istruzioni e comandi per

BASIC più DOS

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)		

SOMMARIO

INTRODUZIONE ..... pag. 2
LA NUOVA EPROM 5
CARICAMENTO DEL DOS-BASIC 6
FORMATTAZIONE DISCHI VERGINI 7
DUPLICAZIONE DEI DISCHETTI 10
NOZIONI PRELIMINARI SUL BASIC 12
VARIABILI NUMERICHE 13
VARIABILI ALFANUMERICA 14
OPERATORI ARITMETICI E LOGICI 15
CODICE ASCII 16
GRAFICA E MAPPA VIDEO 17
COMANDI DOS 18
APPEND 19
ATTRIB 22
AUTO 23
BASIC 26
COPY 26
DATE 26
DEBUG 27
DIR 31
FORMAT 35
FRE 35
KILL 36
LIB 36
LIST 36
LOAD 36
PRINT 37
PROT 37
RENAME 37
TIME 37
VERIFY 37
COMANDI E ISTRUZIONI BASIC 38
AND, AUTO, CLEAR, CLOSE 39
CCMD"D, CMD"funzione DOS", CMD"S, CONT 39
DATA, DEFDBL, DEFFN, DEFINT 39
DEFSNG, DEFSTR, DEFUSR 39
DELETE, DIM, EDIT, END 40
EOF, ERL, ERR, ERROR 40
FIELD, FOR...NEXT, FRE 41
GET, GOSUB, GOTO, IF...THEN...ELSE 41
INKEY$, INPUT, INPUT# 42
INSTR, KILL, LET, LINEINPUT 42
LINEIMPUT# , LIST, LLIST, LOAD 43
LOC, LOF, LPRINT, LSET 43
MEM, MERGE, NEW, NOT, ON...GOSUB 44
ON...GOTO, ONERRORGOTO, OPEN 44
OR, PRINT, PRINT# , PRINT@ 45
PRINTTAB, PRINTUSING, PUT, READ 45
REF, REM, RENUM, RESTORE 46
RESUME, RETURN, REVOFF, REVON, RSET 46
RUN, SAVE, STOP 47
SYSTEM, TROFF, TRON 47
DESCRIZIONE DELLE ISTRUZIONI BASIC 47
MEM, SAVE, LOAD 48
TABELLA DEGLI ERRORI 50
FUNZIONI DI STRINGA 51
ASC, CHR$, CVD, CVI 51
CVS, INSTR, LEFT$, LEN, MID$ 52
MKD$, MKI$, MKS$, RIGHT$ 53
STR$, STRING$, VAL 53
FUNZIONI ARITMETICHE 54
&H, &0, ABS, ATN, CDBL 54
CINT, COS, CSNG, E, EXP, FIX, INT 55
LOG, RANDOM, RND, SGN 56
SIN, SQR, TAN, ↑ 56
FUNZIONI GRAFICHE 57
CLS, POINT, RESET, SET 57
FUNZIONI SPECIALI 57
INP, OUT, PEEK, POKE 58
POSE, TIME$, USR 58
VARPTR 59
SINBOLI CHIAVE 59-62
CONTROLLI SPECIALI DA TASTIERA 62-63
COMANDI DI EDITING 64-73
SUNTO DELLA GESTIONE DEI FILES 73
FILE DI TIPO SEQUENZIALE 75
FILE DI TIPO RANDOM 76
LA GESTIONE DEI FILES 78
SCHEMI A BLOCCHI E DIAGRAMMI DI FLUSSO 78
PROGRAMMA GESTIONE AZIENDALE 80
RIFERIMENTI DEL PROGRAMMA 84
SIMBOLI DEI FLOW-CHART 86
SEMPLICI PROGRAMMI 115

Oggi, costruire e rendere funzionante un computer non è più un'impresa limitata esclusivamente a determinate persone, anzi, è forse più facile montare un computer che realizzare un amplificatore di BF.

Oltretutto rinunciando, a priori, di interessarsi a questo nuovo settore dell'elettronica potrebbe significare in futuro essere considerati 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 a capirci qualcosa, né può essere una scusante il fatto che quando si parla di computer è necessario imparare termini astrusi e apparentemente incomprensibili come DEBUG, EDITING, MONITOR, DOS, DIRECTORY, CPU, SQR o altre cose di questo genere. Poiché nessuno di essi nasconde dietro di sé cose tanto difficili da risultare incomprensibili, l'unico ostacolo esistente è cercare di assimilare questi termini anglosassoni usati in qualsiasi sviluppo di programma, e per questo, non occorre molto tempo anche se non si ha una perfetta conoscenza delle lingue estere.

Una volta tradotti questi termini e spiegato in parole povere che cosa significano, tutto diviene

ISTRUZIONI sull'uso

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 sorprenderci.

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 lamentano 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 insegnare 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 programmarne.

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».

del BASIC e del DOS

IMPORTANTE

Per poter lavorare con questo BASIC + NE-DOS che occupa un totale di 26K occorre che il computer risulti così configurato:

SCHEDA CPU,

INTERFACCIA VIDEO,

MONITOR VIDEO,

INTERFACCIA FLOPPY DISK,

ALMENO UN DRIVE PER FLOPPY DISK,

MEMORIE RAM PER ALMENO 40 KILOBYTE.

Naturalmente al posto del monitor come già sapete potrete utilizzare anche un comune televisore.

Sarebbe poi quanto mai consigliabile avere una stampante da 80 colonne ad aghi: in quanto solo con essa si possono fare bolle di accompagnamento, fatture, contabilità, comunicazioni varie, cataloghi, lettere, elenchi. Può andar bene utilizzare una stampante termica, ma non per lavorare a livello commerciale.

- LA NUOVA EPROM

Coloro che ci hanno seguito finora nella realizzazione del computer avranno senza dubbio già apportato le modifiche indicate sul numero 76 e 77 di NUOVA ELETTRONICA per quanto riguarda la scheda CPU, l'espansione di memoria e l'INTERFACCIA FLOPPY.

Assieme al disco BASIC + NE-DOS avete ricevuto anche una Eprom siglata 1.390 che dovrete sostituire nella scheda floppy LX.390 con quella già esistente siglata 390 per questi semplici motivi:

1) Con la nuova EPROM formattiamo i dischetti floppy (vedi fig. 1) in modo da aumentare la capacità che da 78 kilobyte passerà così a 102,4 kilobyte.

2) Il disco BASIC + NE-DOS contiene molti programmi di utilità e a questi sono stati aggiunti pure quelli di formattazione e di duplicazione. Dopo il caricamento del disco, in pochi secondi è pronto il BASIC e pur se la cosa non è evidenziata, è stato caricato anche il DOS. Tutti gli altri programmi di utilità sono disponibili immediatamente portandosi a livello DOS.

Quando inserirete la nuova EPROM al posto della vecchia, nella scheda INTERFACCIA FLOPPY, controllate che i piedini siano inseriti tutti correttamente nei corrispondenti fori dello zoccolo e che la tacca di riferimento risulti rivolta come per la vecchia EPROM.

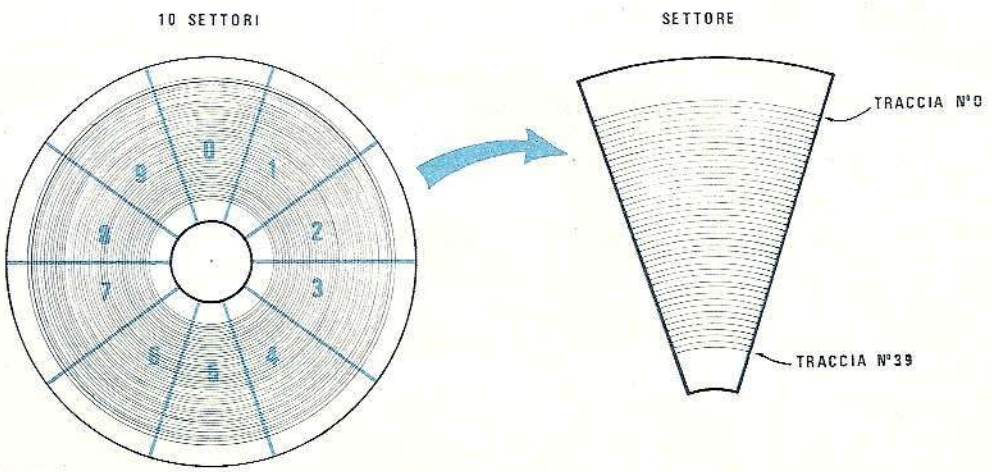


Fig. 1 Con questo Basic, il dischetto del floppy viene formattato a 10 settori anziché a 17 come avevamo in precedenza indicato a pag. 103 del n. 75 della rivista Nuova Elettronica. Così facendo si aumenta la capacità in kilobyte che da 78 passerà a 102,4 kilobyte. Con tale formattazione sulla scheda LX.390 va sostituita la Eprom con la nuova siglata 1.390.

- CARICAMENTO DEL DOS E DEL BASIC

Con la nuova EPROM inserita sull'interfaccia Floppy e con almeno 32 K di RAM montati accendete il computer e pigiate il pulsante di reset posto sulla scheda CPU. Sul monitor apparirà la scritta:

```
BOOTSTRAPPING VERSION 1.0          (lancio versione 1.0)
INSERT NE-DOS-DISK AT DRIVE 0      (inserire disco NE-DOS nel drive 0)
THEN TYPE "SPACE"                  (poi premere "spazio")
```

Come si vede, lo scambio di informazioni col sistema avviene in lingua Inglese, che domina sovrana in questo campo. Se non aveste dimestichezza con l'inglese non preoccupatevi: le parole da conoscere sono poche e con l'uso frequente le imparerete velocemente. Ad ogni buon conto tenete sempre sott'occhio le tabelle che forniremo, in modo da poterle consultare facilmente.

Dopo che è apparsa sul video la scritta, inserite il disco NE-DOS nel DRIVE 0 quindi pigiate il tasto dello SPAZIO, cioè quello più lungo posto in basso sulla tastiera alfanumerica.

Dopo circa 15 secondi il computer comincia l'esecuzione del programma dimostrativo MOSTRA.

Nell'intervallo di tempo menzionato il computer ha caricato in memoria il DOS, il BASIC e per ultimo il programma MOSTRA e parte con la sua esecuzione.

Il programma MOSTRA da solo non si fermerà più, quindi dovete farlo voi battendo contemporaneamente i due tasti BREAK.

Pigiate ora il tasto DEL (serve per cancellare il video e togliere la scrittura espansa), poi il tasto RETURN e successivamente i due tasti CTRL e T contemporaneamente (serve per togliere la scrittura in negativo). Ovviamente se all'atto del BREAK non c'era scrittura espansa potete risparmiarvi la prima operazione, come potete non fare la seconda se non era in atto la scrittura in negativo (REVERSE).

Per poter proseguire dopo aver pigiato il tasto DEL occorre premere anche il tasto RETURN.

Se vi sbagliate alla partenza e non inserite il dischetto del DOS, tra le due prime scritte appare:

```
NO NE-DOS-SYSTEM                   (non c'è il dischetto NE-DOS)
```

Tale scritta appare anche se esistono errori di lettura dovuti magari alla testina sporca o a mancanza di schermatura sul DRIVE. Se invece avete battuto lo «spazio» senza inserire nessun disco nel drive 0, dopo poco il drive stesso si ferma e il **led** rimane acceso; a questo punto potete **solo** premere nuovamente il tasto RESET sulla scheda CPU e ripartire da capo.

Se in seguito ogni qualvolta ricaricate il BASIC non volete avviare anche il programma MOSTRA, dovete procedere come segue: alla partenza, invece di premere il tasto SPAZIO, premete il tasto RETURN e tenetelo premuto fino a quando il drive non si ferma; a questo punto lasciate il tasto RETURN e sul video vedrete apparire la scritta:

```
DOS READY                          (DOS pronto)
```

Per passare al BASIC basta scrivere sul video la parola BASIC e pigiare il tasto RETURN: dopo circa sette secondi appare la scritta

```
READY
>.
```

E questo significa che siete a livello BASIC.

Vediamo ora brevemente cosa vuol dire LIVELLO DOS e LIVELLO BASIC.

LIVELLO DOS

(DOS = DISK OPERATIVE SYSTEM).

Questo è il livello del SISTEMA OPERATIVO, che rende possibile l'uso dei Floppy Disks. Questo livello può essere raggiunto in via diretta o attraverso il BASIC.

A livello DOS si possono eseguire diversi programmi di utilità (come formattare dischetti vergini, fare copie di dischetti, ecc.).

LIVELLO BASIC

(BASIC = **B**EGINNER'S **A**LL-PURPOSE **S**YMBOLIC **I**NSTRUCTION **C**ODE).

A questo livello è possibile eseguire comandi diretti, oppure scrivere, correggere o eseguire programmi in linguaggio BASIC. Ovviamente dal livello BASIC si può passare al livello DOS o viceversa.

Non fatevi prendere dalla frenesia di voler eseguire subito dei programmi e imparare tutto in una sola volta; le nozioni sono abbastanza numerose e collegate tra di loro, esporle tutte assieme è abbastanza difficile. Cercate sempre di assimilare bene un concetto prima di passare a quello successivo. Seguiteci e vedrete che poco per volta acquisterete familiarità col computer.

- DOS

Dopo aver caricato il BASIC, è possibile passare al livello DOS scrivendo semplicemente:

```
CMD"S"
```

e pigiate il tasto RETURN per rendere operativo il comando impartito.

Ricordatevi sempre che dopo aver scritto una qualsiasi istruzione per renderla effettiva, **occorre sempre pigiare il tasto RETURN.**

Dopo aver pigiato il tasto RETURN, sul monitor apparirà la scritta:

```
DOS READY
```

la quale ci confermerà che ci troviamo al **livello DOS.** Dopo che appare tale scritta per tornare a livello BASIC sarà sufficiente scrivere sulla tastiera.

```
BASIC
```

e ovviamente dopo averlo scritto dovremo ricordarci di battere il tasto RETURN e dopo poco sul monitor vedremo la scritta:

```
READY
>.
```

e questo ci conferma che siamo ritornati al livello BASIC.

Ritornando alla scritta DOS READY potremo ora imparare a formattare i dischetti vergini e a duplicarli.

- FORMATTAZIONE DEI DISCHI VERGINI

Tutti i dischetti vergini per risultare idonei a ricevere dei dati devono essere prima formattati. Per far questo occorre sapere se avete un solo drive, oppure due o più drive in quanto il procedimento è leggermente diverso.

AVETE UN SOLO DRIVE

Dopo la comparsa della scritta DOS READY scrivete sulla tastiera

```
FORMAT
```

e pigiate il tasto RETURN.

IMPORTANTE: d'ora in poi l'operazione «pigiare il tasto RETURN» anche se verrà omessa per non complicare inutilmente la descrizione, dovrà essere sempre eseguita.

Dopo poco appare la scritta

```
1  NEDOS DISK FORMATTER PROGRAM 1.0    (programma 1.0 per formattare dischi)
   WHICH DRIVE IS TO BE USED?          (quale drive vuoi usare?)
```

ossia il computer ci informa che il programma di formattazione è pronto per essere avviato e prima di iniziare ci chiede quale DRIVE vogliamo usare. A questo punto avendo un solo drive dovrete togliere dal drive il disco col DOS-BASIC e sostituirlo con quello vergine da formattare.

Ricordatevi che se avete un solo DRIVE il suo numero è **0**, quindi dovrete pigiare il tasto dello 0 (zero) attenzione a non confonderlo con la lettera O.

NOTA - Se pigiate il numero 0 senza aver prima cambiato il dischetto con uno vergine, sul monitor vedrete scritto

```
2 WRITE PROTECTED DISKETTE!          (Il disco è protetto contro la scrittura)
  REPLY "N" TO END FORMAT.           (scrivi N se vuoi uscire dalla formattazione)
  REPLY "Y" AFTER ERROR CONDITION    (scrivi Y dopo aver rimediato all'errore e
  CORRECTED TO EXECUTE FORMAT        se vuoi proseguire nella formattazione
  COMMAND AGAIN.                     del disco inserito)
```

Cosa è successo? Semplicemente che il sistema si rifiuta di formattare il disco col DOS, perché ha il nastro adesivo di protezione, e quindi non può eseguire il comando FORMAT.

Ecco perché il computer vi offre due alternative:

se rispondete **N** (N = NO) uscite dal comando di formattazione,

se rispondete **Y** (Y = YES) prosegue con la formattazione. Prima di rispondere Y occorre togliere il disco DOS-BASIC e inserire il disco vergine.

Conviene sempre, comunque, scrivere il numero 0 'zero' solo dopo aver cambiato il dischetto; se succede il contrario, è preferibile rispondere alla domanda successiva con N, in quanto può accadere che il sistema perda il controllo e che si debba ricominciare da capo col caricamento del DOS.

Per completare il quadro della situazione, ricordiamo che lungo il bordo del dischetto è ricavato un intaglio rettangolare di circa cinque millimetri; qualora si chiuda questa tacca con una strisciolina di adesivo non è più possibile registrare dati su quel disco. Anche in questo caso quindi il sistema rifiuta la formattazione. (All'interno di ogni confezione di dischi si trova un adesivo per questo uso).

Scrivendo il numero **0** sul video apparirà questa scritta:

```
3 DISKETTE NAME?                     (quale nome vuoi dare al dischetto?)
```

cioè viene richiesto il nome da assegnare al dischetto.

Potete rispondere introducendo un nome (il vostro, ad esempio MARIO) o, qualche lettera; esempio:. NE, oppure DISKA, DISKB ecc.

se non scrivete nulla e battete solo RETURN il computer vi porrà nuovamente la domanda. La lunghezza del nome non deve superare gli otto caratteri: ve ne accorgerete subito perché oltre tale limite il cursore non si muove più.

Eseguita questa operazione, vedrete sul video la scritta

```
4 CREATION DATE                      (data di formattazione)
  (MM/DD/YY)?                        (mese/giorno/anno)
```

Dovete cioè introdurre, la data di formattazione. Notare che questa viene richiesta nella forma anglosassone, cioè prima il **mese** poi il **giorno** e quindi l'**anno**. Conviene rispettare questo ordine, altrimenti può succedere che la data venga rifiutata per un'evidente incompatibilità di valore nel numero del mese. Questo succede solo quando la data deve essere introdotta nel comando di duplicazione, che vedremo tra poco; in quel caso se la prima cifra di MM non è un numero oppure è un numero uguale o maggiore di due, la data è rifiutata dal sistema. Tanto vale quindi abituarsi ad introdurla nel modo richiesto, oppure a metterne una di comodo sempre uguale, ad esempio 11/11/11.

Rispettate rigorosamente la forma, vale a dire scrivere sempre per il mese due cifre (per 1-2-3 occorre sempre scrivere 01-02-03), poi la barra, poi altre due cifre, poi un'altra barra e successivamente le ultime due cifre. **Esempio:** 03/07/82 - 03/18/82 - 10/30/82.

Se non rispettate queste indicazioni il sistema non accetterà quanto introdotto e ve lo richiederà.

Dopo aver scritto la data appare la seguente domanda

```
5 MASTER PASSWORD?                   (parola chiave principale?)
```

Qui valgono esattamente tutte le considerazioni fatte poc'anzi per rispondere alla domanda DISKETTE NAME. Comportatevi quindi allo stesso modo. In entrambi i casi, se volete far presto, potete limitarvi a battere uno spazio e il tasto RETURN.

Non preoccupatevi per ora di sapere esattamente a cosa servono questi dati; li vedremo andando avanti nella descrizione.

Proseguendo nella formattazione. Dopo aver risposto all'ultima domanda, sul monitor compare

```
6 DO YOU WANT TO LOCK OUT            (vuoi inibire
  ANY TRACKS?                        qualche traccia?)
```

il computer chiede se si vuole bloccare qualche traccia. Se rispondete con Y (YES) vi verrà posta la seguente domanda

```
WHICH TRACKS (1-39)?                 (quale traccia da 1 a 39?)
```

La risposta deve consistere in uno o più numeri compresi tra 1 e 39, separati da un trattino (—). Rammentatevi che le tracce sono 40 numerate da 0 a 39; la traccia 0 è riservata al sistema, e per questo motivo nel dischetto è esclusa dal bloccaggio, quindi apparirà sempre 1-39.

Dopo aver dato il numero delle tracce che intendete bloccare, appare la scritta

```
FORMAT THE LOCKED OUT TRACKS?    (vuoi formattare le tracce inibite?)
```

cioè il sistema chiede se deve formattare anche le piste interdette. Rispondete con una Y se è si, e con una N se è no.

Raramente se non in casi particolari si ha necessità di bloccare qualche traccia quindi alla domanda 6 rispondete sempre con una N.

Se per ipotesi avete messo nel drive un disco già formattato (con o senza dati); sullo schermo apparirà questa scritta

```
FORMAT REJECTED,                    (formattazione respinta)
DISKETTE CONTAINS DATA!             (il disco contiene dati)
DISKETTE TO BE FORMATTED ANYWAY     (vuoi formattare ugualmente il disco?)
(REPLY "Y" OR ""N")?                (rispondi "Y" oppure "N")
```

cioè il computer vi informa che il disco inserito è già formattato e contiene dei dati; quindi prima di procedere dovremo confermare se effettivamente vogliamo formattarlo (il che significa anche cancellare tutti i dati) ed in questo caso scriveremo Y, se invece distrattamente abbiamo preso un disco che conteneva dei dati e che noi ritenevamo vergine, per salvarlo dovremo scrivere N.

Ritornando alla domanda 6 alla quale avremo scritto N in quanto non ci interessa bloccare nessuna traccia, il computer inizierà a formattare il dischetto, facendo apparire sul video il seguente messaggio

```
DIRECTORY WILL BE PLACED    (l'indice sarà messo
ON TRACK 17                 alla pista 17)
```

significa che il computer colloca nella traccia 17 l'elenco dei programmi; poi sotto si vedrà la dicitura.

```
FORMATTING TRACK 'n'        (sto formattando la traccia n)
```

dove il numero 'n' varia da 0 a 39 mano a mano che l'operazione di formattazione procede. Raggiunto il numero 39, sul video apparirà questa scritta

```
VERIFYING TRACK 'n', SECTOR 'm'  (sto verificando la traccia n, settore m)
```

dove 'n' varia come prima e 'm' va da 0 a 9: il computer cioè verificherà automaticamente, traccia per traccia e settore per settore controllando che la formattazione risulti corretta e che non vi siano tracce difettose nel disco.

Può succedere (anche se molto di rado) che il dischetto risulti difettoso in qualche sua parte. In tal caso potreste vedere sul monitor una dicitura come questa

```
7 VERIFYING TRACK 18, SECTOR 04  (sto verificando la traccia 18, settore 04)
  NOT FOUND! TRACK LOCKED OUT!   (non l'ho trovata! traccia inibita!)
```

In sede di verifica nella formattazione il computer si è accorto che la traccia 18 del settore 04 non si è formattata correttamente, e in questo caso la considera non disponibile per le future incisioni, e prosegue oltre.

Se non trova nessuna traccia difettosa apparirà invece

```
8 INITIALIZING SYSTEM INFORMATION  (informazione nella inizializzazione)
  FORMATTING COMPLETE              (formattazione completata)
  PRESS "ENTER" WHEN **SYSTEM**    (premere "ENTER" quando il disco
  DISKETTE MOUNTED ON DRIVE 0      **SISTEMA* è montato sul drive 0)
```

La formattazione è terminata quindi potete togliere dal drive il dischetto formattato e inserirvi nuovamente quello col DOS-BASIC. Premendo il tasto RETURN o ENTER verrà ricaricato il DOS (se non succede niente o se vedete simboli strani, premete il RESET della scheda CPU LX.382 e ricaricate il DOS nel modo esposto all'inizio).

Viceversa, se qualcosa non è andato come di dovere, il computer segnala errore e si deve ripetere il procedimento da capo.

Se vi succede di fare un BREAK durante una qualsiasi delle fasi di formattazione, il computer non risponderà più a nessun comando. In tal caso dovrete ripartire da capo nel modo solito: pigiando il tasto RESET sulla scheda CPU e ricaricando il DOS-BASIC.

Se non riuscite a proseguire in nessun modo, controllate se siete a livello DOS oppure se il disco inserito è alla rovescia (ricordate che l'etichetta del disco va dalla parte opposta del LED).

Non lasciatevi comunque trarre in inganno dalla lunghezza di queste spiegazioni: a dirlo sembra lungo e difficile, a farlo ci si rende conto che invece è estremamente facile. A titolo informativo diremo che il tempo necessario per formattare un disco, si aggira su 50 secondi.

Un'altra cosa: se la velocità del drive non è quella giusta, il sistema ve lo segnala con

```
CAN'T FORMAT,                (non posso formattare)
MOTOR SPEED TOO FAST (SLOW)! (velocità del motore troppo alta o bassa)
```

```
REPLY "N" TO END FORMAT.          (rispondi "N" per uscire dalla formattazione)
REPLY "Y" AFTER ERROR CONDITION   (rispondi "Y" dopo che la condizione d'errore
CORRECTED TO EXECUTE FORMAT       è stata corretta per eseguire
COMMAND AGAIN.                    nuovamente la formattazione)
```

Vale a dire: è impossibile formattare perché la velocità del motore è troppo alta (o bassa), ecc. Come vedesi abbiamo un sistema operativo molto «intelligente» che ci segnala anche i difetti del drive.

E c'è dell'altro. Se non è stato chiuso lo sportellino del drive o se non è stato inserito il dischetto, il computer segnala una o entrambe le condizioni con le scritte

```
DOOR NOT CLOSED ON DRIVE!         (lo sportello del drive non è chiuso)
NO DISKETTE IN DRIVE!             (non avete inserito il dischetto)
REPLY "N" TO ....                 (idem come sopra)
```

Da quanto spiegato finora potete comprendere che il DOS-BASIC da noi fornito è quanto di meglio esiste attualmente in commercio.

Per facilitare il più possibile l'operazione di formattazione, ecco un breve sunto di quello che dovete fare (i numeri tra parentesi richiamano i messaggi già menzionati).

— PORTARSI A LIVELLO DOS
— DIGITARE FORMAT
— ALLA DOMANDA (1) RISPONDETE DIGITANDO 0
— ALLA DOMANDA (3) DATE UN NOME A VOSTRA SCELTA
— INTRODUCETE LA DATA ALLA (4) ESEMPIO 11/11/11
— DIGITATE UN NOME O UNO SPAZIO ALLA DOMANDA (5)
— RISPONDETE N ALLA DOMANDA (6)
— QUANDO APPARE LA (8) LA FORMATTAZIONE È FINITA, RIMETTETE NEL DRIVE IL DISCO BASIC PREMETE POI IL TASTO RETURN E TORNERETE AL LIVELLO DOS.

AVETE DUE O PIÙ DRIVE

Avendo due o più drive vi risparmiate di togliere il disco col DOS-BASIC dal drive 0: infatti il disco da formattare lo metterete nel drive 1 e alla prima domanda (quale drive intendete usare) risponderete pigiando il tasto del numero 1.

Tutto si ripeterà come nel caso precedente e arrivati alla fine, il sistema torna da solo al livello DOS.

DUPLICAZIONE DEI DISCHETTI

Vediamo ora un'altra operazione importante che è possibile fare col sistema operativo DOS: la DUPLICAZIONE dei dischetti.

Come già sapete, non si possono fare operazioni su dischetti vergini quindi per fare una duplicazione occorre disporre di un disco già formattato.

Parleremo innanzitutto del caso in cui abbiate un solo drive.

La partenza deve essere sempre fatta dal livello DOS (vedremo poi che tutte le operazioni DOS sono fattibili anche a livello BASIC).

Se vogliamo ad esempio fare una copia del disco DOS-BASIC, cosa che del resto è estremamente consigliabile in quanto il dischetto potrebbe rovinarsi per tanti motivi (quindi meglio averne una o due copie fedeli di scorta) procederemo come segue.

Prima di partire controllate che il disco DOS-BASIC abbia il nastro adesivo di protezione contro le registrazioni. In caso contrario, per una errata manovra, potreste cancellarlo.

A livello DOS scrivete quanto segue

```
COPY :0 TO :0 01/25/82
```

È IMPORTANTISSIMO LASCIARE GLI SPAZI INDICATI: cioè 0 (spazio) TO (spazio): 0 (spazio) 01/25/82 se non li mettete o ne inserite più di uno il computer segnala errore con

```
BAD FILESPEC                      (errore di procedura)
```

Anche i due punti sono importanti, ovviamente. Attenzione anche a non confondere lo zero con la lettera O. Segue poi la data, da dare nel modo che abbiamo già visto per la formattazione. Quella fornita vale come esempio quindi potrete sceglierne anche una diversa.

Se avete scritto la data in modo errato, esempio 1/25/82 o 25/01/82 la segnalazione è

```
BAD DATE                          (errore di data)
```

quindi occorre riscriverla. Se tutto è corretto, apparirà il seguente messaggio

```
9  PRESS "ENTER" WHEN SOURCE        (premere "ENTER" quando il disco sorgente
   DISKETTE MOUNTED ON DRIVE 0      è montato è nel drive 0)
```

ossia premere il tasto RETURN quando il disco sorgente è nel **drive 0.**

Dato che il disco da duplicare è quello del DOS-BASIC (quindi inserito nel DRIVE 0) potete premere il tasto RETURN. Il drive partirà e sul monitor leggerete

```
10 READING TRACK, SECTOR n,m        (sto leggendo la traccia e il settore n,m)
```

Il computer sta leggendo il disco sorgente da copiare, segnalando coi numeri 'n' ed 'm' a quale punto sta leggendo. Vi accorgerete che ad un certo momento il drive si fermerà e sul monitor apparirà

```
11 PRESS "ENTER" WHEN DEST.         (premi "ENTER" quando il disco
   DISKETTE MOUNTED ON DRIVE 0      destinatario è montato nel drive 0)
```

A questo punto **togliete** il disco DOS-BASIC dal drive ed **inserite** il disco formattato su cui volete fare il duplicato; poi battete il tasto RETURN.

Dopo aver eseguito anche questa operazione, il drive riparte e sul video appare la scritta

```
12 WRITING TRACK, SECTOR n,m        (cioè scrivo le tracce e i settori n e m)
```

che dopo poco cambierà in

```
VERIFYING TRACK, SECTOR n,m         (verifico le tracce e i settori n e m)
```

Quindi, analogamente a quanto visto per la formattazione, il sistema scrive e controlla che i dati registrati siano uguali a quelli immessi.

Quando questo processo è finito riappare la scritta riportata in (9); a questo punto occorre rimettere il disco sorgente DOS-BASIC nel drive e rifare tutte le operazioni viste man mano che le domande si ripetono **e questo per un certo numero di volte consecutive dipendente dalla quantità di memoria a disposizione.**

In pratica avviene che la grande massa di dati da copiare quando si dispone di un solo drive viene divisa e caricata a 4 blocchi separati, quindi ciclicamente dovremo ripetere le istruzioni (9), (10), (11) e (12).

Se non ci sono stati errori, tutto termina con la scritta

```
13 FULL DISKETTE COPY DONE          (la copia di tutto il disco è finita)
   PRESS "ENTER" WHEN SYSTEM        (premi "ENTER" quando il disco
   DISKETTE MOUNTED ON DRIVE 0      sistema è nel drive 0)
```

ossia rimettete il disco sistema DOS-BASIC nel drive 0, premete il tasto RETURN per tornare al livello DOS. Per assicurarvi che la copia duplicata sia perfetta potrete inserirla nel drive e utilizzare questa al posto dell'originale.

Quindi premendo il tasto RESET presente sulla scheda CPU e provando a ricaricare il BASIC tutto dovrebbe filare liscio.

Gli unici inconvenienti per non riuscire a portare a termine una duplicazione possono essere i seguenti: avete inserito nel drive un dischetto provvisto di protezione contro le registrazioni, oppure avete lasciato nel drive il disco DOS invece di sostituirlo con quello formattato: in questo caso il sistema prova a scrivere, ma non riuscendoci ve lo segnala

```
14 PRESS "ENTER" WHEN **SYSTEM**    (premere "ENTER" quando il disco sistema
   DISKETTE MOUNTED ON DRIVE 0      è nel drive 0)
```

Dopo aver rimesso il disco DOS-BASIC nel drive, si leggerà

```
ILLEGAL FILE NAME                   (manovra illecita)
DOS READY
```

In questo caso, eliminate la causa d'errore, ricominciate da capo e tutto girerà come precedentemente descritto.

Esiste comunque un'ultima causa che può provocare un'interruzione del processo di duplicazione: ricordate il caso in cui appariva la scritta di una traccia difettosa nella formattazione? (vedi paragrafo formattazione dei dischi vergini operazione 7). Bene; allora rammenterete che tale disco aveva la traccia 18 del settore 04 difettosa. Se quindi provate a fare una duplicazione usando come destinatario proprio quel disco difettoso, il sistema ad un certo punto comunica

```
PRESS "ENTER" WHEN **SYSTEM**       (premere "ENTER" quando il disco sistema
DISKETTE MOUNTED ON DRIVE 0         è nel drive 0)
```

dopo aver soddisfatto alla richiesta, si legge

```
PARITY ERROR DURING READ            (errore di parità durante la lettura)
```

Il sistema rifiuta, come è giusto che sia, di fare un duplicato su di un dischetto che abbia qualche traccia fuori uso. Facciamo presente che un disco con tracce difettose lo potremo utilizzare ugualmente per qualsiasi altro uso che non sia una duplicazione.

SE AVETE DUE DRIVE

Se avete due drive il procedimento è più semplice e veloce. Infatti basta mettere il disco DOS-BASIC nel drive 0 e quello destinatario della copia nel drive 1; e scrivere semplicemente

```
COPY :0 TO :1 01/25/82
```

Nell'esempio noi vi abbiamo messo una data 01/25/82 ma è ovvio che voi potete inserire la data che preferite; importante è non dimenticare lo spazio da: COPY (spazio) : 0 (spazio) TO (spazio) : 1 (spazio) data.

Il computer vi farà la domanda (9), poi la (11) presentate precedentemente (dove ovviamente al posto dello zero scriverete il numero 1); a questo punto è sufficiente premere due volte consecutivamente il tasto RETURN per dare inizio alla copiatura. IN QUESTO CASO ESSA AVVIENE IN UN'UNICA PASSATA.

Dopo circa sei minuti la copia è pronta.

Ovviamente è possibile anche effettuare delle duplicazioni di dischi che non siano quello DOS-BASIC. In questo caso bisogna prima portarsi al livello DOS, poi introdurre il comando COPY con le stesse modalità alla domanda (9).

Togliere dal drive 0 il disco sistema e mettere quello da duplicare e procedere normalmente come già spiegato. Naturalmente a duplicazione avvenuta il disco DOS-BASIC va rimesso nel drive 0.

Una curiosità: operando con un solo drive è possibile fare la duplicazione di un disco sullo stesso disco (a patto che non abbia il nastro adesivo di protezione). La cosa ovviamente non ha molto senso, però serve a farvi capire la «potenza» di questo sistema operativo.

Si possono anche copiare solo programmi o archivi, senza dover fare obbligatoriamente la duplicazione di tutto un disco. Per queste procedure vi rimandiamo al paragrafo COPY dei comandi DOS.

NOZIONI PRELIMINARI SUL BASIC

Il dischetto da noi fornito contiene il DOS e il BASIC V1.0. Dopo aver effettuato il caricamento come è già stato spiegato in precedenza, il sistema può essere usato in 5 modi diversi:

1 - Comandi diretti
2 - Esecuzione di un programma
3 - Correzione di un programma
4 - Livello monitor
5 - Livello DOS

Vediamoli brevemente uno per uno.

ESECUZIONE DI COMANDI DIRETTI

A questo livello il computer esegue tutti i comandi introdotti da tastiera, dopo la pressione del tasto RETURN.

È possibile operare in COMANDI DIRETTI, solo quando sul monitor compare la scritta

```
READY
>_
```

ESECUZIONE DI UN PROGRAMMA

Questo livello è reso operativo dal comando RUN: il programma presente nella memoria del computer diventa esecutivo.

CORREZIONE DI UN PROGRAMMA

Operando a questo livello è possibile fare l'EDITING, ossia modificare un programma preesistente.

LIVELLO MONITOR

In questo modo è possibile caricare in memoria dei programmi in linguaggio macchina (OBJECT FILES).

LIVELLO DOS

Questo è il livello del SISTEMA OPERATIVO, che rende possibile l'uso dei Floppy Disks. Come già detto, questo livello può essere raggiunto in via diretta (uscendo dal BASIC) o attraverso il BASIC. Vedremo l'importanza di tutto questo.

CONCETTI FONDAMENTALI

Prima di proseguire occorre che alcuni concetti basilari risultino ben chiari. Le righe che seguono sono di estrema importanza per chi non conosce i metodi di programmazione in BASIC, quindi vanno seguite con molta attenzione, esempi compresi. Per accertarvi di aver ben capito, provate a fare degli esempi vostri sullo schema di quelli qui forniti.

NOTA BENE - In tutti gli esempi che seguiranno si presuppone quanto segue:

— BASIC già caricato

— Cancellazione di programmi eventualmente già esistenti (scrivere NEW e pigiare il tasto RETURN).

PAROLE CHIAVE

Le PAROLE CHIAVE sono i comandi che il computer è in grado di riconoscere. L'elenco completo delle PAROLE CHIAVE è nelle TABELLE 2-3-4-5-6-7 riportate in prima pagina.

Usando nelle istruzioni parole diverse da quelle chiave il computer segnala errore.

VARIABILI NUMERICHE

I nomi delle variabili devono cominciare con una lettera (da A a Z); possono essere formate da un solo carattere (che quindi deve essere obbligatoriamente una lettera) o da due o più caratteri. Nel caso che si usino variabili con più di un carattere, ricordatevi che i caratteri dal secondo in poi possono essere anche numeri. Ricordatevi pure che il computer accetta definizioni di variabili con più di due caratteri, ma tiene conto solo dei primi due.

Ecco qualche esempio di variabili numeriche ammesse:

```
1 C=32
2 A=C*28-3.4
3 S3=K^4/2
4 XQ=3.1415
5 RO=7+(A*2.3-C)^2/5
6 ROMANO=7
7 ANNO=1982
8 ROVIGO=4.78
9 S3712=8
```

I simboli che non sono chiari saranno spiegati tra breve. Attenzione a non confondere il punto con la virgola: nella notazione americana non si scrive 3,4 bensì 3.4.

Per controllare gli esempi o per provare a farne di diversi basta fare come segue:

scrivere NEW e il tasto RETURN, poi scrivere un esempio e nuovamente RETURN; successivamente digitare PRINT e il nome assegnato alla variabile, e poi RETURN.

Immediatamente il computer stamperà il valore di quella variabile.

Provare quindi a scrivere l'esempio 1 facendolo seguire da RETURN; scrivete poi PRINTC (sempre seguito da RETURN). Sul monitor compare il numero 32, come era stato assegnato.

Anziché digitare PRINT si può ottenere lo stesso risultato pigiando in sua vece il punto interrogativo.

Notare che, come detto, le variabili degli esempi 5-6-8 in effetti per il computer hanno lo stesso nome 'RO' in quanto i caratteri oltre il secondo vengono ignorati. Le stesse considerazioni valgono per gli esempi 3 e 9.

Ecco ora qualche esempio di definizioni di variabili numeriche inefficaci o non ammesse:

```
10 3D=63
11 8=2*Z
12 L.=3/4
13 IF=2
14 STORIA=392
```

Se provate a controllare gli esempi 10 e 11 vedrete che in effetti quando date il comando PRINT3D oppure PRINT8 il computer vi darà rispettivamente i valori 3 e 8. È evidente quindi che non avete definito delle variabili.

Gli esempi 13 e 14 sono illeciti perché contengono istruzioni BASIC (IF la 13, TO e OR la 14).

In questo caso il monitor segnala

```
SYNTAX ERROR IN 15359          (errore di sintassi in 15359)
READY
UNDEFINED LINE IN 15359        (linea non specificata in 15359)
```

La linea 15359 la vedrete stampata spesso, quando commettete degli errori. Il numero fa riferimento al BASIC scritto in linguaggio macchina, quindi non ha nessuna relazione con ciò che state facendo.

Le variabili numeriche possono essere di tre tipi: INTERE - A SINGOLA PRECISIONE - A DOPPIA PRECISIONE.

Quelle INTERE sono seguite dal simbolo '%', quelle a SINGOLA PRECISIONE dal simbolo '!', quelle a DOPPIA PRECISIONE dal simbolo '#'.

La differenza tra variabili a SINGOLA e DOPPIA precisione consiste nel fatto che le prime sono formate da 6 cifre significative e le seconde da 16 cifre significative.

Qualche esempio:

```
15 W%=96
16 D416!=.123456
17 Q#=3.141592653589
18 P=16
19 P%=1729
20 P!=34.72
21 P#=23.45289342
```

Negli esempi dal 18 al 21 siamo in presenza di 3 variabili ben distinte, trattate come tali dal computer nonostante inizino tutte con la stessa lettera (P e P! definiscono la stessa variabile).

VARIABILI ALFANUMERICHE

Per VARIABILI ALFANUMERICHE (dette più spesso STRINGHE) si intende un insieme di caratteri qualsiasi: lettere, numeri e simboli grafici in genere.

Si definiscono con gli stessi criteri dati per le variabili numeriche, tranne il fatto che sono contraddistinte dal simbolo '$'. Ogni variabile alfanumerica può contenere al massimo 255 caratteri.

Qualche esempio:

```
22 A$="PERA"
23 CN$="CONSEGNA A DOMICILIO SENZA SPESE"
24 SOMMA$="IL RISULTATO FINALE E':"
25 F1$="7*A+C"
```

Come si vede, dopo il segno '=' bisogna mettere le virgolette, altrimenti il computer segnala errore.

OPERATORI ARITMETICI

Gli OPERATORI ARITMETICI servono per eseguire i calcoli matematici. I simboli usati sono i seguenti:

```
+   SOMMA
-   SOTTRAZIONE
*   MOLTIPLICAZIONE
/   DIVISIONE
^   ELEVAMENTO A POTENZA
```

OPERATORI RELAZIONALI O DI CONFRONTO

Gli OPERATORI RELAZIONALI servono quando in un programma si devono prendere decisioni in base a dei confronti. Essi sono:

```
<    (MINORE DI)
>    (MAGGIORE DI)
<>   (DIVERSO DA)
<=   (MINORE O UGUALE A)
>=   (MAGGIORE O UGUALE A)
=    (UGUALE A)
```

Esempio:

```
IF A>56 PRINT "A MAGGIORE DI 56"
```

In questo caso, se la condizione A>56 è verificata, il computer stampa la frase tra virgolette; in caso contrario passa alla riga di istruzione successiva.

OPERATORI LOGICI

Gli OPERATORI LOGICI accettati dal computer sono: **OR, AND, NOT.**

Assieme alle parole chiave IF, THEN, ELSE (che vedremo più avanti) permettono di fare confronti logici anche molto complessi e potenti.

OPERATORI DI STRINGA

Gli OPERATORI DI STRINGA sono:

```
<    (MINORE DI)
>    (MAGGIORE DI)
<>   (DIVERSO DA)
<=   (MINORE O UGUALE A)
>=   (MAGGIORE O UGUALE A)
=    (UGUALE A)
+    (SOMMA)
```

I primi sei servono per ordinare alfabeticamente le stringhe, mentre l'ultimo serve per sommarle. Vedremo esempi esplicativi più avanti.

COMANDI DIRETTI E ISTRUZIONI

Prima di proseguire occorre capire bene qual'è la differenza tra i due concetti di COMANDI DIRETTI ed ESECUZIONE DI UN PROGRAMMA.

A questi due modi operativi si collegano le definizioni di COMANDO e di ISTRUZIONE:
un COMANDO è impartito direttamente dalla tastiera, e diventa esecutivo appena si preme il tasto RETURN;
una ISTRUZIONE è una parola chiave inserita in un programma, e che agisce solo in sede di esecuzione del programma stesso.

La distinzione tra COMANDI e ISTRUZIONI è generalmente nettissima nel senso che provando ad utilizzare una parola di istruzione come se fosse un comando, il computer segnala errore (e viceversa). In effetti, però, molte istruzioni sono accettate e diventano operative anche se usate come comandi, e analogamente alcuni comandi sono operativi anche inseriti in una linea di programmazione. Affronteremo comunque questo aspetto caso per caso, in sede di spiegazione più dettagliata delle parole chiave.

Basti un solo esempio di quanto abbiamo appena detto: la parola chiave PRINT è sia COMANDO che ISTRUZIONE, vale a dire che diventa esecutiva sia introducendola direttamente da tastiera che inserendola all'interno di un programma.

Proprio per puntualizzare la differenza, almeno concettuale, tra COMANDO ed ISTRUZIONE, abbiamo adottato il seguente metodo: negli esempi che accompagnano le descrizioni delle parole chiave il simbolo > sta ad indicare che si tratta di un COMANDO, mentre se ci sono numeri di linea (ovviamente solo esemplificativi) vuol dire che la parola in questione rappresenta una ISTRUZIONE.
Esempio:

```
> CONT    (comando)
60 NEXT   (istruzione)
```

IL CONTENUTO DI EVENTUALI PARENTESI VICINO ALLE PAROLE CHIAVE RAPPRESENTA IL MODO IN CUI E POSSIBILE ABBREVIARE LE PAROLE STESSE.

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 IF I=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	64	—	@	192	96	—	spazio
33	—	!	65	—	A	193	97	—	a
34	—	”	66	—	B	194	98	—	b
35	—	#	67	—	C	195	99	—	c
36	—	$	68	—	D	196	100	—	d
37	—	%	69	—	E	197	101	—	e
38	—	&	70	—	F	198	102	—	f
39	—	’	71	—	G	199	103	—	g
40	—	(	72	—	H	200	104	—	h
41	—	)	73	—	I	201	105	—	i
42	—	*	74	—	J	202	106	—	j
43	—	+	75	—	K	203	107	—	k
44	—	,	76	—	L	204	108	—	l
45	—	—	77	—	M	205	109	—	m
46	—	.	78	—	N	206	110	—	n
47	—	/	79	—	O	207	111	—	o
48	—	0	80	—	P	208	112	—	p
49	—	1	81	—	Q	209	113	—	q
50	—	2	82	—	R	210	114	—	r
51	—	3	83	—	S	211	115	—	s
52	—	4	84	—	T	212	116	—	t
53	—	5	85	—	U	213	117	—	u
54	—	6	86	—	V	214	118	—	v
55	—	7	87	—	W	215	119	—	w
56	—	8	88	—	X	216	120	—	x
57	—	9	89	—	Y	217	121	—	y
58	—	:	90	—	Z	218	122	—	z
59	—	;	91	—	[	219	123	—	[
60	—	<	92	—	\	220	124	—	\
61	—	=	93	—	]	221	125	—	[
62	—	>	94	—	↑	222	126	—	↑
63	—	?	95	—	←	223	127	—	←

Fig. 1

NOTA: i codici ASCII compresi tra 128 e 191 sono caratteri grafici (vedi pagina di destra), quelli da 192 a 223 riportati in colore coincidono con gli stessi alfabetici ma stampati in REVERSE, cioè in negativo.

CODICE ASCII DEI CARATTERI GRAFICI SU VIDEO

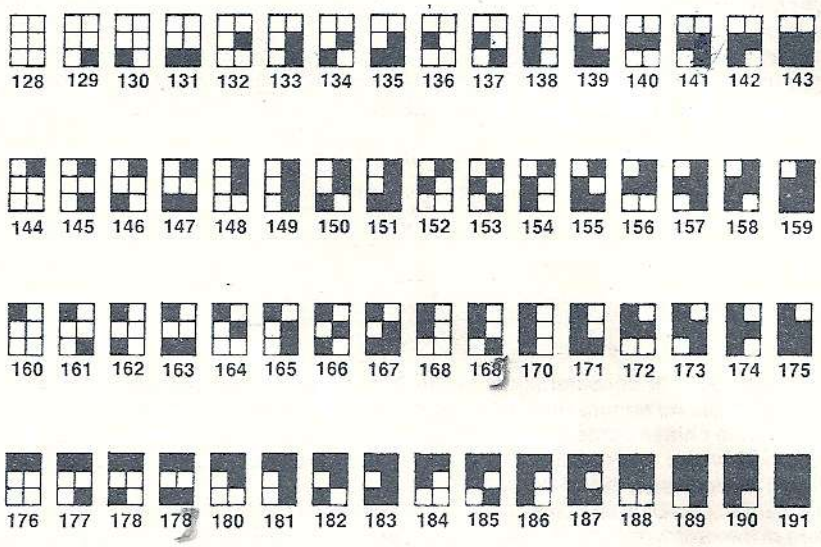


Fig. 2

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 'O' rappresenta un carattere; all'inizio e alla fine di ogni riga c'é il numero del carattere d'estremità.

```
0   OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 31
32  OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 63
64  OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 95
96  OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 127
128 OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 159
160 OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 191
192 OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 223
224 OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 255
256 OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 287
288 OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 319
320 OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 351
352 OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 383
384 OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 415
416 OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 447
448 OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 479
480 OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO 511
```

Fig. 3

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

COMANDI DOS (TABELLA n. 1)

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.

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 nel paragrafo precedente avete imparato l'uso corretto di questo comando, dato che 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 trovandovi a livello BASIC, potete farlo senza essere costretti a passare a livello DOS scrivendo semplicemente

```
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 questa istruzione 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 avete Inserito il disco DOS-BASIC, si mette in moto; il led acceso segnala che sta avvenendo uno scambio di datl tra il disco e la memoria del computer.

Anche questo intervallo di tempo risulta essere un po' più lungo di quello che si sarebbe verificato 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 verrà spiegato anche alle voci CMD"funzione DOS" e CMD"S".

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 preferisce, costruendo così un sistema che risulta incompatibile per 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 (vedi pag. 16 e 17): in quella di sinistra si hanno i numeri progressivi da 32 a 223, 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... 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 attuati rispecchiano i codici ASCII dal numero 32 al numero 122, 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 a pag. 16 e 17. 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$ descritta nel sunto del BASIC.

Dopo questa lunga parentesi, necessaria per chi non conosce ancora 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".

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 a pag. 17.

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 (vedi relativo paragrafo BASIC pag. 47); 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 E' 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 E' 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 è stato scritto, cioè: CMD "APPEND (spazio) PROG 1 (spazio) TO (spazio) PROG 2", diversamente il computer segnala errore.

Infatti se non osserverete gli spazi, il floppy parte e dopo un po' appare la scritta:

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

che segnala appunto un errore di scrittura nel comando impartito.

Quello appena 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 E' IL PRIMO PROGRAMMA
QUESTO E' 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 E' IL PRIMO PROGRAMMA"
50 REM - PROGRAMMA 2 -
60 PRINT
70 PRINT "QUESTO E' IL SECONDO PROGRAMMA"
80 PRINT
```

Avrete notato che abbiamo sempre parlato di attaccare il programma PROG1 in coda al PROG2; invece nel listato vedrete 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 cui 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 E' IL PRIMO PROGRAMMA
```

Se poi fate RUN" PROG2 vedrete la scritta:

```
QUESTO E' 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 che vedremo in seguito comunque speriamo abbiate 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 avrete dovuto vedere 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.

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:

```
(1) ATTRIB PROG1 (I)
(2) ATTRIB PROG1:1 (I)
(3) 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 IN 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-
UNPRINTABLE ERROR IN 15359   (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;

```
CMD"AUTO"
```

D'ora in poi non illustreremo più entrambi i casi: 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 già detto. Se non avete commesso errori, il comando viene eseguito. 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 si possono commettere tre errori: o sbagliare 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; dopodichè ve lo segnala con tale 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 avranno 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.

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 il comando DOS messo in AUTO è la parola chiave BASIC, il computer carica prima il DOS e poi il BASIC.

In 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; 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 e vi consigliamo di seguirlo bene: è molto istruttivo.

In pratica con esso diciamo al computer di eseguire il comando AUTO; dopo averlo impartito, fateun 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 cariato il DOS); successivamente appare la scritta DOS READY. Fate ora nuovamente un RESET e ripartite da capo: vedrete che alla terza riga non sta 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.

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

Mettendo uno o più spazi prima e dopo la virgola otterrete una segnalazione d'errore. Potrete scrivere in maiuscolo o in minuscolo indifferentemente: il computer cambia tutto automaticamente in maiuscolo.

BASIC

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

Eccovi una considerazione particolare: se scrivete, dal livello BASIC, CMD"BASIC" passate il 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, nelle pagine precedenti, però in questi esempi 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 sul 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 il precedente paragrafo di COPY nella duplicazione di dischi a pag. 10. 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.

Se provate, 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 TIME$, 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 operate a 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», illustrato nei comandi BASIC a pag. 39.

Per attivare il DEBUG, scrivete 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, ometterete 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 (vedi fig. 4).

2) rappresentazione di una pagina video comprendente solo memoria per un totale di 128 bytes. (vedi figura 5)

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

Guardando la Fig. 4, cerchiamo di interpretare correttamente quello rappresentato:

— nella 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 contenuto (in ESADECIMALE) della coppia di registri visualizzata. (ad esempio HL = 51 C3 significa che il registro H contiene il numero esadecimale 51 e il registro L contiene il numero esadecimale C3).

— 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 =51C3> E9 D8 C6 06 38 03 C6
```

significa che HL contengono rispettivamente 51 e C3 e che questi due registri insieme formano un puntatore in memoria alla locazione 51 C3 dove da quell'indirizzo in poi sono contenuti i bytes E9 D8 C6 06 38 03 C6 cioè:

51C3	contiene	E9
51C4	contiene	D8
51C5	contiene	C6
51C6	contiene	06
51C7	contiene	38
51C8	contiene	03
51C9	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 0D, 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 dei caratteri visualizzati. Tutti i caratteri ASCII inferiori a 20H (Esadecimale) vengono rappresentati con un punto (.). Vedi figure 6 e 7.

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. 4 e 5).
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. Per uscire dal Debug e tornare al Basic, se non è stato modificato il PC, è sufficiente scrivere G seguito dallo spazio.

Gaaaa,bbbb

Seguito da RETURN esegue un programma dalla locazione aaaa alla locazione bbbb.
Es. G750A,75CF esegue dalla 750A alla 75CF. L'arresto a 75CF avviene per inserimento di un BREAK-POINT automatico a tale 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 M7000 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 5 e 7).

U

È una funzione che non ha una grossa utilità, serve ad aggiornare in modo dinamico il display video.
Se battete U osserverete sul 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 Registri e Memoria (figg. 4 e 6). 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. 4 e 5 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. 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
```

Fig. 4

```
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
```

Fig. 5

```
AF =0D4A-Z--1-N-
BC =090D>  A  N  #  *  G  .  .
DE =401D>  .  X  .  T  .  .  D
HL =51C3>  .  .  .  .  B  .  .
AF'=0044-Z---P--
BC'=4D69>  .  *  §  .  ß  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
```

Fig. 6

```
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  .  !  *
```

Fig. 7

DIR

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

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 0
NE-DOS    -- 01/01/82
FORMAT/CMD IP
MOSTRA
BASIC/CMD IP
COPY/CMD IP
DOS READY
```

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

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.

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 da registrare. 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 files 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 illustrato a pag. 7.

FREE

Riprendete la figura 1 dove è rappresentata 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 soltanto 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 chiavi.

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:0 MOSTRA TO PROVA"
```

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.

TABELLA N. 2 - COMANDI E ISTRUZIONI BASIC

1	AND	27	GET	53	OPEN
2	AUTO	28	GOSUB	54	OR
3	CLEAR	29	GOTO	55	PRINT (?)
4	CLOSE	30	IF...THEN...ELSE	56	PRINT# (?#)
5	CMD"D"	31	INKEY$	57	PINT@ (?@)
6	CMD"funzione DOS"	32	INPUT	58	PRINTTAB (?TAB)
7	CMD"S"	33	INPUT#	59	PRINTUSING (?USING)
8	CONT	34	INSTR	60	PUT
9	DATA	35	KILL	61	READ
10	DEFDBL	36	LET	62	REF
11	DEFFN	37	LINEINPUT	63	REM (')
12	DEFINT	38	LINEINPUT#	64	RENUM
13	DEFSNG	39	LIST (L)	65	RESTORE
14	DEFSTR	40	LLIST	66	RESUME
15	DEFUSR	41	LOAD	67	RETURN
16	DELETE (D)	42	LOC	68	REVOFF
17	DIM	43	LOF	69	REVON
18	EDIT (E)	44	LPRINT	70	RSET
19	END	45	LSET	71	RUN
20	EOF	46	MEM	72	SAVE
21	ERL	47	MERGE	73	STOP
22	ERR	48	NEW	74	SYSTEM
23	ERROR	49	NOT	75	TROFF
24	FIELD	50	ON...GOSUB	76	TRON
25	FOR...NEXT	51	ON...GOTO		
26	FRE	52	ONERRORGOTO		

AND

```
40 IF C=3 AND H2>B GOTO 100
```

Se entrambe le condizioni poste sono verificate, l'esecuzione del programma passa alla linea 100, altrimenti va alla riga successiva alla 40.

AUTO

```
> AUTO
> AUTO,100
> AUTO1250,35
```

Genera numeri di riga che cominciano da 10 e aumentano con passo 10
Genera numeri di riga che cominciano da 0 e aumentano con passo 100
Genera numeri che cominciano da 1250 e aumentano con passo 35

L'Auto viene annullato premendo contemporaneamente i tasti **CTRL-A.**

CLEAR

```
 10 CLEAR
120 CLEAR 2000
```

Azzera tutte le variabili numeriche e di stringa; riserva spazio per 50 caratteri di stringa
Azzera e riserva spazio per 2000 caratteri di stringa

CLOSE

```
 80 CLOSE
240 CLOSE2,7,11
```

Chiude l'accesso a tutti i FILES
Chiude l'accesso solo ai FILES 2,7,11

CMD"D"

```
> CMD"D"
```

Fa eseguire a livello BASIC il programma di DEBUG.

CMD"funzione DOS"

```
> CMD"funzione DOS"
```

Rende possibile eseguire a LIVELLO BASIC tutte le operazioni eseguibili a LIVELLO DOS. Tra le virgolette si può mettere uno qualsiasi dei comandi DOS. Alla fine si ha il ritorno automatico al LIVELLO BASIC.

CMD"S"

```
> CMD"S"
```

Genera il passaggio dal LIVELLO BASIC al LIVELLO DOS, con perdita del BASIC e di eventuali programmi presenti in memoria centrale.

CONT

```
> CONT
```

Permette la prosecuzione del programma dopo una interruzione causata da un BREAK (tastiera) o da uno STOP (programma).

DATA

```
50 DATA 318,8.23,"CAMPER",71
```

Memorizza dei dati all'interno del programma. Questi dati sono accessibili esclusivamente in modo SEQUENZIALE con l'istruzione READ (vedere anche RESTORE).

DEFDBL

```
75 DEFDBL C-H
30 DEFDBL Q,F
```

Dichiara che tutte le variabili che iniziano con le lettere da C ad H saranno trattate e memorizzate in DOPPIA PRECISIONE
Idem per le lettere Q e F

I simboli % e ! annullano questa istruzione.

DEFFN

```
10 DEFFN A(X,Y)=X+3%Y
25 DEFFN C$(A$,I)=RIGHT$(A$,I)
```

Definisce la funzione di A e di X e Y come il risultato di X + 3*Y
Definisce la funzione C$ di A$ e I come una stringa di I caratteri presi a destra della stringa C$

DEFINT

```
20 DEFINT A-H
```

Dichiara che tutte le variabili che cominciano con le lettere da A ad H saranno trattate e memorizzate come INTERI, eccettuato quelle che recano i simboli ! e #

DEFSNG

```
300 DEFSNG E-L,R
```

Dichiara che tutte le variabili che cominciano con le lettere da E a L e con la lettera R saranno trattate e memorizzate in SINGOLA PRECISIONE, escluse quelle coi simboli % e #

DEFSTR

```
10 DEFSTR A,C
```

Le variabili che iniziano con le lettere indicate sono trattate e memorizzate come STRINGHE, ad eccezione di quelle coi simboli % ! e #

DEFUSR

```
120 DEFUSR 2=&H33CD
```

Definisce la ROUTINE espressa in LINGUAGGIO MACCHINA, memorizzata a partire dalla locazione 33CD in esadecimale. Le ROUTINES possibili sono 10 (0-9).

DELETE (D)

```
> DELETE 40
> DELETE 120-200
> D 0-300
> D.
```

Cancella la linea 40
Cancella le linee dalla 120 alla 200
Cancella le linee dall'inizio fino alla 300
Cancella l'ultima linea generata od editata

DIM

```
10 DIM F(23)
```

Definisce le variabili trattate come VETTORI ad una o più DIMENSIONI: F è definito come VETTORE ad una dimensione formato da 24 elementi (0-23)

```
10 DIM A$(12,58),C(6,14,E+4,B%2,78)
```

AS è una MATRICE a 2 dimensioni di 767 elementi, C è una MATRICE a 5 dimensioni alcune delle quali sono definite da variabili. Senza una DIM, un vettore può essere al massimo di 11 elementi.

EDIT (E)

```
> EDIT 340
> E30
> E.
```

Fa passare al LIVELLO DI EDITING (correzione) della linea 340
Fa passare al LIVELLO DI EDITING (correzione) della linea 30
Fa passare al LIVELLO DI EDITING (correzione) dell'ultima linea listata o editata (vedi anche pag. 64)

END

```
1200 END
```

Pone termine all'esecuzione del programma. Non è indispensabile.

EOF

```
50 IF EOF(1) THEN CLOSE 1
```

In lettura di un FILE DATI (BUFFER 1) controlla se si è giunti alla fine del FILE; in tal caso esegue la CLOSE, altrimenti prosegue alla linea successiva.

ERL

```
90 PRINT ERL
```

ERL è una variabile speciale riservata al sistema; assume il valore del numero di linea dove si è verificato un errore. Generalmente si usa in coppia con l'istruzione ONERRORGOTO.

ERR

```
260 PRINT ERR/2+1
```

Anche ERR è una variabile speciale riservata al sistema. Il suo valore è legato al codice dell'errore che si è verificato; la formuletta dell'esempio da' appunto il valore del codice. Normalmente si usa accoppiato con ONERRORGOTO, per ROUTINES di trattamento degli errori.

ERROR

```
60 ERROR 1
```

Alla linea 60 il computer si comporta come se si fosse veramente verificato l'errore di tipo 1. Serve per fare dei test sulla routine ONERRORGOTO.

FIELD

```
720 FIELD 3, 15 AS D$, 10 AS CC$,
              112 AS A$
```

Suddivide in CAMPI il BUFFER di un FILE RANDOM. Nell'esempio il BUFFER 3 (di 255 caratteri) è stato diviso in CAMPI in questo modo: 15 bytes per D$, 10 per CC$, 112 per A$; gli ultimi 118 bytes non sono stati definiti.

FOR - - - NEXT

```
200 FOR I=1 TO 20 STEP 4
210 PRINT A+1
220 NEXT I
 30 FOR E2=A TO B*3+C
 50 LPRINT "BOLOGNA"
 60 NEXT
```

Queste due istruzioni danno origine ad un LOOP tra le due linee in cui compaiono. Nell'es. tutto ciò che si trova tra le linee 200 e 250 viene eseguito 5 volte. Il valore iniziale, quello finale ed il passo possono essere espressi con variabili o espressioni (Esempio compreso tra le linee 30 e 60)

FRE

```
> PRINT FRE(0)
```

È equivalente al comando MEM: dà le cellule di memoria ancora disponibili. Il numero tra parentesi può essere qualsiasi.

GET

```
520 GET 7,4
```

Serve per leggere da disco il RECORD 4 del FILE RANDOM 7. Se vengono omessi virgola e secondo numero, si ha la lettura del RECORD successivo all'ultimo letto (del primo, se non è stato letto ancora nulla).

GOSUB

```
90 GOSUB 8000
```

Causa l'interruzione del programma principale per saltare alla SUBROUTINE che inizia alla linea 8000 (vedere anche RETURN).

GOTO

```
15 GOTO 210
```

Il programma dalla linea 15 passa alla linea 210 (SALTO INCONDIZIONATO).

IF - - - THEN - - - ELSE

```
30 IF A>34 THEN CE=5 ELSE GOTO 100
30 IFA>34CE=5ELSE100
70 IF Z THEN 340
70 IF Z GOTO 340
```

IF permette di fare un TEST su un'espressione logica o relazionale. Se il risultato del test è vero, il controllo del programma prosegue nella linea; in caso contrario il programma passa all'istruzione ELSE (se esiste) o alla riga di programma successiva. Gli esempi col numero di linea 30 sono equivalenti, come pure quelli numerati 70.

INKEY

```
40 A$=INKEY$ : IF A$="" GOTO 40
```

L'istruzione INKEY$ predispone il computer ad accettare una stringa di un solo carattere (A$ nell'es.) da tastiera. Se durante l'esecuzione dell'istruzione non si preme alcun tasto, il computer assegna automaticamente ad A$ una stringa nulla (" "), altrimenti assegna il carattere digitato. Nell'esempio riportato il programma si arresta alla linea 40 fino a che non si preme un tasto qualsiasi.

INPUT

```
50 INPUT C,D$
```

Il computer si ferma alla linea 50 e, dopo aver visualizzato un punto interrogativo, resta in attesa che venga introdotto da tastiera un numero che verrà assegnato alla variabile C; subito dopo il computer visualizza ancora? e resta in attesa di una stringa da assegnare alla variabile D i due numeri possono essere introdotti anche contemporaneamente separandoli con una virgola.

```
30 INPUT "NOME DEL CLIENTE = ";NC$
```

Ha un effetto analogo al precedente, con la differenza che il computer, prima del punto interrogativo, visualizza la frase tra virgolette.

INPUT#

```
230 INPUT#12, A, E$
```

Nel trattamento di FILES SEQUENZIALI abilita il computer a leggere dati. Nell'esempio, dal FILE gestito dal BUFFER 12 il computer introita prima un dato numerico che assegna alla variabile A, poi una stringa che va alla variabile E.

INSTR

```
60 P=INSTR(7,A$,"MANO")
```

È una funzione di ricerca di una stringa all'interno di un'altra. Nell'esempio il computer va a vedere se ad iniziare dal settimo carattere della stringa A$ c'è la stringa MANO. In caso affermativo la variabile P assume il valore 7, altrimenti il valore 0. Se non si mette il carattere d'inizio, il computer parte sempre dal primo.

KILL

```
> KILL "CLIENTI"
```

Serve per cancellare dal disco il FILE che porta il nome specificato (CLIENTI).

LET

```
20 LET D=56
20 D=56
```

Serve ad assegnare i valori alle variabili.
Si può scrivere anche omettendo LET.

LINEINPUT

```
85 LINEINPUT "TITOLO DEL FILM";TF$
```

Simile all'istruzione INPUT, con la differenza che manca la visualizzazione del punto interrogativo ed accetta anche le virgole (le virgolette sono accettate da entrambe).

LINEINPUT#

```
55 LINEINPUT#9,A$
```

Serve per leggere una riga di testo da disco. Es.: dal FILE del BUFFER 9 legge la prima riga di dati e la pone nella variabile AS.

LIST (L)

```
> LIST
> LIST 30
> L 300-400
> L120-
> L.
```

Visualizza tutto il programma
Visualizza solo la linea 30
Visualizza tutte le linee tra la 300 e la 400
Visualizza tutte le linee dalla 120 alla fine
Visualizza l'ultima linea generata od editata (anche battendo solo il punto si ottiene lo stesso risultato)

LLIST

```
> LLIST
```

Fa il listato del programma sulla stampante. Vale la casistica di LIST. (Non sono ammesse però le abbreviazioni)

LOAD

```
> LOAD "PAOLO"
> LOAD "PAOLO",R
```

Carica da disco il programma il cui nome è quello specificato
Dopo il caricamento, si ha anche l'esecuzione automatica del programma caricato (vedi anche pag. 48).

LOC

```
40 PRINT LOC(3)
```

Nel trattamento dei FILES RANDOM serve per visualizzare il numero di RECORD che è stato letto per l'ultima volta con l'istruzione GET. Da' quindi la POSIZIONE ATTUALE del PUNTATORE

LOF

```
50 FOR I=1 TO LOF(3) : INPUT#3,A$ : NEXT
```

Serve ad individuare la fine di un FILE DATI; LOF assume il valore del numero d'ordine dell'ultimo RECORD del FILE. Nell'esempio, dopo aver aperto il FILE DATI 3, è possibile con il LOOP indicato leggere tutti i dati del FILE anche senza conoscerne il numero totale.

LPRINT

```
30 LPRINT G
```

Serve per fare stampare dati alla stampante (nell'es. si stampa il valore di G). Vale la casistica prevista in PRINT.

LSET

```
430 LSET F$="NUOVA ELETTRONICA"
```

Dopo una assegnazione di FIELD, serve per mettere la stringa indicata nel CAMPO assegnatole, allineandola a SINISTRA.

MEM

```
> PRINT MEM
120 IF MEM<1000 END
```

Nella variabile speciale MEM viene depositato automaticamente il numero di bytes di memoria ancora liberi. Gli esempi mostrano che è possibile un suo uso sia come comando che nelle istruzioni.

MERGE

```
> MERGE "CLIENTI"
```

Serve per fondere il programma in memoria centrale col programma indicato su disco.

NEW

```
> NEW
```

Cancella tutte le righe di programma ed azzera tutte le variabili.

NOT

```
 30 IF NOT A GOTO 100
470 IF NOT D>K+3*F THEN 500
```

Se A vale zero il programma va alla linea 100
Se D non è maggiore dell'espressione assegnata (minore o uguale), il programma prosegue alla linea 500.

ON - - - GOSUB

```
45 ON B GOSUB 7000,8000,9000
```

Nel caso di B = 1 l'esecuzione del programma salta alla SUBROUTINE che inizia col numero di linea 7000; se B = 2 va alla SUBROUTINE 8000; se B = 3 va alla 9000. Al posto di B può anche esserci un'espressione. Se il valore di B non è compreso tra 1 e 3 il programma va alla linea successiva. **Se B O il computer segnala errore.**

ON - - - GOTO

```
100 ON A1*2.5+S-3 GOTO 120,200,300,450
```

Funzionamento analogo all'istruzione precedente, tranne il fatto che le linee di salto non corrispondono a SUBROUTINES.

ONERRORGOTO

```
110 ONERRORGOTO 1000
```

Se si verifica un errore quando il programma è oltre la linea 110, si va alla linea 1000 (vedere anche RESUME).

OPEN

```
100 OPEN"E",2,"LISTA"
```

Con questa istruzione si aprono FILES DATI sia SEQUENZIALI (SEQUENTIAL) che CASUALI (RANDOM). La lettera tra virgole imposta il metodo di accesso (I-O-E-R); il numero assegna il BUFFER al FILE specificato tra le virgolette successive.

OR

```
40 IF C=3 OR H2>B THEN 100
```

Se una sola delle condizioni poste (o anche entrambe) è verificata, l'esecuzione del programma passa alla linea 100, altrimenti va alla riga successiva alla 40.

PRINT (?)

```
30 PRINT A
70 ? "PARTITA IVA"
50 PRINT
```

Stampa sul video la variabile A
Stampa sul video PARTITA IVA
Sul video il puntatore avanza di una riga

PRINT# (?#)

```
80 PRINT#12,A,E$
```

Nel trattamento di FILES SEQUENZIALI abilita il computer a registrare dati sul FILE gestito dal BUFFER 12 (prima il numero contenuto nella variabile A e poi la stringa E$).

PRINT@ (?@)

```
30 PRINT@330, "NUOVA ELETTRONICA"
```

Ha funzioni analoghe a quelle di PRINT, con la differenza che visualizza quanto specificato a cominciare dalla posizione indicata dal numero prima della virgola. Questo numero deve essere compreso tra 0 (primo carattere in alto a sinistra nel monitor) e 511 (ultimo carattere in basso a destra).

PRINTTAB (?TAB)

```
70 ?TAB(13)"NUOVA ELETTRONICA"
```

Consente di visualizzare quanto richiesto nella posizione di riga specificata dal numero tra parentesi (vedi 13). Questo può assumere i valori da 0 a 255. Serve per incolonnare dati sul monitor. Se il numero supera 31 (lunghezza di una riga), verranno utilizzate le righe successive.

PRINTUSING (?USING)

```
300 PRINTUSING "##.###";U
```

Visualizza il valore della variabile U nel FORMATO specificato dalla stringa tra virgolette (al suo posto si può richiamare una variabile di stringa definita in precedenza).

PUT

```
340 PUT 2,16
```

Scrive su disco il RECORD 16 nel FILE RANDOM 2. Deve essere preceduta dalle dichiarazioni FIELD e RSET (o LSET).

READ

```
20 READ X,D
```

Legge sequenzialmente i dati contenuti nella istruzione DATA e li assegna alle variabili X e D (vedere anche RESTORE).

REF

`> REF*`	È un comando che permette di avere dei RIFERIMENTI. Da' sul monitor l'elenco completo dei RIFERIMENTI (con $ al posto di * l'elenco viene anche stampato)
`> REF DE`	Da' i RIFERIMENTI della variabile DE
`> REF *DE`	Da' i RIFERIMENTI di tutte le variabili, a cominciare da DE
`> REF 2350`	Da' i RIFERIMENTI del numero intero 2350. Serve per trovare i numeri di linea dove compaiono i numeri interi richiesti.

REM (')

```
200 REM --- ROUTINE CALCOLO TRAVE ---
530 'MOVIMENTI MAGAZZINO
```

Serve per inserire REMARKS (COMMENTI) lungo il programma e sono da questo ignorati.

RENUM

`> RENUM,`	RINUMERA il programma partendo dal numero di linea 10 con passo 10
`> RENUM 1000`	Rinumera dal 1000 con passo 10
`> RENUM 1000,100`	Rinumera dal 1000 con passo 100.
`> RENUM 700,10,500,8000`	Rinumera dalla linea 500 fino alla 8000 iniziando con 700 con passo 10
`> RENUM,,,9000`	Rinumera dalla linea 10 con passo 10 iniziando dalla vecchia linea 10 fino alla linea 9000.

Se ci sono errori nei rimandi del programma, non effettua la RINUMERAZIONE e segnala il numero di linea inesistente.

RESTORE

`45 RESTORE`	Permette alla successiva istruzione READ di leggere il primo dato della prima istruzione DATA.

RESUME

`12000 RESUME`	Va inserita nella ROUTINE di gestione degli errori (ONERRORGOTO) e serve ad indicare da quale linea deve riprendere il programma dopo che si è verificato l'errore.

Se RESUME non è seguito da numero, si torna alla linea dell'errore, altrimenti si va alla linea specificata.
Se RESUME è seguito da NEXT, il programma prosegue alla linea successiva a quella d'errore.

RETURN

`730 RETURN`	Chiude l'esecuzione di una SUBROUTINE e fa tornare il programma all'istruzione successiva alla relativa GOSUB.

REVOFF

`100 REVOFF`	Annulla la visualizzazione in REVERSE (NEGATIVO) (vedere anche REVON).

REVON

`60 REVON`	Inserisce la scrittura su monitor in REVERSE (NEGATIVO) (vedere anche REVOFF).

RSET

```
170 RSET D3$="CLIENTI IN MORA"
```

Analoga alla LSET: serve per mettere la stringa indicata nel FIELD assegnatole, allineandola a DESTRA.

RUN

```
> RUN
```

Da' inizio all'esecuzione del programma cominciando dal numero di linea più basso; se dopo RUN si mette un numero, il programma parte dalla linea indicata

```
3450 RUN"MOSTRA"
```

Con questa istruzione si ottiene il CONCATENAMENTO del programma in corso con quello richiamato dal disco.

SAVE

```
> SAVE "MOSTRA" ,A
```

Salva il programma presente in memoria su disco, assegandogli il nome specificato. Dando anche la parte dopo le virgolette, si ha la registrazione in FORMATO ASCII, altrimenti essa avviene in FORMATO COMPATTATO.

STOP

```
250 STOP
```

Interrompe l'esecuzione del programma e fa tornare a livello di COMANDI DIRETTI (vedere anche CONT).

SYSTEM

```
> SYSTEM
```

Porta il computer a LIVELLO MONITOR, permettendo così l'utilizzo di SOUBRUTINES a linguaggio macchina.

TROFF

```
> TROFF
```

Elimina la funzione TRACE innescata dal comando TRON.

TRON

```
> TRON
```

Dà inizio alla funzione TRACE. Dopo averlo introdotto, pigiare RUN: il programma parte e sul monitor appare il FLUSSO completo del programma, col numero di riga via via eseguito visualizzato tra parentesi. Serve per analizzare l'esecuzione del programma.

DESCRIZIONE DELLE ISTRUZIONI BASIC

Come avete visto, i comandi e le funzioni BASIC da illustrare dettagliatamente sono in numero considerevole.

Prima di proseguire, ci preme ora fare qualche raccomandazione importante e illustrare qualche comando che vi permette di iniziare a lavorare con il BASIC.

Innanzitutto una considerazione di carattere generale: nell'uso del BASIC gli spazi non sono necessari. Facciamo un esempio:

```
PRINT 23 * 102
```

Questo è un comando diretto che serve per vedere sul monitor il risultato della moltiplicazione di 23 per 102. Ebbene, se scrivete `PRINT23*102` raggiungete lo stesso scopo.

Quanto detto vale anche all'interno delle righe di programmazione: per poter leggere più agevolmente un programma dopo averlo listato sul monitor o sulla stampante vi conviene staccare le parole le une dalle altre; ricordate però che in questo modo occupate uno spazio maggiore nella memoria RAM del computer. Sappiatevi quindi regolare: se fate un programma molto lungo, eliminando gli spazi guadagnerete in memoria.

Detto questo, cominciano pure con il comando:

MEM

DOPO AVER CARICATO IL BASIC provate a digitare PRINT MEM (potete anche digitare ?MEM, che è lo stesso, se ricordate) e vedrete visualizzato sul monitor un numero che rappresenta le celle di memoria RAM che avete a disposizione per i vostri programmi. Da tenere presente che il DOS-BASIC occupa circa 26K BYTES; il numero che vedete è la differenza tra la memoria RAM totale che avete nelle varie schede (statiche o dinamiche che esse siano) e la memoria occupata dal DOS-BASIC.

SAVE

Dopo aver fatto un programma, se spegnete il computer sapete già che lo perdete irrimediabilmente. Per conservarlo utilizzate questo comando nel seguente modo: dovete dare un nome a quel programma (per esempio "FATTURE") e salvarlo su disco digitando

```
SAVE "FATTURE"
```

Per abbreviare al massimo potete anche digitare SAVE"FATTURE (eliminando cioè lo spazio e le virgolette finali); otterrete lo stesso risultato.

Appena avrete premuto il tasto RETURN il drive partirà e registrerà sul disco il vostro programma. Il LED acceso segnala che sta avvenendo uno scambio di dati tra computer e floppy disk.

NON DATE UN COMANDO DI SAVE SE IL DISCO E PROTETTO CONTRO LE REGISTRAZIONI: potreste perdere il controllo del BASIC e con esso anche il vostro programma!

Normalmente questo non succede, e le rare volte che ci è capitato era dovuto ad un non perfetto funzionamento della memoria.

Se tutto è regolare, provando a fare la manovra che vi abbiamo sconsigliato vedrete la scritta

```
UNPRINTABLE ERROR IN 15359
```

Ad ogni buon conto controllate preventivamente qual'è il comportamento del vostro computer in questo frangente tanto importante, e nel caso fate un attento controllo delle schede di memoria. Nel nostro computer è bastato sostituire un integrato del tipo 4116 in una scheda di memoria dinamica da 32K, ed il difetto è scomparso.

Potete anche fare in questo modo: mettete nel drive un duplicato del DOS-BASIC senza l'adesivo di protezione (così se vi capita di rovinarlo avete sempre l'originale per farvene un'altra copia, come vi abbiamo già consigliato); la registrazione non è mai interdetta e l'inconveniente non può verificarsi.

Ricordate poi un'altra cosa molto importante: SE ESISTE GIÀ SUL DISCO UN PROGRAMMA CON LO STESSO NOME, ESSO VIENE CANCELLATO E SOSTITUITO DAL NUOVO. Fate quindi molta attenzione.

Usate il comando **CMD"DIR"** per sapere i nomi dei programmi e degli archivi presenti su di un dischetto; potrete così evitare l'inconveniente appena menzionato.

IMPORTANTE: Se disponete di un solo DRIVE potete utilizzare i comandi SAVE e LOAD solo su dischi che contengono il DOS-BASIC (copia).

LOAD

Con questo comando potete richiamare dal disco i vostri programmi e successivamente eseguirli col comando RUN. Il formato del comando è il seguente:

```
LOAD "FATTURE"
```

dove il nome tra virgolette richiama l'esempio precedente. Se provate a richiamare un programma con un nome diverso da quello con cui lo avevate salvato, il computer ovviamente non lo trova e vi segnala il fatto con

```
FILE NOT FOUND IN 15359
```

Quindi, almeno fino a che non saprete richiamare l'indice dei nomi assegnati ai programmi, segnatevell in un foglio di carta e conservatelo; vi servirà senz'altro!

Anche in questo comando si possono omettere lo spazio e le virgolette finali.

Un'ultima cosa, che interessa in modo particolare chi è in possesso di più di un drive. Se fate ad esempio In questo modo

```
SAVE "FATTURE:1"
```

il computer va ad incidere il programma sul drive 1. Potete anche digitare `SAVE"FATTURE:1` ed ottenere lo stesso risultato.

Analogamente, per il comando LOAD, aggiungendo i due punti ed un numero (che ovviamente deve essere compreso tra 0 e 3) date istruzione al sistema di mettersi in comunicazione col drive specificato.

Anche in questo caso è prevista una segnalazione di errore se la manovra è sbagliata: infatti provando a chiedere il collegamento con un drive che non c'è, il monitor visualizza

```
TOO MANY FILES IN 15359
```

Non vi resta che controllare il tutto per capirne bene il meccanismo d'uso.

RACCOMANDAZIONI IMPORTANTI

Prima di chiudere, diamo qualche importante raccomandazione.

NON ACCENDETE NÈ SPEGNETE MAI IL COMPUTER COI DISCHETTI INSERITI NEI DRIVE. Potrebbero subire delle alterazioni e non dare più i risultati ottimali. Prendete quindi la buona abitudine di estrarli prima di spegnere e di inserirli dopo aver acceso il computer e premuto il tasto RESET della scheda CPU.

NON FATE MAI UN BREAK QUANDO I DRIVE HANNO I LED ACCESI. Perderete quasi sempre il controllo del sistema, con perdita del BASIC, del DOS e di tutto quello che si trova nella memoria centrale.

SE LA STAMPANTE TERMICA NON GIRA COL COMANDO LPRINT, SCRIVERE DA BASIC

```
OUT 3,255
```

E VEDRETE CHE POI FUNZIONA SUBITO; L'INIZIALIZZAZIONE SUDDETTA VA FATTA OGNIQUALVOLTA SI ACCENDE IL COMPUTER (SOLO PER LA STAMPANTE TERMICA).

Fino a quando la scheda grafica non sarà disponibile, sul monitor vedrete solo lettere maiuscole. **RICORDATE PERÒ CHE TUTTO QUELLO CHE VIENE SCRITTO COME STRINGA (CIOÈ TRA VIRGOLETTE) È VISUALIZZATO IN MAIUSCOLO, MA VIENE MEMORIZZATO MAIUSCOLO O MINUSCOLO A SECONDA DI COME ERA STATO DIGITATO.**

Vale a dire che se avete scritto una stringa senza premere il tasto SHIFT delle maiuscole, la stringa stessa viene visualizzata in maiuscolo sul monitor, ma presentata in minuscolo dalla stampante. Questa considerazione è importante anche quando si opera a livello di EDITING, come avremo modo di chiarire in seguito.

NEL DRIVE O DEVE SEMPRE STARE UN DISCO COL DOS-BASIC, altrimenti diverse funzioni operative BASIC e DOS non possono funzionare. Sarebbe quindi opportuno operare con DUE DRIVE, in modo da poter registrare e prelevare programmi e dati col secondo floppy. Proprio per questo stiamo approntando un contenitore in grado di alloggiare due drive.

Fermiamoci qui, per questa volta. Avete di che sbizzarrirvi in prove e controprove, ma non mancherete di divertirvi. Ora avete in mano uno strumento molto potente e perfezionato, che eseguirà fedelmente e con precisione i vostri programmi. I risultati che ne trarrete dipenderanno, oltre che dalla nostra chiarezza, anche dalla vostra abilità e fantasia.

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 OLREADY 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

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

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 a mano le pagine precedenti e di andarvi a rileggere quelle nozioni.

Ricordate anche che eventuali parentesi accanto alle parole chiave rappresentano 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 MKD$. (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)
```

Pone nella variabile di stringa CR$ la stringa NUOVA.

```
70 PRINT LEFT$(S1$+S2$,C*5-D+2)
```

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 ELETTRONICA",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$ va 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$=MKD$(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.

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 ↑

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=&035
```

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)
```
La variabile I assume il valore della parte intera di H.

```
770 PRINT CINT(12.4);CINT(-356.9801)
```
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)
```
Si ottengono i numeri 28 e —13.

```
100 I=FIX(A)
```
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 segulto da numeri.

```
210 PRINT 3^5
```
Eleva alla quinta potenza il numero 3, cioè scrive il numero 243.

TABELLA N. 5 - FUNZIONI GRAFICHE			
1 CLS	2 POINT	3 RESET	4 SET

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.

TABELLA N. 6 - FUNZIONI SPECIALI			
1 INP 2 OUT	3 PEEK 4 POKE	5 POS 6 TIME$	7 USR 8 VARPTR

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=USR3(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.

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 ↑

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=&0345
```
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 annotazioni 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 DIM K(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 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 sommata a 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.

—

È 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.

```
> ? 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.

/

È 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=6+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.

↑

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

```
400 WS=L^F
> PRINT 5^8
```

Assegna alla variabile WS il risultato di L elevata a F.
Stampa il numero 390625, cioè il valore di 5 elevato all'ottava potenza.

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)

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 occore digitare alcuna parola, ma è sufficiene 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 scrive 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, su alcuni tasti e inciso BS). (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 e per uscire dalla numerazione automatica delle linee di programmazione.

(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 puntl del monitor (16 righe per 32 colonne).

(CTRL + P)

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

(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.

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)	

G - COMANDI DI EDITING (TABELLA n. 9)

Con la descrizione delle voci della TABELLA 9 si chiude il sommario delle parole chiave del BASIC.

Delle TABELLE 1-2-3-4-5-6-7-8 abbiamo dato un veloce sunto, voce per voce; questo per mettere in grado tutti i 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.

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          328 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 seguire 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 userete 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 (II), 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 premessimo 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 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 di programma successiva all'ultima listata o 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 di 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 ai 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 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 ← (BACKSPACE) 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 'virgolete'. 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, aggiun-

gendo 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 editing, 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!

SUNTO DELLA GESTIONE DEI FILES

Dopo aver visto la descrizione di tutte le parole chiave del DOS e del BASIC, non ci resta che darvi le nozioni necessarie alla gestione dei FILES su disco. Prima di addentrarci nell'argomento, cosa che facciamo dettagliatamente nelle pagine seguenti, ci limitiamo ora ad un breve riassunto della procedura da seguire per avere sottomano un quadro succinto ma completo. Ancora a proposito delle istruzioni BASIC e DOS, annunciamo sin da ora che torneremo sull'argomento con esempi applicativi per un opportuno approfondimento del loro uso.

Seguiteci con fiducia e vedrete che in pochi mesi avrete a disposizione 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$(I), dove il numero variabile I serve per individuare i vari nominativi: N$ 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 seguiranno 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 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 se 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",1,"AMICI"
PRINT#1,N$(1),N$(2),N$(3),....
CLOSE
```

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#1,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.

FILES DI TIPO RANDOM

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 CVI sono concettualmente equivalenti alle STR$ e VAL che operano sulle stringhe.

NOTA: la seconda trasformazione è corretta solo se la stringa da trasformare in numero è a sua volta un numero già trasformato in stringa.

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 (FORLI')"
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"
FIELD1,25 AS N1$,40 AS I1$,12 AS T1$,8 AS A1$,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$ nel 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.

LA GESTIONE DEI FILES

Nelle pagine precedenti abbiamo dato qualche breve istruzione sull'uso dei files sequenziali e random. Ora cercheremo di spiegarveli in modo più comprensibile con un esempio pratico.

Quello che vi proponiamo è stato concepito, sempre a fini didattici: non stupitevi perciò se qualcuna delle soluzioni adottate non è la più appropriata per risolvere il problema. L'importante è che possiate vedere varie applicazioni pratiche e corrette dell'uso dei files. Una volta che le avrete capite non vi sarà difficile apportare al programma proposto tutte le modifiche che riterrete opportune; meglio ancora se vi cimenterete in un esercizio tutto vostro, in modo da mettere subito in pratica le nuove nozioni.

Col nostro programma ci proponiamo di gestire i clienti ed i fornitori di una piccola azienda. In pratica vogliamo crearci un archivio dei nostri clienti, dove registreremo i loro nomi, indirizzi e telefoni. Esso conterrà anche la situazione dei pagamenti relativa ad ogni cliente.

Questo file lo faremo di tipo **random**. Ogni nuovo cliente sarà quindi aggiunto in coda ai precedenti.

Trattandosi di un **file random**, potremo andare a modificare i dati relativi ad un cliente senza dover riscrivere tutto l'archivio. In tal modo cambiamenti di indirizzo o di numero telefonico oppure la variazione di cifre relative ai pagamenti saranno molto semplici e riguarderanno esclusivamente il cliente interessato.

Faremo poi un archivio dei fornitori, organizzato in modo perfettamente analogo a quello dei clienti.

Organizzeremo successivamente un archivio sequenziale per registrarvi tutte le vendite effettuate ai vari clienti, coi relativi importi e uno simile a quello appena descritto per gestire tutti i movimenti che riguardano invece i fornitori.

Vi ricordiamo quelle che sono le principali differenze tra files di tipo **random** e **sequenziale**. I files **sequenziali** vengono registrati in formato ASCII, quindi occupano uno spazio maggiore sul disco, a discapito della quantità di informazioni che potremo inserirvi; inoltre la loro incisione o lettura richiede un tempo più lungo rispetto a quelli di tipo random. Il contenuto di un file sequenziale non è possibile modificarlo: per modificare un suo elemento intermedio occorre riscrivere tutto l'archivio. A tale riguardo vi rammentiamo che il BASIC da noi fornito è uno dei pochi in grado di aggiungere dati in coda ad un file sequenziale. Per finire, i files sequenziali devono sempre essere letti a cominciare dall'inizio; se l'archivio contiene, ad esempio, 350 elementi e a noi interessa il contenuto di quello che occupa la posizione numero 307, dovremo leggerci tutti i 306 elementi precedenti prima di poter avere il dato che cerchiamo.

I files di tipo **random** non hanno tali problemi. Infatti possono essere scritti o letti in un punto qualsiasi, e la loro scrittura avviene in formato compattato, con guadagno in tempo e spazio. La contropartita consiste in una complicazione maggiore nella loro gestione. Volta per volta si fa uso quindi del tipo più adatto al problema da risolvere.

Il programma che vi proponiamo non contiene delle grosse raffinatezze, ma non è neppure troppo semplificato; tale soluzione ci sembra la più adatta sia a chi è alle prime armi che ai più esperti. Coloro che appartengono alla prima categoria, seppure con un po' di fatica, trarranno sicuramente molti vantaggi della comprensione del programma. Gli altri potranno apprendere tutte quelle nozioni sul trattamento dei files che sono indispensabili; e in un secondo momento, di cambiare tutto ciò che non li soddisfa, per adattarlo alle loro esigenze.

SCHEMI A BLOCCHI E DIAGRAMMI DI FLUSSO

Prima di proseguire è necessario introdurre il concetto di DIAGRAMMA DI FLUSSO (FLOW CHART in inglese). Infatti da un listato di programma non è facile capire quali sono le funzioni svolte dalle varie parti, anche perché le istruzioni GOTO e GOSUB fanno eseguire dei salti da una parte all'altra.

Se un **diagramma** di flusso risulta utile al programmatore, lo è ancora di più a chi legge un programma scritto da un altro. Risulta infatti estremamente arduo capire da un listato quali erano le intenzioni di chi l'ha fatto; quindi occorre spesso impiegare parecchio tempo per decifrare un programma. La stessa cosa può succedere anche coi propri listati, leggendoli dopo qualche tempo non ci si raccapezza più. Ovviamente le varie linee REM eventualmente inserite possono aiutare molto nella sua comprensione, ma lo schema logico d'assieme su cui è basato il suo funzionamento non appare mai troppo evidente.

Per tutti questi motivi riteniamo molto importante che prendiate confidenza con gli schemi a blocchi e coi diagrammi di flusso. Per SCHEMI A BLOCCHI intendiamo riferirci a quelle rappresentazioni grafiche non troppo particolareggiate che danno a grandi linee lo schema del programma; i DIAGRAMMI DI FLUSSO, invece, scendono nei particolari delle varie istruzioni, rappresentandole graficamente. La potenza espressiva di tali grafici giustifica ampiamente quel poco d'impegno che occorre mettere per apprenderli. Chi non fosse convinto di quanto diciamo, provi pure a leggere direttamente il listato del programma di queste pagine: gli ci vorrà molto tempo per orientarsi, e buon per lui che ci sono i vari REM ad aiutarlo.

Innanzitutto esaminiamo la figura 1; essa contiene i vari simboli grafici usati per comporre i diagrammi di flusso. Col passare degli anni essi si sono imposti nella foggia in cui ve li presentiamo; sui libri o sulle riviste specializzate li ritroverete quindi uguali o con differenze minime. Ogni simbolo porta la relativa spiegazione, perciò non ci dilunghiamo in lunghe descrizioni. Più avanti, a cominciare dalla figura 4, vedremo poi il diagramma di flusso del programma proposto, quindi avrete modo di esercitarvi in questa tecnica. Il diagramma è stato suddiviso in varie parti, per comodità di consultazione; ogni sezione corrisponde ad un pezzo di programma compreso tra due REM consecutive.

La figura 2 è invece lo schema a blocchi dello stesso programma: come vedete, poche linee riescono efficacemente ad esprimere la sua struttura, meglio di quanto potrebbero fare molte parole. Un solo colpo d'occhio trasmette un intero concetto.

Non fatevi impressionare dall'apparente complessità di queste figure; man mano che procederemo nella spiegazione del programma, esse vi diverranno sempre più chiare ed utili. Comunque è nostra intenzione fornire i diagrammi di flusso di tutti gli esempi che faremo, e dopo poco tempo, quando li avrete assimilati, dovrete convenire che ne valeva la pena.

IL PROGRAMMA

Approntare un programma a scopi didattici non è mai una cosa semplice, soprattutto perché la semplicità necessaria alla chiarezza dell'esposizione male si concilia col desiderio di offrire qualcosa che abbia anche una utilità pratica. Succede infatti che gli esempi che sono stati scelti per essere chiari fin dal primo momento, sembrino raggiungere lì per lì lo scopo, ma poi presto ci si accorge che il lettore non riesce mai a metterli in pratica perché non li ha capiti. È successo che l'esempio era sì di facile comprensione, ma era anche talmente schematizzato e semplificato da non mettere in grado l'allievo di farsi un'idea esatta e soprattutto pratica di quel concetto.

Ci perdonerete se torniamo spesso sul discorso della pratica, ma siamo convinti che solo essa riesca a dare la padronanza delle varie tecniche; la teoria è una bellissima cosa e rappresenta la base indispensabile per poter veramente capire un fenomeno, ma solo se riusciamo a tradurla in pratica ci sarà servita a qualcosa. Per questo vi invitiamo ad eseguire alla tastiera i vari esempi proposti, oppure a farne di vostri; l'esperienza che acquisterete provandoli sarà maggiore rispetto a quella che potrete apprendere limitandovi a leggere qualsiasi tasto che tratti l'argomento.

Detto questo, passiamo ad esaminare il programma che abbiamo chiamato GESTIONE, dando innanzitutto un elenco degli obiettivi che esso si propone:

1 - memorizzare fornitori e clienti
2 - possibilità di correggere i nominativi (errori, cambio di indirizzo, ecc.)
3 - memorizzare le fatture, i movimenti di merci, i pagamenti, ecc.
4 - avere un quadro immediato della situazione finanziaria
5 - ottenere una certa velocità d'esecuzione
6 - non occupare troppa memoria

Gli ultimi due punti sono palesemente in contrasto tra loro; infatti una elevata velocità nell'esecuzione dei programmi si ottiene soprattutto caricando in memoria i vari dati da elaborare, ossia generando dei vettori. Così, quando necessiteremo di un dato, esso si troverà già in macchina e potrà essere elaborato immediatamente. Ciò però comporta una buona disponibilità di memoria, che cresce a dismisura se si ha a che fare con stringhe.

Provate infatti a fare un semplice conto: ogni nominativo, sia esso un cliente od un fornitore, occupa (tra nome, indirizzo completo, telefono, tipo di attività e di merce trattata, importi dei movimenti, ecc.) parecchi caratteri.

Supponiamo di dedicare ad ogni nominato 255 caratteri; tale cifra risulta molto comoda, come vedremo, perché corrisponde alla minima quantità trattabile con un file di tipo RANDOM. I vari dati, poi, sono quasi tutti sotto forma di stringhe, quindi per memorizzarli in vettori occorre anche riservare loro il posto necessario con una adeguata istruzione di CLEAR. Infatti se provate a memorizzare più di 50 caratteri di stringa, il computer vi segnala errore.

Oltrepassata quella cifra, che esso assegna automaticamente con il caricamento del BASIC, occorre dare da programma l'ingombro di memoria necessario.

Supponendo allora di voler memorizzare in vettori 1000 nominativi tra clienti e fornitori, dovremmo riservare 1000 volte 255 caratteri, vale a dire oltre 250 kilobytes di memoria; tanta! per un personal computer.

Mancano poi ancora le registrazioni di tutti i movimenti, ossia le descrizioni degli acquisti effettuati presso i vari fornitori, quelle delle vendite ai clienti, gli importi, le date delle fatture, i termini di pagamento, le eventuali riparazioni o scadenze di garanzia, e chi più ne ha più ne metta. A tutto questo si aggiunga che non tutti i lettori che hanno realizzato il nostro computer hanno una capacità di memoria estesa fino al

massimo di 56 K, mentre a noi interessa che anche chi ha meno memoria abbia la possibilità di far girare questo programma.

Diciamo tutto questo non solo per farvi capire le difficoltà che si presentano nel nostro caso specifico, quando dobbiamo spiegarvi qualcosa, ma anche perché le considerazioni appena svolte vanno fatte ogniqualvolta ci si accinge alla stesura di un nuovo programma.

Infatti, se non si ragiona in questi termini, ossia se non si fa un bilancio tra le esigenze del programma e le capacità del computer, si rischia di costruire un bel castello di istruzioni che non riusciremo mai a collocare per carenza di memoria.

Il programma che segue non usa affatto i vettori, quindi non pone problemi di memoria. La velocità d'esecuzione risulta pertanto un po' rallentata, ma meno di quanto si possa pensare.

Detto questo, possiamo passare ad esaminare riga per riga il listato. **Mano a mano che andremo avanti, seguite il relativo diagramma di flusso; in tal modo capirete meglio le spiegazioni ed anche il diagramma stesso, poco per volta, vi apparirà chiaro. I numeri a fianco dei vari simboli grafici indicano la linea od il gruppo di linee del programma che eseguono quelle funzioni; c'è pertanto una perfetta corrispondenza tra grafico e righe di programmazione.**

Descriveremo, più o meno diffusamente, tutte le linee del listato, soffermandoci ovviamente più a lungo sulle istruzioni più importanti. Coloro che sono alle prime armi faranno bene a leggersi tutto molto attentamente: crediamo che, seppure con qualche comprensibile difficoltà, riusciranno a capire tutto. I più esperti potranno andare più veloci e soffermarsi solo nelle parti che riguardano la gestione dei files.

Ancora una cosa: abbiamo pensato di fare cosa utile riportando in figura 3 l'elenco degli identificatori di variabile usati, con accanto le relative linee in cui compaiono. Tale elenco si ottiene, a programma finito, introducendo il comando REF$.

```
1000 '=============================================================================
1010 '              GESTIONE AZIENDALE 1                    1982 NE-R.C.
1020 '=============================================================================
1030 CLEAR 3000
1040 ONERRORGOTO 10220
1050 CLS
1060 REVON: PRINT@166," gestione aziendale " : REVOFF
1070 PRINT : PRINT : PRINT
1080 GOSUB 10040
1090 OPEN"r",1,"dati"
1100 FIELD1, 8 AS S$, 2 AS N$, 2 AS M$, 243 AS A$
1110 GET1,1
1120 SC#=CVD(S$) : NC%=CVI(N$) : MC%=CVI(M$)
1130 GET1,2
1140 SF#=CVD(S$) : NF%=CVI(N$) : MF%=CVI(M$)
1150 CLOSE
1160 '+++++++++++++++++++++++++++++++ MENU' PRINCIPALE ++++++++++++++++++++++++
1170 RM=0 : CR=0
1180 CLS
1190 PRINT : REVON: PRINT"1"; : REVOFF: PRINT" > registrazioni clienti"
1200 PRINT : REVON: PRINT"2"; : REVOFF: PRINT" > registrazioni fornitori"
1210 PRINT : REVON: PRINT"3"; : REVOFF: PRINT" > letture clienti"
1220 PRINT : REVON: PRINT"4"; : REVOFF: PRINT" > letture fornitori"
1230 PRINT : REVON: PRINT"5"; : REVOFF: PRINT" > situazione pagamenti"
1240 M=5
1250 GOSUB 10010
1260 IF X=5 GOTO 3180
1270 IF X=1 OR X=3 THEN C=1 : N=NC% : NO$="cliente" : NN$="movcl"
1280 IF X=2 OR X=4 THEN C=0 : N=NF% : NO$="fornitore" : NN$="movfo"
1290 ON X GOTO 1310,1310,2620,2620
1300 '++++++++++++++++++++++++++++++ MENU' REGISTRAZIONI ++++++++++++++++++++++
1310 CLS
1320 REVON: PRINT"1"; : REVOFF: PRINT" > per registrare movimenti"
1330 PRINT : REVON: PRINT"2"; : REVOFF: PRINT" > per aggiungere un ";NO$
1340 PRINT : REVON: PRINT"3"; : REVOFF: PRINT" > per correggere un ";NO$
1350 PRINT : REVON: PRINT"4"; : REVOFF: PRINT" > per tornare al menu'"
1360 M=4
1370 GOSUB 10010
1380 ON X GOTO 2220,1400,1980,1170
1390 '------------------------------ AGGIUNTA NOMINATIVI ----------------------
1400 TX$=" nome          " : Q=30
1410 GOSUB 10110
1420 LINEINPUT N$
1430 IF LEN(N$)>30 GOTO 1400
1440 IF LEN(N$)<30 THEN N$=N$+" " : GOTO 1440
```

```
1450 TX$=" via          "
1460 GOSUB 10110
1470 LINEINPUT V$
1480 IFLEN(V$)>30 GOTO 1450
1490 IF LEN(V$)<30 THEN V$=V$+" " : GOTO 1490
1500 TX$=" citta'       "
1510 GOSUB 10110
1520 LINEINPUT C$
1530 IF LEN(C$)>30 GOTO 1500
1540 IF LEN(C$)<30 THEN C$=C$+" " : GOTO 1540
1550 TX$=" telefono     " : Q=20
1560 GOSUB 10110
1570 LINEINPUT T$
1580 IF LEN(T$)<20 GOTO 1550
1590 IF LEN(T$)>20 THEN T$=T$+" " : GOTO 1590
1600 TX$=" annotazioni " : Q=126
1610 GOSUB 10110
1620 LINEINPUT R$
1630 IF LEN(R$)>126 GOTO 1600
1640 IF CR=0 THEN I#=0 : P#=0 : GOTO 1710
1650 TX$=" tot.fattura " : Q=16
1660 GOSUB 10110
1670 INPUT I#
1680 TX$=" acconto      "
1690 GOSUB 10110
1700 INPUT P#
1710 Z$=N$+V$+C$+T$+R$
1720 CLS
1730 GOSUB 10040
1740 IF C=0 THEN NF%=NF%+1 : PU=NF% ELSE NC%=NC%+1 : PU=NC%
1750 IF CR=1 THEN PU=N
1760 OPEN"r",1,NO$
1770 FIELD1, 8 AS I$, 8 AS P$, 239 AS W$
1780 LSET I$=MKD$(I#) : LSET P$=MKD$(P#) : LSET W$=Z$
1790 PUT1, PU
1800 OPEN"r",2,"dati"
1810 FIELD2, 8 AS S$, 2 AS N$, 2 AS M$, 243 AS A$
1820 IF C=0 GOTO 1890
1830 IF RM=0 THEN SC#=SC#+I#-P# ELSE SC#=SC#+IM#-PM#
1840 LSET S$=MKD$(SC#)
1850 LSET N$=MKI$(NC%)
1860 LSET M$=MKI$(MC%)
1870 LSET A$=R$
1880 PUT2,1 : GOTO 1950
1890 IF RM=0 THEN SF#=SF#+I#-P# ELSE SF#=SF#+IM#-PM#
1900 LSET S$=MKD$(SF#)
1910 LSET N$=MKI$(NF%)
1920 LSET M$=MKI$(MF%)
1930 LSET A$=R$
1940 PUT2,2
1950 CLOSE
1960 IF RM=1 GOTO 2220 ELSE GOTO 1170
1970 '-------------------------------- CORREZIONE NOMINATIVI -------------------
1980 CR=1
1990 CLS
2000 PRINT"quale numero di ";NO$
2010 PRINT"vuoi correggere ?" : PRINT : PRINT "(0 per tornare al menu')" : PRINT
2020 INPUT N
2030 IF N=0 GOTO 1170
2040 IF C=0 AND N>NF% GOTO 1980
2050 IF C=1 AND N>NC% GOTO 1980
2060 CLS
2070 GOSUB 10040
2080 OPEN"r",1,NO$
2090 FIELD1, 8 AS I$, 8 AS P$, 239 AS Z$
2100 GET1,N
2110 CLOSE
2120 CLS
2130 PRINT NO$ : PRINT
2140 REVON: PRINT LEFT$(Z$,30) : REVOFF
2150 GOSUB 10050
2160 IF X$="0" GOTO 1170
2170 CLS
2180 IF C=0 THEN NF%=NF%-1 ELSE NC%=NC%-1
```

```
2190 I#=0 : P#=0
2200 GOTO 1400
2210 '---------------------------------- REGISTRAZIONE MOVIMENTI ----------------
2220 RM=1
2230 CLS
2240 PRINT"numero del ";NO$ : PRINT"(0 per finire)" : PRINT
2250 INPUT N
2260 IF N=0 GOTO 1170
2270 IF C=1 AND N>NC% GOTO 2220
2280 IF C=0 AND N>NF% GOTO 2220
2290 CLS
2300 GOSUB 10040
2310 OPEN"r",1,NO$
2320 FIELD1, 8 AS I$, 8 AS P$, 239 AS Z$
2330 GET1,N
2340 II#=CVD(I$) : PP#=CVD(P$)
2350 CLS
2360 PRINT NO$;" n. ";N : PRINT
2370 REVON: PRINT LEFT$(Z$,30) : REVOFF
2380 GOSUB 10050
2390 IF X$="0" THEN CLOSE : GOTO 1170
2400 PRINT@128,CHR$(31);
2410 LINEINPUT "data (gg.mm.aa): ";D$
2420 PRINT"merce    :  (max. 95 caratteri)" : LINEINPUTM$
2430 IF LEN(M$)>95 PRINT@160,CHR$(31); : GOTO 2420
2440 PRINT@288,""; : INPUT"importo :";IM#
2450 INPUT"acconto :";PM#
2460 PRINT"note     :  (max. 63 caratteri)" : LINEINPUT A$
2470 IF LEN(A$)>63 PRINT@352,CHR$(31); : GOTO 2460
2480 REVON: PRINT@480," confermi ? (s/n) "; : REVOFF
2490 X$=INKEY$
2500 IF X$="s" OR X$="n" GOTO 2520
2510 GOTO 2490
2520 IF X$="n" THEN CLOSE : GOTO 2220
2530 CLS
2540 GOSUB 10040
2550 OPEN"e",2,NN$
2560 PRINT#2, N; ","; D$; ","; M$; ","; IM#; PM#; ","; A$; ",";
2570 CLOSE
2580 I#=II#+IM# : P#=PP#+PM#
2590 IF C=1 THEN MC%=MC%+1 : NC%=N-1 ELSE : MF%=MF%+1 : NF%=N-1
2600 GOTO 1740
2610 '++++++++++++++++++++++++++++++++ MENU' LETTURE ++++++++++++++++++++++++++++
2620 CLS
2630 REVON: PRINT"1"; : REVOFF: PRINT" > per leggere movimenti"
2640 PRINT : REVON: PRINT"2"; : REVOFF: PRINT" > per leggere nomi"
2650 PRINT : REVON: PRINT"3"; : REVOFF: PRINT" > per tornare al menu'"
2660 M=3
2670 GOSUB 10010
2680 ON X GOTO 2950,2700,1170
2690 '---------------------------------- LETTURA NOMINATIVI ----------------------
2700 CLS
2710 IF C=1 AND NC%=0 GOTO 1170
2720 IF C=0 AND NF%=0 GOTO 1170
2730 GOSUB 10040
2740 IF C=1 THEN H=NC% ELSE H=NF%
2750 OPEN"r",1,NO$
2760 FIELD1, 8 AS I$, 8 AS P$, 239 AS Z$
2770    FOR I=1 TO H
2780    GET1,I
2790    CLS
2800    REVON: PRINT NO$;" numero";I : REVOFF
2810    N$=LEFT$(Z$,30) : V$=MID$(Z$,31,30) : C$=MID$(Z$,61,30)
2820    T$=MID$(Z$,91,20) : A$=RIGHT$(Z$,129)
2830    I#=CVD(I$) : P#=CVD(P$)
2840    PRINT : PRINTN$ : PRINTV$ : PRINTC$
2850    PRINT : PRINT"telefono : ";T$
2860    REVON: PRINT" note " : REVOFF: PRINTA$
2870    PRINT"tot.fatt.:";I#
2880    PRINT"acconto  :";P#
2890    GOSUB 10050
2900    IF X$="0" GOTO 2920
2910    NEXT
2920 CLOSE
```

```
2930 GOTO 1170
2940 '--------------------------------- LETTURA MOVIMENTI ----------------------
2950 CLS
2960 IF C=1 AND MC%=0 GOTO 1170
2970 IF C=0 AND MF%=0 GOTO 1170
2980 GOSUB 10040
2990 IF C=1 THEN H=MC% ELSE H=MF%
3000 OPEN"i",1,NN$
3010    FOR I=1 TO H
3020    INPUT#1, N, D$, M$, I#, P#, A$
3030    CLS : REVON
3040    IF C=0 PRINT"fornitore";N;
3050    IF C=1 PRINT"cliente";N;
3060    PRINT" * movim.";I
3070    PRINT : PRINT" data       " ; : REVOFF: PRINT " "; : PRINT D$
3080    PRINT : REVON: PRINT" merce      " : REVOFF: PRINT M$
3090    REVON: PRINT" tot.fatt. "; : REVOFF: PRINT I#
3100    PRINT : REVON: PRINT" acconto    "; : REVOFF: PRINT P#
3110    PRINT : REVON: PRINT" note       " : REVOFF: PRINT A$
3120    GOSUB 10050
3130    IF X$="0" GOTO 3150
3140    NEXT
3150 CLOSE
3160 GOTO 1170
3170 '--------------------------------- SITUAZIONE DEI PAGAMENTI ----------------
3180 CLS
3190 REVON: PRINT" situazione fornitori: " : REVOFF
3200 PRINT : PRINT
3210 IF SF#>0 PRINT"   da pagare l.";SF#
3220 IF SF#<0 PRINT"   pagate in pu' l.";ABS(SF#)
3230 IF SF#=0 PRINT"   fatture tutte saldate"
3240 PRINT : REVON: PRINT" situazione clienti  : ":REVOFF
3250 PRINT : PRINT
3260 IF SC#>0 PRINT"   da incassare l.";SC#
3270 IF SC#<0 PRINT"   incassate in piu' l.";ABS(SC#)
3280 IF SC#=0 PRINT"   nessun cliente insoluto"
3290 PRINT : REVON: PRINT" > ";
3300 IF SF#>SC# PRINT "dare l.";SF#-SC#
3310 IF SC#>SF# PRINT"avere l.";SC#-SF#
3320 IF SC#=SF# PRINT"    pareggio  "
3330 REVOFF
3340 GOSUB 10070
3350 GOTO 1170
10000 '--------------------------------- SUBROUTINES ----------------------------
10010 PRINT@448,"scegli";
10020 X=VAL(INKEY$)
10030 IF X<1 OR X>M GOTO 10020 ELSE RETURN
10040 PRINT"programma in corso  -  attendere" : RETURN
10050 REVON: PRINT@480," premi un tasto (0 per finire) "; : REVOFF
10060 GOTO 10080
10070 REVON: PRINT@480," premi un tasto "; : REVOFF
10080 X$=INKEY$
10090 IF X$="" GOTO 10080
10100 RETURN
10110 CLS
10120 REVON: PRINT@192,TX$ : REVOFF
10130    FOR I=1 TO 5
10140    PRINT@207,"                "
10150       FOR J=1 TO 15
10160       NEXT J
10170    PRINT@207,Q;" caratteri"
10180       FOR J=1 TO 15
10190       NEXT J
10200    NEXT I
10210 PRINT@288,""; : RETURN
10220 PRINT ERR/2+1 : STOP
```

RIFERIMENTI del Programma: GESTIONE AZIENDALE 1

```
    0  1170/2 1280 1640/2
       1740 1820 1830 1890
       2030 2040 2180 2220/2
       2260 2280 2710 2720/2
       2940 2950/2 3180
       3190 3200 3230 3240
       3250
    1  1090 1100 1110/2
       1130 1270/2 1740/2
       1750/2 1760 1770
       1790 1880 1960 1980
       2050 2080 2090 2100
       2180 2190/2 2220
       2270 2310 2320 2330
       2590/3 2710 2740
       2750 2760 2770 2780
       2940 2970 2980 2990
       3000 10030 10130
       10150 10180 10220
    2  1100/2 1130 1280
       1800 1810/3 1880
       1940/2 2550 2560
       10220
    3  1270 2660
    4  1280 1360
    5  1240 1260 10130
    8  1100 1770/2 1810
       2090/2 2320/2 2760/2
   11  1180
   15  10150 10180
   16  1650
   20  1550 2760
   30  1400 2140 2370 2760/3
   31  2400 2430 2470
   34  1030
   63  2470
   95  2430
  128  2400
  129  1600 2760
  160  2430
  166  1060
  192  10120
  207  10140 10170
  239  1770 2090 2320
  243  1100 1810
  288  2440
  352  2470
  448  10010
  480  2480 10050 10070
 1000  1030
 1170  1380 1960 2030 2160
       2260 2390 2680 2710
       2720 2910 2940 2950
       3130 3320
 1310  1290/2
 1400  1380 1430 2200
 1440  1440
 1450  1480
 1490  1490
 1500  1530
 1540  1540
 1550  1580
 1590  1590
 1600  1630
 1750  2600
 1890  1820
 1950  1880
 1980  1380 2040 2050
```

```
 2220  1380 1960 2270 2280
       2520
 2420  2430
 2460  2470
 2490  2510
 2520  2500
 2620  1290/2
 2700  2680
 2900  2880
 2930  2680
 3120  3100
 3150  1260
10010  1250 1370 2670
10020  10030
10040  1080 1730 2070 2300
       2540 2730 2960
10050  2150 2380 2870 3090
10070  3310
10080  10060 10090
10110  1410 1460 1510 1560
       1610 1660 1690
10220  1040
   A   1100/$ 1810/$ 1870/$
       1930/$ 2460/$ 2470/$
       2560/$ 2760/$ 2840/$
       3000/$ 3080/$
   C   1270 1280 1520/$
       1530/$ 1540/$3 1710/$
       1740 1820 2040 2050
       2180 2190 2270 2280
       2590 2710 2720 2740
       2760/$ 2820/$ 2940
       2950 2970
  CR   1170 1750 1980
   D   2410/$ 2560/$ 3000/$
       3040/$
  FF   1060 1190 1200 1210
       1220 1230 1320 1330
       1340 1350 2140 2370
       2480 2630 2640 2650
       2800 2840 3040 3050
       3060 3070 3080 3160
       3210 3300 10050 10070
       10120
   H   2740/2 2770 2970/2
       2990
   I   1640/# 1670/# 1770/$
       1780/$ 1780/# 1830/#
       1890/# 2090/$ 2320/$
       2340/$ 2340/# 2580/#2
       2760/$ 2770 2780
       2800 2810/$ 2810/#
       2850/# 2990 3000/#
       3030 3060/# 10130
       10200
  IM   1830/# 1890/# 2220/#
       2440/# 2560/# 2580/#
   J   10150 10160 10180
       10190
   M   1100/$ 1120/$ 1140/$
       1240 1360 1810/$
       1860/$ 1920/$ 2420/$
       2430/$ 2560/$ 2660
       3000/$ 3050/$ 10030
  MC   1120/% 1860/% 2590/%2
       2940/% 2970/%
  MF   1140/% 1920/% 2590/%2
       2950/% 2970/%
```

```
N    1060 1100/$ 1120/$
     1140/$ 1180 1200
     1210 1220 1230 1320
     1330 1340 1350 1420/$
     1430/$ 1440/$3 1710/$
     1750 1810/$ 1850/$
     1910/$ 2020 2030
     2040 2050 2100 2140
     2250 2260 2270 2280
     2330 2360 2370 2480
     2560 2630 2640 2650
     2760/$ 2800 2820/$
     2840 3000 3010 3020
     3050 3060 3070 3080
     3160 3210 3260 10050
     10070 10120
MC   1120/% 1740/%3 1850/%
     2050/% 2190/%2 2270/%
     2710/% 2740/%
MF   1140/% 1740/%3 1910/%
     2040/% 2180/%2 2280/%
     2720/% 2740/%
MM   1270/$ 1280/$ 2550/$
     2980/$
MO   1270/$ 1280/$ 1330/$
     1340/$ 1760/$ 2000/$
     2080/$ 2130/$ 2240/$
     2310/$ 2360/$ 2750/$
     2800/$ 3020/$
P    1640/# 1700/# 1770/$
     1780/$ 1780/# 1830/#
     1890/# 2090/$ 2320/$
     2340/$ 2340/# 2580/#2
     2760/$ 2810/$ 2810/#
     2860/# 3000/# 3070/#
PM   1830/# 1890/# 2220/#
     2450/# 2560/# 2580/#
PU   1740/2 1750 1790
Q    1400 1430 1440 1480
     1490 1530 1540 1550
     1580 1590 1600 1630
     1650 10170
R    1620/$ 1630/$ 1710/$
RM   1170 1750 1830 1890
     1960 2220
S    1100/$ 1120/$ 1140/$
     1810/$ 1840/$ 1900/$
SC   1120/# 1830/#4 1840/#
     3230/#2 3240/#2 3250/#
     3270/#2 3280/#2 3290/#
SF   1140/# 1890/#4 1900/#
     3180/#2 3190/#2 3200/#
     3270/#2 3280/#2 3290/#
T    1570/$ 1580/$ 1590/$3
     1710/$ 2760/$ 2830/$
TX   1400/$ 1450/$ 1500/$
     1550/$ 1600/$ 1650/$
     1680/$ 10120/$
V    1470/$ 1480/$ 1490/$3
     1710/$ 2760/$ 2820/$
W    1770/$ 1780/$
X    1260 1270/2 1280/2
     1290 1380 2160/$
     2390/$ 2490/$ 2500/$2
     2520/$ 2680 2880/$
     3100/$ 10020 10030/2
     10080/$ 10090/$
Y    1030/$ 2560/$6
Z    1710/$ 1780/$ 2090/$
     2140/$ 2320/$ 2370/$
```

DESCRIZIONE DEI RIFERIMENTI

La lista dei riferimenti del programma esemplificativo della Gestione Aziendale riportata qui di lato potremo ricavarla utilizzando i comandi REF* e REF$ come spiegato a pag. 46.

Se digitiamo REF*, questi riferimenti vengono listati sul monitor video e poiché la lista è piuttosto lunga potrete arrestarla temporaneamente con i tasti SHIFT più @ (vedi pag. 63) e pigiare in seguito un qualsiasi tasto per farla proseguire.

Se digitiamo REF$ tutto il listato viene stampato su carta.

Ci preme precisare che non tutti i microcomputer permettono di ottenere le condizioni sopra riportate, cioè listare i riferimenti sia su video che su stampante, quindi il Basic in vostro possesso va apprezzato anche per questa caratteristica. Infatti quando avrete imparato a decifrare questi numeri, vi sarà in seguito molto più facile correggere o modificare programmi molto complessi.

Guardando la lista dei riferimenti nella prima colonna di sinistra abbiamo dei numeri crescenti in ordine di grandezza cioè 0-1-2-3-160-166 ecc. e delle lettere A-C-CR ecc. e in corrispondenza di questa sulla colonna di destra degli altri numeri, o gruppi di numeri che corrispondono alla linea del programma.

I numeri e le lettere alfabetiche dalla colonna di sinistra indicano a quali **linee di riferimento** esistono variabili o istruzioni che fanno riferimento a quel valore. Ad esempio il numero 0 lo troviamo due volte alla linea 1770 (vedi 1170/2) per l'assegnazione delle variabili RM e CR una volta alla linea 1280 e due volte alla linea 1640.

Il numero 11 lo troviamo una sola volta alla linea 1180 e nel programma gestionale questo si riferisce all'istruzione CLS del Basic.

Oltre alla segnalazione di variabili, alle istruzioni e ai test condizionati del programma, i numeri sulla sinistra indicano anche un richiamo a quella linea, ad esempio le istruzioni GOTO, GOSUB, THEN ecc. (ad esempio al numero 10220 troviamo a destra 1040 che è un salto condizionato a tale linea).

Proseguendo, terminati i numeri troveremo delle lettere A-C-CR-D ecc. che corrispondono a variabili, e il numero che troveremo sulla colonna di destra corrispondono al numero di linea dove queste variabili vengono utilizzate, sia come assegnazione che come calcolo.

I simboli e numeri che troviamo riportati dopo il numero di linea vanno così decifrate.

0 1170M2 significa che alla linea 1170 lo 0 è utilizzato 2 volte

H 2740/2 significa che alla linea 2740/2 la variabile H è utilizzata 2 volte

D 2490/$ significa che la variabile D$ è utilizzata una sola volta nella linea 2510

I 1640/# significa che la variabile I è assegnata alla riga 1640 in doppia precisione

MC 1120/% significa che la variabile MC assegnata alla riga 1120 è definita intera.

Con questo simbolo si usa indicare l'inizio e la fine del programma. All'interno di questa figura si scrive solo START per indicare il punto d'inizio del programma e END per il termine dello stesso programma.

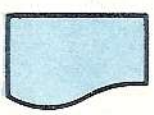
Questo simbolo rettangolare serve per l'elaborazione dei dati, cioè viene usato genericamente per indicare uno o più operazioni svolte nel programma, quali ad esempio: preparazione, calcolo, trattamento di stringhe ecc. ecc.

Il rombo si usa per test decisionali, cioè serve quando nel corso dell'elaborazione bisogna operare il cosidetto «salto condizionato» ovverosia decidere su che strada proseguire, tra diverse possibili, a secondo delle condizioni verificatesi in precedenza. Si usa con le istruzioni tipo IF... THEN... ELSE ed altre.

Un rettangolo con doppia linea ai lati serve per indicare un sottoprogramma e subroutine, cioè quel particolare programma che avremo necessità di utilizzare più volte e che possiamo chiamare con l'istruzione GOSUB. Le subroutine sono molto importanti per semplificare la struttura di qualsiasi programma.

Il simbolo del trapezio viene usato per l'entrata e l'uscita di dati (INPUT - OUTPUT). Indifferentemente può essere utilizzato per indicare sia l'entrata o uscita dati da porte periferiche, che da consolle di dialogo con il computer quale ad esempio la tastiera alfanumerica.

Questo simbolo lo si utilizza quando si desiderano visualizzare dei dati o disegni grafici sul monitor video. Serve quando necessitano dei tabulati, mascherine grafiche ecc. ecc.



Simbolo da utilizzare per far uscire dei dati su stampante, viene usato cioè nel corso del programma quando si desidera che vengano stampati su carta dei dati elaborati (ad esempio, elenchi, tabulati, fatture, cataloghi, ecc. ecc.).

Quando in una FLOW-CHART (stesura di un programma disegnata con i simboli qui riportati) occorre collegare tra di loro più simboli anzichè tracciare delle righe che potrebbero complicare il disegno si preferisce mettere questo simbolo nei due punti da collegare scrivendo nell'interno delle lettere dall'alfabeto per distinguerli A-A, B-B; ecc. ecc.

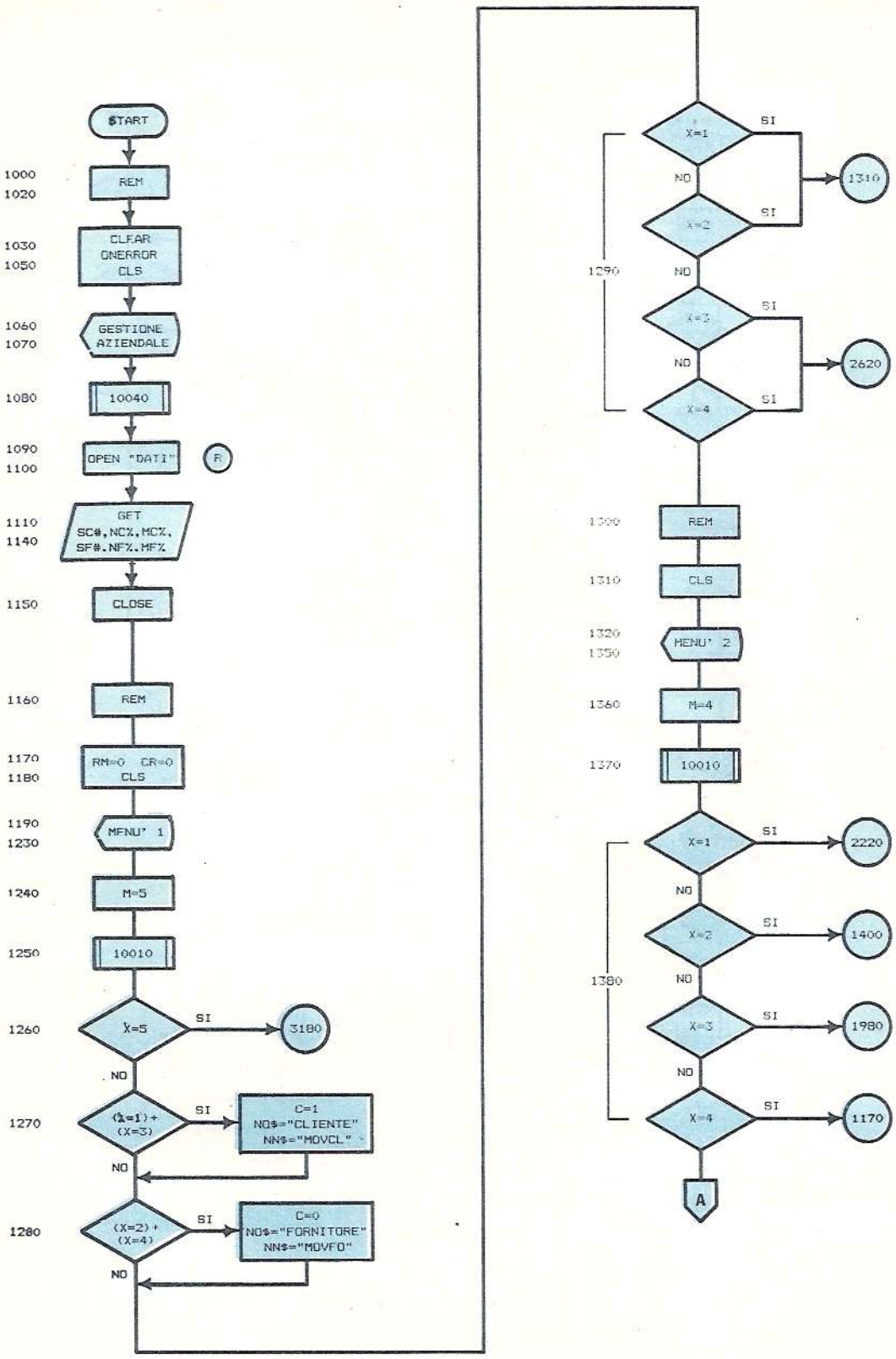
Simbolo impiegato quando disegnando una FLOW-CHART si raggiunge il fine pagina, senza aver terminato il programma, quindi si deve proseguire in altro foglio. Se la parte triangolare è rivolta in basso significa che tale programma segue in altra pagina, se rivolta in alto vuol dire che questo programma va collegato ad una già preesistente. Nell'interno di tale simbolo si scriverà un numero, quindi è ovvio che se scriviamo 1 il secondo pezzo di programma da collegare porterà il n. 2 ecc. ecc.

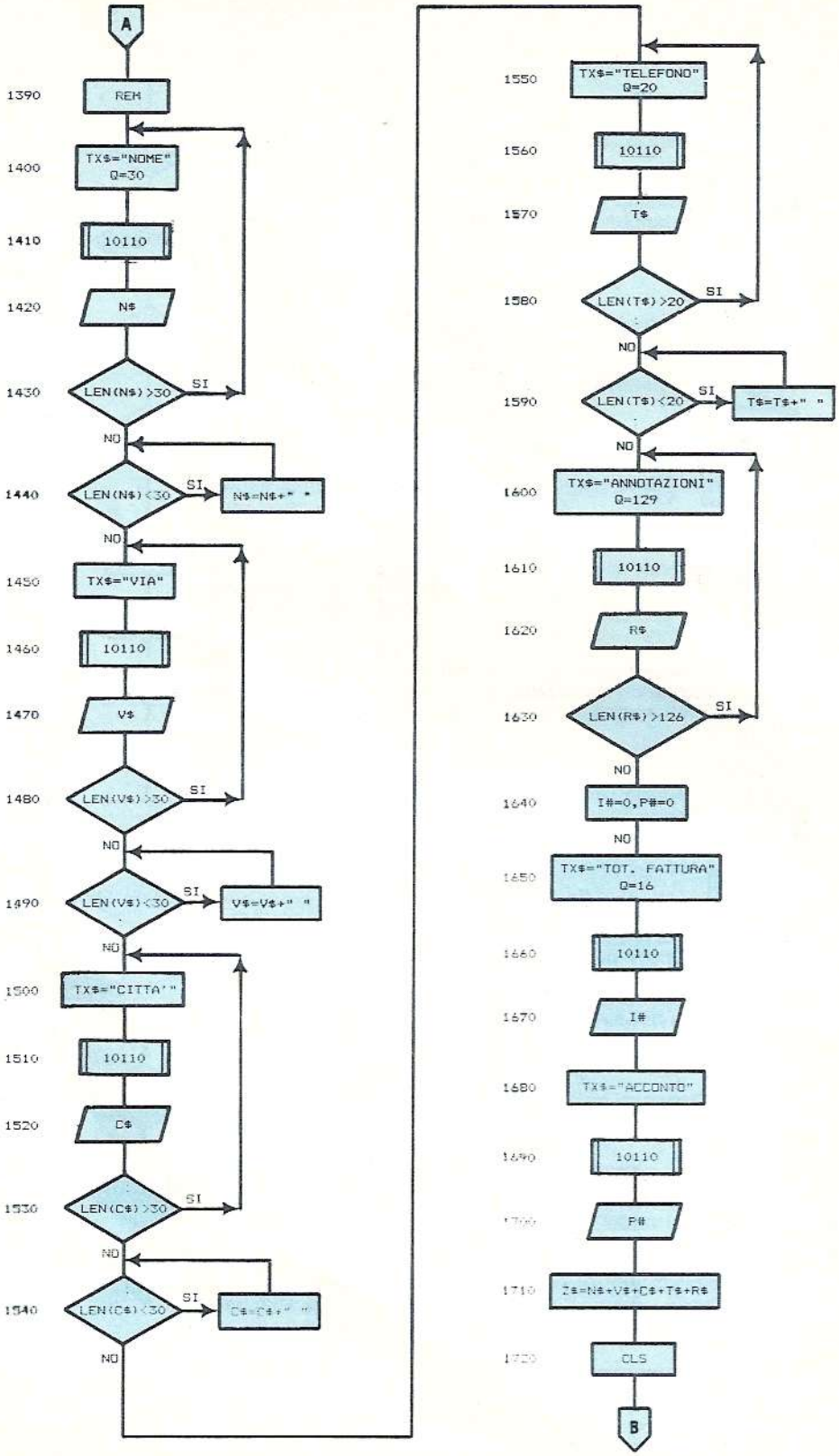
Questa linea viene utilizzata per collegare tra di loro varie parti di programma. La freccia indica il senso di percorrenza per poter facilitare la lettura della Flow-Chart.

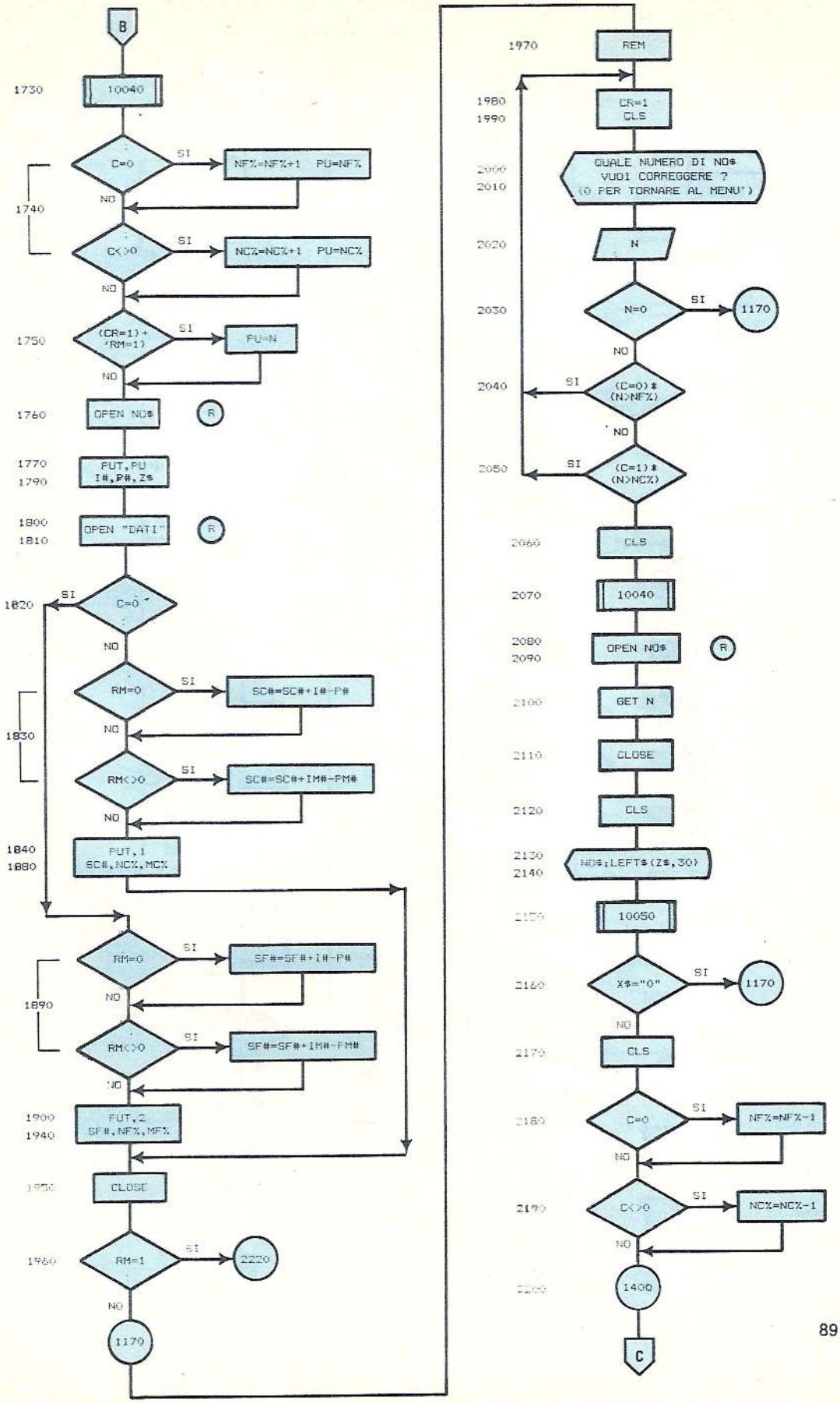
La linea tratteggiata viene utilizzata per evidenziare particolari parti di programma, ad esempio viene impiegata per delimitare i cicli di FOR....NEXT.

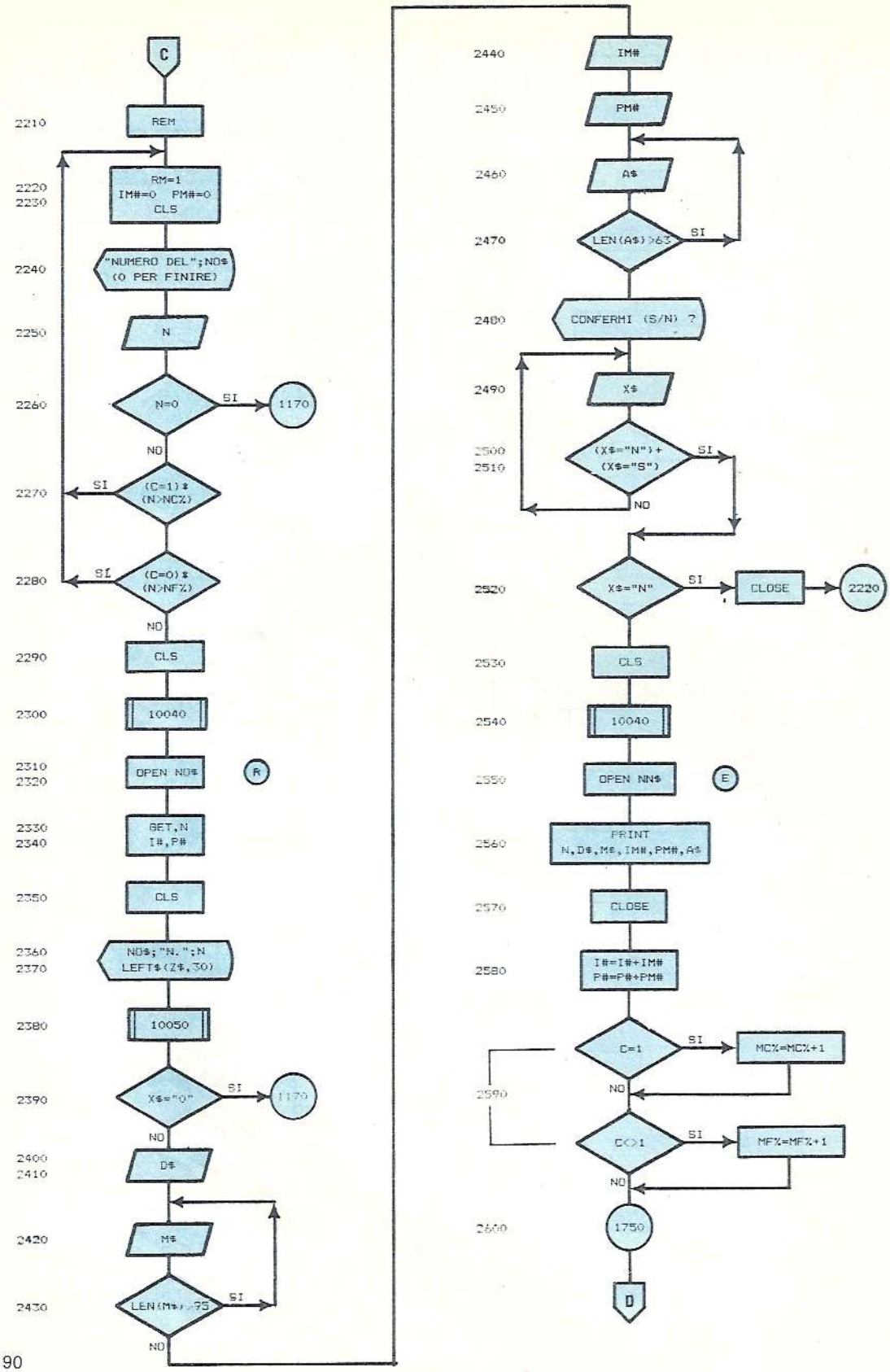
Un cilindro indica una memoria di massa per archiviazione. Più precisamente serve ad indicare che i dati elaborati verranno memorizzati, su dischetti Floppy, oppure su nastro magnetico.

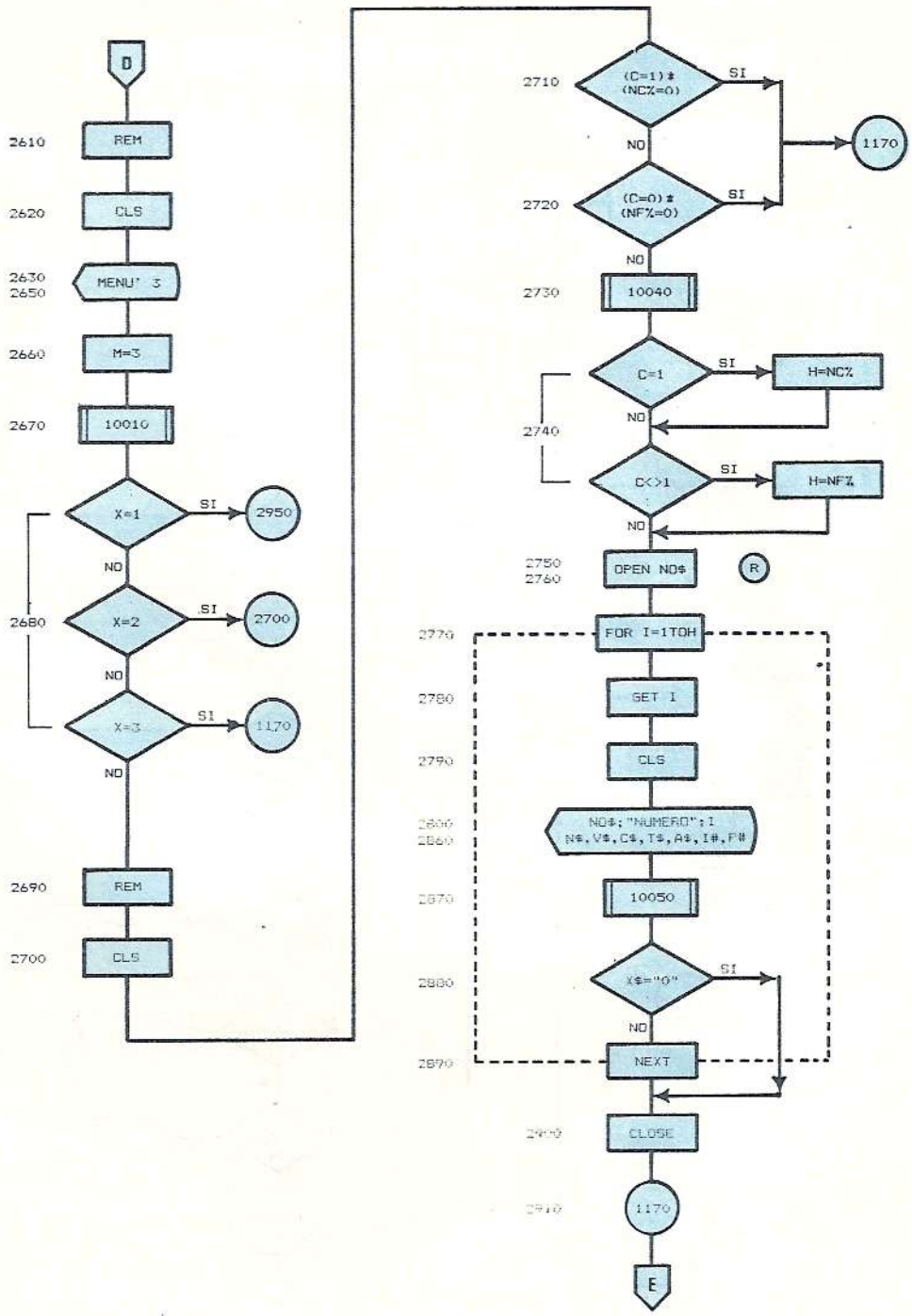
NOTA - Per comprendere meglio come possiamo impiegare questi simboli per realizzare una Flow Chart consigliamo di leggere attentamente il programma gestionale descritto nelle pagine che seguono. In tale programma abbiamo cercato di inserirli tutti per cercare di capire come dobbiamo impiegarli per preparare altri programmi.

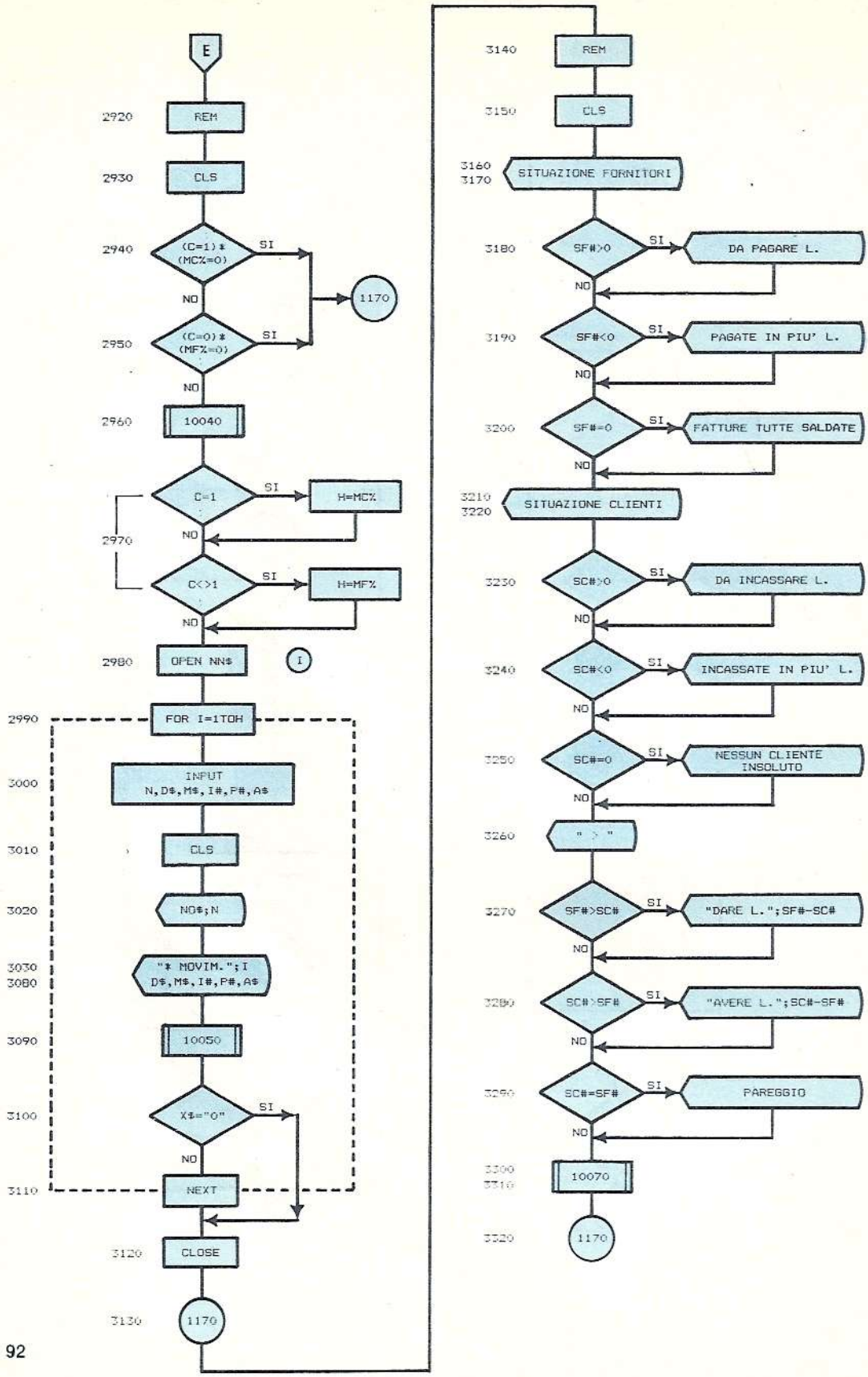

START
1000
1020
REM
1030
1050
CLEAR
ONERROR
CLS
1060
1070
GESTIONE
AZIENDALE
1080
10040
1090
1100
OPEN "DATI"
R
1110
1140
GET
SC#,NC%,MC%,
SF#,NF%,MF%
1150
CLOSE
1160
REM
1170
1180
RM=0 CR=0
CLS
1190
1230
MENU' 1
1240
M=5
1250
10010
1260
X=5
SI
3180
NO
1270
(X=1)+
(X=3)
SI
C=1
NO$="CLIENTE"
NN$="MOVCL"
NO
1280
(X=2)+
(X=4)
SI
C=0
NO$="FORNITORE"
NN$="MOVFO"
NO
1290
X=1
SI
1310
NO
X=2
SI
NO
X=3
SI
2620
NO
X=4
SI
1300
REM
1310
CLS
1320
1350
MENU' 2
1360
M=4
1370
10010
1380
X=1
SI
2220
NO
X=2
SI
1400
NO
X=3
SI
1980
NO
X=4
SI
1170
A

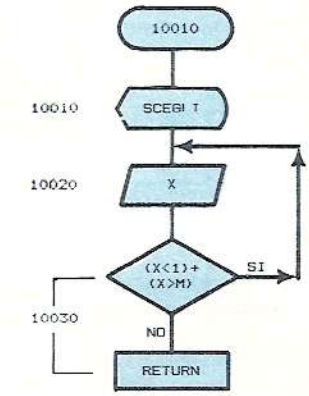

A
1390
REM
1400
TX$="NOME"
Q=30
1410
10110
1420
N$
1430
LEN(N$)>30
SI
NO
1440
LEN(N$)<30
SI
N$=N$+" "
NO
1450
TX$="VIA"
1460
10110
1470
V$
1480
LEN(V$)>30
SI
NO
1490
LEN(V$)<30
SI
V$=V$+" "
NO
1500
TX$="CITTA'"
1510
10110
1520
C$
1530
LEN(C$)>30
SI
NO
1540
LEN(C$)<30
SI
C$=C$+" "
NO
1550
TX$="TELEFONO"
Q=20
1560
10110
1570
T$
1580
LEN(T$)>20
SI
NO
1590
LEN(T$)<20
SI
T$=T$+" "
NO
1600
TX$="ANNOTAZIONI"
Q=129
1610
10110
1620
R$
1630
LEN(R$)>126
SI
NO
1640
I#=0,P#=0
NO
1650
TX$="TOT. FATTURA"
Q=16
1660
10110
1670
I#
1680
TX$="ACCONTO"
1690
10110
1700
P#
1710
I$=N$+V$+C$+T$+R$
1720
CLS
B

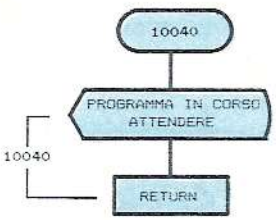

B
1730
10040
C=0
SI
NF%=NF%+1 PU=NF%
NO
1740
C<>0
SI
NC%=NC%+1 PU=NC%
NO
1750
(CR=1)+(RM=1)
SI
PU=N
NO
1760
OPEN NO$
R
1770
1790
PUT,PU
I#,P#,Z$
1800
1810
OPEN "DATI"
R
1820
SI
C=0
NO
RM=0
SI
SC#=SC#+I#-P#
NO
1830
RM<>0
SI
SC#=SC#+IM#-PM#
NO
1840
1880
PUT,1
SC#,NC%,MC%
RM=0
SI
SF#=SF#+I#-P#
NO
1890
RM<>0
SI
SF#=SF#+IM#-PM#
NO
1900
1940
PUT,2
SF#,NF%,MF%
1950
CLOSE
1960
RM=1
SI
2220
NO
1170
1970
REM
1980
1990
CR=1
CLS
2000
2010
QUALE NUMERO DI NO$
VUOI CORREGGERE ?
(0 PER TORNARE AL MENU')
2020
N
2030
N=0
SI
1170
NO
2040
SI
(C=0)*(N>NF%)
NO
2050
SI
(C=1)*(N>NC%)
2060
CLS
2070
10040
2080
2090
OPEN NO$
R
2100
GET N
2110
CLOSE
2120
CLS
2130
2140
NO$=LEFT$(Z$,30)
2150
10050
2160
X$="0"
SI
1170
NO
2170
CLS
2180
C=0
SI
NF%=NF%-1
NO
2190
C<>0
SI
NC%=NC%-1
NO
2200
1400
C

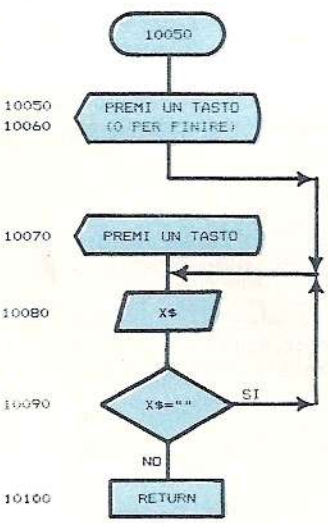

C
2210
REM
2220
2230
RM=1
IM#=0 PM#=0
CLS
2240
"NUMERO DEL";NO$
(0 PER FINIRE)
2250
N
2260
N=0
SI
1170
NO
2270
SI
(C=1)$
(N>NCX)
2280
SI
(C=0)$
(N>NFX)
NO
2290
CLS
2300
10040
2310
2320
OPEN NO$
R
2330
2340
GET,N
I#,P#
2350
CLS
2360
2370
NO$;"N.";N
LEFT$(Z$,30)
2380
10050
2390
X$="0"
SI
1170
NO
2400
2410
D$
2420
M$
2430
LEN(M$)>75
SI
NO
2440
IM#
2450
PM#
2460
A$
2470
LEN(A$)>63
SI
2480
CONFERMI (S/N) ?
2490
X$
2500
2510
(X$="N")+
(X$="S")
SI
NO
2520
X$="N"
SI
CLOSE
2220
2530
CLS
2540
10040
2550
OPEN NN$
E
2560
PRINT
N,D$,M$,IM#,PM#,A$
2570
CLOSE
2580
I#=I#+IM#
P#=P#+PM#
C=1
SI
MCX=MCX+1
NO
2590
C<>1
SI
MFX=MFX+1
NO
2600
1750
D

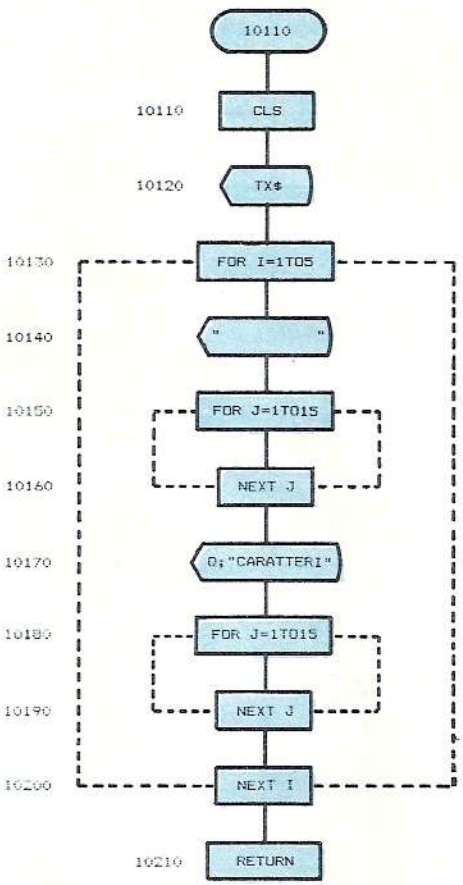

D
2610
REM
2620
CLS
2630
2650
MENU' 3
2660
M=3
2670
10010
X=1
SI
2950
NO
2680
X=2
SI
2700
NO
X=3
SI
1170
NO
2690
REM
2700
CLS
2710
(C=1) ¤ (NC%=0)
SI
1170
NO
2720
(C=0) ¤ (NF%=0)
SI
NO
2730
10040
C=1
SI
H=NC%
NO
2740
C<>1
SI
H=NF%
NO
2750
2760
OPEN ND$
R
2770
FOR I=1TOH
2780
GET I
2790
CLS
2800
2860
ND$;"NUMERO";I
N$,V$,C$,T$,A$,I#,F#
2870
10050
2880
X$="0"
SI
NO
2890
NEXT
2900
CLOSE
2910
1170
E

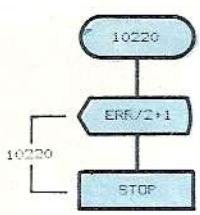

E
2920
REM
2930
CLS
2940
(C=1)*(MC%=0)
SI
NO
1170
2950
(C=0)*(MF%=0)
SI
NO
2960
10040
2970
C=1
SI
H=MC%
NO
C<>1
SI
H=MF%
NO
2980
OPEN NN$
I
2990
FOR I=1TOH
3000
INPUT N,D$,M$,I#,P#,A$
3010
CLS
3020
NO$;N
3030 3080
"* MOVIM.";I D$,M$,I#,P#,A$
3090
10050
3100
X$="0"
SI
NO
3110
NEXT
3120
CLOSE
3130
1170
3140
REM
3150
CLS
3160 3170
SITUAZIONE FORNITORI
3180
SF#>0
SI
DA PAGARE L.
NO
3190
SF#<0
SI
PAGATE IN PIU' L.
NO
3200
SF#=0
SI
FATTURE TUTTE SALDATE
NO
3210 3220
SITUAZIONE CLIENTI
3230
SC#>0
SI
DA INCASSARE L.
NO
3240
SC#<0
SI
INCASSATE IN PIU' L.
NO
3250
SC#=0
SI
NESSUN CLIENTE INSOLUTO
NO
3260
" > "
3270
SF#>SC#
SI
"DARE L.";SF#-SC#
NO
3280
SC#>SF#
SI
"AVERE L.";SC#-SF#
NO
3290
SC#=SF#
SI
PAREGGIO
NO
3300 3310
10070
3320
1170

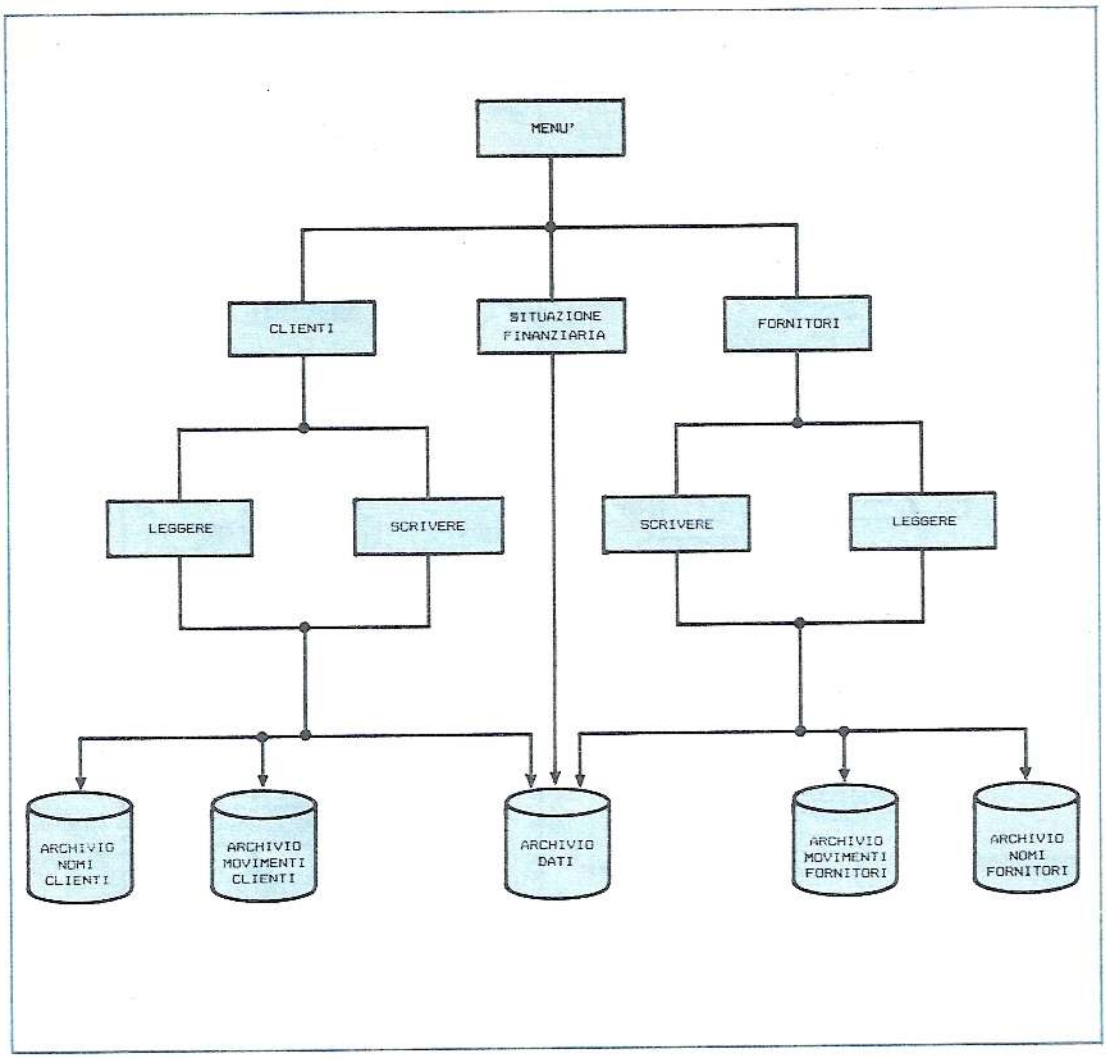

10010
10010
SCEGLI
10020
X
(X<1)+
(X>M)
SI
10030
NO
RETURN


10040
PROGRAMMA IN CORSO
ATTENDERE
10040
RETURN


10050
10050
10060
PREMI UN TASTO
(0 PER FINIRE)
10070
PREMI UN TASTO
10080
X$
10090
X$=""
SI
NO
10100
RETURN


10110
10110
CLS
10120
TX$
10130
FOR I=1TO5
10140
" "
10150
FOR J=1TO15
10160
NEXT J
10170
0;"CARATTERI"
10180
FOR J=1TO15
10190
NEXT J
10200
NEXT I
10210
RETURN


10220
ERR/2+1
10220
STOP



1000-1020

Innanzitutto vedete che il programma inizia col numero di linea 1000: ciò ha unicamente lo scopo di fornire un listato tutto perfettamente marginato a sinistra, cosa che ne facilita la lettura. A parte questo, il numero d'inizio può essere uno qualunque.

Le prime tre linee contengono solo delle annotazioni REM (abbreviate con l'apostrofo). È sempre bene mettere un'intestazione ad ogni programma: in tal modo sia l'estensore del medesimo che chiunque altra persona possono capire immediatamente qual'è l'argomento trattato. Rammentiamo che il computer ignora il contenuto di tali linee, saltandole durante l'esecuzione.

1030

Questa riga serve per dichiarare lo spazio necessario alle stringhe. Infatti i 50 caratteri assegnati automaticamente dal computer non sono sufficienti. Il motivo per cui si è assegnato il numero 1000 potrà sembrarvi oscuro, visto che non facciamo uso di vettori. La giustificazione risiede nel fatto che esistono dei LOOP di lettura degli archivi creati sul dischetto; quando li incontra, il computer esegue la lettura di più elementi e li memorizza per presentarceli successivamente. Il numero di elementi letti non è fisso, quindi è

difficile dire quanti caratteri dobbiamo rlservare alle stringhe. Col valore 1000 siamo al sicuro, ma anche valori un po' più bassi dovrebbero andare bene. Se avete problemi di memoria potete mettere un valore inferiore; in questo caso, durante lo svolgimento del programma, il calcolatore va in errore se non si ritrova spazio sufficiente per memorizzare le stringhe.

Nella stessa linea si ha poi la definizione della variabile di stringa Y$: essa viene posta uguale al carattere corrispondente al codice ASCII 34, ossia alle virgolette. Vedremo a suo tempo a cosa serve Y$ (linea 2560).

1040

L'istruzione ONERRORGOTO svolge un importante controllo. Pensate infatti che mano a mano che aggiungiamo dati ai nostri archivi, il posto vuoto sul disco si riduce sempre più; arriva il momento in cui il dischetto risulta pieno, e si avrebbe la segnalazione d'errore DISK FULL (codice d'errore 62).

Con la routine di gestione d'errore ONERRORGOTO si potrebbe controllare il proseguimento del programma anche in tal caso, senza dover uscire da esso. In effetti, per i soliti motivi di spazio, ci siamo limitati, alla linea 3650, a fare uno STOP, fornendo la stampa del codice d'errore che si è verificato. Quando arriveremo alla linea 3650 vi daremo comunque qualche indicazione sul modo di gestire alcuni errori senza uscire dal programma.

Se darete ascolto alle nostre raccomandazioni, introdurrete il programma nel vostro computer, scrivendo una riga per volta. Inevitabilmente commetterete qualche errore di battitura, che si tradurranno in altrettanti errori di sintassi. Succede allora che quando farete partire il programma, lo stesso si fermerà sempre alla linea 3650 a causa della presenza di ONERRORGOTO 3650.

Vi consigliamo pertanto di mettere tale istruzione solo alla fine, quando la stesura del programma risulterà esente da errori. La funzione di ONERRORGOTO è quella di gestire errori esterni al programma, come errori nell'introduzione di dati da tastiera, oppure il verificarsi dell'errore menzionato poco fa.

Per ultimo vi diciamo di non fare mai ONERRORGOTO prima di una CLEAR: questa ultima istruzione cancella ONERROGOTO, che risulta così come non assegnato.

1050-1070

Prima di tutto vi ricordiamo che gli spazi vuoti (in inglese BLANK = spazio bianco) non sono necessari al funzionamento di un programma. Noi li abbiamo messi per rendere molto più agevole la lettura del listato, ma voi potete sopprimerli tutti: così facendo risparmierete anche memoria. Considerando che abbiamo circa 260 linee di programa e pensando ad una media di 3 blanks per ognuna di esse, si ottengono più o meno 750 caratteri risparmiati.

Gli unici casi in cui non abbiamo messo gli spazi sono quelli che riguardano le parole chiave REVON e REVOFF; abbiamo già detto, in una precedente occasione, che il computer segnala sempre errore se tra REVON ed i due punti successivi c'è uno spazio; tale comportamento si verifica talvolta anche con REVOFF, quindi è bene che anche voi prendiate l'abitudine di non mettere mai blanks dopo queste due parole chiave.

Andando con ordine, la linea 1050 fa cancellare lo schermo del monitor. La 1060 innesca la scrittura in negativo (REVERSE ON). La successiva istruzione di stampa ordina al computer di scrivere la frase tra virgolette ad iniziare dal carattere 166 della mappa video, cioè dal settimo carattere della sesta riga. In mancanza della virgola prima delle virgolette, si ha una segnalazione d'errore.

REVOFF fa tornare ad una scrittura normale (REVERSE OFF).

Con la linea 1070 si ottiene l'avanzamento di tre linee di monitor, senza effettuare alcuna stampa.

1080 E SUBROUTINE 10040

Il programma spesso effettua delle letture o delle registrazioni su disco; in quei casi il monitor continuerebbe a presentare le scritte precedenti, e l'operatore non saprebbe bene cosa stia succedendo. Meglio quindi scrivere sul video che il programma sta proseguendo e di portare un attimo di pazienza. Questa è proprio la funzione svolta dalla SUBROUTINE 10040. Subroutine sta ad indicare che tale parte di programma termina con l'istruzione RETURN: quando la incontra, il computer fa tornare il programma alla istruzione successiva alla GOSUB che aveva causato la chiamata alla subroutine stessa.

Il motivo per cui si ricorre all'uso di subroutines è abbastanza intuitivo: quando occorre eseguire più volte la stessa sequenza di istruzioni, anzichè riscriversele torna comodo richiamare ogni volta quella parte di programma che le contiene. Si risparmiano così righe e memoria, e si evitano possibilità di errori. La contropartita consiste in un piccolo rallentamento nell'esecuzione del programma, ma ciò generalmente non si avverte e non disturba.

Nel nostro caso la subroutine che inizia alla linea 10040 provoca la scritta della dicitura "PROGRAMMA IN CORSO - ATTENDERE". La linea termina poi con l'istruzione RETURN che causa il ritorno all'istruzione successiva alla GOSUB di partenza. Da notare che abbiamo detto "istruzione successiva" e non "linea successiva": in caso di linea contenente più di una istruzione, con GOSUB in mezzo ad esse, il programma ritorna subito dopo al GOSUB e non alla linea seguente.

1090

Fino ad ora, ricapitolando, il programma ha riservato il posto alle stringhe, ha cancellato lo schermo ed ha scritto le frasi anzidette. Ora il calcolatore deve andarsi a leggere quei dati che sono veramente indispensabili per un corretto funzionamento del tutto. Dato che non vogliamo creare vettori, dovremo leggere pochi dati, e precisamente il numero dei clienti memorizzati, il numero dei movimenti riguardanti i clienti e la situazione generale dei pagamenti per i clienti; altrettanto per i fornitori. In tutto quindi ci servono sei numeri.

Dato che questi numeri variano in continuazione, meglio porli in un file random, così potremo registrare i loro valori aggiornati senza riscriverli tutti.

I FILES RANDOM POSSONO ESSERE REGISTRATI SOLO A BLOCCHI DI 255 BYTES PER VOLTA (vale a dire solo una traccia-settore per volta). Inoltre ricordare anche che IL MINIMO SPAZIO RISERVATO AD UN FILE, SIA ESSO SEQUENZIALE CHE RANDOM, È SEMPRE DI UN GRANULO (1 granulo = 5 tracce-settore).

Di questo abbiamo già parlato alla voce FREE dei comandi DOS.

Andiamo avanti; i numeri che esprimono i clienti ed i fornitori, nonchè il totale dei movimenti sui due conti, sono sempre numeri interi: avremo, ad esempio, 152 clienti, 2 fornitori, 305 movimenti di merce o denaro sul conto clienti e 129 movimenti sul conto fornitori. Tanto vale allora fare uso, per questi numeri, di variabili intere. Esse sono identificate dal simbolo % ed occupano meno memoria degli altri tipi di variabili.

I numeri che esprimono la situazione finanziaria dei due conti è meglio invece depositarli in variabili in doppia precisione: con sei cifre solo, infatti, non potremmo trattare importi superiori a lire 999.999, che è decisamente poco per un'attività commerciale che si rispetti!

Ricordate che lavorare con sei cifre, in realtà non significa affatto non poter superare il limite suddetto, ma bensì che quando un numero supera quella lunghezza viene automaticamente rappresentato con notazione in virgola mobile. Ad esempio, il numero 106388732 diventa 1.06389E + 08 (cioè 1,06... moltiplicato per 10 elevato alla ottava: esso vale ora 106389000). È evidente che non possiamo nè presentare numeri in virgola mobile, né perdere delle cifre significative.

I numeri in doppia precisione sono contraddistinti dal simbolo # e sono formati da 16 cifre, tutte esatte. Anche per questi numeri vale il discorso della virgola mobile, ma occorre superare il numero 9.999.999.999.999.999. Se la vostra attività commerciale supera questa cifra, potete ben permettervi un computer con una maggior capacità di calcolo!

NEI FILES RANDOM POSSONO ESSERE INCISE SOLO STRINGHE E NON NUMERI.

Pertanto esistono delle istruzioni apposite per trasformare i numeri in stringhe e viceversa. Se pensate poi che abbiamo tre tipi diversi di numeri, in totale avremo sei funzioni per la trasformazione suddetta.

MKI$, MKS$, MKD$ servono rispettivamente per trasformare numeri interi, in singola precisione ed in doppia precisione in stringhe. Inversamente, CVI CVS e CVD servono per riottenere numeri interi, in singola precisione ed in doppia precisione dopo averli letti, sotto forma di stringa, da un file random.

Dopo aver subito la trasformazione in stringhe, i numeri interi occupano due bytes, quelli in singola precisione quattro bytes, quelli in doppia precisione otto bytes.

Detto questo torniamo al nostro programma. Evidentemente non avrebbe molto senso occupare un RECORD (ossia 255 bytes) mettendovi solo un numero, di qualunque tipo esso sia: infatti sprecheremmo molto spazio, almeno 247 bytes su 255. Lo stesso discorso vale anche per le stringhe: incidere in un record solo un nome, ad esempio GABRIELE BURBASSI (17 caratteri), significa sprecare 238 caratteri. Allora si procede come segue.

I 255 caratteri di un record vanno suddivisi in CAMPI, ossia in porzioni parziali la cui somma, al massimo, deve fare appunto 255. Ad ogni porzione corrisponde una stringa.

Cominciamo a tirare le fila del discorso. Per leggere o registrare un file occorre innanzitutto aprire un canale di comunicazione tra memoria del computer e disco: tutti i dati scambiati da queste due entità passano attraverso di esso. Tale via di comunicazione si chiama BUFFER; altro non è che una parte di memoria riservata a questo uso. Se non viene diversamente specificato all'atto del caricamento del BASIC, il computer riserva tre buffers per lo scambio di dati coi dischi; perciò non possiamo tenere aperti più di tre files contemporaneamente. La cosa generalmente non pone problemi; ad ogni buon conto vedremo come si fa per aumentare il numero dei buffers.

La parola chiave che serve per aprire un buffer di comunicazione è OPEN (= apri) e deve essere seguita da una lettera posta tra virgolette. Se la lettera usata è R ordiniamo di aprire un file random, e tale istruzione vale sia per effettuare letture che registrazioni. Questo è un fatto molto importante, da tenere bene a mente: con un unico tipo di apertura, un file di tipo random può essere indifferentemente letto o scritto, o entrambe le cose contemporaneamente. Se invece vogliamo trattare un file sequenziale, abbiamo tre possibilità diverse: per leggerlo dobbiamo usare la lettera I (INPUT = fa entrare), per scriverlo dall'inizio la lettera O (OUTPUT = fa uscire), per registrare dati in coda a quelli già esistenti la lettera E (EXTEND = estendi). Risulta pertanto evidente che un file sequenziale necessita di aperture diverse in lettura o in registrazione; tali due operazioni non possono mai avvenire assieme.

Dopo aver specificato il tipo di file che si intende aprire, occorre assegnare il buffer di comunicazione; ciò si ottiene mettendo una virgola dopo le virgolette e specificando un numero tra uno e tre. Bisogna poi mettere un'altra virgola e far seguire un nome tra virgolette. Tale nome è quello che identifica il file sul disco. Infatti su di esso esisteranno diversi files, e per usare quello voluto dobbiamo chiamarlo col suo nome. Esso deve iniziare con una lettera e può contenere anche numeri; se contiene spazi o altri caratteri, il nome viene memorizzato solo fino al carattere non ammesso escluso. Inoltre la sua lunghezza non deve superare gli otto caratteri; quelli in soprannumero vengono ignorati.

Tale nome può essere dato tra virgolette, come abbiamo già detto, ma anche sotto forma di variabile di stringa. Ecco qualche esempio di apertura di files:

```
OPEN " O " , 1 , "LUCA"
OPEN"E",3,"ASD12"
OPEN"i",2,"telefono"
OPEN "O",2,G$
OPEN"R",3,"FRANCESCA"
```

Gli spazi sono facoltativi; nell'ultimo caso il nome assegnato è FRANCESC, di otto caratteri. Nel quarto esempio, se la variabile G$ è vuota, ossia se non è formata da alcun carattere, il computer segnala errore; il nome deve contenere almeno una lettera. Le lettere che identificano i tipi di files ed i nomi ad essi assegnati possono essere scritti indifferentemente in minuscolo o maiuscolo.

Torniamo ora al nostro programma. La linea 1090 dovrebbe essere chiara: in essa viene aperto un file random di nome DATI, passando attraverso il buffer 1.

1100

Dopo aver aperto il file random DATI, occorre dare l'istruzione di FIELD; essa è indispensabile sia che intendiamo effettuare letture che registrazioni. Nel caso in esame, come vedremo all'istruzione successiva, vogliamo fare una lettura di dati. Essi, lo ripetiamo ancora una volta, sono esclusivamente sotto forma di stringhe. Ogni lettura preleva l'intero contenuto di un record (255 bytes); questa stringa così lunga, all'atto dell'incisione, sarà stata ottenuta attaccando assieme più dati, come vedremo nella parte di programma che tratta della registrazione di questo file. La dichiarazione di field della linea 1100 ricalca quella della registrazione (riga 1810), e serve per ricostruire i vari dati partendo dall'unica stringa in cui sono contenuti, separandoli gli uni dagli altri nel modo indicato da FIELD. Infatti questa istruzione dice che i primi 8 caratteri di quei 255 vanno depositati nella variabile S$, i successivi 2 nella variabile N$, altri 2 in M$, ed i rimanenti 243 in A$.

Ci preme ribadire il fatto che tale suddivisione rispetta lo schema della registrazione; talvolta, comunque, la lettura può essere fatta anche con schemi di campo diversi da quello impiegato in incisione. Ad esempio, se avevamo registrato il numero di telefono di alcuni nominativi mettendo prima il prefisso e poi il numero vero e proprio in un'unica stringa di 12 caratteri, nulla ci vieta di andare a leggere solo le prime quattro cifre invece delle 12 totali: potremmo infatti desiderare di avere tutti i telefoni di una certa città, quindi la lettura delle prime quattro cifre è quella che serve al nostro scopo.

Vedremo tra breve il senso della dichiarazione di campo effettuata. Vi facciamo notare che gli spazi sono facoltativi. A questo proposito, occorre prestare attenzione se si usa una variabile di stringa che

cominci con la lettera C; eliminando gli spazi otterreste, ad esempio, 8ASC$: il computer segnalerebbe errore, perché la dicitura ASC corrisponde ad una parola chiave del BASIC. Se volete usare la variabile C$ dovete ricordarvi di mettere lo spazio dopo AS.

1110

Questa linea contiene l'istruzione che effettua la lettura di un record: GET1,1 significa infatti che il calcolatore deve prelevare dal file random DATI, cui è stato assegnato il buffer 1 (questo è il senso del primo numero dopo GET), il record che si trova al primo posto. Infatti, avendo a che fare con un file ad accesso casuale, dobbiamo specificare quale elemento vogliamo leggere. Se non mettiamo questo numero, il computer va a leggere ugualmente il primo dato; noi lo abbiamo messo ugualmente per darvi il formato completo di GET. Ogni GET successivo, se non viene specificato l'elemento desiderato, causa la lettura del record che segue l'ultimo letto.

Con GET abbiamo ottenuto una stringa di 255 caratteri, da suddividere nel modo indicato da FIELD.

1120

In questa linea, formata da tre istruzioni, si vede l'uso che intendiamo fare di S$, N$ e M$. Andando con ordine, troviamo innanzitutto SC# = CVD(S$): nella variabile numerica in doppia precisione SC# viene depositato il numero che in registrazione era stato associato ai primi otto caratteri del primo record. Tale numero, prima dell'incisione, era stato trasformato in stringa con l'istruzione MKD$, e ora deve subire il processo inverso. La funzione di CVD è appunto quella di prendere gli otto caratteri di S$ e di trasformarli in un numero in doppia precisione.

Proseguendo nell'esame della riga 1120, troviamo poi NC% = CVI(N$).

Analogamente al caso appena visto, si tratta della trasformazione della stringa N$, di 2 caratteri, in un numero intero che viene depositato nella variabile intera NC%. Nello stesso modo, con l'istruzione successiva, alla variabile MC% viene assegnato il numero intero che risulta decodificando la stringa di due caratteri M$.

Il tutto, se volete, può sembrare un po' macchinoso, e forse lo è; tuttavia si tratta di un metodo escogitato per sfruttare al massimo la capacità di immagazzinare dati nel dischetto, tenendo conto anche dei tempi necessari per registrare e prelevare dati. Del resto tale metodo è ampiamente collaudato, essendo applicato da parecchio tempo. Anche computers recentissimi, come i personal IBM ed OLIVETTI, adottano tale procedura.

1130-1150

Le linee 1130 e 1140 sono perfettamente equivalenti, rispettivamente, alle 1110 e 1120: con esse si legge il secondo record, cioè altri 255 caratteri, ed i valori numerici decodificati da CVD e CVI vengono depositati in SF# , NF% e MF%.

La linea 1150 serve per chiudere il buffer di comunicazione col file: lo scambio di dati tra disco e computer è per ora terminato.

Facciamo un breve riassunto del gruppo di linee che vanno dalla 1090 alla 1150. Prima apriamo il file random DATI, poi dichiariamo la suddivisione dei 255 caratteri che verranno letti con ogni GET La prima di esse ci fornisce tre numeri; il primo, in doppia precisione, esprime il saldo dei clienti (SC#), ossia l'ammontare totale dei crediti che vantiamo nei loro confronti. Il secondo numero è intero ed indica il numero dei clienti (NC%) registrati nel relativo archivio. Il terzo numero MC% indica il numero dei movimenti (di merce o di denaro) che sono registrati come vedremo in un apposito archivio.

Qualche considerazione ancora. Innanzitutto vi sarete chiesti che fine abbiano fatto i rimanenti 243 caratteri di A$ ottenuti da ogni record letto. In effetti li abbiamo definiti in FIELD ma non li abbiamo utilizzati. La cosa è voluta, in quanto intendiamo ampliare in futuro questo programma, aggiungendo anche la parte che riguarda la gestione del magazzino. Per poterlo fare ci serviranno altri dati, che andremo a registrare proprio in quello spazio, che per ora rimane inutilizzato. La procedura adottata ci sembra poi molto indicativa, nel senso che ogniqualvolta si crea un archivio di dati è bene prevedere una

sua possibile espansione futura: quello che ci sembra sufficiente oggi potrebbe non esserlo più domani, quindi è saggio cautelarsi in vista di probabili aggiunte.

Le linee 1110 e 1130 potevano essere scritte GET1, come abbiamo già detto. La chiusura poteva essere fatta anche scrivendo CLOSE1, richiamando cioè il numero del buffer interessato. In questo caso le due chiusure si equivalgono, ma qualora avessimo aperto più di un file e volessimo chiuderne uno solo, il modo giusto per farlo sarebbe stato il secondo qui indicato. Quando si è in presenza di diversi files aperti, la chiusura fatta semplicemente con CLOSE ha effetto su tutti.

È il caso di esaminare, ora, cosa succede nelle righe 1090-1150 se il file DATI non esiste ancora; infatti ciò è quello che succede quando il programma non ha ancora girato per la prima volta. Avviene quanto segue: il computer va a cercare sul primo floppy il file DATI; non trovandolo, effettua tale ricerca su tutti i floppy che esistono, senza trovare alcunchè. Allora va sul primo floppy che non porta la protezione contro le registrazioni, e crea il file in questione; ricordate, in proposito, che l'apertura di un file random è la stessa sia per incidere che per leggere dati.

Sucessivamente il calcolatore incontra due GET; dato che il file DATI ora esiste, ma è vuoto, le stringhe che dovevano essere lette sono nulle. Pertanto i sei valori numerici ad esse associati valgono automaticamente zero. I conti perciò tornano, ed il computer non segnala errore. Se invece non c'è un dischetto privo di protezione, il controllo del programma passa all'istruzione ONERRORGOTO 10220: otterremmo il numero d'errore 52 (INTERNAL ERROR), che indica appunto tale evento.

Se avete messo in atto il suggerimento di seguire le spiegazioni, oltre che sul listato del programma, anche sul diagramma di flusso relativo a questa parte (figura 4), non dovreste aver trovato troppe difficoltà ad interpretarlo correttamente. Se provate, a questo punto, a riguardare il listato e il relativo diagramma di flusso, converrete che quest'ultimo è estremamente più immediato ed indicativo: l'occhio può correre veloce e trasmetterci così il filo logico del programma.

1160-1230

Le operazioni fin qui svolte sono le uniche necessarie ad un corretto funzionamento del programma: i pochi numeri letti dal disco sono gli unici dati caricati in memoria permanentemente. Essi verranno eventualmente aggiornati, nel caso si effettuino delle registrazioni di nuovi nominativi o movimenti.

Il programma prevede, a questo punto, diverse possibilità: la scelta è offerta dal MENU', ossia dall'elenco delle diverse vie che possiamo scegliere. La parola MENU' è diventata oramai d'uso comune, perché in realtà esprime molto bene il concetto suddetto.

La linea 1160 è un'altra riga di annotazioni: come vedete, ne abbiamo inserite diverse, in modo da semplificare la lettura del listato. Tali REM sono state scritte in modo particolare; i vari menù (ce ne sono tre) sono contrassegnati da una lunga fila di + mentre le altre parti hanno inizio alle linee marcate col simboli —. Questa scelta, ovviamente, è arbitraria, ma ci sembra efficace: l'ordine è un obiettivo importante anche in un listato, ed è indice di chiarezza d'idee e di professionalità.

Nella riga 1170 le variabili RM e CR vengono azzerate: lo scopo di questa operazione verrà chiarito al momento opportuno.

La CLS che compare alla linea 1180 cancella lo schermo del monitor. Segue poi la stampa in negativo (REVERSE) della scritta MENU', effettuata a cominciare dal dodicesimo carattere della prima riga del monitor (il primo carattere porta il numero zero). La linea 1190 fa lasciare due righe (PRINT:PRINT), poi fa stampare, sempre in reverse, il numero 1. Subito dopo si torna alla scrittura in positivo, grazie all'istruzione REVOFF; la frase compresa tra virgolette fa seguito al numero 1.

Le quattro righe che seguono servono per scrivere il menù; ognuna di esse inizia con PRINT, ossia lasciando una linea vuota prima di stampare quello che segue. C'è poi REVON, che fa passare alla scrittura in negativo, e l'ordine di stampare il numero 2 (oppure 3, 4 o 5, nelle varie linee). Con REVOFF si torna poi ad una scrittura in positivo della frase successiva. Da notare che quelle frasi vengono scritte di seguito ai numeri corrispondenti perché, ad esempio, PRINT"1" è seguito dal punto e virgola. Questo indica appunto al computer che le stampe successive devono essere attaccate alle precedenti; senza quei punti e virgola le frasi verrebbero visualizzate a capo della linea successiva a quella del numero in reverse.

Vi ricordiamo per l'ultima volta la necessità di non mettere spazi vuoti dopo REVON e REVOFF.

1240-1250 E SUBROUTINE 10010

Per capire il senso della linea 1240 occorre prima vedere il contenuto della subroutine richiamata nella riga 1250. Andate quindi al numero di linea 10010; lì c'è l'ordine di scrivere SCEGLI, ad iniziare dal

carattere video numero 448 (ossia dall'inizio della penultima riga, come si vede dalla mappa video pubblicata a pag.

La linea 10020 significa "fa entrare una stringa di un carattere dalla tastiera (INKEY$); calcola il valore numerico di quella stringa (VAL(INKEY$)) e depositalo nella variabile X".

Abbiamo già spiegato, nella descrizione delle varie parole chiave nel BASIC, il funzionamento dl INKEY$; quando la incontra, il computer si predispone ad accettare una carattere da tastiera. Qualora non venga premuto alcun tasto, il calcolatore assume automaticamente una stringa nulla (nessun carattere).

La linea 10030 dice: "se X è minore di uno oppure maggiore di M, allora vai alla linea 10020; altrimenti ritorna". Ecco allora a cosa serve M (= 5 nel caso attuale); in pratica, fino a che non premiamo alcun tasto, VAL(INKEY$) ossia X, vale zero; dato che tale valore è minore di uno, il programma va alla linea 10020, cioè il computer ricomincia il ciclo. In altre parole, fino a che non si effettua una battuta, il programma resta intrappolato in quel circolo vizioso. Tale comportamento si ha anche se il tasto premuto è diverso da un numero: infatti il valore di una lettera o di un simbolo è zero. Anche se il numero battuto è superiore a 5 il programma resta fermo, come ordina la riga 10030. SOLO LA PRESSIONE DI UN TASTO NUMERICO DA UNO A CINQUE FA AVANZARE IL PROGRAMMA; LA PRESSIONE DI QUALSIASI ALTRO TASTO RISULTA INEFFICACE.

Per inciso, facciamo notare che il programma riparte automaticamente non appena viene premuto uno dei tasti ammessi: non c'è alcun bisogno di digitare anche il tasto RETURN.

Dato che questa stessa subroutine viene richiamata altre due volte per i menù che troveremo nel corso del programma, e poichè ogni menù ha un certo numero di possibilità di scelta, tale valore non può essere fisso; esso viene dato con la variabile M prima di entrare nella subroutine. Ad esempio, se tale parametro viene posto uguale a 2, solo la pressione dei tasti 1 o 2 fa avanzare il programma. Pur con questi diversi valori, la subroutine è unica.

Il valore di M deve, in ogni caso, essere minore di 10, poichè L'ISTRUZIONE INKEY$ INTROITA UN SOLO CARATTERE: se scrivete 10, in effetti unicamente la prima battuta è quella analizzata, quindi è come se aveste premuto il tasto del numero uno.

1260

Quando il programma, nella subroutine che inizia alla linea 10010, trova l'istruzione RETURN, il controllo passa all'istruzione successiva alla GOSUB che aveva richiamato la subroutine stessa. Quindi la prima istruzione incontrata dopo il rientro è quella contenuta nella linea 1260: "se X è uguale a 5 va alla riga 3150". Dal menù si vede che la scelta numero 5 fornisce la situazione dei pagamenti; per farlo il computer non ha bisogno di acquisire altri dati, quindi si passa direttamente al numero di linea indicato, dove ha inizio la parte di programma che visualizza appunto la situazione finanziaria. Noi procederemo con ordine lungo il listato, pertanto tratteremo più avanti questo argomento; anticipiamo solo che si tratta di visualizzare gli importi SC# e SF# già letti.

Seguendo il diagramma di flusso relativo a questa porzione di programma, vedete che all'istruzione IF corrisponde il simbolo grafico di un rombo; visivamente ci si rende quindi conto immediatamente dei punti in cui esistono test relazionali.

1270-1280

Dal diagramma di flusso di figura 5 si vede che d'ora in poi questa parte di programma contiene solo Istruzioni di confronto. All'interno del rombo che rappresenta la linea 1270 troviamo la scritta:

```
(X=1)+(X=3)
```

È il modo più breve ed intuitivo di rappresentare una istruzione IF...OR...THEN, che si chiama anche SOMMA LOGICA: ecco spiegato il segno + .

Il contenuto della linea 1270 è il seguente: "se X vale 1 oppure 3, allora C vale 1, la stringa NO$ è uguale a CLIENTE e la NN$ è uguale a MOVCL" cioè MOVIMENTO CLIENTI.

Non spaventatevi, perché tutto è presto spiegato. Nel prosieguo del programma dobbiamo spesso controllare se stiamo trattando clienti oppure fornitori; per evitare di fare parti di programma riservate esclusivamente ad ognuno dei due gruppi di nominativi, è comodo usare una variabile di controllo (appunto C) che ci illumina su quella condizione. Infatti nelle linee in esame poniamo C = 1 se la scelta fatta sul menù coinvolge i clienti (possibiltà 1 e 3), oppure C = 0 se si tratta di fornitori (scelte 2 e 4).

Il tutto, ripetiamo è fatto per effettuare le letture e le registrazioni dei files clienti e fornitori con una sola parte di programma in grado di trattarli entrambi.

Negli esempi visti di apertura di files, vi abbiamo fatto notare che il nome può essere dato anche sotto forma di variabile di stringa; ecco allora spiegato il motivo delle definizioni fatte per NO$ ed NN$, nelle due righe che stiamo esaminando. Se trattiamo clienti il file dei nominativi si chiama CLIENTE e quello dei movimenti si chiama MOVCL; coi fornitori gli stessi files prendono i nomi FORNITORE e MOVFO cioè MOVIMENTO FORNITORI. è superfluo far notare che è molto comodo l'uso di parole che si ricollegano alle funzioni svolte, in modo da riconoscere d'acchito cosa si sta facendo.

Notate anche che il nome FORNITORE è di nove caratteri: il relativo file, in effetti, verrà registrato su disco col nome FORNITOR, lungo otto. Ci fa comodo porre FORNITORE perché tale stringa viene usata anche per messaggi stampati sul monitor: scrivere FORNITOR MARIO ROSSI è forse poetico, ma poco commerciale!

Ognuna delle linee 1260, 1270 e 1280 viene eseguita solo se la relativa IF è soddisfatta; in caso contrario il programma prosegue alla riga successiva.

Secondo la teoria, infine, qualora X assuma valori diversi da quelli indicati, il computer passerebbe sempre alla linea 1300; ciò in pratica non succede mai perché la subroutine 10010 elimina automaticamente tale possibilità.

1290

Nella linea 1290 si impartisce al computer il seguente ordine: "se X vale 1 va alla linea 1310, se vale 2 va alla 1310, se vale 3 va alla 2620, se vale 4 va alla 2620".

Assieme alla riga 1260, che abbiamo già esaminato, costituisce il punto di smistamento alle varie parti dl programma, a seconda della scelta effettuata sul primo menù, che d'ora in poi chiameremo MENÙ PRINCIPALE (per distinguerlo dai due successivi).

Dai numeri di linea in cui vengono fatti i rinvii, si vede che le parti di programma che trattano le registrazioni e le letture sono comuni a clienti e fornitori, come avevamo preannunciato poco fa.

Il modo migliore per rappresentare in un diagramma di flusso l'effetto dell'istruzione ON...GOTO è quello indicato: ad ogni rinvio corrisponde un rombo.

Con questa riga termina l'esame del secondo blocco di istruzioni. Passiamo ora all'esame della parte che segue, il cui diagramma di flusso è riportato in figura 6. Non mancate di seguirlo.

1300-1380

Questo blocco di istruzioni serve per scrivere il primo menù secondario; esso gestisce la sezione dl registrazione dei dati.

Non staremo a descrivere le linee in dettaglio, perché dovremmo ripetere gran parte di quello che è stato detto per il menù principale. Diciamo solo che le possibilità di scelta ora sono quattro; quindi tale valore è assegnato alla variabile M che controlla poi, nella subroutine 10010, l'acquisizione della variabile X.

In questo menù l'istruzione ON...GOTO rinvia a linee diverse, in quanto sono differenti le funzioni da espletare.

Notare che c'è la possibilità (4) di tornare al menù principale; in caso contrario, se avessimo effettuato una scelta errata al primo menù, non potremmo più tornare sui nostri passi, ma saremmo costretti a proseguire su quella strada.

1390

Inizia con questa linea la serie di istruzioni che servono per registrare su disco i nomi dei nuovi clienti o dei nuovi fornitori.

Se non avete ancora dato uno sguardo d'assieme al programma, vi invitiamo a farlo adesso; dovreste rendervi conto che la successione dei vari blocchi segue un filo logico ben preciso. Infatti, quando si usa Il programma per la prima volta, la prima cosa da fare è quella di registrare nominativi di clienti e fornitori. Senza di essi non potremo registrare movimenti, in quanto fanno riferimento a quei nomi.

Per questo motivo troviamo prima la registrazione dei nominativi, poi la loro correzione, quindi la registrazione dei movimenti. Alla fine ci sono le parti che riguardano le varie letture e quella che dà la situazione dei pagamenti.

1400-1700 E SUBROUTINE 10110

Alla riga 1400 inizia una delle parti più lunghe, almeno da spiegare. Tutto il blocco che arriva fino al numero di linea 1700 serve per l'introduzione dei dati che accompagnano i vari nominativi. Il ciclo che vedremo ora si riferisce sia ai clienti che ai fornitori.

Dalla linea 1400 alla 1630 abbiamo il susseguirsi di cinque moduli funzionanti nello stesso modo.

La variabile di stringa TX$ è posta, volta a volta, uguale a NOME, VIA, CITTÀ, TELEFONO, ANNOTAZIONI; notare che le relative stringhe sono tutte lunghe uguali, per avere un effetto estetico uniforme quando si farà la loro stampa in reverse nella subroutine 10110.

Prima di entrare in essa, occorre dare anche il valore ad una variabile di controllo, indicata con la lettera Q, destinata alla verifica della lunghezza delle stringhe introdotte.

La registrazione dei vari nominativi segue questo schema: ognuno di essi occupa un record di 255 bytes; di questi, 30 sono dedicati al nome e cognome (oppure alla ragione sociale, nel caso di ditte). Altri 30 caratteri per la via ed il numero, poi altri 30 per codice di avviamento postale, città, provincia e nazione; 20 posti sono riservati al telefono, e 129 alle annotazioni. Facendo i conti, restano ancora 16 caratteri che sono utilizzati per registrare gli importi che riguardano ogni nominativo, ossia il totale delle vendite (o degli acquisti) ed il totale degli acconti ricevuti o versati.

Tale suddivisione torna comoda perché ogni nome ingombra un record, ma non è l'ideale per quanto riguarda l'economia di spazio sul dischetto. In proposito occorre dire che tale economia si ottiene introducendo codici di identificazione. Ad esempio, facendo corrispondere ad ogni città un numero di tre cifre (quindi con 1.000 possibilità diverse, da 000 a 999), si evita di scrivere i loro nomi per intero; alla località Bologna potrebbe corrispondere il numero 034, a Milano il numero 003, a Cagliari il numero 107, e così via. Và da sè che in un file a parte si registrano le corrispondenze numeri-località; tale file sarà utilizzato per dare, in sede di stampa su video o su carta, il nome invece del numero.

Analogamente, anziche dedicare tanto spazio alle annotazioni, potremmo lavorare ancora con codici: i vari tipi di articoli, le forme di pagamento e ogni altra cosa da memorizzare potrebbero essere registrati riferendoci ad altri numeri convenzionali. Anche queste corrispondenze andrebbero depositate in un file per essere usate all'occorrenza.

Tale modo di procedere è un po' più lungo come impostazione, ma fa risparmiare molto spazio ed anche del tempo di elaborazione: si fa prima a ricercare il numero 218 che non la stringa SENIGALLIA tra i nomi delle località, tanto per fare un esempio a caso.

Noi potevamo avere impostato tutto in questo modo, ma il programma sarebbe diventato troppo complesso per chi è alle sue prime esperienze col BASIC; quindi abbiamo preferito scegliere per ora questa soluzione, riservando esempi futuri al secondo metodo accennato.

Analizziamo il contenuto della subroutine che inizia al numero di riga 10110. Il suo scopo è quello di scrivere in reverse il titolo del dato da introdurre, e poi quello di porre ben in evidenza il numero massimo di caratteri che possono entrare a farne parte.

Supponiamo, tanto per fissare le idee, di arrivare a questa subroutine dalla linea 1410. TX$ vale NOME (trascurando gli spazi) e Q è stato posto uguale a 30. La linea 10110 cancella il monitor; con la 10120 si ottiene la scrittura, in reverse, della stringa TX$, all'inizio della settima riga video. Al numero di programmazione 10130 inizia un ciclo FOR...NEXT nella variabile I, che viene fatta partire (INIZIALIZZATA) da uno e deve arrivare a cinque.

Vi sarete resi conto che il contenuto di alcune righe è spostato, in maggiore o minor misura, verso destra; la cosa è voluta, per indicare in modo inequivocabile dove inizia e finisce un LOOP FOR...NEXT; così, nel nostro caso, salta subito agli occhi che il loop della variabile I comincia alla linea 10130 e termina alla 10200. Il listato che si trova ancora più a destra indica che si tratta di altri loops; nel caso in esame, abbiamo a che fare (righe 10150-10160 e 10180-10190) con cicli NEXT...FOR nella variabile J, che servono unicamente per creare un ritardo temporale. Infatti con quei cicli diciamo al computer, in pratica, di contare da 1 a 15 senza compiere nessun'altra operazione. Più alto è il numero d'arrivo, più lungo è il tempo impiegato per arrivarci.

Procediamo con ordine, esaminando la linea 10140. Essa dice al computer di scrivere, ad iniziare dal carattere 207 della mappa video, la stringa indicata tra virgolette: 14 spazi vuoti.

Le due righe successive danno un piccolo ritardo, poi la linea 10170 fa scrivere, sempre alla posizione precedente, prima il numero Q e poi, di seguito (per la presenza del punto e virgola) la frase CARATTERI.

Se pensate che Q può essere, come vedrete, di tre cifre, e se considerate che un tale numero è lungo 5 caratteri (aggiungendo un carattere per il segno ed uno alla fine, come già sapete), otterrete che la stampa in oggetto arrivi ad un numero massimo di 14 caratteri, come per la stringa di spazi della linea 10140.

Segue poi un altro loop di ritardo, uguale al precedente, e finalmente l'istruzione NEXT.

Cosa avviene allora?

Dopo l'ingresso nella subroutine, si cancella il monitor e viene scritta la stringa TX$, in reverse. Poi si compie per cinque volte quanto segue: viene prima cancellato uno spazio di 14 caratteri, alla destra della stringa già scritta; dopo un piccolo intervallo di tempo, sempre nella stessa posizione vengono scritti Q e la parola CARATTERI. Passa un altro po' di tempo, ed il ciclo descritto ricomincia. In pratica, arrivando dalla linea 1410, si ha la scritta in reverse NOME, poi, in rapido lampeggio, la scritta 30 CARATTERI. Si è così ottenuto il risalto desiderato per quella comunicazione.

Speriamo di esserci spiegati a sufficienza; il listato sfalsato ed il diagramma di flusso della subroutine (riportato in figura 14 assieme alle altre) dovrebbero aiutarvi molto. In particolare, dal diagramma di flusso si vede chiaramente il concetto di LOOP NIDIFICATO: i vari cicli FOR...NEXT sono indicati in tratteggio, e quello nella variabile I è all'esterno e contiene in sé i due loops di ritardo nella variabile J.

Ancora qualche considerazione. La linea 1430 controlla che la lunghezza della stringa N$ non sia superiore a Q (30 in questo caso). Se supera tale limite, essa viene rifiutata e si riparte dalla linea 1400.

La riga 1440 porta la stringa N$ alla lunghezza di 30 caratteri: infatti se quella digitata è più corta, il computer percorre più volte la 1440, aggiungendo ad ogni tornata uno spazio in coda; quando la lunghezza di N$ è uguale a trenta, la condizione IF non è più verificata, ed il programma prosegue oltre.

Bisogna dire che questo procedimento è stato introdotto per semplificare la gestione dei files interessati. La cosa verrà chiarita al momento opportuno. Il ciclo visto per TX$ = "NOME" si ripete identico per le stringhe VIA, CITTÀ, TELEFONO e ANNOTAZIONI; cambia solo il valore di Q.

Siamo così giunti alla linea 1640. In essa le variabili I# e P# sono azzerate; questo per evitare errori nelle istruzioni seguenti, dove occorre introdurre tali valori. Infatti, se alla domanda di introdurre l'importo della fattura (1650-1670) si risponde premendo RETURN, allo scopo di introdurre la cifra zero, in realtà verrebbe mantenuto per I# il valore presente in memoria, registrato al giro precedente (ammesso che ci sia stato). Meglio quindi azzerare quei valori.

1710-1730

Tutti i dati introdotti finora, ad esclusione degli importi, vengono raccolti in un'unica variabile di stringa Z$, che è posta uguale alla somma delle stringhe precedenti. Sarà Z$ ad essere registrata nel file interessato.

Dopo la cancellazione del monitor (linea 1720), dobbiamo andare ad effettuare la registrazione dei dati appena introdotti. L'operazione richiederà un po' di tempo, quindi si passa nuovamente nella subroutine 10040 che abbiamo già esaminato all'inizio: essa scrive PROGRAMMA IN CORSO — ATTENDERE.

1740-1750

La linea 1740 impartisce le seguenti istruzioni: "se C = 0 (ossia se stiamo trattando un fornitore), allora il numero dei fornitori va aumentato di uno (NF% = NF% + 1) e la variabile PU deve assumere il valore di NF%; altrimenti (ossia se si tratta di un cliente) il numero dei clienti va incrementato di uno e PU assume questo valore".

Sul fatto di incrementare di uno il numero dei clienti o dei fornitori c'è poco da dire: aggiungendo nominativi il loro totale aumenta. La variabile PU è quella utilizzata come PUNTATORE per decidere su quale record del file random interessato dobbiamo andare ad incidere. In sede di aggiunta tale puntatore si deve posizionare dopo l'ultimo nome precedentemente registrato (PU uguale al nuovo numero di clienti o di fornitori). Invece se siamo in fase di correzione, o di registrazione di movimenti, il puntatore deve andare sul nominativo richiesto. Vedremo che ciò avviene indicando il numero del nome interessato, e che questo numero sarà chiamato N (linee 2020 e 2250). Ecco allora che in fase di correzione (CR = 1) o di registrazione di movimenti (RM = 1) PU deve valere N; questo spiega il contenuto della linea 1750.

Spendiamo qualche parola sull'espressione NF% = NF% + 1: essa, in termini aritmetici, è un errore, in quanto un numero non può mai essere uguale al suo successivo. In realtà la relazione scritta non è un'uguaglianza, ma un'operazione di ASSEGNAZIONE del valore di una variabile, e va letta in questo

modo: "il nuovo valore della variabile NF% è uguale al vecchio valore aumentato di uno". Questo è un ragionamento lecito anche in aritmetica, ma la sua traduzione in linguaggio BASIC assume questa forma anomala. Tali considerazioni sono molto importanti, e meritano, da parte vostra, un buon approfondimento.

1760-1790

La parte di programma che va dalla linea 1760 alla 1790 registra i dati introdotti nel relativo archivio. Infatti dapprima c'è la dichiarazione di apertura del file random di nome NO$, ossia CLIENTE o FORNITORE a seconda della scelta effettuata al menù principale (ricordare quanto detto alle righe 1270 e 1280). In un secondo tempo (1770) c'è la dichiarazione di FIELD: alle stringhe I$ e P$ sono riservati otto caratteri, in quanto in esse depositeremo i valori numerici di I$ e P$; essendo dei numeri in doppia precisione, essi hanno l'ingombro suddetto.

A proposito dei files random, fino ad ora abbiamo visto solo letture. In quel caso, dopo aver fatto l'apertura e la FIELD, si poteva passare al prelevamento dei dati. In registrazione, invece, è necessaria anche la dichiarazione LSET oppure RSET. Vediamo di che cosa si tratta.

Nelle varie porzioni di campo dichiarate dobbiamo registrare delle stringhe. Ben raramente, come abbiamo già accennato all'inizio, la loro lunghezza coincide coi valori di campo; allora bisogna sempre dire se la stringa da incidere va posizionata a destra (RSET = RIGHT SET, metti a destra) oppure a sinistra (LSET = LEFT SET, metti a sinistra) nella rispettiva porzione di campo assegnatale. Se dopo la FIELD manca la dichiarazione LSET o RSET, i dati vengono tutti azzerati, quindi si ottengono risultati completamente errati.

Occorre anche precisare che allineando a sinistra con LSET, gli eventuali caratteri mancanti al raggiungimento del valore assegnato da FIELD sono automaticamente riempiti con dei blanks; essi vengono invece messi a sinistra nel caso si usi RSET. Qualora poi il dato da posizionare a destra o a sinistra risulti più lungo del campo assegnatogli, i caratteri in eccesso vengono tagliati via e persi irrimediabilmente. Per tutti questi motivi è consigliabile valutare molto attentamente quali sono le esigenze richieste dalle varie stringhe: se si pecca in abbondanza si spreca molto spazio, mentre nel caso opposto si perdono parte delle informazioni.

Al solito, sarà l'esperienza l'unica buona consigliera al riguardo; più che mai vale l'adagio "sbagliando s'impara"!

La riga 1790 effettua la registrazione dei dati. Il formato completo è quello indicato: PUT1,PU. PUT (= metti) ordina l'incisione; 1 è, al solito, il numero che vogliamo registrare. Con PUT si muovono sempre 255 caratteri; qualora la FIELD ne specifichi un numero minore, i rimanenti vengono posti uguali a degli spazi, automaticamente. Se la dichiarazione di campo supera i 255 caratteri, si ha una segnalazione d'errore (codice 51 = FIELD OVERFLOW).

Precisiamo anche che lo spazio assegnato alle varie stringhe con FIELD non ha nulla a che fare con quello da dare con CLEAR: la FIELD suddivide solamente il buffer, perciò lo spazio riservato alla memorizzazione delle stringhe può anche essere inferiore a 255.

Un'ultima considerazione. Se in un file random che contenga, ad esempio, 13 records incisi, andiamo a registrare il trentottesimo record (cosa possibile assegnando il numero 38 al puntatore), il sistema lascia automaticamente tutto lo spazio vuoto equivalente ai records non registrati. Attenzione, in questo caso, alle successive operazioni di lettura. Se il disco conteneva in precedenza dei files che poi sono stati cancellati, lo spazio liberatosi è diventato disponibile. L'operazione di registrazione oltre la posizione dell'ultimo record effettivamente registrato riserva, come abbiamo appena detto, lo spazio anche a quelli mancanti. Però la precedente operazione di KILL non aveva in effetti cancellato i dati del file eliminato: ora essi sono ancora lì, ed occupano i records non effettivamente registrati. Potete facilmente immaginare che una loro lettura darebbe risultati completamente errati. Meglio sarebbe lavorare sempre su dischetti vergini, oppure incidere delle stringhe nulle negli elementi che vengono saltati.

1800-1960

Nella linea 1800 viene aperto il file random DATI, utilizzando il buffer 2; la riga seguente fa l'assegnazione di campo. Gli otto caratteri riservati ad S$ servono per registrare SC# e SF#, i due caratteri di N$ ed M$ sono per NC%, NF%, MC%, MF%. Il file DATI è formato da due records; nel primo registriamo i dati che

riguardano i clienti (SC#, NC%, MC%), nel secondo quelli dei fornitori. Le linee da 1840 a 1880 registrano il primo record, mentre quelle da 1900 a 1940 riguardano i fornitori. Quando si passa per questa parte dl programma, solo uno di questi due blocchi viene percorso (o cliente o fornitore), quindi serve un controllo per sapere quale strada dobbiamo fare imboccare al computer. Per questo scopo esiste la linea 1820, che va a vedere il valore assunto della variabile C.

La riga 1830 serve per assegnare il giusto valore alla variabile SC#, a seconda che stiamo lavorando sui nominativi (aggiunta o correzione) o che stiamo effettuando la registrazione di un movimento: il tratto dl programma che parte dalla linea 1750 viene utilizzato anche per le registrazioni dei movimenti, come potete vedere leggendo la linea 2600. La variabile che controlla quale operazione si sta facendo è, in questo caso, RM; se RM = 1 significa che l'operazione in corso è un movimento, in quanto passando per la linea 2220 RM assume quel valore. Essa viene poi posta uguale a zero ogni volta che si passa per il menù principale (riga 1170), quindi tale valore di RM caratterizza sia la registrazione di un nuovo nome che la correzione di uno vecchio.

Dopo queste spiegazioni, il contenuto della riga 1830 dovrebbe risultare chiaro. SE RM = 0, I# e P# sono quelli trattati fino ad ora dalla parte di programma che stiamo esaminando; se invece RM = 1 significa che si è arrivati qui dalla routine dei movimenti, e quindi I# e P# devono essere collegate agll importi dei movimenti (che sono IM# e PM#, come vedremo a suo tempo).

In ogni caso, come vedete, il nuovo valore di SC# è uguale a quello precedente aumentato di I# (o IM#) e diminuito di P# (o PM#), come è logico che sia.

Se stiamo trattando clienti, il relativo blocco arriva fino alla riga 1880; in questa linea c'è l'ordine di saltare la parte che serve per i fornitori: GOTO 1950. A questo numero di riga si arriva anche dopo aver percorso il tratto dei fornitori, che termina al numero 1940. A questo riguardo, rammentiamo che PUT2,2 significa "passando per il buffer 2, registra il record 2".

Con la 1950 si effettua la chiusura dei due files che sono stati aperti: quello di nome NO$ e quello DATI. La CLOSE fatta senza specificare il numero o i numeri di buffer provvede alla chiusura di tutti quelli aperti in precedenza. Per finire, la linea 1960 dice al computer di andare alla 2220 se l'operazione appena conclusa era la registrazione di un movimento, e di tornare invece al menù principale (1170) se si trattava di un nominativo aggiunto o corretto.

1970-2070

Dopo che CR è stata posta uguale ad 1 (abbiamo detto poco fa che questo ci serve per sapere quando siamo in correzione di nominativi), si fa la cancellazione del video. Le linee 2000 e 2010 fanno descrivere le frasi tra virgolette; notare che la differenza tra cliente e fornitore è data da NO$.

Nella riga 2020 il computer resta in attesa di un numero introdotto dalla tastiera, il quale rappresenta il cliente o il fornitore che si intende correggere.

Se nel menù secondario delle registrazioni si era scelta la strada delle correzioni, giunti a questo punto saremmo costretti ad effettuarne una, anche se ci fossimo pentiti della decisione presa. Allora è stata Inserita la possibilità di battere il numero 0 (che ovviamente non ha corrispondenza né nei clienti né nel fornitori) per tornare al menù principale.

La linea 2030 è proprio quella preposta al compito suddetto: "se N = 0 va alla riga 1170".

La linea 2040 contiene un'istruzione del tipo IF...AND...THEN: l'alternativa THEN è eseguita solo se entrambe le condizioni IF e AND sono soddisfatte. In questi casi si parla anche di PRODOTTO LOGICO, e tale concetto è stato usato nel diagramma di flusso per rappresentare questa riga.

In pratica qui si stabilisce che se stiamo trattando fornitori (C = 0) e se il numero N introdotto risulta maggiore del loro numero totale, allora bisogna tornare alla linea 1980: il tutto viene ripetuto, perché è ovvio che non può essere corretto quello che ancora non c'è.

La linea successiva provvede allo stesso tipo di controllo, effettuato però sui clienti.

Dopo aver cancellato il monitor, si passa per la sobroutine 10040 che già conosciamo (PROGRAMMA IN CORSO — ATTENDERE).

2080-2110

Si è passati per la sobroutine 10040 perché ora apriamo il file NO$ (CLIENTE o FORNITOR) per andare a prelevare il nominativo corrispondente al numero N richiesto.

Il contenuto delle linee 2080-2090-2100-2110 vi è oramai familiare. GET1,N serve per leggere il record di ordine N, corrispondente al nominativo cercato.

2120-2200 E SUBROUTINE 10050

L'operazione precedente è stata fatta per fornire sullo schermo il nominativo collegato al numero introdotto: solo questo dà la certezza di effettuare la correzione sul cliente (o fornitore) desiderato. Ora infatti si cancella il monitor, poi con la linea 2130 si scrive CLIENTE (o FORNITORE, a seconda del contenuto di NO$) e si lascia una riga. La linea 2140 serve per scrivere in reverse i primi 30 caratteri della stringa 2$, quella definita con la FIELD precedente. Essi contengono il nome del cliente (o del fornitore) specificato.

Ci preme, al riguardo, farvi notare una cosa importante: ogni FIELD serve per assegnare alle variabili indicate una fetta dei 255 caratteri che formano un record; NULLA VIETA DI ADOTTARE IN LETTURA UNA FIELD DIVERSA DA QUELLA USATA IN REGISTRAZIONE. Nel nostro caso, per fare un esempio pratico, si sarebbe potuta dare questa dichiarazione di campo:

```
2090 FIELD1, 8 AS I$, 8 AS P$, 30 AS Z$, 209 AS U$
```

Così facendo, in Z$ ci sono solo i caratteri riservati al nome, ed in U$ andrebbero a finire il resto delle informazioni. Ci risparmieremmo la separazione del nome fatta nella linea 2140, dove sarebbe sufficiente fare la stampa di Z$.

Ma torniamo al nostro programma.

A questo punto vogliamo offrire due possibilità: proseguire nella correzione oppure tornare al menù principale.

Per farlo passiamo per la subroutine 10050, che scrive in reverse la frase tra virgolette. La riga 10060 fa saltare la 10070 (che ci servirà in altre occasioni). La 10080 mette il computer in attesa di una battuta di tastiera, ed il relativo carattere viene depositato nella stringa X$. La riga 10090 stabilisce che fino a quando X$ contiene una stringa nulla, il programma deve continuare a chiedere la pressione di un tasto. Tale istruzione è necesssaria in quanto INKEY$ funziona nel seguente modo: il computer, incontrandola, si mette in comunicazione con la tastiera; se in quell'attimo non viene premuto alcun tasto, ad X$ viene assegnata una stringa nulla ed il programma prosegue. Noi invece vogliamo che esso si arresti fino a che non venga premuto un tasto, e tale funzione è realizzata appunto dalla linea 10090.

La 10100 dà il comando di ritorno al programma principale; con essa si rientra alla riga 2160, dove si verifica se il contenuto di X$ è la cifra zero: in tal caso significa che l'operatore desidera annullare la scelta di fare una correzione sul nominativo visualizzato poco prima, e quindi il computer torna al menù principale (GOTO 1170).

Se invece è stato premuto un tasto qualsiasi, diverso da quello dello zero, si arriva alla linea 2170 che cancella nuovamente il monitor. La 2180 serve per diminuire di uno il numero NF% dei fornitori, se il nome da correggere appartiene a quelli (ossia se C = 0). Analoghe considerazioni valgono per la riga 2190 (caso C = 1, ossia clienti). Queste due istruzioni sono necessarie in quanto alla linea 1740 (blocco di aggiunta dei nominativi) il numero totale dei clienti (o dei fornitori) viene aumentato di uno ad ogni registrazione. Dato che facciamo uso della tessa parte di programma anche per effettuare le correzioni dei nomi, se non facessimo nel modo appena visto il numero dei fornitori (o quello dei clienti) crescerebbe di uno, cosa che non deve succedere. Si è rimediato diminuendo qui quel valore di una unità, in modo che l'effetto della linea 1740 riporti le cose a posto.

La riga numero 2200 contiene il salto incondizionato alla linea 1400: si va all'inizio della routine di registrazione dei nominativi, che abbiamo già visto. Ora essa viene utilizzata per effettuare le correzioni desiderate sul nome scelto, quello corrispondente al numero N. Se ricordate, la registrazione avviene nel file random di nome NO$, al record individuato dal puntatore PU. In questo caso PU viene posto uguale ad N, come abbiamo già visto alla linea 1750.

2210-2300

Esaminiamo ora l'ultima parte di programma dedicata alle registrazioni. Dopo rimarranno quelle delle letture, molto più semplici.

Questo settore tratta la registrazione dei movimenti. Si arriva a queste linee dopo aver già scelto tra clienti e fornitori, al solito.

La linea 2220 pone uguale ad uno la variabile di controllo RM, il cui uso è già stato esaminato alla riga 1830. Inoltre le variabili IM# e PM# vengono azzerate, per considerazioni analoghe a quelle fatte per I e P# alla riga 1640.

Le linee dalla 2230 alla 2300 svolgono funzioni simili a quelle viste alle righe 1990-2070 e servono per i soliti controlli.

2310-2390

Anche le linee dalla 2310 alla 2890 sono simili al blocco 2080-2160: l'unica differenza consiste nella riga 2340, dove nelle variabili I# e P# vengono memorizzati gli importi del nominativo indicato col numero N (linea 2250). Se il nome che appare in reverse (2370) non è quello che volevamo, premendo il tasto dello zero si chiude prima il file, poi si torna al menù principale (2390). Facciamo osservare che il fatto di vedere un nome diverso da quello che ci si aspettava vuole dire che abbiamo introdotto un valore di N errato: non è il computer a sbagliare, ma l'operatore nell'introdurre i dati richiesti. Se si rinuncia alla registrazione di movimenti sul nominativo che era stato scelto, è necessario fare la CLOSE, altrimenti il calcolatore andrebbe in errore incontrando una successiva istruzione OPEN fatta sullo stesso file (codice 56 = FILE ALREADY OPEN).

Avremmo potuto evitare di aprire e chiudere in continuazione i files di tipo random che ci servono lungo il corso del programma: infatti sarebbe stato sufficiente aprirli tutti all'inizio e chiuderli alla fine delle elaborazioni. Abbiamo parlato solo di files random perché la loro apertura vale sia per leggere che per scrivere. Il discorso non fila nel caso dei files sequenziali, poiché la loro apertura per leggere non permettere operazioni di registrazione e viceversa. Per essi quindi saremmo in ogni caso obbligati alle aperture e chiusure fatte quando servono. Questo è un valido motivo per utilizzare files random al posto dei sequenziali, a meno che la loro gestione non diventi troppo complessa per il fatto di dover registrare sempre blocchi di 255 bytes.

Il fatto di usare solo files random accorcia anche i tempi di esecuzione, perché tutte le operazioni OPEN e CLOSE portano via del tempo.

I files dei movimenti (MOVCL e MOVFO) sono sequenziali, e la loro gestione dopo l'apertura dei tre files random DATI-CLIENTE-FORNITOR sarebbe inibita in quanto i buffers a disposizione sono solo tre. Del resto, per uno scopo didattico, il metodo adottato è senz'altro più valido.

Chi volesse sperimentare l'altro modo, può farlo ugualmente a patto di estendere innanzitutto il numero di files trattabili contemporaneamente. Prima di caricare il programma occorre allora portarsi a livello DOS, poi caricare il BASIC con questo comando:

```
BASIC 5
```

Così facendo si assegnano 5 buffers per lo scambio dei dati tra dischi e memoria centrale. Lo SPAZIO DOPO LA PAROLA BASIC È OBBLIGATORIO; se non viene messo il computer va alla ricerca del file BASIC5 e non trovandolo segnala errore. Il comando, invece viene accettato se si mette più di uno spazio. IL NUMERO MASSIMO DI BUFFERS È 15. Ogni buffer aperto occupa 290 bytes della memoria utente, quindi è meglio non aprirne troppi, se non servono. Cogliamo l'occasione per dire che esiste sempre, oltre ai buffers dei files, anche un buffer che viene utilizzato per caricare, salvare o fondere programmi; esso viene creato automaticamente dal computer e non ha nulla a che fare con quelli che servono per la gestione dei files.

Continuando ad esaminare la modifica proposta, c'è da dire ancora che il programma va poi cambiato aprendo tutti i files random all'inizio, per esempio alla linea 1090, assegnando ad ognuno di essi un numero di buffer diverso, naturalmente. Tutte le linee in cui compare l'apertura di uno di questi files vanno eliminate. Anche le righe in cui ci sono FIELD-GET-PUT devono essere riviste, per sistemare il numero di buffer. Quello che non cambia è la gestione dei files sequenziali, che deve rimanere così com'è adesso, tranne il numero di buffer che deve essere superiore a tre (infatti i primi tre sono usati per quelli random). Si deve poi mettere una voce in più nel menù principale, chiamandola ad esempio CHIUSURA oppure FINE, per effettuare la CLOSE dei files random, che altrimenti rimarrebbero aperti. Ovviamente aggiungendo una voce bisogna aggiornare la linea 1240 ponendo M = 6.

2400

La riga 2400 può sembrarvi strana: l'abbiamo messa proprio perché in essa si fa uso di un'istruzione poco nota, ma molto comoda. Se chiediamo la stampa video di CHR$(31), otteniamo la cancellazione del

monitor a cominciare però solo dalla riga in cui si trova il cursore in quel momento; le righe al di sopra non sono toccate.

Il richiamo della subroutine 10050 ha provocato la scritta PREMI UN TASTO (0 PER FINIRE) a cominciare dal carattere video 480, ossia nell'ultima riga del monitor. Con l'istruzione 2400 ordiniamo al computer di portare il cursore al carattere numero 128 (inizio della quinta riga) e di cancellare di lì in giù. In questo modo restano visualizzate le scritte che erano state fatte alle righe precedenti, ossia il numero del cliente (o del fornitore) ed il suo nome.

2410-2470

Alla linea 2410 ha inizio l'introduzione dei dati che caratterizzano un movimento; si comincia con la data, poi seguono la descrizione della merce, l'importo, l'acconto e le note.

Per la merce e le note esistono due routines di controllo della loro lunghezza, alle linee 2430 e 2470. Se esse superano i rispettivi valori di 95 e 63, vengono rifiutate e richieste. I vari spostamenti dati al cursore e l'uso di CHR$(31) servono per scrivere le varie cose al posto che loro compete, anche nel caso che il computer richieda dei dati.

La limitazione imposta alla lunghezza di M$ ed A$ (merce e note) è stata fatta per fare stare tutti i dati in una pagina video. All'occorrenza è possibile variare quei limiti, ricordando che in ogni caso una stringa non può contenere più di 255 caratteri. Dato che stiamo trattando files sequenziali, non ci sono problemi di campo.

Alle righe 2440 e 2450 si ha l'introduzione degli importi IM# e PM#, che abbiamo già menzionato parlando delle linee 1830 e 1890.

2480-2540

Con la riga 2480 viene chiesta la conferma dei dati introdotti; essi sono ancora tutti visibili sul monitor. Le linee che vanno dalla 2490 alla 2520 controllano la risposta data alla domanda CONFERMI?. Infatti con la riga 2490 il computer resta in attesa di un carattere da tastiera; se esso è N o S il programma prosegue alla linea 2520, altrimenti ritorna alla 2490. In tal modo solo la pressione dei due tasti indicati provoca l'avanzamento, mentre ogni altro viene ignorato.

Se la risposta è N, il file aperto viene chiuso e si ritorna alla 2220. Se invece la risposta è stata S, il programma prosegue: viene cancellato il monitor e si ottiene la scritta PROGRAMA IN CORSO — ATTENDERE passando ancora per la subroutine 10040, prima di andare a lavorare sui files.

2550-2600

La linea 2550 apre il file sequenziale di nome NN$, ossia MOVCL o MOCFO, a seconda che sia stata percorsa la linea 1270 o la 1280 (menù principale). Abbiamo già detto che un file sequenziale può essere aperto in tre modi: con la lettera O, con I o con E. Per registrare si usano la O oppure la E; al fine di non cancellare le registrazioni precedenti abbiamo messo la E, che attacca in coda; se avessimo usato la O, ogni nuova registrazione su quel file sarebbe ripartita dal primo elemento, cancellando tutto il resto. Tenere presente che SE L'APERTURA AVVIENE CON LA LETTERA E, NON SI HA ERRORE ANCHE SE IL FILE NON ESISTE ANCORA; in questo caso la registrazione avviene partendo dall'inizio.

Si è messo il numero di buffer 2 perché l'uno è ancora occupato dal file NO$, che non è stato ancora chiuso.

Con la linea 2560 si effettua la registrazione dei dati introdotti. Come vedete, occorre prima specificare il numero del buffer, poi mettere una virgola e gli identificatori delle variabili da incidere. Esse possono essere, nel caso di files sequenziali, anche di tipo numerico.

Vediamo di spiegare la presenza dei punti e virgola e della variabile Y$.

Per rendervi conto di come avvenga la registrazione dei dati in un file di tipo sequenziale, dovete pensare che essa segue lo schema delle visualizzazioni sul monitor causate da una istruzione PRINT. Infatti se per registrare nel file noi mettiamo delle virgole tra i vari dati da incidere, questi non risultano uno

di seguito all'altro, ma separati da uno spazio di 10 caratteri; si è quindi sprecato inutilmente dello spazio. Per evitare che questo avvenga occorre interporre tra i vari dati un punto e virgola; con questo metodo essi vengono registrati attaccati, tenendo presente che i numeri hanno un posto per il segno prima delle cifre ed uno alla fine di esse. Facciamo qualche esempio.

Da livello BASIC, usando il computer in modo diretto, potete scrivere via via quanto segue, operando fuori da un programma:

```
OPEN"O",1,"PROVA"
PRINT#1,11,22
CLOSE
```

In questo modo avete aperto un file sequenziale di nome PROVA, usando il buffer numero 1; in seguito avete registrato in esso due numeri (11 e 22); poi avete chiuso il file. Se ora scrivete:

```
OPEN"I",1,"PROVA"
INPUT#1,A,B
CLOSE
PRINT A : PRINT B
```

andate a leggere il file appena registrato, prelevando due valori numerici A e B. L'ultima riga dà sul monitor i valori letti, e vi ritrovate quelli che avevate introdotto.

Provate ora a scrivere:

```
CMD"PRINT PROVA"
```

Questo è un comando DOS che dà sulla stampante il contenuto del file PROVA.

Vi renderete così conto di come è stata effettuata la registrazione: tra i due numeri 11 e 22 sono stati lasciati 10 spazi (se li contate, arriverete a 12, ma i due in più sono quelli del segno e di fine numero).

Provate ora a scrivere quanto segue, sempre in modo diretto:

```
OPEN"E",1,"PROVA"
PRINT#1,-33;-44,-55;
CLOSE
```

Con ciò avete aperto lo stesso file per registrarvi in coda i numeri —33 —44 e —55, coi primi due separati da un punto e virgola, con —44 seguito da una virgola e con un punto e virgola dopo l'ultimo numero. Effettuate ora la lettura nel seguente modo:

```
OPEN"I",1,"PROVA"
INPUT#1,A,B,C,D,E
CLOSE
?A : ?B : ?C : ?D : ?E
```

Al solito, riotterrete i 5 numeri che ora si trovano registrati nell'archivio. Date ancora il comando CMD"PRINT PROVA" e vedrete come sono stati incisi gli ultimi tre numeri: dopo la stampa di 11 e 22, si ha un ritorno a capo e la scrittura dei numeri —33 —44 e —55, spaziati in vario modo. Noterete che tra —44 e —55 ci sono 9 spazi: la spaziatura di 10 caratteri causata dalla virgola che si trova tra —44 e—55 ha effetto cominciando a contare dal primo numero che non è preceduto da un punto e virgola. Se fate i conti, avete 3 caratteri per —44 e i nove spazi successivi fanno arrivare a 12.

È evidente che al variare del numero delle cifre che compongono i vari numeri registrati, si ottengono spaziature diverse.

Fate qualche altra prova per rendervi conto di come funziona il tutto.

Scrivete ora come segue:

```
OPEN"E",1,"PROVA"
PRINT#1,66;-77;"NUOVA";"ELETTRONICA"
CLOSE
```

Effettuate una lettura nel seguente modo:

```
OPEN"I",1,"PROVA"
FOR I=1 TO 7 : INPUT#1,A(I) : PRINT A(I) : NEXT
INPUT#1,A$ : PRINT A$
CLOSE
```

ottenete i cinque numeri introdotti e la stringa NUOVAELETTRONICA.

Come vedete, il problema di registrare e leggere numeri è risolto, mentre quello delle stringhe ancora no: due stringhe incise interponendo un punto e virgola vengono attaccate sul disco e non sono più separabili in lettura.

Provate allora a fare così:

```
OPEN"O",1,"PROVA"
PRINT#1,123;-4567;",";"NUOVA";",";"ELETTRONICA";
CLOSE
```

```
OPEN"I",1,"PROVA"
INPUT#1,A,B,C$,D$
CLOSE : ?A : ?B : ?C$ : ?D$
```

Dato che si è aperto il file PROVA con O, esso viene inciso dall'inizio ed il contenuto del precedente vlene perso.

Se avete fatto esattamente come indicato, ora la parola NUOVA è in C$ ed ELETTRONICA è in D$: come vedete, la registrazione della virgola ha separato le due stringhe. Abbiamo così fatto un buon passo avanti, ma non abbiamo ancora risolto tutti i problemi; infatti, se una stringa contiene al suo interno una virgola, questa viene poi interpretata come un carattere separatore, e in lettura causa errori. Vediamo:

```
OPEN"o",1,"PROVA"
PRINT#1,"BOLOGNA";",";"VIA CRACOVIA, 19"
CLOSE

OPEN"I",1,"PROVA"
INPUT#1,A$,B$ : PRINT A$ : PRINT B$
```

Non fate CLOSE, ancora. Vedrete che A$ vale BOLOGNA, e B$ VIA CRACOVIA. Provate ora a fare:

```
INPUT#1,C$ : PRINT C$ : CLOSE
```

Si ottiene per C$ la stringa 19. Non lo abbiamo detto più, ma sarebbe bene che ad ogni esperimento voi faceste la stampa del file appena registrato, in modo che vi possiate rendere conto di quello che accade sul dischetto.

Per evitare che accada l'inconveniente appena visto, si fa in questo modo:

```
OPEN"O",1,"PROVA"
PRINT#1,CHR$(34);"BOLOGNA";CHR$(34);CHR$(34);"VIA CRACOVIA, 19";CHR$(34)
CLOSE

OPEN"I",1,"PROVA"
INPUT#1,A$,B$ : PRINT A$ : PRINT B$ : CLOSE
```

CHR$(34) corrisponde alle virgolette. Tale carattere, registrato tra una stringa l'altra nel modo visto, viene interpretato come unico separatore di stringhe, le quali così possono contenere al loro interno anche delle virgole. Dopo BOLOGNA il carattere virgolette è messo due volte: le prime servono per chiudere la stringa BOLOGNA, le seconde per aprire la stringa successiva. Questo punto è molto importante, quindi tenetelo ben presente. Mettendo solo un carattere separatore, la lettura dà risultati errati.

Con l'esempio precedente si ottengono allora le due stringhe BOLOGNA e VIA CRACOVIA, 19. Finalmente siamo arrivati!

Ecco allora qualche regola, come riepilogo di quanto visto fino ad ora.

NEL TRATTAMENTO DEI FILES SEQUENZIALI I NUMERI POSSONO ESSERE REGISTRATI INTERPONENDO VIRGOLE O PUNTI E VIRGOLA; MEGLIO SCEGLIERE IL SECONDO SISTEMA, PERCHÉ FA RISPARMIARE SPAZIO.

LE STRINGHE INVECE VANNO SEPARATE CON DELLE VIRGOLE DATE SOTTO FORMA DI STRINGA.

SE COMPAIONO DELLE VIRGOLE ALL'INTERNO DELLE STRINGHE, QUESTE VANNO SEPARATE LE UNE DALLE ALTRE CON DELLE VIRGOLETTE, SEMPRE DATE SOTTO FORMA DI STRINGA.

Ovviamente alle scritte "," e CHR$(34) possono essere sostituite delle variabili che le contengono. Ad esempio, ponendo

```
V$=","  : Y$=CHR$(34)
```

al posto di "," si può scrivere V$, e invece di CHR$(34) si può mettere Y$.

Torniamo finalmente al nostro programma. Ora la linea 2560 non ha più segreti. La variabile Y$ era stata definita, se ricordate, alla riga 1030.

Con la 2570 chiudiamo i due files aperti; con la 2580 aggiorniamo gli importi. Il significato della linea 2590 è chiaro: se stiamo trattando clienti (C = 1) il numero dei loro movimenti viene incrementato di uno, altrimenti si fa la stessa cosa col numero dei movimenti dei fornitori MF%.

La riga 2600 contiene l'istruzione di salto alla 1750, dove i dati finora introdotti vengono registrati nei files opportuni. Tale processo è già stato descritto.

2610-2680

Le linee che vanno dalla 2610 alla 2680 vi sono sicuramente chiare, in quanto, trattandosi del terzo

menù, la spiegazione dei due precedenti vi ha fatto capire il procedimento. Al solito, prima di andare alla subroutine 10010, occorre dare il numero delle voci contenute nel menù, ossia 3. Il ritorno dalla subroutine entra nella linea 2680, che effettua i rinvii opportuni, in dipendenza della scelta operata nel menù.

2690-2730

La routine che esaminiamo ora serve per leggere nominativi di clienti e di fornitori, come indica anche la REM della linea 2690. A questo proposito, avrete forse notato che le righe delle istruzioni REM non vengono mai percorse (tranne la 1160). Si è adottato tale sistema per rendere un po' più veloce l'esecuzione del programma: se non le incontra mai, il computer si risparmia il disturbo di ignorarle. Comunque, anche se succedesse il contrario, non si noterebbe una differenza apprezzabile.

I controlli effettuati dalle righe 2710 e 2720 rimandano al menù principale qualora sia stato richiesto un nominativo che non può esistere (NC% o NF% uguali a zero). Le condizioni da porre sono multiple, e corrispondono al prodotto logico; per questo sono rappresentate come vedete nel relativo diagramma di flusso.

Con l'istruzione successiva si passa per la routine 10040, per scrivere la frase in attesa che il computer effettui le operazioni sui dischi.

2740-2910

Ha inizio il ciclo vero e proprio di lettura dei nomi e di quanto li accompagna. La funzione viene svolta con un loop FOR...NEXT nella variable I; essa viene fatta partire da uno e deve arrivare, al massimo, al numero totale dei nomi (NC% per i clienti ed NF% per i fornitori). Per raggiungere lo scopo con un unico loop, prima di entrarvi la variabile H viene posta uguale ad uno dei due numeri citati. Quella variabile rappresenta poi il valore d'arrivo di I. La riga 2740 svolge appunto la funzione di assegnare il giusto valore ad H.

Con la 2750 si apre il file NO$ (CLIENTE o FORNITOR), e subito dopo si effettua la necessaria dichiarazione di campo. Incontriamo poi il loop già menzionato, che risulta ben evidente dalla marginatura sfalsata.

Tutte le operazioni comprese in questo tratto vengono compiute almeno una volta (e questo spiega la presenza delle linee 2710-2720, altrimenti saremmo andati a leggere un nome anche se formato da stringhe vuote e da numeri zero); il numero di percorrenze dipende dal valore di H. Se esso vale 1, il ciclo è fatto per una volta soltanto; se vale due, per due volte, e così via. Ripetiamo ancora: esso verrebbe compiuto una volta anche con H minore di I; provare per credere. Tale comportamento è comune a tutti i computers, e bisogna tenerlo ben presente quando si costruiscono dei loops il cui termine finale è messo sotto forma di parametro, come in questo caso; se succede che tale variabile assuma un valore inferiore a quella che fa eseguire il ciclo, questo viene compiuto ugualmente. Se la cosa non era stata prevista, si può incorrere in errori anche gravi.

La 2780 ordina al computer di leggere dal file aperto l'elemento di ordine I, ossia, visto che I parte da 1, il primo record. Esso contiene tutti i dati relativi al primo nominativo. Dal momento che presenteremo sul video i valori letti, dovremo prima cancellarlo con CLS.

Con la riga 2800 si fa la stampa in negativo di NO$ e del suo numero d'ordine, ossia vedremo scritto CLIENTE NUMERO 1, poi CLIENTE NUMERO 2, e così via man mano che andremo avanti (supponendo naturalmente di avere a che fare con clienti).

La linea 2810 decodifica le stringhe I$ e P$, ritornandoci i relativi numeri I# e P# . La funzione usata per fare questo è CVD, che è l'inverso della MKD$ adoperata per codificare in stringhe i numeri in doppia precisione.

Facciamo qualche considerazione sulla dichiarazione di campo. Quando il file NO$ era stato inciso, la dichiarazione di campo (linea 1 770) era diversa da quella in esame ora. Tale dichiarazione era molto semplice, perché conteneva le due stringhe necessarie per I# e P# e la stringa Z$ (239 AS Z$). Ciò era reso possibile dal fatto di aver portato tutte le stringhe (introdotte da tastiera nelle linee 1400-1630) ad una lunghezza prestabilita. NOME, VIA, CITTÀ a 30 caratteri, TELEFONO a 20, ANNOTAZIONI a 129. Avrete

però visto che, mentre per le altre stringhe esistevano due controlli sulla lunghezza (>Q e <Q), per le annotazioni la verifica veniva fatta solo per appurare che non superasse il valore 129. In quel caso era infatti superfluo portare la stringa R$ a quella lunghezza, in quanto vi avrebbe provveduto automaticamente il computer in sede di registrazione su disco della stringa Z$. Questa è la somma delle varie diciture introdotte (linea 1710), dove solo R$ è rimasta della sua lunghezza originaria; quando si passa alla registrazione, la stringa Z$ viene incisa seguendo la FIELD e la LSET specificate in 1770 e 1780. Ecco allora che gli eventuali caratteri mancanti a destra vengono riempiti con spazi dal computer.

Quello che ci preme far notare è che le varie stringhe componenti la Z$ sono sempre registrate in posizioni ben precise, nell'ambito dei 239 caratteri di Z$. Per farvi capire bene la gestione dei files, abbiamo ora fatto una dichiarazione di campo diversa da quella adottata in registrazione; la cosa è ammessa e la FIELD usata può essere qualunque. Va da sé, però, che tale affermazione è puramente teorica, in quanto la lettura può essere fatta in modo diverso dalla registrazione, ma sempre non perdendo d'occhio lo schema che si era seguito in quella sede.

Se allora ci pensate bene, la FIELD che esaminiamo ora equivale in effetti a quella della registrazione: i primi otto caratteri dei 255 letti sono di I$, i successivi otto vanno a P$, poi 30 rispettivamente a N$ (NOME), a V$ (VIA) e a C$ (CITTÀ). Seguono poi 20 caratteri per T$ (TELEFONO) e 129 per A$ (ANNOTAZIONI). Tutto quindi collima con l'incisione. Facendo così ci risparmiamo di dividere l'unica stringa di 239 caratteri; tale suddivisione è effettuata con la dichiarazione di campo.

Le linee dalla 2820 alla 2860 servono per visualizzare tutti i dati relativi ad un nominativo, e non ci dilungheremo in descrizioni ovvie.

Si arriva quindi alla riga 2870, che richiama la subroutine 10050 già vista: se si preme il tasto dello zero si esce dal loop, andando subito alla chiusura del file e tornando poi al menù principale (linee 2880, 2900 e 2910). Se invece il tasto premuto è un altro qualsiasi, il programma prosegue alla riga 2890: NEXT. Questa istruzione rimanda il programma al FOR ad essa collegata, cioè di nuovo alla linea 2770. Il ciclo si ripete fino a che la variabile I non assume un valore uguale o superiore ad H: vengono letti, uno dopo l'altro, tutti i nominativi del file, a meno che non si prema il tasto dello zero. Quando si è arrivati ad H, ossia al numero totale dei nomi presenti nell'archivio NO$, si esce dal loop, si fa la chiusura del file e si va al menù principale.

Il valore di I all'uscita del ciclo FOR...NEXT è sempre maggiorato di uno rispetto ad H, tranne che se usciamo prima della fine naturale, premendo appunto lo zero. Vi facciamo anche osservare che nel nostro caso l'istruzione STEP (PASSO) non è stata messa perché vale 1, e tale valore è appunto quello messo automaticamente dal computer in mancanza di STEP (è entrato nell'uso comune, ormai, dire che 1 è assunto per DEFAULT, che in inglese significa appunto mancanza. Ve lo diciamo perché sappiate cosa vuol dire se vi capita di leggerlo da qualche parte).

2920-3140

La parte di programma che va dalla linea 2920 alla 3140 è adibita a gestire la lettura dei movimenti, sia dei clienti (file MOVCL) che dei fornitori (MOVFO). Lo schema di funzionamento è identico a quello appena visto per la lettura dei nomi; trattandosi ora di files sequenziali, mancano le parti che qui non servono.

3140-3320 E SUBROUTINE 10070

Tra le linee 3140 e 3320 è compresa l'ultima parte del programma, quella che fa vedere la situazione dei pagamenti. Si utilizzano i valori di SC# e SF#, ossia delle cifre in doppia precisione che riassumono rispettivamente la situazione dei pagamenti di tutti i fornitori e di tutti i clienti. Le cifre e le frasi visualizzate dal monitor dipendono tutte dai valori di quelle due variabili, e vi risparmiamo la descrizione di questa parte in quanto risulta di comprensione pressoché immediata. Alla linea 3310 si va alla subroutine 10070, che differisce dalla 10050 (già vista) per il tipo di dicitura stampata; il resto è in comune con la 10050.

Alla fine, dopo la pressione di un tasto qualunque, si ha il ritorno al menù principale.

LA GESTIONE DEGLI ERRORI

Avendo spiegato riga per riga il contenuto del programma, non ci resta che accennare brevemente alla gestione degli errori.

Dopo aver fatto RUN, c'è da aspettare un po' perché il computer fa la ricerca del file DATI, che naturalmente non trova; allora va sul primo disco che non abbia la protezione contro le registrazioni per creare quel file. Se non c'è un disco registrabile, il programma si arresta alla linea 10220, poiché ogni errore porta in quel punto. Il codice d'errore sarà 52 (INTERNAL ERROR). Per trattarlo occorre modificare il contenuto della linea 10220, che potrebbe diventare il seguente:

```
10220 IF ERR/2+1=52 PRINT "NON POSSO REGISTRARE - PROVVEDI" : GOSUB 10070 :
      RESUME 1030
```

Sul monitor apparirà il messaggio, e la scritta data dalla subroutine 10070. Dopo aver provveduto in merito, premendo un tasto il programma prosegue alla linea 1030, come stabilito da RESUME. Ricordate che ogni routine di gestione degli errori deve terminare sempre con RESUME.

Se invece supponiamo che si arrivi alla 10220 perché il disco è pieno, allora l'errore relativo (62) va trattato in modo diverso. Bisogna aggiungere un'altra linea, che potrebbe essere questa:

```
10230 IF ERR/2+1=62 PRINT "DISCO PIENO - PROVVEDI" : GOSUB 10070 : CLOSE :
      RESUME 1170
```

Occorre fare CLOSE perché l'errore si verifica con un file aperto in registrazione. Dopo aver cambiato disco, si preme un tasto ed il programma riprende col menù principale. Occorrerà naturalmente ripetere la registrazione degli ultimi dati, poiché non sono stati registrati.

C'è da dire ancora qualcosa, a questo riguardo. Se avete un solo floppy, cambiando disco se ne va anche il file DATI, che invece è indispensabile per il corretto funzionamento del programma. Non vi resta che operare a livello di comandi diretti per registrarlo sul disco nuovo, coi valori SC# e SF# del disco vecchio. Invece NC%, MC%, NF%, MF% vanno posti uguali a zero, dal momento che il nuovo disco non contiene nessun altro file. Così facendo vi portate dietro gli importi, per avere sempre una situazione finanziaria corretta. Un modo più semplice per effettuare il cambio del disco, sarebbe il seguente. Mettete un disco nuovo (avendo un solo drive, deve contenere il BASIC), poi fate RUN. Per non perdere la situazione finanziaria (infatti il file DATI ricreato su questo disco sarà vuoto), potete registrare un cliente ed un fornitore fittizi, mettendo per ognuno di essi il saldo che compete loro. Quando farete una lettura di nominativi, sapete che il primo rappresenta la situazione finale del disco precedente. Naturalmente in ogni nuovo dischetto le altre variabili ripartono da zero.

Se invece avete due floppy, potreste mettere il nastro di protezione sul disco del BASIC, in modo da lavorare solo sul secondo drive. Quando il dischetto è pieno, basta cambiarlo e fare come detto poco fa per ripartire.

COME USARE IL PROGRAMMA

Dopo aver fatto RUN, il computer crea il file DATI e passa al menù principale. A questo punto solo la scelta 5 e la registrazione di nominativi vengono accettate; le letture non possono aver luogo perché non esiste ancora nulla. Cominciate allora ad introdurre clienti e fornitori, controllando volta per volta ciò che è venuto registrato sul disco. Potete fare anche le prime registrazioni dei movimenti, e vi renderete così conto di come funziona il tutto. Vi consigliamo di fare molte incisioni di prova, prima di cominciare coi dati buoni, per poter verificare il corretto funzionamento del programma. Se non avete fatto errori di battuta, tutto andrà per il meglio. A questo proposito vi anticipiamo che il primo disco delle lezioni di BASIC contiene anche questo programma; chi deciderà di seguire il corso d'istruzione su disco risparmierà così la fatica (non indifferente) di dover battere il programma dall'inizio alla fine.

Quando vi sarete sarete familiarizzati con l'uso delle varie possibilità offerte dal menù principale, potete passare a registrare i dati definitivi; prima però ricordate di cancellare il file DATI, altrimenti rimarranno tutte le prove che avete fatto. Basta quindi che scriviate KILL"DATI", e tutto viene azzerato. Non è necessario cancellare anche gli altri files: il programma è fatto in modo che essi non verranno mai letti fino a che non fate delle registrazioni nuove; quelle vecchie verranno così gradualmente cancellate.

BIORITMI - Sig. Roberto Torcini - FIRENZE

```
10 '*********************************
20 '
30 '  PROGRAMMA PER STAMPARE I
40 '  bioritmi
50 '
60 'DI TOM RUGG & PHIL FELDMAN
70 '
80 'RIVISTO E MODIFICATO PER IL
90 'MICRO N.E. DA ROBERTO  TORCINI
100 ' FIRENZE  ( 14/02/82 )
110 '
120 '*********************************
130 '
140 '*** ASSEGNAZIONE VARIABILI ***
150 '
160 CLEAR2000:DEFINTK,L:DEFDBL B,J,M-Z
170 L=0:T=25:P=3.1415926535:CLS
180 '
190 '***INIZIO RICHIESTA DATI***
200 '
210 PRINT:PRINT:PRINT
220 INPUT"come ti chiami   (premi shift)";N$:CLS
230 PRINTTAB(11);"bioritmi":PRINT
240 PRINT"data di nascita"
250 GOSUB 730
260 M1=M :D1=D : Y1=Y
270 GOSUB 830
280 JB=JD
290 PRINT:PRINT"da che giorno vuoi i bioritmi"
300 GOSUB 730
310 FOR H=0 TO 1000
320 NEXT
330 CLS:PRINT:PRINT
340 PRINT"PER UNA CORRETTA ESECUZIONE MI   SERVONO I SEGUENTI DATI:"
350 PRINT
360 INPUT"LA DATA CHE MI DAI E' DI OGGI   (SI/NO)";X$
370 M2=M:D2=D:Y2=Y
380 PRINT:PRINT:INPUT"di che sesso sei (f o m)";S$
390 IF S$="F" OR S$="f" THEN SS$="a" ELSE SS$="o"
400 PRINT
410 INPUT"PER QUANTI GIORNI VUOI CONOSCERE I TUOI BIORITMI";L1
420 GOSUB830
430 JC=JD
440 IFJC>=JBTHEN 510
450 '***CONTROLLA GIUSTEZZA DATA***
460 CLS
470 PRINT"la data di partenza non puo'    essere";
480 PRINT" anteriore alla data di   nascita":PRINT:PRINT"rifai"
490 PRINT:PRINT
500 GOTO 240
510 FORK=1 TO1000:NEXT
520 GOSUB 910
530 CLS :PRINT@235,"** o.k.**"
540 N=JC-JB
550 G=N
560 '*** PONE STRINGA PER I TRE CICLI***
570 V=23:GOSUB 1030:GOSUB1060
580 V=28:GOSUB1030:GOSUB1060
590 V=33:GOSUB1030:GOSUB1060
```

```
600 GOSUB1230
610 '***STAMPA LINEA E AGGIUSTA LUNGHEZZA STAMPA***
620 '***CAMBIANDO (IF) CAMBIA LA LUNGHEZZA***
630 LPRINTC$;TAB(13);L$
640 JC=JC+1:L=L+1:IF L(L1    THEN540
650 CLS:PRINT
660 PRINT "*  F  * PER FINIRE"
670 PRINT "*SPACE* PER CONTINUARE"
680 R$=INKEY$:IFR$=""THEN680
690 IFR$="f" THEN 1390
700 INPUT"PER QUANTI GIORNI ANCORA         VUOI I BIORITMI";L1
710 L=0:CLS:GOTO530
720 '***INPUT DATE E CAMBIO IN FORMATO GIULIANO***
730 PRINT
740 INPUT"giorno(da 1 a 31)";D
750 D=INT(D):IFD(1 OR D)31 THEN 740
760 INPUT"mese (da 1 a 12)";M
770 M=INT(M):IFM(1 OR M)12 THEN 760
780 INPUT"anno";Y
790 Y=INT(Y):IFY(0 THEN 780
800 IF Y)99 THEN 820
810 Y=Y+1900:PRINT"va bene, ";Y
820 RETURN
830 W=FIX((M-14)/12)
840 JD=INT(1461*(Y+4800+W)/4)
850 B=FIX(367*(M-2-W*12)/12)
860 JD=JD+B
870 B=INT(INT(3*(Y+4900+W)/100)/4)
880 JD=JD+D-32075-B
890 RETURN
900 '*** STAMPA INTESTAZIONE***
910 CLS
920 PRINT:PRINT
930 PRINT"sto' calcolandoti i bioritmi"
940 LPRINTTAB(29);"BIORITMI di ";N$
950 LPRINTTAB(28);"nat";SS$;" il";D1;"/";M1;"/";Y1
960 GOSUB 1790
970 LPRINT"--data--";TAB(22);
980 LPRINT"MALE";TAB(38);"0";
990 LPRINTTAB(50);"BENE"
1000 LPRINTTAB(13);
1010 FOR K=1 TO T+T+1:LPRINT CHR$(61);:NEXT K
1020 LPRINT:RETURN
1030 W=INT(N/V):R=N-(W*V)
1040 RETURN
1050 '***SOUBRUTINE PER STRINGA DIAGRAMMA***
1060 IFV()23THEN 1110
1070 L$=CHR$(32):FORK=1TO 5:L$=L$+L$:NEXT
1080 L$=L$+LEFT$(L$,19)
1090 L$=LEFT$(L$,T)+CHR$(108)+RIGHT$(L$,T)
1100 IFV=23 THENC$="F"
1110 IF V=28 THENC$="E"
1120 IFV=33 THENC$="I"
1130 W=R/V:W=W*2*P
1140 W=T*SIN(W):W=W+T+1.5
1150 W=INT(W):A$=MID$(L$,W,1)
1160 IFA$="F" OR A$="E"OR A$="*" THEN C$="*"
1170 IFW=T+T THEN 1210
1180 L$=LEFT$(L$,W-1)+C$+RIGHT$(L$,T+T+1-W)
1190 RETURN
1200 L$=C$+RIGHT$(L$,T+T):RETURN
1210 L$=LEFT$(L$,T+T)+C$:RETURN
1220 '***SOUB. DATA GIULIANA IN D/M/A
```

```
1230 W=JC+68569:R=INT(4*W/146097)
1240 W=W-INT((146097*R+3)/4)
1250 Y=INT(4000*(W+1)/1461001)
1260 W=W-INT(1461*Y/4)+31
1270 M=INT(80*W/2447)
1280 D=W-INT(2447*M/80)
1290 W=INT(M/11):M=M+2-12*W
1300 Y=100*(R-49)+Y+W
1310 A$=STR$(D):W=LEN(A$)-1
1320 C$=MID$(A$,2,W)+"/"
1330 A$=STR$(M):W=LEN(A$)-1
1340 C$=C$+MID$(A$,2,W)+"/"
1350 A$=STR$(Y):W=LEN(A$)-1
1360 C$=C$+MID$(A$,W,2)
1370 RETURN
1380 '***STAMPA DATI FINALI***
1390 GOSUB 1790:GOSUB1790
1400 IFX$="NO" OR X$="no" THEN 1440
1410 LPRINT"al ";D2;"/";M2;"/";Y2;" HAI GIA' VISSUTO ";N;" GIORNI"
1420 GOSUB 1790
1430 PRINT:PRINT
1440 INPUT"VUOI LA SPIEGAZIONE DEI BIORITMI";S$
1450 IF S$="no" GOTO 1650
1460 GOSUB1790:LPRINT" COME LEGGERE    I  B I O R I T M I"
1470 GOSUB 1790
1480 LPRINT"CICLI : F=FISICO   E=EMOTIVO  I=INTELLETTIVO"
1490 GOSUB 1790
1500 LPRINT"Il giorno di cambiamento di segno da BENE a MALE e viceversa"
1510 LPRINT"e' ( GIORNO CRITICO )"
1520 LPRINT"Il giorno piu' critico e' quello che corrisponde a un congiungimento
"
1530 LPRINT"di giorni critici"
1540 LPRINT"Il giorno (critico) del ciclo INTELLETTUALE non e' considerato"
1550 LPRINT"pericoloso, se da solo"
1560 LPRINT"IL MASSIMO GIORNO NEGATIVO si ha nel congiungimento allo **0**"
1570 LPRINT"di tutti e tre i CICLI"
1580 GOSUB 1790:
1590 LPRINT "CICLO EMOTIVO (di 28 giorni)nella parte posiva sarete contenti e ot
timisti,     "
1600 LPRINT"nella parte negativa sarete di cattivo umore e pessimisti"
1610 LPRINT"CICLO FISICO (di 23 giorni) nella parte positiva ci sara' una buona
attivita'    "
1620 LPRINT"fisica * nella parte negativa si avra' poca resistenza con facile af
faticamento"
1630 LPRINT"CICLO INTELLETTUALE(di 33 giorni) nella parte positiva ci sara' gran
de         "
1640 LPRINT"facilita' di apprendimento, nella parte negativa sara' bene evitare
di           affrontare nuovi concetti"
1650 GOSUB 1790:GOSUB 1790:GOSUB 1790
1660 GOSUB 1790:GOSUB 1790
1670 CLS:PRINT:PRINT:
1680 PRINT" VUOI FARE ALTRI CICLI      DI BIORITMI"
1690 PRINT:PRINT" (SI O NO) ?"
1700 U$=INKEY$: IF U$="" GOTO1700
1710 IFU$="S" OR U$="s" GOTO 160
1720 CLS
1730 FOR H=1 TO 40
1740 FOR A=1 TO 35
1750 NEXT:PRINT@235,"* ciao *"
1760 FOR A=1  TO 15
1770 NEXT: CLS :NEXT
1780 END
1790 LPRINT CHR$(13),CHR$(10)
1800 RETURN
```

1) RICHIESTA NOME

```
COME TI CHIAMI    (PREMI SHIFT)?
ROBERTA*
```

2) RICHIESTA DATE

se viene dato, per la data dei bioritmi, una data precedente a quella di nascita segnala errore al quadro 4

```
DATA DI NASCITA

GIORNO(DA 1 A 31)? 7
MESE (DA 1 A 12)? 5
ANNO? 50
VA BENE,  1950

DA CHE GIORNO VUOI I BIORITMI

GIORNO(DA 1 A 31)? 5
MESE (DA 1 A 12)? 12
ANNO? 1949*
```

3) La richiesta avviene per questi motivi:

a) PRIMA DOMANDA: se si rispone SI alla fine della stampa vengono specificati il numero dei giorni vissuti , se si risponde NO salta questa informazione

b) SECONDA DOMANDA: serve per aggiustare sull'intestazione la parola "nato" o "nata", secondo il sesso

C) TERZA DOMANDA: fà la lunghezza della risposta stampata

```
PER UNA CORRETTA ESECUZIONE MI
 SERVONO I SEGUENTI DATI:

LA DATA CHE MI DAI E' DI OGGI
(SI/NO)? SI


DI CHE SESSO SEI (F O M)? F

PER QUANTI GIORNI VUOI CONOSCERE
I TUOI BIORITMI? 6*
```

4) per l'errata risposta al quadro 2 viene segnalato errore e riparte dall'inizio delle domande

```
LA DATA DI PARTENZA NON PUO'
ESSERE ANTERIORE ALLA DATA DI
NASCITA

RIFAI


DATA DI NASCITA

GIORNO(DA 1 A 31)? *
```

5) ripetizione anche di questo quadro per l'errore di cui sopra

```
DATA DI NASCITA

GIORNO(DA 1 A 31)? 7
MESE (DA 1 A 12)? 1
ANNO? 50
VA BENE,  1950

DA CHE GIORNO VUOI I BIORITMI

GIORNO(DA 1 A 31)? 8
MESE (DA 1 A 12)? 2
ANNO? 82
VA BENE,  1982
```

6) come sopra

```
PER UNA CORRETTA ESECUZIONE MI
 SERVONO I SEGUENTI DATI:

LA DATA CHE MI DAI E' DI OGGI
(SI/NO)? SI

DI CHE SESSO SEI (F O M)? F

PER QUANTI GIORNI VUOI CONOSCERE
I TUOI BIORITMI? 6*
```

DOPO L'ESATTA INTRODUZIONE DI DATI
INIZIA LA STAMPA

```
                               BIORITMI di ROBERTA
                              nata :: 7 / 1 / 1950

   --data--            MALE            Ø            BENE
               ==========================================
1) 8/2/82                   EF         !               I
2) 9/2/82                *             !                 I
3) 10/2/82            *                !                  I
4) 11/2/82     FE                      !                  I
5) 12/2/82    FE                       !                  I
6) 13/2/82    EF                       !                  I
```

```
*  F  * PER FINIRE
*SPACE* PER CONTINUARE

*  F  * PER FINIRE
*SPACE* PER CONTINUARE
PER QUANTI GIORNI ANCORA
VUOI I BIORITMI? 5*
```

7) IN questo caso e stato premuto lo spazio e richiesde i dati per la lunghezza del proseguo di stampa

PROSEGUE LA STAMPA

```
14/2/82      E F                   !                  I
15/2/82       E   F                !                I
16/2/82         E    F             !             I
17/2/82           E      F         !          I
18/2/82              E         F        I
```

F COME AL QUADRO 7 - IN questo caso viene premuto F per finire e poichè al quadro 6 alla ~~prima di~~ domanda sulla data di oggi era stato risposto SI stampa:

```
al  8 / 2 / 1982  HAI GIA' VISSUTO  11730  GIORNI
```

```
VUOI LA SPIEGAZIONE DEI BIORITMI
? *SI
```

SE SI RISPONDE SI stampa le spiegazioni

SE SI RISPONDE NO SALTA ALLA FINE

COME LEGGERE I B I O R I T M I

CICLI : F=FISICO E=EMOTIVO I=INTELLETTIVO

Il giorno di cambiamento di segno da BENE a MALE e viceversa
e' (GIORNO CRITICO)
Il giorno piu' critico e' quello che corrisponde a un congiungimento
di giorni critici
Il giorno (critico) del ciclo INTELLETTUALE non e' considerato
pericoloso, se da solo
IL MASSIMO GIORNO NEGATIVO si ha nel congiungimento allo **Ø**
di tutti e tre i CICLI

CICLO EMOTIVO (di 28 giorni)nella parte posiva sarete contenti e ottimisti;
nella parte negativa sarete di cattivo umore e pessimisti
CICLO FISICO (di 23 giorni) nella parte positiva ci sara' una buona attivita'
fisica * nella parte negativa si avra' poca resistenza con facile affaticamento
CICLO INTELLETTUALE(di 33 giorni) nella parte positiva ci sara' grande
facilita' di apprendimento, nella parte negativa sara' bene evitare di
affrontare nuovi concetti

BIORITMI di ROBERTO
nato il 7 / 12 / 1950

--data--	MALE	Ø	BENE
8/2/90	I	E	F
9/2/90	I	E	F
10/2/90	I	E	F
11/2/90	I	E	F
12/2/90	I	E	F
13/2/90	I	E	F
14/2/90	I	F	E
15/2/90	I	E	E
16/2/90	I	F	E
17/2/90	I	F	E
18/2/90	I F	E	
19/2/90	I F	E	
20/2/90	F I	E	
21/2/90	F I	E	
22/2/90	F	*	
23/2/90	F	I	
24/2/90	F E	I	
25/2/90	F E	I	
26/2/90	E F	I	
27/2/90	E F	I	

BIORITMI di AGNESE
nata il 7 / 7 / 1970

--data--	MALE	Ø	BENE
8/5/82	F	I	E
9/5/82	F	I E	
10/5/82		EF I	
11/5/82		E FI	
12/5/82	E	IF	

al 8 / 5 / 1982 HAI GIA' VISSUTO 4327 GIORNI

IL GIOCO DELLA VITA - Sig. G. Forti - FIRENZE

Per scoprire le possibilità di un linguaggio quale il basic ci sono 2 vie:

1° Leggere il relativo manuale.

2° Provare a scrivere un programma e poi andare a vedere sul manuale le ragioni del mancato funzionamento del programma vuoi per un errore di sintassi, vuoi per qualche errore più serio.

Essendo l'autore un appassionato sostenitore del 2° punto (i manuali sono noiosi e spesso, almeno all'inizio, scritti per iniziati) ho provato a scrivere un programma con il basic di N.E. Poi un altro, finché mi sono ricordato di un passatempo tipo solitario inventato da Jhon Conway che avevo visto già programmato in basic.

A questo punto ho fedelmente trascritto col basic di N.E. il programma e dato run! Non girava! Fortunatamente, chi aveva a sua volta trovato questo programma su di un libro, oltre che a citare un articolo di Scientific American che trattava l'argomento, spiegava le 3 regole fondamentali del giuoco. Trovare a questo punto il «baco» nel programma è stato abbastanza facile.

Le 3 regole enunciate da Conway sono le seguenti:

1° SOPRAVVIVENZA = ogni «counter» (che può esser tradotto con pedina) con 2 o 3 pedine adiacenti sopravvive per la generazione successiva.

2° MORTE = ciascuna pedina con 4 o più pedine adiacenti muore per sovrappopolazione. Ogni pedina con una o nessuna pedina adiacente muore per isolamento.

3° NASCITA = ogni cella vuota adiacente a esattamente 3 pedine è una cella in cui nasce una pedina. Una pedina è posta su di essa alla mossa successiva.

IMPORTANTE: nascita e morte sono avvenimenti simultanei nel senso che costituiscono una mossa del gioco della «vita» cioè una «generazione».

Il listato che compare qui sotto «gira» con il basic di N.E. e sul Micro di N.E. con 32K di memoria per iniziare il gioco-passatempo basta far entrare una configurazione qualunque di pedine che viene chiusa digitando OK. Da questo punto in poi si potrà seguire sullo schermo l'evoluzione delle pedine. Buon divertimento.

```
10 CLS
20 DEFINT A,B
30 PRINTTAB(16);"life"
40 X1=1:Y1=1:X2=16:Y2=30
50 DIM A(16,30),B$(16),B(16,30)
60 C=1
70 PRINT"introduci i tuoi dati"
80 LINEINPUT B$(C)
90 IF B$(C)="ok" THEN B$="":GOTO 120
100 C=C+1
110 GOTO 80
120 C=C-1:L=0
130 FOR X=1 TO C-1
140 IF LEN(B$(X)) L  THEN L=LEN(B$(X))
150 NEXT X
160 X1=7-FIX(C/2)
170 Y1=13-FIX(L/2)
180 FOR X=1 TO C
190 FOR Y=1 TO LEN(B$(X))
200 IF MID$(B$(X),Y,1) () " " THEN A(X1+X,Y1+Y)=2:P=P+1
210 NEXT Y
220 NEXT X
230 PRINT:PRINT:PRINT
240 PRINT"gen.";G,"pop.";P;:IF I9 THEN PRINT"no"
250 IF P=0 THEN PRINT"morte":PRINT:PRINT"ancora ?":INPUT A$ :GOTO 60
260 X3=16:Y3=30:X4=1:Y4=1:P=0
```

```
270 G=G+1
280 FOR X=1 TO X1-1:PRINT:NEXT X
290 FOR X=X1 TO X2
300 PRINT
310 FOR Y=Y1 TO Y2
320 IF A(X,Y)=2 OR A(X,Y)=3 THEN A(X,Y)=1: GOTO 340
330 A(X,Y)=0:GOTO 390
340 PRINTTAB(Y)+"*";
350 IF X<X3 THEN X3=X
360 IF X>X4 THEN X4=X
370 IF Y<Y3 THEN Y3=Y
380 IF Y>Y4 THEN Y4=Y
390 NEXT Y
400 NEXT X
410 FOR X=X2+1 TO 15:PRINT:NEXT X
420 X1=X3:X2=X4:Y1=Y3:Y2=Y4
430 IF X1<3 THEN X1=3:I9=-1
440 IF X2>14 THEN X2=14:I9=-1
450 IF Y1<3 THEN Y1=3:I9=-1
460 IF Y2>30 THEN Y2=30:I9=-1
470 FOR X=X1-1 TO X2+1
480 FOR Y=Y1-1 TO Y2+1
490 B(X,Y)=A(X,Y)
500 NEXT Y,X
510 P=0
520 FOR X=X1-1 TO X2+1
530 FOR Y=Y1-1 TO Y2+1
540 C=0
550 FOR I=X-1 TO X+1
560 FOR J=Y-1 TO Y+1
570 IF I=X AND J=Y GOTO 590
580 IF B(I,J)=1 THEN C=C+1
590 NEXT J
600 NEXT I
610 IF C=2 AND B(X,Y)=0 THEN A(X,Y)=0 ELSE A(X,Y)=C
620 IF(B(X,Y)=1 AND A(X,Y)=2) OR C=3 THEN P=P+1
630 NEXT Y,X
640 X1=X1-1:Y1=Y1-1:X2=X2+1:Y2=Y2+1
650 GOTO 240
660 END
```

GIMCANA - Sig. Romano Antonio - FIRENZE

Dopo averlo caricato in memoria e dato il run, il programma si presenta e chiede alcuni limiti: per spostare la macchina si deve digitare a destra il carattere «,»: a sinistra il carattere «.»: se la macchina incontra i cigli della strada si ferma: si raccomanda di non tenere pigiati i tasti di controllo altrimenti il programma si ferma, naturalmente al rilascio del tasto riprende.

```
100 '
110 '
120 CLEAR200:RANDOM:Q=0:Q$=""
130 LC=.45:RC=1-LC
140 LS$=CHR$(191)+CHR$(32):RS$=CHR$(32)+CHR$(191)
150 L$=",":R$="."
160 LT$=CHR$(184)+CHR$(135):RT$=CHR$(139)+CHR$(180)
170 B=32:EL=2:ER=32:C1=175:C2=159: DEFINTV,W
180 GOSUB800
200 PRINT:T=0:N=0
```

```
210 INPUT"larghezza della strada (4-12)";W
220 IFW(40RW)12 THEN200
230 PRINT:PRINT"condizioni di visibilita'"
240 PRINT"  1 - PESSIME"
250 PRINT"  2 - CATTIVE"
260 PRINT"  3 - DISCRETE"
270 PRINT"  4 - BUONE":PRINT
280 INPUT "visibilita' (1-4)";V
290 IFV(1OR V)4 THEN 280
300 N=N+1:L=8 :R=L+W+2:Z=-5120+64*V
310 C=INT((L+R)/2):H=0
320 FORJ=1TO16:Q$=INKEY$:GOSUB600:NEXT
330 GOSUB700
350 H=H+1:Q=RND(0.1):IFQ)RCANDR(ER THEN GOSUB640:GOTO400
360 IFQ(LC ANDL)EL THEN GOSUB620:GOTO400
370 GOSUB600
400 ZC=Z+C:ZP=ZC+1:Q$=INKEY$:IFQ$=L$ THENC=C-1
410 IFQ$=R$ THENC=C+1
420 IF PEEK(ZC)=B AND PEEK(ZP)=B THEN POKEZC,C1:POKEZP,C2:GOTO350
430 FORJ=1TO8:Q$=INKEY$:POKEZC,B:POKEZP,B:FORK=1TO50
440 NEXT:POKEZC,C1:POKEZP,C2:FORK=1TO50:NEXT:NEXT
450 M=H*5:T=T+M:PRINT:PRINT"HAI FATTO";M;" CHILOMETRI PER UN"
460 PRINT"totale di";T;" km. in";N;"giorn";
465 IFN=1THENPRINT"o"ELSEPRINT"i"
470 PRINT"batti 'c' per continuare"
480 PRINT"       'r' per ripartire"
490 PRINT"       'a' per abbandonare"
500 Q$=INKEY$:IFQ$="c"THEN300
510 IFQ$()"r"ANDQ$()"a"THEN500
520 PRINT:PRINT"media di";T/N;" km.al giorno"
540 IFQ$="r" THEN200
550 END
600 PRINTTAB(L);RS$;TAB(R);LS$
610 RETURN
620 PRINTTAB(L);LT$;TAB(R-1);LT$
630 L=L-1:R=R-1:RETURN
640 PRINTTAB(L+1);RT$;TAB(R);RT$
650 L=L+1:R=R+1:RETURN
700 P=Z+C:GOSUB940:P=Z+C-(-5120-20)
710 Q$=CHR$(191)+CHR$(B)+"0"+CHR$(B)+CHR$(191)
720 P=P-94 :PRINT@P,STRING$(5,176);
730 FORJ=1TO5:P=P+32:PRINT@P,Q$;:NEXT
740 PRINT@P+32,STRING$(5,131);
750 FORJ=1TO300:NEXT
760 FORJ=4 TO0 STEP-1:FORK=1TO300:NEXT
770 PRINT@P+2-32*J,CHR$(143);:NEXT:PRINT@448,CHR$(B)
780 RETURN
800 DIMD(9):L= 0:R=13:CLS
810 FORJ=1TO 9:READD(J):NEXT
820 DATA 32,82,65,76,76,89,32,32,32
830 GOSUB600
840 GOSUB640
850 FORJ=1TO2:GOSUB620:NEXT
860 FORJ=1TO10:GOSUB640:NEXT
870 P=-5120:POKEP,C1:POKEP+1,C2
880 FORJ=1TO250:NEXT
890 P=P+33:GOSUB940
900 FORJ=1TO2:P=P+33:GOSUB940:NEXT
910 P=P+33:GOSUB940
920 FORJ=1TO9:P=P+33:GOSUB940
930 POKEP,D(J):POKE P+1,B:NEXT
940 POKEP,C1:POKEP+1,C2:FORK=1TO50:NEXT:RETURN
READY
)IL PROGRAMMA E STATO MODIFICATO
 PER IL NE-COMPUTER DA ROMANO AN
TONIO*
```

OSTACOLI - Sig. Antonio Romano - FIRENZE

Il programma si presenta chiedendo il nome dei due giocatori di destra e di sinistra. Il gioco consiste nel muovere due pedine e di cercare di chiudere l'avversario in modo che vada ad urtare contro un ostacolo che può essere il bordo del campo o anche la scia lasciata dalle due pedine. Le due pedine all'inizio si muovono verso l'altro per cambiare la direzione di marcia il giocatore ha a disposizione le seguenti mosse:

Giocatore di destra.

Per andare in alto = «O»
Per girare a sinistra = «K»
Per andare in basso = «.»
Per girare a destra = «;»

Giocatore di sinistra.

Per andare in alto = «W»
Per girare a sinistra = «A»
Per andare in basso = «X»
Per girare a destra = «D»

I tasti non devono essere tenuti premuti ma solo «sfiorati».

```
100 'OSTACOLI
110 '
120 CLEAR200:DEFINT A-Z:RANDOM:GOSUB600
125 CLS:PRINT:PRINT
130 PRINTTAB(13);"ostacoli"
140 PRINT:PRINT
150 PRINTTAB(3);"batti un tasto per partire"
155 W=2:Z=191:Z$=STRING$(W,CHR$(Z)):A=234:B=245
160 A$=CHR$(183)+CHR$(179)+CHR$(187):B$=STRING$(W,CHR$(153))
165 S=-5120:E$=STRING$(W,CHR$(173)):AD=1:BD=1
170 GOSUB900: GOSUB950: FORJ=1TO10:R$=INKEY$:NEXT
180 R$=INKEY$:IF R$=""THEN180
190 CLS
200 GOSUB 950:GOSUB900:FORJ=1TO10:R$=INKEY$:NEXT
210 X=A:D=AD:GOSUB1000
220 AR=R:A=X
230 X=B:D=BD:GOSUB1000
240 BR=R:B=X
245 IF AR=1 OR BR=1 THEN 400
250 GOSUB 900
260 FORJ=1TO8:R$=INKEY$
270 IFR$="w"THEN AD=1
280 IFR$="x"THEN AD=2
290 IFR$="a"THEN AD=3
300 IFR$="d"THEN AD=4
310 IFR$="o"THEN BD=1
320 IFR$="."THENBD=2
330 IFR$="k" THENBD=3
340 IFR$=";"THEN BD=4
350 NEXT
360 GOTO210
400 GOSUB 700:X=A
410 IFBR=1 THEN X=B
420 FORJ=1TO15
430 PRINT@X,Z$;
440 FORK=1TO200:NEXT
450 PRINT@X,"  ";
460 FORK=1TO200:NEXT
470 NEXT
490 GOTO125
600 CLS:PRINTTAB(8);"ostacoli":PRINT
610 INPUT"nome del giocatore di sinistra";AN$
620 INPUT"giocatore di destra";BN$
630 RETURN
700 PRINT@480," ";
710 IFAR<>1 ORBR<>1 THEN730
```

```
720 PRINT"avete perso entrambi ! ";:RETURN
730 R$=AN$:IF AR=1 THENR$=BN$
740 PRINT"ha vinto ";R$;
750 RETURN
900 PRINT@A,A$;:PRINT@B,B$;:RETURN
950 FOR X=0TO30 STEP 3
960 PRINT@X,E$;:PRINT@X+448,E$;:NEXT
970 FORX=0TO 448 STEP32
980 PRINT@X,E$;:PRINT@X+30,E$;:NEXT
990 RETURN
1000 IFD=1 THENX=X-32
1010 IFD=2 THENX=X+32
1020 IFD=3 THENX=X-3
1030 IFD=4 THENX=X+3
1040 R=0
1041 PRINT@448,A;S+X
1050 IF PEEK(S+X)<>32THENR=1
1060 RETURN
```

SOUBRUTINE GRAFICA 6 - Sig. Luca Taraborrelli - BOLOGNA

Il programma permette il ripristino delle possibilità grafiche della scheda video del Computer di Nuova Elettronica. Infatti, con l'introduzione del NE-DOS-BASIC, le istruzioni SET, RESET rendono disponibile una gestione X,Y del video limitatamente a 32x16 punti contro i 64x48 utilizzabili con il vecchio BASIC da 5,5K.

Per permettere il completo adattamento dei programmi sviluppati per i primo Basic con il nuovo, si è studiato un semplice programma di gestione in X,Y del video utilizzabile come soubrutine. Il programma è scritto a partire dalla linea 59000 in modo da poter essere facilmente attaccato in fondo a qualsiasi programma che ne fa uso utilizzando l'istruzione MERGE.

Per chiamare il programma è sufficiente una GOSUB 60000 dopo aver definito una serie di VARIABILI INGRESSO per la funzione richiesta.

Le variabili richieste sono:

-X- compresa tra 0 e 63, rappresenta la coordinata ORIZZONTALE.

-Y- compresa tra 0 e 47, rappresenta la coordinata VERTICALE.

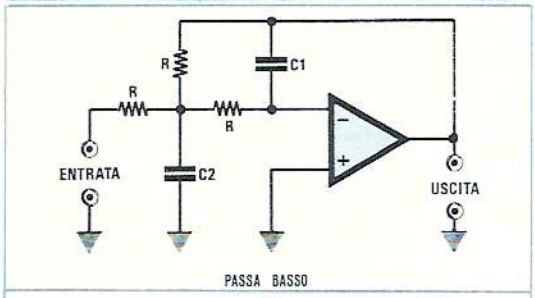
-F- assume i valori 0 1 2 per le funzioni SET, RESET, POINT.

Con F = 3, il programma ritorna con la variabile P = 0 se il punto è SPENTO e P = 1 se è ACCESO.

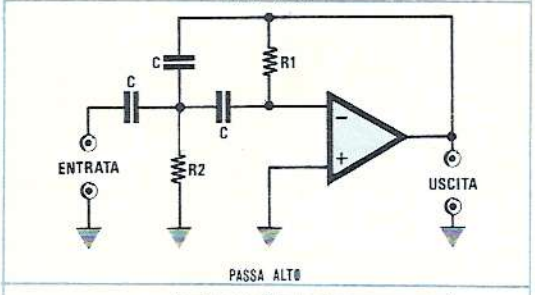
Copiate il programma dalla 59000 alla 60130 e salvatelo su disco con il nome GRAPHIC usando il comando SAVE «GRAPHIC», A.

```
59000 STOP
60000 IF(X<0)OR(X>63)OR(Y<0)OR(Y>47)THEN RETURN
60010 GOSUB60050:ON F GOTO 60030,60040
60020 POKE XY,XKORYJ:RETURN
60030 POKE XY,XK AND(255-YJ):RETURN
60040 P=0:IF(128+YJ)=(XKAND(128+YJ))THENP=1:RETURN
60050 XX=X/2:YY=(47-Y)/3
60060 XH=INT((XX-INT(XX))*2+1)
60070 YH=INT((YY-INT(YY))*10/3+1)
60080 XY=(INT(XX)+INT(YY)*32)-5120
60090 IFYH=2 THEN YJ=4 ELSE YJ=16
60100 IFYH=1 THEN YJ=1
60110 IFXH=1 THEN YJ=32/YJ ELSE YJ=32/(YJ*2)
60120 XK=PEEK(XY):IF((XK<128)OR(XK>191))THENXK=128
60130 RETURN
```

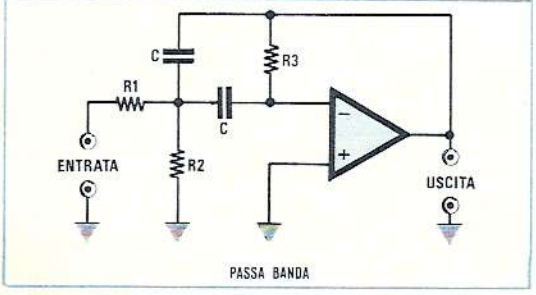
CALCOLO FILTRI ATTIVI CON OPERAZIONALI - Sig. Luca Taraborelli - Bologna



PASSA BASSO



PASSA ALTO



PASSA BANDA

Chi deve progettare dei filtri «passa basso» «passa alto» «passa banda», utilizzando degli amplificatori operazionali configurati come vedesi negli schemi elettrici riportati a lato, potrà con tale programma ottenere immediatamente i valori da assegnare ai condensatori e alla resistenze.

Dopo il RUN, sul video viene chiesto quale dei tre filtri si desidera calcolare, una volta scelto 1-2-3 (dopo aver selezionato il numero occorre pigiare il tasto RETURN) il computer chiederà quindi la frequenza (anche la banda passante per il filtro «passabanda») ed un valore da assegnare arbitrariamente (a una resistenza od a un condesatore); dopo averlo digitato si pigierà il tasto RETURN e si otterranno i valori richiesti.

NOTA = la frequenza va espressa in HERTZ, le resistenze in KILOHOM e in condensatori in PICOFARAD.

Qui sotto possiamo vedere qualche esempio di stampa dei tre filtri calcolati con diverse frequenze.

```
****** PASSA BASSO ******

F= 300   HERTZ                    F= 50000   HERTZ
R= 100   KILOHM                   R= 10   KILOHM
C1= 2500.87   PICOFARAD           C1= 150.052   PICOFARAD
C2= 11253.9   PICOFARAD           C2= 675.236   PICOFARAD
GUADAGNO=1                        GUADAGNO=1
```

```
****** PASSA ALTO ******

F= 800   HERTZ                    F= 2000   HERTZ
C= 100000   PICOFARAD             C= 1000   PICOFARAD
R1= 1.40674   KILOHM              R1= 56.2696   KILOHM
R2= 2.81348   KILOHM              R2= 112.539   KILOHM
GUADAGNO=1                        GUADAGNO=1
```

```
****** PASSA BANDA ******

F= 50   HERTZ                     F= 2400   HERTZ
B.PASS.= 10   HERTZ               B.PASS.= 300   HERTZ
C= 100000   PICOFARAD             C= 10000   PICOFARAD
R1= 159.155   KILOHM              R1= 53.0515   KILOHM
R2= 3.24805   KILOHM              R2= .417729   KILOHM
R3= 318.309   KILOHM              R3= 106.103   KILOHM
GUADAGNO= 1.01321                 GUADAGNO= 1.12579
```

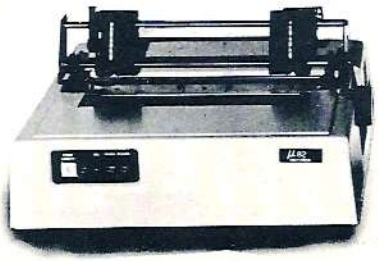
```
10 CLEAR300:CLS:REVON
20 PRINT"******* FILTRI ATTIVI ******":REVOFF
30 PRINT:PRINT"
  1.= PASSA BASSO
  2.= PASSA ALTO
  3.= PASSA BANDA
40 PRINT@384,CHR$(31)
50 PRINT@416,"-->";:INPUTA:IF(A<0)OR(A>3)THEN40
60 PI=6.2832:E=SQR(2)/2
70 CLS:ON A GOTO 80,190,300
80 PRINT"****** PASSA BASSO ******"
90 PRINT:INPUT"FREQUENZA IN HERTZ";F:W=PI*F
100 INPUT"RESISTENZA (R) IN KILOHM";R:R1=R/1000
110 C1=1000000*E/(R1*1.5*W):C2=1000000*1.5/(R1*E*W)
120 PRINT@64,CHR$(31)
130 PRINT"F=";F;" HERTZ"
140 PRINT"R=";R;" KILOHM"
150 PRINT"C1=";C1;" PICOFARAD"
160 PRINT"C2=";C2;" PICOFARAD"
170 PRINT"GUADAGNO=1"
180 GOSUB440:ONXGOTO70,10,120
190 CLS:PRINT"****** PASSA ALTO ******"
200 PRINT:INPUT"FREQUENZA IN HERTZ";F:W=PI*F
210 INPUT"CAPACITA' IN PICOFARAD";C:C1=C/1000000
220 R1=1000*E/(C1*W):R2=1000/(C1*E*W)
230 PRINT@64,CHR$(31)
240 PRINT"F=";F;" HERTZ"
250 PRINT"C=";C;" PICOFARAD
260 PRINT"R1=";R1;" KILOHM"
270 PRINT"R2=";R2;" KILOHM"
280 PRINT"GUADAGNO=1"
290 GOSUB440:ON X GOTO190,10,230
300 CLS:PRINT"****** PASSA BANDA ******"
310 PRINT:INPUT"FREQUENZA IN HERTZ";F:W=PI*F
320 INPUT"BANDA PASS. IN HERTZ";F1:W1=PI*F1
330 INPUT"CAPACITA' IN PICOFARAD";C:C1=C/1000000
340 R1=1000/(C1*W1):R2=1000*W1/(C1*(2*W^2-W1^2)):R3=2000/(C1*W1)
350 PRINT@64,CHR$(31)
360 PRINT"F=";F;" HERTZ"
370 PRINT"B.PASS.=";F1;" HERTZ"
380 PRINT"C=";C;" PICOFARAD"
390 PRINT"R1=";R1;" KILOHM"
400 PRINT"R2=";R2;" KILOHM"
410 PRINT"R3=";R3;" KILOHM"
420 PRINT"GUADAGNO=";R3/C*R3
430 GOSUB440:ONXGOTO300,10,350
440 PRINT@352,:INPUT"ANCORA (S/N)
('P' PER LA STAMPA)";A$
450 IF(A$="p")OR(A$="P")THEN490
460 X=1:IF(A$="s")OR(A$="S")THEN RETURN
470 X=2:IF(A$="n")OR(A$="N")THEN RETURN
480 X=3:RETURN
490 L=-5120:FORI=0TO11:FORH=0TO31
500 A=PEEK(L+I*32+H):IF A>31 THEN 520
510 A=A+64
520 A$=MKI$(A):LPRINTA$;:NEXTH:LPRINT:NEXTI
530 LPRINT:LPRINT" ":PRINT@384,STRING$(96," ");:GOTO440
```

ANNOTAZIONI

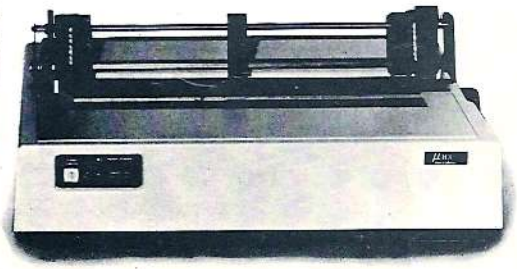
μL: la stampante ideale per ogni sistema a μP

La serie delle stampanti Microline della OKI, oltre ad essere veloci, silenziose e robuste si adattano ad ogni tipo di microcomputer disponendo di interfaccia seriale o parallelo.

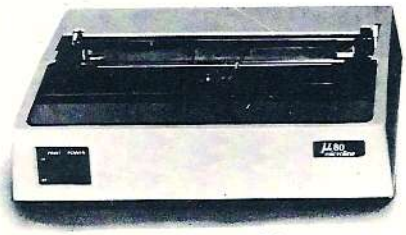
Tutti i modelli presentati sono a percorso ottimizzato e risultano affidabili nel tempo, inoltre potrete acquistarli a prezzi altamente concorrenziali rispetto ad altre marche con analoghe caratteristiche.



modello uL 82



modello uL 83



modello uL 80

Modello uL 80 = 80 colonne, 80 CPS monodirezionale compreso trattore
L. 850.000 IVA inclusa

Modello uL 82 = 80 colonne bidirezionale con logica selettiva di percorso
L. 1.270.000 IVA inclusa

Modello uL 83 = 132 colonne, 120 cps bidirezionale su carta da 38 cm.
L. 1.750.000 IVA inclusa

Tutti i modelli illustrati sono reperibili presso Nuova Elettronica e Concessionari.

MOBILE modello 500
alt. 72 largh. 120 prof. 140 mm.
prezzo L. 7.000

MOBILE modello 501
alt. 72 largh. 160 prof. 140 mm.
prezzo L. 7.500

MOBILE modello 502
alt. 105 largh. 200 prof. 140 mm.
prezzo L. 8.500

MOBILE modello 503
alt. 80 largh. 200 prof. 180 mm.
prezzo L. 9.000

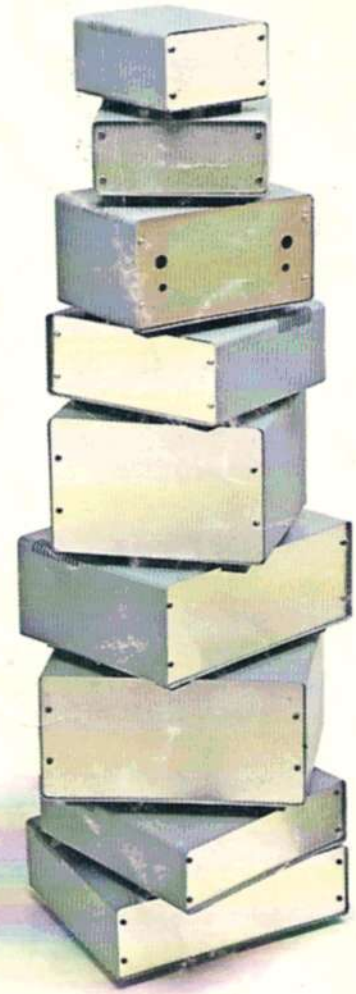
MOBILE modello 504
alt. 135 largh. 212 prof. 190 mm.
prezzo L. 12.500

MOBILE modello 505
alt. 71 largh. 212 prof. 230 mm.
prezzo L. 11.000

MOBILE modello 506
alt. 106 largh. 260 prof. 212 mm.
prezzo L. 13.000

MOBILE modello 507
alt. 80 largh. 260 prof. 220 mm.
prezzo L. 12.000

MOBILE modello 508
alt. 131 largh. 260 prof. 200 mm.
prezzo L. 13.500



CARATTERISTICHE MOBILI SERIE M.500

Verniciatura a fuoco, buccia di arancio ad alta resistenza e antigraffiante.
Pannello frontale in alluminio anodizzato.
Telaio interno zincato oro già forato per facilitare il fissaggio dei circuiti stampati.
Sistema di aereazione appositamente studiato per un rapido smaltimento del calore interno.
Ogni mobile è completo di quattro piedini in gomma dura antisdrucciolevole.

per ogni progetto un giusto mobile ad un giusto prezzo

Mobile per luci Psichedeliche per Auto LX.474 completo di pannello anteriore già forato e serigrafato L. 7.000
Mobile per Luci Psichedeliche a Diodi Led LX.476 completo di pannello anteriore già forato e serigrafato L. 8.000

È disponibile, per il pannello frontale dell'organo, la serie completa dei deviatori professionali, visibili in foto.

Mobile metallico per ORGANO ELETTRONICO LX.462 completo di pannello laterale sinistro in alluminio anodizzato già forato L. 38.000