OLIVETTI M 10: guida all'uso

Massimo Mangia

GRUPPO EDITORIALE JACKSON

OLIVETTI M10:
guida all'uso

di
Massimo Mangia

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

Il Gruppo Editoriale Jackson ringrazia per il prezioso lavoro svolto nella stesura dell'edizione originale le signore Francesca Di Fiore, Cristina De Venezia e Daria Gianni.

Fotocomposizione: Composit s.a.s. — Via Giordano Bruno, 8 — 56100 Pisa

Stampato in Italia da: S.p.A. Alberto Matarelli — Milano — Stabilimento Grafico

Sommario

Capitolo 0

Introduzione

Nonostante il gran parlare e discutere di cui è stato fatto oggetto, il computer è ben lungi dal diventare un oggetto di uso comune, un qualcosa insomma di familiare. Certamente ha contribuito a questa situazione una circostanza che non esito a definire paradossale. È infatti accaduto che si è reso disponibile un prodotto, il personal computer, senza che fosse ben chiaro alla gente l' utilitità, lo scopo per cui era stato creato.

"È molto bello, ma a cosa serve?", "Cosa posso farci?", sono alcune delle domande più comuni che ho sentito rivolgere a commercianti di vario genere. Domande il più delle volte senza risposta.

Questo problema si è ripresentato più di recente con la comparsa dei primi calcolatori portatili, quale ad esempio l' Olivetti M10. Effettivamente, visto che molti ancora ignorano l' utilità dell' home o del personal computer, figurarsi se pensano di acquistare un "portatile". Ma per portarselo dove?, e per farci cosa? A queste e ad altre domande ho tentato di rispondere con questo libro, rivolto sia a chi già possiede un Olivetti M10, magari comprato nell' incertezza di cui sopra, sia a chi ancora non ce l' ha, e vuol saperne qualcosa di più.

Pertanto questo testo nasce, non già come un sostituto o un doppione della guida operativa della Olivetti, ma come un qualcosa di profondamente diverso, al più, volendo fare confronti a tutti i costi, come un compendio ad essa. Di conseguenza in questo libro non troverete né informazioni su come, ad esempio, cambiare le pile nel computer, né nozioni su come imparare a programmare, né ancora spiegazioni riguardo la tastiera. Viceversa questa guida vuole rivelarvi le notevoli potenzialità dell' Olivetti M10, nella sua duplice veste di calcolatore programmibile e programmato. In poche parole, questo è un libro di idee, suggerimenti ed esempi, una scommessa contro l' attuale "buio" che circonda l' argomento. Vedere il computer (nella fattispecie l' Olivetti M10) in un ottica differente, sotto un' altra luce: è questo in definitiva l' obiettivo che mi sono posto scrivendo questo libro. A voi giudicare se l' abbia centrato, oppure no.

Due parole infine sulla struttura del testo. Considerando la già menzionata duplice natura dell' M10, ho ritenuto appropriato suddividere il libro in due parti distinte.

La prima è interamente dedicata al linguaggio basic Microsoft. Nella sua stesura ho supposto che il lettore avesse almeno qualche nozione di programmazione, onde poter trattare l' argomento nel modo da me desiderato. Otto sono i capitoli in cui è suddivisa questa prima parte. Ognuno di essi è dedicato ad un aspetto particolare del linguaggio: comandi di sistema, di editing, di I/O, ecc... Ho preferito questa organizzazione in quanto, a mio avviso, più omogenea di altre, più logica ed infine più funzionale. Ogni istruzione è spiegata in dettaglio, insieme ad alcuni esempi e suggerimenti sul loro uso.

La seconda parte tratta invece dei programmi applicativi integrati nel computer. Ad ognuno di essi è dedicato un intero capitolo, nel quale oltre ad un' esauriente spiegazione riguardo il loro funzionamento, troverete molte proposte ed esempi su possibili usi. Vedrete così come può essere impiegato l' Olivetti M10 a casa, a scuola, in viaggio, in ufficio, ecc...

Concludono infine il testo una serie di utili appendici riassuntive.

Buona lettura, dunque!

RINGRAZIAMENTI

Molte persone mi hanno aiutato nella stesura di questo libro. Desidero pertanto ringraziare: mio fratello Mariano, che mi ha suggerito l' idea della sua realizzazione, Franco Caccamisi che ha gentilmente messo a disposizione il computer con cui ho scritto il testo, Paolo e Rita Pisanti per la comprensione e la disponibilità mostrate durante il mio soggiorno sulla Blue Moon, la Composit di Brunetto Casini per il prezioso aiuto prestatomi durante la composizione del libro.

Un particolare riconoscimento va inoltre ai familiari, agli amici e a tutti coloro che mi sono stati vicini e che mi hanno sostenuto durante la sua stesura.

Capitolo 1

Il menù principale

Il computer portatile Olivetti M10 è la soluzione ideale per hobbisti, studenti, professionisti, scrittori, giornalisti, agenti di commercio ecc..., di tutte cioè quelle persone che necessitano di un calcolatore potente, affidabile, e perché no?, portatile. L' M10 è disponibile in due versioni: rispettivamente con (M10 - MODEM) e senza modem integrato, ed in numerosi modelli che si differenziano per la capacità della memoria RAM e per la nazionalità della tastiera. I modelli disponibili sono pertanto:

M10 senza modem, tastiera americana, 8 Kb di memoria RAM.
M10 senza modem, tastiera americana, 24 Kb di memoria RAM.
M10 senza modem, tastiera italiana, 8 Kb di memoria RAM.
M10 senza modem, tastiera italiana, 24 Kb di memoria RAM.
M10 con modem, tastiera americana, 8 Kb di memoria RAM.
M10 con modem, tastiera americana, 24 Kb di memoria RAM.

non essendo disponibili i modelli con modem e tastiera italiana.

I modelli equipaggiati con il modem sono particolarmente indicati per coloro che adoperano spesso il computer come terminale remoto. Le differenze tra questi modelli e quelli che invece ne sono sprovvisti, sono limitate alla disponibilità di alcune funzioni aggiuntive del programma integrato TELCOM.
La diversa quantità di memoria RAM disponibile non comporta invece alcuna differenza tra i vari modelli, salvo ovviamente la minore o maggiore capacità di memorizzazione di dati e programmi.
La nazionalità della tastiera comporta invece molte differenze tra i caratteri disponibili. La tastiera americana, oltre ad una diversa disposizione delle lettere e dei numeri (questa tastiera è anche chiamata QWERTY dal nome dei primi 6 tasti), possiede dei carat-

teri non disponibili su quella italiana, e viceversa. Nella tastiera americana mancano ad esempio le vocali accentate, il simbolo della lira, quello del paragrafo, ecc.... La tabella seguente mostra le differenze e le equivalenze tra le due tastiere:

CODICE ASCII	EQUIVALENTE	
DECIMALE	USA	ITALIA
35	#	£
36	$	$
64	@	§
91	[	°
92	\	ç
93	]	é
96	`	ù
123	{	à
124	|	ò
125	}	è
126	~	ì

Figura 1.1

Nei capitoli successivi sono stati adottati i caratteri italiani (lettere accentate, simbolo del paragrafo, c con cediglia, ecc....), eccezion fatta per il simbolo della lira, al quale è stato preferito il carattere #.

Un' altra differenza esistente tra i modelli americani ed italiani è il formato della data del sistema. Nel modello USA ha il seguente formato:

DATE$ = "mese / giorno / anno"

mentre nel modello italiano:

DATE$ = "giorno / mese / anno"

Altre differenze esistenti tra i vari modelli sono illustrate nel seguito del libro.

1.1 Architettura hardware

L' Olivetti M10 è molto diverso da altri personal computer, sia per la particolare tecnologia impiegata che per la sua originale architettura hardware. L' M10 è stato infatti realizzato in tecnologia CMOS, una particolare tecnica che impiega componenti elettronici a bassissimo consumo di energia. Questo ha permesso di alimentare la memoria RAM con batterie ricaricabili al Ni-Cd, che la rendono non volatile: anche a macchina spenta la memoria conserva le informazioni ed i programmi in essa contenuti.
Anche il video è stato realizzato con tecnologia d' avanguardia, impiegando un display a cristalli liquidi (LCD) di 8 righe per 40 colonne. Il suo contrasto può essere regolato mediante una manopola posta sul lato destro del computer, oppure inclinando tutto il display.
Completano l' originale architettura hardware dell' M10 una numerosa serie di interfacce, come illustrato in figura 1.2

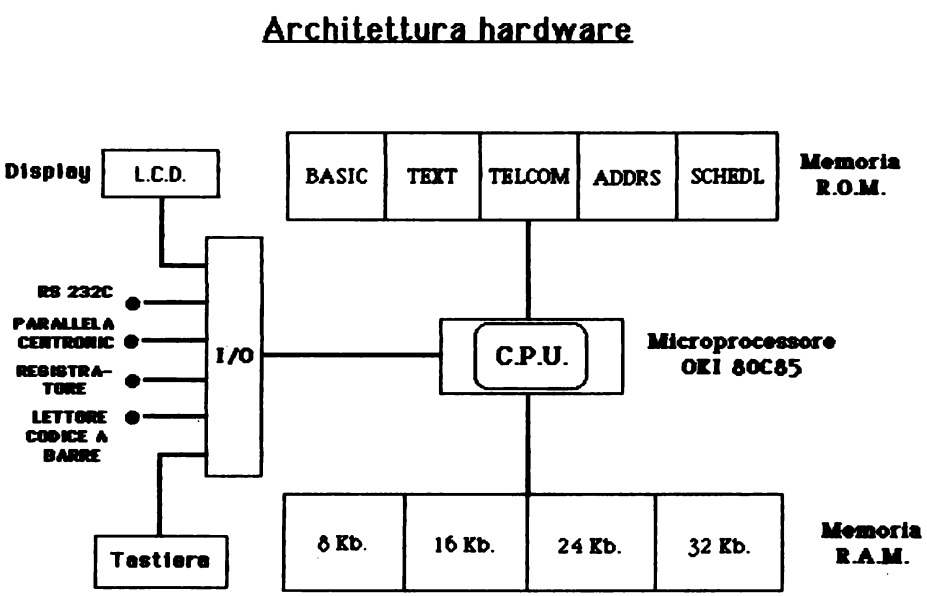


Figura 1.2

1.2 Architettura software

La diversità tra l' Olivetti M10 e gli altri personal computer si manifesta anche nella sua particolare architettura software. La memoria ROM, della capacità di 32 Kb., contiene cinque programmi integrati:

- L' interprete BASIC, che consente di creare, memorizzare ed eseguire programmi

- TEXT, per creare ed editare dei testi

- TELCOM, per inviare e ricevere dati a distanza, nonché per trasformare l' M10 in un terminale remoto

- ADDRS, per creare e gestire un indirizzario elettronico

- SCHEDL, per creare e gestire un' agenda elettronica.

L' Olivetti M10 possiede, in altri termini, oltre all' interprete basic, quattro utili programmi che risiedono nella sua memoria ROM.

Queste cinque funzioni integrate condividono le risorse dell' M10 e le strutture dati di tipo testo. È possibile infatti creare un testo con TEXT, gestirlo con SCHEDL ed infine inviarlo ad un altro computer con TELCOM.
Alla prima funzione (l' interprete basic) è interamente dedicata la prima parte del libro, mentre alle rimanenti quattro è dedicata la seconda.

1.3 Accessori e periferiche

L' Olivetti M10 possiede una ricca serie di accessori e di periferiche collegabili. Tra gli accessori disponibili, particolarmente utili risultano:

- Cavo per stampante parallela
- Cavo RS 232C diritto
- Cavo RS 232C incrociato
- Cavo modem (solo per M10 modem)
- Chip di espansione della memoria RAM (8 Kb.)

Con il primo cavo è possibile collegare all' M10 una stampante dotata di interfaccia parallela Centronics, con il secondo ed il terzo altri computer o periferiche dotate di intefaccia seriale RS 232C, con il quarto connettere l' M10 Modem alla linea telefonica, mentre i moduli di espansione consentono di aumentare la quantità di memoria RAM, fino ad un massimo di 32 Kb.

Coerentemente con la filosofia del computer, sono state realizzate alcune periferiche portatili che aumentano la versatilità del sistema.

Il PL 10 è un microplotter a 4 colori, dotato di alimentazione autonoma con batterie ricaricabili ed alimentatore incorporato, che scrive su carta di 11 centimetri e mezzo una riga di massimo 80 caratteri. Il PL 10 si collega tramite cavo ed interfaccia parallela. Questo microplotter consente di scrivere caratteri in diverse dimensioni ed in diversi orientamenti, nonché tracciare linee continue o tratteggiate.

L' MC 10 è un accoppiatore acustico con modem integrato, che consente di inviare e ricevere dati in full duplex tramite normale linea telefonica ad una velocità di 300 baud (bit al secondo). L' MC 10, destinato solamente al modello senza modem, è dotato di alimentazione a pile incorporate o con alimentatore esterno. L' accoppiatore si connette all' M10 tramite cavo ed interfaccia RS 232C.

Sono inoltre collegabili al computer un lettore di codice a barre, e registratore a cassette. Per ulteriori informazioni su interfacciamento e periferiche, consultare l' appendice A.

1.4 Il menù principale

All' accensione del sistema, appare sul display il menù principale. La prima riga del video è occupata a partire da sinistra, dal giorno, il mese, l' anno, il giorno della settimana, l' ora, i minuti, i secondi ed il marchio Microsoft, che ha realizzato il software di base. Le successive sei righe sono utilizzate dal sistema per mostrare l' indice, o directory, dei programmi o dei file presenti nella memoria RAM, più le cinque funzioni integrate contenute nella ROM. L' ultima riga contiene a sinistra la parola *Select:*, mentre a destra è visualizzata la quantità di memoria disponibile.

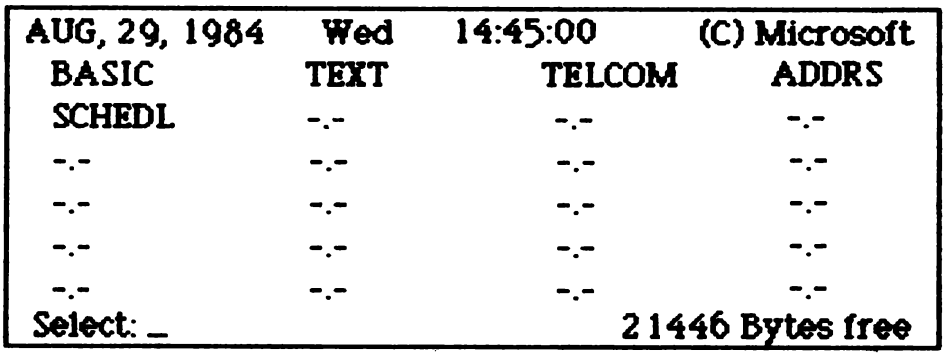


Figura 1.3

All' inizio il cursore è posizionato su BASIC, ma può essere agevolmente spostato

adoperando i tasti movimento cursore, contraddistinti dalle frecce. Per selezionare una funzione integrata, un programma (file di tipo .BA) o un file di testo (file di tipo .DO) occorre posizionare il cursore su di esso e quindi premere il tasto ENTER, oppure digitare il nome del file concluso dal tasto ENTER. Se si è selezionata una funzione integrata, si entra immediatamente in essa, se è stato selezionato un programma basic ne viene subito lanciata l' esecuzione, se infine è stato scelto un file di testo questo viene passato a TEXT.

PARTE PRIMA
IL BASIC

Capitolo 2

Caratteristiche generali

L' interprete basic implementato nell' Olivetti M10 è un' estensione di questo diffuso linguaggio, grazie al quale l' utente può realizzare ed eseguire i propri programmi applicativi. Per accedere al basic dal menù pricipale, è necessario posizionare il cursore sul nome BASIC e premere il tasto ENTER, oppure digitare *BASIC [ENTER]* (anche in minuscolo), oppure ancora selezionare tramite cursore o nome un qualsiasi programma basic contenuto in memoria. Nei primi due casi si accederà allo stato comandi del basic, mentre nel terzo si farà partire l' esecuzione del relativo programma.

Stato comandi

Lo stato comandi è la modalità operativa nella quale è possibile memorizzare linee di programma o eseguire comandi in modo immediato. Questo stato è evidenziato dalla presenza sul video del messaggio Ok, che avvisa l' utente che il sistema è pronto a ricevere ed eseguire i comandi. Una descrizione dettagliata dei comandi, delle istruzioni e delle funzioni del linguaggio è fornita nei capitoli 3, 4, 5, 6, 7, 8 e 9.

Stato di elaborazione

L' esecuzione di un programma o di una linea immediata fa passare il sistema in stato di elaborazione. In questa modalità l' interprete basic interpreta ed esegue la linea immediata o le righe del programma contenuto in memoria di lavoro. Al termine dell' elaborazione viene nuovamente passato il controllo allo stato comandi.

Stato di editor (funzione TEXT)

L' esecuzione dell' istruzione EDIT fa entrare il sistema in stato di editor. Questa modalità operativa, che utilizza la funzione integrata TEXT, viene richiamata per modificare e correggere agevolmente il programma contenuto nella memoria di lavoro. Al termine dell' editing del programma, la pressione del tasto funzione F8 riporta il sistema in stato comandi.

Dopo aver descritto le tre modalità operative dell' M10, è necessario chiarire la differenza tra *esecuzione in modo immediato* ed *esecuzione in modo differito*.

Alcune istruzioni possono essere eseguite in modo immediato. Nella modalità di esecuzione immediata, un' istruzione deve essere battuta senza essere preceduta da un numero di riga. La pressione del tasto ENTER fa eseguire immediatamente l' istruzione per poi ritornare il controllo al sistema, stato comandi.

Esempio: PRINT "Questa è un' istruzione eseguita in modo immediato!"

Le istruzioni adoperate nella modalità di esecuzione differita devono essere precedute da un numero detto numero di riga. Quando viene premuto il tasto ENTER, il sistema memorizza la riga numerata nella memoria di lavoro. Le istruzioni contenute in una linea di programma sono eseguite soltanto quando viene eseguita la relativa riga di programma (RUN).

Esempio: 10 REM Questa è una linea usata in modo differito.

2.1 Notazione adottata e definizioni sintattiche

Nei capitoli successivi sono descritte le istruzioni dell' interprete basic dell' M10. Al fine di facilitare la lettura e la comprensione del testo, è stata adottata una notazione uniforme in tutti e sette i capitoli seguenti. Vediamo allora alcuni simboli ed alcune abbreviazioni adoperate:

Ogni comando viene esaminato descrivendone il nome, la natura (comando, istruzione o funzione) e le modalità in cui esso può essere eseguito: imm. indica il modo immediato, diff. quello differito. La presenza dì tutte e due le abbreviazioni significa che il comando può essere eseguito in entrambe le modalità. Così ad esempio:

PRINT istruzione imm. & diff.

definisce l' istruzione di nome PRINT che può essere eseguita in modo immediato e differito.

Dopo la specifica dell' istruzione segue una sua breve descrizione, e quindi la definizione della sua sintassi, nella quale sotto la voce *dove:* vengono illustrati i termini e le abbreviazioni adottati. Concludono infine la descrizione del comando una serie di informazioni e suggerimenti utili, nonché uno o più esempi esplicativi.

Talvolta nel testo si è adoperato il *corsivo* per evidenziare termini, istruzioni o commenti importanti. Le parentesi quadre sono state utilizzate per definire dei tasti. Così ad esempio:

[ENTER] indica il tasto ENTER (a capo).

Nel corso dei successivi capitoli sono frequentemente adoperati alcuni termini fondamentali. Sebbene essi siano descritti prima del loro uso, diamo qui di seguito l' elenco completo delle definizioni sintattiche di base.

Una costante numerica è un numero.

Esempio: 15 è una costante numerica.

Una costante stringa è una sequenza di caratteri racchiusi tra doppi apici.

Esempio: "Questa è una costante stringa! "

Il nome di una variabile è una sequenza di caratteri alfanumerici di lunghezza arbitraria. La sequenza deve cominciare con una lettera (maiuscola o minuscola) e proseguire con lettere o numeri. Ogni altro carattere che non sia una lettera o un numero non può pertanto apparire nel nome di una variabile.

Esempio: Massimo, K2, AZ21bis sono nomi legali, mentre 5C, A), Barbara.Paolo sono illegali.

L' uso di nomi illegali provoca errori di sintassi (?SN Error). L' interprete basic riconosce soltanto i primi due caratteri di un nome: pertanto Gianni e Giovanni sono per il sistema la stessa persona.

Una variabile numerica intera è un nome concluso dal simbolo %.

Esempio: Tizio%, C2% sono due variabili intere.

Una variabile numerica reale (singola precisione) è un nome, che facoltativamente è concluso dal simbolo !

Esempio: Max, C5! sono due variabili reali in singola precisione.

Una variabile numerica reale in doppia precisione è un nome concluso dal carattere #.

Esempio: ALFA#, H13# sono due variabili reali in doppia precisione.

Una variabile stringa è un nome concluso dal simbolo $.

Esempio: Nome$, A$ sono due variabili stringa.

Un' espressione numerica è:

una costante numerica, oppure
una variabile numerica (intera o reale), oppure
una funzione numerica (intera o reale), oppure
due o più dei suddetti termini separati da +, -, *, /, ç (divisione tra numeri interi), o ^ (elevamento a potenza).

Esempio: 3, A%, ABS (-7) e 6 + AZ / SIN (X) sono tre espressioni numeriche.

Un' espressione stringa è:

una costante stringa, oppure
una variabile stringa, oppure
una funzione stringa, oppure
due o più dei suddetti termini concatenati con +.

Esempio: "Espressione", G$, LEFT$(B$,5) e "Ciao" + CHR$(65) + S$ sono espressioni stringa.

Un operatore relazionale è:

=, >, >, >=, <=, =>, =<, <>, ><

Un operatore logico è:

NOT, AND, OR, XOR, IMP o EQV.

Un' espressione relazionale è:

Due o più espressioni (numeriche o stringa) separate da operatori relazionali.

Esempio: A > 12 e N$ = "Mario" sono due espressioni relazionali.

Un' espressione logica è:

Un' espressione relazionale
Due o più espressioni relazionali separate da operatori logici.

Esempio: A > 12 e K < = 7 AND H$ > "CCC" sono due espressioni logiche.

2.2 Dati numerici e aritmetica

Il basic dell' M10 gestisce due tipi fondamentali di dati: quelli stringa e quelli numerici. La ripartizione tra questi due classi di dati è mantenuta anche nella memoria di lavoro; infatti lo spazio destinato alle stringhe è separato da quello riservato alle variabili numeriche. I comandi per la gestione della memoria sono trattati nel capitolo successivo.

Dati stringa

Le stringhe, sequenze di massimo 255 caratteri ASCII, sono normalmente adoperate per memorizzare e gestire dati non numerici, come ad esempio nomi, codici, parole, e così via. Lo spazio riservato alle stringhe è per default di 256 byte (1 byte = 1 carattere), ma può essere aumentato mediante l' istruzione CLEAR (vedi cap. 3). Poiché i caratteri sono codificati secondo lo standard ASCII, è possibile eseguire confronti tra stringhe; così ad esempio "albero" è minore di "casa".

Dati numerici

Il basic dell' M10 gestisce ben tre tipi di dati numerici: gli interi, i reali in singola precisione, e quelli in doppia. Tutti i dati numerici sono rappresentati in formato decimale (base 10).

I numeri interi consentono la migliore efficienza, sia come occupazione di memoria che come velocità di calcolo. Tuttavia il loro campo di definizione è piuttosto limitato, essendo formato dagli interi compresi tra -32768 e 32767.

I numeri reali in singola precisione sono rappresentati dall' interprete basic con sette cifre significative, che vengono arrotondate per eccesso (arrotondamento matematico). Il campo di definizione di questi numeri è compreso tra $\pm 10^{-64}$ e $\pm 10^{62}$. Pur essendo adoperate nei calcoli sette cifre significative, ne vengono visualizzate o stampate soltanto sei. Rispetto ai numeri interi, i reali in singola precisione richiedono

molta più memoria per la memorizzazione e più tempo per l' elaborazione, ma consentono di gestire una varietà maggiore di numeri.

I numeri reali in doppia precisione sono rappresentati dall' interprete basic con sedici cifre significative, che vengono arrotondate per eccesso (arrotondamento matematico). Il campo di definizione di questi numeri è compreso tra $\pm 10^{-64}$ e $\pm 10^{62}$. Pur essendo adoperate nei calcoli sedici cifre significative, ne vengono visualizzate o stampate soltanto quindici. Rispetto ai numeri interi e ai reali, i reali in doppia precisione richiedono molta più memoria per la memorizzazione e più tempo per l' elaborazione, ma consentono di gestire numeri con precisione molto accurata.

L' interprete basic, se non diversamente specificato, considera le variabili numeriche come dati in doppia precisione. È possibile però definire il tipo della variabile aggiungendo un carattere particolare al suo nome, e precisamente:

% per gli interi

! per i reali in singola precisione

per i reali in doppia precisione

$ per le stringhe

Il carattere sopra indicato deve essere aggiunto come ultimo carattere del nome della variabile (consultare anche il paragrafo precedente).

Se occorre ridefinire molte variabili, è preferibile adoperare le seguenti istruzioni: DEFINT, DEFSNG, DEFDBL e DEFSTR. Vediamole in dettaglio:

DEFINT lid
DEFSNG lid
DEFDBL lid
DEFSTR lid

dove: lid è un elenco di lettere (iniziali) che identificano l' appartenenza di una variabile a quel determinato tipo.

DEFINT definisce tutte le variabili del programma, il cui nome comincia per una lettera contenuta nell' elenco, come variabili intere, DEFSNG come variabili reali in singola precisione, DEFDBL in doppia precisione ed infine DEFSTR come variabili stringa.

Esempio: DEFINT G-K, X, Z definisce come variabili intere tutte le variabili del programma i cui nomi iniziano per G, H, I, J, K, X, Z.

Conversioni numeriche

L' esistenza di tre classi diverse di numeri rende indispensabile esaminare le conversioni da un tipo all' altro. Queste conversioni numeriche possono essere eseguite implicitamente dal sistema quando si effettua un' operazione di assegnamento (K = R, o LET K = R), oppure esplicitamente adoperando delle apposite funzioni di conversione. In questo paragrafo esamineremo solo il primo caso, mentre il secondo sarà ampiamente trattato nel capitolo 9, paragrafo 3.

Essendo tre le classi di numeri a nostra disposizione, avremo quindi sei tipi di conversione da esaminare.

Da intero a reale in singola precisione

Questa conversione è banalmente eseguibile in quanto gli interi sono un sottoinsieme dei reali. Il numero sarà convertito in reale senza perciò alcun problema di troncamento o di arrotondamento. Il risultato sarà un numero reale senza cifre decimali.

Da intero a reale in doppia precisione

Anche questa conversione è banalmente eseguibile in quanto gli interi sono un sottoinsieme dei reali. Il numero sarà convertito in reale senza perciò alcun problema di troncamento o di arrotondamento. Il risultato sarà un numero reale senza cifre decimali.

Da reale (singola precisione) ad intero

Questa conversione è eseguita per troncamento; tutte le cifre decimali vengono quindi eliminate. Poiché il campo di definizione degli interi è più ristretto di quello dei reali, occorre prestare attenzione affinché non si verifichino errori di supero di capacità (overflow); pertanto la parte intera del reale da convertire deve essere compresa tra -32768 e 32767.

Da reale (singola precisione) a reale in doppia precisione

Questa conversione è eseguibile senza alcun problema in quanto i reali in singola precisione sono un sottoinsieme di quelli in doppia (stesso campo di definizione, ma diverso numero di cifre significative, 7 per la singola precisione, 16 per la doppia). Il numero sarà convertito in reale in doppia precisione senza perciò alcun problema di troncamento o di arrotondamento. Il risultato sarà un numero reale in doppia precisione con le prime 7 cifre decimali non nulle.

Da reale (doppia precisione) ad intero

Questa conversione è eseguita per troncamento; tutte le cifre decimali vengono quindi eliminate. Poiché il campo di definizione degli interi è più ristretto di quello dei reali, occorre prestare attenzione affinché non si verifichino errori di supero di capacità (overflow); pertanto la parte intera del reale da convertire deve essere compresa tra -32768 e 32767.

L' aritmetica

Abbiamo già descritto nel paragrafo precedente i tre diversi tipi di operatori posseduti dal basic dell' M10: matematici, relazionali e logici.
In questa paragrafo esamineremo come questi operatori influiscono sul calcolo di un' espressione.

Operatori matematici

Questi operatori sono utilizzabili sono su numeri, eccezion fatta per il simbolo + che, adoperato con le stringhe, rappresenta l' operazione di concatenamento (vedere anche cap. 5). Qui di seguito diamo un breve elenco di questi operatori:

+ rappresenta l' addizione
- rappresenta la sottrazione o il cambio di segno
* rappresenta la moltiplicazione
/ rappresenta la divisione
ç rappresenta la divisione tra interi
^ rappresenta l' elevamento a potenza

Prima di una divisione tra interi gli operandi sono arrotondati al numero intero più vicino, arrotondamento che viene eseguito anche sul quoziente.

Il calcolo delle espressioni viene eseguito dal basic seguendo le tradizionali regole dell' algebra. Pertanto la priorità tra gli operatori matematici è così definita:

1. Elevamento a potenza
2. Cambio di segno
3. Moltiplicazione e divisione
4. Divisione tra interi
5. Addizione e sottrazione

A parità di livello di priorità, il calcolo dell' espressione viene eseguito procedendo da sinistra verso destra. È naturalmente possibile modificare le priorità mediante parentesi, in base alle comuni regole dell' algebra. È altresí possibile impiegare più livelli di pa-

rentesi, oppure impiegarle anche là dove non strettamente necessario per rendere più chiara l' espressione.

Il calcolo dell' espressione può dar luogo ad errori di supero di capacità (?OV Error), quando il risultato ottenuto non rientra nel campo di definizione di una variabile, a errori di divisioni per 0 (?/0 Error), quando si cerca di dividere un numero per zero, o infine ad un errore di mancanza di operandi (?MO Error), se è stato dimenticato qualche operando o qualche parentesi.

Operatori relazionali

Questi operatori, utilizzabili sia per i dati numerici che per le stringhe, servono per confrontare due espressioni dello stesso tipo. Il confronto tra le stringhe viene così eseguito: se le due stringhe sono di lunghezza differente, la più lunga viene troncata per poterla confrontare con l' altra, e quindi vengono confrontate (mediante codice ASCII) carattere per carattere, procedendo da sinistra (carattere più significativo) verso destra. L' operazione di confronto restituisce un valore numerico che è pari a -1 se il confronto è vero, 0 se è falso. Per completezza riportiamo l' elenco degli operatori relazionali:

=	Uguale a
>	Maggiore di
<	Minore di
> = o = >	Maggiore o uguale a
< = o = <	Minore o uguale di
< > o > <	Diverso da

Operatori logici

Questi operatori consentono di eseguire operazioni logiche su espressioni relazionali. Per completezza ne riportiamo l' elenco (in ordine di priorità):

NOT	indica la negazione
AND	indica la congiunzione
OR	indica la disgiunzione
XOR	indica l' OR esclusivo
IMP	indica l' implicazione
EQV	indica l' equivalenza

Qui di seguito è riportata la tabella di verità per questi operatori.

Ø = Falso 1 = Vero	A Ø	B Ø	A 0	B 1	A 1	B 0	A 1	B 1
NOT A	1		1		0		0	
A AND B		0		0		0		1
A OR B		0		1		1		1
A XOR B		0		1		1		0
A EQV B		1		0		0		1
A IMP B		1		1		0		1

Figura 2.1

È possibile adoperare gli operatori di confronto con numeri interi, eseguendo così operazioni logiche bit a bit. Il risultato dell' operazione è fornito in formato decimale. Occorre tenere presente che i numeri interi sono rappresentati dall' interprete basic in "complemento a due".

Esempio: PRINT 2 OR 8 fornisce 10 come risultato, mentre

PRINT 8 AND 10 dà 2 come risultato.

Capitolo 3

Comandi del sistema e di utilità

Una delle prime necessità che si avverte quando si lavora con un computer, sia in fase di programmazione che di utilizzo di programmi "pronti", è quella di conoscere alcuni comandi e istruzioni base per eseguire delle operazioni fondamentali: tipicamente azzerare la memoria, arrestare un programma, e così via. In altre parole, utilizzando una terminologia più formale, quello che ci occorre è conoscere comandi, istruzioni e funzioni relativi alla gestione del sistema, ove con questo sostantivo intendiamo il complesso di tutte le parti costituenti un computer: la memoria, il video, la tastiera, ecc... Come avrete quindi certamente intuito, è proprio questo il contenuto di questo terzo capitolo, suddiviso in otto paragrafi, ciascuno dei quali dedicato ad un diverso aspetto del sistema.

3.1 L' orologio-calendario interno.

L' Olivetti M10 possiede al suo interno un orologio-calendario alimentato da una batteria tampone che ne assicura il funzionamento anche a macchina spenta. La sua consultazione e la sua regolazione possono essere effettuate tramite tre istruzioni / funzioni BASIC: DATE$, DAY$ e TIME$. La prima consente di visualizzare e modificare la data, la seconda il giorno della settimana, l' ultima l' ora. Tutte e tre queste istruzioni sono molto utili non solo per consultazione, ma soprattutto in tutte quelle situazioni dove occorre una manipolazione e un controllo del tempo. Prima di mostrare alcune tipiche applicazioni, descriveremo brevemente la sintassi di queste istruzioni.

DATE$ istruzione e funzione imm. & diff.

DATE$ = "gg / mm / aa" (istruz.)

DATE$ (funz.)

dove: gg è il giorno, numero di due cifre compreso tra 01 e 31

mm è il mese, numero di due cifre compreso tra 01 e 12

aa è l' anno, numero di due cifre compreso tra 01 e 99

Esempio: *DATE$ = "06 / 11 / 61"* pone la data al 6 novembre 1961

PRINT DATE$ fornisce la data attuale.

Nel modello M10 versione USA (tastiera QWERTY, vedi cap. 1), il formato della data è leggermente differente:

DATE$ = "mm / gg / aa"

dove mm, gg e aa hanno lo stesso significato e le stesse limitazioni viste prima.

DAY$ istruzione e funzione imm. & diff.

DAY$ = "xxx" (istruz.)
DAY$ (funz.)

dove xxx può essere una tra le seguenti stringhe:

Mon	per lunedì
Tue	per martedì
Wed	per mercoledì
Thu	per giovedì
Fri	per venerdì
Sat	per sabato
Sun	per domenica

Esempio: *DAY$ = "Fri"* pone il giorno della settimana a venerdì

PRINT DAY$ fornisce il giorno della settimana

TIME$ istruzione e funzione imm. & diff.

TIME$ = "hh:mm:ss"	(istruz.)
TIME$	(funz.)
TIME$ ON	(istruz.)
TIME$ OFF	(istruz.)
TIME$ STOP	(istruz.)

dove: hh è l' ora, numero di due cifre compreso tra 00 e 23

mm sono i minuti, numero di due cifre compreso tra 00 e 59

ss sono i secondi, numero di due cifre compreso tra 00 e 59

Esempio: *TIME$ = "10:35:00"* pone l' orologio alle ore 10, 35 minuti e 0 secondi

PRINT TIME$ fornisce l' ora attuale.

L' istruzione TIME$ ON / OFF / STOP permette di rispettivamente abilitare, disabilitare ed inibire l' istruzione ON TIME$....GOSUB (vedi cap. 7). Praticamente TIME$ ON / OFF / STOP funge da "interruttore" per l' istruzione ON TIME$....GOSUB: TIME$ ON consente l' aggancio alla subroutine specificata, TIME$ OFF no, TIME$ STOP congela l' aggancio finché non viene eseguita TIME$ ON. In tal caso se la condizione di salto è verificata (l' ora indicata è già trascorsa), viene immediatamente eseguito un salto alla subroutine.

Il tentativo di impostare valori diversi da quanto specificato per le istruzioni DATE$, DAY$ e TIME$, comporterà la comparsa di un messaggio di errore di sintassi (?SN Error).

Vediamo ora degli esempi sull' impiego di queste istruzioni. Il programma che segue realizza una sveglia elettronica.

```
 10 REM Sveglia
 20 CLS
 30 INPUT "Orario della sveglia (hh:mm:ss)";T$
 40 TIME$ ON
 50 ON TIME$ = T$ GOSUB 100
 60 PRINT §176,TIME$
 70 GOTO 60
100 REM Subroutine sveglia
110 BEEP:BEEP:BEEP
120 END
```

Questa semplice applicazione mostra come può essere utilizzata TIME$: alla linea 40 viene impiegata come istruzione, alla 60 come funzione. È importante notare che l' istruzione ON TIME$ = T$ GOSUB 100 contenuta alla linea 50, pur essendo eseguita all' inizio del ciclo di attesa (linee 60-70), grazie all' istruzione TIME$ ON della linea 40, viene verificata continuamente.

Naturalmente esistono molte altre situazioni in cui è necessario avere un controllo sul tempo: giochi, quiz, controlli di processi e così via. Per estrapolare dati parziali dalle funzioni in questione, occorrerà utilizzare funzioni per la manipolazione delle stringhe (vedi cap. 5).

Ad esempio: *ANNO$ = RIGHT$(DATE$,2)* assegna alla stringa ANNO$ l' anno corrente.

3.2 La memoria

L' organizzazione della memoria dell' Olivetti M10 è molto diversa da quella dei personal computer, poiché nell' M10 la memoria RAM funge sia da memoria centrale che da memoria di massa. Pertanto il BASIC Microsoft della macchina possiede sia comandi per la gestione della memoria di lavoro, che comandi e istruzioni per la gestione dei dati (file). In questo paragrafo ci occuperemo dei primi, lasciando i secondi ad un paragrafo successivo.

L' interprete BASIC dell' M10 si riserva una porzione di memoria RAM come memoria di lavoro (memoria centrale). In essa vengono memorizzati i dati e le istruzioni dei programmi BASIC da eseguire o da redigere. Questa memoria, come del resto tutta la RAM di cui è dotato l' M10, è non volatile, ovverossia mantiene i programmi e le informazioni contenuti anche a macchina spenta. Naturalmente, onde poter gestire la memoria di lavoro, l' interprete BASIC possiede un insieme di comandi per svolgere alcune operazioni fondamentali: cancellare un programma, azzerare i dati, visualizzare la quantità di memoria disponibile, e così via. Vediamo allora in dettaglio questi comandi:

NEW comando imm. & diff.

NEW cancella il programma contenuto nella memoria di lavoro, ed inizializza tutte le variabili: quelle numeriche vengono poste a zero, quelle stringa a stringa nulla. La sua sintassi è:

NEW

Il comando NEW cancella il programma contenuto nella memoria di lavoro, ma non il relativo (se esistente) file di tipo .BA contenuto nella directory della memoria (per chiarimenti consultare il cap. 1).
È comunque un comando da usarsi con attenzione, specie quando il programma attualmente in memoria di lavoro non è stato salvato in RAM (comando *Save "RAM:..."*), poiché è irreversibile: non esiste alcun modo per "resuscitare" il programma cancellato.

CLEAR comando imm. & diff.

La memoria di lavoro viene ulteriormente suddivisa dall' interprete BASIC in spazio dati variabili stringa e in spazio dati variabili numeriche e istruzioni programma. Il comando CLEAR consente di inizializzare tutte le variabili e di definire lo spazio dati per le stringhe. Ancora CLEAR permette di definire la più alta locazione di memoria accessibile al BASIC.
Ma vediamone la sintassi:

```
CLEAR
CLEAR SS
CLEAR SS,HM
```

dove: SS è il numero di byte disponibili per le variabili stringa

HM è la più alta locazione di memoria disponibile al BASIC.

CLEAR inizializza tutte le variabili: pone quelle numeriche a zero, quelle stringa uguali alla stringa nulla, nonché chiude tutti i file rimasti aperti. CLEAR definisce anche il numero di byte riservati alle variabili stringa (SS): se omesso viene assunto come valore di default 256. Analogamente se non specificata esplicitamente, per la locazione più alta viene assunto come default il valore della funzione HIMEM (vedi più avanti).

Nel caso realizziate programmi che utilizzano molte variabili stringa, sarà allora probabilmente necessario aumentare lo spazio dati loro riservato (quello standard è 256 byte), eseguendo un CLEAR. Per avere invece tutta la memoria disponibile, specificate come secondo argomento del CLEAR la funzione MAXRAM (vedi più avanti). Diminuire la quantità di memoria accessibile al BASIC, può essere utile nel caso di impiego di programmi in linguaggio macchina.
Vediamo infine degli esempi per meglio puntualizzare quanto detto.

Esempio: *CLEAR* inizializza tutte le variabili, chiude tutti i file, assegna alle variabili stringa 256 byte.

CLEAR 500 ha lo stesso effetto di sopra eccetto che lo spazio per le stringhe è di 500 byte.

CLEAR 256,MAXRAM inizializza tutte le variabili, assegna alle stringhe 256 byte, rende accessibile al BASIC tutta la memoria.

FRE funzione imm. & diff.

Questa funzione calcola il numero di byte di memoria liberi per l' utente. FRE compare automaticamente nel menù principale, e quando si entra in ambiente BASIC. La sua sintassi è:

FRE(X)
FRE(X$)

dove: X è un argomento numerico fittizio

X$ è un argomento stringa fittizio

FRE usata con argomento numerico restituisce il numero di byte disponibili per programmi BASIC, file di testo, programmi in linguaggio macchina, variabili numeriche. Se impiegata invece con argomento di tipo stringa, restituisce il numero di byte disponibili per le stringhe.

Esempio: *PRINT FRE(0)* restituisce il numero di byte di memoria liberi per programmi, variabili numeriche, ecc...

PRINT FRE("") restituisce il numero di byte liberi per le stringhe.

HIMEM funzione imm. & diff.

Questa funzione fornisce il più alto indirizzo di memoria accessibile al BASIC. Il suo valore è determinato dall' ultima istruzione CLEAR eseguita in cui sia stato specificato l' indirizzo in questione. L' esecuzione di un CLEAR senza che venga specificata la locazione più alta di memoria, non modifica il valore di HIMEM.
Sintassi:

HIMEM

Qualora non siano state eseguite CLEAR volte a modificarne il valore, HIMEM è uguale a MAXRAM (62960).

MAXRAM funzione imm. & diff.

Questa funzione ritorna, come il suo stesso nome fa intuire, il più alto indirizzo di memoria disponibile al BASIC. La sua sintassi è:

MAXRAM

Solitamente MAXRAM viene utilizzata nell' istruzione CLEAR come secondo argomento per rendere disponibile al BASIC la maggior quantità possibile di memoria.

3.3 Gestione dei file

La memoria RAM dell' Olivetti M10 può contenere fino a 19 file contemporaneamente. Il suo contenuto è mostrato da un indice (directory), che specifica i nomi ed il tipo dei file presenti. In questo paragrafo tratteremo di comandi relativi alla gestione dei file: come cancellarli, come ridenominarli, ecc... Non esamineremo invece quelle istruzioni e quelle funzioni relative all' ingresso / uscita dei dati tramite file o periferiche. Ad esse è interamente dedicato il capitolo 6.
Vediamo ora questi comandi:

KILL comando imm. & diff.

Questo comando serve per cancellare un file dalla memoria RAM. La sua sintassi è:

KILL "N.T"

dove: N è il nome del file (max 6 caratteri)

T specifica il tipo di file, e precisamente:

BA per un file programma BASIC
CO per un file programma in linguaggio macchina
DO per un file testo.

Ad esempio: *KILL "note.do"* cancella il file testo denominato NOTE.

Per cancellare un programma contenuto nella memoria RAM e nella memoria di lavoro, è necessario far precedere il comando KILL da NEW: viceversa comparirà un messaggio di chiamata illegale di funzione (?FC Error).

Come per il comando NEW, KILL è da usarsi con cautela, specialmente quando il file interessato non è stato salvato su nastro.

NAME comando imm. & diff.

Questo comando consente di cambiare nome ad un file esistente. La sua sintassi è:

NAME "N1.T" AS "N2.T"

dove: N1 è il vecchio nome del file

N2 è il nuovo nome del file

T è il suffisso del file (tipo)

Esempio: *NAME "note.do" AS "appunt.do"* cambia il nome del file testo NOTE in APPUNT.

Il suffisso del nuovo nome del file deve essere identico al vecchio: per esempio non è possibile tramutare un file di tipo .BA in uno di tipo .DO.

Se uno dei due suffissi non viene specificato, compare un errore di file non trovato (?FF Error), mentre se non esiste alcun file con il vecchio nome, oppure se esiste già un file con lo stesso nome di quello nuovo, viene visualizzato un errore di chiamata illegale di funzione (?FC Error).

IPL comando imm. & diff.

Con IPL è possibile rendere un programma "autopartente", ovverossia far si che esso venga eseguito immediatamente dopo l' accensione del sistema. La sua sintassi è:

IPL "Nf"
IPL

dove: Nf è il nome completo di un file

È possibile specificarè come file non solo programmi BASIC, ma anche file di testo, programmi in linguaggio macchina e persino gli applicativi di base: BASIC, TEXT, TELCOM, SCHEDL e ADDRS.

Ad esempio: *IPL "BASIC"* fa si che all' accensione l' M10 entri in ambiente BASIC subito dopo la visualizzazione del menù

IPL eseguito senza argomenti cancella il precedente comando.

MENU comando imm. & diff.

Il comando MENU provoca l' uscita dall' ambiente BASIC e la relativa visualizzazione del menù. La sua sintassi è:

MENU

In ambiente BASIC il tasto funzione F8, a meno di successive ridefinizioni, svolge la stessa funzione.

3.4 Caricamento e salvataggio di file

La particolare organizzazione della memoria dell' M10 si riflette anche nella struttura e nell' uso dei comandi per il trasferimento programmi da e verso il computer. Ogni operazione di caricamento o di salvataggio programmi verso una periferica avviene tramite la memoria di lavoro. La memoria RAM viene vista dal sistema come una periferica, al pari del registratore a cassette e della porta seriale. Così ad esempio per salvare su nastro un programma è necessario che esso sia contenuto nella memoria di lavoro: solo a questo punto è possibile utilizzare il comando SAVE per registrarlo su cassetta. Analogamente per l' operazione inversa: effettuando un caricamento del programma dal registratore, questo verrà posto nella memoria di lavoro, senza però essere inserito nella directory. Per far ciò è necessario eseguire un SAVE ala RAM. In altri termini il sistema usa la memoria di lavoro come una memoria temporanea, di transito (buffer), attraverso cui trasferire i programmi destinati al registratore a cassette, alla porta seriale, o infine alla memoria RAM. Non esiste dunque alcun collegamento diretto tra queste tre periferiche: qualsiasi transizione deve passare dalla memoria di lavoro (figura 3.1).

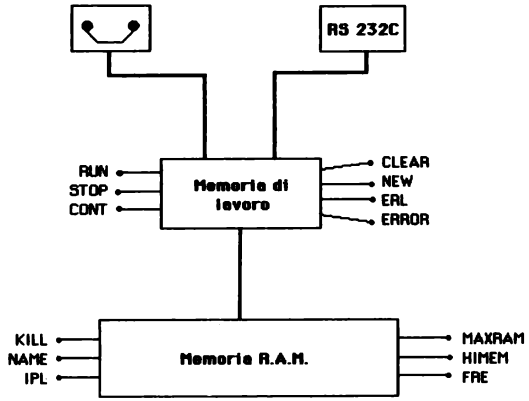


Figura 3.1

Ma vediamo allora queste istruzioni:

CLOAD, CLOAD M comandi imm. & diff.

Questi comandi servono per caricare programmi dal registratore nella memoria di lavoro. Il programma non viene inserito nella directory della RAM. Per questo occorre eseguire un SAVE alla RAM (vedi più avanti). La loro sintassi è la seguente:

```
CLOAD
CLOAD "Nf"
CLOAD "Nf",R
CLOADM
CLOADM "Nf"
```

dove: Nf è il nome della registrazione.

CLOAD carica da cassetta programmi BASIC, mentre CLOADM carica programmi in linguaggio macchina e li memorizza alla locazione specificata al momento della registrazione. La R posta dopo il nome del file, fa eseguire il programma subito dopo il caricamento.

Esempio: *CLOAD "prova",R* carica dal registratore nella memoria di lavoro il programma PROVA e lo esegue.

Digitato il comando, occorre avviare il registratore. Se il programma specificato è preceduto da altri, apparirà sul video la scritta *SKIP nome-file*, ad indicare il programma trovato. Quando viene raggiunto il programma desiderato, appare invece la scritta *FOUND nome-file*. Se nel comando non è stato specificato alcun nome, verrà caricato il primo programma trovato.

Durante il caricamento del programma sono udibili i segnali della trasmissione dei dati; per evitare ciò può essere utilizzato il comando SOUND OFF (vedi cap. 8).

CLOAD? comandi imm. & diff.

CLOAD? consente di confrontare un programma BASIC registrato su cassetta con uno contenuto in memoria RAM. CLOAD? verifica la correttezza della registrazione del programma salvato in formato .BA. La sintassi di questo comando è:

```
CLOAD? "Nf"
```

dove: Nf è il nome della registrazione (file)

Esempio: *CLOAD? "PROVA"* verifica il programma PROVA contenuto in memoria con quello salvato sul nastro.

Il programma registrato su nastro viene confrontato byte per byte con quello presente nella RAM. Nel caso venga riscontrata qualche differenza, viene visualizzato il messaggio VERIFY FAILED ad indicare che la registrazione effettuata non è corretta; occorrerà in questo caso ripetere la registrazione.

CLOAD? è un comando molto utile, da usarsi puntualmente dopo ogni SAVE per controllare la correttezza della registrazione: così facendo si eviterà la perdita di quei programmi che non sono stati salvati correttamente.
"Meglio un CLOAD? in più che un programma in meno", è il motto che dovete tenere presente quando salvate i vostri programmi.

LOAD, LOADM comandi imm. & diff.

Questi comandi consentono di caricare programmi nella memoria di lavoro. Vediamone la sintassi:

```
LOAD "Nf"
LOAD "Nf",R
LOAD "CAS:Nf"
LOAD "CAS:Nf",R
LOAD "RAM:Nf"
LOAD "RAM:Nf",R
LOAD "COM:vfpsx"
LOAD "COM:vfpsx",R
LOADM "Nf"
LOADM "Nf",R
LOADM "CAS:Nf"
LOADM"CAS:Nf",R
LOADM "RAM:Nf"
LOADM "RAM:Nf",R
```

dove: Nf è il nome della registrazione.

CAS indica che il programma viene caricato dalla cassetta.

RAM indica che il programma viene caricato da RAM.

COM indica che il programma viene caricato da linea seriale.

R fa eseguire il programma subito dopo il caricamento.

vfpsx sono i parametri di trasmissione.

LOAD serve per caricare programmi BASIC nella memoria di lavoro, LOADM per programmi in linguaggio macchina. Entrambi i comandi caricano i programmi nella memoria di lavoro, ma non aggiornano la directory della RAM. Per memorizzarli nella RAM, occorre eseguire l' appropriato comando SAVE (vedi più avanti).

LOAD cancella il precedente contenuto della memoria di lavoro, e chiude tutti i file. LOADM cancella quei dati memorizzati nelle locazioni di memoria occupate dal programma in linguaggio macchina, il cui indirizzo di partenza e la cui lunghezza sono specificati al momento della registrazione.

L' opzione R (posta dopo il nome del file), manda in esecuzione il programma a caricamento avvenuto, lasciando aperti i file dati.

LOAD, LOADM, LOAD "RAM: e LOADM "RAM: servono a trasferire programmi dalla RAM alla memoria di lavoro.
Se per esempio è necessario listare e correggere un programma BASIC presente in RAM, occorrerà per prima cosa portarlo nella memoria di lavoro: per questo si effettuerà un comando *LOAD "RAM:nome-programma"*.

LOAD "CAS: e LOADM "CAS: hanno la stessa funzione di CLOAD E CLOADM; entrambi caricano in memoria di lavoro un programma registrato su cassetta. Spiegazioni e modalità d' uso di questi comandi sono gli stessi di CLOAD e CLOADM; rimando pertanto il lettore a quanto scritto per essi.

LOAD "COM:vfpsx e LOADM "COM:vfpsx caricano programmi da linea di comunicazione. È indispensabile specificare i parametri di trasmissione. Per un corretto uso della linea di comunicazione, consultare l' appendice A.

CSAVE, CSAVEM comandi imm. & diff.

CSAVE e CSAVEM servono per memorizzare su nastro programmi contenuti nella memoria di lavoro. La loro sintassi è la seguente:

CSAVE "Nf"
CSAVEM "Nf",a,b,c

dove: Nf è il nome dato alla registrazione

a,b,c sono rispettivamente: l' indirizzo di partenza, di fine e la lunghezza del programma in linguaggio macchina

Esempio: *CSAVE "Prova"* memorizza su nastro il programma contenuto nella memoria di lavoro, assegnando alla registrazione il nome Prova.

CSAVE consente di memorizzare su nastro programmi BASIC, CSAVEM programmi in linguaggio macchina. Nel caso che il programma che si desidera salvare non sia nella memoria di lavoro, eseguire preventivamente un LOAD dello stesso.

Prima di digitare uno dei due comandi, avviare il registratore in registrazione. Al termine del salvataggio, verificare la correttezza della registrazione con il comando CLOAD?.

SAVE, SAVEM comandi imm. & diff.

SAVE e SAVEM permettono di memorizzare su diverse periferiche programmi contenuti nella memoria di lavoro. Se il programma che si desidera salvare non vi è presente, eseguire un comando *LOAD "nome-programma"*. Sintassi:

```
SAVE "Nf"
SAVE "Nf",A
SAVE "RAM:Nf"
SAVE "RAM:Nf",A
SAVE "CAS:Nf"
SAVE "CAS:Nf",A
SAVE "COM:vfpsx"
SAVEM "Nf",a,b,c
SAVEM "RAM:Nf",a,b,c
SAVEM "CAS:Nf",a,b,c
```

dove: Nf è il nome dato alla registrazione

RAM salva il programma in RAM

CAS salva il programma su cassetta

COM salva il programma su linea di comunicazione

A salva il programma in formato ASCII (suffisso del file .DO)

vfpsx sono parametri di trasmissione

a,b,c sono rispettivamente l' indirizzo di partenza, di termine e la lunghezza del programma

Esempio: *SAVE "CAS:Prova"* memorizza su nastro il programma attuale e denomina la registrazione Prova.

SAVE memorizza programmi BASIC in formato binario (tipo .BA) o in formato ASCII (.DO) ponendo dopo il nome una lettera A. SAVEM memorizza programmi in linguaggio macchina; è necessario specificare l' indirizzo di partenza, quello finale e la lunghezza in byte del programma.

SAVE, SAVEM, SAVE "RAM: e SAVEM "RAM: memorizzano il programma attuale nella RAM, aggiornandone la directory. Dopo il caricamento di un programma BASIC, oppure dopo la sua stesura è necessario trasferirlo in RAM con SAVE.

SAVE "CAS: e SAVEM "CAS: memorizzano il programma attuale su cassetta. Senza l' opzione A svolgono la stessa funzione di CSAVE e CSAVEM. Leggere anche le avvertenze fornite per questi comandi.

SAVE "COM:vfpsx viene usato per trasferire un programma ad una periferica tramite linea di comunicazione. I parametri di trasmissione devono essere identici: consultare per essi l' appendice A.

Esistono infine i comandi SAVE "LCD: e SAVE "CRT: per listare il programma attuale sul display o su video. Hanno la stessa funzione del comando LIST (vedi cap. 4).

MERGE comando imm. & diff.

MERGE permette di fondere assieme il programma contenuto nella memoria di lavoro con uno proveniente da nastro, RAM o linea di comunicazione. La sua sintassi è:

MERGE "CAS:Nf"
MERGE "RAM:Nf"
MERGE "COM:vfpsx"

dove: Nf è il nome del file

CAS indica che il programma integrante proviene da cassetta

RAM indica che il programma integrante risiede in RAM

COM indica che il programma integrante proviene dalla linea di comunicazione

vfpsx sono parametri di trasmissione

Esempio: *MERGE "RAM:Prova2"* fonde il programma in memoria di lavoro con Prova2 residente in RAM.

Le linee del programma integrante sono fuse (aggiunte) a quelle del programma in memoria. Nel caso vi siano numeri di linea duplicati, la linea del programma attuale viene sostituita da quella del nuovo programma. Nell' operazione di fusione viene mantenuto l' ordinamento dei numeri di linea.

Il programma integrante deve essere in formato ASCII, ovverossia deve essere del tipo .DO. Ad operazione terminata, il controllo torna al sistema; nella memoria di lavoro sarà contenuto il programma prodotto dalla fusione.

Nel caso venga utilizzato un file non in formato .BA, verrà visualizzato un messaggio di nome file errato (?NM Error). Se il programma prodotto da MERGE risulta più grande della memoria a disposizione, comparirà un errore di memoria insufficiente (?OM Error).

MOTOR ON / OFF comando imm. & diff.

MOTOR ON e MOTOR OFF consentono di controllare da BASIC il motore di un registratore a cassetta. La loro sintassi è:

MOTOR ON
MOTOR OFF

MOTOR ON abilita il funzionamento del registratore a cassetta, MOTOR OFF lo disabilita. MOTOR OFF consente inoltre di controllare automaticamente il motore del registratore durante il trasferimento dei dati.

3.5 Controllo programmi.

Questo paragrafo contiene comandi e istruzioni per il controllo dei programmi. Lanciare in esecuzione un programma, fermarlo, continuarne l' esecuzione: sono alcuni dei comandi BASIC per la gestione delle risorse di elaborazione dell' M10.

RUN e RUNM comandi imm. & diff.

Questi due comandi consentono di caricare un programma nella memoria di lavoro e di lanciarne l' esecuzione. La loro sintassi è così definita:

```
RUN
RUN n
RUN n"Nf"
RUN n"Nf",R
RUN "Nf"
RUN "Nf",R
RUN "CAS:Nr"
RUN "CAS:Nr",R
RUN "RAM:Nr"
RUN "RAM:Nr",R
RUN "COM:vfpsx"
RUN "COM:vfpsx",R
RUNM "CAS:Nr"
RUNM "RAM:Nr"
```

dove: n è un numero di riga

Nf è il nome del file, completo di suffisso

Nr è il nome della registrazione

R mantiene i file dati aperti

CAS indica che il programma da eseguire proviene da cassetta

RAM indica che il programma da eseguire risiede in RAM

COM indica che il programma da eseguire proviene da linea di comunicazione

vfpsx sono parametri di trasmissione.

Esempio: *RUN* esegue il programma contenuto nella memoria di lavoro

RUN e RUNM caricano in memoria ed eseguono programmi BASIC e in linguaggio macchina rispettivamente. Il programma eventualmente contenuto in precedenza nella memoria di lavoro viene cancellato.

RUN e RUNM eseguono il programma contenuto nella memoria di lavoro.

RUN "RAM: e RUNM "RAM: caricano il programma specificato dalla RAM nella memoria di lavoro, e lo eseguono.

RUN "CAS: e RUNM "CAS: caricano il programma dal registratore e lo eseguono. Per un corretto caricamento del programma, consultare al paragrafo precedente i comandi CLOAD e LOAD.

RUN "COM:vfpsx carica un programma da linea di comunicazione. È indispensabile specificare correttamente i parametri di trasmissione: per un loro corretto uso, consultare l' appendice A.

L' esecuzione del programma comincia, se non diversamente specificato, dal più basso numero di linea. Il comando RUN inizializza tutte le variabili: pone quelle numeriche a zero, quelle stringa uguali alla stringa nulla. Se non specificato esplicitamente (R dopo il nome del file), RUN chiude inoltre tutti i file dati aperti.

STOP istruzione diff.

STOP provoca l' arresto del programma e ritorna il controllo al sistema. La sua sintassi è:

STOP

L' istruzione può essere utilizzata soltanto in modo differito; quando viene eseguita compare il messaggio BREAK in n, dove n è il numero di linea dell' istruzione STOP.

STOP lascia aperti i file dati. L' esecuzione del programma può essere ripresa con il comando CONT purché il programma non abbia subito modifiche.

Questa istruzione può risultare particolarmente utile nella messa a punto di programmi; inserita in opportuni punti chiave, arrestando l' esecuzione del programma, consente di verificare se il contenuto delle variabili è quello atteso.

CONT comando imm.

Questo comando serve per riprendere l' esecuzione di programma interrotto da un' istruzione STOP o dalla pressione del tasto BREAK. La sua sintassi è la seguente:

CONT

CONT può essere utilizzato solo in modo immediato; se messo all' interno di un programma provocherà un errore di impossibilità a continuare (?CN Error). Lo stesso errore si verifica quando si tenta di far riprendere l' esecuzione di un programma che è stato modificato.

CONT è molto utile per la messa a punto dei programmi: consente infatti di riprendere l' esecuzione di un programma arrestato con STOP.

END istruzione diff.

Questa istruzione fa terminare l' esecuzione di un programma e ritorna il controllo al sistema. La sua sintassi è:

```
END
```

Esempio:

```
10 REM Esempio
20 INPUT "Numero ";N
30 IF N>0 THEN PRINT "Numero positivo":END
40 IF N<0 THEN PRINT "Numero negativo":END
50 PRINT "Il numero è zero"
```

L' istruzione END può essere utilizzata soltanto in modo differito, cioè all' interno di un programma. La sua esecuzione determina l' arresto del programma in corso e la chiusura di tutti i file dati.

END viene posta nei punti terminali del programma. Il suo impiego è facoltativo nel caso dell' ultima istruzione (vedi esempio).

3.6 Gestione degli errori

Nella realizzazione di buoni programmi è indispensabile considerare la possibilità che si verifichino errori derivanti sia dal non corretto uso del programma stesso, che dal malfunzionamento di qualche periferica utilizzata. Il BASIC dell' M10 possiede un insieme di comandi dedicati al trattamento dell' errore. Con essi è possibile realizzare delle routine di gestione degli errori, evitando così l' arresto del programma e la comparsa del relativo messaggio. Ad esempio, in un programma che legge dei dati da un file, è possibile contemplare l' eventualità che il file non sia presente in RAM in un' apposita routine per il trattamento dell' errore. In questo caso il programma potrebbe visualizzare quanto accaduto, e richiedere per esempio il nome del file.
Vediamo ora in dettaglio queste istruzioni:

ERL funzione imm. & diff.

La funzione ERL restituisce il numero di riga dell' ultimo errore occorso. La sua sintassi è la seguente:

ERL

Generalmente questa funzione è utilizzata in routine per il trattamento dell' errore, o in modo immediato per individuare in quale linea di programma si è verificato un errore.

Nel caso di una linea di istruzioni eseguita in modo immediato, ERL ritornerà il numero 65535.

Per esempi su questa funzione, consultare ERR ed ERROR.

ERR funzione imm. & diff.

La funzione ERR ritorna il codice di identificazione dell' ultimo errore occorso. La sua sintassi è così definita:

ERR

ERR insieme con ERL consente di realizzare routine per il trattamento dell' errore. Precedute da una o più istruzioni ON ERROR GOTO (vedi cap. 7), queste funzioni vengono impiegate all' interno di istruzioni IF...THEN, ON...GOTO e ON...GOSUB. Ad esempio:

```
10 REM Esempio
20 ON ERROR GOTO 100
30 INPUT "Nome file ";F$
40 OPEN F$ FOR INPUT AS 1

100 REM Routine trattamento errori
110 IF ERR=52 AND ERL=40 THEN PRINT "File assente":RE-
SUME 30
120 IF ERR=55 AND ERL=40 THEN PRINT "Nome file
errato":RESUME 30
```

L' istruzione ON ERROR GOTO determina un salto alla routine della linea 100 ogniqual-

volta si verifica un errore.
Se ERR restituisce 52 (File non trovato) ed ERL 40 (la linea in cui viene eseguita la OPEN), viene visualizzato un messaggio e rieseguita la linea 30. Analogamente per il codice di errore 55 (nome file errato) viene presentato il messaggio del caso e rieseguita la linea 30.

In appendice B troverete la lista completa degli errori e dei loro codici.

ERROR istruzione imm. & diff.

L' istruzione ERROR consente di simulare il verificarsi di un errore, oppure di definire l' errore corrispondente al codice indicato. La sua sintassi è:

ERROR n

dove: n è un intero compreso tra 0 e 255.

N specifica il codice di errore che si vuole simulare o definire. Nel primo caso verrà visualizzato il messaggio corrispondente.

Esempio: *ERROR 52* provocherà la comparsa del messaggio *?FF Error*.

Se inserita all' interno di un programma, ERROR n simulerà l' errore corrispondente.

Se n invece non è un codice di errore già assegnato (vedere la tabella dell' appendice B), ERROR può essere impiegata per definire un nuovo codice di errore. Ad esempio:

```
10 REM Area triangolo
20 ON ERROR GOTO 100
30 INPUT "Base ";B
40 INPUT "Altezza ";H
50 IF B<=0 THEN ERROR 254
60 IF H<=0 THEN ERROR 255


100 REM Routine trattamento errori
110 IF ERR=254 THEN PRINT "La base deve essere
positiva":RESUME 30
120 IF ERR=255 THEN PRINT "L' altezza deve essere
positiva":RESUME 40
```

Questo programma esegue, dati base ed altezza, il calcolo dell' area del triangolo. Poi-

ché un triangolo deve avere per definizione base ed altezza positive, nel programma vengono definiti a tal scopo due errori, il 254 ed il 255, che vengono gestiti dalla routine della linea 100.

3.7 Istruzioni di utilità.

In questo paragrafo sono descritte due istruzioni di utilità del sistema. La prima, KEY, consente di definire i tasti funzione di cui è dotato il computer. La seconda, POWER, permette di spegnere automaticamente il calcolatore dopo un certo intervallo di tempo.

KEY istruzione imm. & diff.

L' istruzione KEY consente di assegnare una stringa ai tasti funzione (F1-F8), mentre KEY ON / OFF / STOP abilita, disabilita o inibisce l' aggancio ad una subroutine specificata dall' istruzione ON KEY GOSUB. La sintassi è così definita:

```
KEY n,"S"
KEY (n) ON
KEY (n) OFF
KEY (n) STOP
```

dove: n è il numero del tasto funzione

S è una stringa di massimo 15 caratteri.

KEY n,"S" consente di assegnare al tasto funzione n la stringa S. In ambiente BASIC il sistema assegna per default le seguenti stringhe:

```
F1 Files
F2 Load "
F3 Save "
F4 Run
F5 List
F6
F7
F8 Menu
```

Le stringhe assegnate possono essere visualizzate premendo il tasto LABEL, oppure eseguendo il comando KEY LIST (vedere cap. 4).

KEY 6,"Edit " assegna al tasto funzione F6 la stringa Edit.

KEY (n) ON abilita l' aggancio alla subroutine specificata nell' istruzione ON KEY GOSUB ... (vedi cap. 7). Tramite quest' ultima istruzione è possibile realizzare programmi che "sentono" la pressione di uno dei tasti funzione, e proseguono la loro esecuzione alle subroutine indicate.

KEY (n) OFF disabilita l' istruzione ON KEY GOSUB. In questo caso anche in presenza di suddetta istruzione, la pressione di un tasto funzione non ha alcuna ripercussione sull' esecuzione del programma.

KEY (n) STOP inibisce l' istruzione ON KEY GOSUB finché non viene eseguita KEY (n) ON. Se durante l' inibizione viene premuto un tasto funzione, ad abilitazione avvenuta (KEY (n) ON), verrà immediatamente eseguito un salto alla subroutine specificata.

Esempio:

```
10 REM Esempio
20 ON KEY GOSUB 100,110
30 KEY (1) ON
40 KEY (2) ON
50 INPUT "Lato 1";L1
60 INPUT "Lato 2";L2


100 REM Tasto F1
105 PRINT "Fine programma":END
110 REM Tasto F2
120 FILES:RETURN
```

Nel programma vengono abilitati i tasti funzione F1 e F2. In qualsiasi istante, la pressione di F1 farà terminare il programma, mentre F2 farà visualizzare i file contenuti nella RAM.

POWER istruzione imm. & diff.

POWER consente di spegnere automaticamente il calcolatore dopo un certo intervallo di tempo. La sua sintassi è:

```
POWER T
POWER CONT
POWER OFF
POWER OFF, RESUME
```

dove: T è un intero compreso tra 10 e 255

POWER T fa spegnere automaticamente il sistema dopo T / 10 minuti che non vengono premuti tasti. Ad esempio:

POWER 20 fa spegnere il sistema dopo 2 minuti che non sono stati premuti tasti.

POWER CONT disabilita l' istruzione POWER T. Il sistema rimane acceso anche se non vengono premuti tasti.

POWER OFF spegne immediatamente il sistema; alla sua riaccensione viene visualizzato il menù principale.

POWER OFF, RESUME spegne immediatamente il sistema, ma a differenza dell' istruzione precedente, alla riaccensione (manuale) del sistema, si ritorna allo stato in cui si era prima dello spegnimento. Ad esempio:

```
10 REM Esempio
20 PRINT "Prima dello spegnimento"
30 POWER OFF, RESUME
40 PRINT "Dopo lo spegnimento"
50 END
```

Quando viene eseguita la linea 30, il sistema si spegne automaticamente. Alla sua riaccensione, l' esecuzione del programma continuerà dalla linea 40.

L' istruzione POWER è molto utile al fine di risparmiare energia, specialmente quando il sistema è alimentato a batterie.

3.8 Controllo e manipolazione della memoria.

Abbiamo già visto nel paragrafo due alcuni comandi per la gestione della memoria, considerata nel suo insieme. È ora la volta di alcuni comandi e alcune funzioni volti al controllo ed alla manipolazione della memoria a livello di singolo byte. È importante tuttavia avvisare il lettore che due di questi comandi, POKE e CALL, sono da usarsi con cautela, in quanto se usati senza alcuna cognizione, possono danneggiare file contenuti in RAM, oppure possono mandare in loop il sistema. Vediamo allora questi comandi:

CALL istruzione imm. & diff.

Questa istruzione chiama una subroutine in linguaggio macchina della ROM o della RAM. La sua sintassi è:

CALL I,A,HL

dove: I è l' indirizzo di partenza della subroutine

A è un intero da introdurre nel registro A del microprocessore

HL è un intero da introdurre nel registro HL del microprocessore.

L' indirizzo di partenza deve essere compreso tra -23758 e 65535; per la mappa di memoria consultare l' appendice C. I parametri da inserire nei registri A e HL del processore devono eesere interi compresi tra 0 e 255.

CALL deve essere impiegata con molta cautela: il suo impiego con una locazione di memoria sbagliata può mandare il sistema in un loop senza fine. In tal caso per riprenderne il controllo occorrerà premere il pulsante di RESET posto sul retro della macchina. Questa operazione però cancella tutti i file contenuti in memoria: pertanto prima di fare prove con questa istruzione, salvate i vostri file su cassetta. Eviterete spiacevoli sorprese.

POKE istruzione imm. & diff.

POKE memorizza un byte nell locazione di memoria specificata. La sua sintassi è:

POKE L,B

dove: L è la locazione di memoria interessata

B è il byte da memorizzare

Esempio: *POKE 65535,0* pone a zero il byte 65535.

L' indirizzo della locazione di memoria deve essere un intero compreso tra -32768 e 65535. Il byte da memorizzare deve essere un intero compreso tra 0 e 255.

Il sistema non esegue alcun controllo sulla locazione specificata: pertanto occorre utilizzare quest' istruzione con molta cautela. È possibile infatti danneggiare i file contenuti in memoria.

PEEK funzione imm. & diff.

La funzione PEEK restituisce il contenuto del byte dell' indirizzo specificato. La sua sintassi è la seguente:

PEEK (L)

dove: L è l' indirizzo della locazione di memoria, compresa tra -32768 e 65535.

Il valore restituito da PEEK è un intero compreso tra 0 e 255.

Questa funzione è particolarmente utile per esplorare il contenuto della memoria, byte per byte.

VARPTR funzione imm. & diff.

VARPTR restituisce l' indirizzo del primo byte di una variabile, o l' indirizzo di un file nella directory. La sua sintassi è:

VARPTR (X)
VARPTR (# Nf)

dove: X è una variabile qualsiasi

Nf è il numero con il quale è stato aperto il file.

Esempio: *VARPTR(A)* fornisce l' indirizzo del primo byte della variabile A.

Se l' argomento di VARPTR è una variabile, o una matrice, viene restituito l' indirizzo del primo byte. Viceversa se l' argomento è un file, (al momento dell' esecuzione di VARPTR il file deve essere aperto), viene restituito l' indirizzo del file nella directory. Ad esempio:

```
OPEN "Prova.do" FOR INPUT AS 1
X = VARPTR(# 1)
```

L' indirizzo del file in directory è dato da PEEK(X + 2) + (PEEK(X + 3)) * 256.

Il valore fornito da VARPTR sarà compreso tra -32768 e 65535.

Le variabili utilizzate come argomenti della funzione devono avere avuto già assegnato

un valore, viceversa si verifica un errore di chiamata di funzione illegale (?FC Error). Nel caso invece in cui il file passato come argomento non è aperto al momento della chiamata di VARPTR, compare un messaggio di errato numero di file (?BN Error).

Capitolo 4

Comandi di editing e relativi ai formati.

La correzione e la modifica dei programmi è una delle attività più frequenti quando si adopera il calcolatore per la programmazione. Il BASIC dell' M10 utilizza per l' editing dei programmi la funzione integrata TEXT, il text editor per la scrittura e la correzione di testi e programmi. Ad esso è interamente dedicato il primo paragrafo di questo capitolo.
Il secondo paragrafo descrive invece i comandi BASIC di editing, mentre il terzo ed il quarto trattano di comandi e funzioni relativi alla gestione del video e della stampante. Infine il quinto esamina i comandi relativi ai formati.

4.1 L' edit.

Nell' esaminare l' editing dei programmi, occorre distinguerne la creazione dalla modifica e correzione.
Esistono due modi differenti di creare un programma BASIC. Il primo è quello di porsi in ambiente BASIC e digitare una sequenza di istruzioni precedute da un numero detto numero di riga e concluse dalla pressione del tasto ENTER. Ad esempio digitando:

```
10 REM Inizio [ENTER]
20 PRINT "Ciao" [ENTER]
30 END [ENTER]
```

si scrive nella memoria di lavoro un piccolo programma di 3 linee.

Nel caso di errori di battitura, è possibile inserire la nuova linea di programma, opportunamente corretta. Nell' esempio visto prima, supponendo che la linea 20 dovesse con-

tenere *PRINT "Ciao !"*, potremmo digitare la nuova linea:

20 PRINT "Ciao !" [ENTER]

che sostituirà la vecchia linea.

Naturalmente questo modo di procedere è particolarmente lento e noioso, specialmente quando si impostano linee di programma con molte istruzioni. Esiste però, per fortuna, la possibilità di scrivere programmi tramite la funzione integrata TEXT. È possibile infatti adoperare questa funzione per creare dei programmi BASIC: per far ciò è necessario entrare in TEXT e scrivere il programma come un testo qualsiasi (consultare parte seconda, cap. 11). Il programma così redatto sarà di tipo .DO; tramite il comando *LOAD "RAM:* è possibile caricarlo nella memoria di lavoro ed eseguirlo. Un successivo SAVE senza l' opzione A salverà il programma in formato .BA (per i comandi LOAD e SAVE consultare il cap. 3, par. 4).

Dopo aver visto come è possibile creare dei programmi, vediamo ora come questi possono essere modificati e corretti. Esiste a tal scopo un comando BASIC, EDIT, tramite il quale è possibile utilizzare la funzione TEXT per editare il programma. Ma vediamo ora in dettaglio questo comando:

EDIT comando imm. & diff.

EDIT consente di passare il programma contenuto nella memoria di lavoro alla funzione TEXT; la pressione del tasto funzione F8 ritorna il controllo al BASIC, stato comandi. La sintassi di EDIT è così definita:

EDIT
EDIT n
EDIT n-
EDIT n-m
EDIT -m

dove: n è il numero della prima riga da passare a TEXT

m è il numero dell' ultima riga da passare a TEXT.

EDIT trasferisce a TEXT tutto il programma contenuto nella memoria di lavoro, mentre EDIT n trasferisce soltanto la linea specificata. EDIT n-m trasferisce a TEXT la porzione di programma compresa tra la linea n ed m, EDIT n- trasferisce il programma a partire dalla linea n, EDIT -m fino alla linea m.

Il programma da editare deve essere contenuto nella memoria di lavoro. Nel caso di eventuali messaggi di memoria insufficiente (?OM Error), rieseguire il comando EDIT specificando un minor numero di linee di programma.

In EDIT sono utilizzabili tutti i comandi di TEXT, tranne il riposizionamento a capo automatico. Per una dettagliata spiegazione su TEXT, si rimanda al capitolo 11, parte seconda.

Terminato il lavoro di correzione del programma, la pressione del tasto F8 restituisce dopo alcuni istanti il controllo al BASIC, stato comandi. Ogni linea di programma deve cominciare con un numero di riga: in caso contrario la funzione integrata TEXT rifiuterà il testo e visualizzerà il messaggio *Text ill-formed Press space bar for TEXT*. Premendo la barra spaziatrice si tornerà così al text editor per correggere nuovamente il programma.

4.2 Comandi di visualizzazione programmi, file e funzioni.

È ora la volta di quattro comandi destinati alla visualizzazione del contenuto della memoria di lavoro, della RAM, e delle stringhe assegnate ad i tasti funzione.

LIST comando imm. & diff.

LIST consente di visualizzare il programma contenuto nella memoria di lavoro. La sua sintassi è così definita:

```
LIST
LIST n
LIST n-
LIST n-m
LIST -m
```

dove: n è il numero di riga della prima linea da visualizzare

m è il numero di riga dell' ultima linea da visualizzare

LIST visualizza tutto il programma contenuto nella memoria, LIST n solo la riga indicata, LIST n- dalla riga n in poi, LIST n-m dalla riga n alla m, LIST -m fino alla linea m.

A meno di diverse definizioni, al tasto funzione F5 è assegnato in ambiente BASIC il comando LIST.

LIST può essere interrotto premendo il tasto BREAK, oppure i tasti CTRL e C contemporaneamente. Per fermare temporaneamente il listing, premere il tasto PAUSE oppure i tasti CTRL ed S contemporaneamente. Per continuare, ripremere gli stessi tasti

LLIST comando imm. & diff.

LLIST permette di visualizzare sulla stampante parallela il programma contenuto nella memoria di lavoro. La sua sintassi è simile a quella di LIST:

LLIST
LLIST n
LLIST n-
LLIST n-m
LLIST -m

dove: n ed m hanno lo stesso significato che nel comando LIST.

LLIST ha la stessa funzione del comando LIST, eccetto che la visualizzazione del programma è effettuata su stampante piuttosto che sul display. Pertanto si vedano le note relative a LIST.

FILES comando imm. & diff.

Questo comando elenca i file contenuti nella memoria RAM. La sua sintassi è:

FILES

FILES visualizza i file contenuti in RAM indicando per ognuno di essi il nome ed il suffisso, che ne specifica la natura:

.BA identifica i programmi BASIC
.CO identifica i programmi in linguaggio macchina
.DO identifica i file dati o i programmi BASIC formato ASCII

Il comando FILES è, a meno di successive ridefinizioni, assegnato al tasto funzione F1.

KEY LIST comando imm. & diff.

KEY LIST elenca le stringhe assegnate ad i tasti funzione. La sua sintassi è così definita:

KEY LIST

L' elenco delle stringhe viene fornito nel seguente formato:

F1	F2
F3	F4
F5	F6
F7	F8

Ad esempio eseguendo KEY LIST si ottiene la seguente lista:

Files	Load "
Save "	Run
List	
	Menu

a meno che i tasti funzione non siano stati ridefiniti.

Un elenco delle stringhe associate ai tasti funzione, sia pure in forma più sintetica, può anche essere ottenuto premendo il tasto LABEL.

4.3 Gestione del video.

Il video a cristalli liquidi dell' M10 è visto dal sistema come una matrice composta di 8 righe e 40 colonne, per un totale di 320 elementi. Il video esterno possiede invece una risoluzione verticale maggiore, 24 righe, e lo stesso numero di colonne. Il cursore video rappresenta la posizione in cui verrà visualizzato il prossimo carattere. Le sue coordinate possono variare tra 0 e 39 per quanto riguarda l' asse X (le colonne), e tra 0 e 7 per l' asse Y (le righe) per il display LCD, tra 0 e 23 per il video esterno. La posizione (0,0) corrisponde all' angolo superiore sinistro.

In questo paragrafo esamineremo alcuni comandi BASIC rivolti alla gestione del video: CLS per cancellare il display, POS e CSRLIN per restituire le coordinate orizzontali e verticali del cursore video, TAB per posizionarsi sul display o sulla linea di stampa, LPOS per restituire la posizione orizzontale della testina.

CLS comando imm. & diff.

CLS cancella il display. La sua sintassi è:

CLS

Questo comando cancella l' intero display e riporta il cursore video nella posizione (0,0) (prima riga e prima colonna). Se sul video sono visualizzate le stringhe assegnate ai tasti funzione, queste non vengono cancellate. Per rimuoverle occorrerà ripremere il tasto LABEL.

POS funzione imm. & diff.

La funzione POS ritorna la posizione attuale del cursore video. La sua sintassi è la seguente:

POS(X)

dove: X è un argomento numerico fittizio.

Il valore restituito indica su quale colonna è posizionato il cursore video. Il margine sinistro coincide con la colonna zero.

Esempio: *PRINT "Ciao !";POS(0)* farà comparire sul video:

Ciao ! 6

dove 6 è la colonna sulla quale risulta posizionato il cursore.

CSRLIN funzione imm. & diff.

CSRLIN fornisce la posizione verticale (riga) del cursore video. La sua sintassi è:

CSRLIN

Il valore restituito dalla funzione è compreso tra 0 e 7 nel caso del display a cristalli liquidi, tra 0 e 24 nel caso del video esterno. La riga zero è la prima riga del display, cioè quella più in alto.

Esempio:
```
10 REM Esempio CSRLIN
20 CLS:PRINT "Olivetti M10"
30 PRINT CSRLIN
40 END
```

Questo programma visualizzerà le seguenti informazioni:

```
Olivetti M10
1
```

Mediante le funzioni POS e CSRLIN è quindi possibile conoscere le coordinate del cursore video. Ciò è particolarmente utile sia in giochi di animazione, che in routine che interagiscono con il display.

TAB funzione imm. & diff.

La funzione TAB è adoperata nelle istruzioni PRINT ed LPRINT (vedi cap. 6) per posizionare il cursore video o la testina di stampa nel punto desiderato di una linea. La sua sintassi è:

TAB(X)

dove: X è un intero compreso tra 0 e 255.

TAB(X) posiziona il cursore video o la testina di stampa ad X caratteri dal margine sinistro: nel caso in cui il valore di X è minore dell' attuale posizione del cursore o della testina, il successivo carattere verrà visualizzato o stampato alla posizione corrente.

Il valore zero corrisponde al margine sinistro. Sul display a cristalli liquidi le posizioni superiori a 39 si riferiscono alle righe successive secondo questo schema:

Riga 0	0	39
Riga 1	40	79
Riga 2	80	119
Riga 3	120	159
Riga 4	160	199
Riga 5	200	239
Riga 6	240	255

Esempio: PRINT "M10";TAB(10);"Ciao";TAB(120);"a tutti !"

visualizzerà: M10 Ciao

a tutti !

LPOS funzione imm. & diff.

LPOS ritorna la posizione attuale della testina della stampante parallela. La sua sintassi, simile a quella di POS, è:

LPOS(X)

dove: X è un argomento numerico fittizio.

LPOS fornisce la posizione della testina di stampa nell' ambito del buffer di linea della stampante. È importante considerare che il valore fornito da LPOS specifica a quale posizione della linea verrà stampato il prossimo carattere, e non la posizione fisica della testina.

4.4 Attributi video.

Oltre alle funzioni ed ai comandi visti al paragrafo precedente, esistono delle sequenze particolari di caratteri per la gestione del video. Tali sequenze sono dette sequenze di ESCAPE poiché il loro primo carattere è il codice ASCII 27, corrispondente al comando ESC. Questo carattere non è generabile da tastiera, e deve pertanto essere generato mediante la funzione stringa CHR$ (vedi cap. 5).

La sintassi tipo di queste sequenze è così definita:

```
PRINT CHR$(27) + "C"
PRINT CHR$(27) + CHR$(n)
PRINT CHR$(27);"C"
PRINT CHR$(27);CHR$(n)
```

dove: C è il secondo carattere della sequenza

n è il codice ASCII di C.

Queste sequenze possono essere prodotte sia in modo immediato che differito; vediamone ora le funzioni:

ESC + E ed ESC + j	cancellano il display e riportano il cursore in posizione (0,0). Hanno la stessa identica funzione del comando CLS
ESC + K	cancella, a partire dalla posizione del cursore video, fino a fine riga
ESC + J	cancella, a partire dalla riga successiva del cursore, fino a fine pagina
ESC + I	cancella interamente la riga su cui è posizionato il cursore video
ESC + L	inserisce una riga vuota nella posizione del cursore, scalando verso il basso le rimanenti

ESC + M	rimuove la riga su cui è posto il cursore, scalando verso l' alto le rimanenti
ESC + A	sposta il cursore alla riga precedente
ESC + B	sposta il cursore alla riga successiva
ESC + C	sposta il cursore di uno spazio a destra
ESC + D	sposta il cursore di uno spazio a sinistra
ESC + H	porta il cursore nella posizione (0,0), senza cancellare il video
ESC + p	abilita la visualizzazione dei caratteri in negativo
ESC + q	disabilita la visualizzazione dei caratteri in negativo
ESC + T	disabilita la scrittura sulla riga 7 del display
ESC + U	abilita la scrittura sulla riga 7 del display
ESC + W	abilita lo scorrimento (scrolling) del display
ESC + V	disabilita lo scorrimento (scrolling) del display
ESC + P	rende visibile il cursore
ESC + Q	rende invisibile il cursore
ESC + Yyx	posiziona il cursore video nel punto di coordinate x,y , dove x rappresenta la riga e y la colonna. I valori di x e y devono essere sommati a 32, codice ASCII del carattere spazio.

Ad esempio: PRINT CHR$(27) + "Y" + " " + " "

posiziona il cursore nella posizione (0,0)

```
10 REM Prova della sequenza ESC Y
20 PRINT "Ciao ":PRINT CHR$(27);"Y";CHR$(34);CHR$(42)
30 END
```

visualizzerà la parola CIAO sulla terza riga, decima colonna.

4.5 Comandi relativi ai formati

In questo paragrafo esamineremo tre istruzioni relative ai formati: REM, SCREEN e WIDTH. La prima consente di inserire dei commenti all' interno di un programma, la seconda definisce il tipo di display utilizzato, la terza infine definisce la larghezza del video esterno. Vediamo allora in dettaglio queste istruzioni:

REM istruzione diff.

REM consente di inserire dei commenti in un programma. La sua sintassi è:

```
REM
'
```

REM, che può anche essere utilizzata nella sua forma abbreviata ('), non influenza l' esecuzione del programma, ma compare nel suo listato. Tutto il testo presente dopo la parola chiave REM (o dopo l' apostrofo) viene considerato come commento; pertanto REM deve essere l' ultima istruzione della riga di programma (o la prima se questa contiene solo REM).

Esempio: 10 REM Questo è un commento

REM può essere impiegata anche in modo immediato, anche se ciò non ha molto senso. REM non può seguire un' istruzione DATA: in questo caso infatti verrebbe interpretata come dato.

SCREEN istruzione imm. & diff.

SCREEN definisce il tipo di display adoperato. La sua sintassi è la seguente:

```
SCREEN d,t
```

dove: d è un' espressione numerica che specifica il display

t è un' espressione numerica che abilita o disabilita la visualizzazione delle stringhe assegnate ai tasti funzione

D può essere 0 o 1: se d = 0 la visualizzazione dei dati sarà effettuata sul display LCD, viceversa sul video esterno. Analogamente per t: se t = 1 è abilitata la visualizzazione del significato dei tasti funzione, viceversa è disabilitata.

Il valore di default di d e t è 0. L' esecuzione di SCREEN con valori diversi da quanto indicato, provoca un errore di chiamata irregolare di funzione (?FC Error), errore che si verifica anche nel caso in cui si definisce d = 1 ma non risulta collegato il video esterno.

WIDTH istruzione imm. & diff.

WIDTH definisce il formato di visualizzazione del video esterno. La sua sintassi è:

WIDTH n

dove: n è un' espressione numerica il cui valore deve essere 40 o 80

WIDTH definisce il formato del video esterno a 40 o ad 80 colonne. Valori differenti di n provocano messaggi di chiamata irregolare di funzione.

Capitolo 5

Matrici e stringhe

Ogniqualvolta occorre visualizzare e manipolare delle parole in un programma, è necessario adoperare delle stringhe, sia in forma di costanti che di variabili.
Una stringa è una sequenza di massimo 255 caratteri (lettere, cifre, simboli vari), codificati secondo lo standard internazionale ASCII. Ogni carattere è identificato da un codice: ad esempio la lettera A maiuscola corrisponde al numero 65, lo zero a 48, e così via (vedi appendice D). Non tutti i caratteri hanno una rappresentazione visiva: mentre il codice 65 rappresenta la a maiuscola, il codice 13 che corrisponde all' a capo (ENTER), non possiede alcuna rappresentazione visiva. Il basic dell' M10 gestisce quindi le stringhe come sequenze di numeri (codici). Ad esempio la stringa:

C i a o

viene codificata come:

67 105 97 111.

Una stringa è quindi caratterizzata dalla lunghezza, ovverossia dal numero di caratteri che la compongono, e dalla sequenza dei codici contenuti.
Due sono i tipi fondamentali di stringhe: il tipo costante e quello variabile.

Una costante stringa è una sequenza di caratteri delimitata da doppi apici. La sua sintassi è così definita:

"abc...."

dove: a, b, c sono dei caratteri ASCII qualsiasi.

La lunghezza massima di una costante stringa è di 255 caratteri.

Ad esempio: "Ciao !"

"Questa è una costante stringa"

sono due costanti stringa.

La stringa "" è chiamata la stringa nulla: essa è la più semplice costante stringa realizzabile. Facendo un parallelo con i numeri, la stringa nulla corrisponde allo zero.

Una variabile stringa è una sequenza di massimo 255 caratteri identificata da un nome terminante con il simbolo del dollaro ($). Il nome di una variabile stringa deve soddisfare gli stessi requisiti di una qualsiasi variabile numerica (vedi capitolo 2). Per completezza ne riportiamo comunque la sintassi:

X$

dove: X è un nome di lunghezza qualsiasi, iniziante con una lettera (maiscola o minuscola).

Il basic dell' M10 riserva di default alle variabili stringa 256 byte (un byte = un carattere). Se nel programma sono contenute variabili stringa che complessivamente occupano più di 256 byte, viene visualizzato un messaggio d' errore di insufficiente spazio per le stringhe (?OS Error). È naturalmente possibile aumentare questa quantità di memoria tramite il comando CLEAR, per la cui spiegazione rimando al capitolo 3, paragrafo 2.

L' operatore matematico " + " ha, per le stringhe, un particolare significato: esso rappresenta infatti la concatenazione di stringhe, l' operazione cioè di congiunzione. Per esempio:

X$ = "Olivetti " + "M 10" assegna alla stringa X$ "Olivetti M 10".

Nel seguito di questo capitolo adopereremo spesso la dicitura "espressione stringa": sarà bene chiarirne pertanto il significato. Un' espressione stringa può essere:

La stringa nulla ("")
Una costante stringa (es. "M 10")
Una variabile stringa (es. X$)
Una qualsiasi funzione stringa (es. CHR$)
Un concatenamento qualsiasi dei suddetti termini

Esempio: ""

"Questa è un' espressione stringa"

X$

MID$(X$,1,5)

X$ + "Olivetti" + LEFT$(Z$,4)

sono tutte espressioni stringa.

5.1 Funzioni di conversione.

Abbiamo già visto come sono codificate le stringhe dall' interprete del computer. Ogni carattere è contraddistinto da un codice e da un simbolo grafico (che in alcuni casi può essere "invisibile"). Il basic dell' M10 possiede due funzioni dedicate a ciò: ASC e CHR$ che restituiscono rispettivamente il codice ASCII e il simbolo della stringa fornita come argomento.
La conversione da numero a stringa e viceversa è invece realizzabile tramite altre due funzioni: STR$ e VAL, ambedue descritte in questo paragrafo.

ASC funzione imm. & diff.

ASC ritorna il codice ASCII del primo carattere di un' espressione stringa. La sua sintassi è la seguente:

ASC(X$)

dove: X$ è un' espressione stringa.

Esempio: *PRINT ASC("Ciao !")* restituisce il numero 67, codice ASCII di C.

Passando la stringa nulla come argomento alla funzione ASC, si provoca un errore di chiamata irregolare di funzione (?FC Error).
ASC risulta molto utile per conoscere la composizione di stringhe che contengono caratteri ASCII non rappresentabili.

CHR$ funzione imm. & diff.

Questa funzione restituisce il carattere ASCII del numero di codice specificato. La sua sintassi è così definita:

CHR$(X)

dove: X è un' espressione numerica indicante il codice ASCII del carattere

Esempio: *PRINT CHR$(37)* visualizza il simbolo %.

Il valore dell' espressione deve essere un intero compreso tra 0 e 255. Nel caso in cui il valore sia un numero reale, questo viene trasformato in numero intero.

CHR$ è una funzione indispensabile per creare i caratteri ASCII non generabili da tastiera. Ad esempio il carattere ESC (codice 27), che non è generabile da tastiera, può essere ottenuto digitando *CHR$(27)*.

VAL funzione imm. & diff.

La funzione VAL converte il valore numerico della stringa in un numero di pari valore. La sua sintassi è:

VAL(X$)

dove: X$ è un' espressione stringa.

VAL restituisce il numero contenuto dalla stringa X$. Nella conversione vengono ignorati gli spazi, le tabulazioni, le interlinee, nonché i caratteri non numerici che seguono l' ultima cifra del numero. Nel caso in cui la stringa non contenga numeri, o se questi non sono espressi correttamente, viene restituito il numero zero.

Esempio: PRINT VAL("34")

PRINT VAL(" 3 4")

PRINT VAL(" 34ciao")

visualizzano 34, mentre:

PRINT VAL("M 10") e

PRINT VAL("*56")

restituiscono zero.

STR$ funzione imm. & diff.

La funzione STR$ converte il valore di un' espressione numerica in una stringa. La sua sintassi è la seguente:

STR$(X)

dove: X è un' espressione numerica.

STR$ restituisce una stringa di lunghezza pari al numero di cifre del valore dell' espressione, più eventualmente il segno meno e/o il punto decimale.

Esempio: *PRINT STR$(12-16.4)*

visualizza la stringa "-4.4" di lunghezza 4, i cui codici ASCII sono nell' ordine 45, 52, 46, 52.

5.2 Funzioni per la gestione delle stringhe.

In questo paragrafo esamineremo alcune funzioni per la gestione delle stringhe. Abbiamo definito le stringhe come sequenze di massimo 255 caratteri; chiameremo d' ora in avanti sottostringhe qualsiasi loro partizione. Le funzioni che descriveremo tra breve gestiscono proprio l' estrazione di sottostringhe da stringhe, la loro ricerca, la restituzione del numero di caratteri contenuti in una stringa. Vediamo allora queste funzioni:

RIGHT$ funzione imm. & diff.

RIGHT$ restituisce una sottostringa di caratteri posti nella parte destra della stringa specificata. La sua sintassi è:

RIGHT$(X$,I)

dove: X$ è un' espressione stringa

I è un' espressione numerica, il cui valore deve essere compreso tra 0 e 255.

RIGHT$ restituisce gli ultimi I caratteri della stringa X$ (i più a destra). Se I è zero, viene restituita la stringa nulla, mentre se I è maggiore della lunghezza della stringa, viene restituita la stringa intera. Nel caso in cui il valore fornito dall' espressione numerica non è intero, questo viene troncato all' intero più vicino.

Esempio: *PRINT RIGHT$("Ecco una stringa",7)* visualizza la parola stringa.

Se il valore dell' espressione numerica è minore di zero, oppure maggiore di 255, viene visualizzato un errore di chiamata irregolare di funzione (?FC Error).

La funzione RIGHT$ può essere sostituita da MID$ nel seguente modo:

RIGHT$(X$,I) = MID$(X$,LEN(X$) + 1-I)

LEFT$ funzione imm. & diff.

LEFT$ restituisce una sottostringa di caratteri posti nella parte sinistra della stringa specicata. La sua sintassi è così definita:

LEFT$(X$,I)

dove: X$ è un' espressione stringa

I è un' espressione numerica, il cui valore deve essere compreso tra 0 e 255.

LEFT$, funzione simmetrica di RIGHT$, restituisce i primi I caratteri di sinistra della stringa X$. Se I è uguale a zero viene restituita la stringa nulla, mentre se I è maggiore della lunghezza di X$, viene restituita l' intera stringa.

Esempio: *PRINT LEFT$("Ecco una stringa",4)* visualizza la parola Ecco.

Nel caso in cui il valore fornito dall' espressione numerica sia minore di zero oppure maggiore di 255, viene visualizzato un errore di chiamata irregolare di funzione (?FC Error).

LEFT$ può essere sostituita dalla funzione MID$ nel seguente modo:

LEFT$(X$,I) = MID$(X$,1)

MID$ funzione imm. & diff.

MID$ fornisce una sottostringa della stringa specificata. La sua sintassi è:

MID$(X$,I,L)

dove: X$ è un' espressione stringa

I ed L sono due espressioni numeriche, i cui valori devono essere compresi tra 0 e 255

MID$ restituisce una sottostringa che comincia dal carattere I ed è lunga L caratteri. Se I è zero oppure è maggiore del numero dei caratteri di X$, viene restituita la stringa nulla. Se L è maggiore della lunghezza di X$, oppure se omesso, viene restituita tutta la stringa a partire dal carattere I.

Esempio: *PRINT MID$("Ecco una stringa",6,3)* visualizza la parola una.

La chiamata di MID$ con valori di I ed L diversi da quelli consentiti, provoca un errore di chiamata irregolare di funzione.

LEFT$, RIGHT$ e MID$ sono funzioni e pertanto possono essere adoperate soltanto a destra di un segno di uguaglianza (=); non possono quindi essere impiegate direttamente per assegnare una sottostringa ad una stringa. Nel caso occorra fare ciò, è necessario operare nel seguente modo:
supponendo di avere la stringa X$ = "Calcolatore Olivetti M10", e di volere aggiungere la parola "portatile" dopo la prima, bisogna suddividere X$ in due, e quindi aggiungervi l' aggettivo desiderato.

Esempio: X$ = "Calcolatore Olivetti M10"
X$ = LEFT$(X$,12) + "portatile " + MID$(13)

Adoperando LEFT$ e MID$ concatenate assieme alla sottostringa da aggiungere, si ottiene la stringa desiderata.

LEN funzione imm. & diff.

LEN restituisce la lunghezza di una stringa. La sua sintassi è:

LEN(X$)

dove: X$ è un' espressione stringa

LEN restituisce il numero di caratteri che compongono la stringa, compresi i caratteri non stampabili, le tabulazioni e gli spazi. Se a LEN viene passato come argomento una variabile o un' espressione numerica, viene visualizzato un errore di tipo di variabile errato (?TM Error).

5.3 Funzioni per la creazione di stringhe.

Il basic dell' M10 possiede due funzioni per la creazione di stringhe: SPACE$ e STRING$. La prima consente di creare una stringa di spazi della lunghezza specificata, mentre la seconda permette di realizzare stringhe con qualsiasi carattere ASCII.

SPACE$ funzione imm. & diff.

SPACE$ restituisce una stringa di spazi della lunghezza specificata. La sua sintassi è la seguente:

SPACE$(X)

dove: X è un' espressione numerica, il cui valore deve essere compreso tra 0 e 255.

Esempio: *PRINT "Tanti";SPACE$(5);"bei";SPACE$(5);"spazi !"*

visualizza: Tanti bei spazi !

Se il valore dell' espressione non è un numero intero, viene ignorata la parte dopo la virgola decimale. Il tentativo di passare come argomento un' espressione il cui valore non rientra nei limiti specificati dà luogo ad un errore di chiamata irregolare di funzione (?FC Error).

STRING$ funzione imm. & diff.

STRING$ crea una stringa di caratteri di lunghezza specificata. La sua sintassi è:

STRING$(N,C)
STRING$(N,X$)

dove: N e C sono espressioni numeriche, i cui valori devono essere compresi tra 0 e 255

X$ è un' espressione stringa.

STRING$ crea una stringa di N caratteri tutti uguali al codice ASCII C oppure all' espressione stringa X$.

Esempio: *PRINT STRING$(20,"k")*

visualizza: kkkkkkkkkkkkkkkkkkkk

Se X$ ha lunghezza maggiore di uno, viene considerato solo il primo carattere. Anche in questo caso, la chiamata della funzione con parametri non consentiti, provoca un errore di chiamata irregolare di funzione (?FC Error).

5.4 Matrici

Naturale estensione del concetto di variabile, sia di tipo numerico che di tipo stringa, sono le matrici. Una matrice è una struttura dati omogenea, un insieme cioè di variabili, tutte aventi lo stesso nome, identificate da uno o più numeri. Fattori caratterizzanti le matrici sono il numero di dimensioni ed il numero di elementi. Il basic dell' Olivetti M10 consente una facile gestione delle matrici, che diversamente dalle variabili, necessitano di essere definite prima del loro uso. Descriviamo allora questa istruzione, per poi esaminare alcuni esempi sull' uso delle matrici.

DIM istruzione imm. & diff.

DIM definisce il numero di dimensioni ed il numero di elementi di una matrice. La sua sintassi è così definita:

DIM A(d1,d2...dn),B(d1,d2,...dn),..

dove: A e B sono variabili numeriche o stringa

d1,d2,dn sono espressioni numeriche che specificano il numero di elementi per ciascuna dimensione.

DIM definisce una o più matrici contemporaneamente, ciascuna separata dalle altre da una virgola. D1,d2,..,dn definiscono il numero di elementi per ogni dimensione, numero cui va aggiunto uno poiché la numerazione degli elementi parte da zero.

Esempio: DIM A(100),B$(10,5),C(4,4,5)

definisce rispettivamente una matrice numerica ad una dimensione di 101 elementi, una matrice di stringhe a due dimensioni di 11 e 6 elementi ciascuna, una matrice numerica di tre dimensioni di 5, 5 e 6 elementi.

In caso di utilizzazione di matrici senza previo dimensionamento, l' interprete del linguaggio assegna come valori di default 10 elementi per ogni dimensione. È comunque preferibile eseguire DIM prima di adoperare matrici, la cui definizione inizializza tutti gli elementi a zero nel caso di matrice numerica, a stringa nulla nel caso di matrice di tipo stringa.

Non esistono limitazioni né per il numero di dimensioni, né per il numero di elementi: unico vincolo presente è quello della quantità di memoria disponibile. Messaggi di errore di memoria insufficiente (?OM Error), possono occorrere quando si gestiscono matrici numeriche troppo grandi; analogamente messaggi di insufficiente spazio per le stringhe (?OS Error) possono verificarsi adoperando matrici di tipo stringa: in questo caso aumentare lo spazio a disposizione con CLEAR (cap. 3, par. 2).

Errori di indici fuori dai limiti (?BS Error), possono verificarsi quando si indirizza un elemento della matrice oltre i limiti fissati dalla DIM.
Dimensionare una matrice già definita in precedenza da un' istruzione DIM o per default dal sistema, provoca un errore di duplicazione di dimensionamento (?DD Error). CLEAR, NEW e RUN cancellano completamente le matrici.

Esempi di uso di matrici

Per poter utilizzare le matrici è allora necessario prima di tutto definirle con l' istruzione DIM: Fatto ciò possiamo tranquillamente adoperare la nostra matrice: per poter indirizzarne un elemento, occorrerà specificarne la posizione. Vediamo subito un esempio:

```
10 REM Programma di prova
20 DIM T(10)
30 FOR I=1 TO 10
40 T(I)=7*I
50 NEXT I
60 END
```

Il programma qui riportato definisce una matrice ad una dimensione di 11 elementi (linea 20), e nel ciclo di FOR delle linee 30-50 memorizza nel vettore la tabellina del 7 (linea 40).

Altro tipico utilizzo di matrici, questa volta a due dimensioni, è la tabella pitagorica o la battaglia navale.

```
10 REM Tabella pitagorica
20 DIM T(10,10)
30 FOR I = 1 TO 10
40    FOR J = 1 TO 10
50    T(I,J) = I * J
60    PRINT T(I,J);
70    NEXT J:PRINT
80 NEXT I
90 END
```

Capitolo 6

Comandi di input / output

Notevole importanza rivestono le istruzioni e le funzioni connesse all' immissione / emissione (input / output) di dati. Possiamo distinguere, relativamente all' argomento, quattro aspetti differenti: l' input dati da programma, l' output, la gestione dei file e quella delle periferiche. Sono proprio questi i titoli dei quattro paragrafi in cui è suddiviso questo capitolo.

6.1 Input dati da programma.

Il termine dato merita qualche approfondimento, considerato che è intorno ad esso che è in definitiva imperniato questo capitolo. Qualsiasi programma, anche il più banale, necessita di dati, cioè di informazioni. Queste informazioni possono essere costanti, definitive, oppure possono variare in base ad alcuni parametri, oppure ancora devono essere inserite dall' utilizzatore del programma. Nel primo caso si parlerà allora di costanti, sia numeriche che stringa, nel secondo invece di variabili dipendenti da altre, nell' ultimo di dati di ingresso o di input. Le costanti possono essere inserite esplicitamente nel programma, in un qualsiasi punto di esso senza adoperare alcuna istruzione basic. Talvolta però, quando si hanno parecchie costanti da assegnare a delle variabili, piuttosto che scrivere lunghe e noiose sequenze di assegnamenti, si preferisce raggruppare le costanti all' inizio o più spesso alla fine del programma mediante l' istruzione DATA. Grazie poi alla coppia di istruzioni READ e RESTORE è possibile leggere ed assegnare queste costanti alle variabili specificate. Nel caso in cui invece è necessario consentire all' utilizzatore del programma l' immissione di dati, occorrerà utilizzare istruzioni adatte. INKEY$, INPUT, INPUT$ e LINE INPUT sono quattro istruzioni differenti, dedicate, come il loro stesso nome evidenzia, all' input di dati. Cominciamo allora ad esaminare questi comandi:

LET istruzione imm. & diff.

LET assegna un valore ad una variabile. La sua sintassi è così definita:

LET X = En
LET X$ = E$

dove: X è una variabile numerica

X$ è una variabile di tipo stringa

En è un' espressione numerica

E$ è un' espressione stringa.

La parola chiave LET è opzionale e pertanto può essere omessa.

Esempio: LET PI = 3.14159

A$ = "Questo è un assegnamento"

sono due assegnamenti, il primo nella forma completa (con la parola LET), il secondo senza.

È importante assegnare valori consentiti alle variabili, viceversa possono verificarsi errori di tipo di variabile errato (?TM Error), di supero di capacità (?OV Error) o infine di insufficiente spazio per le stringhe (?OS Error).

DATA istruzione diff.

DATA memorizza costanti numeriche e stringa che possono essere lette mediante l' istruzione READ. La sua sintassi è:

DATA c1,c2,....,cn

dove: c1, c2 ... cn sono costanti numeriche o stringa.

È possibile inserire sulla stessa linea costanti sia numeriche che stringa; quest' ultime non devono essere racchiuse obbligatoriamente tra virgolette, a meno che non contengano spazi iniziali o finali, virgole o punti. Non esiste un limite al numero di costanti inseribili, salvo non superare la massima lunghezza consentita per una linea di programma basic. Per una spiegazione più esauriente su DATA, consultare l' istruzione READ.

Esempio: *DATA 1,Esempio di data,-45,"fine"*

memorizza due costanti numeriche, 1 e -45, e due costanti stringa.

READ istruzione imm. & diff.

READ assegna valori memorizzati con istruzioni DATA alle variabili specificate. La sua sintassi è:

READ v1,v2,....vn

dove: v1, v2, ... vn sono variabili numeriche o stringa.

READ legge dati da una o più istruzioni DATA in sequenza (per ordine di numero di linea). Le variabili indicate devono essere dello stesso tipo dei dati memorizzati, altrimenti viene evidenziato un errore di sintassi (?SN Error).

Esempio: *READ A,B$,C%,D$*

assegna alle variabili indicate dati memorizzati con un' istruzione DATA.

Non è necessario che la o le istruzioni DATA precedano le READ: l' interprete basic, quando viene eseguita quest' ultima istruzione, ricerca i dati in tutto il programma, procedendo dal numero di linea più basso verso il più alto. La prima variabile della prima istruzione READ sarà associata al primo elemento della prima DATA, la seconda al secondo, e così via. Le successive READ cominceranno a leggere a partire dal primo dato non ancora letto.

È possibile non assegnare tutti i valori memorizzati con delle DATA, viceversa è impossibile leggere più dati di quelli disponibili: in questo caso viene visualizzato un messaggio di dati mancanti (?OD Error).

READ può anche essere adoperata in modo immediato per leggere dati memorizzati in istruzioni DATA contenute in un programma in memoria di lavoro. Se invece non esiste alcun programma, o se in questo non esistono DATA, l' esecuzione in modo immediato di READ provoca un errore di dati mancanti (?OD Error).

RESTORE istruzione imm. & diff.

RESTORE consente di rileggere i dati memorizzati nelle istruzioni DATA. La sua sintassi è la seguente:

```
RESTORE
RESTORE n
```

dove: n è un numero di riga.

RESTORE fa si che la successiva READ legga il primo dato della prima istruzione DATA contenuta nel programma, RESTORE n il primo elemento della prima DATA della linea n.

Esempio:

```
10 REM Esempio di RESTORE
20 DATA 1,2,3
30 READ A,B
40 RESTORE
50 READ C
60 PRINT A,B,C
70 END
```

L' esecuzione di questo programma fa visualizzare i numeri 1, 2, 1, essendo presente alla linea 40 RESTORE che fa assegnare alla variabile C il primo dato della prima istruzione DATA (linea 20).

Nel caso in cui la linea di programma specificata non esiste, compare un errore di linea indefinita (?UL Error).

INKEY$ funzione diff.

INKEY$ restituisce una stringa di un solo carattere corrispondente al tasto premuto sulla tastiera. La sua sintassi è:

```
INKEY$
```

INKEY$ restituisce il primo carattere presente nel buffer della tastiera, oppure la stringa nulla nel caso in cui il buffer sia vuoto (non è stato premuto alcun tasto).

INKEY$ è particolarmente utile per realizzare giochi di movimento, oppure per fermare temporaneamente il programma.

Esempio:

```
10 PRINT "Premi un tasto qualsiasi per continuare"
20 A$ = INKEY$
30 IF A$ = "" THEN 20
40 REM Continuazione del programma
```

In questo esempio INKEY$ è adoperata per scandire la tastiera e per far continuare il programma non appena sia stato premuto un tasto qualsiasi.

INPUT istruzione diff.

INPUT serve per introdurre dati in un programma mediante la tastiera. La sua sintassi è così definita:

```
INPUT A,B,....,Z
INPUT "Messaggio";A,B,....,Z
```

dove: A,B,...,Z sono variabili numeriche o stringa

"Messaggio" è una frase che specifica il tipo di dato richiesto.

Quando il sistema esegue un' istruzione di INPUT visualizza, se specificato, il messaggio guida racchiuso tra virgolette più un punto interrogativo ed arresta temporaneamente l' esecuzione del programma in attesa che l' utente inserisca i dati richiesti. I dati che l' utente deve fornire devono essere coerenti in numero e tipo con le variabili dichiarate nell' istruzione.

Nel caso in cui un' istruzione INPUT richieda più di un dato, questi dovranno essere separati dai successivi mediante delle virgole. Premendo il tasto ENTER (oppure i tasti CTRL ed M contemporaneamente) si termina l' inserimento dei dati; il sistema assegna i valori impostati alle variabili specificate e se non sono stati commessi errori fa proseguire l' esecuzione del programma, mentre nel caso contrario viene rieseguita l' istruzione di INPUT.

Se i dati introdotti sono insufficienti, l' interprete basic visualizza un doppio punto interrogativo (??) e richiede il dato mancante, mentre se i dati immessi sono di più di quelli richiesti, viene visualizzato il messaggio *?Extra ignored* (i dati in sovrappiù vengono ignorati) e ripresa l' esecuzione del programma. L' immissione di dati non coerenti con le variabili dichiarate comporta la comparsa del messaggio *?Redo from start* (rifare da capo) e la riesecuzione dell' istruzione.

Esempio:

```
10 REM Programma calcolo area triangolo
20 INPUT "Base, altezza";B,H
30 PRINT "Area triangolo = ";(B*H)/2
40 END
```

Questo semplice programma, dati base ed altezza, calcola l' area del triangolo.

Se l' utente si limita a premere soltanto il tasto ENTER come risposta ad una richiesta di dati, le variabili dichiarate nell' istruzione di INPUT conserveranno gli stessi valori che avevano prima dell' esecuzione dell' istruzione.

L' esecuzione di INPUT in modo immediato provoca un errore di comando immediato illegale (?ID Error). È sempre consigliabile inserire nell' istruzione INPUT il messaggio racchiuso tra virgolette: esso costituirà una preziosa guida, una spiegazione esplicita sul tipo di dato che il programma richiede all' utente. Evitate quindi messaggi eccessivamente concisi o abbreviati; così facendo faciliterete la vita a chi dovrà utilizzare i vostri programmi.

LINE INPUT istruzione diff.

LINE INPUT consente di introdurre da tastiera una stringa in una variabile. La sua sintassi è:

```
LINE INPUT A$
LINE INPUT "Messaggio";A$
```

dove: A$ è una variabile stringa

"Messaggio" è una frase guida che viene visualizzata dal sistema prima dell' introduzione della stringa.

Quando il sistema esegue l' istruzione LINE INPUT visualizza, se specificato, il messaggio guida racchiuso tra virgolette ed arresta temporaneamente l' esecuzione del programma in attesa che l' utente inserisca la stringa richiesta. La pressione del tasto ENTER termina l' inserimento dei dati e fa proseguire l' esecuzione del programma.

Esempio:

```
10 REM Esempio di LINE INPUT
20 LINE INPUT "Stringa da esaminare";A$
30 PRINT "La stringa è lunga ";LEN(A$);" byte"
40 END
```

Se l' utente si limita a premere soltanto il tasto ENTER come risposta all' istruzione LINE INPUT, la variabile stringa dichiarata non modificherà il proprio contentuto.

L' esecuzione di LINE INPUT in modo immediato provoca un errore di comando immediato illegale (?ID Error).

INPUT$ funzione diff.

INPUT$ restituisce dalla tastiera una stringa di caratteri di lunghezza specificata. La sua sintassi è la seguente:

```
INPUT$(n)
```

dove: n è il numero di caratteri da restituire.

Esempio:

```
10 REM Prova di INPUT$
20 A$ = INPUT$(3)
30 PRINT "Il primo carattere è ";LEFT$(A$,1)
40 PRINT "Il secondo carattere è ";MID$(A$,2,1)
50 PRINT "Il terzo carattere è ";RIGHT$(A$,1)
60 END
```

Questo programma legge da tastiera 3 caratteri e li visualizza uno alla volta.

INPUT$ legge n caratteri da tastiera e li restituisce in una stringa di lunghezza n. Qualsiasi carattere generabile da tastiera è accettato come carattere valido. [ENTER] viene codificato come CHR$(13), mentre [BS] come CHR$(8).

I caratteri digitati non vengono visualizzati sul display; per questo motivo INPUT$ è spesso utilizzata per inserire delle parole chiavi in un programma.

Esempio:

```
10 REM Accesso al programma TOPSEC
20 PRINT "Inserire parola chiave"
30 A$ = INPUT$(3)
40 IF A$ = "007" THEN RUN "RAM:TOPSEC" ELSE 20
```

Il programma richiede una parola chiave di 3 lettere. Se la risposta è giusta, verrà caricato ed eseguito il programma TOPSEC, viceversa verrà nuovamente richiesta la parola chiave.

6.2 Output dati.

Il basic dell' M10 possiede delle istruzioni di output molto semplici da usare e molto potenti. PRINT e PRINT USING consentono infatti di visualizzare su video dati in formato standard o definito. LPRINT ed LPRINT USING svolgono la stessa funzione su stampanti parallele, mentre infine LCOPY permette di avere la stampa del testo presente sul video. Vediamo allora come sono definite e come si adoperano queste istruzioni:

PRINT istruzione imm. & diff.

PRINT visualizza i dati sul display. La sua sintassi è così definita:

PRINT
?
PRINT lista
? lista
PRINT §n,lista
? §n,lista

dove: lista è un elenco di espressioni numeriche o stringa, separate da virgole o da punti e virgola

n definisce la posizione iniziale di visualizzazione dei dati.

PRINT, che può anche essere abbreviata con il punto interrogativo (?), visualizza, a partire dalla posizione corrente del cursore video (vedi anche il cap. 4, par. 3), o dalla posizione n se specificata, l' elenco dei dati contenuti nella lista.

L' elenco dei dati può essere composto da espressioni numeriche e/o stringa, separate da virgole o da punti e virgola. È quindi possibile visualizzare costanti, variabili ed espressioni; in quest' ultimo caso verrà visualizzato il risultato del calcolo dell' espressione.

Esempio: *PRINT 4 * 6 + 4*

visualizza il numero 28, che è appunto il risultato dell' espressione 4 * 6 + 4. Analogamente:

? RIGHT$("Olivetti M10",3) + " Computer portatile"

visualizza la frase M10 Computer portatile, risultato dell' espressione stringa indicata.

Il carattere separatore punto e virgola fa sì che i valori dell' espressioni siano visualizzati uno di seguito all' altro, tranne che per i valori numerici, che sono sempre seguiti da un carattere spazio. I numeri negativi sono preceduti dal segno -, mentre i positivi sono preceduti da uno spazio. Il carattere separatore virgola fa visualizzare i dati in forma tabellare, su due colonne distanti 14 caratteri una dall' altra.

Esempio: *PRINT 1, 2, 3, 4*

visualizza i primi quattro numeri nel seguente formato:

```
1               2
3               4
```

Analogamente per le stringhe:

PRINT "Abc","def","ghi","lmn"

visualizza le stringhe nella seguente forma tabellare:

```
Abc             def
ghi             lmn
```

Se la lista contiene come ultimo carattere un separatore (, o ;), la successiva istruzione di stampa avrà luogo a partire dalla prossima posizione di visualizzazione definita dal tipo di separatore (spazio successivo se punto e virgola, colonna successiva se virgola). Viceversa, se l' ultimo carattere della lista non è un separatore, oppure se PRINT non è seguita da alcuna lista, viene eseguita sul video un' interlinea (codice ASCII 10) ed un ritorno a capo (codice ASCII 13).

Esempio: *PRINT 1, 2, 3, :PRINT 4*

visualizza i primi quattro numeri nel seguente formato:

```
1               2
3               4
```

mentre: *PRINT 1, 2, 3 :PRINT 4*

visualizza gli stessi numeri diversamente:

```
1               2
3
4
```

Nel caso del punto e virgola invece:

PRINT "Questo è";" il paese del ";:PRINT "sole"

visualizza la frase "Questo è il paese del sole" su di una sola riga

mentre: *PRINT "Questo è il paese del ":PRINT "del sole"*

visualizza la stessa frase su due righe:

```
Questo è il paese
del sole
```

PRINT §n permette di visualizzare qualsiasi dato nella posizione voluta. Il display a cristalli liquidi dell' M10 è composto di 8 righe e 40 colonne, mentre il video esterno si compone di 25 righe e lo stesso numero di colonne. N rappresenta un numero compreso tra 0 e 319 per il display LCD, e tra 0 e 999 per il video esterno, che sta ad indicare la posizione del primo carattere visualizzato. Il valore 0 corrisponde all' angolo in alto a sinistra, mentre le successive posizioni sono numerate procedendo da sinistra verso destra e dall' alto verso il basso, seguendo questo schema:

```
  0..... 39
 40..... 79
 80.....119
120.....159
160.....199
200.....239
240.....279
280.....319
```

L' angolo destro in basso corrisponde pertanto al numero 319.

Esempio:

```
10 REM Esempio di ? §n
20 CLS:? §97,"Ciao!"
30 ? §139,"a"
40 ? §177,"tutti"
50 END
```

Questo programma visualizza la seguente scritta:

```
Ciao!
a
tutti
```

È possibile adoperare l' istruzione PRINT anche senza caratteri separatori, facendo però bene attenzione ad evitare possibili ambiguità. Ad esempio:

```
A=2 : B$="senza delimitatori"
? "Prova"A"di PRINT"B$
```

visualizza la seguente frase:

Prova 2 di PRINTsenza delimitatori

È comunque preferibile per ragioni di chiarezza separare sempre le diverse epsressioni con i relativi caratteri separatori.

PRINT USING istruzione imm. & diff.

PRINT USING visualizza i dati sul display (a cristalli liquidi o esterno) con un formato definito. La sua sintassi è la seguente:

PRINT USING X$;lista

dove: X$ è un' espressione stringa che definisce il formato dei dati

lista è un elenco di espressioni, numeriche o stringa, che seguono le stesse regole viste per PRINT.

Diversamente dall' istruzione PRINT, anche se esplicitamente indicato dal carattere separatore virgola, non vengono inseriti spazi tra gli elementi da visualizzare, a meno che questi non siano inclusi nel formato adottato.
L' espressione stringa X$ descrive il formato di output desiderato. Sono disponibili ben dieci diversi comandi di formato, due dei quali destinati alle stringhe, i rimanenti ai numeri (reali ed interi):

Comando	descrizione
!	Specifica che deve essere visualizzato solo il primo carattere dell' espressione stringa data.
ç ç	Definisce quanti caratteri della stringa data devono essere visualizzati, il cui numero è pari al numero degli spazi compresi tra i simboli, più due.
#	Specifica quante cifre del numero dato devono essere visualizzate. I numeri con un minor numero di cifre di quanto indicato, saranno preceduti da spazi, cioè incolonnati

a destra. È possibile, mediante il punto decimale, definire anche il numero di cifre da visualizzare dopo il punto. I numeri con più cifre decimali di quanto richiesto, vengono arrotondati, mentre quelli maggiori del formato definito sono visualizzati per intero, preceduti dal simbolo %.

+ Questo simbolo, che può essere posto prima o dopo gli altri comandi di formato, definisce la posizione del segno rispetto al numero (rispettivamente prima o dopo).

- Questo simbolo, posto dopo gli altri comandi di formato, specifica che i numeri negativi siano visualizzati con il segno meno in posizione finale.

** I doppi asterishi, posti prima degli altri comandi di formato, definiscono il riempimento con asterischi degli eventuali spazi iniziali. Questo simbolo specifica anche due ulteriori cifre da visualizzare.

$$ Il doppio dollaro, posto all' inizio della stringa di formato, fa precedere il numero da visualizzare dal simbolo del dollaro. Questo comando aumenta di due posizioni il numero delle cifre del formato, di cui una è occupata dal carattere del dollaro. Il simbolo $$ non è utilizzabile con il formato esponenziale oppure con numeri negativi, a meno che il segno non sia visualizzato come ultimo carattere del numero (comandi + e -).

**$ Questo comando svolge le stesse funzioni dei comandi ** e $$: il numero visualizzato viene preceduto dal simbolo del dollaro ed incolonnato a destra, riempendo gli spazi iniziali con asterischi. **$ aumenta il numero di cifre di tre posizioni, una delle quali è occupata dal simbolo del dollaro.

,	La virgola, posta immediatamente alla sinistra del separatore decimale (.), visualizza i numeri con delle virgole ogni tre cifre, per facilitarne la lettura. Questo comando, che aumenta di uno il numero delle cifre del formato, non ha effetto su numeri in notazione esponenziale.
^^^^	Questo comando, che va posto al termine della stringa di formato, specifica che il numero deve essere visualizzato con notazione esponenziale (E + xx). Le cifre significative sono allineate a sinistra.

Vediamo ora numerosi esempi su questi comandi:

```
PRINT USING "!";"Ciao","a","tutti"
```

visualizza: Cat

```
PRINT USING "ç ç";"Ciao","a","tutti"
```

stampa: Ciaa tut

```
A = 45:B = 262.43:PI = 3.1415:M = -6
PRINT USING "# #.#";A,B,PI,M
```

visualizza: 45.00 %262.43 3.14 -6.00

```
A = 45:B = 262.43:PI = 3.1415:M = -6
PRINT USING "* * # #.# # +";A,B,PI,M
```

produce: * *45.00+ *262.43+ *a*6.00-

```
A = 45:B = 262.43:PI = 3.1415:M = -6
PRINT USING "# #.#^^^^";A,B,PI,M
```

visualizza: 4.5E+01 2.6E+02 3.1E+00-6.0E+00

infine:

```
A = 10000:B = 2450.6:C = -324000
PRINT USING ".";A,B,C
```

10,000.00 2,450.60 %-324,000.00

Se l' espressione stringa di formato X$ non è esatta, ovverossia contiene comandi illeciti, il sistema visualizza X$ come una qualsiasi espressione stringa e immediatamente dopo compare un messaggio di chiamata di funzione irregolare (?FC Error).

LPRINT istruzione imm. & diff.

LPRINT stampa i dati su stampante o microplotter. La sua sintassi è:

```
LPRINT
LPRINT lista
```

dove lista ha lo stesso significato visto nell' istruzione PRINT.

LPRINT svolge le stesse funzioni di PRINT, eccetto che la stampa dei risultati è prodotta su stampante o microplotter. Il dispositivo di stampa deve essere collegato all' M10 tramite interfaccia parallela con l' apposito cavo fornito dalla Olivetti.

Al momento dell' esecuzione di LPRINT, la stampante deve essere in funzione e collegata correttamente al calcolatore: viceversa possono verificarsi errori di input / output (?IO Error), messaggio che compare anche nel caso in cui si interrompe la stampa dei dati premendo [BREAK] o [CTRL] e [C] contemporaneamente.

Per ulteriori ragguagli sul funzionamento di LPRINT consultare l' istruzione PRINT.

LPRINT USING istruzione imm. & diff.

LPRINT USING stampa i dati su stampante o microplotter con un formato definito. La sua sintassi è:

```
LPRINT USING X$,lista
```

dove X$ e lista hanno lo stesso significato visto nell' istruzione PRINT USING.

LPRINT USING svolge le stesse funzioni dell' istruzione PRINT USING, eccetto che i dati vengono prodotti su stampante o microplotter.

Il dispositivo di stampa deve essere collegato al calcolatore tramite interfaccia parallela con l' apposito cavo fornito dalla Olivetti.

Al momento dell' esecuzione di LPRINT USING, la stampante deve essere in funzione e collegata correttamente al calcolatore: viceversa possono verificarsi errori di

input/output (?IO Error), messaggio che compare anche nel caso in cui si interrompe la stampa dei dati premendo [BREAK] o [CTRL] e [C] contemporaneamente.

Per ulteriori ragguagli sul funzionamento di LPRINT USING consultare l' istruzione PRINT USING.

LCOPY istruzione imm. & diff.

LCOPY stampa il testo attualmente presente sul display su stampante o microplotter. La sua sintassi è così definita:

LCOPY

LCOPY produce la stampa del testo presente sul display solo su stampanti o microplotter collegati al computer tramite interfaccia parallela. La stessa funzione di LCOPY può ottenersi premendo il tasto PRINT.

Anche per LCOPY valgono le stesse raccomandazioni e gli stessi suggerimenti visti per LPRINT e LPRINT USING.

6.3 La gestione dei file

Nei due paragrafi precedenti abbiamo esaminato istruzioni e funzioni connesse all' input / output dei dati. In questo stesso ambito, particolare importanza rivestono i comandi per la gestione dei file.
Naturali estensioni del concetto di variabile o di quello di matrice, i file sono delle strutture dati particolarmente indicate per la memorizzazione e l' archiviazione di informazioni di vario genere. I file sono delle sequenze di dati che possono risiedere in memoria RAM, sul nastro di una normale cassetta, o trasferiti ad altre periferiche tramite porta seriale o parallela. Queste sequenze di dati possono essere consultate sia per lettura che per scrittura o infine per aggiungere ulteriori informazioni a quelle già esistenti. Nel primo caso parleremo pertanto di "aprire" il file in lettura (in INPUT), nel secondo in scrittura (in OUTPUT), nel terzo in aggiunta (in APPEND). Aperto il file, possiamo quindi svolgervi una di queste tre possibili operazioni, ad esempio possiamo scrivere, cioè memorizzare una serie di nomi. Fatto ciò, è necessario "richiudere" il file, che rimane a disposizione per altre operazioni di lettura o di aggiunta.
Facendo quindi un paragone con dei sistemi tradizionali di archiviazione, i file sono degli archivi (magnetici se memorizzati su nastro, elettronici se su RAM, ecc..) cui si può accedere previa apertura dell' archivio stesso (file), e che chiuderemo a lavoro terminato. L' archivio può essere consultato solo sequenzialmente: se ad esempio vogliamo

leggere il quarto elemento del file, dovremo prima leggere i tre dati che lo precedono. Il basic dell' M10, considerata la particolare architettura hardware della macchina, consente solamente la creazione e la gestione di file sequenziali, diversamente da altre versioni dello stesso linguaggio che possiedono anche istruzioni per la gestione di file ad accesso casuale, strutture dati in cui è possibile accedere istantaneamente a qualsiasi informazione (record).
Dopo questa breve descrizione sui file, esaminiamo ora le istruzioni connesse alla loro gestione.

OPEN istruzione imm. & diff.

L' istruzione OPEN serve per aprire i file memorizzati in RAM o su cassetta, o per gestire le periferiche come dei file. La sua sintassi è così definita:

```
OPEN "CAS:Nf" FOR mod AS #num
OPEN "RAM:Nf" FOR mod AS #num
OPEN "COM:vbpsx" FOR mod AS #num
OPEN "LCD:" FOR OUTPUT AS #num
OPEN "LPT:" FOR OUTPUT AS #num
OPEN "WAND:" FOR INPUT AS #num
```

dove:

CAS indica che il file è su nastro

RAM indica che il file è in RAM

COM indica che il file è inviato / ricevuto tramite porta seriale

LCD indica che il file è inviato al display dell' M10

LPT indica che il file è inviato alla stampante parallela

WAND indica che il file ricevuto tramite lettore di codici a barre

Nf è il nome con cui è stato salvato il file

num è il numero assegnato al file

mod può essere: INPUT, OUTPUT o APPEND (solo per la RAM).

È indispensabile eseguire un' istruzione OPEN prima di effettuare qualsiasi operazione sul file; in caso contrario viene visualizzato un messaggio di file non aperto (?CF Error).

Num è un numero che identifica univocamente il file; le successive istruzioni di input / output faranno riferimento ad esso proprio tramite questo numero. Num deve essere compreso tra 1 e il valore di MAXFILES, funzione che definisce il massimo numero di file gestibili contemporaneamente.

OPEN "CAS: apre file memorizzati su nastro. Nel caso di apertura del file per input, occorrerà predisporre il registratore a cassette in modo PLAY, dopo previa regolazione del volume di riproduzione. Viceversa per operazioni di output, occorrerà avviare il registratore in registrazione. Non è invece possibile utilizzare la modalità APPEND con il registratore.

OPEN "RAM: apre file residenti in RAM. I file creati in memoria sono del tipo DO, e pertanto possono essere editati con la funzione integrata TEXT.

OPEN "COM:vbpsx abilita la porta seriale alla trasmissione / ricezione di dati (file). È indispensabile specificare i parametri di trasmissione (vbpsx), nonché collegare correttamente la periferica interessata al computer. Per maggiori informazioni riguardo il collegamento di periferiche, consultare l' appendice A.

OPEN "LCD: abilita il display alla ricezione (visualizzazione) dei dati. È chiaramente un dispositivo di solo OUTPUT.

OPEN "LPT: abilita la stampante parallela alla stampa dei dati. È consentita solamente la modalità OUTPUT. Prima dell' esecuzione di istruzioni di scrittura (stampa), verificare che la stampante sia funzionante e collegata al calcolatore (on line).

OPEN "WAND: consente la ricezione di dati letti mediante lettore di codice a barre. Consultare l' appendice A per maggiori ragguagli.

Esempio: *OPEN "RAM:nomi" FOR INPUT AS #1*

apre il file nomi residente in RAM per lettura di dati.

Un file aperto in una modalità non può essere riaperto in un' altra (?AO Error), né possono essere eseguite istruzioni di INPUT in un file aperto in OUTPUT, o viceversa (?CF Error).

CLOSE istruzione imm. & diff.

CLOSE chiude l' accesso ad un file o a una periferica. La sua sintassi è:

```
CLOSE
CLOSE f1,f2,....,fn
```

dove: f1,f2,....,fn sono i numeri con cui sono stati aperti i file.

CLOSE chiude tutti i file attualmente aperti, CLOSE f1,f2,...,fn chiude i file f1,f2,....,fn.

L' esecuzione di END, NEW, RUN, LOAD, o la modifica del programma attualmente in memoria di lavoro comporta la chiusura automatica di tutti i file aperti.

MAXFILES funzione imm. & diff.

La funzione MAXFILES definisce o restituisce il massimo numero di file che possono essere aperti contemporaneamente. La sua sintassi è la seguente:

```
MAXFILES = n
MAXFILES
```

dove: n è un numero intero compreso tra 0 e 15.

MAXFILES = n definisce n come massimo numero di file che possono essere aperti contemporaneamente, MAXFILES restituisce questo valore.

Esempio: *MAXFILES=3*

pone 3 come massimo numero di file aperti contemporaneamente.

Il valore di default di MAXFILES è 0.

EOF funzione imm. & diff.

La funzione EOF identifica il raggiungimento della fine del file. La sua sintassi è:

```
EOF(nf)
```

dove: nf è il numero del file che si vuole esaminare.

EOF va adoperata dopo un' istruzione di OPEN e prima di ogni istruzione di input (INPUT, INPUT$, LINE INPUT) relative al file in questione. La funzione restituisce il valore numerico -1 se è stata raggiunta la fine del file, cioè se non esistono ulteriori caratteri da leggere, altrimenti restituisce il numero 0.

Esempio:

```
10 REM Esempio di uso di EOF
20 OPEN "RAM:DATI" FOR INPUT AS #1
30 IF EOF(1) THEN PRINT "Fine file":CLOSE:END
40 INPUT #1,A$
50 GOTO 30
```

EOF consente di individuare la fine del file dal quale si stanno leggendo dei caratteri o delle parole, evitando così la comparsa di un errore di fine file (?EF Error). Il tentativo di utilizzare la funzione EOF per un file chiuso comporta un errore di file non aperto (?CF Error), mentre adoperare EOF per un file aperto in OUTPUT o in APPEND provoca un errore di nome file errato (?NM Error).

INPUT istruzione diff.

L' istruzione INPUT consente di introdurre dati in un programma da un file. La sua sintassi è:

INPUT #nf,A,B,....,N

dove: nf è il numero con il quale è stato aperto il file

A,B,...,N sono variabili numeriche o stringa.

INPUT assegna gli elementi del file alle variabili specificate, che devono essere dello stesso tipo. È indispensabile che il file in oggetto sia aperto al momento dell' esecuzione di INPUT, viceversa viene visualizzato un errore di file non aperto (?CF Error). Se la modalità di apertura del file è in contrasto con l' istruzione INPUT (file aperto in OUTPUT o in APPEND), compare un errore di errato numero di file (?BN Error).

Spazi, comandi di ritorno a capo ed interlinee sono ignorati se in posizioni iniziali. Qualsiasi carattere che non sia uno di questi appena visti, sarà pertanto il primo carattere del numero o della stringa. Un numero termina con uno spazio, un ritorno a capo, un' interlinea o infine una virgola. Una stringa se iniziata con doppie virgolette si conclude alle successive, viceversa la stringa termina con un ritorno a capo, un' interlinea, una virgola oppure dopo 255 caratteri.

Esempio:

```
10 OPEN "RAM:prova" FOR INPUT AS #1
20 INPUT #1,A,B$
30 PRINT A,B$
40 END
```

Questo semplice programma assegna alle variabili A e B$ i primi due elementi del file prova, e li visualizza.

INPUT può essere adoperata solo in modo differito; qualsiasi tentativo di utilizzarla in modo immediato provoca un errore di comando diretto illegale (?ID Error).

INPUT$ funzione imm. & diff.

INPUT$ restituisce una stringa di lunghezza prefissata da un file. La sua sintassi è:

INPUT$(n,nf)

dove: n è il numero dei caratteri da leggere

nf è il numero con cui è stato aperto il file.

INPUT$ ritorna una stringa di n caratteri prelevati dal file numero nf. Il file può trovarsi in RAM, su cassetta, oppure essere letto tramite linea seriale. Nel caso in cui il file sia memorizzato su nastro, prima dell' esecuzione di INPUT$ il registratore deve essere avviato in riproduzione, assicurandosi che questo sia correttamente collegato all' M10.

Esempio:

```
10 REM Esempio di INPUT$
20 OPEN "RAM:prova" FOR INPUT AS 1
30 IF EOF(1) THEN END
40 A$ = INPUT$(1,1)
50 PRINT A$,ASC(A$)
60 GOTO 30
```

Questo piccolo programma legge un carattere per volta dal file prova e lo stampa assieme al suo codice ASCII.

INPUT$ legge sequenzialmente ogni carattere presente nel file, compresi quindi spazi, tabulazioni, interlinee e ritorni a capo. Il tentativo di leggere più caratteri di quelli disponibili comporta un errore di fine del file (?EF Error).

LINE INPUT istruzione imm. & diff.

LINE INPUT consente di assegnare una stringa prelevata da un file ad una variabile stringa. La sua sintassi è così definita:

LINE INPUT #nf,X$

dove: nf è il numero con il quale è stato aperto il file

X$ è una variabile stringa.

LINE INPUT assegna ad X$ tutti i caratteri del file numero nf fino al successivo a capo (codice ASCII 13) se questo è compreso entro 254 caratteri, altrimenti assegna a X$ i successivi 254 caratteri. Le sequenze ritorno a capo + interlinea (codici 13 e 10) vengono ignorate, cioè non incluse nella variabile X$. Altre eventuali sequenze di codici di controllo sono incluse nelle stringhe assegnate.

Esempio:

```
10 REM Esempio di LINE INPUT
20 OPEN "CAS:DATI" FOR INPUT AS #1
30 IF EOF(1) THEN CLOSE:END
40 LINE INPUT #1,A$
50 PRINT A$
60 GOTO 30
```

LINE INPUT può essere adoperata con file presenti in RAM, su cassetta (come da esempio) o provenienti da linea di comunicazione. Nel caso in cui il file sia memorizzato su nastro, prima dell' esecuzione di LINE INPUT il registratore deve essere avviato in riproduzione, assicurandosi che questo sia correttamente collegato all' M10.

Se il file specificato nell' istruzione LINE INPUT non è aperto, si verifica un errore di file chiuso (?CF Error), mentre se il file è stato aperto in OUTPUT o in APPEND compare un errore di errato numero di file (?BN Error).

PRINT istruzione imm. & diff.

PRINT consente di scrivere dati su un file sequenziale. La sua sintassi è la seguente:

```
PRINT #nf, lista-dati
? #nf, lista-dati
```

dove: nf è il numero con cui è stato aperto il file

lista-dati è una lista di costanti, variabili o espressioni, separate da virgole o punti e virgola.

PRINT permette di scrivere una lista di dati sul file numero nf, che al momento dell' esecuzione deve essere aperto in modalità di OUTPUT o APPEND, altrimenti viene visualizzato un errore di file chiuso (?CF Error).

PRINT scrive i dati nel file prescelto con lo stesso formato adottato per il video; i caratteri separatori virgola e punto e virgola svolgono pertanto la stesse funzioni viste al paragrafo precedente.

Esempio:

```
10 REM Scrittura di un file di prova
20 OPEN "RAM:Prova" FOR OUTPUT AS #1
30 PRINT #1,"Questo è ";"un testo di prova"
40 PRINT #1,"Punto a capo."
50 END
```

Questo programma crea (linea 20) un file denominato *Prova* e scrive le frasi:

```
Questo è un testo di prova
Punto a capo
```

PRINT può essere adoperata con file residenti in RAM, trasmessi da linea di comunicazione o memorizzati su cassetta. In quest' ultimo caso prima della sua esecuzione occorrerà avviare il registratore in registrazione, assicurandosi che questo sia correttamente collegato all' M10

Se il file specificato è stato aperto in INPUT verrà visualizzato un messaggio di errato numero di file (?BN Error).

PRINT USING istruzione imm. & diff.

PRINT USING permette di scrivere dati su file sequenziali con un formato specificato. La sua sintassi è così definita:

```
PRINT #nf, USING X$; lista-dati
? #nf, USING X$; lista-dati
```

dove: nf è il numero con cui il file è stato aperto

lista-dati è una lista di costanti, variabili o espressioni separate da virgole o da punti e virgola

X$ è un' espressione stringa che descrive il formato dei dati

PRINT USING permette di scrivere una lista di dati con il formato prescelto sul file numero nf, che al momento dell' esecuzione deve essere aperto in modalità di OUTPUT o APPEND, altrimenti viene visualizzato un errore di file chiuso (?CF Error).

Il formato di output, definito dall' espressione stringa X$, è uguale a quello visto nel paragrafo precedente per la medesima istruzione.

PRINT USING può essere adoperata con file residenti in RAM, trasmessi da linea di co-

municazione o memorizzati su cassetta. In quest' ultimo caso prima della sua esecuzione occorrerà avviare il registratore in registrazione, assicurandosi che questo sia correttamente collegato al computer.

Se il file specificato è stato aperto in INPUT verrà visualizzato un messaggio di errato numero di file (?BN Error).

6.4 La gestione delle periferiche

L' Olivetti M10 possiede numerose interfacce: quella per il lettore di codice a barre, la parallela Centronics, quella per il registratore a cassette e quella seriale (detta anche RS 232C). Abbiamo già visto nel paragrafo precedente come alcune di queste periferiche possono essere utilizzate per leggere o scrivere dati (istruzioni OPEN, INPUT, ecc...); l' interfaccia seriale, che consente di comunicare sia in ricezione che in trasmissione, può essere gestita dal sistema in maniera analoga ai tasti funzione. È possibile infatti, tramite le istruzioni COM ON / OFF / STOP abilitare, disabilitare o inibire l' interfaccia RS 232C. INP e OUT consentono invece di leggere o scrivere un byte su una "porta" specificata.

COM ON / OFF / STOP istruzione imm. & diff.

COM ON / OFF / STOP consente di abilitare, disabilitare o inibire l' interfaccia seriale RS 232C. La sua sintassi è la seguente:

```
COM ON
COM OFF
COM STOP
```

COM ON abilita la trasmissione e la ricezione dell' interfaccia RS 232C, nonché la possibilità di agganciare la subroutine specificata in un' istruzione ON COM GOSUB (consultare il capitolo successivo). COM OFF disabilita l' aggancio alla subroutine indicata, mentre COM STOP lo inibisce fino al successivo COM ON. Se durante l' inibizione dell' interfaccia arriva qualche carattere, al ripristino della linea di comunicazione verrà immediatamente eseguito un salto alla subroutine.

Esempio:

```
10 REM Esempio
20 OPEN "COM:57E1E" FOR INPUT AS #1
30 COM ON
40 ON COM GOSUB 100


100 REM Subroutine arrivo carattere
110 A$ = INPUT$(1,1)
```

Questo programma evidenzia l' uso dell' istruzione COM ON (linea 30), e dell' istruzione ON COM GOSUB (linea 40). All' arrivo di un carattere qualsiasi, il sistema passerà il controllo alla subroutine della linea 100.

INP funzione imm. & diff.

INP restituisce il byte presente alla porta specificata. La sua sintassi è:

INP(X)

dove: X è un' espressione numerica, il cui valore deve essere un intero compreso tra 0 e 255.

INP(X) legge il byte della porta numero X. Un errore di chiamata illegale di funzione si verifica quando X non rientra nei limiti fissati.

OUT istruzione imm. & diff.

OUT invia un byte alla porta di output indicata. La sua sintassi è così definita:

OUT n,b

dove: n è il numero della porta di output

b è il byte di dati

OUT invia il byte dati b alla porta di output n. B ed n devono essere interi compresi tra 0 e 255, viceversa compaiono messaggi di chiamata irregolare di funzione (?FC Error).

Capitolo 7

Comandi relativi al flusso di controllo

L' interprete basic esegue il programma contenuto nella memoria di lavoro sequenzialmente, a partire dal numero di linea specificato (istruzione RUN, cap. 3, par.5). La successione, il susseguirsi delle linee di programma da eseguire è chiamato più formalmente "flusso di controllo".
Il flusso di controllo può essere alterato da due particolari classi di istruzioni: quelle di salto incondizionato e quelle di salto condizionato. Le prime modificano permanentemente il flusso del programma: appertengono a questa classe GOSUB, RETURN, GOTO e RESUME. Le seconde invece alterano il flusso di controllo solo al verificarsi di certe condizioni; FOR, NEXT, IF THEN ELSE, ON GOSUB e ON GOTO sono le istruzioni che compongono invece questa categoria.

7.1 Istruzioni di salto incondizionato

Il normale flusso del programma che procede in maniera sequenziale può essere alterato da alcune istruzioni, che prendono pertanto il nome di istruzioni di salto. Tra queste abbiamo appena visto che può essere fatta un' ulteriore suddivisione: possiamo infatti dividere le *istruzioni di salto incondizionato* da quelle di *salto condizionato*. Questo paragrafo è interamente dedicato alle prime, mentre le seconde sono trattate nel paragrafo successivo.

Il flusso del programma può essere modificato con l' istruzione GOTO, la cui esecuzione provoca un salto alla riga di programma specificata. È bene non adoperare questa istruzione troppo frequentemente: sebbene risulti talvolta molto utile per eseguire parti del programma non disposte sequenzialmente, e quindi per evitare noiose ripetizioni, l' uso ricorrente di GOTO rende il programma poco "leggibile", cioè poco comprensibile.

Le porzioni di programma che ricorrono frequentemente possono invece essere incluse in *SUBROUTINE*. Le subroutine o sottoprogrammi, sono insiemi di istruzioni, generalmente posti all' inizio o al termine del programma, che possono essere richiamati da punti diversi del programma tramite l' istruzione GOSUB. L' interprete basic quando esegue una GOSUB memorizza il suo indirizzo (il numero di riga) in un' apposita area di memoria chiamata *stack*, e quindi effettua un salto al numero di riga specificato, proseguendo poi da questa l' esecuzione del programma fino a quando non viene eseguita un' istruzione RETURN che provocherà il ritorno all' istruzione successiva la chiamata della subroutine.
Il suddividere il programma principale in più sottoprogrammi prende il nome di programmazione strutturata. Un programma molto complesso potrà quindi essere scritto cominciando a scomporre il programma principale in tanti sottoprogrammi più semplici; questi potranno, se necessario, essere scomposti ancora in altri sottoprogrammi, e così via fino a giungere a sottoprogrammi di facile realizzazione. Vedremo qualche esempio su questo metodo di lavoro, simile a quello delle scatole cinesi, dopo la descrizione di GOSUB.
Le subroutine per il trattamento e il recupero degli errori terminano, diversamente dalle altre, con l' istruzione RESUME, che consente la prosecuzione del programma dal numero di riga specificato.
Esaminiamo quindi più in dettaglio queste quattro istruzioni:

GOTO istruzione imm. & diff.

GOTO consente di eseguire un salto incondizionato alla linea di programma specificata. La sua sintassi è la seguente:

GOTO n

dove: n è un numero di riga.

GOTO provoca un salto alla linea di programma numero n. GOTO può anche essere adoperata in modo immediato per mandare in esecuzione un programma da una deteminata riga di programma. Diversamente da RUN, GOTO non azzera le variabili, né chiude i file aperti.

Esempio:
```
10 REM Tanti bei salti!!!!
20 I=0
30 IF I=10 THEN END
40 PRINT "Salto numero";I
50 I=I+1
60 GOTO 30
```

Questo programma esegue per dieci volte il ciclo formato dalle linee 30-60, visualizzando ad ogni iterazione il messaggio *Salto numero I*, dove I varia da 0 a 9.

Nel caso in cui il numero di riga specificato da n non esista, compare un messaggio di linea non definita (?UL Error).

GOSUB istruzione imm. & diff.

GOSUB consente di chiamare la subroutine specificata e di trasferirle il controllo del programma. La sua sintassi è così definita:

GOSUB n

dove: n è il numero di riga della prima istruzione della subroutine.

GOSUB provoca il salto alla subroutine che comincia alla riga n, da cui prosegue poi l'esecuzione del programma. È possibile "nidificare", cioè inserire subroutine una nell'altra, oppure realizzare subroutine che ne chiamano altre.

Vediamo ora un esempio di impiego dell'istruzione GOSUB. Il programma che segue, date 3 terne di numeri, stampa per ognuna di esse il minimo, il massimo ed il valore medio, nonché la somma finale dei 9 numeri.

```
 10 REM Programma di esempio di GOSUB
 20 INPUT "Prima terna di numeri ";A,B,C
 30 N1=A : N2=B : N3=C
 40 GOSUB 200
 50 INPUT "Seconda terna di numeri ";D,E,F
 60 N1=D : N2=E : N3=F
 70 GOSUB 200
 80 INPUT "Terza terna di numeri ";G,H,I
 90 N1=G : N2=H : N3=I
100 GOSUB 200
110 PRINT "Somma totale ";A+B+C+D+E+F+G+H+I
120 END
 .
200 REM Subroutine calcolo min, max e media
210 IF N1>=N2 THEN IF N1>=N3 THEN MAX=N1 ELSE MAX=N3
220 IF N2>N1 THEN IF N2>=N3 THEN MAX=N2 ELSE MAX=N3
230 IF N1<=N2 THEN IF N1<=N3 THEN MIN=N1 ELSE MIN=N3
240 IF N2<N1 THEN IF N2<=N3 THEN MIN=N2 ELSE MIN=N3
250 PRINT "Massimo ";MAX;" Minimo ";MIN;" Media
";(N1+N2+N3)/3
260 RETURN : REM Fine della subroutine
```

La realizzazione di questo stesso programma senza adoperare l' istruzione GOSUB avrebbe comportato la ripetizione per tre volte del blocco di istruzioni comprese tra la linea 200 e la linea 250.

GOSUB è particolarmente utile non soltanto per evitare lunghe ed inutili ripetizioni di sezioni di programma simili, ma anche per strutturare il programma in maniera più razionale ed efficace. Se il programma da realizzare è complesso, conviene suddividerlo in più sottoprogrammi, ciascuno dei quali risulterà più semplice del programma iniziale. La scomposizione dei sottoprogrammi può continuare se il caso lo richiede, fino a giungere ad un livello di difficoltà minimo. A questo punto la realizzazione dei vari sottoprogrammi comporterà la risoluzione del problema iniziale. Ma vediamo ora un esempio che illustra questo particolare modo di procedere; naturalmente il problema iniziale non sarà eccessivamente complesso, onde evitare di scrivere troppe righe di programma.

Supponiamo che un insegnante voglia scrivere un programma che dati i nominativi di 20 alunni e 3 voti ciascuno, stampi un elenco degli alunni su cui siano riportati i tre voti avuti ed il voto finale (dato come media fra le tre votazioni ricevute), ed un elenco dettagliato degli insufficienti.
Attenendoci ai dettami della programmazione strutturata, possiamo dividere il nostro problema in tre piccoli sottoproblemi: il problema dell' immissione dei dati, quello della stampa del primo elenco, ed infine quello relativo al secondo elenco. Esaminiamo allora in dettaglio questi tre problemi.
Per calcolare e stampare i nominativi ed i voti degli alunni avremo bisogno di usufruire di un vettore (matrice ad una dimensione, consultare anche il cap.5) stringa e di tre vettori numerici. L' immissione dei dati vera e propria potrà essere effettuata tramite un ciclo di FOR.
La stampa del primo elenco comporterà la visualizzazione degli elementi contenuti nei quattro vettori, nonché il calcolo della media.
Il secondo elenco dovrà contenere soltanto gli alunni insufficienti, che andranno selezionati con un apposito IF.

Vediamo quindi in conclusione il programma nel suo insieme:

```
10 REM Programma principale
20 DIM NOMI$(20), V1(20), V2(20), V3(20)
30 REM Chiamata prima subroutine
40 GOSUB 100
50 REM Chiamata seconda subroutine
60 GOSUB 200
70 REM Chiamata terza subroutine
80 GOSUB 300
90 END
```

```
100 REM Subroutine immissione dati
110 FOR I = 1 TO 20
120 INPUT "Nome alunno ";NOMI$(I)
130 INPUT "Primo, secondo e terzo voto";V1(I),V2(I),V3(I)
140 NEXT I
150 RETURN : REM Fine subroutine

200 REM Subroutine stampa elenco generale
210 PRINT "Quadro finale votazioni"
220 FOR I = 1 TO 20
230 PRINT NOMI$(I);V1(I);V2(I);V3(I);" media ";(V1(I) + V2(I) + V3(I))/3
240 NEXT I
250 RETURN : REM Fine subroutine

300 REM Subroutine stampa elenco insufficienti
310 PRINT "Qradro parziale insufficienti"
320 FOR I = 1 TO 20
330 MEDIA = (V1(I) + V2(I) + V3(I))/3
340 IF MEDIA<6 THEN PRINT NOMI$(I);V1(I);V2(I);V3(I);" media
";MEDIA
350 NEXT I
360 RETURN : REM Fine subroutine
```

Se il numero di linea specificato in GOSUB non esiste, compare un messaggio di linea non definita (?UL Error).

È più conveniente porre le subroutine all' inizio del programma, piuttosto che alla fine, specialmente in programmi molto lunghi, poiché l' interprete basic effettua la ricerca del numero di riga della subroutine a partire dalle istruzioni con il più basso numero di linea.

RETURN istruzione diff.

RETURN, posta al termine di una subroutine, ritorna il controllo al programma principale. La sua sintassi è:

```
RETURN
```

Una subroutine può contenere più di un' istruzione RETURN. L' esecuzione di RETURN non preceduta dal relativo GOSUB provoca un errore di RETURN senza GOSUB (?RG Error). RETURN non può naturalmente essere adoperata in modo immediato.

Per eventuali esempi sull' uso di RETURN consultare l' istruzione GOSUB.

RESUME istruzione diff.

RESUME permette la prosecuzione del programma dopo l' esecuzione di una subroutine per la gestione degli errori (ON ERROR GOTO). La sua sintassi è:

```
RESUME
RESUME 0
RESUME NEXT
RESUME n
```

dove: n è un numero di riga.

RESUME e RESUME 0 fanno proseguire l' esecuzione del programma dall' istruzione che ha causato l' errore, RESUME NEXT dalla successiva, RESUME n dalla riga numero n.

Esempio:

```
 10 REM Esempio di RESUME
 20 ON ERROR GOSUB 100
 30 LPRINT "Questa è una prova"
 .
 .
100 REM Routine trattamento degli errori
110 BEEP : PRINT "Errore !!!"
120 RESUME
```

RESUME può essere impiegata solo in apposite routine per la gestione degli errori (dopo ON ERROR GOTO), viceversa viene evidenziato un messaggio di RESUME senza errore (?RW Error).

7.2 Istruzioni di salto condizionato

In questo paragrafo sono descritte alcune istruzioni di salto condizionato, istruzioni che provocano alterazioni al flusso del programma dipendentemente al verificarsi di certe condizioni. FOR e NEXT consentono di realizzare dei cicli iterativi, IF THEN ELSE consente di eseguire o meno le istruzioni che seguono le parole chiavi THEN ed ELSE, ON GOSUB di passare il controllo ad una subroutine, ON GOTO di eseguire un salto alla riga di programma specificata.
Poiché nel prosieguo del paragrafo adopereremo i termini *espressione logica* ed

espressione relazionale, sarà bene chiarirne il significato.
Un' espressione logica può essere:

una costante numerica
una variabile numerica
un' espressione numerica
Es or Es

dove: Es è un espressione numerica o stringa

or è uno dei seguenti operatori relazionali: <, >, =, <=, =<, >=, =>, <>, ><

Esempio: 6, X, X>7, A$="Stringa", B$<>C$ sono tutte espressioni logiche

Un' espressione relazionale può essere:

El
El ol El

dove: El è un' espressione logica

ol è uno dei seguenti operatori logici: NOT, AND, OR, XOR, IMP, EQV, che descrivono i seguenti operatori booleani:

NOT la negazione

AND la congiunzione

OR la disgiunzione

XOR l' OR esclusivo

IMP l' implicazione

EQV l' equivalenza

In figura 7.1 sono riportate le tabelle di verità per queste funzioni.

	A F	B F	A F	B V	A V	B F	A V	B V
NOT A	V		V		F		F	
A AND B		F		F		F		V
A OR B		F		V		V		V
A XOR B		F		V		V		F
A EQV B		V		F		F		V
A IMP B		V		V		F		V

Figura 7.1

Esempio: G$ > H$, A = 5 AND C$ = "ciao", X > 7 XOR V > > Y sono espressioni relazionali.

Definiti operatori ed espressioni logici e relazionali possiamo finalmente esaminare le istruzioni di salto condizionato.

IF .... GOTO / THEN .... ELSE .... istruzioni imm. & diff.

IF ... GOTO ... ELSE e IF ... THEN ... ELSE sono due istruzioni che consentono di eseguire dei salti condizionati. La loro sintassi è così definita:

```
IF Er THEN nr
IF Er THEN i1 : i2: .... : in
IF Er THEN nr ELSE i1 : i2: .... : in
IF Er THEN nr ELSE nr
IF Er THEN i1 : i2: .... : in ELSE nr
IF Er THEN i1 : i2: .... : in ELSE i1 : i2: .... : in
IF Er GOTO nr
IF Er GOTO nr ELSE nr
IF Er GOTO nr ELSE i1 : i2: .... : in
```

dove: Er è un' espressione relazionale

nr è un numero di riga

i1 : i2: ..... : in sono istruzioni basic qualsiasi.

Se l' espressione relazionale Er è vera, viene eseguita l' istruzione GOTO o le istruzioni che seguono la parola chiave THEN, viceversa se Er è falsa vengono eseguite le istruzioni che seguono ELSE.

Esempio:

```
10 REM Esempio di IF
20 INPUT "Primo e secondo numero";A,B
30 IF A>B THEN PRINT "A è maggiore di B" ELSE PRINT "B è maggiore o uguale ad A"
40 END
```

Questo programma, dati A e B, se A è maggiore di B (Er è vera) visualizza la frase *A è maggiore di B*, viceversa la frase *B è maggiore o uguale ad A*; in ambedue i casi, dopo l' esecuzione di una delle due PRINT, il programma termina alla linea 40 per la presenza dell' istruzione END.

Le istruzioni IF possono essere nidificate, sempre però nell' ambito della stessa linea di programma. In questo caso ogni ELSE viene sempre associata al più vicino THEN, rispettando la sintassi sopra descritta.

Esempio:

```
10 REM Esempio di IF nidificato
20 INPUT "Numero da esaminare";N
30 IF N>0 THEN PRINT "Il numero è positivo" ELSE IF N=0 THEN
PRINT "Il numero è zero" ELSE PRINT "Il numero è negativo"
40 END
```

Questo programma mostra un esempio di IF nidificato. La linea 30 è così strutturata:

```
30 IF N>0 THEN PRINT "Il numero è positivo"
              ELSE IF N=0 THEN PRINT "Il numero è zero"
                          ELSE PRINT "Il numero è negativo"
```

FOR ... NEXT istruzioni imm. & diff.

Le istruzioni FOR e NEXT consentono di realizzare dei cicli iterativi. La loro sintassi è:

```
FOR I=E1 TO E2
FOR I=E1 TO E2 STEP E3
NEXT
NEXT v1,v2,...,vn
```

dove: E1, E2, E3 sono tre espressioni numeriche

I è una variabile numerica

v1,v2,...,vn sono variabili di cicli di FOR

Le istruzioni contenute tra FOR ed il relativo NEXT vengono eseguite ciclicamente, finché la variabile I che funge da contatore non raggiunge il valore di E2. Al momento dell' esecuzione dell' istruzione FOR, viene assegnato alla variabile I il valore dell' espressione numerica E1 (valore iniziale). Ad ogni iterazione viene sommato ad I il valore di E3 (incremento), e questo nuovo valore viene confrontato con E2: se Iè maggiore o uguale a E2 il programma proseguirà dall' istruzione successiva a NEXT.

Esempio:

```
10 REM Esempio di FOR...NEXT
20 FOR J=1 TO 10
30 PRINT "Iterazione numero ";J
40 NEXT J
50 END
```

Questo programma illustra l' uso delle istruzioni FOR...NEXT. La linea 30 verrà eseguita 10 volte, dopodiché il programma proseguirà la sua esecuzione alla linea 50, dove si arresterà a causa di END.

Il numero complessivo delle iterazioni è dato da (E2-E1)/E3. Se E3 è omessa, viene assunto come valore di default E3 = 1. È possibile adoperare incrementi negativi, purché il valore iniziale sia maggiore di quello finale (E1 > E2). Anche in caso di valori anomali, ad esempio E1 > E2 con E3 positivo, la sequenza di istruzioni comprese tra FOR e NEXT verrà eseguita una volta.

Esempio:

```
10 REM Esempio di FOR anomalo
20 FOR I=1 TO 0
30 PRINT "Eseguito una volta"
40 NEXT
50 END
```

L' esecuzione di questo programma provocherà la comparsa del messaggio *Eseguito una volta*, anche se il valore iniziale della variabile I è maggiore di quello finale.

È possibile nidificare più sequenze di FOR NEXT una nell' altra, purché ciascuna si concluda con NEXT, e sia completamente contenuta dalla sequenza di FOR precedente; in altre parole non è possibile realizzare sequenze che si intersecano vicendevolmente.

Esempio:

```
10 REM Uso di FOR errato
20 FOR I=1 TO 10
30 FOR J=1 TO 5
40 NEXT I
50 NEXT J
```

Questo programma non è corretto, in quanto i due FOR si intersecano, ovverossia viene chiuso per primo il FOR della linea 20 e poi quello della linea 30. L' esecuzione di questo programma si arresta perciò alla linea 50 per un errore di NEXT senza FOR (?NF Error).

Il motto che dovete ricordare quando operate con dei FOR è che *il primo FOR da chiudere è l' ultimo aperto*.

È possibile adoperare la stessa istruzione NEXT per chiudere più sequenze di FOR; in questo caso NEXT sarà seguita dalle variabili dei FOR da chiudere, ciascuna separata dalle altre con una virgola. Se NEXT non fa riferimento ad alcuna variabile, si assume chiuso l' ultimo FOR aperto. NEXT non preceduta da alcun FOR provoca un errore di NEXT senza FOR (?NF Error).

Qualora i valori delle espressioni lo consentano, è preferibile adoperare come variabili dei FOR variabili intere: ciò rende più veloce la loro esecuzione. Analogamente, quando ciò non pregiudica la comprensibilità del programma, conviene utilizzare NEXT senza specificare la variabile cui fa riferimento.

ON ... GOSUB istruzione imm. & diff.

ON ... GOSUB provoca, al verificarsi di certe condizioni, il salto ad una subroutine. La sua sintassi è la seguente:

```
ON COM GOSUB nr
ON TIME$ = "hh : mm: ss" GOSUB nr
ON KEY GOSUB r1,r2, ...,rn
ON En GOSUB r1,r2, ...,rn
```

dove:

nr è un numero di riga

COM causa un salto alla riga nr se è presente almeno un carattere nel buffer di linea

TIME$ = "hh : mm: ss" specifica l' istante in cui eseguire il salto alla linea nr

KEY determina il salto ad una delle righe elencate, in base al tasto funzione premuto

r1,r2, ...,rn sono numeri di riga

En è un' espressione numerica, il cui valore determina il salto ad una delle righe indicate.

ON COM GOSUB, adoperata insieme alle istruzioni COM ON / OFF / STOP (cap. 6, par. 4), consente, se abilitata da COM ON, di eseguire un salto alla linea specificata se vi sono caratteri nel buffer di linea, ovverossia se sono stati ricevuti dati tramite linea di comunicazione. Dopo l' esecuzione di ON COM GOSUB, il sistema verifica continuamente, cioè dopo ogni istruzione, il contenuto del buffer di linea, provocando un salto alla riga specificata non appena viene ricevuto un carattere. Con COM OFF è possibile disabilitare l' aggancio alla subroutine specificata, mentre COM STOP lo inibisce temporaneamente. Le istruzioni COM ON / OFF / STOP sono descritte al capitolo 6, paragrafo 4.

Esempio:

```
10 REM Esempio di ON COM GOSUB
20 COM ON
30 ON COM GOSUB 100
 .
 .
100 REM Subroutine per RS 232C
110 PRINT "Carattere ricevuto"
120 RETURN
```

Se durante l' esecuzione del programma viene ricevuto un carattere, verrà eseguito un salto alla subroutine della linea 100.

L' istruzione ON TIME$ = "hh : mm : ss" GOSUB, se preceduta da TIME$ ON, provoca un salto alla subroutine specificata quando la stringa TIME$ corrisponde al valore indicato ("hh : mm: ss"). Dopo ON TIME$ GOSUB, se il programma è in esecuzione, il sistema confronta continuamente, cioè istruzione per istruzione, TIME$ con il valore indicato, passando il controllo alla subroutine specificata non appena TIME$ sia uguale a "hh : mm :ss". Con il salto alla subroutine viene anche eseguita una TIME$ STOP, mentre il successivo RETURN provoca una TIME$ ON. Per maggiori informazioni su TIME$ ON / OFF/ STOP consultare il capitolo 3, paragrafo 1.

Esempio:

```
10 REM Esempio di ON TIME$ GOSUB
20 TIME$ ON
30 INPUT "Orario sveglia (hh:mm:ss)";T$
40 ON TIME$ = T$ GOSUB 100
```

```
 50 GOTO 50 : REM ciclo di attesa
100 REM Sveglia !!
110 BEEP : BEEP : BEEP
120 END
```

L' istruzione ON KEY GOSUB, che può essere seguita da massimo otto numeri di riga, provoca un salto alla subroutine specificata quando viene premuto un tasto funzione che deve essere stato abilitato mediante KEY (n) ON. Con il salto alla subroutine viene anche eseguita una KEY (n) STOP, mentre il successivo RETURN provoca una KEY (n) ON. Per maggiori informazioni su KEY (n) ON / OFF/ STOP consultare il capitolo 3, paragrafo 7.

Esempio:

```
 10 REM Esempio di ON KEY (n) GOSUB
 20 KEY (1) ON : KEY (2) ON : KEY (3) ON
 30 ON KEY GOSUB 100,120,140,160
 40 GOTO 40 : REM ciclo di attesa
100 REM Tasto F1
110 PRINT "Tasto funzione F1" : END
120 REM Tasto F2
130 PRINT "Tasto funzione F2" : END
140 REM Tasto F3
150 PRINT "Tasto funzione F3" : END
160 REM Tasto F4
170 END
```

Questo programma mostra l' uso dell' istruzione ON KEY (n) GOSUB. Dopo aver attivato i tasti funzione F1, F2 e F3 (linea 20), viene eseguita successivamente ON KEY GOSUB ed infine posto il programma in attesa (linea 40, che richiama se stessa). La pressione di uno dei primi tre tasti funzione provocherà la comparsa del messaggio *Tasto funzione Fx*, e quindi la terminazione del programma. Va sottolineato che pur essendo presente nell' ON KEY GOSUB un riferimento alla linea 160, la pressione del tasto F4 non provocherà alcun salto alla subroutine indicata poiché F4 non è stato abilitato.

ON En GOSUB consente di eseguire un salto ad una delle subroutine indicate. Il valore dell' espressione numerica En determina a quale delle linee elencate sarà passato il controllo del programma. Se En = 1 verrà eseguito un salto alla prima linea, se En = 2 alla seconda, e così via. Nel caso in cui il valore di En non è intero, ne verrà troncata la parte decimale. Se En = 0 o se En è maggiore del numero di righe elencate ma è minore di 255, l' esecuzione del programma proseguirà dall' istruzione successiva. Se il valore di En non rientra nei limiti cui devono appartenere i numeri interi, il sistema visualizza un errore di chiamata irregolare di funzione (?FC Error).

ON ... GOTO istruzione imm. & diff.

ON ... GOTO permette di eseguire un salto ad un' altra riga di programma. La sua sintassi è:

```
ON ERROR GOTO nr
ON En GOTO r1,r2, ...,rn
```

dove: nr è un numero di riga

En è un' espressione numerica

ERROR abilita la gestione degli errori

r1,r2, ...,rn sono numeri di riga

L' esecuzione di ON ERROR GOTO abilita la gestione degli errori, la cui occorrenza provoca un salto alla prima linea (nr) della subroutine di gestione degli errori, evitando così la comparsa del relativo messaggio e l' arresto del programma. L' errore occorso può essere identificato mediante la funzione ERR, mentre la linea in cui questo si è verificato può essere identificato con ERL (consultare il capitolo 3, paragrafo 6). La gestione degli errori può essere disabilitata con l' istruzione *ON ERROR GOTO 0*, mentre invece il proseguimento dell' esecuzione del programma si ottiene con l' istruzione RESUME (vedi paragrafo precedente).

Esempio:

```
10 REM Esempio di routine per il trattamento dell' errore
20 ON ERROR GOTO 100
 .
 .
100 REM Subroutine gestione errori
110 BEEP : BEEP
120 PRINT "Errore numero ";ERR;" alla linea ";ERL
130 PRINT "Premi ENTER per continuare"
140 A$ = INKEY$(1)
150 IF A$ = CHR$(13) THEN RESUME ELSE 140
```

Questo esempio mostra una tipica routine di gestione degli errori (linee 100 - 150)

ON En GOTO consente di eseguire un salto ad una delle linee di programma indicate. Il valore dell' espressione numerica En determina a quale delle linee elencate sarà passato il controllo del programma. Se En = 1 verrà eseguito un salto alla prima linea, se En = 2 alla seconda, e così via. Nel caso in cui il valore di En non è intero, ne verrà troncata la parte decimale. Se En = 0 o se En è maggiore del numero di righe elencate

ma è minore di 255, l' esecuzione del programma proseguirà dall' istruzione successiva. Se il valore di En non rientra nei limiti cui devono appartenere i numeri interi, il sistema visualizza un errore di chiamata irregolare di funzione (?FC Error).

Esempio:

```
 10 REM Esempio di ON GOTO
 20 INPUT "Numero ";N
 30 ON N GOTO 100,110,120
100 PRINT "Troppo poco !!" : GOTO 20
110 PRINT "Indovinato !" : GOTO 20
120 PRINT "Troppo alto !!" : GOTO 20
```

Capitolo 8

Grafica e suono

Sebbene l' Olivetti M10 non è un computer nato per i giochi o per la musica, possiede egualmente discrete possibilità grafiche e sonore. Con un pò di fantasia potrete quindi realizzare bei programmi pieni di grafica e di musica, che certamente allieterà chi deve adoperare i vostri programmi (magari voi stessi).
Il primo paragrafo è pertanto dedicato agli aspiranti Raffaello, cioè alla grafica, mentre il secondo è invece rivolto agli emuli di Beethoven, ovverossia alla musica, che forse sarebbe meglio definire suono.

8.1 La grafica

Il display dell' M10 possiede una risoluzione grafica di 240 punti (o pixel) in orizzontale e di 64 punti in verticale, per un totale di 15360 pixel. Ogni punto può essere indirizzato singolarmente, visualizzato cioè in positivo (nero su bianco) o in negativo (bianco su nero). Il basic M10 possiede tre potenti istruzioni per gestire la grafica: PSET, PRESET e LINE. PSET e PRESET consentono di visualizzare in positivo o in negativo un singolo punto, mentre LINE permette, dati due punti, di tracciare linee continue o rettangoli. Ma vediamo allora queste istruzioni:

PSET istruzione imm. & diff.

PSET consente di visualizzare in positivo o in negativo un punto sul display. La sua sintassi è:

```
PSET (X,Y)
PSET (X,Y,C)
```

dove: X è un' espressione numerica che indica l' ascissa del punto, che deve essere compresa tra 0 e 239

Y è un' espressione numerica che indica l' ordinata del punto, che deve essere compresa tra 0 e 63

C è un' espressione numerica che specifica il colore con cui deve essere visualizzato il punto.

PSET visualizza il punto di coordinate (X,Y) con il colore C: se C = 0 il punto viene visualizzato in bianco (negativo), viceversa se C = 1 o se C viene omesso, l' elemento viene visualizzato in nero (positivo).

Esempio:

```
10 REM Stelle a volontà
20 CLS
30 X = (RND(1) * 239) + 1
40 Y = (RND(1) * 63) + 1
50 PSET (X,Y,1)
60 GOTO 30
```

Questo programma, dopo aver cancellato il display (linea 20), visualizza punti in maniera casuale, che sembrano tante stelle (con un pò di fantasia) che compaiono una alla volta.

Analogamente a quanto visto per il testo, la posizione (0,0) corrisponde all' angolo superiore sinistro. L' esecuzione di PSET con valori illeciti, comporta la visualizzazione di un errore di chiamata illegale di funzione (?FC Error).

PRESET istruzione imm. & diff.

PRESET consente di visualizzare in positivo o in negativo un punto sul display. La sua sintassi è:

```
PRESET (X,Y)
PRESET (X,Y,C)
```

dove: X è un' espressione numerica che indica l' ascissa del punto, che deve essere compresa tra 0 e 239

Y è un' espressione numerica che indica l' ordinata del punto, che deve essere compresa tra 0 e 63

C è un' espressione numerica che specifica il colore con cui deve essere visualizzato il punto.

PRESET visualizza il punto di coordinate (X,Y) con il colore C: se C = 1 o se C viene omesso, il punto viene visualizzato in bianco (negativo), viceversa se C = 0 l' elemento viene visualizzato in nero (positivo).

Esempio:

```
 10 REM La notte ed il giorno
 20 CLS : PRINT " La notte"
 25 DIM CX(1000), CY(1000)
 30 FOR I = 1 TO 1000
 40 X = (RND(1) * 239) + 1
 50 Y = (RND(1) * 63) + 1
 60 CX(I) = X : CY(I) = Y : PRESET (X,Y,0)
 70 NEXT
 80 PRINT §0,"Il giorno"
 90 FOR I = 1000 TO 1 STEP -1
100 X = CX(I) : Y = CY(I)
110 PRESET (X,Y,1)
120 NEXT
130 END
```

Questo programma, dopo aver cancellato il display (linea 20), visualizza 1000 punti in maniera casuale, che sembrano tante stelle (con un pò di fantasia) che compaiono una alla volta. Le coordinate dei punti sono memorizzate nei vettori CX e CY. Dopo la notte, il giorno: le stelle scompaiono così come sono apparse (linee 90 - 120).

Analogamente a quanto visto per il testo, la posizione (0,0) corrisponde all' angolo superiore sinistro. L' esecuzione di PRESET con valori illeciti, comporta la visualizzazione di un errore di chiamata illegale di funzione (?FC Error).

LINE istruzione imm. & diff.

LINE consente di tracciare linee o rettangoli sul video. La sua sintassi è la seguente:

```
LINE - (X2,Y2)
LINE - (X2,Y2) , C
LINE - (X2,Y2) , C , B
LINE - (X2,Y2) , C , B F
LINE (X1,Y1) - (X2,Y2)
LINE (X1,Y1) - (X2,Y2) , C
LINE (X1,Y1) - (X2,Y2) , C , B
LINE (X1,Y1) - (X2,Y2) , C , B F
```

dove: X1 ed X2 sono due espressioni numeriche che indicano rispettivamente l' ascissa del punto di partenza e di arrivo

Y1 e Y2 sono due espressioni numeriche che indicano rispettivamente l' ordinata del punto di partenza e di arrivo

C è un' espressione numerica che definisce il colore

B (Box che in inglese significa scatola) specifica che verrà tracciato un rettangolo

F (Fill che in inglese vuol dire riempire) specifica il riempimento del rettangolo definito da B.

LINE traccia una linea continua del colore C (C = 1 nero, C = 0 bianco) tra i punti di coordinate (X1,Y1) e (X2,Y2). Se il punto iniziale non è specificato, questo viene assunto uguale all' ultimo punto visualizzato in una precedente LINE, PSET o PRESET, o in assenza di queste a (0,0).

La B posta dopo l' espressione numerica C, che in questo caso è obbligatoria, determina il disegno di un rettangolo con i lati paralleli ai bordi del display e con la diagonale uguale alla linea definita. L' opzione F determina il riempimento (filling) del rettangolo tracciato con il colore C.

Esempio:

```
10 REM Esempio di LINE
20 CLS : Y = 63
30 FOR I = 0 TO 239 STEP 8
40 LINE (I,63-Y) - (239-I,Y) ,1, B
50 Y = Y-2
60 NEXT
70 END
```

Questo programma visualizza una serie di rettangoli concentrici che convergono al centro.

Se C è omesso, viene assunto come valore di default 1. L' esecuzione di LINE con valori non consentiti provoca un errore di chiamata illegale di funzione (?FC Error)

8.2 Il suono

L' Olivetti M10 possiede un generatore di suono programmabile dall' escursione di 5 ottave e mezzo. Il basic del computer possiede due istruzioni per la generazione del suono: BEEP e SOUND. La prima consente di emettere una breve segnalazione acustica, mentre la seconda emette un suono di nota e durata specificate.

BEEP istruzione imm. & diff.

BEEP emette un breve segnale acustico della durata di quasi mezzo secondo. La sua sintassi è così definita:

```
BEEP
```

BEEP è molto utile per segnalare delle particolari situazioni. Ad esempio:

```
10 REM Esempio di BEEP
20 INPUT "Altezza del triangolo ";H
30 IF H < = 0 THEN BEEP : GOTO 20
```

In questo breve programma BEEP è utilizzata per segnalare l' immissione di un dato errato.

SOUND istruzione imm. & diff.

SOUND emette un suono della nota e durata specificate. La sua sintassi è:

```
SOUND ON
SOUND OFF
SOUND n,d
```

dove: n è un' espressione numerica intera che definisce la nota

d è un' espressione numerica intera che specifica la durata della nota

SOUND ON abilita l' altoparlante interno, mentre SOUND OFF lo disabilita. Entrambe possono essere utilizzate per abilitare o disabilitare l' altoparlante durante il caricamento di programmi da cassetta (cap. 3, par. 4).

SOUND n,d emette una nota di lunghezza specificata. N deve essere un intero compreso tra 0 e 16383, mentre d deve essere compreso tra 0 e 255. La lunghezza della nota è data da (d + 1) * 20 ms. N corrisponde alle seguenti note secondo lo schema di figura 8.1.

OTTAVE NOTE	1	2	3	4	5	6
DO		9394	4697	2348	1171	587
DO		8866	4433	2216	1103	554
RE		8368	4184	2092	1043	523
RE	15800	7900	3950	1975	987	493
MI	14912	7456	3728	1864	932	466
FA	14064	7032	3516	1758	879	439
FA	13284	6642	3321	1660	830	415
SOL	12538	6269	3134	1567	783	
SOL	11836	5918	2954	1479	739	
LA	11172	5586	2793	1396	693	
LA	10544	5272	2636	1318	659	
SI	9952	4968	2484	1244	622	

Figura 8.1

Esempio:

```
10 REM Esempio di scala musicale
20 SOUND ON
30 FOR I = 9394 TO 4968 STEP -468
40 SOUND I,50
50 NEXT
60 END
```

Capitolo 9

Funzioni matematiche e di conversione

Insieme alle operazioni matematiche fondamentali, addizione (+), moltiplicazione (*), sottrazione (-) e divisione (/), l' interprete basic dell' M10 possiede una gran varietà di funzioni matematiche e di conversione.
Il primo paragrafo è dedicato esclusivamente alle funzioni trigonometriche. SIN, COS, TAN e ATN sono le funzioni implementate, che calcolano rispettivamente il seno, il coseno, la tangente e l' arcotangente di un angolo espresso in radianti.
Il secondo paragrafo tratta invece delle restanti funzioni matematiche: ABS, che fornisce il valore assoluto di un numero, EXP che calcola l' esponenziale, LOG il logaritmo, SQR la radice quadrata, MOD il modulo (resto di una divisione), SGN che fornisce il segno di un numero ed infine RND che genera dei numeri casuali.
Il terzo paragrafo è invece dedicato alle funzioni di conversione, quelle funzioni cioè che trasformano un numero reale in intero, o viceversa, un numero in singola precisione in doppia, ecc...

9.1 Funzioni trigonometriche

SIN, COS, TAN ed ATN sono le quattro funzioni trigonometriche implementate sull' M10, che calcolano rispettivamente il seno, il coseno, la tangente e l' arcotangente di un angolo espresso in radianti. Per chi non ha molta dimestichezza con la trigonometria, diremo che un radiante equivale a 180 gradi trecentosessantesimali; pertanto la conversione tra gradi e radianti è banalmente eseguibile come indicato:

```
ANGOLO IN RADIANTI = 3.1415 * ANGOLO IN GRADI / 180
```

mentre la conversione inversa è data da:

ANGOLO IN GRADI = ANGOLO IN RADIANTI * 180 / 3.1415

Cominciamo dunque ad esaminare queste funzioni:

SIN funzione imm. & diff.

La funzione SIN calcola il seno di un angolo. La sua sintassi è così definita:

SIN (X)

dove: X è un' espressione numerica indicante l' ampiezza dell' angolo in radianti.

Il risultato fornito da SIN è in doppia precisione. Il valore di X può anche essere negativo.

Il programma che segue disegna il grafico della funzione SIN, nell' intervallo tra 0 e 720 gradi.

```
 10 REM Grafico SIN (X)
 20 CLS : PRINT §10,"Grafico di SIN (X)"
 30 XI = 0 : XF = 12.566 : REM XI è il valore iniziale, XF quello finale
 40 XL = (XF-XI) / 240 : XP = 240-XF / XL
 50 LINE (XP,1) - (XP,62) : REM asse Y
 55 YI = -1 : YF = 1 : REM YI è il minimo, YF il massimo
 60 YL = (YF-YI) / 64 : YP = 63 + YI / YL
 70 LINE (0,YP) - (239,YP) : REM asse X
 80PSET (0,0)
 90REM Grafico funzione
100 FOR I = 0 TO 239
110 X = XF / 239 * I + XI
120 YP = 64-(SIN (X) - YI) / YL
130 LINE - (I,YP)
140 NEXT
150 END
```

COS funzione imm. & diff.

COS calcola il coseno di un angolo espresso in radianti. La sua sintassi è:

COS (X)

dove: X è un' espressione numerica indicante l' ampiezza dell' angolo in radianti.

Il risultato fornito da COS è in doppia precisione. Il valore di X può anche essere negativo.

Il programma che segue disegna il grafico della funzione COS, nell' intervallo tra 0 e 720 gradi.

```
 10 REM Grafico COS (X)
 20 CLS : PRINT §10,"Grafico di COS (X)"
 30 XI = 0 : XF = 12.566 : REM XI è il valore iniziale, XF quello finale
 40 XL = (XF-XI) / 240 : XP = 240-XF / XL
 50 LINE (XP,1) - (XP,62) : REM asse Y
 55 YI = -1 : YF = 1 : REM YI è il minimo, YF il massimo
 60 YL = (YF-YI) / 64 : YP = 63 + YI / YL
 70 LINE (0,YP) - (239,YP) : REM asse X
 80PSET (0,0)
 90REM Grafico funzione
100 FOR I = 0 TO 239
110 X = XF / 239 * I + XI
120 YP = 64-(COS (X) - YI) / YL
130 LINE - (I,YP)
140 NEXT
150 END
```

TAN funzione imm. & diff.

TAN calcola la tangente di un angolo espresso in radianti. La sua sintassi è la seguente:

TAN (X)

dove: X è un' espressione numerica indicante l' ampiezza dell' angolo in radianti.

Il risultato fornito da TAN è in doppia precisione. Il valore di X può anche essere negativo.

Il programma che segue disegna il grafico della funzione TAN, nell' intervallo tra 0 e 85 gradi.

```
 10 REM Grafico TAN (X)
 20 CLS : PRINT §10,"Grafico di TAN (X)"
 30 XI = 0 : XF = 1.5 : REM XI è il valore iniziale, XF quello finale
 40 XL = (XF-XI) / 240 : XP = 240-XF / XL
 50 LINE (XP,1) - (XP,62) : REM asse Y
 55 YI = 0 : YF = 14.2 : REM YI è il minimo, YF il massimo
 60 YL = (YF-YI) / 64 : YP = 63 + YI / YL
 70 LINE (0,YP) - (239,YP) : REM asse X
 80PSET (0,0)
 90REM Grafico funzione
100 FOR I = 0 TO 239
110 X = XF / 239 * I + XI
120 YP = 64-(TAN (X) - YI) / YL
130 LINE - (I,YP)
140 NEXT
150 END
```

ATN funzione imm. & diff.

ATN calcola l' arcotangente di un numero. La sua sintassi è:

ATN (X)

dove: X è un' espressione numerica qualsiasi.

Il risultato è fornito in radianti, nell' intervallo - PIGRECO / 2, + PIGRECO / 2. ATN è la funzione inversa di TAN.

Funzioni derivate

Utilizzando le funzioni trigonometriche di base, è possibile derivare le seguenti funzioni:

Secante: SEC (X) = 1 / COS (X)

Cosecante: CSC (X) = 1 / SIN (X)

Seno inverso: ARCSIN (X) = ATN (X / SQR (-X * X + 1))

Coseno inverso: ARCCOS (X) = - ATN (X / SQR (-X * X + 1)) + 1.5708

Secante inversa: ARCSEC (X) = ATN (SQR (X * X - 1)) + (SGN (X)-1) * 1.5708

Cosecante inversa: ARCCOS(X) = ATN (1 / SQR (X * X - 1)) + (SGN (X) - 1) * 1.5708

Cotangente inversa: ARCOT (X) = - ATN (X) + 1.5708

Seno iperbolico: SINH (X) = (EXP (X) - EXP (-X)) / 2

Coseno iperbolico: COSH (X) = (EXP (X) + EXP (-X)) / 2

Tangente iperbolica: TANH (X) = - EXP (-X) / (EXP (X) + EXP (-X)) * 2 + 1

Secante iperbolica: SECH (X) = 2 / (EXP (X) + EXP (-X))

Cosecante iperbolica: CSCH (X) = 2 / (EXP (X) - EXP (-X))

Cotangente iperbolica: COTH (X) = EXP (-X) / (EXP (X) - EXP (-X)) * 2 + 1

Seno iperbolico inverso:ARCSINH (X) = LOG (X + SQR (X * X + 1))

Coseno iperbolico inverso: ARCCOSH (X) = LOG (X + SQR (X * X - 1))

Tangente iperbolica inversa: ARCTANH (X) = LOG ((1 + X) / (1 - X)) / 2

Secante iperbolica inversa: ARCSECH (X) = LOG ((SQR (-X * X + 1) + 1)/ X

Cosecante iperbolica inversa: ARCCSCH (X) = LOG (SGN (X) * SQR (X * X + 1) + 1) / X

Cotangente iperbolica inversa: ARCCOTH (X) = LOG ((X + 1) / (X - 1)) / 2

9.2 Funzioni matematiche

Dopo le funzioni trigonometriche, passiamo alle rimanenti funzioni matematiche. ABS e SGN forniscono rispettivamente il valore assoluto ed il segno di un numero, EXP, LOG e SQR calcolano l' esponenziale, il logaritmo e la radice quadrata, mentre MOD restituisce il modulo (resto di una divisione) e RND genera sequenze di numeri casuali.

ABS funzione imm. & diff.

ABS calcola il valore assoluto di un' espressione numerica. La sua sintassi è così definita:

ABS (X)

dove: X è un' espressione numerica qualsiasi

Ad esempio: PRINT ABS (-7)

visualizza *7.*

SGN funzione imm. & diff.

SGN restituisce il segno di un' espressione numerica. La sua sintassi è la seguente:

SGN (X)

dove: X è un' espressione numerica qualsiasi.

SGN (X) restituisce 1 se il valore di X è maggiore di zero, 0 se X è uguale a zero, -1 se X è minore di zero.

Esempio:

```
10 REM Esempio di SGN
20 INPUT "Numero ";N
30 IF SGN (N) < 0 THEN PRINT "N è negativo" ELSE IF SGN (N) >
0 THEN PRINT "N è positivo" ELSE PRINT "N è nullo"
40 GOTO 20
```

Questo programma, dato un numero, descrive se il numero è nullo, minore o maggiore di zero.

EXP funzione imm. & diff.

EXP calcola il valore di e^X. La sua sintassi è così definita:

```
EXP (X)
```

dove: X è un' espressione numerica, il cui valore deve essere compreso tra -149.664 e 145.0628.

Se il valore dell' espressione X è fuori dai limiti indicati, occorre un errore di supero di capacità (?OV Error).

Il programma che segue disegna il grafico della funzione EXP, nell' intervallo compreso tra 0 e 3.

```
 10 REM Grafico EXP (X)
 20 CLS : PRINT "Grafico di EXP (X)"
 30 XI = 0 : XF = 3 : REM XI è il valore iniziale, XF quello finale
 40 XL = (XF-XI) / 240 : XP = 240-XF / XL
 50 LINE (XP,1) - (XP,62) : REM asse Y
 55 YI = 0 : YF = 20.2 : REM YI è il minimo, YF il massimo
 60 YL = (YF-YI) / 64 : YP = 63 + YI / YL
 70 LINE (0,YP) - (239,YP) : REM asse X
 80PSET (0,0)
 90REM Grafico funzione
100 FOR I = 0 TO 239
110 X = XF / 239 * I + XI
120 YP = 64-(EXP (X) - YI) / YL
130 LINE - (I,YP)
140 NEXT
150 END
```

LOG funzione imm. & diff.

LOG calcola il logaritmo naturale di un' espressione numerica. La sua sintassi è:

```
LOG (X)
```

dove: X è un' espressione numerica.

Il valore dell' espressione X deve essere positivo (la funzione logaritmo non è definita per valori di X < = 0), altrimenti viene visualizzato un errore di chiamata illegale di funzione (?FC Error)

Il programma che segue disegna il grafico della funzione LOG, nell' intervallo compreso tra 1 e 100.

```
 10 REM Grafico LOG (X)
 20 CLS : PRINT §10,"Grafico di LOG (X)"
 30 XI = 1 : XF = 100 : REM XI è il valore iniziale, XF quello finale
 40 XL = (XF-XI) / 240 : XP = 240-XF / XL
 50 LINE (0,1) - (0,62) : REM asse Y
 55 YI = 0 : YF = 4.65 : REM YI è il minimo, YF il massimo
 60 YL = (YF-YI) / 64 : YP = 63 + YI / YL
 70 LINE (0,YP) - (239,YP) : REM asse X
 80PSET (0,0)
 90REM Grafico funzione
100 FOR I = 0 TO 239
110 X = XF / 239 * I + XI
120 YP = 64-(LOG (X) - YI) / YL
130 LINE - (I,YP)
140 NEXT
150 END
```

SQR funzione imm. & diff.

SQR calcola la radice quadrata di un' espressione numerica. La sua sintassi è la seguente:

```
SQR (X)
```

dove: X è un' espressione numerica.

Il valore dell' espressione numerica X deve essere positivo o nullo (la funzione radice quadrata non è definita per valori di X < 0), altrimenti viene visualizzato un messaggio di chiamata illegale di funzione. (?FC Error).

Il programma che segue disegna il grafico della funzione SQR, nell' intervallo compreso tra 0 e 100.

```
 10 REM Grafico SQR (X)
 20 CLS : PRINT §10,"Grafico di SQR (X)"
 30 XI = 0 : XF = 100 : REM XI è il valore iniziale, XF quello finale
 40 XL = (XF-XI) / 240 : XP = 240-XF / XL
 50 LINE (XP,1) - (XP,62) : REM asse Y
 55 YI = 0 : YF = 10.1 : REM YI è il minimo, YF il massimo
```

```
 60 YL = (YF-YI) / 64 : YP = 63+YI / YL
 70 LINE (0,YP) - (239,YP) : REM asse X
 80PSET (0,0)
 90REM Grafico funzione
100 FOR I=0 TO 239
110 X = XF / 239 * I + XI
120 YP=64-(SQR (X) - YI) / YL
130 LINE - (I,YP)
140 NEXT
150 END
```

MOD funzione imm. & diff.

MOD ritorna il resto di una divisione tra espressioni intere. La sua sintassi è:

N MOD M

dove: N è un' espressione numerica intera che definisce il dividendo

M è un' espressione numerica intera che definisce il divisore.

Se i valori di N o M sono espressioni numeri reali, questi vengono trasformati in interi.

Esempio: PRINT 10.6 MOD 3

restituisce 1 come risultato.

RND funzione imm. & diff.

RND genera una sequenza di numeri pseudo-casuali compresi tra 0 ed 1. La sua sintassi è la seguente:

RND (X)

dove: X è un' espressione numerica qualsiasi.

Se il valore di X è maggiore di zero, viene sempre generata la stessa sequenza di numeri pseudo-casuali ad ogni esecuzione del programma, viceversa se X è minore o uguale a zero viene ripetuto l' ultimo numero generato.

La funzione RND è molto utile per la realizzazione di giochi, o più in generale di quei programmi che necessitano di eventi casuali.

```
10 REM Partita a dadi
20 INPUT "Premi ENTER per lanciare i dadi";A
25 R1 = RND (1) * 6 + 1 : R2 = RND (1) * 6 + 1
30 PRINT "Hai fatto ";INT (R1);" e ";INT (R2)
40 PRINT "Adesso tocca a me"
50 C1 = RND (1) * 6 + 1 : C2 = RND (1) * 6 + 1
60 PRINT "Io ho fatto ";INT (C1);" e ";INT (C2)
70 IF INT (C1) + INT (C2) > INT (R1) + INT (R2) THEN PRINT "Ho
vinto io!!!!!" ELSE PRINT "Hai vinto tu, peccato !"
80 GOTO 20
```

Questo semplice programma vi consente di giocare ai dadi contro il computer. Buona fortuna!

Per avere sequenze di numeri casuali compresi in un determinato intervallo, occorre adoperare la seguente formula:

RND (1) * Es + Ei

dove: Ei è un' espressione numerica che specifica l' estremo inferiore dell' intervallo

Es è un' espressione numerica che specifica l' estremo superiore dell' intervallo.

Ad esempio nel programma dei dadi visto prima, poiché occorrevano numeri compresi tra 1 e 6, si è impiegata la funzione RND come mostrato alla linee 25 e 50.

9.3 Funzioni di conversione

L' interprete basic dell' M10 gestisce tre tipi di dati numerici: gli interi, i reali in singola ed in doppia precisione (consultare il cap. 2). CDBL converte un numero da singola a doppia precisione, mentre CSNG esegue l' operazione inversa. La conversione da numero reale ad intero può essere eseguita con tre funzioni specifiche: CINT ,FIX ed INT.

CDBL funzione imm. & diff.

CDBL converte un numero, reale o intero, in doppia precisione. La sua sintassi è la seguente:

CDBL (X)

dove: X è un' espressione numerica intera o reale.

CDBL è la funzione inversa di CSNG. Maggiori informazioni su come vengono effettuate le conversioni sono contenute nel capitolo 2.

CSNG funzione imm. & diff.

CSNG converte un numero, reale o intero, in singola precisione. La sua sintassi è la seguente:

CSNG (X)

dove: X è un' espressione numerica intera o reale (doppia precisione).

CSNG è la funzione inversa di CDBL. Maggiori informazioni su come vengono effettuate le conversioni sono contenute nel capitolo 2.

CINT funzione imm. & diff.

CINT converte un numero reale in intero, tralasciando tutte le cifre decimali. La sua sintassi è:

CINT (X)

dove: X è un' espressione numerica reale in singola o doppia precisione.

Il valore dell' espressione X viene convertito in intero eliminando tutte le cifre decimali (troncamento matematico). Se il risultato ottenuto non rientra nell' intervallo -32768 + 32767, viene visualizzato un errore di supero di capacità (?OV Error). Maggiori informazioni su come vengono effettuate le conversioni sono contenute nel capitolo 2.

FIX funzione imm. & diff.

FIX tronca un numero reale, in singola o doppia precisione, in un intero. La sua sintassi è così definita:

FIX (X)

dove: X è un' espressione numerica qualsiasi.

Il valore dell' espressione X viene convertito in intero eliminando tutte le cifre decimali (troncamento matematico). Maggiori informazioni su come vengono effettuate le conversioni sono contenute nel capitolo 2.

INT funzione imm. & diff.

INT restituisce la parte intera di un' espressione numerica. La sua sintassi è:

```
INT (X)
```

dove: X è un' espressione numerica qualsiasi.

Se il valore dell' espressione X è positivo, INT restituisce la parte intera del numero (tralasciando la parte decimale). Ad esempio:

```
PRINT INT (8.5)
```

visualizza il numero *8.*

Viceversa se X è negativo, INT restituisce la parte intera di X + 1.

```
PRINT INT (-8.5)
```

restituisce *-9.*

PARTE SECONDA
GLI APPLICATIVI

Capitolo 10

Il concetto di integrazione

L' Olivetti M10, a differenza degli altri personal computer, si prefigura come sistema programmabile, grazie all' interprete basic diffusamente esaminato nella prima parte del testo, e programmato. È proprio questo l' aspetto nuovo di questo calcolatore che possiede, implementati nella R.O.M. (read only memory - memoria di sola lettura), quattro programmi applicativi appositamente realizzati ed inseriti nell' architettura hardware e software del computer. L' M10 diventa così, basic a parte, una sofisticata macchina per l' uomo moderno che desidera adoperarla al pari di una macchina da scrivere, di una agenda, di un indirizzario o di un terminale collegato ad un sistema informatico. La differenza tra l' M10 e gli altri calcolatori è proprio questa: gli applicativi contenuti nel primo sono perennemente ed istantaneamente disponibili all' utente, in maniera del tutto trasparente. Ma non solo. L' implementazione delle quattro funzioni appena descritte nella ROM del calcolatore ha consentito di realizzare programmi applicativi che utilizzano le stesse strutture dati, con tutti gli immaginabili vantaggi che la cosa comporta. Il software di base del computer viene così ad essere un tutt' uno, un nucleo multifunzionale che interagisce con l' utente.

10.1 Gli applicativi

In questo paragrafo daremo una sommaria descrizione degli applicativi di base, soffermandoci sui loro aspetti più salienti.

Lo scrivere rimane una delle attività più importanti dell' uomo anche in un' epoca come questa in cui i mezzi audiovisivi hanno una diffusione sempre maggiore. Lettere, annotazioni, documenti, sono e resteranno per un lungo periodo mezzi insostituibili di co-

municazione tra gli uomini, sia per lavoro che per diletto. Il programma applicativo TEXT è stato realizzato proprio per soddisfare queste esigenze nel modo più funzionale possibile. Con esso è possibile infatti scrivere, modificare, memorizzare, impaginare e stampare un testo di qualsiasi tipo, la cui stesura può essere facilitata da una serie di funzioni particolari che consentono la ricerca, la cancellazione, l' inserimento, la duplicazione e lo spostamento di interi blocchi di testo. I testi così redatti possono poi essere archiviati su nastro magnetico, oppure trasferiti ad un altro calcolatore.

La necessità di possedere un archivio elettronico è un esigenza che viene sentita da un numero sempre maggiore di persone,indipendentemente dalla professione svolta. L' archivio occorrente può variare da un semplice elenco di numeri telefonici ed indirizzi, ad un più completo archivio di informazioni su oggetti, persone o società. L' M10, coerentemente con la sua filosofia, dispone di un programma applicativo denominato ADDRSS che consente la gestione di un archivio elettronico, la cui struttura può essere definita dall' utente in base alle sue esigenze. Sull' archivio potranno così essere eseguite ricerche parziali o totali delle informazioni memorizzate.

Chiunque abbia la necessità di ricordare appuntamenti, commissioni, ed informazioni di qualsiasi altro genere, ed abbia la memoria corta, si serve di un' agenda. L' Olivetti M10 possiede un programma applicativo, SCHEDL, realizzato appositamente per gestire un' agenda elettronica. Anche in questo caso, come per il programma ADDRSS, è possibile effettuare ricerche parziali o totali sulle informazioni memorizzate, oppure produrre tabulati sulla base di alcune specifiche.

Il crescente sviluppo della telematica e la sempre maggiore disponibilità di centri servizi, nonché la diffusione ormai raggiunta dai personal computer, evidenziano la necessità di disporre di interfacce e programmi destinati alla telecomunicazione. La porta seriale di comunicazione ed il programma TELCOM sono i potenti mezzi hardware e software che l' M10 mette a disposizione dell' utente. Il programma TELCOM consente di utilizzare il computer come un terminale remoto collegato ad un altro calcolatore tramite linea telefonica o cavo diretto. La possibilità di poter variare i parametri di trasmissione, o di poter inviare e ricevere file dati, conferisce a TELCOM notevole flessibilità d' uso. In altre parole, grazie a questo programma l' M10 può collegarsi a centri di calcolo, banche dati, o anche più semplicemente al computer dell' amico.

I programmi TEXT, TELCOM, ADDRSS e SCHEDL sono ampiamente descritti rispettivamente nei capitoli 11, 12, 13 e 14.

10.2 La struttura dati

Tutti e quattro i programmi applicativi descritti adoperano strutture dati costituite da

file di tipo testo (.DO). Questa caratteristica conferisce particolare importanza al programma TEXT, che consente la creazione, la modifica e la memorizzazione dei file di testo. Infatti con esso è possibile generare ed aggiornare gli archivi ADRS.DO e NOTE.DO con i quali operano i programmi ADDRSS e SCHEDL, oppure preparare i testi da inviare con TELCOM o ancora visualizzare e stampare quelli ricevuti dallo stesso programma.

I vantaggi di adoperare una stessa struttura dati sono evidenti: a parte l' interscambiabilità dei file, c' è anche da sottolineare la possibilità di una loro manipolazione tramite linguaggio basic. Quest' ultimo viene così a completare la già ampia flessibilità dell' M10.

In questo paragrafo, piuttosto che fornire listati di programmi che gestiscono gli archivi di SCHEDL e ADDRSS, oppure che eseguono calcoli su testi redatti con TEXT, daremo alcuni suggerimenti sulle possibilità offerte dal software di base.

L' estrema facilità con la quale possono essere redatti i testi mediante il programma TEXT, consente di memorizzare agevolmente informazioni di qualsiasi tipo con un formato ben specificato. Fatto ciò, non è difficile realizzare dei semplici programmi basic che gestiscano queste informazioni. Ad esempio è possibile memorizzare delle fatture con TEXT e scrivere un programma che svolga alcune funzioni contabili, oppure gestire con lo stesso metodo i rimborsi spese, il conto corrente bancario, ecc...

Per sfruttare appieno le caratteristiche di integrazione del sistema, è consigliabile definire uno o più formati standard con i quali realizzare i file di testo. Così ad esempio nel realizzare gli archivi di ADDRSS e SCHEDL, si può decidere di separare i dati mediante un a capo e di dividere le varie informazioni di un singolo dato con due punti (:). Così facendo sarà possibile scrivere dei programmi standard di calcolo dei dati contenuti in questi file, o ancora programmi statistici, contabili, scientifici ed altri ancora.

In definitiva le possibilità offerte dall' insieme BASIC - TEXT, SCHEDL, ADDRSS e NOTE sono vastissime: è solo questione di fantasia.

Nei capitoli successivi, insieme ad un' esauriente spiegazione sui programmi di base, forniremo alcune idee guida su diverse applicazioni.

Capitolo 11

Il programma applicativo TEXT

Lo scrivere è una delle attività più frequenti dell' uomo; c' è chi scrive per lavoro lettere, annotazioni, rapporti, chi invece scrive manoscritti e articoli, chi ancora scrive per passatempo racconti, poesie o memorie. A parte la penna, il mezzo di scrittura per eccelenza è stato finora la macchina da scrivere, diventata con il tempo prima elettrica e poi elettronica. Chiunque abbia avuto a che fare con essa, ne avrà certamente apprezzato la qualità di stampa e la facilità d' uso. Ma purtroppo "Errare humanum est", anche per un provetto dattilografo. E allora? Correzioni fatte a penna, oppure con qualche cancellatore, che inevitabilimente rovinano l' estetica del testo. Se poi occorre modificare qualche periodo, o semplicemente spostarlo, occorre armarsi di tanta pazienza e ricominciare da capo.
Immaginate invece di avere un sistema elettronico di scrittura che vi consenta di scrivere, memorizzare, impaginare e stampare i vostri testi, che risulteranno così a prova d errore. E poiché si scrive non solo a casa o in ufficio, ma anche in treno, in aereo o sulla spiaggia, immaginate anche che il sistema sia veramente portatile: ecco a voi il binomio M10 e TEXT, il programma applicativo integrato per la gestione dei testi.

11.1 Il funzionamento di TEXT

Il programma applicativo TEXT può essere richiamato dal menù principale semplicemente posizionando il cursore sul suo nome e premendo il tasto ENTER, oppure digitando TEXT [ENTER]. Immediatamente dopo l'esecuzione di uno dei due comandi si cancella il video ed appare in alto a sinistra il messaggio *File to edit?*, in risposta del quale l' utente deve introdurre il nome del file che desidera creare o rivedere. Se il file specificato non è contenuto nella memoria RAM, TEXT provvederà a creare il file con il

nome indicato, mentre nel caso contrario richiamerà il testo memorizzato e lo mostrerà sul display.

La scelta del nome

Nella scelta del nome del file occorre tenere presente che:

- il nome deve essere lungo al massimo 6 caratteri

- come primo carattere del nome non possono essere utilizzati:

  i numeri, i simboli !, '', #, $, %, &, ', *, (,), +, :, la virgola (,), . , /

- il nome del file non deve contenere i caratteri due punti (:) e punto (.), salvo se questo è adoperato per specificare il suffisso del file (.do)

Nel caso in cui il nome introdotto viola una delle regole appena descritte, TEXT rifiuta il nome emettendo un breve segnale acustico e ripete la richiesta *File to edit?* alla riga successiva. È importante notare che il nome introdotto viene convertito dal sistema in maiuscolo: pertanto i nomi libro e LIBRO identificano lo stesso file. Quando non specificato dall' utente, il programma aggiunge il suffisso .DO al nome introdotto. Così ad esempio:

introd diventa INTROD.DO, così come
InTrod diventa INTROD.DO, e
INTROD.DO rimane INTROD.DO

mentre invece vengono rifiutati i seguenti nomi:

cap.1, P2:uno, CAPITOLO, 4cani, ''testo'', e così via.

La scrittura di un testo

Introdotto un nome valido alla richiesta apparsa sul video, il programma TEXT creerà, se il file non è presente in RAM, un file di testo con il nome assegnatogli. Subito dopo l' inserimento del nome viene cancellato il display ed appare nell' angolo superiore sinistro il cursore video contenente una freccia rivolta a sinistra, che indica la fine del testo. Prima di proseguire nelle spiegazioni sui comandi e le funzioni del programma, sarà bene chiarire come TEXT gestisce il testo redatto dall' utente.

Il testo può avere qualsiasi lunghezza, nei limiti della memoria disponibile. Poiché il

display dell' M10 contiene al più 320 caratteri (8 righe per 40 colonne), mentre generalmente un testo risulterà più lungo, TEXT considera il video come una "finestra" sul testo, finestra che può essere spostata a piacimento (scrolling). L' utente può pertanto modificare soltanto la parte del testo mostrata sul display, dal momento che i comandi e le funzioni di TEXT agiscono solo su esso. Alcuni comandi spostano automaticamente la "finestra", come ad esempio la funzione di ricerca (spiegata in dettaglio più avanti) che cerca una stringa nel testo: se la ricerca ha esisto positivo, il display mostra la parte del testo dove è stata trovata la stringa. Analogamente quando l' utente scrive, la finestra si sposta automaticamente sul testo per seguire il cursore video.

Ritorniamo là dove eravamo rimasti prima della spiegazione su come TEXT gestisce i testi. Come già detto, il testo digitato sulla tastiera appare sul video che contiene 40 caratteri per riga. A differenza di una macchina da scrivere, non è necessario che l' utente prema [ENTER] alla fine di ogni riga per andare a capo, poiché il programma esegue, quando necessario, il riposizionamento a capo delle parole non inseribili nella riga. Il tasto ENTER andrà quindi premuto solo quando occorre esplicitamente un a capo, come ad esempio alla fine di un paragrafo. Il cursore, che indica la posizione del prossimo carattere digitato dall' utente, durante la battitura del testo procede automaticamente di colonna in colonna e di riga in riga.

I tasti movimento cursore

Il cursore video può essere spostato mediante i tasti di movimento cursore, situati nella parte superiore destra della tastiera ed identificati da frecce. La funzione di questi tasti è illustrata nella seguente tabella:

TASTO	MOVIMENTO CURSORE
→	uno spazio a destra
<SHIFT> + →	all'inizio della parola seguente
<CTRL> + →	alla fine della riga in corso
←	uno spazio a sinistra
<SHIFT> + ←	all'inizio della parola precedente (o attuale)
<CTRL> + ←	all'inizio della riga in corso
↑	una riga più in alto
<SHIFT> + ↑	alla prima riga del display
<CTRL> + ↑	all'inizio del file
↓	una riga più in basso
<SHIFT> + ↓	all'ultima riga del display
<CTRL> + ↓	alla fine del file

Figura 11.1

Analogamente a quasi tutti gli altri tasti, anche quelli per il movimento del cursore svolgono, se mantenuti premuti, la ripetizione automatica della funzione. Tenendo premuto il tasto con la freccia a sinistra, il cursore procede continuatamente da destra verso sinistra, risalendo dall' inizio di una riga al termine della precedente. Giunto il cursore nell' angolo superiore sinistro, appare la linea di testo precedente, facendo di conseguenza spostare la "finestra" video di una riga verso l' alto. Il tasto con la freccia a destra esegue la scansione inversa, da destra verso sinistra, dall' alto verso il basso. Una pressione continuata del tasto con la freccia in alto provoca invece una scansione verticale, dal basso verso l' alto, facendo via via apparire le righe di testo precedenti, fino a raggiungere l' inizio del file. Funzione inversa svolge infine il tasto con la freccia in basso che fa procedere il cursore video dall' alto verso il basso fino al termine del file.

Correzioni

Eventuali correzioni agli inevitabili errori di battitura possono essere eseguite posizionando, in uno dei modi visti, il cursore sul carattere successivo a quello da cancellare e premendo [DEL/BS], oppure posizionandosi proprio sul carattere da eliminare e premendo contemporaneamente [SHIFT] e [DEL/BS]. Per inserire invece uno o più caratteri, è necessario posizionare il cursore sul punto desiderato e digitare l' inserto; questo verrà inserito spostando automaticamente verso destra il rimanente testo.

Comandi e funzioni

Il programma applicativo TEXT possiede, come già detto, comandi e funzioni per l' elaborazione del testo che sono assegnati ai tasti funzione F1 - F8 e che possono essere visualizzati premendo [LABEL]. Esaminiamo allora in dettaglio questi comandi:

[F1] - Funzione di ricerca (Find)

La funzione di ricerca *Find* consente di cercare, a partire dalla posizione corrente del cursore, una stringa di caratteri nel testo. *Find* può essere richiamata premendo il tasto funzione F1, che fa apparire il messaggio

String:

nell' angolo inferiore destro del video. In risposta a questa richiesta occorre digitare la stringa di massimo 24 caratteri che si vuole ricercare nel testo. La pressione di

[ENTER] farà cominciare la ricerca della stringa indicata a partire dalla posizione attuale del cursore video procedendo verso la fine del file. Se nel testo è presente la sequenza cercata, il cursore si posiziona all' inizio della stringa trovata e scompare il messaggio *String:*. Se si desidera continuare ancora la ricerca basterà premere nuovamente [F1] che riproporrà la ricerca della stringa precedente. Premendo [ENTER] il programma TEXT ricomincerà la ricerca a partire dall' ultima sequenza trovata. Nel caso invece che la stringa non è o non è più presente nel testo, viene visualizzato il messaggio:

No match (non trovata)

che evidenzia l' esito negativo della ricerca.

Nell' utilizzare *Find* per la ricerca di stringhe, occorre tenere presente che questa funzione non fa distinzioni tra lettere maiuscole e minuscole, e che pertanto la sequenza trovata può essere diversa da quella impostata. Analogamente, se la stringa cercata non è racchiusa tra spazi, *Find* identificherà come valide anche parole che contengono al loro interno la sequenza cercata. Ad esempio, supponendo di avere il seguente testo:

"Ragazzo", mi chiese il caro Giovanni, "conosci la leggenda di Icaro?". "No zio", risposi felice .....

e supponendo di voler cercare tutte le parole *caro* contenute nel testo, se si specifica la stringa "caro", *Find* restituirà prima caro e poi Icaro, mentre se si specifica la stringa " caro " (caro preceduto e seguito da uno spazio), *Find* riconoscerà solo la parola caro.

Ricordatevi inoltre che la ricerca è eseguita dalla posizione attuale del cursore verso la fine del file. Pertanto se si desidera eseguire la ricerca in tutto il testo è necessario portare il cursore all' inizio del file premendo [CTRL] e [freccia in alto] contemporaneamente.

Il comando LOAD

Il comando *Load* consente di caricare un testo dal registratore a cassetta. Per selezionare questo comando premere [F2], in risposta del quale viene visualizzato il messaggio:

Load from:

Dopo essersi assicurati che il registratore è correttamente collegato e che il suo volu-

me di riproduzione è sufficiente, avviare, se connesso lo spinotto REM, il registratore in riproduzione, altrimenti attendere finché non si sia premuto [ENTER]. Svolte queste operazioni, digitare il nome della registrazione desiderata. Durante la ricerca del file l' M10 emette un suono acuto, a meno che non sia stato disabilitato l' altoparlante interno con l' istruzione basic SOUND OFF (vedi cap. 8). Se il primo file presente sul nastro è quello cercato, TEXT visualizza il messaggio:

FOUND: nome del file

per indicare il ritrovamento ed il conseguente caricamento del testo, mentre se il file trovato non è quello specificato, viene emesso il messaggio:

SKIP: nome del file

ad evidenziare che il testo trovato viene ignorato.

Con il comando *Load* è possibile caricare testi precedentemente memorizzati con *Save* per rivederli, modificarli o stamparli, oppure aggiungere il testo su nastro ad uno già esistente in memoria. Per fare ciò è sufficiente richiamare il programma TEXT, selezionare il testo contenuto in memoria e premere [F2] per aggiungere in coda al testo il file specificato.

Eventuali danneggiamenti al nastro potranno causare l' interruzione del caricamento del testo, evento che viene segnalato da un breve segnale acustico e dal messaggio *aborted*, messaggio che compare anche quando il file non può essere contenuto interamente in memoria (memoria insufficiente) oppure quando si interrompe manualmente il caricamento con [SHIFT] + [BREAK/PAUSE].

Maggiori informazioni sull' uso del registratore a cassetta sono contenute nell' appendice A.

Il comando SAVE

I testi creati con TEXT vengono automaticamente memorizzati nella memoria RAM. Può tuttavia essere utile, sia per ragioni di spazio che per avere una copia di sicurezza, salvare su nastro il testo redatto. Il comando *Save*, associato al tasto funzione F3, consente proprio di svolgere questa operazione. Premendo [F3] appare sull' ultima riga del display il messaggio:

Save to:

che richiede il nome (massimo 6 caratteri) con cui si vuole salvare il testo.

Prima di digitare il nome, dopo esservi assicurati sull' esatta connessione del regitratore all' M10, avviatelo in registrazione (se non dovesse essere collegato lo spinotto REM, eseguite questa operazione dopo l' introduzione del nome). La scomparsa del messaggio dallo schermo indica il termine dell' operazione di registrazione

Maggiori informazioni sull' uso del registratore a cassetta sono contenute nell' appendice A.

La funzione SELECT

La funzione *Select* (seleziona) consente di definire un blocco di testo sul quale eseguire successivamente operazioni di copiatura ([F5] - *Copy*), cancellazione ([F6] - *Cut*), o di trasferimento (*Cut e PASTE*). Per definire un blocco di testo occorre posizionarvi all' inizio il cursore video, premere [F7] e successivamente portare il cursore al termine del blocco; tutto il testo definito in questo modo apparirà in negativo (bianco sul fondo nero). Lo spostamento del cursore durante lo svolgimento della funzione *Select* può essere effettuato sia mediante i tasti movimento cursore, sia mediante la funzione di ricerca *Find*. In questo caso, per definire la parte di testo desiderata, dopo aver posizionato il cursore all' inizio del blocco, premere nell' ordine prima [F7] e poi [F1]. Digitata la stringa con la quale termina il blocco di testo, premete [ENTER]: tutto il testo compreso tra il primo carattere definito e la posizione attuale del cursore sarà visualizzato in negativo.

Il blocco di testo può avere lunghezza arbitraria, compatibilmente però con la quantità di memoria disponibile; messaggi di *Memory full* stanno proprio ad evidenziare questa eventualità.

La funzione *Select*, al pari di *Find*, *Load* e *Save*, può essere interrotta premendo i tasti [SHIFT] e [BREAK/PAUSE] contemporanemente. In tal caso il testo definito in precedenza tornerà ad essere visualizzato in positivo.

La funzione COPY

Il programma applicativo TEXT utilizza, per poter svolgere operazioni di copia e di spostamento, un buffer di memoria entro cui poter memorizzare temporaneamente blocchi di testo definiti con la funzione *Select*. Il tasto PASTE consente poi di inserire nel testo, a partire dalla posizione corrente del cursore, quanto contenuto nel buffer. Due sono le funzioni che lo adoperano: *Copy* e *Cut*.

La funzione *Copy* consente di copiare (duplicare) parti di testo che ricorrono frequentemente. Per adoperare questa funzione è necessario definire, tramite *Select*, il blocco di testo che si desidera duplicare (come appena descritto), quindi premere [F5] che memorizza il testo visualizzato in negativo nel buffer PASTE, cancellando ciò che vi era contenuto in precedenza. Per duplicare la parte di testo definita, sarà a questo punto sufficiente portare il cursore video nella o nelle posizioni desiderate e premere [PASTE].

Il blocco selezionato rimarrà memorizzato nel buffer PASTE finché non verrà eseguita una nuova funzione *Select*.

La funzione CUT

Questa funzione consente di svolgere due operazioni differenti: cancellare un blocco di testo oppure spostarlo di posizione. Per eseguire una di queste due operazioni è necessario prima di tutto selezionare il blocco di testo interessato con *Select*, quindi premere [F6]. Il testo visualizzato in negativo verrà così cancellato e memorizzato nel buffer PASTE, disponibile per essere inserito in un' altra posizione del file. Nel caso si voglia spostare quanto cancellato, occorre portare il cursore nella posizione desiderata e premere [PASTE]: il testo cancellato sarà inserito nella posizione in dicata.

Il buffer PASTE non viene cancellato quando si esce dal programma TEXT, ma mantiene intatto il suo contenuto fino a che un viene eseguita una nuova *Select*. Questa caratteristica può essere utilmente sfruttata per duplicare o spostare blocchi di testo da un file in un altro.
Selezionato il blocco desiderato con [F7] e premuto [F5] o [F6], occorrerà uscire dal testo attuale con [F8] e richiamare TEXT con l' altro file. Dopodiché, dopo essersi posizionati nel punto desiderato, premere [PASTE].

Stampa del testo

Il comando di stampa associato al tasto PRINT è utilizzabile solo con stampanti parallele, che devono essere collegate all' M10 tramite l' apposito cavo della Olivetti (per maggiori ragguagli consultare l' appendice A). Per stampare il contenuto del display occorre premere [PRINT], mentre la pressione di questo tasto accompagnato da [SHIFT] provoca la stampa dell' intero testo. In quest' ultimo caso per cominciare la stampa occorre in risposta al seguente messaggio:

Width? (ampiezza)

specificare la larghezza della riga di stampa, valore che deve essere compreso tra 10 e 132 ed infine premere [ENTER]. Se viene premuto [ENTER] senza avere introdotto alcun valore, verrà assunto l' ultimo valore impostato. Il valore di default per questo parametro è 80.

La stampa del testo viene eseguita con incolonnamento a sinistra ed il numero specificato di caratteri per riga. La stampa può essere bloccata in qualsiasi momento con [SHIFT] e [BREAK/PAUSE].

La funzione menù

La pressione del tasto F8 (menù) consente di chiudere il file di testo e abbandonare il programma TEXT tornando al menù principale. Il testo così creato o modificato sarà memorizzato come file nella directory della memoria RAM.

11.2 Consigli sull' utilizzazione di TEXT

Il programma applicativo TEXT sarà per voi un prezioso aiuto nella stesura di lettere, documenti, articoli o qualsiasi altro tipo di scritto. Tuttavia, specie se non possedete molta pratica con sistemi di videoscrittura, per ottenere da esso il miglior rendimento possibile, dovete attenervi a delle semplici regole:

Adoperate per i vostri testi nomi possibilmente significativi, come ad esempio LETT per una lettera, CIRC per una circolare, ARTIC per un articolo, e così via. Evitate quindi nomi come AAA, X3, P. Ricordatevi poi di annotare sempre i nomi, il contenuto e la posizione dei file memorizzati su cassette: all' occorrenza risparmierete molto tempo.

Evitate di battere [ENTER] al termine di ogni riga, in quanto questo tasto deve essere adoperato solo dove esplicitamente richiesto, come ad esempio al termine di un paragrafo. Se siete abituati alla macchina da scrivere le prime volte dovrete probabilmente fare molta attenzione per corregere la vecchia abitudine, ma con un po' di buona volontà riuscirete in breve tempo a scrivere correttamente.

Indipendentemente dalla memoria disponibile, non realizzate testi troppo lunghi, piuttosto divideteli in più parti. Questo vi permetterà di manipolare testi più brevi con i relativi vantaggi: tempi di ricerca più brevi, tempi di caricamento e di salvataggio ridotti, possibilità di coesistenza di più file ed infine minore vulnerabilità del testo a cancellazioni accidentali.

Realizzate sempre, almeno dei testi più importanti, due copie su nastri diversi. Eviterete così la spiacevole sorpresa di perdere qualche documento o scritto importante.

Ricordate che ogni modifica fatta al testo durante l' impiego del programma TEXT si ripercoute automaticamente sul file contenuto nella memoria RAM. Se desiderate pertanto fare delle modifiche ad un testo importante, salvate l' originale su nastro.

Se siete indecisi su una parte del testo da cancellare, eseguite questa operazione con la funzione *Cut*; nel caso cambiaste idea ciò vi permetterà di recuperare il testo cancellato dal buffer PASTE.

Se adoperate l' M10 in luoghi aperti per scrivere testi, evitate di abbandonarlo anche per pochi istanti; al vostro ritorno potrebbe non esserci più!

Capitolo 12

Il programma applicativo Telcom

L' interfaccia seriale RS 232C ed il programma TELCOM vi aprono letteralmente le porte del mondo intero. Con esso potrete infatti collegarvi, tramite linea telefonica, a centri di calcolo, banche dati, o inviare e ricevere messaggi da altri computer (posta elettronica). Nel caso invece che l' M10 sia il vostro secondo calcolatore, potrete, tramite cavo diretto, riversare al computer principale testi e programmi elaborati con l' M10 a casa, in viaggio, sulla spiaggia, ecc...

Il funzionamento di TELCOM

Prima di cominciare ad esaminare il programma TELCOM, sarà bene spiegare come si collega l' M10 ad una linea telefonica o ad un altro calcolatore.

Per connettere l' M10 Modem ad una linea telefonica è sufficiente disporre dell' apposito cavo fornito dalla Olivetti che andrà inserito nella presa PHONE situata sul retro del computer. Se invece si possiede l' M10 senza modem integrato è necessario disporre di un accoppiatore acustico e del relativo cavo (cavo seriale RS 232C). Dopo aver connesso quest' ultimo alla presa RS 232-C dell' M10 e a quella dell' accoppiatore, verificare che il commutatore CAL/ANS sia in posizione CAL (di chiamata) e quindi portare l' interruttore di accensione sulla posizione ON; a questo punto l' M10 è pronto per essere collegato ad un telefono. Maggiori informazioni sul funzionamento dell' accoppiatore acustico potranno trovarsi nelle istruzioni fornite dal costruttore. La connessione diretta tra l' M10 ed un altro calcolatore viene eseguita tramite cavo seriale RS 232-C "incrociato". Per maggiori ragguagli consultare l' appendice A.

Collegato l' M10 ad una linea telefonica o ad un altro calcolatore, è possibile accedere al programma TELCOM dal menù principale. Per fare ciò, posizionare il cursore sul nome TELCOM e premere [ENTER], oppure digitare TELCOM [ENTER]. Il display visualizzerà quanto mostrato nella figura seguente:

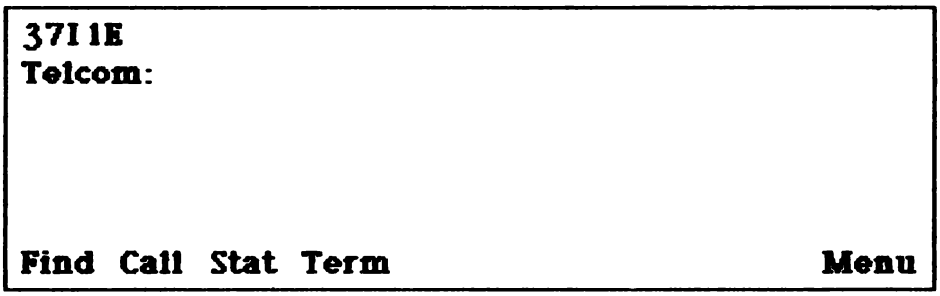


Figura 12.1

La prima riga del video contiene una lista di parametri di comunicazione, spiegati in dettaglio più avanti. La seconda contiene invece il nome del programma, mentre nell' ultima sono visualizzati i significati dei tasti funzione.

Il programma TELCOM possiede due diverse modalità operative: la modalità *Entry* (accesso) e quella *Term* (terminale).

12.1 La modalità Entry

La modalità Entry, selezionata automaticamente dal programma al momento della sua chiamata, consente di eseguire operazioni di ricerca e chiamata di numeri telefonici, variare i parametri di comunicazione, passare alla modalità Term o tornare al menù principale. Tutte queste funzioni sono associate ai tasti funzione F1 - F8, come riportato nella figura 12.1. Vediamo allora in dettaglio questi comandi:

[F1] - Find

Questa funzione, disponibile solo sul modello Modem, consente di cercare un nome contenuto nel file ADRS.DO e di comporre automaticamente il numero telefonico associato. È indispensabile che al momento della chiamata di questa funzione sia presente

in RAM il file ADRS.DO, il cui formato deve essere:

Nominativo : numero di telefono

(per maggiori informazioni consultare il capitolo 13, paragrafo 1, alla voce il formato indirizzi).

Per cercare un numero bisogna premere il tasto funzione F1 che visualizza il messaggio:

Telcom: Find

che richiede il nome della persona (o di qualsiasi altra cosa) che si desidera chiamare. Digitato quanto richiesto, la funzione Find cercherà nel file ADRS.DO il nome specificato, o se questo non fosse presente in RAM, emetterà un breve segnale acustico. La ricerca del nome viene eseguita partendo dall' inizio del file fino a trovare il primo dato che contenga la stringa specificata, che non deve necessariamente riferirsi al nome, ma che può anche riferirsi al numero telefonico, all' indirizzo, ecc.... Alla prima occorrenza della stringa cercata viene visualizzato il nome ed il numero di telefono, mentre l' ultima riga del display mostra le nuove funzioni disponibili:

[F2] - Call, [F3] - More e [F4] - Quit

Premendo [F3] il computer proseguirà la ricerca del nominativo indicato, mentre la pressione di [F4] fà terminare la ricerca e ritornare alla situazione di figura 12.1. Il tasto funzione F2 consente invece di comporre il numero telefonico visualizzato, operazione che viene evidenziata sullo schermo mediante la scritta:

Calling nominativo: numero telefonico

dove il numero telefonico viene visualizzato cifra per cifra contemporaneamente alla sua composizione.

Affinché la funzione di auto-chiamata *Find* abbia esito positivo, è indispensabile che la connessione M10 Modem - linea telefonica sia correttamente eseguita. Inoltre, prima che il programma termini la composizione del numero, è necessario che l' utente sollevi il ricevitore telefonico. Effettuata la chiamata, TELCOM ritorna alla situazione di figura 12.1.

[F2] - Call

Questa funzione, attiva solo sull' M10 modello Modem, consente di chiamare un qualsiasi numero telefonico. Per eseguire la chiamata, premere il tasto funzione F2, in risposta del quale comparirà il messaggio:

Telcom: Call

che indica la richiesta del numero telefonico da chiamare. Digitato il numero, il programma lo compone visualizzandolo cifra per cifra accanto al messaggio *Calling* che evidenzia l' operazione in corso.

Affinché la funzione di chiamata manuale *Call* abbia esito positivo, è indispensabile che la connessione M10 Modem - linea telefonica sia correttamente eseguita. Inoltre, prima che il programma termini la composizione del numero, è necessario che l' utente sollevi il ricevitore telefonico. Effettuata la chiamata, TELCOM ritorna alla situazione di figura 12.1.

[F3] - Stat

La comunicazione tra calcolatori si svolge secondo alcuni parametri che definiscono la velocità di comunicazione, la lunghezza dei dati, la presenza ed il tipo dei bit di parità, il numero dei bit di stop. Per fare comunicare l' M10 con un altro sistema è quindi necessario conoscere questi parametri ed impostarli a TELCOM per renderli compatibili. La funzione *Stat* consente di visualizzare i parametri di comunicazione selezionati e di modificarne i valori.

Per visualizzare questi parametri, elencati automaticamente alla chiamata del programma, occorre premere il tasto funzione F4 seguito da [ENTER]. Sul display verranno elencati i parametri, i cui possibili valori ed il cui significato sono riportati nella figura 12.2.

I parametri di default sono *M7I1E,10 pps* per il modello Modem e *37I1E* per il modello standard. Per modificare questi parametri occorre premere [F3] ed impostare la nuova lista. Se la lista non è corretta, il programma emette un breve segnale acustico e rifiuta il comando. La verifica dell' avvenuta modifica dei parametri può essere eseguita ripremendo [F3] e subito dopo [ENTER]. Sul display verranno così visualizzati i nuovi valori.

PARAMETRI	VALORI	SIGNIFICATO
Velocita' di comunicazione (v)	M	Modem (300 baud)
	1	75 baud
	2	110 baud
	3	300 baud
	4	600 baud
	5	1200 baud
	6	2400 baud
	7	4800 baud
	8	9600 baud
	9	19200 baud
Lunghezza parola (f)	6	6 bit
	7	7 bit
	8	8 bit
Parita' (p)	O	Dispari
	E	Pari
	N	Nessuna parita'
	I	Ignora parita'
Bit di stop (s)	1	1 bit di stop
	2	2 bit di stop
Linea di stato (x)	E	Abilitata
	D	Disabilitata
Velocita' impulsi	10	10 pps
	20	20 pps

Figura 12.2

[F4] - Term

La pressione di questo tasto funzione fà passare il programma in modalità terminale, per la cui spiegazione rimandiamo a dopo.

[F8] - Menù

Il tasto funzione F8 provoca l' uscita dal programma e il conseguente ritorno al menù principale.

12.2 La modalità Terminal

La modalità *Terminal* trasforma l' M10 in un terminale capace di comunicare, via linea telefonica o cavo seriale, con altri sistemi o con centri di servizi telematici.

Alla modalità *Terminal* si accede premendo il tasto funzione F4 da *Entry*, che modifi-

ca anche l' ultima riga del video per visualizzare le nuove funzioni associate ai tasti [F1] - [F8]. L' M10 diventa così, se collegato correttamente, un terminale video il quale invia i caratteri digitati sulla tastiera e visualizza sul display quelli ricevuti dal sistema connesso. Ma vediamo ora in dettaglio le funzioni ed i comandi di *Terminal*:

[F1] - Prev

Questa funzione permette di visualizzare le otto righe precedenti di testo. Premendo [F1] queste vengono visualizzate finché una nuova pressione del tasto non fà tornare il video alla situazione originale.

[F2] - Down

Molte volte quando si opera con un terminale si desidera memorizzare le informazioni ricevute per poterle poi successivamente rivedere ed elaborare. Il comando *Down* consente di memorizzare in un file di testo i dati trasmessi dal sistema collegato. Per eseguire il trasferimento, dopo aver digitato (**non premere ancora [ENTER]**) i comandi richiesti dal sistema per la trasmissione del file, premere il tasto funzione F2, in risposta del quale compare il messaggio:

File to download?

che richiede il nome del file su cui memorizzare i dati inviati. Il nome introdotto deve rispettare le norme già viste per i file di tipo testo. Se il file specificato non è presente in RAM, questo verrà creato da TELCOM ed inserito in memoria. Introdotto il nome e premuto nuovamente [ENTER] per completare il comando di trasmissione del sistema, se non sono state commesse irregolarità, inizierà la memorizzazione dei dati evidenziata dalla visualizzazione in negativo della scritta *Down*; a trasferimento terminato, questa tornerà in "positivo". La nuova pressione di [F2] chiuderà la memorizzazione dei dati. Il trasferimento può anche essere concluso premendo [SHIFT] e [BREAK/PAUSE] contemporaneamente, che causano la comparsa del messaggio *Download aborted* (trasferimento abortito), scritta che compare anche se il nome del file introdotto non è corretto.

I file così trasferiti vengono memorizzati nella memoria RAM come file di tipo testo (.DO) che pertanto possono essere richiamati ed elaborati dal programma TEXT. Se il file trasferito è un programma basic, questo può essere trasformato in formato binario

(.BA) eseguendo da ambiente basic *LOAD "RAM:nome del file"* ed una successiva *SAVE "RAM:nome del file"* che memorizzerà il file specificato in formato binario (per maggiori informazioni consultare il capitolo 3, paragrafo 4).

[F3] - Upload

Il comando *Upload* permette di trasmettere al sistema collegato un file di testo contenuto in RAM. Dopo aver predisposto il sistema alla ricezione dei dati inviati da TELCOM, premere [F3], in risposta del quale viene visualizzato il messaggio:

File to upload?

che richiede il nome del file di testo da trasmettere. Introdotto il nome, compare un nuovo messaggio:

Width?

in risposta al quale bisogna definire la larghezza di riga del file da inviare, il cui valore deve essere compreso tra 10 e 132. Alcuni sistemi di comunicazione possono ricevere soltanto righe di una determinata ampiezza, terminate da un ritorno a capo. Poiché il testo preparato con TEXT, avendo quest' ultimo il riposizionamento a capo automatico, può contenere paragrafi di lunghezza arbitraria, il parametro *Width* stabilisce ogni quanti caratteri, se non viene trovato un ritorno a capo, bisogna inviare un ritorno a capo. Se viceversa il sistema può ricevere paragrafi di qualsiasi lunghezza, premendo soltanto [ENTER] non verrà aggiunto alcun ritorno a capo durante la trasmissione del file.

Digitato il valore richiesto e premuto [ENTER] inizierà la trasmissione dei dati evidenziata dalla visualizzazione in "negativo" della scritta *Upload*; a trasmissione terminata questa tornerà in "positivo". Il trasferimento può anche essere concluso premendo [SHIFT] e [BREAK/PAUSE] contemporaneamente, che causano la comparsa del messaggio *Upload aborted* (invio abortito), scritta che compare anche nel caso in cui il nome del file introdotto non sia corretto.

[F4] - Full/Half

Questo tasto funzione seleziona la modalità di comunicazione tra *Full duplex* (trasmissione e ricezione contemporanee) e *Half duplex* (trasmissione e ricezione alterna-

te). In *Full duplex*, la modalità di comunicazione più diffusa, i caratteri trasmessi dal terminale (in questo caso l' M10) vengono ritrasmessi dal sistema e visualizzati sul display, consentendo così un controllo visivo sulla buona riuscita della trasmissione. In *Half duplex* invece, mancando la possibilità di trasmettere e ricevere contemporaneamente, i dati inviati al sistema sono visualizzati al momento del loro invio, senza consentire quindi alcun controllo su quanto ricevuto.

Per selezionare una delle due modalità basta premere il tasto funzione F4 che funge da "interruttore": la sua pressione seleziona alternativamente Full o Half duplex.

[F5] - Echo

Questa funzione attiva l' eco in stampa di quanto visualizzato sul display. Ogni messaggio che compare sul video viene stampato sulla stampante parallela collegata all' M10, la quale fornirà così un resoconto della sessione di lavoro con il sistema. Premendo una volta il tasto funzione F5 viene abilitato l' eco e visualizzata la scritta *Echo* sul display, ripremendolo la seconda volta si disabilita la funzione e scompare la scritta dal video.

[F8] - Bye

La pressione del tasto F8 provoca l' uscita da *Terminal* ed il ritorno alla modalità *Entry*.

12.3 Modalità di accesso a sistemi informatici

Abbiamo già visto come TELCOM consente di trasformare l' M10 in un terminale remoto. In questo paragrafo esamineremo come si accede a centri di calcolo o banche dati, sia in modo manuale, che in automatico (solo per M10 Modem), tramite linea telefonica.

Per poter connettersi ad un centro di calcolo o ad una banca dati è innanzitutto necessario avere avuto una regolare autorizzazione all' accesso del centro, o avere stipulato un abbonamento con la società che gestisce il servizio. In ambedue i casi vi verranno fornite tutte le informazioni necessarie per accedere correttamente al sistema, e precisamente:

— I parametri di trasmissione

— Il carattere di chiamata del sistema

— Il vostro codice di identificazione

— La vostra parola chiave

— Informazioni sull' uso delle risorse del sistema

Conosciute queste informazioni è possibile collegarsi al centro desiderato tramite linea telefonica, sia in modo manuale che automatico.

Accesso manuale

Per prima cosa è necessario collegare l' accoppiatore acustico (preferibilmente l' Olivetti MC10) all' M10 tramite cavo seriale RS 232-C, posizionare il commutatore ANS/CAL su CAL e portare l' interruttore di accensione sulla posizione ON come già illustrato in apertura di capitolo.
Predisposto l' accoppiatore, chiamare il programma TELCOM e modificare, se necessario, i parametri di trasmissione sulla base delle informazioni ricevute. Sollevato il ricevitore telefonico, componete il numero del centro e, non appena stabilita la comunicazione, inserite il ricevitore nell' apposito alloggiamento dell' accoppiatore acustico. A questo punto, premete [F4] per passare in modalità terminale e digitate il carattere di chiamata del sistema per segnalare la vostra predisposizione alla procedura di collegamento (LOG ON). Se si sono eseguite correttamente tutte le operazioni precedenti, il sistema risponderà inviando un messaggio di richiesta del codice di identificazione, fornito il quale chiederà la parola chiave. Digitata la parola chiave, se sono stati forniti dati esatti, il sistema riterrà concluso il LOG ON e concederà l' accesso al servizio.

La procedura di LOG ON può leggermente variare da sistema a sistema. Maggiori informazioni sulle modalità di accesso potranno eeservi fornite dalla società che gestisce il servizio.

La modalità di accesso manuale può anche essere eseguita dai possessori dell' M10 modello Modem, seguendo la stessa procedura descritta sopra, con le ovvie differenze dovute alla presenza del modem al posto dell' accoppiatore acustico.

Accesso automatico

Questa modalità di accesso, riservata soltanto ai possessori del modello Modem, consente di eseguire automaticamente la procedura di LOG ON.

Nel paragrafo 12.2 abbiamo già visto come è possibile, tramite la funzione *Find*, cercare e chiamare un numero telefonico contenuto nel file ADRS.DO, di cui abbiamo anche illustrato il formato. Per eseguire in modo automatico la procedura di LOG ON è necessario memorizzare nel file ADRS, alla destra del numero telefonico, una stringa racchiusa tra i simboli < e > secondo il seguente schema:

Nominativo : numero telefonico < stringa >

La stringa racchiusa tra < e < serve per codificare la procedura di LOG ON mediante quattro comandi identificati dai seguenti caratteri:

? Questo simbolo pone l' M10 in attesa finché non viene ricevuto il carattere o il codice di controllo specificato dopo il punto interrogativo. Così ad esempio se il sistema richiede il numero di identificazione con il messaggio *ID utente:*, la stringa potrebbe contenere *?I*, il cui significato è quello di porre in attesa TELCOM fino a quando non viene ricevuto il messaggio di richiesta del codice di identificazione.

= Questo simbolo determina una pausa di due secondi prima di continuare la procedura di LOG ON.

! Il punto interrogativo fà si che il successivo carattere della stringa venga trasmesso come un qualsiasi carattere ASCII. Questo simbolo è usato per inviare i caratteri ?, !, = e ∧, che altrimenti sarebbero interpretati dal programma come comandi. La sequenza *!* = viene pertanto trasmessa come carattere =, diversamente dal solo = che provocherebbe una pausa di due secondi.

∧ L' accento circonflesso fà si che il successivo carattere della stringa venga trasmesso come carattere di controllo (CTRL + carattere). Così ad esempio ∧X viene trasmesso come *CTRL X*, codice ASCII 24.

Per illustrare l' uso di questi comandi supporremo di codificare passo per passo una stringa che definisca una procedura automatica di LOG ON. Come già visto per l' accesso manuale, per connettersi ad un centro di calcolo o ad una banca dati, è necessario conoscere alcune informazioni, e precisamente:

- Il numero telefonico

- Il carattere di chiamata del sistema

- Il proprio codice di identificazione

- La parola chiave assegnata

- Eventuali messaggi di conferma dell' avvenuto LOG ON.

Supponiamo che la nostra banca dati, la B.D.I. (banca dati inventata) abbia numero telefonico 050 502434, CTRL X come carattere di chiamata, che il nostro codice di identificazione sia ML6045 e che la nostra parola chiave sia ABRACADABRA. Supponiamo ancora che i messaggi di richiesta del sistema siano nell' ordine: *ID utente:*, *Parola chiave:* e *LOG-ON completato con successo*. Vediamo allora come codificare la stringa racchiusa tra < e >.

È buona norma iniziare la stringa con il comando = per lasciare un breve intervallo di tempo tra la chiamata del numero e l' inizio della procedura di LOG ON onde considerare il tempo necessario per la commutazione della comunicazione. Considerato che il carattere di chiamata del sistema è CTRL X, i primi caratteri della stringa saranno:

= ∧X.

Prima di inviare il proprio codice di identificazione bisogna attendere l' invio del messaggio *ID Utente:*, in risposta del quale deve essere trasmesso *ML6045* (il codice di identificazione del nostro esempio). Pertanto sarà necessario porre l' M10 in attesa del messaggio del sistema. Poiché il comando ? definisce solo un carattere, è necessario scegliere una delle lettere costituenti il messaggio, a meno che non lo si voglia specificare carattere per carattere. Per semplificare le cose sceglieremo la prima lettera del messaggio. I successivi caratteri della stringa saranno pertanto:

?IML6045

Analogo discorso vale per l' inserimento della parola chiave. In risposta alla richiesta *Parola chiave:* andrà quindi trasmesso ABRACADABRA.

?PABRACADABRA

Infine per considerare concluso con successo il LOG ON occorrerà attendere il messaggio *LOG-ON completato con successo*.

?L

Mediante il programma TEXT potremo quindi inserire nel file ADRS.DO i dati relativi alla banca dati esaminata:

B.D.I. (banca dati inventata) : 050 = 502434 > = ∧X?IML6045?PA-BRACADABRA?L >

Preparato il file ADRS.DO, vediamo finalmente come eseguire il LOG ON automatico.

Verificata l' esatta connessione del modem alla linea telefonica, posti gli interruttori DIR/ACP su DIR e CAL/ANS su CAL, accedere al programma TELCOM e verificare che i parametri di trasmissione siano esatti. Eseguiti questi controlli, premere [F1] ed in risposta al messaggio *Find* digitare il nome richiesto (B.D.I. nel nostro esempio). Sul video verrà visualizzato il nominativo ed il relativo numero telefonico seguito da < > e non dalla stringa della procedura di LOG ON, al fine di non compromettere la segretezza del proprio codice di identificazione e della propria parola chiave. La pressione del tasto funzione F2 (Call) seguito subito dopo da [ENTER] inizierà la procedura di LOG ON automatico.

12.4 Consigli sull' uso di TELCOM

Per ottenere il massimo risultato dal programma TELCOM è consigliabile attenersi ad alcuni suggerimenti elencati qui di seguito:

Se pensate di adoperare spesso l' M10 come terminale, è preferibile acquistare la versione Modem, che è senz' altro più indicata per le comunicazioni. A differenza del modello che ne è privo, l' M10 Modem possiede alcune funzioni per comporre numeri di telefono ed eseguire procedure di LOG ON in automatico. Inoltre il modem incorporato è indubbiamente più affidabile e pratico dell' accoppiatore acustico.

Adottate per il file ADRS.DO il formato dati riportato nei paragrafi precedenti; eventuali differenze potrebbero provocare ambiguità ed errori nell' uso della funzione *Find* o durante il LOG ON automatico.

Verificate i parametri di comunicazione prima di ogni connessione. Anche un solo parametro non appropriato potrebbe rendere incomprensibili i caratteri trasmessi e ricevuti. Prestare particolare attenzione alla velocità di comunicazione, alla lunghezza del dato ed al numero dei bit di stop.

Se il sistema al quale siete collegati lo consente, adoperate la modalità *Full duplex*, che vi consentirà di controllare visivamente la correttezza della trasmissione. Prestate

particolare attenzione ai collegamenti telefonici interurbani e a quelli fortemente disturbati.

Ricordate che la funzione *Download* cancella completamente il contenuto precedente del file specificato. Prestate pertanto particolare attenzione a non confondere [F2] con [F3]: viceversa potreste cancellare completamente il file da trasmettere. Anche per questo motivo vi esorto a fare sempre copie dei testi importanti.

Nel caso abbiate un permesso di accesso ad un centro di calcolo o ad una banca dati, conservate gelosamente il vostro numero di codice e la parola chiave del sistema. In questo modo eviterete che qualcuno, conoscendo il vostro codice, possa usufruire o accedere ai vostri dati. Abbiate quindi la massima segretezza sul vostro codice e sulla vostra parola chiave.

Capitolo 13

Il programma applicativo ADDRSS

Il programma applicativo ADDRSS trasforma l' M10 in una versatile rubrica elettronica nella quale possono essere memorizzate informazioni di vario genere: nomi, numeri di telefono, indirizzi, dati anagrafici, ecc... I dati contenuti possono essere ricercati in base ad una loro qualsiasi voce: ad esempio in base al nome, al numero di telefono, ecc...

Sebbene realizzato essenzialmente per svolgere funzioni di rubrica, il programma ADDRSS può anche essere adoperato per archiviare qualsiasi altro genere di informazioni. Mentre il paragrafo uno descrive il funzionamento del programma, i successivi due paragrafi mostrano due applicazioni guida: ADDRSS adoperato per memorizzare le ricette di cucina e per archiviare la storia medica familiare. Il quarto paragrafo contiene infine alcuni suggerimenti finali sull' uso del programma.

13.1 Il funzionamento di ADDRSS

ADDRSS ricerca e stampa le informazioni contenute nel file ADRS.DO, che deve essere necessariamente presente in RAM al momento della chiamata del programma. Se così non fosse, verrebbe visualizzato il messaggio:

```
ADRS.DO not found
Press space bar for MENU
```

la cui traduzione è: ADRS.DO non trovato, premere barra spazio per MENU.

La creazione del file ADRS.DO

In considerazione di ciò volgeremo la nostra attenzione alla creazione del file ADRS.DO.

Selezionato il programma TEXT dal menù principale, create un file di nome ADRS. Prima di inserire i dati nel file è importante scegliere un formato con cui memorizzare tutte le informazioni. Il formato può essere scelto liberamente secondo le proprie esigenze, ma se si dispone del modello M10 Modem e si desidera fare uso della funzione di chiamata automatica (funzione Find, vedi cap. precedente), è necessario adottare il seguente formato:

> Nome : numero telefonico : indirizzo : eventuali altre informazioni separate da due punti [ENTER]

Questo formato è particolarmente indicato per memorizzare nominativi, numeri telefonici, indirizzi e così via.

Ogni dato deve essere concluso premendo [ENTER], il quale non deve viceversa comparire in nessuna altra posizione del file; il carattere ritorno a capo è infatti il simbolo che separa un dato dal successivo. Nel prosieguo del capitolo adopereremo spesso i termini record e campo. Un record è un singolo dato, ovverossia una sequenza di caratteri terminata da un ritorno a capo. Un campo del record è invece la singola voce del dato, o per essere più precisi, una sequenza di caratteri terminata da due punti. Ad esempio:

> Massimo Mangia : 050=21273 : Via Catalani 35, 56100 - Pisa [ENTER]

è un record con tre campi: nome, numero telefonico e indirizzo.

Come avrete forse notato, nel campo numero telefonico compare il carattere =. Analogamente a quanto visto nel capitolo precedente, il simbolo = determina una pausa di due secondi nella chiamata automatica del numero (funzione Find di TELCOM).

Una volta adottato un formato, potrete cominciare ad inserire i record nel file ADRS.DO, utilizzando, se necessario, qualsiasi funzione o comando di TEXT. È importante adoperare il formato prescelto in tutto il file; ciò renderà uniforme il vostro archivio, con tutti gli ovvi vantaggi che la cosa comporta.

L' utilizzo del programma ADDRSS

Creato il file ADRS.DO ed inseriti tutti i nominativi necessari, possiamo accedere dal menù principale al programma ADDRSS. Per fare ciò, posizionare il cursore sul nome ADDRSS e premere [ENTER], oppure digitare la sequenza *ADDRSS [ENTER]*. Eseguita una di queste due operazioni, il display visualizzerà sulla prima riga la scritta *Adrs:* (indirizzo) e sull' ultima le funzioni assegnate ai tasti [F1] - [F8], che ora andiamo ad esaminare.

[F1] - Find

La funzione *Find* consente di visualizzare il contenuto del file ADRS.DO e di ricercare un record in esso contenuto. Per visualizzare il file premere [F1] ed [ENTER]: sul display compariranno le prima sei righe del file mentre i tasti [F3] ed [F4] assumeranno rispettivamente le funzioni *More* e *Quit*. Premendo [F3] verranno visualizzate le successive sei righe del file. Per interrompere la visualizzazione premere [F4] che disabilita le funzioni associate ai tasti [F3] ed [F4] e fà tornare il programma allo stato precedente la chiamata di *Find*.

Per ricercare un record nel file ADRS.DO premere il tasto [F1] e, in risposta al messaggio *Find*, digitare una stringa di caratteri contenuti nel record desiderato. La stringa può contenere più campi del record, o semmai specificare l' intero record. Il programma ADDRSS visualizzerà tutti i record contenenti la stringa specificata. Se i record trovati occupano più di sei righe del display, verranno assegnate ai tasti [F3] ed [F4] le funzioni *More* e *Quit* già viste in precedenza.

Per evidenziare meglio come opera la funzione *Find*, mostreremo alcuni esempi:

Supponiamo che il file ADRS.DO contenga i seguenti record:

A) Paolo Rossi : 02 = 435676 : Via Manzoni 70, 20121 - MILANO

B) Alberto Manzoni : 02 = 7463511 : Pzza F.lli Rossetti 3, 20131 - MILANO

C) Salvatore Milano : 050 = 88665 : Via de' Borghi Nuovi 65, 56100 - PISA

D) Luigi Borghi : 081 = 8712936 : Viale Europa 175, C/mare di Stabia 80053 - (NA)

E) Giacomo Rossetti : 06 = 765212: Viale Libia 109, ROMA

L' esecuzione di *Find Rossi* provocherà la visualizzazione del solo record A, mentre *Find Manzoni* visualizzerà i record A e B, essendo la stringa "Manzoni" contenuta in entrambi, rispettivamente nel campo indirizzo e nel campo nome.

Analogamente per la stringa "Milano" che visualizzerà i record A, B e C (il programma ADDRS non distingue le minuscole dalle maiuscole). Se però la stringa viene modificata in "- Milano", verranno ritrovati i soli record A e B. Da ciò si deduce che per trovare uno specifico record la stringa andrà definita con quante più informazioni possibili. Viceversa quanto più generica sarà la stringa specificata, tanti più saranno i record visualizzati.

[F5] - Lfnd

La funzione *Lfnd* svolge esattamente le stesse funzioni di *Find*, eccetto che i risultati ottenuti vengono stampati dalla stampante parallela.

[F8] - Menù

Il tasto funzione F8 provoca l' uscita dal programma ADDRSS e il successivo ritorno al menù principale.

Abbiamo così visto come va impiegato il programma ADDRSS per gestire un' agenda telefonica o un indirizzario. La flessibilità del programma consente tuttavia di svolgere numerose altre funzioni. Variando infatti il formato dei dati, è possibile memorizzare qualsiasi genere di informazioni. Nei paragrafi successivi vedremo due simpatiche ed utili applicazioni di questo programma.

13.2 Ricette a volontà

Tutti gli appassionati di cucina e tutte le persone golose troveranno molto utile quest' applicazione, che risolve uno dei più "angosciosi" interrogativi delle casalinghe: "Cosa cucinare oggi?". Vediamo come preparare il file ADRS.DO per memorizzare le ricette di cucina.

Il primo campo del record sarà destinato a contenere il nome della pietanza, mentre nel secondo potremmo inserire gli ingredienti e nel terzo una breve spiegazione sulla sua preparazione. Il formato del file può essere quindi così definito:

Nome pietanza : ingredienti : preparazione

Ad esempio:

Penne all' arrabbiata : olio, pancetta, cipolla, pomodori pelati, sale, peperoncino : rosolare la cipolla e la pancetta nell' olio, cuocere la pasta ed aggiungervi il condimento

Spaghetti aglio e olio : olio, aglio, pomodori pelati, sale, basilico : soffriggere olio, aglio e peperoncino, cuocere la pasta ed aggiungervi il condimento

e così via.....

Naturalmente questo non è il solo formato esistente per memorizzare delle ricette: potreste ad esempio mettere gli ingredienti prima del nome, o aggiungere altri campi contenenti altre informazioni, come ad esempio il tempo di cottura, il grado di difficoltà, ecc... Vediamo ora come adoperare il programma ADDRSS per gestire il nostro ricettario.

Se desiderata rivedere la ricetta di un piatto, selezionate la funzione *Find* e digitate il nome della pietanza: vedrete apparire sul video il record contenente le informazioni desiderate. Se viceversa avete alcuni ingredienti e non sapete cosa cucinare, selezionate la funzione *Find* e digitate l' ingrediente (o gli ingredienti) che possedete: avrete immediatamente una lista di pietanze che contengono l' ingrediente specificato. Buon appetito!

Appello ai lettori: se qualche lettore/lettrice particolarmente portata per la cucina realizzasse, tramite l' applicazione suggerita, qualche succulento dolce e volesse farmene dono, non per golosità ma per testimonianza sulla utilità di quanto scritto, può trovare il mio indirizzo nel paragrafo precedente.

13.3 Storia medica familiare

Anche se mi auguro che quest' applicazione non debba mai servire, poiché i bambini vanno soggetti a malattie dell' infanzia, e per evitare di dimenticare delle informazioni (anamnesi) che possono aiutare il medico nella sua diagnosi, ecco pronta per voi un' altra utile applicazione del programma ADDRSS.

Per definire il formato del file ADRS.DO vediamo un momento le informazioni che ci vanno memorizzate. Innanzitutto la data in cui si è ammalata la persona, poi il nome, la

malattia, la sua durata ed altre informazioni. Il formato del file potrebbe quindi essere il seguente:

Data (gg/mm/aa) : nome : malattia : durata : note

Ecco una storia medica familiare esemplificativa:

12/04/62 : Marco : morbillo : 15 : ....

16/02/63 : Marco : influenza : 4 : tonsille infiammate

08/07/63 : Francesca : varicella : 10 :

23/11/65 : Marco : operazione tonsille : 5 : meglio così

03/03/66 : Francesca : influenza : 3

Vediamo allora come si adopera il programma ADDRSS per gestire la storia medica familiare.

Se si desidera conoscere tutte le malattie avute da Marco occorrerà selezionare dal programma ADDRSS la funzione *Find* e digitare il nome Marco: sul video appariranno tutti i record in compare il nome Marco (nel nostro esempio il primo, il secondo ed il quarto). Se invece si vuole sapere chi ha avuto una determinata malattia, allora bisognerà selezionare la funzione *Find e specificare la malattia richiesta. Analogo discorso può esser fatto per la data, o per le note.*

13.4 Consigli sull' uso di ADDRSS

Elenchiamo qui di seguito i consueti consigli sull' utilizzazione dei diversi programmi applicativi.

La versatilità del programma ADDRSS consente di realizzare molte applicazioni nei campi più svariati. Nei paragrafi due e tre ho illustrato due simpatiche applicazioni, che potranno essere lo spunto da cui partire per realizzarne delle altre. Le possibilità che possiede ADDRSS sono veramente notevoli: saperle cogliere è solo questione di fantasia.

Dal momento che il programma ADDRSS funziona soltanto con il file ADRS.DO, per realizzare altre applicazioni è necessario memorizzare l' indirizzario su nastro e digitare il nuovo file con il nome ADRS.DO. Un' altra possibilità è quella di chiamare i file del-

le altre applicazioni con nomi diversi, e prima di accedere ad ADDRSS, cambiare da basic (comando NAME cap.3) i nomi del vecchio fila ADRS e del file con cui si desidera operare. Facendo un esempio se in RAM sono contenuti i file ADRS.DO contenente indirizzi ed il file CUCINA contenente il ricettario, se si desidera consultare quest' ultimo occorrerà eseguire:

NAME "ADRS.DO" AS "INDIR.DO" e NAME "CUCINA.DO" AS "ADRS.DO"

In questo modo il programma ADDRSS opererà con il ricettario. Per ripristinare la situazione di partenza, eseguire due nuove NAME.

Ricordate sempre di salvare su cassetta i vostri file dati: eviterete brutte sorprese.

Capitolo 14

Il programma applicativo SCHEDL

Il programma applicativo SCHEDL trasforma l' M10 in una versatile agenda elettronica nella quale possono essere memorizzate informazioni di vario genere: annotazioni, appuntamenti, ecc... I dati contenuti possono essere ricercati in base ad una loro qualsiasi voce: ad esempio in base alla data, al tipo, ecc...

Sebbene realizzato essenzialmente per svolgere funzioni di agenda, il programma SCHEDL può anche essere adoperato per archiviare qualsiasi altro genere di informazioni. Mentre il paragrafo uno descrive il funzionamento del programma, i successivi due paragrafi mostrano due applicazioni guida: SCHEDL adoperato per memorizzare un dizionario elettronico e per archiviare delle diapositive. Il quarto paragrafo contiene infine alcuni suggerimenti finali sull' uso del programma.

14.1 Il funzionamento di SCHEDL

SCHEDL ricerca e stampa le informazioni contenute nel file NOTE.DO, che deve essere necessariamente presente in RAM al momento della chiamata del programma. Se così non fosse, verrebbe visualizzato il messaggio:

```
NOTE.DO not found
Press space bar for MENU
```

la cui traduzione è: NOTE.DO non trovato, premere barra spazio per MENU.

La creazione del file NOTE.DO

In considerazione di ciò volgeremo la nostra attenzione alla creazione del file NOTE.DO.

Selezionato il programma TEXT dal menù principale, create un file di nome NOTE. Prima di inserire i dati nel file è importante scegliere un formato con cui memorizzare tutte le informazioni. Il formato può essere scelto liberamente secondo le proprie esigenze, attenendosi però a queste regole:

Ogni dato (record) deve essere concluso premendo [ENTER], il quale non deve viceversa comparire in nessuna altra posizione del record; il carattere ritorno a capo è infatti il simbolo che separa un dato dal successivo.

Le informazioni devono essere memorizzate nella maniera più logica ed ordinata possibile. È conveniente adottare dei prefissi per identificare il tipo dell' informazione e consentire una più facile ricerca de dati.

Una volta adottato un formato, potrete cominciare ad inserire i record nel file NOTE.DO, utilizzando, se necessario, qualsiasi funzione o comando di TEXT. È importante adoperare il formato prescelto in tutto il file; ciò renderà uniforme il vostro archivio, con tutti gli ovvi vantaggi che la cosa comporta.

L' utilizzo del programma SCHEDL

Creato il file NOTE.DO ed inseriti tutti i nominativi necessari, possiamo accedere dal menù principale al programma SCHEDL. Per fare ciò, posizionare il cursore sul nome SCHEDL e premere [ENTER], oppure digitare la sequenza *SCHEDL [ENTER].* Eseguita una di queste due operazioni, il display visualizzerà sulla prima riga la scritta *Schd:* (registro) e sull' ultima le funzioni assegnate ai tasti [F1] - [F8], che ora andiamo ad esaminare.

[F1] - Find

La funzione *Find* consente di visualizzare il contenuto del file NOTE.DO e di ricercare un record in esso contenuto. Per visualizzare il file premere [F1] ed [ENTER]: sul display compariranno le prima sei righe del file mentre i tasti [F3] ed [F4] assumeranno rispettivamente le funzioni *More* e *Quit*. Premendo [F3] verranno visualizzate le successive sei righe del file. Per interrompere la visualizzazione premere [F4] che disabilita le

funzioni associate ai tasti [F3] ed [F4] e fà tornare il programma allo stato precedente la chiamata di *Find*.

Per ricercare un record nel file NOTE.DO premere il tasto [F1] e, in risposta al messaggio *Find*, digitare una stringa di caratteri contenuti nel record desiderato. La stringa può essere di lunghezza arbitraria, fino a specificare l' intero record. Il programma SCHEDL visualizzerà tutti i record contenenti la stringa specificata. Se i record trovati occupano più di sei righe del display, verranno assegnate ai tasti [F3] ed [F4] le funzioni *More* e *Quit* già viste in precedenza.

Per evidenziare meglio come opera la funzione *Find*, mostreremo alcuni esempi:

Supponiamo che il file NOTE.DO contenga i seguenti record:

04/09 $	Pagare le ricevute bancarie arrivate
05/09 !	Telefonare a Paolo Telefonare alla Jackson
06/09 *	09:30 Volo AZ 245 per Milano Prenotare albergo
07/09 §	Appuntamento con Pancaldi
08/09 *	14:30 Volo AZ 246 per Pisa

dove il carattere $ rappresenta i viaggi, il punto interrogativo le telefonate, l' asterisco i viaggi e il simbolo del paragrafo gli appuntamenti.

Per visualizzare le incombenze di una giornata, premere [F1] e digitare la data del giorno. Se invece si vuole conoscere tutti i viaggi in programma, premere [F1] e digitare * [ENTER], che nel nostro esempio visualizzerà i record 6 e 8 settembre. Analogamente per qualsiasi altra voce contenuta nel file.

[F5] - Lfnd

La funzione *Lfnd* svolge esattamente le stesse funzioni di *Find*, eccetto che i risultati ottenuti vengono stampati dalla stampante parallela.

[F8] - Menù

Il tasto funzione F8 provoca l' uscita dal programma SCHEDL e il successivo ritorno al menù principale.

Abbiamo così visto come va impiegato il programma SCHEDL per gestire un' agenda elettronica. La flessibilità del programma consente tuttavia di svolgere numerose altre funzioni. Variando infatti il formato dei dati, è possibile memorizzare qualsiasi genere di informazioni. Nei paragrafi successivi vedremo due simpatiche ed utili applicazioni di questo programma.

14.2 Un dizionario elettronico

Chi non ha molta pratica con le lingue gradirà senz' altro quest' applicazione del programma SCHEDL. Per realizzare un semplice dizionario elettronico è necessario preparare convenientemente il file NOTE.DO. Esistono naturalmente molti formati idonei per questa applicazione; quello che vi presento è uno dei più semplici.

Il nostro file dovrà contenere delle coppie di parole di cui una sarà la traduzione dell' altra. Così ad esempio se volessimo realizzare un dizionario italiano - inglese, potremo memorizzare i dati in questo formato:

Parola italiana [TAB] [TAB] parola inglese [ENTER]

dove [TAB] identifica il tasto tabulatore (freccia a destra). Le tabulazioni sono state incluse per conferire al dizionario una estetica più gradevole. Ad esempio:

Casa	House home
Viaggio	Travel trip
Denaro	Money
Comprare	To buy

ecc....

Vediamo ora come si consulta questo dizionario elettronico con il programma SCHEDL. Per conoscere ad esempio la traduzione di un vocabolo, premere [F1] e digitare la parola di cui si desidera conoscere la traduzione. Sul video verrà visualizzato il vocabolo selezionato con accanto la sua traduzione. Va sottolineato che il dizionario è

bivalente: può essere consultato in italiano o in inglese. In qualsiasi caso viene sempre visualizzato il record così come è stato definito (nel nostro esempio prima il vocabolo italiano e poi quello inglese).

Se si desidera avere un piccolo dizionario da portare con sé, selezionare la funzione *Lfnd* e premere subito dopo [ENTER]: avrete così stampato un piccolo dizionario (in questo caso inserite i record in ordine alfabetico).

14.3 Archivio per diapositive

Tutti gli amanti della fotografia e delle diapositive avranno avuto spesso il problema di non ricordare in quale contenitore si trovasse una determinata diapositiva, o di volere selezionare tutte le diapositive aventi un soggetto ben preciso. Questa applicazione risolve tutti questi problemi.

Per catalogare delle diapositive occorrerà memorizzare un numero di codice, la sua posizione, la data in cui è stata realizzata, il tipo di pellicola e i soggetti contenuti. Pertanto il nostro formato sarà:

Codice posizione data pellicola soggetti

Ad esempio:

A23 C12 09/05/76 Kenya Kodak E64 animali alberi foresta

A24 C12 10/05/76 Kenya Kodak E64 elefante animali alberi cacciatori

F45 C43 23/12/83 New York Agfa traffico strade grattacieli gente

G65 C32 31/12/83 Napoli Agfa fuochi artificiali notte città

ecc....

L' archivio fotografico potrà essere consultato per qualsiasi delle voci che lo costituiscono. Così ad esempio per visualizzare tutte le diapositive che contengono animali, premere [F1] e digitare animali (record 1 e 2). Analogamente se si desidera cercare le diapositive fatte a New York, digitare [F1] New York.

Il catalogo completo dell' archivio potrà essere ottenuto selezionando le funzioni *Find* o *Lfnd* seguite subito dopo da [ENTER].

13.4 Consigli sull' uso di SCHEDL

Elenchiamo qui di seguito i consueti consigli sull' utilizzazione dei diversi programmi applicativi.

La versatilità del programma SCHEDL consente di realizzare molte applicazioni nei campi più svariati. Nei paragrafi due e tre ho illustrato due simpatiche applicazioni, che potranno essere lo spunto da cui partire per realizzarne delle altre. Le possibilità che possiede SCHEDL sono veramente notevoli: saperle cogliere è solo questione di fantasia.

Dal momento che il programma SCHEDL funziona soltanto con il file NOTE.DO, per realizzare altre applicazioni è necessario memorizzare l' indirizzario su nastro e digitare il nuovo file con il nome NOTE.DO. Un' altra possibilità è quella di chiamare i file delle altre applicazioni con nomi diversi, e prima di accedere ad SCHEDL, cambiare da basic (comando NAME - cap.3) i nomi del vecchio fila NOTE e del file con cui si desidera operare. Facendo un esempio se in RAM sono contenuti i file NOTE.DO contenente indirizzi ed il file DIA contenente il ricettario, se si desidera consultare quest' ultimo occorrerà eseguire:

NAME "NOTE.DO" AS "INDIR.DO" e NAME "DIA.DO" AS "NOTE.DO"

In questo modo il programma SCHEDL opererà con l' archivio diapositive. Per ripristinare la situazione di partenza, eseguire due nuove NAME.

Ricordate sempre di salvare su cassetta i vostri file dati: eviterete brutte sorprese.

Appendice A

Interfacce e periferiche

L' M10 dispone di diverse interfacce alle quali possono essere collegate molte periferiche. In questa appendice forniremo spiegazioni e consigli sul loro uso.

Il registratore a cassetta

L' interfaccia per il registratore a cassetta consente di collegare all' M10 qualsiasi registratore che, oltre le normali funzioni di riproduzione, arresto, registrazione, ecc.., abbia le seguenti caratteristiche:

Jack di ingresso per microfono (MIC)

Jack di uscita per cuffia o altoparlante ausiliario (EAR)

e che preferibilmente abbia anche:

Jack REM di controllo motore

Contanastri

Il registratore andrà collegato all' M10 mediante l' apposito cavo fornito con il computer. Il connettore circolare ad otto piedini va inserito nella presa TAPE posta sul retro del calcolatore. Il jack bianco va inserito nel jack EAR del registratore, quello rosso nell' ingresso microfono (MIC), ed infine, se presente, lo spinottino nero va inserito nel jack REM.

I comandi e le istruzioni basic che fanno uso del registratore azionano, tramite l' ingresso REM, il motore del registratore, che può anche essere avviato o arrestato mediante i comandi basic MOTOR ON e MOTOR OFF (consultare cap.3)

Per ottenere ottimi risultati dal vostro registratore, abbiate cura di seguire le raccomandazioni del costruttore per la manutenzione e la pulizia delle testine. Non adoperate cassette di scarsa qualità e non adoperate un volume di riproduzione troppo basso o troppo alto: come dicevano gli antichi "nel mezzo è la virtù".

Interfaccia seriale RS 232-C

A differenza di quanto comunemente detto, i collegamenti seriali RS 232-C hanno ben poco di standard. Quasi ogni periferica possiede infatti delle caratteristiche proprie che talvolta rendono difficoltoso il collegamento con il computer. Il catalogo Olivetti prevede due tipi di cavi seriali: quelli normali e quelli incrociati. I primi si impiegano ad esempio per l' accoppiatore acustico MC10, i secondi invece nel collegamento tra calcolatori.

Nella connessione della periferica all' M10, oltre alla scelta idonea del cavo (che potrà essere fatta con l' aiuto del personale tecnico Olivetti), vanno anche impostati correttamente i parametri di comunicazione, di cui riportiamo per completezza la tabella riassuntiva nella pagina a fronte.

Se effettuata la connessione la periferica non dovesse funzionare, provate, dopo aver attentamente rivisto i parametri di stampa e riletto il manuale d' uso della periferica, ad invertire i fili connessi ai piedini 2 e 3 (il 2 al posto del 3 e viceversa) ed il 4 e il 5 su uno dei due connettori del cavo. Se anche dopo questa operazione la periferica non funzionasse correttamente, rivolgetevi al più vicino concessionario Olivetti.

La stampante parallela

Non ci sono molti suggerimenti da dare per l' uso della stampante parallela con l' M10.

La connessione tra questa periferica ed il computer deve essere effettuata tramite l' apposito cavo della Olivetti. Se una volta collegata la stampante, questa non dovesse funzionare, controllate che sia in "linea", cioè effettivamente collegata al calcolatore. Se la stampante scrive tutto il testo sulla stessa riga, selezionate tramite gli appositi "dip switch" l' autolinefeed, mentre se viceversa la stampante lascia una riga bianca ogni volta che va a capo, disabilitate l' autolinefeed.

Nel caso di ulteriori difficoltà, rivolgetevi al più vicino concessionario Olivetti.

PARAMETRI	VALORI	SIGNIFICATO
Velocita' di comunicazione (v)	M	Modem (300 baud)
	1	75 baud
	2	110 baud
	3	300 baud
	4	600 baud
	5	1200 baud
	6	2400 baud
	7	4800 baud
	8	9600 baud
	9	19200 baud
Lunghezza parola (f)	6	6 bit
	7	7 bit
	8	8 bit
Parita' (p)	O	Dispari
	E	Pari
	N	Nessuna parita'
	I	Ignora parita'
Bit di stop (s)	1	1 bit di stop
	2	2 bit di stop
Linea di stato (x)	E	Abilitata
	D	Disabilitata
Velocita' impulsi	10	10 pps
	20	20 pps

Figura A.1

Appendice B

Codici d' errore

Gli errori su M10 vengono segnalati con codici alfabetici, elencati nel seguito. I numeri indicati a fianco di ciascun codice di errore sono degli interi, riconoscibili dal BASIC, che devono essere utilizzati con l'istruzione ERROR. Per facilitare la ricerca, gli errori sono riportati anche in ordine numerico (a fine appendice).

Sono disponibili 255 codici d'errore, ma non tutti sono attualmente utilizzati dal BASIC. Alcuni numeri d'errore, fra cui i numeri riservati per uso futuro, provocano una segnalazione di codice d'errore non stampabile (UE).

ERRORE CODICE	NUM	DESCRIZIONE
AO	53	File già aperto; è stata emessa una OPEN in modalità sequenziale per un file già aperto; oppure si è tentato di eseguire un comando KILL su un file aperto.
BS	9	Indice fuori dai limiti; si è fatto riferimento a un elemento di matrice o con un indice fuori dalle dimensioni della matrice, o con un numero sbagliato di indici.
BN	51	Errato numero di file; un'istruzione o un comando fa riferimento a un file il cui numero non appartiene all'intervallo specificato con MAXFILE, oppure il file corrispondente non è aperto.
CF	58	File non aperto; il file specificato in un'istruzione PRINT, INPUT o simili non è stato aperto.
CN	17	Non possibile continuare; si è cercato di riprendere l'esecuzione di un programma: a. che si è bloccato per un errore; b. che è stato modificato durante una pausa nell'-esecuzione (ottenibile con <CTRL> + C); c. che non esiste in memoria.
DD	10	Vettore ridimensionato; due o più istruzioni DIM fanno riferimento alla stessa matrice, oppure viene eseguita un'istruzione DIM dopo che per una data matrice erano state assegnate le dimensioni di default.
DS	56	Istruzione immediata in un file; durante il caricamento di un file in formato ISO è stata incontrata una

		istruzione immediata (diretta) e il caricamento si è bloccato.
EF	54	Input oltre la fine del file; si è tentato di eseguire un'istruzione di input su un file già completato oppure su un file vuoto. Usare la funzione EOF per sondare la fine del file ed evitare questo errore.
FC	5	Chiamata funzione illegale; viene passato un argomento fuori dai limiti prescritti ad una funzione numerica o stringa.
		Un errore di questo tipo può succedere, anche quando: a. un indice di matrici risulti negativo o eccezzionalmente grande; b. una funzione LOG abbia un argomento negativo o nullo; c. una funzione SQR abbia un argomento negativo; d. è stato dato un argomento non corretto a una delle seguenti istruzioni:INP, INSTR, LEFT$, MID$, ON ... GOTO, OUT, PEEK, POKE, RIGHT$, SPACE$ o STRING$.
FF	52	File non trovato; un'istruzione LOAD, KILL o OPEN fanno riferimento a un file che non esiste in memoria.
FL	57	Troppi file; è stato tentato di creare un nuovo file, utilizzando un comando SAVE o una OPEN, quando tutte le directory sono piene.
ID	12	Comando immediato illegale; è stata introdotta come linea ad esecuzione immediata una istruzione non consentita in Stato Comandi.
IE	50	Errore interno; si è verificata un'anomalia interna. Segnalare le condizioni d'errore all'Organizzazione di Assistenza.
IO	23	Errore di input/output; si è verificato un errore di input/output durante operazioni su cassetta, stampante o video. Si tratta di un errore non recuperabile da BASIC.
LS	15	Stringa troppo lunga; si è tentato di creare una stringa con più di 255 caratteri.
MO	22	Operando mancante; una espressione contiene un operatore non seguito da operando.
NF	1	NEXT senza FOR; una variabile in un'istruzione NEXT non corrisponde ad alcuna variabile di una istruzione FOR eseguita in precedenza.
NM	55	Nome file errato; è stato usato un formato non consentito per specificare un nome di file, in uno dei comandi LOAD, SAVE, KILL, NAME, etc.

NR	19	Manca l'istruzione RESUME; il controllo è stato passato a una routine di gestione degli errori che non contiene l'istruzione RESUME.
OD	4	Dati insufficienti; viene eseguita un'istruzione READ quando nel programma non sono rimaste istruzioni DATA con dati da leggere.
OM	7	Memoria insufficiente; un programma è troppo grande, ha troppi cicli o subroutine, o troppe variabili, o espressioni troppo complesse.
OS	14	Spazio per stringhe insufficiente; le variabili stringa hanno provocato il superamento della memoria libera rimanente (il BASIC alloca lo spazio per le stringhe in modo dinamico, finché non succede un errore di questo tipo).
OV	6	Overflow; il risultato di un calcolo è troppo grande per essere rappresentato con il formato BASIC di rappresentazione dei numeri.
RG	3	RETURN senza GOSUB; viene incontrata un'istruzione RETURN alla quale non corrisponde alcuna precedente istruzione GOSUB.
RW	20	RESUME senza routine d'errore; viene incontrata un'istruzione RESUME senza prima essere entrati in una routine di gestione degli errori.
SN	2	Errore sintattico; una riga programma contiene una sequenza non corretta di caratteri, ad esempio parentesi non accoppiate, parole chiave con errori ortografici, punteggiatura non corretta.
ST	16	Espressione stringa troppo complessa; un'espressione stringa è troppo lunga o troppo complessa. E' opportuno dividerla in più espressioni semplici.
TM	13	Tipo di variabile errato; un nome di una variabile stringa è stato associato ad un valore numerico o viceversa. Oppure una funzione che richiede argomento numerico ha ricevuto un argomento stringa o viceversa.
UE	21	Errore non stampabile; non è disponibile un messaggio per la condizione d'errore trovata.
UE	24-29	Errore non stampabile; questi codici non sono stati definiti, ma sono riservati ad utilizzi futuri del BASIC.
UE	59-255	Errore non stampabile; questi codici non sono definiti, e possono essere utilizzati da programmi utente.
UL	8	Linea non definita; un'istruzione DELETE, GOTO, GOSUB o IF ... ELSE fa riferimento a una linea inesistente.

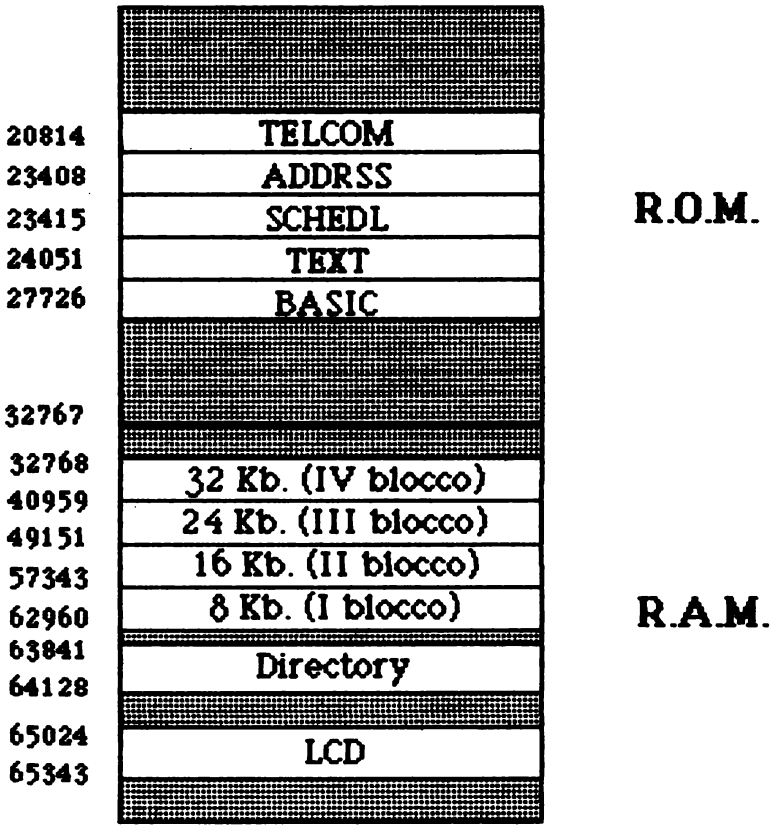
/0	11	Divisione per zero; viene incontrata una divisione per zero, oppure si ha un elevamento a potenza dove il valore della base è zero e l'esponente è negativo.

La tabella che segue elenca i codici d'errore in ordine numerico.

1	NF	NEXT senza FOR
2	SN	Errore sintattico
3	RG	RETURN senza GOSUB
4	OD	Dati insufficienti
5	FC	Chiamata funzione illegale
6	OV	Overflow
7	OM	Memoria insufficiente
8	UL	Linea non definita
9	BS	Indice fuori dai limiti
10	DD	Vettore ridimensionato
11	/0	Divisione per zero
12	ID	Comando immediato illegale
13	TM	Tipo di variabile errato
14	OS	Spazio per stringhe insufficiente
15	LS	Stringa troppo lunga
16	ST	Espressione stringa troppo complessa
17	CN	Non possibile continuare
19	NR	Manca l'istruzione RESUME
20	RW	RESUME senza routine d'errore
21	UE	Errore non stampabile
22	MO	Operando mancante
23	IO	Errore di input/output
24-49	UE	Errore non stampabile
50	IE	Errore interno
51	BN	Errato numero di file
52	FF	File non trovato
53	AO	File già aperto
54	EF	Input oltre la fine del file
55	NM	Nome file errato
56	DS	Istruzione immediata in un file
57	FL	Troppi file
58	CF	File non aperto
59-255	UE	Errore non stampabile

Appendice C

La mappa di memoria



Appendice D

I codici ASCII

COD. ASCII	EQUIVALENTE IN TASTIERA	FUNZIONE IN: BASIC	TEXT	TELCOM
00	<CTRL> + @	nessuna	nessuna	nessuna
01	<CTRL> + A	nessuna	Curs. a inizio parola precedente	nessuna
02	<CTRL> + B	nessuna	Curs. a ultima riga del display	nessuna
03	<CTRL> + C	Break *	Annulla Find, Load, Save, Select, Print	nessuna
04	<CTRL> + D	nessuna	Curs. a inizio parola seguente	nessuna
05	<CTRL> + E	nessuna	Curs. una riga in alto	nessuna
06	<CTRL> + F	nessuna	Curs. uno spazio a destra	nessuna
07	<CTRL> + G	Cicalino	Save (salva)	Cicalino
08	<CTRL> + H	Cancella carattere	Cancella carattere	Cancella carattere
09	<CTRL> + I	Tabula *	Tabula	Tabula
10	<CTRL> + J	Interlinea	nessuna	Interlinea
11	<CTRL> + K	nessuna	nessuna	Interlinea
12	<CTRL> + L	Cancella schermo	Select (scegli)	Cancella schermo
13	<CTRL> + M	Ritorno a capo	Ritorno a capo	Ritorno a capo
14	<CTRL> + N	nessuna	Find (trova)	nessuna
15	<CTRL> + O	nessuna	Copy (copia)	nessuna
16	<CTRL> + P	nessuna	P (se premuto due volte)	nessuna
17	<CTRL> + Q	nessuna	Curs. a inizio riga	XON (linea abilitata)
18	<CTRL> + R	nessuna	Curs. a fine riga	nessuna
19	<CTRL> + S	Pause *	Curs. uno spazio a sinistra	XOFF (linea disabilitata)

COD. ASCII	EQUIVALENTE IN TASTIERA	FUNZIONE IN: BASIC	TEXT	TELCOM
20	<CTRL> + T	nessuna	Curs. a prima riga del display	nessuna
21	<CTRL> + U	Cancella linea *	Cut (taglia)	Cancella riga attuale
22	<CTRL> + V	nessuna	Load (carica)	nessuna
23	<CTRL> + W	nessuna	Curs. a inizio file	nessuna
24	<CTRL> + X	Cancella linea *	Curs. una riga in basso	Cancella riga attuale
25	<CTRL> + Y	nessuna	Stampa il file	nessuna
26	<CTRL> + Z	nessuna	Curs. a fine file	nessuna
27	<CTRL> + 0	Escape	Salva il file e vai al Menù (se premuto due volte)	Escape
28	< ← >	nessuna	Curs. uno spazio a destra	Curs. uno spazio a destra
29	< → >	Cancella carattere	Curs. uno spazio a sinistra	Curs. uno spazio a sinistra
30	< ↑ >	nessuna	Curs. una riga in alto	Curs. una riga in alto
31	< ↓ >	nessuna	Curs. una riga in basso	Curs. una riga in basso

CARATTERI GRAFICI

COD. ASCII	CARATTERE GRAFICO	CARATTERE IN TASTIERA	COD. ASCII	CARATTERE GRAFICO	CARATTERE IN TASTIERA
32	Spazio	<SPAZIO>	57	9	9
33	!	!	58	:	:
34	"	"	59	;	;
35	£	£	60	<	<
36	$	$	61	=	=
37	%	%	62	>	>
38	&	&	63	?	?
39	'	'	64	§	§
40	(	(	65	A	<SHIFT> + A
41	)	)	66	B	<SHIFT> + B
42	*	*	67	C	<SHIFT> + C
43	+	+	68	D	<SHIFT> + D
44	,	,	69	E	<SHIFT> + E
45	-	-	70	F	<SHIFT> + F
46	.	.	71	G	<SHIFT> + G
47	/	/	72	H	<SHIFT> + H
48	0	0	73	I	<SHIFT> + I
49	1	1	74	J	<SHIFT> + J
50	2	2	75	K	<SHIFT> + K
51	3	3	76	L	<SHIFT> + L
52	4	4	77	M	<SHIFT> + M
53	5	5	78	N	<SHIFT> + N
54	6	6	79	O	<SHIFT> + O
55	7	7	80	P	<SHIFT> + P
56	8	8	81	Q	<SHIFT> + Q

COD. ASCII	CARATTERE GRAFICO	CARATTERE IN TASTIERA	COD. ASCII	CARATTERE GRAFICO	CARATTERE IN TASTIERA
82	R	<SHIFT> + R	107	k	K
83	S	<SHIFT> + S	108	l	L
84	T	<SHIFT> + T	109	m	M
85	U	<SHIFT> + U	110	n	N
86	V	<SHIFT> + V	111	o	O
87	W	<SHIFT> + W	112	p	P
88	X	<SHIFT> + X	113	q	Q
89	Y	<SHIFT> + Y	114	r	R
90	Z	<SHIFT> + Z	115	s	S
91	°	°	116	t	T
92	ç	ç	117	u	U
93	é	é	118	v	V
94	^	^	119	w	W
95	_	_	120	x	X
96	ù	ù	121	y	Y
97	a	A	122	z	Z
98	b	B	123	à	à
99	c	C	124	ò	ò
100	d	D	125	è	è
101	e	E	126	ì	ì
102	f	F	127-159	(non usato)	
103	g	G	160	◀	<ENTER>
104	h	H	161	Ä	<G + S> + &
105	i	I	162	Ö	<G + S> + 1
106	j	J	163	Ü	<G + S> + 2

COD. ASCII	CARATTERE GRAFICO	CARATTERE IN TASTIERA	COD. ASCII	CARATTERE GRAFICO	CARATTERE IN TASTIERA
164	Å	<G + S> + 3	189	³	<G + S> + F
165	Æ	<G + S> + 4	190	∫	<G + S> + G
166	Ø	<G + S> + 5	191	#	<G + S> + H
167	Ã	<G + S> + 6	192	≠	<G + S> + J
168	Õ	<G + S> + 7	193	≤	<G + S> + K
169	Ç	<G + S> + 8	194	±	<G + S> + L
170	Ñ	<G + S> + 9	195	¼	<G + S> + M
171	←	<G + S> + 0	196	½	<G + S> + %
172	→	<G + S> + °	197	☎	<G + S> + >
173	↑	<G + S> + +	198	☗	<G + S> + W
174	↓	<G + S> + §	199	[x	<G + S> + X
175	△	<G + S> + Q	200	±	<G + S> + C
176	µ	<G + S> + Z	201	⁜	<G + S> + V
177	π	<G + S> + E	202	£	<G + S> + B
178	Σ	<G + S> + R	203	¥	<G + S> + N
179	Ω	<G + S> + T	204	¢	<G + S> + ?
180	√	<G + S> + Y	205	¤	<G + S> + .
181	≤	<G + S> + U	206	@	<G + S> + /
182	≥	<G + S> + I	207	°	<G + S> + !
183	∞	<G + S> + O	208	□	<GRPH> + õ
184	[	<G + S> + P	209	ä	<GRPH> + $
185	]	<G + S> + =	210	ö	<GRPH> + £
186	×	<G + S> + A	211	ü	<GRPH> + é
187	÷	<G + S> + S	212	å	<GRPH> + "
188	²	<G + S> + D	213	æ	<GRPH> + '

COD. ASCII	CARATTERE GRAFICO	CARATTERE IN TASTIERA	COD ASCII	CARATTERE GRAFICO	CARATTERE IN TASTIERA
214	ø	<GRPH> + (	235	ô	<GRPH> + S
215	ă	<GRPH> + _	236	ù	<GRPH> + D
216	õ	<GRPH> + è	237	û	<GRPH> + F
217	ç	<GRPH> + ^	238	§	<GRPH> + G
218	ñ	<GRPH> + ç	239	|	<GRPH> + H
219	ì	<GRPH> + à	240	\	<GRPH> + J
220	¿	<GRPH> +)	241	º	<GRPH> + K
221	ij	<GRPH> + -	242	ª	<GRPH> + L
222	ß	<GRPH> + *	243	■	<GRPH> + M
223	à	<GRPH> + Q	244	▨	<GRPH> + ù
224	â	<GRPH> + Z	245	┌	<GRPH> + <
225	é	<GRPH> + E	246	┐	<GRPH> + W
226	è	<GRPH> + R	247	└	<GRPH> + X
227	ê	<GRPH> + T	248	┘	<GRPH> + C
228	ë	<GRPH> + Y	249	┃	<GRPH> + V
229	ì	<GRPH> + U	250	━	<GRPH> + B
230	î	<GRPH> + I	251	┃	<GRPH> + N
231	ï	<GRPH> + O	252	▁	<GRPH> + ,
232	{	<GRPH> + P	253	┌┐	<GRPH> + ;
233	}	<GRPH> + ì	254	└┘	<GRPH> + :
234	ò	<GRPH> + A	255	(non usato)	

AVVERTENZA: per ragioni di spazio, in questa tabella si è modificata la notazione standard; <G + S> equivale a <GRPH + SHIFT>

Una guida all'uso, ma anche una precisa fonte di idee e di possibili applicazioni.
Il libro è diviso in 2 sezioni: nella prima sono descritti i comandi e le istruzioni del linguaggio BASIC, classificati in gruppi funzionali, con un criterio che ne semplifica l'apprendimento e la consultazione.
Nella seconda parte vengono presentati i programmi applicativi integrati nel calcolatore, che ne fanno di volta in volta una versatile macchina da scrivere, un'agenda, un indirizzario o un terminale di un sistema remoto.

L. 18.000 Cod. 401B ISBN 88-7056-067-8

OLIVETTI M10: guida all'uso

Massimo Mangia

GRUPPO EDITORIALE JACKSON