si capisce, a questo punto, che la funzione del tasto C è quella di permettere correzioni. Anche per questo tasto esiste un comportamento analogo ai due precedenti: se prima di premere C premete un numero, il computer vi consente di correggere un numero di caratteri uguale a quello richiesto. Chiariamo con un altro esempio, in modo che non abbiate dubbi.

Passiamo ad editare la linea 220 per correggere da minuscole a maiuscole le lettere 'ffic' al centro della parola COEfficIENTE.

Scrivete quindi E220 e premete RETURN: appare la scritta 220 col cursore dopo il numero. Premete ora prima il tasto del 3 e poi quello del 4, formando così il numero 34. Premete successivamente il tasto SPAZIO: il cursore si sposta velocemente a destra, scoprendo i primi 34 caratteri della linea 220.

Gli ultimi tre caratteri visualizzati saranno COE, inizio corretto della parola COEfficIENTE.

Premete ora il tasto del numero 4 e poi il tasto C. Fin qui non accade nulla, ma voi proseguite ugualmente. Inserite quindi il blocco delle maiuscole premendo a fondo il tasto SHIFT LOCK e premete i tasti FFIC uno dopo l'altro. Così facendo avrete sostituito ai caratteri minuscoli errati i corrispondenti maiuscoli. Premete RETURN per uscire dall'editing.

Se per errore pigiate più di 4 tasti in correzione, vedrete che non succede niente: il computer accetta solo le 4 correzioni impostate e rifiuta le altre. In realtà questo non è sempre vero; se premete infatti qualcuno dei tasti elencati in questa tabella innescherete le funzioni ad essi associate. Meglio quindi che per ora non facciate questa prova, altrimenti potreste trovarvi in difficoltà.

Ricordate che sul monitor le lettere compaiono sempre in maiuscolo, ma il computer le memorizza maiuscole o minuscole a seconda che siano battute premendo contemporaneamente il tasto SHIFT, oppure no. Per rendervi conto se la correzione è effettivamente avvenuta, fate un LLIST del programma: vedrete che le lettere in questione sono cambiate da minuscole a maiuscole.

## (D)

Anche questo comando presuppone che si sia già entrati in editing in una certa linea di programma. In tal caso, la pressione del tasto D effettua la cancellazione del carattere successivo all'ultimo visualizzato.

Al solito, ci spieghiamo con un esempio tratto dal nostro programma.

Osservate la linea numero 270. Contiene solo la parola ENND, che è errata e va corretta in END. Come fare? Usando il comando associato al tasto D la cosa è molto semplice.

Entrate allora in editing nella linea 270 (scrivendo E270 e premendo poi RETURN); premete una volta il tasto SPAZIO per visualizzare il primo carattere della linea, cioè la lettera E. A questo punto premete il tasto D. Sul monitor vedrete la seguente scritta:

**270 E!N!**

la quale ha il seguente significato: il carattere N è stato eliminato dal contenuto originario della linea 270.

Per rendere definitivo l'effetto della correzione appena effettuata bisogna al solito premere il tasto RETURN.

Per controllare velocemente se la correzione è stata giusta, premete ora il tasto del punto: come sapete già, questo comando visualizza la linea appena editata. Vedrete che appare la scritta

**270 END**

Abbiamo quindi ottenuto il risultato desiderato.

Per questo comando valgono poi, ancora una volta, le considerazioni svolte per quelli precedenti. Infatti se prima di premere il tasto D impostate un numero, vedrete che alla pressione di D la quantità dei caratteri cancellati sarà uguale al numero impostato.

Tra poco vedremo l'applicazione pratica di questo caso.

## (I)

Il comando di editing associato al tasto I permette di inserire caratteri nel mezzo di una riga di programma già esistente.

Vediamo un'applicazione. Ci proponiamo di mettere uno spazio (nella linea 220 del nostro esempio) tra il numero 2.4 e

la parola VALE, che per ora sono attaccati. Il procedimento è il seguente: scrivete E220 e premete RETURN; premete poi i tasti dei numeri 4 e 7 e poi la barra dello spazio. Il cursore si sposta verso destra di 47 caratteri e sul monitor vedrete allora

**220 PRINT "IL PRODOTTO DI C PER IL COEFFICIENTE 2.4**

Premete ora il tasto della lettera I, poi la barra dello spazio e RETURN. Come vedete, avete inserito uno spazio tra il numero 2.4 e la parola VALE.

Non abbiamo ancora detto tutto sull'uso del tasto I in editing. Infatti dovete sapere che per annullare l'effetto dl inserimento da esso causato bisogna premere o il tasto RETURN, come nell'esempio appena fatto, oppure il tasto ESC (che è il primo a sinistra della seconda fila).

Anche qui, meglio di tante parole, valga un esempio pratico, condotto sempre sul nostro programma.

Se avete seguito tutte le modifiche apportate finora, la linea 220 risulta corretta; invece ora ci serve nuovamente sbagliata come all'inizio, quindi riscrivetela ancora a quel modo iniziando a scrivere il numero di linea 220 e proseguendo con tutti gli altri caratteri. Ricordate di premere RETURN alla fine. Per controllare il risultato scrivete LLIST220 così la stampante vi farà vedere il contenuto di quella linea, evidenziando le maiuscole e le minuscole. Proviamo ora a correggere ancora la parola errata COEfficIENTE. Dovete prima entrare in editing nella linea 220 (digitale E220 e RETURN); poi premete il tasto del 3, quello del 4 e la barra dello spazio. Il cursore va verso destra e scopre la linea fino al caratteri COE, come abbiamo del resto già visto in precedenza. Ora premete il tasto del 4 e poi il tasto D.

La linea appare ora scritta in questo modo:

**220 PRINT "IL PRODOTTO DI C PER IL COE!ffic!**

Conoscete già il significato di quella scrittura tra due punti esclamativi: il computer vi visualizza i caratteri che avete cancellato digitando 4D.

Evidentemente, a questo punto, la parola errata COEfficIENTE è diventata COEIENTE. Dovremo quindi inserire, al punto in cui si trova ora il cursore, i 4 caratteri eliminati battendoli però in maiuscolo. Per fare questo è sufficiente premere il tasto della lettera I, digitare FFIC in lettere maiuscole, e poi premere il tasto ESC. Se ora premete ripetutamente la barra dello spazo, vedete che alle lettere appena scritte fa seguito il gruppo di caratteri IENTE. La correzione è quindi finita.

Da notare che la pressione del tasto ESC ci ha fatto uscire dall'effetto di inserzione prodotto dalla pressione del tasto della lettera I.

Ricordate però che fino a che non premete il tasto RETURN le modifiche apportate non sono realmente avvenute a livello di programma memorizzato, ma solo sul monitor. Quando avrete premuto il RETURN, vedrete sì la linea corretta, ma in essa sono contenute anche le lettere cancellate e i punti esclamativi. Per visionare la linea appena editata è sufficiente premere il tasto del punto: eccola lì, per intero, corretta.

Al solito, per appurare che le lettere introdotte siano maiuscole, potete eseguire un listato della linea in questione (LLIST220): la stampante non mente!

## (K)

Il tasto della lettera K ha la seguente funzione: quando siete a livello di editing in una certa linea di programma, se portate il cursore in una parte intermedia e premete il tasto K e RETURN, vedrete che il contenuto della linea, a cominciare da quella posizione, viene cancellato fino alla fine. Si tratta quindi di una funzione del tutto simile a quella del tasto D, con la differenza che parte dalla posizione del cursore e va fino a fine linea.

Ribadiamo il fatto che la pressione del tasto K non causa nulla fino che non viene premuto per una prima volta il tasto RETURN. Alla seconda pressione di RETURN la cancellazione viene memorizzata ed uscite dall'editing. Si tratta quindi di una doppia sicura contro le cancellazioni involontarie.

Vedremo tra poco come fare se abbiamo premuto il tasto K per errore e vogliamo quindi annullare il suo effetto deleterio.

Come sempre, se dopo aver premuto RETURN volete vedere la linea modificata non dovete fare altro che premere il tasto del punto e sarete soddisfatti.

Il tasto K però può essere utilizzato anche in un altro modo, un po' diverso da quello appena esaminato. Provate ad entrare in editing nella linea 230. Scrivete poi 2S: (premete cioè prima il tasto del numero 2, poi quello della lettera S ed infine quello dei due punti). Così facendo il cursore va sul secondo carattere uguale ai due punti. Ora scrivete 2K: e vedrete la scritta:

**230 FOR I = 1 TO 23 : PRINT CF(I); !: FOR J = 1 TO 2000 : NEXT J !**

Conoscete già il significato di questa scritta: il computer ha cancellato dal contenuto originario della riga tutto quello che risulta racchiuso tra i due punti esclamativi. Ecco allora spiegato il secondo modo di utilizzare il tasto K in editing. Se si scrive un numero, poi si preme K, poi si batte un carattere, il computer cancella il contenuto originario della riga a cominciare dalla posizione in cui si trova il cursore; la cancellazione non arriva fino alla fine della linea, ma si arresta al primo carattere specificato se il numero che precede K è 1 (o se viene omesso), al secondo carattere uguale a quello indicato se il numero è 2, e così via.

Premete RETURN per uscire dall'editing. Il contenuto della linea 230 risulterà cambiato, secondo le modifiche apportate.

## (X)

La spiegazione della funzione del comando collegato alla lettera X è semplice: quando si è a livello di editing in una linea di programma, premendo il tasto della lettera X si va alla fine della linea e si può digitare quello che si vuole, attaccandolo così in coda a quanto già esisteva.

Esempio: portiamoci in editing nella linea 240 per aggiungervi una REM (vedere la relativa spiegazione al sunto della TABELLA 2).

Oramai dovreste essere in grado di entrare in editing, quindi d'ora in poi non descriveremo più le operazioni relative, evitando così di tediarvi inutilmente con la ripetizione continua delle stesse frasi.

Allorché sarete in editing sulla 240, portatevi alla fine della linea premendo appunto il tasto della lettera X. Il cursore si posizionerà dopo la cifra 8. Ora premete il tasto dello spazio, poi quello dei due punti. Digitate poi la parola REM e la scritta ————VALORE DEL CARICO————. Premete quindi RETURN e poi il tasto del punto per visualizzare la linea appena editata; sul monitor comparirà la scritta

**240 CA = 328 :REM————VALORE DEL CARICO————**

Con queste operazioni abbiamo trasformato la linea 240 in una riga di programma contenente una annotazione che può essere utile in qualsiasi momento per capire meglio il programma leggendone il relativo listato.

Anche la funzione associata al tasto X è annullata dal tasto ESC. Infatti provate ad entrare in editing in una riga qualsiasi e premete poi il tasto X; spingete ora il tasto ESC e provate a battere ripetutamente prima il tasto ll (BACK-SPACE) e poi il tasto dello spazio: vedrete che il contenuto della linea è rimasto invariato fino alla fine. Se ripetete il tutto senza premere ESC vi accorgerete che l'arretramento del cursore causato dal tasto BACKSPACE ha cancellato i caratteri interessati, che infatti non riemergeranno più quando premete il tasto dello SPAZIO.

Anche ad un eventuale errore in questo senso si può porre rimedio, come vedremo tra poco.

## (S)

Pure il comando associato al tasto della lettera S è facile da apprendere. Esso serve per spostarsi rapidamente all'interno della linea in editing, prendendo come riferimento per questi movimenti i caratteri stessi presenti nella riga di programma. Vediamo un esempio.

Supponiamo di voler cambiare nella linea 230 il numero 2000 in 3000.

Se avete seguito alla lettera le nostre istruzioni, il contenuto di quella linea non è più quello originario a causa degli interventi già fatti.

Dovrete quindi riscriverla seguendo il listato dato all'inizio.

Per fare le correzioni dette poco fa sarà sufficiente sostituire alla cifra 2 la cifra 3. Conoscete già un paio di modi adatti ad ottenere il risultato voluto; ora vedremo come arrivare molto velocemente sul numero 2000. Dopo essere entrati in editing nella linea 230, premete il tasto del numero 2, poi quello della lettera S e per ultimo ancora quello del numero 2. Appena avrete premuto il terzo tasto, vedrete che il cursore si sposta velocemente verso destra e lascia leggere la seguente scritta:

**230 FOR I = 1 TO 23 : PRINT CF(I) : FOR J = 1 TO**

Cosa è avvenuto? Scrivendo 2S2 abbiamo ordinato al computer di posizionare il cursore sul secondo carattere uguale a 2. Infatti sul monitor il quadratino del cursore si trova ora uno spazio oltre la scritta TO, esattamente sovrapposto alla prima cifra del numero 2000, cioè al numero 2; ovviamente in quella posizione esso ricopre la cifra 2, che però è la prima interessata ad una eventuale correzione. Adesso quindi siamo pronti per fare la modifica del 2 in 3, premendo il tasto C, poi il numero 3, e per ultimo RETURN.

Allora si può dire che la funzione esplicata dal tasto S serve per cercare un carattere all'interno di una linea di programma; per applicarla bisogna dare prima un numero, poi premere S, poi dare un carattere. Questo carattere è quello da ricercare. Però nella linea quel carattere può comparire più di una volta; allora il numero da specificare prima della lettera S serve per fermarsi sul primo (se il numero è 1), sul secondo (se è 2), e così via.

Entriamo ancora una volta in editing nella linea 230.

Se scriviamo 2S = ci portiamo sul secondo carattere uguale a ' = ' (quello di J = 1). Se invece avessimo scritto 4S: il cursore si sarebbe fermato sul quarto carattere uguale a ':', cioè dopo la scritta NEXT J. Dovrebbe essere tutto chiaro, a questo punto. Casomai il carattere richiamato non esista nella linea in editing, il cursore va alla fine di essa, come pure se ordiniamo il posizionamento sul quarto carattere di un certo tipo, e di quei caratteri ce ne sono meno di quattro, ad esempio due soli.

A proposito di questo comando c'è da dire ancora una cosa molto importante. Il carattere da ricercare, nel caso si tratti di una lettera, deve essere battuto in maiuscolo o in minuscolo a seconda di come compare nella linea di programma richiamata, altrimenti la ricerca darà risultati errati.

Se, ad esempio, provate ad editare la linea 240, quando è ancora scritta nel modo errato (parola COEfficIENTE), avrete che, scrivendo 1Sf, il cursore va sul primo carattere 'f' (lettera f minuscola) come desiderato. Se invece scrivete 1SF, il cursore va a fine riga in quanto non trova alcun carattere 'F' (lettera F maiuscola).

Un'ultima cosa: scrivendo (sempre nel caso della linea 240) S" il cursore si posiziona sul primo carattere 'virgolette' che trova; se scriviamo ancora S" esso va al seguente carattere 'virgolette'. Ciò significa che, nel caso si ometta il numero che precede la lettera S, detto numero viene assunto automaticamente uguale a 1.

Vi consigliamo una volta ancora di fare molte prove per conto vostro. Fatene parecchie e di tutti i tipi. Ricordate di premere RETURN per uscire dall'editing.

## (H)

Il tasto della lettera H permette di cancellare il resto della riga (rispetto alla posizione del cursore, ovviamente), e di inserire da quel punto in poi nuovi caratteri).

Supponiamo di dover cambiare il contenuto della linea 250, scivendo 'tonnellate' al posto di 'kg.'. Potremmo fare nel modo seguente.

Dopo essere entrati in editing su quella riga, scriviamo Sk (k minuscolo!). Il cursore va sul carattere 'k' della scritta 'kg.'.

Come sempre, non vedremo il carattere 'k', ma ci siamo sopra. Ora premiamo il tasto della lettera H e poi scriviamo tonnellate" e premiamo successivamente RETURN. Il contenuto della linea è diventato il seguente:

**250 LPRINT "CARICO 328 tonnellate"**

Naturalmente questo non era l'unico modo di effettuare la correzione voluta. Vi sarete resi conto che spesso ogni modifica ad una linea di programma può essere fatta in diversi modi; sta alla vostra abilità scegliere il più veloce, volta per volta. Gli esempi fatti servono solo per spiegare le funzioni svolte dai vari comandi di editing.

## (E)

Questo comando di editing permette di passare a livello di comandi diretti. In pratica, mentre si sta editando una linea di programma, se si preme il tasto della lettera E si ha un effetto uguale a quello ottenuto premendo RETURN. Le correzioni fatte sono memorizzate; l'eventuale parte di riga non visualizzata non viene però presentata sul monitor. Per vedere il contenuto dalla linea appena editata occorre quindi premere il tasto del punto, come già sapete.

Da notare che la pressione del tasto E dà il risultato spiegato solo se non ci si trova sotto l'effetto di altri comandi. Vale a dire che se in precedenza è stata attivata una delle funzioni associate ai tasti I-H-X-C, la pressione del tasto E in queste condizioni ha come risultato quello di scrivere la lettera E di seguito al testo già scritto.

Prendiamo ad esempio la linea 260, e proponiamoci di correggere LPRINT in PRINT. Entriamo allora in editing nella linea 260.

Premendo una volta il tasto della lettera D otteniamo:

**260 !L!**

il cui significato vi è certamente familiare ormai. Provate ora a premere il tasto della lettera E: la scritta precedente non cambia, ed il monitor ci mostra che il sistema è tornato a livello di comandi diretti. Nella riga successiva infatti appaiono i simboli >. che significano proprio questo. Avrete senz'altro la sensazione di aver cancellato tutto il contenuto della linea 260; invece non è così: al solito, basta premere il tasto del punto per convincersene.

Il prossimo obiettivo che vogliamo raggiungere è quello di sostituire alla scritta 'm.' della stessa linea la nuova scritta 'metri'.

Prima però riscrivetela come era in origine. Se siete bravi ci riuscirete senza ribatterla tutta, aggiungendo solo la lettera L di LPRINT or ora cancellata. Se invece non ci riuscite, non vi resta che riscriverla tutta: per questa volta noi non vi aiuteremo, in modo da stimolarvi a procedere da soli.

Allora rientrate in editing nella stessa riga di programma (il modo più veloce è sicuramente quello di premere il tasto della virgola, se ricordate).

Successivamente premete i tasti del 2 e del 6, poi quello dello SPAZIO.

In tal modo il cursore si sposta a destra di 26 caratteri e, se avete scritto la riga come nell'esempio fornito all'inizio (anche il numero degli spazi è importante se volete che tutto quello che diciamo coincida col vostro caso), si posiziona sulla lettera minuscola 'm', che segue il numero 3.5 e lo spazio. Premete ora per due volte il tasto della lettera D, e sul monitor vedrete:

**260 PRINT "LUNGHEZZA 3.5 !m!!.!**

Ora potete premere il tasto della lettera I ed inserire la parola 'metri'. Se, subito dopo averla scritta, provate a premere il tasto della lettera E pensando di ottenere un risultato analogo a quello di prima, rimarrete delusi. Infatti, dato che siete a livello di inserzione (per aver premuto il tasto della I), non fate altro che scrivere la lettera E. Cancellatela allora premendo per una volta il tasto BACKSPACE; adesso premete il tasto ESC per uscire dal comando di inserimento. Se riprovate a premere, a questo punto, il tasto della lettera E otterrete finalmente la funzione ad esso associata. La linea risulterà modificata come voluto, con la parola 'metri' al posto della scritta 'm.'. Per verificarlo premete il tasto del punto.

## (Q)

Il comando che si inserisce premendo il tasto della lettera Q è analogo a quello appena visto, associato alla lettera E. La differenza consiste nel fatto che premendolo si torna a livello di comandi diretti, ma senza rendere effettive le correzioni eventualmente apportate alla linea che era in editing.

Come esempio, provate a ripetere quello del caso precedente (prima dovrete riscrivere la linea 260 del programma, perché era stata corretta).

Fate le correzioni dette poco fa, premendo però il tasto della lettera Q anziché quello della lettera E. Vedrete, premendo il tasto del punto, che le correzioni fatte non sono state memorizzate dal computer.

Questo comando, contrariamente a quanto può sembrare, esplica una funzione molto importante. Infatti vi permette di salvare il contenuto originario di una linea di programma, qualora vi siate sbagliati nel correggerla o decidiate di non correggerla affatto dopo che avete già iniziato a farlo. La semplice pressione del tasto Q vi farà uscire dal livello editing e farà conservare alla linea il suo precedente contenuto.

Leggete a questo proposito la spiegazione della voce seguente: vi troverete descritto un secondo modo per uscire dall'editing senza rendere effettive le correzioni fatte.

## (A)

Il comando associato alla pressione della lettera A ottiene appunto il risultato di riportare il cursore all'inizio della linea editata, cancellando tutte le modifiche fatte precedentemente.

La differenza tra il tasto Q ed il tasto A consiste nel fatto che il primo fa uscire dalla condizione di edlting, mentre il secondo no.

Riprendiamo, come esempio, quello fatto poco fa: decidiamo di modificare nuovamente il contenuto della linea 260, per scrivere 'centimetri' al posto di 'metri'. Passiamo in editing e, come prima, ci portiamo all'inizio della parola 'metri' scrivendo prima 26 e poi schiacciando il tasto dello spazio. Ora cancelliamo la parola 'metri' scrivendo 5D ; vedremo infatti la scritta:

**260 PRINT "LUNGHEZZA 3.5 !metri!**

Pigiamo quindi il tasto della lettera I e scriviamo 'centimetri'.

Giunti a questo punto, se ci pentiamo della modifica che stiamo facendo e decidiamo di editare la 260 in modo diverso, possiamo annullare le modifiche già fatte e rientrare in editing nella stessa linea di programma semplicemente uscendo prima dalla condizione di inserzione premendo il tasto ESC, poi schiacciando il tasto della lettera A.

Sul monitor appare il numero 260 seguito dal cursore: ciò significa che siamo tornati all'inizio della 260, sempre a livello di editing. Per vedere quale sia il contenuto attuale della linea, premete il tasto della lettera L, comando che vi dà, come già detto, il contenuto di tutta la riga: vedrete che esso è identico a quello precedente all'ultima correzione.

La pressione del tasto A ha quindi ordinato al computer di ignorare le correzioni fatte in precedenza e di rientrare nella riga 260 mantenendo la condizione di editing. Con la pressione di RETURN potete ora confermare il contenuto della linea stessa.

Come nel caso precedente, la pressione del testo A quando ci si trova ancora a livello di inserzione provoca solo la stampa del carattere A e non la funzione appena descritta.

La conoscenza del funzionamento di questi ultimi due comandi, ripetiamo, è importante perché essi permettono di annullare eventuali errori avvenuti durante la correzione di una linea.

## (CTRL + J)

Questo comando è un doppione del tasto del LINE FEED (quello accanto al RETURN) già descritto. Quindi la pressione simultanea di questi due tasti serve per visualizzare la linea di programma successiva all'ultima editata o listata. (Vedere anche il tasto ■).

## (ESC)

Parlando di EDITING, il tasto ESC assolve a due funzioni ben distinte.

La prima è la seguente: se provate a premerlo quando siete a livello di comandi diretti, vedrete che appare sempre la prima linea del programma che si trova in memoria. Ogni pressione di quel tasto visualizza sempre e solo il contenuto della prima linea di programma. È comodo specialmente quando non si conosce il numero assegnato a quella linea, in quanto non si saprebbe quale numero introdurre per listarla.

L'altro uso del tasto ESC è quello che abbiamo già esposto nelle descrizioni precedenti: serve per uscire dalle funzioni associate ai tasti I e X.

Quando si è premuto uno di quei due tasti ci si è portati a livello di inserzione (nel mezzo della linea con I e alla fine di essa con X); premendo il tasto ESC si esce da quella condizione, rendendo così possibile il funzionamento degli altri comandi di editing, come abbiamo già visto sopra.

## (RETURN)

La pressione del tasto RETURN memorizza effettivamente le correzioni fatte: inoltre serve per rendere operante l'effetto del comando associato alla lettera K, se ci avete fatto caso quando lo abbiamo trattato.

Con ciò si conclude la descrizione dettagliata della TABELLA n. 9 relativa ai comandi di EDITING.

Anche per queste nozioni possiamo ripetere le cose già dette altre volte, ma pur sempre valide.

Non fatevi scoraggiare dall'apparente complessità delle istruzioni; il loro uso continuo ve le renderà semplici ed immediate. Non cercate poi di badare più alla quantità piuttosto che alla qualità del vostro apprendimento.

Affrontate quindi tutto con calma, rileggendo e ripetendo gli esempi che non avete ben capito. Sarebbe poi quantomai consigliabile che provaste ad eseguire anche altri esercizi o, meglio ancora, che provaste a fare qualche semplice programma.

Solo la pratica reale della programmazione potrà darvi la chiarezza di idee che non potete pretendere di avere dopo la prima lettura di queste righe.

Di una cosa siamo certi: queste cose sono più difficili da dire che da fare; quindi se siete momentaneamente in difficoltà consolatevi. La nostra fatica nello stendere queste istruzioni non è stata inferiore a quella che state facendo voi per cercare di apprenderle!

## COMANDI DOS (TABELLA n. 1)

Per la descrizione dei comandi DOS il contenuto della TABELLA 1, verrà esaminata voce per voce, in maniera abbastanza dettagliata.

Ricorderete certamente le considerazioni svolte in precedenza sui livelli operativi (LIVELLO DOS e LIVELLO BASIC); ribadiamo ancora una volta che si tratta di concetti molto importanti. Invitiamo pertanto coloro che ancora non li avessero ben chiari a rileggere quanto abbiamo già detto in merito sul numero precedente di NUOVA ELETTRONICA.

Ognuna delle parole chiave contenute nella TABELLA 1 rappresenta un comando DOS.

Ciascun comando svolge una ben determinata funzione, ed è accessibile in modo diretto ed immediato allorché ci si trovi ad operare a livello DOS.

Ad esempio, se a livello DOS scriviamo FORMAT e poi premiamo il tasto RETURN, otteniamo l'avvio della procedura di formattazione dei dischetti; non a caso abbiamo scelto quest'esempio, in quanto già nell'articolo precedente avete imparato l'uso corretto di questo comando, dato che noi vi abbiamo insegnato ad usarlo proprio operando a livello DOS.

Il motivo per cui ricordiamo ora l'uso del comando FORMAT è molto importante.

Desideriamo infatti chiarire definitivamente un concetto già esposto in precedenza, vale a dire la possibilità di usare i comandi del DOS anche quando ci si trova a livello BASIC.

Per intenderci meglio riprendiamo l'esempio del comando FORMAT.

Se desiderate formattare un dischetto partendo dal livello BASIC, non siete costretti a passare a livello DOS uscendo dal livello BASIC a cui vi trovate, ma è sufficiente che scriviate

**CMD"FORMAT"**

Vedrete che il computer parte con la procedura di formattazione, esattamente come se avessimo impartito il comando al livello DOS.

In effetti noterete tre differenze. Innanzitutto vi accorgerete che dopo aver scritto quanto detto e premuto il tasto del RETURN il computer resta inattivo per circa un secondo, anziché partire immediatamente con l'esecuzione del comando. Poi noterete che il floppy numero zero, dove ovviamente deve essere inserito il disco DOS-BASIC, si mette in moto; il led acceso segnala che sta avvenendo uno scambio di dati tra il disco e la memoria del computer.

Anche questo intervallo di tempo risulta essere un po' più lungo di quello che sarebbe operando direttamente al livello DOS.

La terza differenza è la seguente: al termine del processo di formattazione il computer torna automaticamente a livello BASIC.

Provate ad eseguire realmente una formattazione in questo modo; vedrete che quando essa è terminata, sul monitor riappare la scritta

**READY**

>.

la quale indica appunto che vi trovate nuovamente al livello BASIC.

Tutto quello che abbiamo detto sul comando FORMAT vale ovviamente anche per qualunque altro comando DOS.

DIGITANDO CMD"....." (al posto dei puntini dovrete riportare un comando DOS) MANDIAMO IN ESECUZIONE QUESTO COMANDO DOS; QUANDO ESSO È TERMINATO IL COMPUTER TORNA SEMPRE AUTOMATICAMENTE AL LIVELLO BASIC, SENZA PERDERE IL CONTENUTO DELLA MEMORIA CENTRALE.

Comprenderete l'importanza di tutto questo: immaginate infatti di essere al livello BASIC, con un programma caricato in memoria. Ad un certo punto (per rimanere sempre nell'ambito dell'esempio del comando FORMAT) decidete di salvare il programma su un dischetto prima di spegnere il computer.

Se il disco DOS-BASIC ha il nastro di protezione, oppure se risulta già riempito con altri programmi, sarete costretti a salvare il vostro programma su un altro disco. Immaginate allora di non avere un disco vergine formattato.

Cosa dovreste fare supponendo di non avere la possibilità di usare i comandi DOS partendo dal livello BASIC? Dovreste passare al livello DOS, scrivendo

**CMD"S"**

e dopo la scritta DOS READY potreste regolarmente eseguire la vostra formattazione.

Così facendo è però accaduto un grosso guaio: passando da BASIC a DOS avete perduto per sempre il contenuto della memoria centrale del computer. Infatti, se al termine della formattazione tornate al livello BASIC (scrivendo BASIC e premendo RETURN), vi accorgerete appunto che il programma che avevate in precedenza scritto ora non c'è più. Il passaggio da BASIC a DOS comporta sempre la perdita del BASIC e di tutto quello che il BASIC stesso gestisce.

Tutto questo era già stato detto succintamente alla pagina 102 del numero precedente di NUOVA ELETTRONICA, alle voci CMD"funzione DOS" e CMD"S".

Con queste ulteriori delucidazioni speriamo di aver chiarito bene tali concetti anche ai meno esperti.

Ora passiamo alla descrizione delle voci della TABELLA 1; eseguite realmente gli esempi che esporremo. Serviranno a fugare gli ultimi eventuali dubbi rimasti.

## APPEND

Il comando APPEND realizza l'unione di un FILE con un altro.

Per FILE si intende o un programma o un archivio dati, registrati su disco.

I FILES possono essere registrati sia in FORMATO ASCII che in FORMATO COMPATTATO.

Innanzitutto spendiamo qualche parola a proposito della dicitura ASCII.

Come in tanti altri campi tecnologici, anche in quello dei computers si è sentita la necessità di standardizzare; in pratica però ogni costruttore adotta le soluzioni che più gli aggradano, costruendo così un sistema che risulta incompatibile con gli altri. Chi conosce un po' il settore sa che ogni Ditta adotta un proprio sistema per formattare i dischi, un proprio BASIC, un proprio DOS, una propria tastiera, un proprio BUS, e così via. I vari costruttori non si sono accordati che su poche cose: le dimensioni dei dischi (minifloppy, floppy, dischi rigidi), alcuni tipi di interfacce standardizzate (RS232, IEEE448, ecc.) e, in teoria, i CODICI ASCII. Di cosa si tratta?

In pratica la tabella dei CODICI ASCII è formata da due colonne: in quella di sinistra si hanno i numeri progressivi da 0 a 255, in quella di destra ci sono i corrispondenti caratteri. Ad esempio, al CODICE ASCII numero 43 (in decimale!) corrisponde, per tutti i computers, il carattere ' + '; al codice 65 corrisponde il carattere 'A'; al codice 97 corrisponde 'a', e così via.

Però...però questo solo in teoria! Basta dare un'occhiata alle tabelle delle istruzioni di qualsiasi calcolatore per rendersene conto.

Ad esempio, consideriamo i modelli 80K, 80B e PC3201 della SHARP; ebbene, al codice ASCII 161 corrispondono rispettivamente per i vari modelli citati il carattere 'a' minuscolo, il carattere '!' in reverse, oppure un carattere dell'alfabeto giapponese! Come vedete, neanche lo stesso costruttore rispetta un certo standard.

La cosa non deve, alla fin fine, suscitare troppa meraviglia; infatti ogni elaboratore elettronico nasce per soddisfare certe esigenze di qualità, di prezzo e d'uso, e quindi la sua architettura risente di tutte queste cose.

In pratica, quasi tutti i personal computers attuali rispecchiano i codici ASCII dal numero 32 al numero 90, cioè dal carattere 'spazio' al carattere 'z' (zeta minuscola). In quell'intervallo sono compresi due alfabeti interi, uno maiuscolo ed uno minuscolo (ovviamente anglosassoni di 26 lettere), più i vari segni grafici di interpunzione e d'uso matematico (;: = $% >, ecc.).

La tabella dei CODICI ASCII del nostro computer è riportata alla fine di questo articolo. Come vedete, essa parte dal codice 32 ( = spazio) e finisce al codice 223 ( = freccia verso sinistra, in reverse).

In realtà esistono anche i codici da 1 a 31 e da 224 a 255; essi contengono spazi o altre funzioni che ora sarebbe troppo lungo spiegare. Potete comunque cercarvele da soli, utilizzando la parola chiave CHR$ già descritta nel sunto del BASIC.

Dopo questa lunga parentesi, necessaria a chi non conosce già bene il mondo dei calcolatori elettronici, proseguiamo con la descrizione del comando APPEND.

Vediamo ora di capire cosa significa "scrivere in FORMATO ASCII" e, al contrario, "scrivere in FORMATO COMPATTATO".

Ne avevamo già parlato brevemente alla pagina 108 del numero precedente della rivista, alla parola chiave SAVE, e a pagina 106, alla parola chiave MERGE.

Quando il computer va a scrivere sul dischetto, lo fa usando di norma un metodo che gli permette di risparmiare spazio: scrive in FORMATO FORMATTATO.

Facciamo un esempio. Se il calcolatore deve scrivere su disco la seguente parte di programma.

**130 PRINT "RIMANENZE DI FINE ANNO = " ; P**

non va ad incidere sul disco, uno dopo l'altro, i BYTES corrispondenti ai vari caratteri (1-3-0-spazio-P-.....ecc.), ma usa una sua "lingua" tutta particolare.

Questa lingua è formata da caratteri alfanumerici (ossia da numeri, lettere e simboli vari come il punto, l'uguale, le parentesi, ecc.) e da simboli grafici.

I simboli grafici usati sono quelli riportati alla fine di questo articolo.

Accade così che invece di scrivere su disco l'equivalente (in BYTES-ASCII) della parola chiave PRINT, il sistema incida in sua vece uno dei simboli grafici suddetti. In tal modo, invece di dover usare 5 BYTES per scrivere PRINT, ossia un BYTE per ogni lettera, il computer scrive solo un BYTE, quello che identifica il simbolo grafico associato a PRINT. Ovviamente alle altre parole chiave corrispondono altri simboli grafici, in modo da non avere doppioni.

Risulta allora evidente il notevole risparmio di spazio che si ottiene sul disco; in tal modo ogni dischetto è in grado di contenere un numero molto più elevato di dati.

Per talune applicazioni, però, è necessario che il computer possa scrivere FILES anche in FORMATO ASCII, senza cioè fare uso dell'artificio precedente ma scrivendo ad esempio i 5 BYTES che corrispondono, in codice ASCII, alle 5 lettere P-R-I-N-T.

Questo secondo modo di scrivere è chiamato appunto SCRITTURA IN FORMATO ASCII, in contrapposizione alla precedente SCRITTURA IN FORMATO COMPATTATO.

Riassumiamo brevemente quanto detto fin qui. Le registrazioni dei dati su disco avvengono normalmente in FORMATO COMPATTATO. I PROGRAMMI vengono incisi in FORMATO COMPATTATO, a meno che non si metta l'opzione indicata nella voce SAVE alla pagina 108 della precedente rivista; salvando su disco, ad esempio, un programma in questo modo:

**SAVE "BIORITMI" , A**

avremo il risultato di incidere il programma chiamato "BIORITMI" in FORMATO ASCII. In tale modo non solo il programma occupa un maggiore spazio sul disco, ma il tempo richiesto per la registrazione è anche maggiore che non scrivendo in FORMATO COMPATTATO.

Gli ARCHIVI di dati di tipo SEQUENZIALE sono sempre incisi in FORMATO ASCII, mentre quelli di tipo RANDOM vengono sempre incisi in FORMATO COMPATTATO.

La necessità di scrivere in FORMATO ASCII si presenta praticamente solo in un caso: per eseguire il comando MERGE (nel BASIC).

Torniamo ora alla descrizione del comando APPEND.

Come abbiamo detto all'inizio, il comando APPEND serve per attaccare un FILE ad un altro; potremo quindi "appendere" un programma ad un altro, oppure un archivio ad un altro.

La prima delle due possibilità non riveste molto interesse, in quanto esiste a livello BASIC l'istruzione MERGE che fornisce lo stesso risultato, quello di fondere assieme due programmi. In realtà col comando APPEND applicato a due programmi scritti in formato ASCII si realizza sul disco un nuovo programma, sempre scritto in ASCII, con tutte le linee del programma aggiunto che fanno seguito a quelle del programma di partenza.

Se avete notato abbiamo parlato di programmi scritti in FORMATO ASCII: il comando APPEND funziona in effetti anche se essi sono scritti in FORMATO COMPATTATO, in quanto sul disco viene formato un FILE unico, risultato dell'unione dei due di partenza. Però se lanciate questo programma, vedrete che esso è identico a quello cui è stato attaccato il secondo. Ciò si spiega col fatto che con APPEND i due programmi sono stati accodati l'uno all'altro, ma tra il primo programma ed il secondo sono rimasti dei caratteri separatori, e precisamente quelli che identificano la fine del primo

programma. In tal modo il secondo esiste come entità fisicamente attaccata al primo, ma non viene mai posto in esecuzione.

Facciamo un esempio. Scrivete il seguente programma (operando logicamente a livello BASIC):

```
10 REM — PROGRAMMA 1 —
20 CLS
30 PRINT "QUESTO È IL PRIMO PROGRAMMA"
```

Dopo averlo scritto tutto, salvatelo su disco in questo modo:

```
SAVE "PROG1" , A
```

Così facendo il programma appena fatto viene trasferito su disco, in FORMATO ASCII. Successivamente cancellate il programma dalla memoria del computer scrivendo NEW e premendo RETURN. Scrivete poi questo secondo programma:

```
50 REM — PROGRAMMA 2 —
60 PRINT
70 PRINT "QUESTO È IL SECONDO PROGRAMMA"
80 PRINT
```

Salvate poi anche questo programma, scrivendo:

```
SAVE "PROG2" , A
```

Così anche il programma 2 viene registrato in FORMATO ASCII.

Adesso cancellate ancora la memoria con NEW e RETURN e poi scrivete:

```
CMD "APPEND PROG1 TO PROG2"
```

Fate esattamente come sta scritto, con uno spazio tra le parole comprese nelle virgolette: se sarà fatto diversamente il computer segnala errore.

Infatti se non mettete nessuno spazio o ne mettete due, il floppy parte e dopo un po' vedrete la scritta:

```
FILE SPEC REQUIRED
UNPRINTABLE ERROR IN 15359
```

che segnala appunto un difetto di scrittura nel comando impartito.

Quello visto è il modo di usare il comando APPEND del DOS partendo dal BASIC.

In pratica abbiamo detto al computer: appendi il programma PROG1 al programma PROG2.

L'operazione è lecita perché entrambi i programmi richiamati esistono su disco. Dopo aver impartito il comando, vedrete che il floppy parte ed esegue gli ordini ricevuti. Come fare per andare a vedere cosa è avvenuto? È molto semplice; scrivete:

```
LOAD "PROG2"
```

(tralasciamo, come al solito, di ripetere in continuazione di premere anche il tasto RETURN, cosa che ormai dovrebbe essere automatica).

Il floppy si rimette in movimento e nella memoria centrale del computer viene trasferito il programma PROG2. Quando riappare la scritta READY lanciate il programma (scrivendo RUN). Vedrete che lo schermo si spegne, appariranno poi le seguenti scritte:

```
QUESTO È IL PRIMO PROGRAMMA
QUESTO È IL SECONDO PROGRAMMA
READY
```

Come vedete, il programma originale PROG2 è stato modificato in quanto ad esso è stato aggiunto il programma PROG1.

Cerchiamo allora di chiarire bene cosa è successo col comando APPEND.

Scrivendo APPEND PROG1 TO PROG2 il contenuto di PROG1 non cambia, mentre PROG2 viene modificato attaccandogli in coda il contenuto di PROG1.

Provate a fare il listato del programma PROG2 (usando il comando LIST):

```
10 REM — PROGRAMMA 1 —
20 CLS
30 PRINT "QUESTO È IL PRIMO PROGRAMMA"
50 REM — PROGRAMMA 2 —
60 PRINT
70 PRINT "QUESTO È IL SECONDO PROGRAMMA"
80 PRINT
```

Avrete notato che abbiamo sempre parlato di attaccare il programma PROG1 in coda al PROG2; invece nel listato vedete che c'è prima PROG1 e poi PROG2.

Le due cose non sono in contraddizione, come può sembrare. Infatti sul disco nel FILE che ha nome PROG2 si ha un reale accodamento di PROG1 a PROG2, come potremo vedere applicando il comando LIST del DOS. Quando invece carichiamo PROG2 nella memoria, dato che si tratta di un programma BASIC esso viene automaticamente ordinato in ordine crescente di numero di linea. Ciò spiega l'apparente incongruenza anzidetta.

Facciamo ora un'ipotesi diversa: partiamo sempre dai programmi PROG1 e PROG2, così come li abbiamo dati all'inizio (ricordate che dopo l'APPEND fatta il programma PROG2 è cambiato; se volete fare la prova di quanto segue dovrete riscriverlo e fare una nuova SAVE "PROG2",A). Adesso però scrivete:

```
CMD "APPEND PROG2 TO PROG1"
```

Il risultato ottenuto è diverso dal precedente: questa volta è il programma PROG2 ad essere rimasto invariato, mentre PROG1 risulta formato dal PROG1 originario più il PROG2 attaccato in coda. Se ora caricate PROG1 e lo lanciate, otterrete lo stesso risultato dato in precedenza dal programma PROG2 dopo la relativa APPEND.

La regola allora è la seguente: col comando APPEND il FILE che viene accodato rimane invariato, mentre quello su oul si fa l'accodamento viene modificato; il contenuto originario del secondo FILE viene perciò perso. Se non volete che questo accada dovrete prima farvene una copia.

Tutto quello che abbiamo detto fino ad ora sul comando APPEND è basato sul presupposto che I FILES trattati siano dei programmi BASIC scritti in FORMATO ASCII. Provate ora a ripetere gli esempi appena fatti, salvando però i programmi senza mettere la dicitura ',A': essi risulteranno scritti in FORMATO COMPATTATO. Con APPEND questi due programmi verranno uniti su disco.

Se ora fate RUN PROG1 vedrete la scritta:

**QUESTO È IL PRIMO PROGRAMMA**

Se poi fate RUN PROG2 vedrete la scritta:

**QUESTO È IL SECONDO PROGRAMMA**

Accade quindi quello che abbiamo già anticipato: un APPEND dato a due programmi scritti in COMPATTATO li unisce solo fisicamente, ma il programma risultante non si comporta come se fosse la somma dei due di partenza.

Vi domanderete perché ci dilunghiamo tanto sul fatto di eseguire APPEND su programmi, quando in pratica questo non si fa mai in quanto c'è il comando MERGE del BASIC che assolve in modo migliore alla stessa funzione. Non pensate che ci piaccia perdere tempo o farlo perdere a voi: le funzioni effettivamente svolte da APPEND si capiscono, per ora, in modo più immediato applicandole al caso di files-programmi che non a quello di files-dati; questo almeno nel caso che non siate già degli esperti di gestione dei FILES.

Come già detto, il comando APPEND può essere usato anche per unire due FILES di DATI.

Ora la cosa è molto più interessante ed importante, in quanto può risultare molto comodo unire due archivi (SEQUENZIALI o RANDOM). Senza la disponibilità di questo comando, per poter fare la medesima cosa sarebbe necessario scrivere un programma apposito. Si tratta quindi di una comodità operativa che spesso fa risparmiare tempo ed anche probabili errori.

Non andremo oltre nel descrivere esempi di questa applicazione di APPEND, in quanto vi mancano ancora le cognizioni per gestire FILES-ARCHIVIO. Li vedremo in un prossimo articolo; dovreste comunque aver capito il meccanismo svolto da questo comando DOS.

Ancora un'ultima cosa. Nei due programmi prima considerati, abbiamo messo espressamente dei numeri di linea tali da non avere doppioni dopo la loro unione con APPEND. Nel caso che vi fossero state due linee con lo stesso numero, nel listato finale del programma somma aveste visto una sola di quelle due linee, e precisamente quella del programma aggiunto in coda all'altro. Fate qualche prova in proposito, e verificherete quanto asserito.

Naturalmente queste ultime considerazioni valgono solo se i programmi sono stati registrati su disco in FORMATO ASCII.

## ATTRIB

Il comando ATTRIB assegna al FILE specificato gli attributi di protezione. Questi attributi sono quattro: I, ACC, UPD, PROT.

Con l'attributo I si rende invisibile un file inciso su disco allorché si impartisce il comando DIR (vedere la voce relativa più avanti).

Gli attributi ACC, UPD, PROT servirebbero per creare delle protezioni sull'uso dei files mediante l'introduzione di due parole chiave diverse (una di accesso ACC ed una di aggiornamento UPD) e di un livello di protezione PROT. Abbiamo detto 'servirebbero' in quanto nel DOS che vi abbiamo fornito questi tre attributi sono accettati ma non diventano mai operativi.

Siamo stati noi a richiedere espressamente un tale comportamento: volevamo infatti darvi un sistema operativo 'aperto', senza trucchi o protezioni di alcun genere. È proprio questa scelta che ha contribuito non poco ad allungare i tempi di allestimento del DOS e del BASIC. Alla fine però siamo riusciti ad avere un sistema che non presenta impedimenti di sorta: basti dire che il nostro programma di COPY fa una copia fisica del disco, e quindi non esistono ostacoli di alcun tipo che possano impedire un'operazione del genere.

Se gli attributi ACC, UPD, e PROT fossero operativi il loro uso impedirebbe certe operazioni sui files da parte di coloro che non fossero a conoscenza delle due parole chiave ACC e UPD, e ciò era esattamente quello che non volevamo, per evitare speculazioni di qualsiasi tipo. Questo è il motivo per cui i tre attributi anzidetti sono inefficaci. Non ci dilungheremo quindi nella loro descrizione, e parleremo solo del primo attributo I.

Abbiamo già detto altre volte che col comando DIR del DOS si ottiene un elenco dei files presenti sul disco. Però in quell'elenco non compaiono i files che portano l'attributo I, di cui stiamo parlando. La cosa riveste un certo interesse, in quanto ogni DIR ci fornirebbe l'elenco di tutti i files, compresi quelli che sicuramente non ci interessano mai come quelli di sistema (queste cose sono spiegate più in dettaglio alla voce DIR).

Il comando ATTRIB si può impartire in una delle seguenti maniere:

**ATTRIB PROG1 (I)**
**ATTRIB PROG1:1 (I)**
**CMD"ATTRIB PROG1 (I)"**

I primi due modi valgono se si opera a livello DOS ed il terzo se invece ci si trova a livello BASIC. Se sul disco esiste un file di nome PROG1, con il comando dato alla prima riga lo rendiamo invisibile ad una DIR normale.

La differenza tra le righe 1 e 2 consiste nel fatto che con la 1 il computer mette l'attributo I al file di nome PROG1 andandolo a cercare sui vari floppy collegati, mentre con la 2 la ricerca del file avviene solo nel drive 1.

Se il file indicato non esiste sul disco o sui dischi presenti nei floppy, si ottiene il seguente messaggio:

**FILE NOT DIRECTORY** (file assente nell'indice)

Come sempre, dovete fare molta attenzione a scrivere il comando nel formato che vi abbiamo indicato: gli spazi e le parentesi vanno messi correttamente, altrimenti avrete segnalazioni d'errore.

Vediamole una alla volta. Se il contenuto della parentesi è diverso dalla lettera I si ha:

**ATTRIBUTE SPECIFICATION ERROR** (errore nella specificazione d'attributo)

Se non mettete nulla dopo il nome del file o se non mettete lo spazio tra il nome del file e la parentesi, avete:

**— NOTHING DONE —** (non è stato fatto niente)

Se non mettete nulla dopo ATTRIB o se non mettete lo spazio tra ATTRIB ed il nome del file vedete:

**FILE SPEC REQUIRED** (ci vuole il nome del file)

Se sbagliate a scrivere la parola chiave ATTRIB avete:

**PROGRAM NOT FOUND** (programma non trovato)

Quest'ultima segnalazione viene scritta tutte le volte che si impartisce un comando DOS che non sia compreso tra quelli riconosciuti dal computer: se scrivete un comando DOS diverso da uno di quelli contenuti nella tabella 1 avrete sempre quella scritta.

Una volta messo l'attributo I ad un programma, lo stesso non è più visibile eseguendo una DIR, come abbiamo già detto. Allora deve esserci un modo per togliere questo attributo, così da poter tornare alla condizione normale. Infatti, dopo aver messo l'attributo I al programma PROG1, se provate a digitare:

**ATTRIB PROG1 (A = )**

se siete a livello DOS, oppure:

**CMD"ATTRIB PROG1 (A = )"**

se siete a livello BASIC, vedrete che al comando DIR il programma PROG1 è nuovamente visibile. Per provare il tutto leggete prima anche la voce DIR di questa stessa tabella, e poi salvate su disco un programma di prova (ad esempio proprio il primo programma della voce precedente a questa), chiamandolo PROG1.

Va da sé che tale nome è solo un esempio, quindi potete metterne uno a vostro piacimento; ovviamente nel seguito dovrete richiamarlo sempre con quel nome.

Eseguite poi il comando DIR sul floppy dove avete inciso il programma: vedrete che comparirà anche il nome PROG1. Provate poi ad assegnare a quel programma l'attributo I, facendo come abbiamo appena detto. Eseguite poi nuovamente una DIR: il nome di quel programma non viene più scritto. Provate ora a togliere l'attributo I, nel modo spiegato poco fa. Facendo nuovamente una DIR vedrete riapparire il nome del programma che state trattando.

RICORDATE CHE IL PROGRAMMA È SEMPRE INCISO SUL DISCO, SIA CHE APPAIA O CHE NON APPAIA COL COMANDO DIR: L'ATTRIBUTO 'I' SERVE SOLO PER INSERIRLO O MENO NELL'INDICE.

Per accertarsene basta provare a caricarlo in memoria quando non è compreso nell'indice: vedrete che il computer lo va a cercare, lo trova e lo carica; con un LIST od un RUN sarete definitivamente convinti. Sotto la voce DIR troverete il modo di farlo apparire nell'indice anche se ha l'attributo I.

## AUTO

Prima di tutto vogliamo farvi notare che la parola chiave AUTO compare sia nella tabella 1 del DOS che nella tabella 2 del BASIC: si tratta di due cose distinte, che ottengono due risultati diversi.

Certamente ricordate che la parola chiave AUTO del BASIC serve per innescare la numerazione automatica delle linee di programmazione. Invece la funzione del comando AUTO del DOS è quella di dire al computer cosa deve fare dopo che ha caricato il DOS, all'accensione della macchina.

Infatti, dopo aver acceso il calcolatore ed aver premuto il tasto dello spazio, appare la scritta:

**NUOVA ELETTRONICA — NE/DOS**
**DISK OPERATING SYSTEM — VER 1.0**
**BASIC,RUN"MOSTRA**

e via di seguito con tutto il resto.

Ebbene, la scritta BASIC, RUN"MOSTRA è proprio quella che dice al computer cosa fare dopo che ha caricato il DOS: CARICA IL BASIC E POI FAI PARTIRE IL PROGRAMMA DI NOME MOSTRA.

Se al posto della scritta BASIC,RUN"MOSTRA ci fosse solo il BASIC il computer caricherebbe prima il DOS, poi il BASIC, poi si fermerebbe. Se non ci fosse scritto niente, dopo aver caricato il DOS il calcolatore si fermerebbe.

Facciamo qualche esempio, non prima però di aver detto una cosa importante: PER POTER USARE LA FUNZIONE ASSOCIATA AL COMANDO AUTO DEL DOS BISOGNA CHE SUL PRIMO DRIVE SIA MONTATO IL DISCO DEL DOS-BASIC E CHE QUESTO NON ABBIA IL NASTRO DI PROTEZIONE CONTRO LE REGISTRAZIONI.

Come sempre, potete dare il comando AUTO (quello del DOS!) sia partendo dal livello DOS che da quello BASIC. Prendete il disco DOS-BASIC e provate innanzitutto a togliere il comando AUTO che contiene già (BASIC,RUN"MOSTRA).

Per ottenere quel risultato dovete scrivere:

**AUTO**

se partite dal livello DOS; invece, dovete digitare

**CMD"AUTO"**

se partite dal livello BASIC. D'ora in poi non faremo più entrambi i casi di partire sia dal DOS che dal BASIC: ormai la differenza la conoscete, quindi supporremo sempre di essere a livello DOS. Parleremo del livello BASIC solo nel caso che ci siano delle differenze rispetto al livello DOS.

Ovviamente il disco non deve avere il nastro di protezione contro le registrazioni, come abbiamo gia detto. Se avete fatto tutto a puntino, il comando viene accettato. Per vedere l'effetto che ha avuto, spegnete il computer e poi riaccendetelo e ripartite nel modo solito (RESET e barra dello spazio).

Dopo pochi secondi il floppy si ferma ed appare la scritta DOS READY: siete cioè al livello DOS. Quindi il COMANDO AUTO SCRITTO DA SOLO SERVE PER TOGLIERE LE PRECEDENTI ISTRUZIONI DI 'AUTO' PRESENTI SUL DISCO.

Nel caso che sbagliate qualcosa, appaiono delle segnalazioni d'errore; vediamole, separando il livello DOS dal livello BASIC in quanto si hanno segnalazioni diverse.

Al punto in cui siamo potete commettere tre errori: o sbagliate a scrivere la parola AUTO, o il disco del primo drive non è DOS-BASIC, oppure esso ha il nastro di protezione contro le registrazioni.

Per prima cosa supponiamo di partire dal livello DOS.
I tre casi ora esposti danno rispettivamente le seguenti segnalazioni d'errore:

**PROGRAM NOT FOUND** (programma non trovato)
**DOS READY** (DOS pronto)
**GAT WRITE ERROR** (errore di scrittura in una pista)

Nel primo caso il computer non può riconoscere la parola chiave scritta in modo errato, e va a cercare un file con quel nome. La seconda scritta non è, in realtà, una segnalazione d'errore: il calcolatore torna al DOS senza aver fatto nulla. Nel terzo caso vedrete che prima di scrivere GAT WRITE ERROR il computer cerca ripetutamente di scrivere sul disco, ma non ci riesce a causa del nastro di sicura; allora vi manda quel messaggio d'errore.
Se invece supponiamo di partire dal livello BASIC, nei tre casi menzionati avremo le seguenti segnalazioni d'errore: nel primo caso

**INTERNAL ERROR IN 15359** (errore interno nella linea 15359)

Nel secondo caso, quello di AUTO applicato ad un disco che non è DOS-BASIC, il floppy parte, poi si ferma ed il computer non torna da solo al livello BASIC: occorre fare un BREAK.
Nel terzo caso si ottiene la scritta:

**DISK I/O ERROR. USE E-CMD FOR SPECIFIC IN 15359**
(errore di INPUT-OUTPUT. Usa il comando CMD"E" per conoscerlo)

Se provate, a questo punto, a scrivere appunto CMD"E" vedrete apparire la scritta GAT WRITE ERROR già citata.
Ora abbiamo visto l'uso di AUTO da solo: cancella le precedenti istruzioni AUTO. Vediamo adesso come si impartiscono nuove istruzioni. Provate a scrivere dal livello DOS:

**AUTO BASIC,RUN"MOSTRA"**

Dopo che il floppy si sarà fermato, provate a fare un RESET e a ripartire da capo: vedrete che viene caricato il BASIC e poi parte il programma MOSTRA.
Al solito, le virgolette dopo MOSTRA sono facoltative.
Il caso di AUTO è l'unico che pone delle differenze reali a seconda che si operi dal livello DOS o da quello BASIC. Infatti se volete dare lo stesso comando appena visto, vi rendete conto che la cosa non è possibile, a causa delle virgolette che vanno dopo RUN. Se ci pensate bene, converrete che la scritta CMD"AUTO BASIC, RUN"MOSTRA" risulta chiaramente errata proprio a causa delle virgolette dopo RUN. Se provate a farlo otterrete la scritta:

**SYNTAX ERROR IN 15359** (errore di sintassi)
**UNDEFINED LINE ≠ IN 15359** (numero di linea indefinito)

Attenzione, però: il comando AUTO impartito è stato ugualmente eseguito; se fate RESET e ripartite da capo, vedrete che l'automatismo ora contiene BASIC,RUN. Ciò è abbastanza logico, in quanto le virgolette dopo RUN nel comando AUTO impartito hanno chiuso il messaggio di automatismo da scrivere; il computer segnala errore perché alle virgolette seguono altri caratteri, che invece non dovrebbero esserci. Se volete convincervene, provate a scrivere

**CMD"AUTO BASIC,RUN"**

Questa istruzione viene eseguita senza segnalazioni d'errore e dà lo stesso risultato (senza senso) dell'esempio precedente.

Il comando AUTO permette ancora un'ultima operazione, quella di stabilire il numero massimo di FILES di dati che possono essere trattati contemporaneamente.
Il formato completo del comando AUTO è infatti il seguente:

**AUTO BASIC,10,RUN"MOSTRA"**

Esso vale a livello DOS; si è supposto di voler caricare in automatico il BASIC, di riservare 10 files dati, e di mandare in esecuzione il programma MOSTRA. Tutte queste scelte sono solo d'esempio, come sempre.
Il numero massimo di files può essere 15; se non si scrive niente il sistema riserva automaticamente 3 files, che normalmente sono sufficienti.
Riservando un maggiore numero per i files, la memoria a disposizione dell'utente diminuisce. Vedere, per maggiori ragguagli a questo proposito, la gestione dei files esterni di tipo sequenziale e random.
Vediamo di tirare un po' le somme sull'uso del comando AUTO del DOS.
Scrivendo solo AUTO si tolgono le istruzioni AUTO precedenti. La parola chiave AUTO può essere seguita da uno qualsiasi dei comandi DOS; una volta caricato il DOS, verrà eseguito quel comando. Se dopo tale comando si specifica un numero, esso è il massimo dei files che possono essere trattati assieme. Se il numero non viene espressamente indicato, esso è assunto uguale a tre dal sistema, in modo automatico. Se il comando DOS messo in auto è la parola chiave BASIC, il computer carica prima il DOS e poi il BASIC.
Il tal caso dopo la parola BASIC si può anche aggiungere una virgola e un comando del BASIC: il calcolatore carica prima il DOS, poi il BASIC, quindi esegue il comando BASIC indicato.
L'unica eccezione a quest'ultima considerazione è data dal caso che si debba mettere un comando BASIC che richieda le virgolette; in tal caso il solo modo per poterlo fare è quello di portarsi a livello DOS.
Vi facciamo qualche esempio; provate ad eseguirli per capire bene il funzionamento di AUTO. Al solito supponiamo di essere a livello DOS.

(1) **AUTO DIR**
(2) **AUTO FREE**
(3) **AUTO AUTO**
(4) **AUTO BASIC,CLS**
(5) **AUTO BASIC,?MEM**
(6) **AUTO BASIC,CLEAR1000**
(7) **AUTO BASIC,REVON**
(8) **AUTO BASIC,7,CLS:?MEM**

L'esempio numero 1 causa una DIR, ossia dà l'indice del primo disco (per questa come per altre voci del DOS vedere le relative spiegazioni).

Il caso 2 visualizza la disponibilità di posto libero sui vari drives.

Il 3 è un po' strano ed essendo molto istruttivo vi consigliamo di seguirlo con attenzione. In pratica con esso diciamo al computer di eseguire il comando AUTO; dopo averlo impartito, fate un RESET e premete la barra dello spazio. Si ha prima il caricamento del DOS (e qui vedrete, alla terza riga, la scritta AUTO che indica appunto che il computer manderà automaticamente in esecuzione il comando AUTO, dopo aver caricato il DOS); successivamente appare la scritta DOS READY. Fate ora nuovamente un RESET e ripartite da capo: vedrete che alla terza riga non c'è scritto più nulla. Infatti il computer carica il DOS e poi si ferma senza fare più nulla: al giro precedente il comando AUTO messo in automatico aveva cancellato il precedente AUTO!

L'esempio 4 causa prima il caricamento del BASIC, poi la cancellazione del monitor.

Col 5 si ottiene il caricamento del BASIC e poi la stampa del numero di celle di memoria disponibili: è il risultato del comando PRINT MEM del BASIC.

Con l'esempio 6, dopo il caricamento del BASIC, si riservano 1000 caratteri alle stringhe, al posto dei 50 soliti.

Col 7 si ottiene l'inserimento automatico della scrittura in REVERSE.

L'esempio 8 è l'insieme del 4 e del 5; dopo aver caricato il BASIC, il calcolatore cancella lo schermo e stampa la disponibilità di memoria.

Inoltre riserva 7 files per lo scambio contemporaneo di dati con dischi.

Come vedete, specie da quest'ultimo caso, le possibilità offerte dal comando AUTO del DOS sono molteplici e molto interessanti.

Se mettete uno o più spazi prima e dopo la virgola ottenete una segnalazione d'errore. Potete scrivere in maiuscolo o in minuscolo indifferentemente: il computer tramuta tutto automaticamente in maiuscolo.

## BASIC

Come risulta da quello che abbiamo detto alla voce precedente, il comando BASIC serve per passare dal livello DOS a quello BASIC.

Eccovi una considerazione particolare: se scrivete, dal livello BASIC, CMD"BASIC" passate al livello DOS e da quello caricate nuovamente il BASIC; attenzione, però così facendo si perde un eventuale programma precedentemente presente in memoria.

## COPY

Abbiamo già parlato a lungo del comando COPY, nel precedente numero di NUOVA ELETTRONICA. Però ci eravamo limitati ad ottenere copie di dischi interi.

Il comando COPY serve invece anche per trasferire un FILE da un disco ad un altro. Le modalità da seguire sono le stesse che già conoscete, con una sola differenza: non si deve mettere la data.

Il comando da impartire per copiare, ad esempio, il programma MOSTRA presente nel drive zero sul disco posto nel drive 1 è il seguente:

**COPY MOSTRA:0 TO :1**

Per ogni altra considerazione su possibilità d'errori vedere la rivista precedente. Esiste una sola differenza; provate a scrivere:

**COPY MOSTRA:0 TO :0**

Ricorderete che nel caso di copia di un intero disco questo si poteva fare.

Invece nel caso di copia di un solo file ciò non è ammesso; otterrete infatti la scritta:

**SOURCE & DEST SAME FILE** (il file è già nel disco destinatario)
**UNPRINTABLE ERROR IN 15359** (errore non stampabile)

La seconda riga è stampata solo se si opera a livello BASIC.

Ciò non significa però che sia impossibile copiare un file avendo un solo drive. Infatti il comando precedente è rifiutato perché sul drive zero esiste già un file col nome MOSTRA (tutto quello che stiamo dicendo vale ovviamente solo se usate per questa prova un disco DOS-BASIC che contenga il programma dimostrativo MOSTRA e che non abbia il nastro adesivo di protezione).

Provate a digitare quanto segue:

**COPY MOSTRA:0 TO MOSTRA1:0**

In questo modo diciamo al computer di prendere il file MOSTRA che è sul drive zero e di trasferirlo in uno nuovo, di nome MOSTRA1, sempre sul drive zero.

Provate, a computer fermo, a fare un DIR: troverete anche il file MOSTRA1.

Ovviamente il nome MOSTRA1 è solo un esempio; qualsiasi altro nome sarebbe andato altrettanto bene, purché diverso da MOSTRA.

## DATE

Il comando DATE serve per assegnare la data; essa viene depositata nella prima parte della variabile speciale TIMES, di cui abbiamo già parlato nella descrizione della tabella 6.

Supponiamo di voler assegnare la seguente data: 22 Aprile 1982; allora dovremo scrivere:

**DATE 04/22/82**

Abbiamo usato la notazione anglosassone; comunque, anche se non lo fate, il computer in questo caso non segnala errore.

Successivamente, a livello BASIC, chiedendo la stampa della variabile TIME otteniamo:

**04/22/82 00:00:00**

Il comando TIME che vedremo tra poco serve per assegnare anche l'ora, che per il momento è rimasta a 00:00:00.

Se sbagliate qualcosa nell'assegnazione del comando avrete diverse segnalazioni d'errore, come BAD FORMAT — PROGRAM NOT FOUND, seguite da INTERNAL ERROR IN 15359 oppure UNPRINTABLE ERROR IN 15359 se siete partiti dal livello BASIC.

## DEBUG

Con questo comando DOS, utilizzabile anche al livello BASIC, è possibile scrivere (solo a linguaggio macchina), correggere, modificare, provare PASSO-PASSO, esaminare registri della CPU e qualsiasi area di memoria nonché Files su disco.

Con il comando DEBUG è possibile anche far girare qualsiasi programma e creare nuovi FILES di UTILITÀ da salvare su disco.

Per illustrare il funzionamento del DEBUG, facciamo riferimento al solo livello DOS; è ovvio che tutto quello che vale a questo livello vale nel medesimo modo anche a livello BASIC utilizzando il comando CMD «funzione DOS», già illustrato sulla rivista n. 79.

Per attivare il DEBUG, scriveremo semplicemente:

**DEBUG**

oppure **DEBUG (ON)**

entrambe le scritte sono valide ed accettate; dopo la pressione del tasto RETURN, comparirà al solito:

**DOS READY**

**NOTA —** d'ora in poi, per semplicità di esposizione, quando non è espressamente richiesta altra procedura, ometteremo l'operazione «Premere RETURN» e la scritta DOS READY.

Dopo l'attivazione della funzione DEBUG, il computer assume un comportamento diverso da quello cui siete abituati e cioè:

1) caricando un programma utilità del DOS, come ad esempio il COPY, il FORMAT, il BASIC, questi non diventa operativo ma viene solo caricato in memoria, il computer entra in debug-mode visualizzando sul video lo STATO-MACCHINA della CPU e, a questo punto, è possibile utilizzare i comandi di DEBUG per compiere vari tipi di operazioni su quel programma.

2) Pigiando semplicemente RETURN, sul video appare la scritta:

**WHAT?**

e subito dopo si entra in debug-mode esattamente come nel caso precedente.

Dopo l'entrata in DEBUG-MODE, il display video assume due formati principali:

1) rappresentazione di una pagina video comprendente tutti i registri della CPU (compresi quelli alternativi) più una parte di memoria di 32 bytes visibile in basso nella figura 1.

2) rappresentazione di una pagina video comprendente solo memoria per un totale di 128 bytes. (vedi figura 2)

Per passare da una rappresentazione all'altra vedremo più avanti quali comandi usare.

Guardando la Fig. 1, cerchiamo di interpretare correttamente quello rappresentato:

— guardando la prima colonna a sinistra potrete vedere rappresentati tutti i registri dello Z-80 compresi quelli alternativi contraddistinti dall'apostrofo. (AF BC DE sono principali, AF' BC' DE' sono alternativi).

— procedendo verso destra, dopo il segno di uguale, segue il cotenuto (in ESADECIMALE) della coppia di registri visualizzata. (ad esempio HL = 513C significa che il registro H contiene il numero esadecimale 51 e il registro L contiene il numero esadecimale 3C).

— andando ancora verso destra (esclusi i soli registri AF e AF') seguono 7 bytes esadecimali contenuti nelle locazioni di memoria a partire da quella individuata dalla coppia dei rispettivi registri usati come PUNTATORE nell'indirizzamento INDIRETTO.

Per essere più chiari la riga scritta come:

**HL 513C > E9 D8 C6 06 38 03 C6**

significa che HL contengono rispettivamente 51 e 3C e che questi due registri insieme formano un puntatore in memoria alla locazione 513C dove da quell'indirizzo in poi sono contenuti i bytes E9 D8 C6 06 38 03 C6 cioè:

| | | |
|---|---|---|
| **513C** | contiene | E9 |
| **513D** | contiene | D8 |
| **513E** | contiene | C6 |
| **513F** | contiene | 06 |
| **5140** | contiene | 38 |
| **5141** | contiene | 03 |
| **5142** | contiene | C6 |

Per i registri AF e AF' viene invece indicato il contenuto dei registri unitamente alla rappresentazione estesa del registro F (FLAG). Ad esempio per AF = 0D4A-Z--1-N- si intende che il registro A (ACCUMULATORE) contiene il numero esadecimale OD, e che nel registro F (FLAG) sono SETTATI (stato logico = 1) i bit di ZERO (Z) e il bit INDICATORE di SOTTRAZIONE (N). Il quarto bit da destra rappresentato a 1, è un flag usato internamente dalla CPU e non è accessibile dal programmatore.

Nel registro F possono comparire i seguenti segni che sono rispettivamente: (da sinistra a destra)

| | |
|---|---|
| **S** | (flag di SEGNO) |
| **Z** | (flag di ZERO) |
| **1** | (flag non usato) |
| **H** | (flag di RIPORTO PARZIALE-HALF CARRY) |
| **1** | (flag non usato) |
| **P** | (flag di PARITÀ o OVERFLOW) |
| **N** | (flag di SOTTRAZIONE) |
| **C** | (flag di CARRY) |

Tutti i simboli riportati nel registro F, compariranno solo se i rispettivi Bit sono SETTATI altrimenti viene stampato solo un TRATTINO(—).

Anche se non l'abbiamo detto, quanto spiegato precedentemente a proposito dei registri HL, BC, DE, AF, vale anche per i registri IX, IY, SP, PC, con la sola differenza che questi sono registri a 16 bit e che quindi la scritta PC = 4033 non significa che P contiene 40 e C contiene 33, bensì che il registro PC (PROGRAM COUNTER) contiene 4033 esadecimale. Gli altri 7 bytes che seguono indicano quanto precedentemente spiegato.

## COMANDI DEBUG:

**A** Battendo semplicemente la lettera A, si ottiene la rappresentazione ASCII di caratteri visualizzati. Tutti i caratteri ASCII inferiori a 20H (Esadecimale) vengono rappresentati con un punto (.). Vedi figure 3 e 4.

**C** Permette l'esecuzione di un programma in Single-Step (Passo-Passo). Il programma parte dalla locazione contenuta nel registro PC (Program Counter), l'esecuzione prosegue di un'istruzione alla volta ad ogni battuta della lettera C.
Dopo l'esecuzione di un'istruzione, il display video viene aggiornato con il nuovo contenuto dei registri e della memoria (se modificati), questo permette di osservare cosa succede esattamente in un programma.
IMPORTANTE: Se nel programma si incontra una chiamata di SOUBROUTINE, questa VIENE ESEGUITA INTERAMENTE. Questo è importante perché permette di seguire solo il flusso del programma PRINCIPALE.

**I** Funziona esattamente come il comando C con l'unica differenza che il passo-passo lavora anche sulle SOUBROUTINE.

**Daaaa** Scrivendo D seguito da un indirizzo aaaa si seleziona l'indirizzo di start per ottenere un display della memoria. Il numero totale di bytes rappresentati dipende dal formato del video su cui si sta lavorando (vedi figg. 1 e 2).
Importante è notare che se per l'indirizzo aaaa battete più di quattro numeri esadecimali, il computer considera solo gli ultimi quattro. Per rendere operativo il comando D, bisogna battere lo SPAZIO.
Esempio: D1234 D561234 selezionano entrambe l'indirizzo 1234 esadecimali.
D0 D66 selezionano rispettivamente gli indirizzi 0000 e 0066 esadecimali.

**Gaaaa** Seguito da RETURN pone l'indirizzo aaaa nel Program-Counter e parte ad eseguire il programma da quella locazione.
Es. G700 esegue un programma alla locazione 700.
G402D esegue un programma alla 402D. NOTA: USATE QUESTA SCRITTA PER USCIRE DAL DEBUG E TORNARE AL DOS.

**Gaaaa,bbbb** Seguito da RETURN esegue un programma dalla locazione aaaa alla locazione bbbb.
Es. G750A,75 CF esegue dalla 750A alla 75CF. L'arresto a 75CF avviene per inserimento di un BREAK-POINT automatico a questa locazione. All'arresto è possibile esaminare lo stato macchina sui registri della CPU.
Sia per Gaaaa che per Gaaaa,bbbb se battete più di quattro numeri, la macchina vi accetta solo gli ultimi QUATTRO.
Questo è utile in caso di errore, poiché è sufficiente ribattere l'indirizzo corretto.

**H** Commuta nella rappresentazione esadecimale la pagina video rappresentata. È esattamente l'inverso del comando A.

**Maaaa** Seguito dallo SPAZIO serve per modificare la memoria della locazione aaaa in poi. Es. se battete M7000 seguito dalla barra di spazio, nell'estremità inferiore a sinistra del video viene scritto in NEGATIVO l'indirizzo 7000 seguito in basso dal contenuto attuale di quella locazione. Se volete modificarla battete il nuovo valore seguito ancora dallo SPAZIO.
Quella locazione viene aggiornata al nuovo valore, quindi si visualizzerà l'indirizzo successivo con il rispettivo valore ecc. Se state lavorando su una pagina video rappresentante l'area di memoria che volete modificare, ai lati del byte indirizzato vengono stampate due barrette verticali; dopo la modifica potete già osservare il nuovo valore e le due barrette si sposteranno al byte successivo.
Se volete che il byte indirizzato non venga modificato battete solo lo SPAZIO.
Per l'inserimento dell'indirizzo aaaa e dei bytes da sostituire vale la casistica illustrata per i comandi D e G, riportiamo qui qualche esempio:
M0 M32 MCF0 M7007345 selezionano rispettivamente: 0000, 0032, 0CF0, 7000, 7345.
Per i bytes: SPAZIO, 0, 3E, 3E22, F, il risultato sarà: NESSUNA MODIFICA 00,3E,22,0F.
Ovviamente i valori riportati sono tutti in ESADECIMALE.

IMPORTANTE: SE USATE IL COMANDO M, NON LAVORATE MAI SOTTO L'INDIRIZZO 7000, ALTRIMENTI RISCHIATE DI «SPORCARE» IL DOS.

Per USCIRE dal comando M, battete la lettera X oppure RETURN. Se volete poi rientrare esattamente nel punto in cui siete usciti, è sufficiente battere solo la lettera M.

**Rrr dddd** Il comando R serve per modificare una coppia di registri da 8 bit o un registro da 16 bit. L'operazione diventa esecutiva battendo lo SPAZIO. Es. RHL (spazio) 78C2 (spazio) pone nel registro H il valore 78 e in L il valore C2.

Se durante l'introduzione dei dati vi accorgeste di aver commesso un errore, potete annullare quanto battuto con la lettera X oppure con il tasto RETURN.

**S** Commuta il display video per una rappresentazione di 128 locazioni di Memoria. (VEDI FIGURE 2 e 4).

**U** È una funzione che non ha una grossa utilità, serve ad aggiornare in modo dinamico il display video.

Se battete U osserverete un video una serie di «disturbi» che indicano appunto la continua scrittura di dati.

Per arrestare basta premere un tasto qualsiasi.

**X** Commuta il display video nel formato comprendente Regisri e Memoria (figg. 1 e 3). È usato anche per annullare l'immissione di dati negli altri comandi.

**;** Incrementa la pagina di memoria rappresenta sul display video.

Nel formato comprendente anche i registri la pagina è incrementata di soli 32 byte. (Vedi figg. 1 e 3 ultime 4 righe), nel formato invece di sola memoria, l'incremento è di 128 byte.

**—** Uguale al comando precedente ma, al contrario di questi, DECREMENTA la pagina di memoria visualizzata.

Per disabilitare il DEBUG, scrive:

**DEBUG (OFF)**

seguito da RETURN. Per tutti i programmi che vorrete scrivere con la funzione DEBUG, ricordatevi che devono partire almeno dalla locazione 7000 esadecimale, possono essere registrati come file su disco con la funzione DUMP e riletti con il LOAD (vedi rispettivi paragrafi).

Un esempio reale di applicazione del DEBUG potrebbe essere la scrittura di routine utente (USR) da BASIC. Per entrare in DEBUG da BASIC utilizzare il comando CMD"D.

```
AF =0D4A-Z--1-N-
BC =090D> 41 4E 23 AE 47 2E 00
DE =401D> 07 58 04 53 EC 00 44
HL =51C3> E9 D8 C6 06 38 03 C6
AF'=0044-Z---P--
BC'=4D69> C3 A1 60 C3 7C 57 C3
DE'=0108> 44 59 20 00 00 0D 00
HL'=4D00> C3 46 5E C3 8E 55 C3
IX =FFF7> 08 01 00 00 00 01 00
IY =FFFF> 08 F3 AF C3 74 06 C3
SP =41FA> 2D 40 FF 00 FF 44 00
PC =4033> C3 BB 44 68 00 00 00
7020>  45 4C 45 43 54 3A 24 0D
7028>  44 45 53 54 49 4E 41 54
7030>  49 4F 4E 20 44 52 49 56
7038>  45 3F 24 0D 49 4E 53 45
```

```
7020> 45 4C 45 43 54 3A 24 0D
7028> 44 45 53 54 49 4E 41 54
7030> 49 4F 4E 20 44 52 49 56
7038> 45 3F 24 0D 49 4E 53 45
7040> 52 54 20 43 4F 50 59 20
7048> 41 54 20 44 52 49 56 45
7050> 20 31 0D 54 48 45 4E 20
7058> 54 59 50 45 20 52 45 54
7060> 55 52 4E 24 0D 46 55 4E
7068> 43 54 49 4F 4E 20 43 4F
7070> 4D 50 4C 45 54 45 24 4C
7078> 55 43 41 20 31 39 38 32
7080> 4E 79 FE 24 C8 CD 06 F0
7088> 23 18 F5 CD 03 F0 F5 4F
7090> CD 06 F0 CD 68 F3 F1 C9
7098> 21 00 EC 22 48 00 21 80
```

```
AF =0D4A-Z--1-N-
BC =090D> A N # * G . .
DE =401D> . X . T . . D
HL =51C3> . . . . 8 . .
AF'=0044-Z---P--
BC'=4D69> . * § . o W .
DE'=0108> D Y . . . . .
HL'=4D00> . F ^ . * U .
IX =FFF7> . . . . . . .
IY =FFFF> . . * . T . .
SP =41FA> - § . . . D .
PC =4033> . * D A . . .
7020>  E L E C T : $ .
7028>  D E S T I N A T
7030>  I O N   D R I V
7038>  E ? $ . I N S E
```

```
7020> E L E C T : $ .
7028> D E S T I N A T
7030> I O N   D R I V
7038> E ? $ . I N S E
7040> R T   C O P Y
7048> A T   D R I V E
7050>   1 . T H E N
7058> T Y P E   R E T
7060> U R N $ . F U N
7068> C T I O N   C O
7070> M P L E T E $ L
7078> U C A   1 9 8 2
7080> N Y . $ . . . .
7088> # . . . . . . O
7090> . . . . H . . .
7098> ! . . " H . ! *
```

## DIR

Questo comando DOS è di grande utilità e di uso molto frequente in quanto ci fornisce l'elenco dei FILES contenuti in un dischetto.

Il formato di questo comando è il seguente (dal livello DOS):

**DIR :N**

dove al posto di N bisogna mettere il numero del drive che interessa. Va da sé che tale numero deve essere compreso tra zero e 3. Se si scrive solo DIR senza aggiungere i due punti ed il numero, si ottiene l'indice del disco montato nel drive zero.

Prendete, ad esempio, il disco DOS-BASIC o una sua copia, e mettetelo nel primo drive. Dal livello BASIC scrivete:

**CMD"DIR"**

(Potete anche omettere le virgolette finali). Il floppy parte e poco dopo vedrete che il monitor si spegne ed appare la scritta:

**FILE DIRECTORY ——— DRIVE 0**
**NE-DOS —— 01/01/82**
**MOSTRA**
**DOS READY**

La prima riga ci informa che si tratta dell'indice dei files contenuti dal disco che si trova inserito nel drive zero. La seconda riga fornisce prima il nome che è stato assegnato al disco (NE-DOS nel nostro caso), poi la data di formattazione o di copia. Di queste ultime cose abbiamo già parlato esaurientemente nel numero precedente della rivista, a proposito della formattazione dei dischetti.

Prima o dopo MOSTRA compaiono altri nomi, se nel disco avete inciso altri FILES.

Se avete altri dischetti contenenti programmi o archivi, provate a fare una DIR su di essi; ricordate però che sul primo drive deve sempre esserci un disco DOS-BASIC. Questo discorso lo conoscete già, e quindi è inutile ripeterlo.

Quello indicato non è l'unico modo di utilizzare il comando DIR. Infatti dovete sapere che dandolo nel modo detto prima si ottiene l'elenco dei programmi che non hanno ricevuto l'attributo I (vedere al riguardo la voce ATTRIB già descritta). Gli eventuali programmi che invece hanno l'attributo in questione, pur essendo presenti sul disco, non compaiono nell'elenco.

Il comando DIR può, essere però, dato specificando anche tre opzioni, ognuna delle quali è facoltativa. Ecco il formato completo di DIR:

**DIR :1(A,I,S)**

Come vedete, esso vale se si è al livello DOS (dal BASIC occorre aggiungere CMD") ed abbiamo supposto di volere il contenuto di un disco che si trova sul secondo drive, quello che porta appunto il numero 1. Lo stesso comando dato per il drive zero sarebbe:

**DIR (A,I,S)**

Al solito, gli spazi e tutto il resto vanno rigorosamente rispettati, altrimenti si hanno segnalazioni d'errore, come vedremo più avanti.

Vediamo ora qual'è la funzione svolta dalle varie opzioni A,I,S esaminandole separatamente.

Fate la seguente prova, inserendo nel primo drive il disco DOS-BASIC e lavorando a livello DOS. Eseguite una DIR diversa da quella che avete fatto in precedenza:

**DIR (I)**

(mettere lo spazio dopo DIR). Vedrete le scritte:

**FILE DIRECTORY ——— DRIVE O**
**NE-DOS —— 01/01/82**
**FORMAT/CMD IP**
**MOSTRA**
**BASIC/CMD IP**
**COPY/CMD IP**
**DOS READY**

Come vedete, compaiono altri due programmi oltre a MOSTRA; BASIC/CMD e COPY/CMD che servono per caricare il BASIC o per fare copie di dischi e files.

La sigla IP significa che si tratta di files con l'attributo I e protetti dalla parola di PASSWORD (questa non è operativa, come abbiamo già detto).

Provate ora a dare il seguente comando:

**DIR (S)**

Compariranno le seguenti scritte:

**FILE DIRECTORY ——— DRIVE 0**
**NE-DOS —— 01/01/82**
**BOOT/SYS SIP**
**DIR/SYS SIP**
**SYS0/SYS SIP**
**SYS1/SYS SIP**
**SYS2/SYS SIP**
**SYS3/SYS SIP**
**SYS4/SYS SIP**
**SYS5/SYS SIP**
**SYS6/SYS SIP**
**SYS7/SYS SIP**

A questo punto non appare la scritta DOS READY: infatti l'elenco non è ancora terminato. La pressione di un tasto qualsiasi provoca la stampa dei titoli successivi, facendo scorrere i precedenti verso l'alto:

**SYS11/SYS SIP**
**SYS12/SYS SIP**
**SYS13/SYS SIP**
**MOSTRA**
**DOS READY**

Ora l'elenco è finito, finalmente! Come vedete, oltre al file MOSTRA e a quelli che avete eventualmente aggiunto voi, ce ne sono parecchi altri; essi portano il DOS, il BASIC, e tutto quello che serve per il corretto funzionamento del computer.

La lettera S della parola SIP indica che si tratta di un file di sistema, cioè di un file che serve al calcolatore per poter svolgere correttamente le varie operazioni su disco.

Se avete due drive, provate a mettere sul secondo un disco vergine e formattatelo; scrivete poi:

**DIR :1(S)**

Di tutte le scritte precedenti appariranno solo le prime due (BOOT/SYS SIP e DIR/SYS SIP): quei due files sono presenti in ogni disco e servono per poter scrivere e leggere su di esso correttamente, nonché per avere la DIR di quel disco.

Non è ancora finita! Provate infatti ad impartire il seguente comando:

**DIR (A)**

Dopo le prime due righe solite, leggerete:

**MOSTRA**
**LRL = 256 / EOF = 1**
**SIZE = 1 GRAN**

Spiegheremo in dettaglio il significato della scritta alla voce FREE; per ora vi basta sapere che essa fornisce vari dati sulla struttura del file MOSTRA.

Se sul disco sono presenti altri files oltre a MOSTRA, per ognuno di essi il computer fornisce una scritta analoga. Se il numero dei files è elevato, il computer ve li presenta quattro per volta; la pressione di un tasto qualunque permette di visionare quelli successivi. Quando appare la dicitura DOS READY significa che non ci sono più files da elencare.

Per provare tutto quello che abbiamo detto in una sola volta, fate così:

**DIR (A,I,S)**

otterrete un lungo elenco di files, coi rispettivi dati d'ingombro. In tale elenco sono compresi, oltre a MOSTRA, sia i files di sistema che quelli che portano l'attributo I.

Ricapitoliamo: il comando DIR assegnato senza le opzioni fornisce il contenuto del disco inserito nel drive specificato; i files con l'attributo I non sono visualizzati. Se si mette l'opzione A si hanno anche i dati di struttura dei vari files. Con l'opzione I si visualizzano anche i nomi dei files che hanno l'attributo I. Con l'opzione S si ottengono anche i files di sistema.

Le opzioni possono essere date una, due o tre per volta, in un ordine qualunque.

Vediamo gli errori che si possono fare. Se scrivete DIR:1 oppure DIR :1, vale a dire se non mettete lo spazio dopo DIR o ne mettete più di uno, il computer non segnala errore, ma fornisce sempre l'indice del primo drive

Se mettete uno o più spazi prima della parentesi che assegna le opzioni tutto procede come desiderato, senza errori. Se commettete altri tipi di errori avrete segnalazioni diverse; ad esempio potreste leggere

**ILLEGAL LOGICAL FILE NUMBER** (numero logico di file non lecito)
**BAD FILE NAME** (nome di file dato male)
**DATA RECORD NOT FOUND DURING READ** (dati di registrazioni non trovati in lettura)
**PARITY ERROR DURING READ** (errore di parità in lettura),

ed altri ancora. La casistica è molto vasta e risulta praticamente impossibile passarla tutta in rassegna; oltretutto non avremmo abbastanza fantasia per cercare di scrivere in tutti i modi errati possibili!

Gli ultimi due esempi di segnalazione d'errore avvengono se la testina di lettura del drive si è sporcata, oppure se il disco si è rovinato; anche se raramente, potrebbe verificarsi pure in caso di incompatibilità tra il drive su cui è stato inciso il disco e quello su cui si fa la DIR. Ad ogni buon conto usate dischetti di buona qualità (che non perdano l'ossido andando così a sporcare le testine) e trattateli con le dovute cautele. Non toccate mai la superficie magnetica del disco con le mani, e conservateli nella loro busta.

Prima di passare alla voce successiva, vogliamo chiarire anche un altro dubbio che forse vi è sorto leggendo gli esempi fatti. Ci riferiamo ai nomi dei files che contengono anche il simbolo della divisione (/) ed altre tre lettere.

Nel numero scorso di NUOVA ELETTRONICA abbiamo parlato abbastanza diffusamente del comando SAVE. Adesso è arrivato il momento di completare quelle nozioni, in quanto ora siete in grado di capire anche il resto.

Innanzitutto bisogna sapere che il nome che si assegna ad un programma può essere di qualsiasi forma, a patto che inizi con una lettera.

Se cercate di salvare un programma scrivendo ad esempio SAVE"2AD" il calcolatore vi manda il messaggio BAD FILE NAME e non effettua la registrazione.

Dopo la prima lettera potete però mettere indifferentemente lettere o numeri; se un carattere non è né una lettera né un numero, il programma viene registrato con un nome uguale a quello che precede il carattere non ammesso.

Ad esempio facendo SAVE"DARIO 2" si registra il programma di nome DARIO; lo stesso risultato darebbero SAVE" DARIO, A" e SAVE"DARIO(12)".

La lunghezza massima di un nome è di otto caratteri; se ne date di più la parte eccedente viene ignorata. Quindi SAVE"FATTURAZIONE" viene accettato, ma registrato col nome FATTURAZ di otto lettere.

L'unico carattere ammesso oltre le lettere ed i numeri è la barra della divisione (/), e questo per un motivo ben preciso.

Dopo che avrete usato anche i files di dati sequenziali e random, vi renderete conto che con una DIR si ottiene l'elenco di tutti i files presenti sul disco, ma i programmi non saranno distinguibili dagli archivi. Inoltre i programmi salvati in

formato ASCII (vedere al riguardo il comando APPEND) non sono diversi, nell'indice, da quelli salvati in formato COMPATTATO.

Per tutti questi motivi sarebbe opportuno assegnare i nomi dei programmi da registrare dando loro anche l'estensione permessa dalla barra; ad esempio se salvassimo un programma in formato ASCII assegnandogli il nome CAVALLO3, è meglio scrivere:

**SAVE"CAVALLO3/TXT",A**

Nonostante il nome sia di 12 caratteri, esso viene accettato perché c'è il carattere /. Quel simbolo permette poi tre ulteriori caratteri.

Normalmente si usa la convenzione di usare TXT per i programmi in formato ASCII, BAS per quelli in formato compattato, SEQ per gli archivi sequenziali, RND per quelli di tipo random. Da notare che si tratta di una cosa facoltativa che potete pure non fare. Anche le sigle suggerite sono facoltative; nulla vieta, ad esempio, di adottare in loro vece sigle formate da una sola lettera (come T-B-S-R rispettivamente). La cosa utile, comunque, è che facendo una DIR sapete immediatamente di che tipo è ogni file, non affidandovi per questo solo alla vostra memoria. Pensate di prendere in mano un disco fatto qualche anno prima: credete veramente di ricordare se il file di nome CLIENTI23 è un programma oppure un archivio dati? Se la risposta è sì, invidiamo la vostra memoria.

## DUMP

Serve a registrare aree di memoria come FILE su disco. Per utilizzarlo bisogna specificare tre parametri e cioè:

**START** vale a dire l'indirizzo di partenza dell'area di memoria da registrare. NON può essere MAI INFERIORE a 7000.

**END** vale a dire l'indirizzo di stop dell'area registrata. NON può MAI essere INFERIORE all'indirizzo di START.

**TRA** è facoltativo. Indica l'indirizzo di ingresso (PROGRAM-COUNTER) nel caso che la memoria registrata sia un programma con l'attributo /CMD. Un esempio di questo tipo di programma potrebbe essere il FORMAT o il COPY già presenti nel disco NE-DOS.

Il parametro TRA è importante poiché un file con l'attributo /CMD diventa esecutivo appena caricato in memoria quindi è indispensabile specificare il punto di inizio del Programma. Se tale parametro viene omesso, si assume automaticamente TRA = 402D.

Esempio

**DUMP LOOP (START = X'7000', END = X'7122', TRA = X'7000')**

Rispettare esattamente gli spazi e la punteggiatura premere quindi RETURN. A questo punto il computer registra su disco la memoria compresa tra 7000 e 7122, il file creato avrà il nome LOOP a cui la macchina avrà attaccato l'attributo /CIM. (CORE-IMAGE).

Infatti se provaste a fare una DIR del disco troverete un file scritto come LOOP/CIM che è appunto quello appena registrato.

Se volete operare una DUMP specificando anche il drive, dovete scrivere:

**DUMP LOOP:1 (START = X'7000', END = X'7122', TRA = X'7000')**

in questo caso il file sarà registrato sul drive 1.

Riportiamo ora qualche esempio d'uso del comando DUMP in cui è stato commesso un errore:

**DUMP** seguito da RETURN, viene segnalato errore come FILE SPEC REQUIRED (mancano le specifiche del file).

**DUMP PIPPO (START = X'5000',END = X'6000')** viene segnalato START LESS THAN X'7000' (l'indirizzo di START è minore di 7000).

**DUMP PIPPO (START = X'7000',END = X'6000')** viene segnalato END LESS THAN START (l'indirizzo di END è minore di quello di START).

**DUMP PIPPO (START = X"8120",END = X"9000")**

**DUMP PIPPO**

In questi ultimi due casi non viene segnalato nessun errore e se eseguite una DIR sul disco trovate effettivamente i file che avete registrato, ma se provate a fare un LIST, di questi files, vi accorgerete che essi sono VUOTI vale a dire non contengono nessun dato.

Nel primo caso l'errore consiste nella sostituzione degli APICI (o APOSTROFO) con le virgolette nella specifica degli indirizzi, nel secondo caso, invece, manca addirittura la specifica stessa.

Se volete che il file registrato si comporti come un programma di utilità quale ad esempio il COPY/CMD o il FORMAT/CMD (già presenti sul NE-DOS), vale a dire, se volete utilizzare il vostro programma come funzione DOS, nella registrazione del file, dovete aggiungere l'attributo /CMD. Esempio

**DUMP GRAFIC/CMD (START = X'7000',END = X'7FCF',TRA = X'7200')**

potrebbe essere un esempio di registrazione di un programma per la gestione del grafico sul video, utilizzabile a livello DOS con la sola scritta GRAFIC. In questo caso il programma verrebbe caricato in memoria dalla 7000 alla 7FCF e inizierebbe a girare dalla locazione 7200.

Il DUMP è un comando DOS particolarmente utile in BASIC quando si fa uso di USR (routine utente). Infatti in BASIC, un programma che fa uso di una o più USR, viene salvato su disco solo come programma BASIC e non c'è altro modo di registrare anche le USR.

Utilizzando invece il comando CMD"DUMP ..... , è possibile registrare anche queste.

## FORMAT

Questo comando DOS è già stato spiegato nel numero precedente di NUOVA ELETTRONICA; per la descrizione di detto comando andate a rileggere quelle pagine.

## FREE

Riprendete la figura 1 del precedente numero della rivista: essa rappresenta la suddivisione in settori ed in tracce del dischetto formattato.

Le tracce sono 40 ed i settori sono 10; ogni traccia è formata da 2560 bytes, 256 per ogni traccia-settore. Se moltiplicate 2560 per 40 ottenete 102400: quello è il numero complessivo di bytes che può contenere un disco.

Un file registrato su disco è formato da uno o più segmenti di memoria; ogni segmento è composto da un minimo di 1 ad un massimo di 32 granuli.

A sua volta, ogni granulo è l'insieme di cinque tracce-settore.

Il granulo rappresenta la minima quantità di memoria di massa assegnata ad un file; se un file viene espanso, il granulo rappresenta ancora la minima quantità aggiunta al file precedente.

In ogni disco possono trovare posto al massimo 48 files.

Questa è la struttura creata dalla formattazione e dalla gestione dei files. Vi serve per capire il significato di quello che appare sul monitor facendo una DIR con l'opzione A (vedere il comando DIR), oppure quando usate il comando FREE.

Fate un DIR sul disco DOS-BASIC messo nel primo drive:

**DIR (A,I,S)**

cosa che avete già fatto alla voce DIR, se avete eseguito, come speriamo, tutti gli esempi forniti. Ottenete un lungo elenco di dati; prendiamo un file qualunque, ad esempio quello di nome SYS7/SYS, e vediamo di capire cosa significano tutti quei dati. A quel file corrisponde la seguente dicitura:

**SYS7/SYS SIP**
**LRL = 256 / EOF = 52**
**SIZE = 11 GRAN**

Otterrete i dati del file SYS7/SYS dopo aver pigiato due volte un tasto qualsiasi, se ricordate.

LRL = 256 significa 'lunghezza logica di ogni record = 256 bytes', e come vedete è uguale per tutti i files essendo una caratteristica del sistema di formattazione. EOF = 52 significa 'estensione del file = 52 settori', e varia da un file all'altro a seconda della loro lunghezza, come per la voce seguente. SIZE = 11 GRAN vuol dire 'ingombro 11 granuli'.

Se ricordate che la memoria su disco varia a gradini di un granulo per volta, e che un granulo è formato da 5 settori, vedete che i conti tornano. Infatti 11 granuli contengono 55 settori, quindi 52 settori in effetti danno un ingombro di 11 granuli in quanto in 10 granuli sono contenuti solo 50 settori; i successivi due settori comportano l'aumento minimo di memoria di massa, che è appunto di un granulo. Al solito, è più difficile dirlo che capirlo. Verificate i dati di altri files e vedrete che non è poi così complicato come sembra.

Ora, sempre dal DOS, scrivete:

**FREE**

Con questo comando ottenete, per ogni drive collegato al computer, due righe di dati simili a queste (che si riferiscono al disco DOS-BASIC):

**DRIVE 0 —— NE-DOS 01/01/82**
**44 FILES, 43 GRANS**

Il significato è ovvio: la prima riga dà il numero del drive, il nome del dischetto, e la data della sua formattazione. La seconda riga fornisce il numero dei files ancora disponibili (dei 48 iniziali), ed il numero dei granuli che sono a disposizione (per ampliamenti dei files esistenti o per la registrazione di nuovi files).

## KILL

La funzione svolta da questo comando DOS è esattamente uguale alla KILL del BASIC: serve per cancellare da un disco il file che porta il nome indicato.

Questo è l'unico comando DOS che risulti inutile a livello BASIC; il comando KILL del BASIC assolve esattamente alle stesse funzioni.

Se si commettono errori nel dare questo comando, si ottengono segnalazioni uguali a quelle già descritte; eviteremo quindi di ripeterle.

## LIB

Scrivete LIB quando siete a livello DOS: ottenete una lista di comandi che costituiscono la 'libreria' del sistema. In pratica ottenete la tabella 1.

In quell'elenco mancano, veramente, le parole BASIC, DEBUG e FORMAT. Il motivo di ciò va ricercato nel fatto che quelli non sono veri e propri comandi DOS, ma programmi da esso richiamati. Questa particolarità non comporta inconvenienti di sorta, se non quello che non vedrete mai scritte quelle parole chiave.

## LIST

Per potervi spiegare bene il comando LIST, dovete fare prima quanto segue.

Montate sul drive zero il disco DOS-BASIC, contenente il programma MOSTRA.

Caricate il BASIC e fate LOAD"MOSTRA" per trasferire quel programma in memoria. Allorché l'operazione sarà terminata, scrivete:

**SAVE"MOSTRA/TXT",A**

Conoscete già il significato di questa istruzione: registrate su disco il programma che è in memoria, chiamandolo MOSTRA/TXT; questo nome ci ricorda anche che si tratta di un programma scritto in formato ASCII. Essendo poi diverso da MOSTRA, sul disco ce li ritroveremo entrambi. Se avete eseguito il tutto a dovere, avrete lo stesso programma scritto nei due formati diversi ASCII e COMPATTATO.

Rimanete pure a livello BASIC, già che ci siete. Introducete il seguente comando DOS:

**CMD"LIST MOSTRA/TXT"**

Sul monitor appare il listato del programma MOSTRA/TXT. Badate bene che non si tratta del programma che avete in memoria; infatti se digitate NEW per cancellarlo e poi ripetete il comando precedente, ottenete lo stesso risultato.

A questo punto, provate a scrivere:

**CMD"LIST MOSTRA"**

Vedrete un susseguirsi di simboli alfanumerici e grafici; il tutto rappresenta il programma MOSTRA scritto in formato compattato.

Ora dovrebbero esservi più chiare anche tutte le considerazioni svolte in precedenza sui formati di scrittura su disco. In particolare noterete che le varie parole chiavi del BASIC sono scritte utilizzando un solo carattere grafico; noterete anche che il listato del compattato è più corto di quello scritto in ASCII: la differenza delle lunghezze dà l'idea dello spazio che ci si risparmia, sul disco, scrivendo i files in formato compattato.

Nel caso di programmi molto lunghi, potete arrestare il LIST con SHIFT , basta poi un tasto qualsiasi per continuare.

Non ripetiamo le segnalazioni d'errore che il computer manda in caso di comando dato male, in quanto sono sempre le stesse.

## LOAD

Questo comando DOS esplica funzione esattamente inversa al comando DUMP, vale a dire serve a caricare in memoria i files presenti su disco. Si usa con una sintassi di questo tipo:

**LOAD LOOP**
**LOAD LOOP/CIM**
**LOAD GRAFIC:1**

Nel terzo esempio è anche specificato il drive dove risiede il programma.

Gli unici due casi di errore sono:

**LOAD** viene segnalato FILE SPEC REQUIRED
**LOAD FILL** PROGRAM NOT FOUND se il file FILL non è presente nel disco.

Da BASIC usare al solito il comando CMD" LOAD ..... Potete usare il LOAD anche in un programma BASIC per caricare ad esempio una USR.

## PRINT

Questo comando DOS fornisce gli stessi risultati del comando LIST già visto, con la differenza che il programma richiesto viene listato dalla stampante.

I caratteri grafici vengono sostituiti da un asterisco.

## PROT

Nella descrizione del comando ATTRIB abbiamo già accennato al fatto che la funzione associata a PROT non l'abbiamo resa operante.

A titolo di curiosità, vi diciamo che PROT servirebbe per regolare l'uso della parola chiave principale del dischetto (PASSWORD), in modo da inibire certe operazioni. Potete tranquillamente ignorarla.

## RENAME

Col comando RENAME si assegna un nuovo nome ad un file già presente su disco.

Sempre col solito disco DOS-BASIC montato nel drive zero, scrivete da BASIC:

**CMD"RENAME:O MOSTRA TO PROVA":1**

Quando il comando è stato eseguito, fate una DIR e vedrete che il programma che si chiamava MOSTRA non compare più con quel nome, ma con quello nuovo PROVA.

Se il nuovo nome che si vuole assegnare è già presente sul dischetto, la segnalazione d'errore è la seguente, dal livello BASIC:

**DUPLICATE FILE NAME**
**UNPRINTABLE ERROR IN 15359**

Dal DOS manca la seconda riga. Le altre segnalazioni d'errore sono sempre quelle solite.

Il modo di dare il comando RENAME può anche essere così abbreviato:

**CMD"RENAME MOSTRA PROVA**

Come si vede, oltre che le virgolette ed il numero del drive, può essere omessa anche la parola TO.

## TIME

Il comando TIME serve per assegnare l'ora da depositare nella seconda parte della variabile speciale TIME$.

Se avete eseguito l'esempio riportato sotto il comando DATE, la prima parte è già stata assegnata; in caso contrario al posto della data troverete degli zeri.

Diamo quindi l'ora, supponendo di voler introdurre le ore 23, 15 minuti e 30 secondi (dal livello BASIC):

**CMD"TIME 23:15:30"**

Se è ancora memorizzata la data assegnata in precedenza, chiedendo la stampa della variabile TIME$ si ottiene:

**04/22/82 23:15:30**

Le eventuali segnalazioni d'errore sono le stesse che possono avvenire con il comando DATE.

## VERIFY

Assegnando il comando VERIFY il computer controlla che tutto il sistema di elaborazione e di scambio di dati coi dischi sia perfettamente a posto.

Il comando VERIFY va dato in questo semplice modo (se si parte dal BASIC):

**CMD"VERIFY"**

La verifica viene effettuata in pochi secondi.

## SUNTO DELLA GESTIONE DEI FILES

Dopo aver visto la descrizione di tutte le parole chiave del BASIC e del DOS, non ci resta che darvi alcune nozioni sulla gestione dei FILES. Per ragioni di spazio ci limiteremo ad un breve sunto; comunque è sottinteso che presto ritorneremo sull'argomento.

Nei prossimi numeri della rivista analizzeremo in dettaglio anche tutte le parole chiave di cui abbiamo dato finora solo una breve definizione. Seguiteci con fiducia e vedrete che in pochi mesi vi formerete un bagaglio notevole di nozioni sull'uso del BASIC e del DOS. Gran parte di questi concetti sono comuni a tutti i personal computers attualmente in commercio, quindi queste vostre conoscenze non vi serviranno solo per usare il calcolatore di NUOVA ELETTRONICA.

L'uso di ogni altro personal computer, a quel punto, sarà pressoché immediato; inoltre trarrete anche il massimo profitto dalla lettura di testi e riviste specializzate del settore. Chissà quante volte avrete provato ad acquistarne qualcuna, rinunciando però quasi subito a capirci qualcosa a causa del loro linguaggio altamente specializzato? Questo costituisce certamente l'unico grosso scoglio che deve superare chi, ignaro di tutto, si avvicina per la prima volta al mondo dei computers. Una volta superata la barriera della terminologia (che spesso è in inglese, come avete visto), il resto non è poi tanto difficile.

Una cosa è certa: solo l'uso continuo e corretto di un computer può dare in poco tempo dei buoni risultati. Fino a che vi limiterete a leggere testi o riviste, oppure a frequentare corsi più o meno cari e seri sull'uso dei computers, non approderete mai a nulla di concreto.

Seguiteci quindi con regolarità, cercando di approfondire meglio che potete tutte le nozioni che via via vi daremo. I risultati non si faranno attendere, e saranno tali da ripagare ampiamente la vostra (ed anche la nostra) fatica.

Ora daremo qualche cenno sull'uso dei FILES ESTERNI, vale a dire sugli archivi di dati registrati su dischetti.

I files possono essere di due tipi: SEQUENZIALI e RANDOM ( = CASUALI).

Vediamo di capire bene cosa significa.

Supponiamo di voler creare un archivio di nominativi, ad esempio quello dei nostri amici, con i relativi indirizzi e numeri di telefono. Se pensate che una stringa (ossia un insieme di lettere, numeri e caratteri vari) può contenere fino a 255 caratteri, capite che tutti i dati di ogni nominativo stanno comodamente in una stringa. Infatti 20-25 caratteri per nome e cognome, 35-40 caratteri per l'indirizzo, 10-12 caratteri per il telefono fanno in tutto, al massimo, 77 caratteri. Organizziamo quindi i nostri dati in un unico vettore (detto anche matrice ad una dimensione), cioè in un'unica variabile multipla di stringa. Chiamiamola N$(I), dove il numero variabile I serve per individuare i vari nominativi: N$(1) sarà il primo nome (con tutti gli altri dati che lo accompagnano), N$(2) sarà il secondo nome, e così via fino all'ultimo. Se in totale memorizziamo, ad esempio, 230 nomi, l'ultimo di essi lo troveremo sotto la variabile N$(230).

Il limite a questo discorso è imposto solo dalla dimensione della memoria RAM che abbiamo a disposizione, in quanto essa limita il crescere a dismisura del numero degli elementi in un vettore. Chiariremo questo concetto più avanti. Abbiamo quindi una lunga fila di stringhe, ognuna delle quali è contraddistinta da un numero progressivo. Quello è il nostro FILE di dati.

Quando incidiamo un FILE su disco, dobbiamo decidere se farlo SEQUENZIALE o RANDOM. La differenza consiste in questo: un FILE SEQUENZIALE può essere scritto o letto cominciando sempre e solo dall'inizio, mentre per il tipo RANDOM possiamo farlo dal punto che desideriamo. Ad esempio, se ci interessa il trentasettesimo elemento del nostro vettore, se abbiamo un FILE SEQUENZIALE dobbiamo cominciare a leggere il primo elemento, poi il secondo, e così via fino a quello che ci interessa. Se invece il nostro FILE è RANDOM è sufficiente dire al computer di andare a leggere l'elemento 37 per avere immediatamente e solamente quel dato. Ciò spiega anche i nomi che sono stati assegnati a questi due tipi di files.

Facciamo una analogia con una musicassetta incisa con canzonette; se non abbiamo il tasto dell'avanzamento veloce nei due sensi non ci resta altro che ascoltarla tutta dall'inizio per trovare il brano che ci interessa. Se invece quella possibilità esiste e se ci siamo segnate le cifre del contanastro che corrispondono ai vari brani, sarà facile e veloce trovare quello che ci serve. In termini di computer, non si parla di contanastro, ma di PUNTATORE. Esso, in pratica, è un numero che indica la posizione di un elemento del nostro file. Quando il puntatore vale 1, esso è posizionato sul primo nominativo; se vale 13 è sul tredicesimo e così via: punta sui vari elementi a seconda dei valori che assume.

Concludendo, in un FILE SEQUENZIALE possiamo leggere solo cominciando col puntatore posizionato all'inizio e procedendo spostandolo di una posizione alla volta, tra l'altro sempre solo in avanti. Invece in un FILE RANDOM possiamo scegliere dove posizionare il puntatore, e possiamo muoverlo a salti sia in avanti che all'indietro.

Speriamo di essere riusciti a chiarire bene il tutto; gli esempi che daremo in futuro vi saranno molto utili al riguardo.

Esistono altre differenze tra i due tipi di files. Infatti quelli sequenziali sono scritti in FORMATO ASCII, mentre quelli RANDOM sono in FORMATO COMPATTATO. Sapete già che questo significa che i secondi occupano meno spazio dei primi. Dovete poi sapere che ogni file, di qualunque tipo sia, per essere scritto o letto deve essere prima aperto: altrimenti sarebbe come voler suonare la musicassetta dell'analogia precedente senza averla tolta dalla custodia e senza averla

inserita nel registratore. In pratica, per poter scambiare dati con un file, occorre prima mettersi in collegamento con esso aprendo un canale di comunicazione; una volta instaurato questo 'filo diretto' possiamo spedire o ricevere messaggi. Alla fine della comunicazione dobbiamo 'chiudere' la linea. Vedremo che le parole chiave usate per fare tutte queste operazioni rispecchiano fedelmente proprio questo schema telefonico!

Ebbene, un file di tipo sequenziale necessita di diversi tipi di apertura a seconda se vogliamo leggerlo o scriverlo; invece un FILE RANDOM ha un solo tipo di apertura, che serve per entrambi gli usi.

Sembrerebbe che i vari elementi siano tutti a favore dei files di tipo random; in effetti, come vedremo tra poco, la gestione di questi files è un po' più laboriosa, specialmente per quello che riguarda i numeri.

Quindi, a seconda dei vari casi può essere più conveniente l'uso di un tipo di file invece che dell'altro.

## FILES DI TIPO SEQUENZIALE

Abbiamo detto poco fa che un file sequenziale necessita di diverse aperture a seconda che vogliamo scriverlo o leggerlo. Prima di scendere nei dettagli desideriamo farvi notare un'altra cosa importante. A differenza della quasi totalità dei personal computers oggi in commercio, in quello di NUOVA ELETTRONICA è possibile scrivere in coda ad un file sequenziale già presente su disco senza dover riscriverlo tutto. Questa grossa comodità si farà sicuramente apprezzare con l'uso, in quanto fa risparmiare errori, tempo e parti di programma che sarebbero altrimenti necessarie per fare un'aggiunta qualsiasi ad un file preesistente. Se, nell'esempio dell'archivio di nomi fatto prima, vogliamo aggiungerne altri 25, dobbiamo semplicemente aprire quel file nel modo che vedremo, e i dati incisi andranno in coda a quelli già esistenti. Negli altri computers dovreste leggere e memorizzare tutti i dati precedenti, poi riincidere il file dall'inizio aggiungendo i dati nuovi.

Andiamo allora con ordine e cominciamo con la gestione dei files sequenziali in registrazione.

Per aprire un file sequenziale allo scopo di registrarvi dei dati si deve scrivere (e la cosa normalmente avviene all'interno di un programma):

```
OPEN "O" , 1 , "AMICI"
```

Il significato di questa istruzione è il seguente: apri un file sequenziale per fare uscire dati dalla memoria ("O" sta per OUTPUT = uscita), riserva per la comunicazione il canale 1 ed assegna a quel file il nome "AMICI".

Più brevemente, è come dire: apri per registrare un file, chiamalo AMICI e passa per il canale 1. OPEN in inglese significa appunto 'apri'.

Noi abbiamo messo gli spazi per facilitare la lettura, ma essi non sono indispensabili. Il numero 1 è solo un esempio. Abbiamo parlato di canale di comunicazione, ma la terminologia esatta è BUFFER. Un buffer è una parte di memoria RAM in cui vengono accumulati i dati prima di trasferirli su disco (l'inverso se si è in lettura invece che in registrazione).

Il numero del buffer può essere compreso tra 1 e 15, ma in realtà risulta limitato superiormente dal numero che è stato dato in fase di caricamento del BASIC. Vedere al riguardo quello che è stato detto sotto il comando AUTO del DOS. Se il BASIC viene caricato senza specificare il numero massimo dei buffers, esso vale 3. Normalmente in un programma non è necessario avere più di tre files contemporaneamente aperti per scambiare dati con essi; quindi il limite di tre va bene. Ricordate allora che il numero dopo la prima virgola non viene accettato se risulta maggiore di tre, a meno che non si faccia l'estensione del numero dei buffers.

Anche il nome assegnato al file è solo esemplificativo; può essere sostituito da una variabile di stringa.

Ora il file è aperto, pronto per ricevere dati. Il modo per darglieli è il seguente.

```
PRINT # 1, N$(1)
```

Con questa istruzione ad esempio, si registra su disco la prima stringa del nostro vettore N$(I). Esisteranno poi altre istruzioni per registrare gli altri elementi del vettore. Quando abbiamo finito di passare dati dobbiamo chiudere il file, facendo in questo modo:

```
CLOSE 1
```

Se abbiamo un solo file aperto basta scrivere CLOSE; se i file aperti sono più di uno, con CLOSE li chiudiamo tutti. Per chiuderne uno solo occorre specificare il numero del buffer corrispondente, nel modo indicato sopra.

Riassumiamo l'intero processo, ricordando ancora che le righe che scriveremo in realtà saranno precedute dai numeri di linea del programma:

```
OPEN "O", "AMICI"
PRINT#1,N$(1),N$(2),N$(3),...
CLOSE
```

Vedremo a suo tempo esempi pratici; per questa volta abbiamo ampiamente superato lo spazio a nostra disposizione, quindi dobbiamo dare il minimo indispensabile di nozioni.

Vediamo ora come si gestisce invece un file sequenziale sempre in registrazione, ma che serve per registrare in coda ad un file già esistente.

L'intero processo è il seguente:

```
OPEN "E",1,"AMICI"
PRINT#1,N$(56),N$(57),...
CLOSE
```

L'unica differenza dal caso precedente sta nella lettera E al posto della O.

Gli elementi del vettore incisi sono solo d'esempio, come sempre.

Il modo corretto di gestire un file sequenziale in lettura è il seguente:

```
OPEN "I",1,"AMICI"
INPUT#,N$(1),N$(2),.....
CLOSE
```

Al posto delle lettere O o E c'è ora la I (da INPUT = entrata), e al posto di PRINT c'è INPUT.

Anche se con rincrescimento, dobbiamo passare ai files random, con la promessa di tornare quanto prima sull'argomento.

## FILES DI TIPO RANDOM

Per i files random daremo alcuni brevi cenni, che basteranno ai più esperti per fare le prime prove. Ritorneremo a parlare dell'argomento, estendendolo, sui prossimi numeri di NUOVA ELETTRONICA.

Il trattamento dei dati nei files random avviene a blocchi di 255 bytes per volta. Per sfruttare il disco occorre quindi cercare di riempire questa quantità; in caso contrario sprecheremo gran parte della capacità di un disco.

Dal momento che normalmente i singoli dati da trattare sono ben al di sotto di questa misura, occorre attaccarli assieme per cercare di avvicinarci il più possibile alla lunghezza complessiva di 255 bytes. Per tale motivo, dopo aver aperto un file random, bisogna assegnare il campo di lavoro dei vari dati. Tali dati, poi, possono essere incisi solo se compaiono sotto forma di stringhe. Abbiamo allora delle istruzioni per trasformare i numeri in stringhe.

Dato che esistono tre tipi di numeri (interi, in singola e in doppia precisione), abbiamo tre istruzioni per trasformare questi tre tipi di variabili numeriche in stringhe. Esse sono MKI$, MKS$ e MKD$. Ad esempio, per trasformare la variabile numerica intera A% in una stringa si può scrivere:

```
A$ = MKI$(A%)
```

Inversamente, abbiamo tre istruzioni che effettuano la trasformazione da stringhe a numeri. Esse sono CVI, CVS e CVD. Continuando con l'esempio precedente, dopo aver letto da disco la stringa A$ (che contiene il numero intero A%), per riottenere il numero dobbiamo scrivere:

```
A% = CVI(A$)
```

In pratica le istruzioni MK$ e CV sono concettualmente equivalenti alle STR$ e VAL che operano sulle stringhe.

Resta da dire che un numero intero occupa 2 bytes, un numero in singola precisione ne occupa 4, mentre uno in doppia precisione ne occupa 8.

Veniamo alla definizione di campo. Il blocco di 255 bytes va assegnato alle variabili (tutte di stringa) con una istruzione FIELD ( = campo).

Supponiamo di dover registrare diversi dati; essi sono rappresentati da tante stringhe, ognuna delle quali potrà anche essere di lunghezza variabile.

Nell'esempio dei nomi dei nostri amici, i nominativi, gli indirizzi, i numeri di telefono sono di varie lunghezze. Prendiamo i valori massimi dati prima, cioè 25-40-12 rispettivamente; in totale 77 bytes. Se noi mettessimo questo blocco di dati nei 255 bytes che abbiamo a disposizione, ne sprechiamo 178.

Dobbiamo perciò raggruppare più nominativi; nel nostro caso in 255 bytes ce ne stanno 3 e restano 24 bytes. Sarebbe quindi opportuno assegnare altri 8 bytes ad ogni nominativo (ad esempio, per scriverci i compleanni), in modo da sfruttare quello spazio che altrimenti perderemo.

Possiamo allora pensare di fare nel modo seguente. Indichiamo con la lettera N la stringa che contiene i 25 caratteri del nome, con I la stringa dei 40 caratteri dell'indirizzo, con T la stringa dei 12 caratteri del telefono e con A la stringa degli 8 caratteri delle annotazioni. A queste lettere facciamo seguire un numero da uno a tre per distinguere i tre blocchi di dati all'interno dei 255 caratteri in cui sono racchiusi.

Ora però c'è ancora un problema: un nome generico non sarà lungo esattamente 25 caratteri, ma ad esempio ne avrà solo 16; come facciamo? Esistono per questo problema altre due istruzioni apposite: LSET e RSET. Esse servono rispettivamente per posizionare i nostri dati a sinistra o a destra dei loro campi di definizione. Dal lato opposto resteranno degli spazi quando le stringhe saranno più corte del loro campo. Se invece saranno più lunghe, la parte eccedente sarà tagliata o a destra o a sinistra, rispettivamente.

Facciamo un esempio, per intenderci meglio.

Supponiamo che i tre blocchi di dati siano i seguenti:

```
N$(1) = "STROCCHI ROBERTO"
I$(1) = "VIA E. MATTEI 34 - CESENATICO (FORLÌ)"
T$(1) = "0574 86230"
A$(1) = "13/12/40"

N$(2) = "DOTT. BARGOSSI STEFANO"
I$(2) = "VIALE DELLE CERAMICHE 18 - FAENZA (RA)"
T$(2) = "0546 28309"
A$(2) = "23/08/48"

N$(3) = "MONTI FRANCESCA"
I$(3) = "PIAZZALE DELLA STAZIONE 13 - MILANO"
T$(3) = "011 820623"
A$(3) = "11/03/52"
```

Il modo di incidere questi dati in un file random, cominciando dall'inizio, potrebbe essere il seguente:

```
OPEN"R",3,"AMICI"
FIELD 1, 25 AS N1$, 40 AS I1$, 12 AS T1$, 8 ASA 1$, 25 AS N2$, 40 AS I2$, 12 AS T2$, 8 AS A2$, 25 AS N3$, 40 AS I3$,
      12 AS T3$, 8 AS A3$
LSET N1$ = N$(1)
LSET I1$ = I$(1)
LSET T1$ = T$(1)
LSET A1$ = A$(1)
LSET N2$ = N$(2)
.....
.....
```

```
LSET A3$ = A$(3)
PUT3,1
CLOSE 3
```

Ovviamente le parti costituenti le definizioni di FIELD non devono superare 255 bytes, mentre possono essere di un numero inferiore. I puntini sostituiscono le righe mancanti, il cui contenuto segue lo schema di quelle date.

Tutte le stringhe sono state posizionate alla sinistra dei loro campi di definizione.

I due numeri che accompagnano PUT hanno il seguente significato: il primo serve per individuare il buffer, il secondo dà la posizione del puntatore.

Per effettuare invece una lettura di dati prelevandoli da quel file random, dobbiamo cambiare solo l'istruzione PUT sostituendola con GET. Il resto rimane invariato.

Finiamo col farvi un esempio di quello che succede con LSET. Prendiamo il dato N$(2) che contiene la stringa 'DOTT. BARGOSSI STEFANO', lunga 22 caratteri. Dopo aver effettuato la lettura, il dato è depositato in N2$, che però è lungo 25 caratteri. Dato che con LSET abbiamo marginato a sinistra, N2$ nei suoi primi 22 caratteri contiene N$(2), ed i rimanenti 3 caratteri sono degli spazi.

Se avessimo fatto RSET anziché LSET, i tre spazi li avremmo ritrovati a sinistra.

Gli spazi nell'istruzione FIELD possono essere eliminati. Un file random deve necessariamente contenere una dichiarazione di FIELD e le assegnazioni LSET o RESET. In caso contrario si hanno segnalazioni d'errore o risultati sbagliati.

## CONCLUSIONE E ERRATA CORRIGE

Con questo articolo abbiamo praticamente concluso l'esame delle più importanti istruzioni del DOS e del BASIC. Alcune di esse sono state particolarmente approfondite; per altre le descrizioni sono state succinte, trattandosi di istruzioni più note e in tal modo speriamo per ora di avervi in parte soddisfatti.

Torneremo ripetute volte su questi argomenti rinnovando una rubrica più o meno fissa riguardante il computer.

Vi informiamo inoltre che è iniziato l'allestimento delle varie lezioni che formeranno il CORSO DI BASIC IN AUTOISTRUZIONE; quanto prima sarà pronto il primo dischetto con le tre lezioni iniziali. Ci sarà quindi soltanto bisogno di portare ancora un po' di pazienza.

Dovete convenire con noi che la preparazione di questi articoli sull'uso del DOS-BASIC è una cosa estremamente laboriosa ed impegnativa, data la grande quantità di cose da dire. L'argomento poi non è dei più immediati, e non deve meravigliare se ci sfugge qualche inesattezza o qualche errore tipografico.

La cosa più importante è sempre la solita: ogni imperfezione o errore, seppur piccoli, saranno via via corretti su queste pagine. Facciamo ora qualche annotazione sugli articoli del numero precedente.

Parlando, a pagina 97, della duplicazione dei dischetti, abbiamo detto che il procedimento di lettura dell'originale e di scrittura sulla copia dura quattro volte; in realtà potreste trovare la cosa inesatta, poiché quel numero dipende dalla disponibilità di memoria RAM che avete. Non preoccupatevi quindi se il processo è più lungo: esso avviene correttamente in ogni caso. Andate ora alla pagina 100, sempre del numero precedente. Negli esempi di variabili numeriche abbiamo commesso un errore dicendo che P e P! sono due variabili distinte; infatti qualche semplice prova vi convincerà che assegnando all'una o all'altra delle due un valore, anche la seconda assume quello stesso valore: P e P! sono depositate nella stessa variabile in singola precisione.

Portatevi ora a pagina 103, alla voce EDIT. Ai due esempi potete aggiungere anche questo:

```
> E.
```

Esso fa passare dal livello di comandi diretti al livello di editing nell'ultima linea di programma listata o editata. Se ci pensate, tale scrittura equivale poi alla pressione del tasto della virgola, come è stato spiegato nei comandi di editing.

Alla pagina 107, nella voce REF, ci siamo dimenticati di segnalare anche il seguente modo di impartire quel comando:

```
> REF$
```

Esso è simile a REF* (comprese le varianti date), ma fornisce i riferimenti anche sulla stampante e non solo sul monitor.

Infine, andate alla pagina 110, alla voce LOAD. La scritta SAVE"FATTURE":1 è errata. Quella esatta è SAVE"FATTURE:1".

Come vi siete resi conto, si tratta di piccole imprecisioni o dimenticanze, che molti di voi probabilmente avevano già corretto da soli.

Alla fine di questo articolo trovate la tabella degli errori e quella dei codici ASCII da 32 a 223 e il listato di un breve programma, se lo provate, vedrete sul monitor i caratteri ASCII del nostro computer. Abbiamo messo gli spazi tra le varie parole per renderlo più facilmente leggibile. Noterete però che dopo le parole chiave REVON e REVOFF tale spazio non c'è: se provate a metterlo il programma non gira, ed avrete la segnalazione d'errore SYNTAX ERROR IN 20. Ogni computer ha i suoi piccoli segreti; il precedente comportamento ne è un esempio. Niente di male, comunque; basta saperlo!

IMPORTANTE: Per un banale errore di duplicazione, alcuni lettori (pochi per fortuna), hanno ricevuto il dischetto NE-DOS in cui non girano correttamente le istruzioni: ON ... GOTO, ON ... GOSUB.

Questi lettori potranno, ovviamente, sostituire SUBITO il dischetto con uno corretto, oppure possono correggerlo direttamente loro utilizzando il programma in BASIC qui riportato.

Dopo aver scritto il programma in memoria, eliminate l'adesivo di protezione al disco, inseritelo nel DRIVE 0, quindi fate girare il programma con il RUN.

Vedrete il led posto sul DRIVE accendersi per qualche secondo dopodiché sul video riapparirà la scritta READY ad indicare che l'operazione è conclusa.

A questo punto pigiate RESET e ricaricate il BASIC corretto.

```
10  CLEAR 300 : OPEN "R", 1, "SYS7/SYS": FIELD 1, 255 ASA $ : GET 1,33 : B $ = A $ : MID $ (B $, 29,1) = CHR $
        (&H28):LSET A $ = B $
20  PUT 1,33 : CLOSE : END
```

## TABELLA DEGLI ERRORI

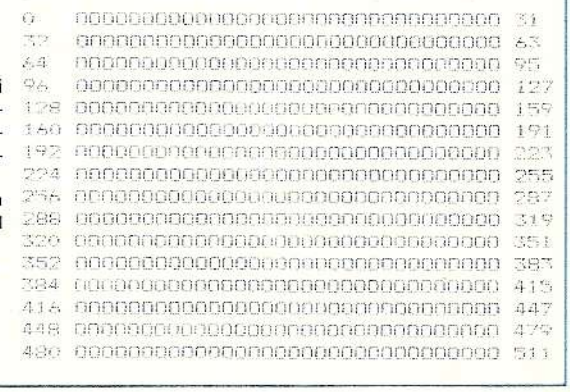
**NOTA** — L'errore viene depositato nella variabile speciale ERR.
Per ottenere il codice d'errore occorre chiedere: PRINT ERR/2 + 1.

| CODICE | MESSAGGIO | SPIEGAZIONE |
|---|---|---|
| 1 | NEXT WITHOUT FOR | NEXT senza FOR |
| 2 | SYNTAX ERROR | Errore di sintassi |
| 3 | RETURN WITHOUT GOSUB | RETURN senza GOSUB |
| 4 | OUT OF DATA | Fine DATA |
| 5 | ILLEGAL FUNCTION CALL | Funzione illecita |
| 6 | OVERFLOW | Numero troppo piccolo o troppo grande |
| 7 | OUT OF MEMORY | Manca memoria |
| 8 | UNDEFINED LINE | Linea inesistente |
| 9 | SUBSCRIPT OUT OF RANGE | L'elemento di matrice va oltre la DIM |
| 10 | RIDIMENSIONED ARRAY | Ridimensionamento di matrice |
| 11 | DIVISION BY ZERO | Divisione per zero |
| 12 | ILLEGAL DIRECT | INPUT come comando diretto |
| 13 | TYPE MISMATCH | Numero assegnato ad una stringa o viceversa |
| 14 | OUT OF STRING SPACE | Poco spazio per le stringhe |
| 15 | STRING TOO LONG | Stringa che supera i 255 caratteri |
| 16 | STRING FORMULA TOO COMPLEX | Operazione su stringa troppo complessa |
| 17 | CAN'T CONTINUE | CONT non può essere eseguito |
| 18 | NO RESUME | Non è stato dato RESUME |
| 19 | RESUME WITHOUT ERROR | RESUME senza ONERRORGOTO |
| 20 | UNPRINTABLE ERROR | Errore non stampabile |
| 21 | MISSING OPERAND | Manca l'operando |
| 22 | BAD FILE DATA | INPUT di dati non corretto |
| 51 | FIELD OVERFLOW | Riservati più di 255 bytes con FIELD |
| 52 | INTERNAL ERROR | Errore interno oppure di IN/OUT |
| 53 | BAD FILE NUMBER | Uso improprio di numero di file |
| 54 | FILE NOT FOUND | Nome di file inesistente |
| 55 | BAD FILE MODE | Uso errato del file |
| 56 | FILE ALREADY OPEN | File già aperto |
| 58 | DISK I/O ERROR | Errore nella trasmissione dati col disco |
| 59 | DUPLICATE FILE NAME | Nome di file già esistente |
| 62 | DISK FULL | Disco pieno |
| 63 | INPUT PAST END | Fine del FILE superata |
| 64 | BAD RECORD NUMBER | Numero di record superiore a 340 |
| 65 | BAD FILE NAME | Nome di file inammissibile |
| 67 | DIRECT STATEMENT IN FILE | LOAD-RUN-MERGE eseguiti su un file non BASIC |
| 68 | TOO MANY FILES | Più di 48 files in un disco |

## MAPPA VIDEO

Una pagina video è formata da 16 righe di 32 caratteri ciascuna. Il primo carattere della prima riga porta il numero zero; i caratteri successivi hanno numeri via via crescenti, fino ad arrivare all'ultimo carattere della sedicesima riga, che porta il numero 511.

Nella figura seguente, ogni lettera '0' rappresenta un carattere; all'inizio e alla fine di ogni riga c'è il numero del carattere d'estremità.



## CODICI ASCII DA 32 A 223

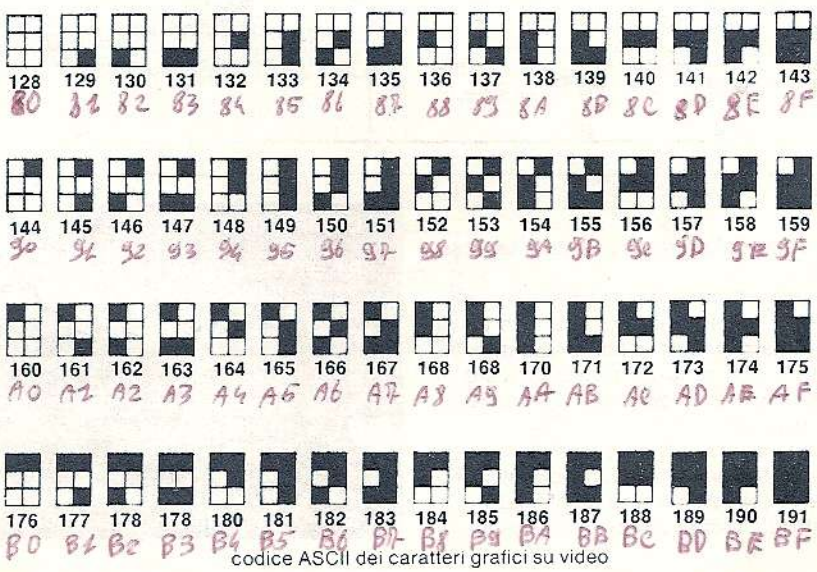
Con questo programma si ottiene sul monitor l'elenco dei caratteri ASCII del computer di NUOVA ELETTRONICA, dal codice 33 al codice 223. Al codice 32 corrisponde uno spazio. Il secondo alfabeto risulta scritto in maiuscolo, ma in effetti sarebbe minuscolo (lo sarà con la nuova scheda grafica).

Non mettete uno spazio dopo REVON e REVOFF, nella linea 20, altrimenti il programma non gira.

```
10 CLS : PRINT"NUOVA ELETTRONICA — COMPUTER Z80"
20 REVON:PRINT @ 39,"CARATTERI ASCII":REVOFF:PRINT
30 FOR I = 33 TO 223: PRINT CHR$(I);
40 IFI = 128 PRINT ELSE IF I > 128 AND I < 192 PRINT" ";
50 IF I = 144 OR I = 160 OR I = 176 PRINT ELSE IF I = 192 PRINT : PRINT
60 NEXT
70 GOTO 70
```

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 32 | — | spazio | 51 | — | 3 | 70 | — | F | 89 | — | Y | 108 | — | l |
| 33 | — | ! | 52 | — | 4 | 71 | — | G | 90 | — | Z | 109 | — | m |
| 34 | — | " | 53 | — | 5 | 72 | — | H | 91 | — | [ | 110 | — | n |
| 35 | — | ≠ | 54 | — | 6 | 73 | — | I | 92 | — | \ | 111 | — | o |
| 36 | — | $ | 55 | — | 7 | 74 | — | J | 93 | — | ] | 112 | — | p |
| 37 | — | % | 56 | — | 8 | 75 | — | K | 94 | — | ↑ | 113 | — | q |
| 38 | — | & | 57 | — | 9 | 76 | — | L | 95 | — | ← | 114 | — | r |
| 39 | — | ' | 58 | — | : | 77 | — | M | 96 | — | | 115 | — | s |
| 40 | — | ( | 59 | — | ; | 78 | — | N | 97 | — | a | 116 | — | t |
| 41 | — | ) | 60 | — | < | 79 | — | O | 98 | — | b | 117 | — | u |
| 42 | — | * | 61 | — | = | 80 | — | P | 99 | — | c | 118 | — | u |
| 43 | — | + | 62 | — | > | 81 | — | Q | 100 | — | d | 119 | — | w |
| 44 | — | , | 63 | — | ? | 82 | — | R | 101 | — | e | 120 | — | x |
| 45 | — | — | 64 | — | @ | 83 | — | S | 102 | — | f | 121 | — | y |
| 46 | — | . | 65 | — | A | 84 | — | T | 103 | — | g | 122 | — | z |
| 47 | — | / | 66 | — | B | 85 | — | U | 104 | — | h | 123 | — | [ |
| 48 | — | 0 | 67 | — | C | 86 | — | V | 105 | — | i | 124 | — | \ |
| 49 | — | 1 | 68 | — | D | 87 | — | W | 106 | — | j | 125 | — | [ |
| 50 | — | 2 | 69 | — | E | 88 | — | X | 107 | — | k | 126 | — | ↑ |
| | | | | | | | | | | | | 127 | — | ← |

**NOTA** — I codici ASCII compresi tra 192 e 223 non sono rappresentati in figura poiché coincidono con i caratteri alfabetici stampati in **REVERSE**.




codice ASCII dei caratteri grafici su video