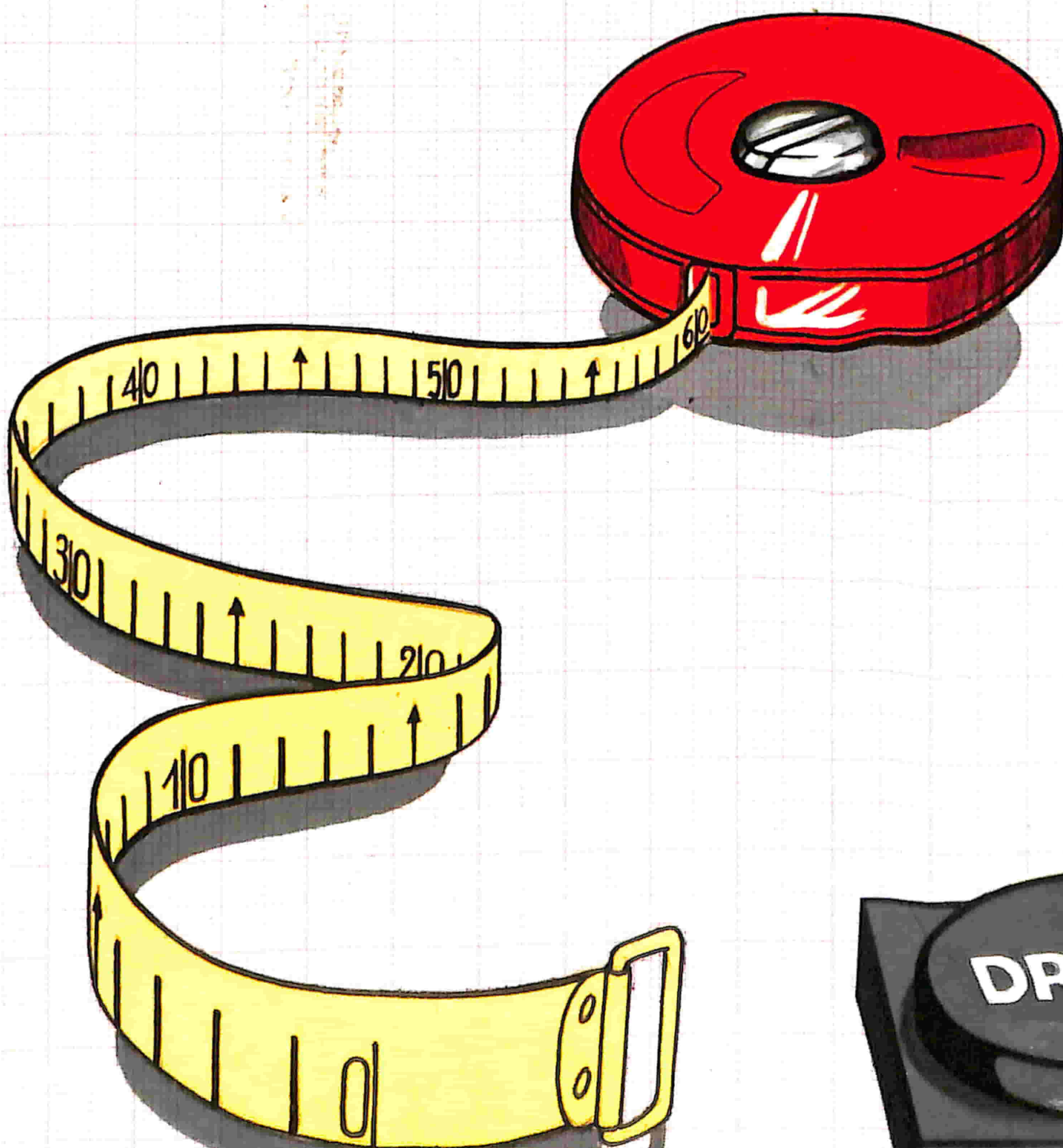


Patrizio Quintili

Basic per i geometri



Il piacere del computer





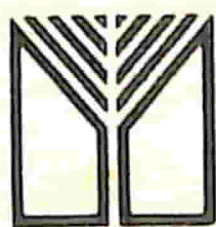
Il piacere del computer

Il piacere del computer

- 1 *Tom Rugg e Phil Feldman* 32 programmi con il PET
- 2 *Rich Didday* Intervista sul personal computer, hardware
- 3 *Tom Rugg e Phil Feldman* 32 programmi con l'Apple
- 4 *Ken Knecht* Microsoft Basic
- 5 *Paul M. Chirlian* Pascal
- 6 *Tom Rugg e Phil Feldman* 32 programmi con il TRS-80
- 7 *Rich Didday* Intervista sul personal computer, software
- 8 *Herbert D. Peckham* Imparate il Basic con il PET/CBM
- 9 *Karl Townsend e Merl Miller* Il personal computer come professione
- 10 *Karen Billings e David Moursund* Te ne intendi di computer?
- 11 *Thomas Dwyer e Margot Critchfield* Il Basic e il personal computer, uno: introduzione
- 12 *Don Inman e Kurt Inman* Imparate il linguaggio dell'Apple
- 13 *Thomas Dwyer e Margot Critchfield* Il Basic e il personal computer, due: applicazioni
- 14 *Luigi Pierro* Il manuale del CP/M
- 15 *Carlo Sintini* A scuola con il PET/CBM
- 16 *David Johnson-Davies* Il manuale dell'Atom
- 17 *David E. Schultz* Il libro del Commodore VIC 20
- 18 *Jim Huffman e Robert Bruce* Il "debug" nei personal computer
- 19 *John M. Nevison* Programmazione in Basic per l'uomo d'affari
- 20 *Mark Harrison* Imparate il Basic con lo ZX81
- 21 *Ronald W. Anderson* Dal Basic al Pascal
- 22 *Herbert D. Peckham* Imparate il Basic con il Texas TI 99/44
- 23 *Sergio Borsani* A scuola con il Texas TI 99/4A
- 24 *Jerry Willis e Deborrah Willis* Come usare il Commodore 64
- 25 *Mark Harrison* Imparate il Basic con lo Spectrum
- 26 *Carlo Sintini e Costantino Mustacchio* A scuola con il Commodore 64
- 27 *David A. Lien* Imparate il Basic con l'IBM PC
- 28 *Ken Tracton* Introduzione al Lisp
- 29 *Fabio Mavaracchio* Programmi in Basic per l'elettronica
- 30 *Ian Stewart e Robin Jones* Il linguaggio macchina dello Spectrum
- 31 *Tom Rugg, Phil Feldman e Clarence S. Wilson* 32 programmi per il VIC 20
- 32 *Merl Miller e Mary A. Myers* Introduzione all'Apple Macintosh
- 33 *Stam Krute* Grafica e suoni con il Commodore 64
- 34 *Jerry Willis e William Manning* Come usare l'IBM PCjr
- 35 *Tom Rugg e Phil Feldmann* 32 programmi con il Commodore 64
- 36 *Sergio Borsani* Matematica e geometria con il Commodore 64
- 37 *David Laine* ZX Spectrum: tecniche avanzate di linguaggio macchina
- 38 *Salvatore Marseglia* Chimica con il pocket computer

Patrizio Quintili

Basic per i geometri



franco muzzio & c. editore

*Dedico questo libro ai miei ex studenti sulle cui spalle (pelli) sono
diventato un insegnante.*

Prima edizione: aprile 1985
ISBN 88-7021-282-3

© 1985 franco muzzio & c. editore spa
Via Makallè 73, 35138 Padova, tel. 049/661147-661873
Tutti i diritti sono riservati.

Indice

- 7 **Presentazione**
- 9 **Introduzione**
 - Organizzazione del libro Avvertenza
- 13 **Primo livello**
- 14 **Hardware**
 - Com'è fatto un sistema di elaborazione dati Che cosa bisogna sapere sul funzionamento di un personal computer Struttura interna Funzionamento Aritmetica binaria
- 22 **Usiamo il computer**
 - La tastiera Le correzioni Caricare e far girare un programma Registrare un programma Cose importanti di Basic Usiamo il computer
- 39 **Secondo livello**
- 40 **Programmazione**
 - Direct mode - program mode Programmi e flow-chart Problemi decisionali e salti Iterazione Conclusione
- 52 **Impariamo il Basic risolvendo problemi di costruzione**
 - Calcolo reazioni di una trave isostatica Miglioriamo l'input e l'output Implementiamo il calcolo delle reazioni
- 62 **Programmi completi**
 - Programma 5.1: Calcolo interesse composto Programma 5.2: Valore potenziale di un fondo Programma 5.3: Teorema dei seni Programma 5.4: Intersezione in avanti Programma 5.5: Reazioni travi appoggiate per carichi qualsiasi Programma 5.6: Calcolo sollecitazioni travi appoggiate
- 81 **Terzo livello**
- 83 **Menù e subroutine**
 - Menù Subroutine Calcolo delle annualità

- 94 **La stampante**
Direct mode Program mode Il programma completo
 - 102 **Programmi completi**
Progetto di sezioni rettangolari in C.A. Verifica di sezioni rettangolari in C.A. Muri di sostegno e gravità Poligonale aperta planoaltimetrica
 - 130 **Soluzione degli esercizi**
 - 136 **Glossario delle sigle e dei termini inglesi usati nel testo**
-

APPENDICI

- 141 **Sintesi del Basic e delle tecniche di programmazione**
Basic Programmazione
- 145 **Comandi e funzioni Basic**
Comandi Funzioni
- 161 **Codici ASCII e CHR\$**
- 163 **Abbreviazioni per le parole chiave del Basic**
- 165 **Messaggi di errore**
- 169 **Bibliografia**

Presentazione

Questo libro si rivolge sia agli studenti allievi-geometri che ai geometri professionisti che sono interessati all'uso del computer per il proprio lavoro.

Presenta numerosi programmi direttamente applicativi nel campo delle costruzioni, topografia e estimo.

Per leggerlo non sono necessarie conoscenze preliminari sui computer e sulla programmazione, ma solo conoscenze dei problemi tecnico-professionali relativi al lavoro del geometra.

Il testo è organizzato su tre livelli di difficoltà crescente: nel primo livello si insegna come usare programmi prodotti da altri e come apportarvi piccole modifiche; nel secondo livello viene insegnato il linguaggio Basic attraverso applicazioni nel campo della materia di costruzioni; nel terzo livello si insegna a costruire programmi complessi, con uso della stampante. Vengono riportati inoltre una serie di programmi completi di costruzioni, estimo e topografia, listati e spiegati.

Ringraziamenti

Devo ringraziare gli amici e colleghi ing. Augusto D'Urso, per l'aiuto fornitomi relativamente alla parte topografica, e ing. Roberto De Cristofaro, per la revisione del testo e le necessarie critiche didattiche, oltre, naturalmente, a Virginio Sala delle edizioni Muzzio.

Richiesta ai lettori

Spero che questo libro vi sia di aiuto nella vostra professione. Come tutte le opere umane, anche la mia conterrà comunque delle imperfezioni e avrà delle manchevolezze. Sarò grato quindi a tutti coloro che vorranno segnalarmi qualunque eventuale errore o vorranno proporre suggerimenti e idee per migliorare ulteriormente questo testo comunicandoli all'Editore Franco Muzzio & C., Via Makallè 73, Padova.
Grazie.

Introduzione

L'uso del personal computer si sta ormai generalizzando e fra pochi anni sarà uno strumento di lavoro indispensabile sia per lo studente che per il geometra professionista.

Le possibili applicazioni del computer nel campo di lavoro del geometra sono molteplici e vanno dai calcoli statici a quelli di topografia, dall'estimo alla contabilità, dalla gestione di un archivio alla stampa di lettere e relazioni, fino alla redazione di disegni esecutivi.

Naturalmente non basta acquistare un computer perché tutti i problemi siano risolti, bisogna anche acquistare o produrre in proprio i programmi (il "software") che istruiscono il computer a risolvere i vostri specifici problemi.

Quando vi accingete a redigere un progetto, sia di un edificio, sia di una strada, il primo passo è quello della ricerca della documentazione. Cercherete cioè in libri, riviste o nel vostro archivio personale progetti simili a quello che dovete realizzare per trarne indicazioni e schemi di lavoro per la stesura del vostro progetto specifico.

Così per il software, a parte alcuni problemi di calcolo standard, difficilmente troverete il programma che soddisfi appieno le vostre necessità. Ma potete prendere dei programmi già fatti e modificarli per il vostro uso specifico, o, non trovandone e volendo risparmiare, potete costruirveli direttamente da soli.

Lo scopo di questo libro è proprio quello di insegnarvi a modificare dei programmi già esistenti per adattarli al vostro uso e insegnarvi anche a realizzare da soli programmi di una certa complessità che siano direttamente applicabili sia nel campo della scuola sia in quello della professione.

ORGANIZZAZIONE DEL LIBRO

Per tener conto dei possibili diversi usi del computer ho organizzato questo libro su tre livelli di difficoltà crescente:

Primo livello. Parte introduttiva sui computer, primi elementi di linguaggio Basic, come caricare e registrare un programma, come farlo girare. Questo livello metterà il lettore in grado di usare software prodotto da altri e di apportare a tale software piccole modifiche per adattarlo alle proprie necessità.

Secondo livello. Elementi di programmazione, avviamento al Basic attraverso la risoluzione di semplici problemi di costruzioni. Sono infine riportati alcuni programmi completi di costruzioni, topografia e estimo. Lo studio di questo livello metterà in grado il lettore di produrre da solo programmi di media complessità.

Terzo livello. Produzione di programmi complessi, uso della stampante. Sono riportati anche qui alcuni programmi completi di livello professionale su problemi di calcolo che si presentano frequentemente.

Il lettore che non ha esperienza di programmazione in Basic dovrà naturalmente iniziare dal primo livello e potrà anche fermarsi a questo, se non intende produrre da sé del software. Chi invece ha già esperienza di programmazione può passare direttamente al secondo livello e chi è esperto può dedicarsi subito al terzo livello.

Di tutti i programmi vengono forniti il listato completo, la descrizione e la spiegazione, oltre a idee e suggerimenti per modifiche e miglioramenti.

Alla fine del volume sono riportate in appendice schede sulle frasi e i comandi Basic, la tabella dei codici ASCII e un glossario dei termini tecnici inglesi usati nel testo.

AVVERTENZA

Esistono in commercio numerosi tipi di personal computer prodotti da diverse case. Computer prodotti da case diverse non sono in genere compatibili tra di loro. Non si è cioè ancora giunti a una standardizzazione che permetta di usare programmi prodotti per un determinato computer su altri elaboratori.

Il linguaggio usato per comunicare con la macchina è in genere, nei personal computer, il Basic. Anche del Basic esistono tuttavia diverse

versioni, che variano leggermente da una casa, e da una macchina, all'altra.

Dato che questo libro si propone finalità pratiche e vuole presentare programmi che siano direttamente operativi, non ho potuto prescindere quindi dal fare riferimento a una specifica macchina reale.

I programmi riportati nel testo sono stati prodotti su un Commodore 64, che ho scelto per la sua discreta diffusione. Tali programmi possono comunque essere usati, cioè "girano", sia su tutti i computer della Commodore, sia sulla maggior parte dei personal computer in circolazione senza particolari modifiche. La versione del Basic usata nel testo, la V. 2 del Commodore Basic, è infatti una versione non troppo sofisticata e le istruzioni usate sono tra le più comuni.

Se qualcuno dei lettori incontrasse difficoltà nel far girare sul proprio computer i programmi riportati nel testo, basterà che consulti un po' il manuale fornito con la sua macchina e sarà in grado di effettuare le piccole modifiche necessarie senza eccessiva difficoltà.

Un'ultima cosa: per lavorare con un computer non è necessario conoscere la lingua inglese, anche se sarebbe tuttavia consigliabile. È comunque indispensabile la conoscenza, e più pratico l'uso, di alcuni termini inglesi. Per facilitare chi non conosce tale lingua ho quindi inserito nel testo la traduzione dei vocaboli inglesi che a mano a mano si incontrano, scrivendoli, almeno le prime volte, in corsivo. Tutte le parole inglesi e le sigle usate nel testo sono comunque riportate, con la relativa traduzione, nel glossario in fondo al volume.

Primo livello

Cosa si insegna in questo livello

Prima di incominciare a lavorare con il computer bisogna conoscere qualcosa sul suo funzionamento: a ciò provvede il primo capitolo sull'hardware. Vedremo dapprima come è costituito un semplice sistema di elaborazione dati. Indagheremo quindi un po' sulla struttura interna e sul funzionamento di un personal computer.

Nel secondo capitolo viene invece insegnato come lavorare con programmi già prodotti da altri, viene cioè insegnato a fare l'"operatore", e vengono spiegate alcune cose da conoscere per poter studiare il linguaggio BASIC nella seconda parte del libro.

Cosa si insegna in questo livello

Prima di incominciare a lavorare con il computer bisogna conoscere qualcosa sul suo funzionamento: a ciò provvede il primo capitolo sull'hardware. Vedremo dapprima come è costituito un semplice sistema di elaborazione dati. Indagheremo quindi un po' sulla struttura interna e sul funzionamento di un personal computer.

Nel secondo capitolo viene invece insegnato come lavorare con programmi già prodotti da altri, viene cioè insegnato a fare l'"operatore", e vengono spiegate alcune cose da conoscere per poter studiare il linguaggio BASIC nella seconda parte del libro.

Hardware

COME È FATTO UN SISTEMA DI ELABORAZIONE DATI

Chi incomincia a leggere questo libro ha già, probabilmente, qualche idea e qualche conoscenza sul mondo dei computer. È bene tuttavia fare una breve panoramica sulle macchine che dobbiamo usare per fissare alcuni concetti di base.

In questo capitolo vedremo quali sono le apparecchiature elettroniche, elettromeccaniche e meccaniche che costituiscono l'*hardware* (*hard* = duro, *ware* = merce) di un sistema di elaborazione dati. Il *software* (*soft* = molle, tenero) è invece l'insieme dei programmi di istruzione della macchina che le permette di sapere come trattare i dati che vengono inseriti.

L'hardware di un sistema di elaborazione dati con personal computer è costituito da quattro tipi di apparecchiature: 1) le *input devices* (*in* = dentro, *to put* = mettere, *device* = apparecchiatura), 2) il *computer*, 3) le *output devices* (*out* = fuori), 4) le *memorie esterne* o di massa.

I dati vengono introdotti per mezzo delle apparecchiature d'ingresso (*input devices*), elaborati dal computer ed escono, in forma comprensibile all'utente, sulle apparecchiature d'uscita (*output devices*). Le memorie esterne sono usate per conservare programmi (il software) o dati.

Per fissare le idee, vediamo ora due configurazioni di un sistema di elaborazione dati realizzate con un Commodore 64 e un Commodore 8032.

In figura 1.2 è riportata la configurazione minima per un Commodore 64. Sono tre oggetti:

- il Commodore 64: costituito dal computer vero e proprio e da una tastiera ASCII;
- un terminale video: una normale televisione, anche in bianco e nero, o un TV monitor;
- un registratore a cassette.

La tastiera ASCII, simile alla tastiera di una macchina da scrivere elettrica, costituisce il dispositivo d'ingresso. Su questa tastiera scriveremo i nostri programmi. Il terminale video è invece il dispositivo d'uscita, cioè l'apparecchiatura su cui appariranno visualizzati i dati elaborati dal computer.

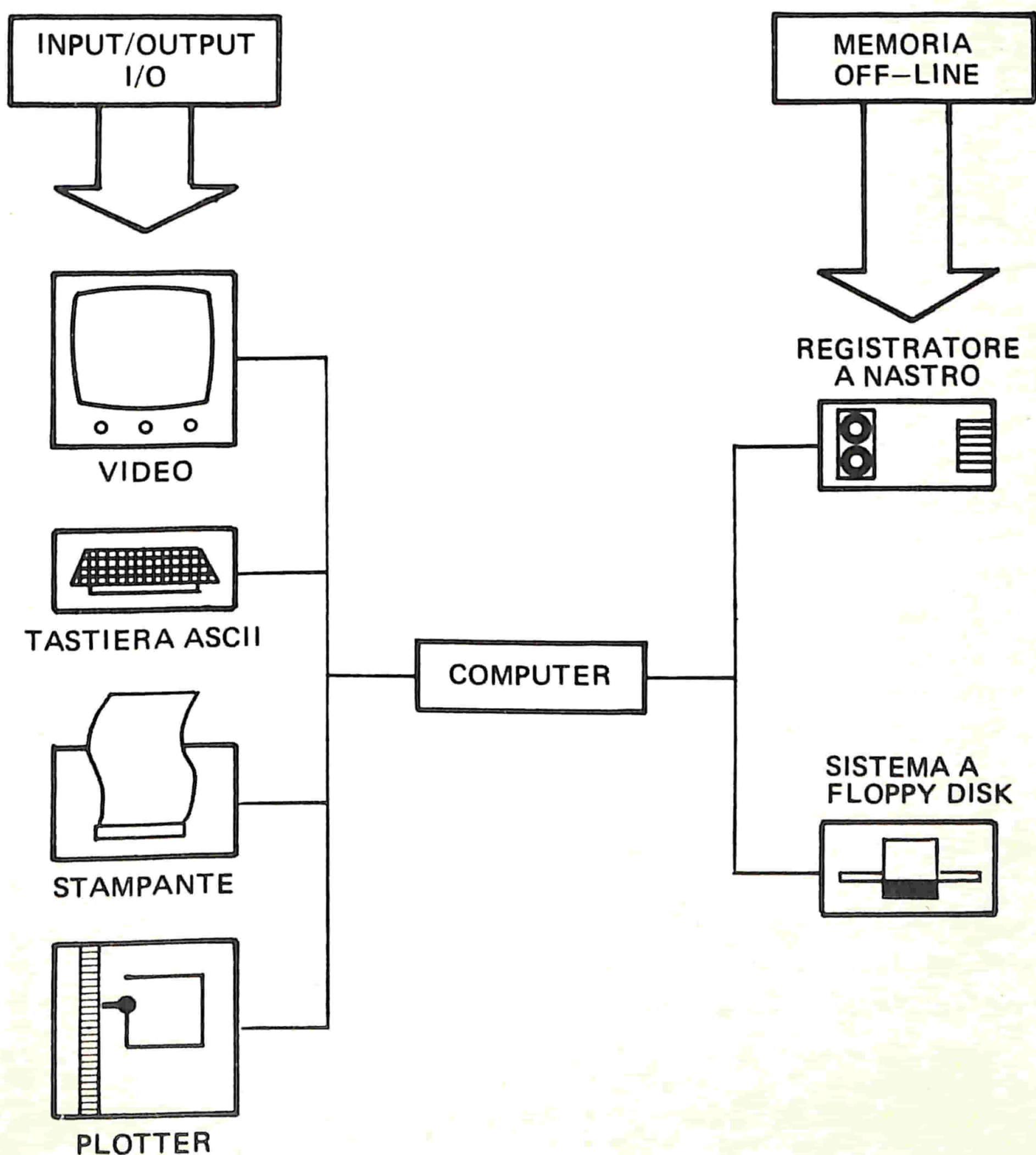


Fig. 1.1 Hardware di un sistema di elaborazione dati con personal computer

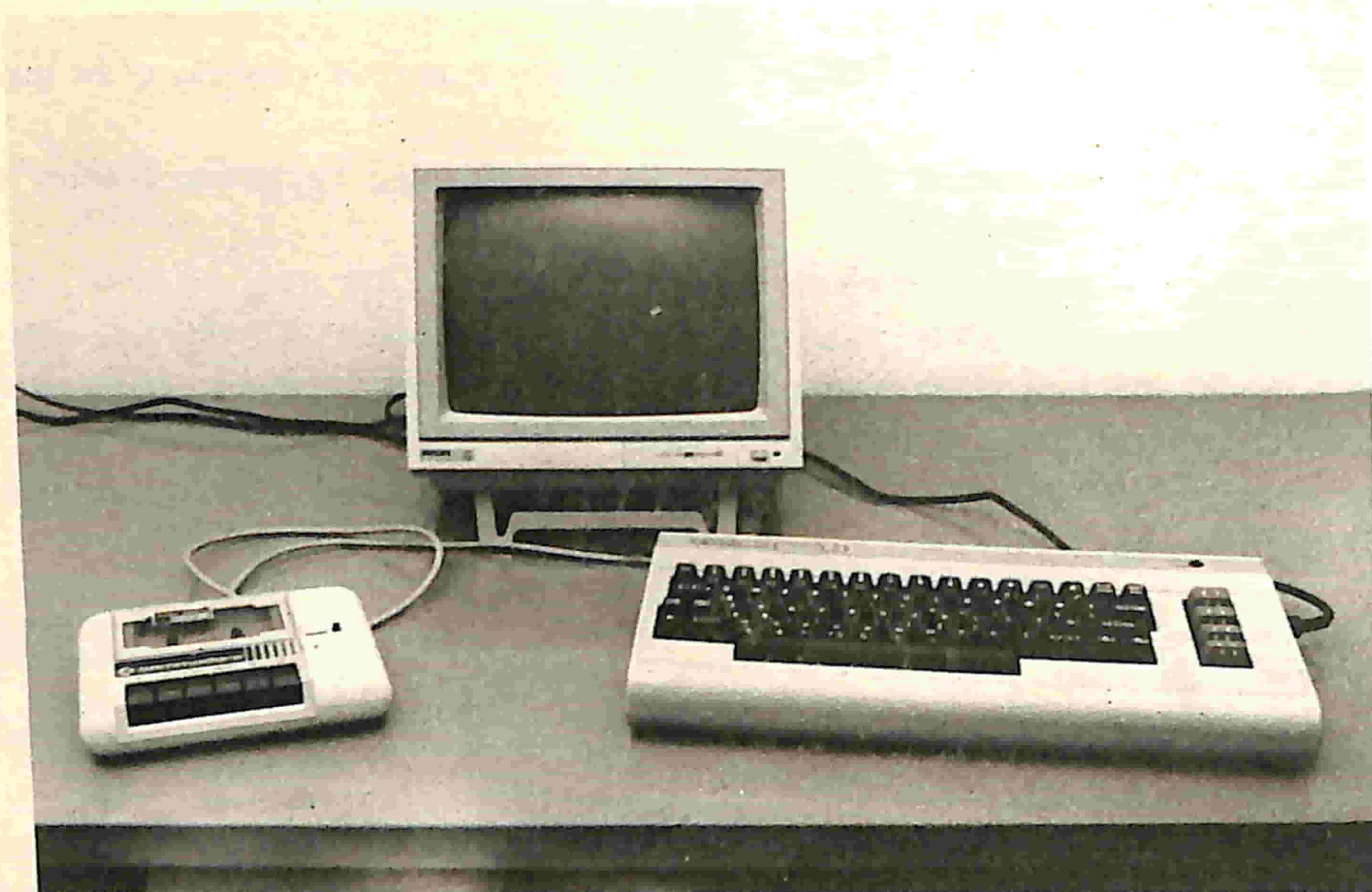


Fig. 1.2 Configurazione minima per un Commodore 64

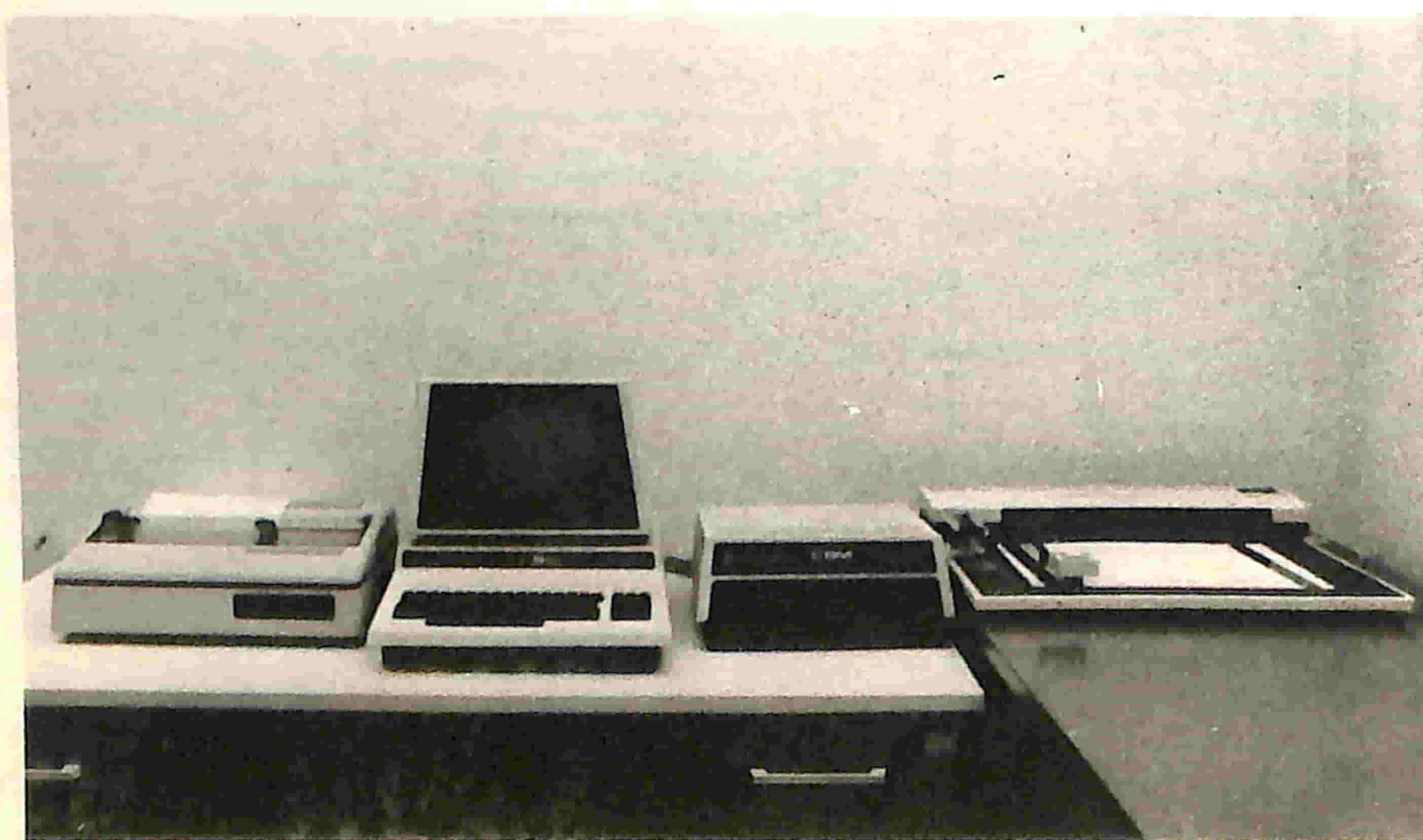


Fig. 1.3 Configurazione completa per il Commodore 8032

Il registratore a cassette, infine, costituisce la memoria esterna. Ce ne serviremo essenzialmente per registrare i programmi già fatti per poterli riutilizzare in seguito.

La seconda configurazione, che si può considerare massima per il Commodore 8032, è riportata in figura 1.3.

Gli oggetti sono quattro:

- il Commodore 8032: costituito dal computer, una tastiera ASCII e un monitor TV;
- un *floppy driver*: è un lettore/registratore di speciali dischetti detti *floppy disk* (*floppy* = flessibile);
- una stampante;
- un *plotter*: è un terminale grafico, per eseguire disegni (*to plot* = disegnare).

Il dispositivo di ingresso è costituito anche qui dalla tastiera ASCII. Il monitor, la stampante e il plotter sono i dispositivi di uscita. Useremo solo il monitor quando non avremo bisogno di conservare i risultati delle nostre elaborazioni; quando invece vorremo conservare i risultati del nostro lavoro sulla carta, useremo la stampante o il plotter.

Per la memoria di massa abbiamo il floppy driver, che registra su dischetti invece che su cassette. Il floppy driver è più veloce e pratico del registratore, ma, naturalmente, costa di più.

Oltre alle due configurazioni che abbiamo visto ne sono possibili altre: il floppy può essere sostituito da un registratore e viceversa, oppure al Commodore 64 possono essere aggiunti la stampante e il plotter.

Ognuno di voi, con un po' di buon senso, sarà in grado di decidere quale sia la configurazione che fa di più al suo caso, tenendo presenti le proprie disponibilità finanziarie e l'uso che intende fare delle attrezzature che acquista. Come consiglio, vi direi di comportarvi come per l'acquisto di un impianto stereo, comperando all'inizio l'essenziale e, a mano a mano che ne sentite la necessità, le altre apparecchiature che vi occorrono.

CHE COSA BISOGNA SAPERE SUL FUNZIONAMENTO DI UN PERSONAL COMPUTER

Per lavorare usando software prodotto da altri, o anche per costruire da soli programmi abbastanza complessi, come quelli che troverete in questo libro, non è in realtà necessario conoscere molto sul funzionamento interno di un personal computer.

Tuttavia, chi vuole saperne di più sullo strumento che intende usare e comprendere meglio quello che verrà spiegato più avanti, è bene legga i tre paragrafi che seguono, ritornandoci magari sopra più tardi se alla prima lettura non tutto gli sarà completamente chiaro.

STRUTTURA INTERNA

Ci riferiamo, nel seguito, al computer Commodore 64, ma la maggior parte delle indicazioni che verranno date è valida anche per altri personal computer.

L'elemento più importante dell'hardware interno del computer è il *microprocessore*. Il Commodore 64 usa il microprocessore 6510 della MOS Technology.

Il microprocessore effettua le elaborazioni, gestisce le operazioni di ingresso e uscita dei dati dalle varie unità periferiche (video, registratore, floppy driver, stampante, ecc.) e sovrintende al funzionamento di tutto il sistema.

L'altro elemento fondamentale della struttura interna di un computer, oltre al microprocessore, è la memoria interna.

Esistono due tipi di memorie interne: la ROM e la RAM.

La ROM (*Read Only Memory* = memoria di sola lettura) è una memoria programmata dal costruttore e le informazioni in essa contenute non possono essere modificate dall'utente; come si desume dal nome, infatti, la memoria ROM può essere solo "letta" (*read*).

Nella ROM sono registrati i programmi che gestiscono l'organizzazione interna del computer, in particolare un insieme di programmi detto *sistema operativo*. Tra i programmi più importanti del sistema operativo vi sono l'*interprete Basic* e l'*editor video*.

L'interprete Basic è un programma che traduce le istruzioni scritte dall'utente in Basic in un linguaggio (*linguaggio macchina*) che è l'unico comprensibile dai circuiti elettronici del computer. L'editor video, invece, controlla la scrittura e la correzione di frasi sul video.

L'altro tipo di memoria interna presente nel computer è la RAM (*Random Access Memory* = memoria ad accesso casuale). È, praticamente, la memoria a disposizione dell'utente per le sue elaborazioni.

I dati e i programmi che immettiano nel computer vengono immagazzinati nella RAM. Tale memoria è detta ad accesso casuale (o diretto) perché il computer può raggiungere direttamente ciascuna locazione di memoria per "leggere" il suo contenuto, o per "scrivere" una nuova informazione.

FUNZIONAMENTO

Il computer opera producendo e manipolando segnali elettrici di forma particolare.

A livello elettrico è in grado di distinguere solo tra un livello di tensione alto e un livello di tensione basso. Può riconoscere cioè solo

informazioni del tipo alta tensione/bassa tensione, che corrispondono, rispettivamente, a 1 e 0. L'unità minima di informazione del tipo 1 o 0 viene chiamata *bit*.

Come è possibile allora far riconoscere al computer una lettera dell'alfabeto, per esempio la lettera A? Facendo corrispondere, in uno speciale codice, alla lettera A una serie ben definita di successivi 1 e 0, cioè di segnali di alta e bassa tensione.

Il codice che si usa comunemente nei computer per tale corrispondenza è il codice ASCII. ASCII è la sigla di *American Standard Code for Information Interchange* (codice americano standard per l'interscambio di informazioni).

Nel codice ASCII a ciascuna lettera dell'alfabeto viene fatta corrispondere una successione di 7 bit, cioè di segnali elettrici del tipo 1 o 0. Ad esempio

lettera	codice
A	1000001
B	1000010
ecc...	

Nel tipo di computer di cui ci occupiamo le informazioni viaggiano all'interno dei circuiti elettronici in gruppi di 8 bit. Un gruppo di 8 bit si chiama *byte*.

Quando premiamo sulla tastiera ASCII il tasto A inviamo al computer un segnale di 8 bit (1 byte) della forma

01000001

sette di questi bit indicano, come abbiamo visto, la lettera A, nel codice ASCII; in questo codice l'ottavo bit viene utilizzato ai fini di un controllo da parte del computer; in alcune macchine, tuttavia (e il Commodore 64 è una di queste) viene utilizzato per estendere il codice standard e permettere l'uso di un maggior numero di caratteri (caratteri grafici in particolare).

Quante differenti combinazioni si possono ottenere combinando gli 1 e gli 0 in gruppi di 8?

con 1 bit sono possibili $2^1 = 2$ combinazioni

con 8 bit sono possibili $2^8 = 256$ combinazioni

Possiamo così far corrispondere a ogni combinazione una lettera dell'alfabeto inglese (maiuscole e minuscole), le cifre arabe, i simboli delle operazioni aritmetiche, i segni di punteggiatura e restano ancora

molte combinazioni libere. Nell'appendice C è riportato l'elenco dei simboli e funzioni che sono fatti corrispondere, nel Commodore 64, ai 256 possibili codici ASCII.

Il byte è l'unità fondamentale di misura dell'informazione nei computer. Le memorie interne, le ROM e le RAM, che abbiamo visto al paragrafo precedente, sono infatti suddivise in byte. Come multiplo del byte si usa il Kbyte ($1 \text{ Kbyte} = 1 \times 2^{10} \text{ byte} = 1024 \text{ byte}$).

Uno degli elementi in base a cui è opportuno valutare un computer è la quantità di memoria a disposizione dell'utente, cioè della RAM. Il Commodore 64, per esempio, ha una RAM di 64 Kbyte.

Se accendete il Commodore 64 vedrete infatti apparire la seguente scritta:

```
**** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM 38911 BASIC BYTES FREE
```

La prima riga presenta il computer, Commodore 64, e avverte quale versione del linguaggio Basic si può usare: il Basic V2 della Commodore.

La seconda riga, invece, dice che la RAM del computer contiene 64 Kbyte (64 K RAM) di memoria, cioè $64 \times 1024 = 65524$ byte di memoria. Una parte della RAM è tuttavia usata dal computer per usi specifici e, quando si opera in Basic, restano liberi (*free*) 38911 byte per l'utente.

Il Commodore 64 ha cioè a disposizione per l'utente 38911 locazioni di memoria, ognuna delle quali può contenere un byte di informazione, quindi, ad esempio, come abbiamo visto, una lettera dell'alfabeto.

ARITMETICA BINARIA

L'aritmetica a cui siamo abituati è quella in base decimale, in cui le cifre sono 10: da 0 a 9.

Consideriamo un numero qualsiasi, per esempio

2157

2 rappresenta le migliaia, 1 le centinaia, 5 le decine e 7 le unità. Possiamo quindi scrivere

$$2 \times 1000 + 1 \times 100 + 5 \times 10 + 7 = 2157$$

oppure, in forma esponenziale

$$2 \times 10^3 + 1 \times 10^2 + 5 \times 10^1 + 7 \times 10^0 = 2157$$

notiamo quindi che ogni cifra assume un valore che dipende dalla sua posizione e che tale valore può essere espresso in funzione di potenze della base 10.

L'aritmetica in base decimale non è l'unica possibile, possiamo costruire una aritmetica in una base qualsiasi.

Possiamo, per esempio, costruire una aritmetica in base due, in cui le cifre siano solo 2: lo 1 e lo 0; tale aritmetica si chiama binaria.

Un numero binario assume quindi la forma, ad esempio,

110101

Come nell'aritmetica decimale, ogni cifra assume un valore a seconda della sua posizione, e tale valore può essere calcolato in funzione delle potenze della base.

Per calcolare il valore decimale del numero binario appena scritto dovremo quindi operare nella seguente maniera:

$$\begin{aligned}
 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 &= \\
 1 \times 32 + 1 \times 16 + 0 &+ 1 \times 4 + 0 + 1 = \\
 32 + 16 + 4 + 1 &= 53
 \end{aligned}$$

Il computer, potendo riconoscere solo segnali del tipo 1 e 0, usa per i suoi calcoli l'aritmetica binaria.

Non fa parte degli scopi di questo libro addentrarci nello studio di tale parte della matematica; voglio solo darvi un esempio di somma:

aritmetica in base
decimale

$$\begin{array}{r}
 10 + \\
 10 = \\
 \hline
 20
 \end{array}$$

aritmetica in base
binaria

$$\begin{array}{r}
 10 + \\
 10 = \\
 \hline
 100
 \end{array}$$

il che conferma il vecchio detto "la matematica è una opinione".

Usiamo il computer

In questo capitolo vedremo come è fatta la tastiera ASCII, come far “girare” dei programmi già fatti e studieremo alcune regole del linguaggio Basic.

Basic è la sigla per *Beginners All-purpose Symbolic Instruction Code* (linguaggio simbolico universale per principianti) e, nello stesso tempo, in inglese significa di base, essenziale. È il linguaggio più comunemente usato nei personal computer per la comunicazione tra utente e macchina.

Esistono diverse versioni di Basic, leggermente diverse tra di loro. Quella che utilizzeremo noi è la versione V.2 del Commodore Basic, che è compresa da tutti i computer della Commodore.

I lettori che hanno o intendono acquistare computer di marche diverse non devono spaventarsi. Se i programmi non dovessero girare perfettamente sulle loro macchine, cioè se durante l'esecuzione del programma dovessero apparire sul video dei messaggi di errore del tipo **SYNTAX ERROR IN LINE XXX** (errore di sintassi alla linea XXX), basterà consultare un po' il manuale fornito con la macchina, per individuare quali modifiche apportare.

LA TASTIERA

La tastiera ASCII è il più comune fra i dispositivi di ingresso dei dati (*input device*) nei personal computer.

Non tutte le tastiere dei personal computer sono uguali; per ancorarci a un problema reale, vedremo come è fatta la tastiera del Commodore 64 (fig. 2.1).



Fig. 2.1 La tastiera del Commodore 64

Per gli scopi di questo libro e per scrivere e usare i programmi che vi sono contenuti, non è necessario conoscere l'uso di tutti i tasti; vedremo quindi solo quelli che ci interessano direttamente tralasciando ciò che riguarda i simboli grafici, i colori e le funzioni speciali.

La tastiera è simile a quella di una macchina da scrivere elettrica: sono presenti tutte le lettere dell'alfabeto inglese e le cifre arabe, più i normali simboli di punteggiatura.

Nel Commodore 64 sono disponibili due gruppi di caratteri: il primo comprende le lettere maiuscole e i simboli grafici ed è chiamato appunto maiuscolo/grafico; il secondo gruppo è composto invece dalle lettere minuscole e da quelle maiuscole, per cui si chiama minuscolo/maiuscolo.

Quando il computer viene acceso è predisposto per il gruppo maiuscolo/grafico; è quello che useremo noi. Per passare da un gruppo di caratteri all'altro bisogna schiacciare contemporaneamente i tasti **SHIFT** e **COMMODORE**.

Oltre ai tasti per i numeri e le lettere, ci sono quelli per le cinque operazioni: somma [+], differenza (o numero negativo) [-], moltiplicazione [*], divisione [/] ed elevamento a potenza [$\uparrow$].

Altri simboli che ci interessano sono: maggiore [>], minore [<], uguale [=], dollaro [\$] e pound (pronuncia "paund") [#].

Il tasto **SHIFT** funziona come quello che in una normale macchina da scrivere fornisce le lettere maiuscole. Se siamo, come è normale, nel gruppo di caratteri maiuscolo/grafico, premendo contemporaneamente il tasto **SHIFT** e, per esempio, il tasto **A**, compare un simbolo grafico. Se facciamo lo stesso con il tasto **1**, cioè come si dice in gergo premiamo il tasto **1** "shiftato", comparirà l'apostrofo [']. Il tasto **SHIFT LOCK** (*lock* = bloccaggio) serve per bloccare il tasto **SHIFT**.

Passiamo ora a vedere i tasti speciali che non si trovano su di una normale macchina da scrivere.

Il tasto con il logo (marchietto) Commodore shiftato serve, come abbiamo visto, a passare dal gruppo di caratteri maiuscolo/grafico al gruppo minuscolo/maiuscolo e viceversa.

Schiacciando il tasto RUN/STOP normalmente si interrompe l'esecuzione di un programma. Se ciò non succede bisogna schiacciare contemporaneamente i tasti RUN/STOP e RESTORE. Questa operazione riporta il computer alle condizioni di quando è stato acceso, tuttavia il programma eventualmente presente in memoria non viene alterato.

Tenendo premuto il tasto CTRL (ConTRoL) se schiacciamo il tasto con il numero 9 attiviamo la funzione di stampa REVERSE (ON = acceso) il cui effetto è scrivere i vari simboli sul video in negativo. Schiacciando contemporaneamente CTRL e il tasto 0 si disattiva la funzione REVERSE (OFF = spento).

LE CORREZIONI

Per correggere gli errori di battitura e per spostare le informazioni sullo schermo sono disponibili diversi tasti.

Il rettangolino lampeggiante che si vede sullo schermo e che indica la posizione in cui stiamo scrivendo si chiama *cursore* . Ci sono due tasti contrassegnati CRSR (*CuRSor*) uno con frecce verso l'alto e il basso e l'altro con frecce verso destra e sinistra. Con questi due tasti possiamo spostare il cursore, e successivamente scrivere, in qualsiasi punto dello schermo. Premendo i due tasti normalmente il cursore si sposta rispettivamente verso il basso e verso destra. Premendoli contemporaneamente al tasto SHIFT il movimento del cursore è verso l'alto e verso sinistra. Il movimento continua finché si tiene premuto uno dei due tasti CRSR.

Posizionato quindi il cursore nel punto dello schermo dove vogliamo effettuare la correzione, per cancellare i caratteri errati usiamo il tasto INST/DEL.

Tale tasto, non shiftato, produce l'effetto DEL (*DELe* = cancella), sposta cioè il cursore di uno spazio indietro cancellando il carattere a sinistra. In questa operazione tutta la parte della riga a destra del cursore, se c'è, si sposta anche essa di uno spazio. Lo stesso tasto, shiftato, produce la funzione INST (*INSerT* = inserisci), sposta cioè tutta la parte a destra del cursore creando lo spazio per inserire, per esempio, una o più lettere mancanti.

Per cancellare tutto lo schermo si usa il tasto CLR/HOME shiftato: il cursore si sposta nella posizione in alto a sinistra (*home* = casa) e lo

schermo viene completamente cancellato (*CLear* = pulire, cancellare). Se il tasto non è shiftato il cursore va in HOME, ma le informazioni presenti sullo schermo rimangono inalterate.

CARICARE E FAR GIRARE UN PROGRAMMA

Nel capitolo sull'hardware abbiamo visto che, oltre al computer, ai dispositivi di input e output, un sistema di elaborazione dati comprende delle memorie esterne, dette anche di massa. Esse consistono normalmente, per i personal computer, o in un registratore a cassette o in un floppy driver e servono per conservare programmi già fatti o archivi di dati.

Nel caso del registratore i programmi sono registrati su normali cassette audio, mentre nel caso del floppy driver sono registrati su speciali dischetti detti floppy disk.

Esistono in commercio, e ne esisteranno sempre di più in futuro, numerosi programmi già realizzati in grado di risolvere particolari problemi ed eseguire determinati calcoli. Alcuni dei programmi sono protetti, altri no. I programmi protetti, normalmente, non possono essere letti, né stampati, ma solo eseguiti, e non è quindi possibile modificarli. Altri programmi si trovano su libri come questo e possono essere copiati e modificati.

Comunque a un certo punto, o perché li avete acquistati o perché li avete prodotti o copiati voi, avrete a disposizione un certo numero di programmi registrati su cassetta o su disco. Vediamo quindi ora come è possibile caricare un programma da una memoria di massa nel computer e come, una volta caricato, sia possibile eseguirlo o, come si dice in gergo, "farlo girare".

Caricamento di programma da cassetta

Insieme alla cassetta con programmi registrati che avrete acquistato viene anche fornito l'elenco dei programmi con il loro nome. Se la cassetta è stata prodotta da voi avrete avuto cura di scrivervi i nomi dei programmi registrati con vicino le indicazioni dei numeri di giri di inizio e fine di ogni programma letti nel contagiri del vostro registratore.

Supponiamo che il programma che volete caricare si chiami TELAI. Eseguite quindi queste operazioni:

1. riavvolgete il nastro all'inizio, o, se avete l'indicazione del numero

di giri dove inizia il programma TELAI, posizionate il nastro un po' prima di tale punto;

2. battete sulla tastiera

LOAD "TELAI"

(*load* = carica) e schiacciate il tasto RETURN;

3. sul video apparirà la scritta

PRESS PLAY ON TAPE

(= premi il tasto PLAY sul registratore): eseguite;

4. dopo un po' (da qualche secondo a qualche minuto, dipende dalla posizione del programma nel nastro) appare la scritta

FOUND "TELAI"

(= ho trovato TELAI);

5. premete il tasto **COMMODORE** ed il programma viene caricato.

Caricamento di programma da disco

Inserite il disco con il programma desiderato (supponiamo sempre che si chiami TELAI) nel driver, facendo attenzione che l'etichetta sia rivolta verso l'alto e verso di voi (fig. 2.2); chiudete lo sportello di protezione premendo sulla leva.



Fig. 2.2 Come si inserisce un disco nel floppy driver

Quindi:

1. battete sulla tastiera

LOAD"TELA",8

e premete il tasto RETURN;

2. sul video compariranno le scritte

SEARCHING FOR TELAI

(sto cercando TELAI)

LOADING

(lo sto caricando)

READY

(sono pronto).

A questo punto il programma è caricato.

Il numero 8, presente nell'istruzione LOAD, è il numero di identificazione del floppy driver, in altre parole per il computer il floppy driver si chiama 8.

Osservazione. Quando si carica un nuovo programma nella memoria del computer, qualsiasi istruzione che vi si trovava precedentemente viene cancellata. Assicuratevi quindi, prima di caricare un nuovo programma, di "salvare" quello su cui eventualmente stavate lavorando, registrandolo in una memoria di massa.

Esecuzione di un programma

Quando un programma è nella memoria del computer, per farlo girare basta battere sulla tastiera il comando RUN (corri, letteralmente) e schiacciare il tasto RETURN: il computer inizia immediatamente l'esecuzione del programma.

REGISTRARE UN PROGRAMMA

Lavorando con il vostro computer vi capiterà, prima o poi, di avere un programma in memoria che non è registrato in una memoria di massa, o perché avete modificato un programma precedentemente caricato oppure perché lo avete appena costruito. Se spegnetè il compu-

ter il programma scomparirà per sempre. Per conservarlo dovete registrarlo in una memoria di massa.

Per continuare con l'esempio precedente, supponiamo che, caricato il programma TELAI, lo abbiate un po' modificato e ora vogliate registrarlo.

Registrare un programma su cassetta

Eseguite le seguenti operazioni:

1. scegliete un nome per il programma: la lunghezza massima del nome è 16 caratteri, compresi gli spazi; il nuovo nome sia ad esempio TELAI 1 (sette caratteri);
2. se la cassetta è vuota riavvolgetela all'inizio, se invece vi sono dei programmi già registrati, posizionatevi con l'aiuto del contagiri un po' dopo la fine dell'ultimo programma;
3. battete sulla tastiera

SAVE"TELAI 1"

(salva TELAI 1) e premete il tasto RETURN;

4. sul video appare la scritta

PRESS PLAY AND RECORD ON TAPE

(premi i tasti PLAY e RECORD sul registratore) eseguite, schiacciando i due tasti contemporaneamente;

5. quando appare la scritta

READY

il programma è registrato; prendete nota del numero di giri dove il programma termina; scrivete nome del programma e numeri di giri di inizio e fine sull'apposito foglietto contenuto nella cassetta.

Registrare un programma su disco

Eseguite le seguenti operazioni:

1. scegliete il nome del programma (massimo 16 caratteri), supponiamo sia TELAI 1;
2. battete sulla tastiera il comando

SAVE"TELAI 1",8

seguito dal tasto RETURN;

3. sul video appaiono, successivamente

SAVING "TELA1 1"

(sto salvando TELAI 1)

OK

READY

e a quel punto il programma è registrato.

Osservazione. All'inizio di questo paragrafo sulla registrazione di un programma in una memoria di massa, abbiamo supposto di aver caricato il programma TELAI, di averlo modificato e di voler quindi registrare la nuova versione. Alla nuova versione abbiamo assegnato il nome TELAI 1.

Dare nomi diversi a diverse versioni di un programma, o, a maggior ragione, a programmi diversi, è assolutamente indispensabile. In caso contrario si verificano degli spiacevoli inconvenienti. Vediamo i due casi.

Cassetta

Se registriamo due programmi con lo stesso nome su cassetta, quando andremo a caricarli il computer non avrà alcuna possibilità di distinguere tra i due: ogni programma è infatti identificato dal suo nome. Il computer quindi caricherà il primo programma con quel nome che incontra sul nastro.

Posizionando opportunamente la cassetta, mediante il contagiri, sarà possibile caricare la versione del programma desiderata, ma si va facilmente incontro ad errori di caricamento.

Disco

Se tentiamo di registrare un programma su un disco in cui vi sia già un programma con lo stesso nome, il programma non verrà registrato e la spia del floppy driver comincerà a lampeggiare segnalandoci l'errore.

Esiste tuttavia la possibilità di sostituire la vecchia versione di un programma registrato su disco con una nuova versione, mantenendo lo

stesso nome: ciò è possibile con il comando **SAVE** con **REPLACE**. Continuando con l'esempio precedente, supponiamo di aver modificato il programma **TELA**I e che la nuova versione sia presente nella memoria del computer. La vecchia versione, presente sul disco, non ci interessa più e vogliamo sostituirla con la nuova. Basta scrivere il comando

SAVE"@0:TELAI",8

e premere il tasto **RETURN**: la vecchia versione del programma di nome **TELA**I viene in tal modo cancellata e sostituita con la nuova.

COSE IMPORTANTI DI BASIC

Come ho già detto, in questo libro usiamo la versione di Basic V.2 del Commodore Basic.

I programmi scritti con questo linguaggio sono utilizzabili su quasi tutti i personal in commercio senza particolari modifiche, dato che la V.2 non è una versione molto sofisticata del Basic e le istruzioni che useremo sono le più comuni.

Vi consiglio, durante la lettura di questo paragrafo, di cominciare a provare sul computer, operando di vostra iniziativa ed eseguendo quello che viene spiegato nel testo.

Per eseguire delle operazioni battete sulla tastiera, per esempio,

PRINT 5 + 2

poi battete il tasto **RETURN** e sul video apparirà il numero 7.
Ancora

PRINT 4 + 2 * 7/5 ↑ 2 * (4 ↑ 3 + 3)

poi battete il tasto **RETURN** e vedrete il risultato dell'operazione:

41.52

Osservazione: Avrete notato che ogni volta che diamo un comando al computer, alla fine della linea battiamo il tasto **RETURN**. Vediamo di capire bene cosa succede.

Quando digitate una qualsiasi serie di tasti (a parte quelli di funzioni speciali) vedrete apparire sullo schermo i caratteri corrispondenti; in quel momento le informazioni sono depositate solo nella memoria del-

lo schermo. Premendo il tasto RETURN si ordina al computer di passare le informazioni contenute nella riga su cui è posizionato il cursore nella parte del computer in cui vengono svolte le operazioni, dove tali informazioni vengono quindi lette ed eseguite.

Finché non premete il tasto RETURN le informazioni restano solo nella memoria dello schermo e non vengono utilizzate.

Numeri

Le grandezze numeriche possono essere espresse in due maniere:

— grandezze intere

— grandezze decimali (*floating point*, ovvero in virgola mobile).

Nello scrivere tali grandezze si può anche usare la notazione scientifica:

Esempi:

0	+0	-0
1	+1	0.1E+1
415.3	4153E-1	0.4153E+3
-.0000613	-6.13E-5	-613E-7

N.B. Rispetto alla notazione aritmetica italiana omettete i punti delle migliaia e sostituite la virgola con il punto.

Esempio:

notazione italiana

5.347.821,47

notazione Basic

5347821.47

Stringhe

Viene chiamata *stringa* una sequenza di caratteri alfanumerici (alfabetici e/o numerici) chiusa tra virgolette. Sono stringhe, per esempio:

"ciao"

"CALCOLO DI UN TELAIO"

"13 gennaio 1998"

"QUANTO VALE IL CARICO Q?"

Variabili

Variabile è un nome che rappresenta numeri, stringhe o gruppi di numeri.

In Basic sono previsti quattro tipi di variabili: numeriche, numeriche intere, di stringa e matriciali.

I nomi delle variabili possono essere di due soli caratteri di cui il primo alfabetico. Per distinguere le variabili numeriche da quelle numeriche intere, di stringa e matriciali queste ultime hanno un suffisso particolare. La forma dei nomi delle variabili è quindi, per esempio, la seguente:

Numeri reali uno o due caratteri

A, AB, K, A1, Z3

Numeri interi uno o due caratteri seguiti dal segno %

A%, K1%, CA%

Stringhe uno o due caratteri seguiti dal segno \$ (dollaro)

A\$, K3\$, CA\$

Vettori e matrici (array) uno o due caratteri (con il suffisso \$ o % se il contenuto della matrice o del vettore sono, rispettivamente, stringhe o numeri interi) seguiti, tra parentesi, dalle dimensioni del vettore o della matrice

V(10), T\$(2,4), K2(3,4,7)

Operatori e formule

Gli operatori sono simboli che indicano particolari operazioni.

Operatori aritmetici. Nell'ordine (*gerarchia*) secondo cui vengono eseguiti dal computer sono:

<i>Operatore</i>	<i>Operazione</i>	<i>Esempio</i>
↑	esponenziale	$X \uparrow Y, X \uparrow 2$
$\times, /$	moltiplicazione, divisione	$X * Y, Z/Y$
$+, -$	addizione, sottrazione	$X + Y, 2 - 5$

Le formule sono costituite da gruppi di operazioni. Se vogliamo alterare l'ordine in cui queste vengono eseguite, rispetto alla "gerarchia" del computer, possiamo usare le parentesi. Per esempio:

formula algebrica

formula in Basic

$$X + 2Y$$

$$X + 2 * Y$$

$$\frac{X + Y}{2}$$

$$(X + Y) / 2$$

$$(X^2)^Y$$

$$(X \uparrow 2) \uparrow Y$$

$$X^{Y^Z}$$

$$X \uparrow (Y \uparrow Z)$$

$$X(-Y)$$

$$X * (-Y)$$

$$B^2 - 4AC$$

$$B \uparrow 2 - 4 * A * C$$

$$[2(A + B)^2 + (3C)^2]^{\frac{m}{n+1}} \quad (2 * (A + B) \uparrow 2 + (3 * C) \uparrow 2) \uparrow (M / (N + 1))$$

Operatori relazionali

Nel capitolo sulla programmazione abbiamo visto che si può chiedere al computer di fare dei confronti tra due grandezze e, a seconda del risultato, eseguire una certa serie di operazioni oppure un'altra. Gli operatori relazionali sono quei simboli che indicano il tipo di confronto che vogliamo fare. Essi sono i seguenti:

<i>Operatore</i>	<i>Relazione</i>	<i>Espressione</i>
=	uguale	$X = Y$
< >	diverso	$X < > Y$
<	minore	$X < Y$
>	maggiore	$X > Y$
< =	minore o uguale	$X < = Y$
> =	maggiore o uguale	$X > = Y$

Quando in una espressione compaiono sia operatori relazionali che aritmetici, viene sempre eseguita prima la parte aritmetica. Per esempio, l'espressione

$$X + 5 < (K - 5) / Z$$

USIAMO IL COMPUTER

è vera se il valore di $X + 5$ è minore del valore di $K - 5$ diviso per Z .
Altri esempi di enunciati Basic contenenti operatori relazionali:

IF $D < 0$ THEN 1000

se (= IF) D è minore di zero *allora* (THEN) esegui l'istruzione 1000

IF $Y + 5 < > Z$ THEN $K = 2$

se il valore di $Y + 5$ è diverso da Z allora poni il valore di K uguale a 2.

Operatori logici

Gli operatori logici consentono di realizzare confronti di più relazioni contemporaneamente e operazioni a livello di matematica binaria.
Essi sono:

AND

OR

NOT

Esempi:

IF $D \geq 0$ AND $D \leq L$ THEN 1000

affinché sia eseguita l'istruzione 1000 devono essere verificate *entrambe* le relazioni, cioè D deve essere sia maggiore o uguale a zero, sia minore o uguale a L .

IF $K + 1 = 5$ OR $Z > 2$ THEN $X = Y$

basta che una delle due relazioni sia vera perché venga posto $X = Y$.

IF NOT $X < Y$ THEN $K = K + 1$

la relazione è vera, e quindi K viene posto uguale a $K + 1$, se $X \geq Y$.

La gerarchia delle operazioni del Basic prevede che se in una espressione sono presenti diversi tipi di operazioni vengono eseguite prima quelle aritmetiche, poi quelle relazionali e per ultime quelle logiche.

Operazioni con stringhe

Le stringhe possono essere concatenate usando il segno +.

Ad esempio il programma seguente

```
10 A$ = "TRI"  
20 B$ = "ZIO"  
30 PRINT "PA" + A$ + B$  
RUN
```

se eseguito al computer dà come output

PATRIZIO

Gli operatori relazionali si usano normalmente anche per confrontare stringhe, non solo valori numerici. Per esempio:

$P\$ \neq Q\$$ la relazione è soddisfatta se la stringa corrispondente a $P\$$ è diversa da quella indicata con $Q\$$.

USIAMO IL COMPUTER

E ora, finalmente, possiamo incominciare a lavorare con il computer con un po' di cognizione di quello che stiamo facendo.

Allora, sedetevi davanti al vostro computer acceso o, se non l'avete, immaginate di sedervi davanti. Come ogni volta prima di incominciare un nuovo programma, battete sulla tastiera il comando NEW (nuovo) e schiacciate il tasto RETURN. Schiacciando RETURN, come sappiamo, tutto ciò che si trova nella riga appena scritta viene introdotto nel computer e se, come in questo caso (il nostro NEW) ciò che abbiamo scritto è un comando Basic, esso viene immediatamente eseguito.

L'esecuzione di NEW provoca la cancellazione dell'eventuale programma presente in memoria, l'azzeramento delle variabili e delle matrici. Possiamo ora scrivere il nostro primo programma in Basic. Battete sulla tastiera la seguente riga

```
10 PRINT "PATRIZIO QUINTILI"
```

schiacciate il tasto RETURN. Poi digitate (cioè battete)

RUN

e vedrete apparire la scritta

PATRIZIO QUINTILI

Purtroppo a me manca una delle doti del buon programmatore: la fantasia, o forse nello scrivere questo programma ho voluto soddisfare il mio narcisismo, ma voi potete anche usare il vostro nome, se volete, o una qualsiasi altra stringa.

Bene! Adesso pulite lo schermo con il tasto CLR/HOME shiftato. Secondo programma: digitate la riga

```
20 PRINT "NATO IL 9.2.19.."
```

schiacciate RETURN, digitate RUN, seguito da RETURN.
Terzo programma. Digitate

```
5 A$ = "PATRIZIO"
```

```
15 B$ = "QUINTILI"
```

```
25 C$ = "NATO IL 9.2.19.."
```

Come sempre, battete il tasto RETURN alla fine di ogni riga.

Le righe possono avere praticamente qualsiasi numero intero: il computer provvede a metterle in ordine crescente e a eseguirle partendo dal numero più basso.

Digitate il comando:

```
LIST
```

Vedrete apparire sul video

```
5 A$ = "PATRIZIO"
```

```
10 PRINT "PATRIZIO QUINTILI"
```

```
15 B$ = "QUINTILI"
```

```
20 PRINT "NATO IL 9.2.19.."
```

```
25 C$ = "NATO IL 9.2.19.."
```

Ora le righe 10 e 20 non mi interessano più, così le tolgo. Per togliere una riga basta scriverne un'altra con lo stesso numero, il computer provvede a cancellare la prima riga. Nel nostro caso non vogliamo sostituire niente alla riga 10 quindi batteremo semplicemente il nume-

ro 10 seguito da RETURN. La riga 10 risulterà vuota nel programma, cioè non ci sarà.

Cancellate anche la riga 20. Eseguite un LIST per controllare se le due righe si sono cancellate. Tutto bene? Allora andate avanti con un CLR/HOME shiftato e un RUN.

Non succede niente; abbiamo infatti tolto dal programma tutte le istruzioni di PRINT. Aggiungiamo quindi

```
30 PRINT A$;B$,C$
```

Pulite lo schermo e date un RUN. La scritta che appare sul video è quasi la stessa che appariva con il secondo programma.

Nei programmi precedenti abbiamo lavorato con stringhe, cioè con sequenze di caratteri alfanumerici racchiuse tra virgolette; le corrispondenti variabili hanno i nomi che terminano con il segno \$.

Vediamo ora un programma che esegue calcoli con variabili numeriche. Scrivete il comando

```
NEW
```

e pulite lo schermo.

Digitate il seguente programma:

```
10 A = 10
```

```
20 PRINT A,A * 2,A ↑ 2,A ↑ 3
```

```
RUN
```

Sul video appariranno i numeri

```
10    20    100    1000
```

spaziati di una certa quantità. Se infatti dividiamo le varie espressioni che seguono l'istruzione PRINT con virgole (,), lo schermo viene diviso in quattro campi di dieci caratteri ciascuno e le espressioni vengono stampate a partire dall'inizio di ciascun campo.

Sostituite adesso nella riga 20 alle virgole dei punti e virgola (;). Pulite lo schermo e fate girare di nuovo il programma. I numeri appariranno ora spaziati di due soli caratteri

```
10  20  100  1000
```

il punto e virgola posto tra le espressioni da stampare le spazia infatti di un solo carattere, lasciando, per i numeri, un altro carattere per il segno. In questo caso il segno è positivo e non viene riportato.

Sostituite ora alla riga 10 il valore della variabile A con -10. Fate girare il programma e vedrete apparire i numeri

-10 -20 100 -1000

come potete notare i numeri negativi sono distanziati dagli altri di un solo carattere.

Battete ora

NEW

pulite lo schermo e scrivete questo nuovo programma

```
10 PRINT"#####PROVA"
```

```
READY.
```

I simboli che appaiono tra virgolette si ottengono, rispettivamente, premendo il tasto CRSR con la freccia verso il basso (DOWN) per 12 volte, e il tasto CRSR con la freccia verso destra (RIGHT) per 17 volte, dopo aver aperto le virgolette. Quando chiudete le virgolette i due tasti riprendono a funzionare normalmente.

Pulite lo schermo e fate girare il programma. Apparirà la scritta PROVA al centro dello schermo, alla dodicesima riga e a partire dalla diciassettesima colonna.

Che cos'è successo? I comandi del cursore, come i tasti CLR/HOME e INST/DEL, possono essere programmati. Possiamo cioè inserirli in una stringa e il computer quando li incontra, durante l'esecuzione di una istruzione PRINT, esegue l'operazione corrispondente.

Nel nostro caso quindi il computer prima sposta il cursore 12 righe in basso, poi 17 colonne a destra e infine inizia a stampare la parola PROVA.

Adesso continuate da soli a sperimentare il comportamento dell'istruzione PRINT.

Costruite dei piccoli programmi che eseguano calcoli e stampino scritte; esercitatevi a inserire, spostare e correggere le righe dei programmi. Usate l'istruzione LIST per vedere quello che avete prodotto.

N.B. Ogni riga di programma non può contenere più di 80 caratteri, cioè non può essere più lunga di due righe dello schermo.

Se incontrate troppe difficoltà, rileggetevi questo capitolo, se invece volete saperne di più leggete il prossimo. Buon divertimento!

Secondo livello

Leggendo questa seconda parte del libro imparerete a costruire programmi in Basic che risolvono dei semplici problemi di calcolo.

Nel capitolo 3 studieremo la logica secondo cui devono essere organizzate le operazioni da far eseguire al computer: studieremo cioè le tecniche di base della programmazione.

Nel capitolo 4 impareremo le principali istruzioni del linguaggio Basic; impareremo cioè a tradurre le nostre idee in un linguaggio comprensibile al computer.

Infine nel capitolo 5 vedremo una serie di programmi completi che risolvono problemi di costruzioni, estimo e topografia.

Studiando questi tre capitoli con attenzione, sarete in grado di costruire da soli programmi anche complessi, senza incontrare particolari difficoltà.

Programmazione

In questo capitolo studieremo le tecniche di base per la costruzione di programmi per computer attraverso alcuni esempi che verranno poi tradotti in Basic al capitolo 4.

Alcuni di voi avranno certamente usato delle calcolatrici programmabili, facendosi così un'idea della programmazione. È bene tuttavia ripetere alcuni concetti per avere le idee più chiare.

Vediamo innanzitutto per quali tipi di problemi è più utile usare un computer, rispetto ad altri strumenti tradizionali.

Se nella vostra vita professionale vi capitasse solo una volta di dover eseguire il progetto di una sezione in cemento armato, risulterebbe più pratico eseguire tale calcolo in maniera tradizionale, con l'aiuto cioè di una calcolatrice e di un manuale di calcolo. Ma se dovete calcolare anche solo cinque sezioni in cemento armato all'anno, allora vi conviene lavorare un giorno per costruirvi il programma che risolva il problema una volta per tutte.

Otterrete così due vantaggi: non dovrete continuamente sprecare le vostre energie intellettuali su un problema che avete già risolto e, soprattutto, sarete sicuri di non commettere errori. Se infatti un programma è costruito in maniera corretta è praticamente impossibile che il computer commetta errori nell'eseguirlo.

DIRECT MODE - PROGRAM MODE

Il computer può funzionare in due modi:

1. *direct mode* (modo diretto, o immediato).

In direct mode le istruzioni che diamo al computer vengono eseguite immediatamente e i risultati delle elaborazioni appaiono sul video o sulla stampante; le istruzioni non vengono però memorizzate e si perdono. Ciò significa che per far ripetere le stesse operazioni al computer dobbiamo ogni volta riscrivere le istruzioni. Lavoreremo in modo immediato quando costruiremo i programmi per la prima volta o quando li correggeremo.

Esempio

Abbiamo già lavorato in modo immediato nel capitolo 2, quando abbiamo imparato a eseguire delle operazioni. Battete sulla tastiera

PRINT 10-4

premete il tasto RETURN e vedrete apparire sul video il risultato dell'operazione

6

Ancora, scrivete:

PRINT "ESERCIZIO DI SCRITTURA IN DIRECT MODE"

battete il tasto RETURN e sul video apparirà la scritta

ESERCIZIO DI SCRITTURA IN DIRECT MODE

2. *program mode* (modo programma, o differito).

In program mode le istruzioni vengono inserite sotto forma di programma; ogni istruzione è preceduta da un numero di linea e dopo l'elaborazione il programma rimane in memoria per successivi usi.

È nel modo programma che il computer esplica meglio la sua potenza. Quando avrete dei programmi già fatti, lavorerete quasi sempre in modo programma.

Esempio

Accendete il computer e scrivete questo breve programma:

10 A = 10

20 B = -4

30 PRINT A + B

Digitate il comando RUN, seguito da RETURN. Apparirà sul video il risultato dell'elaborazione:

6

Eseguite ora il comando

LIST

vedrete comparire di nuovo il programma che abbiamo appena scritto. Il programma resta in memoria finché non diamo il comando NEW o spegniamo il computer.

Se volete ripetere l'operazione basta digitare di nuovo RUN.

PROGRAMMI E FLOW-CHART

Abbiamo già fatto eseguire al computer alcuni brevi programmi, avrete quindi capito che *programma* non è altro che un elenco di istruzioni che fanno eseguire al computer operazioni ben specificate.

Siccome la struttura elettronica del computer funziona secondo una logica binaria (scelte del tipo sì/no) i problemi che vogliamo risolvere dovranno essere organizzati secondo tale logica e senza possibili ambiguità.

Per descrivere i programmi si usa comunemente un semplice linguaggio grafico, i cui simboli sono riportati in fig. 3.1.

Il primo simbolo, terminale, viene usato per indicare l'inizio e la fine di un programma. Il parallelogramma serve per individuare le operazioni di ingresso/uscita (input/output) di dati dal computer. Una generica operazione viene invece indicata con un rettangolo.

Una operazione molto importante che può eseguire il computer è quella di confrontare due grandezze tra di loro e, in base al risultato del confronto, prendere una decisione sul come proseguire nella esecuzione del programma. Tale operazione di decisione viene indicata con un rombo o, a volte, quando è necessario inserire nel riquadro molto testo, con un esagono allungato.

Per unire diversi parti del disegno, sia perché questo risulta troppo complesso, sia perché non entra più nel foglio su cui si sta scrivendo, si usano i connettori.

Infine, la direzione secondo cui fluisce il programma tra un simbolo e l'altro viene evidenziata con frecce.

Per costruire un programma, dunque occorre *organizzare il problema in maniera logica* e per questo è opportuno *scriverlo sotto forma di schizzo grafico* con i simboli di fig. 3.1. Gli schizzi di questo tipo vengono chiamati *flow-chart* o *diagrammi di flusso*.

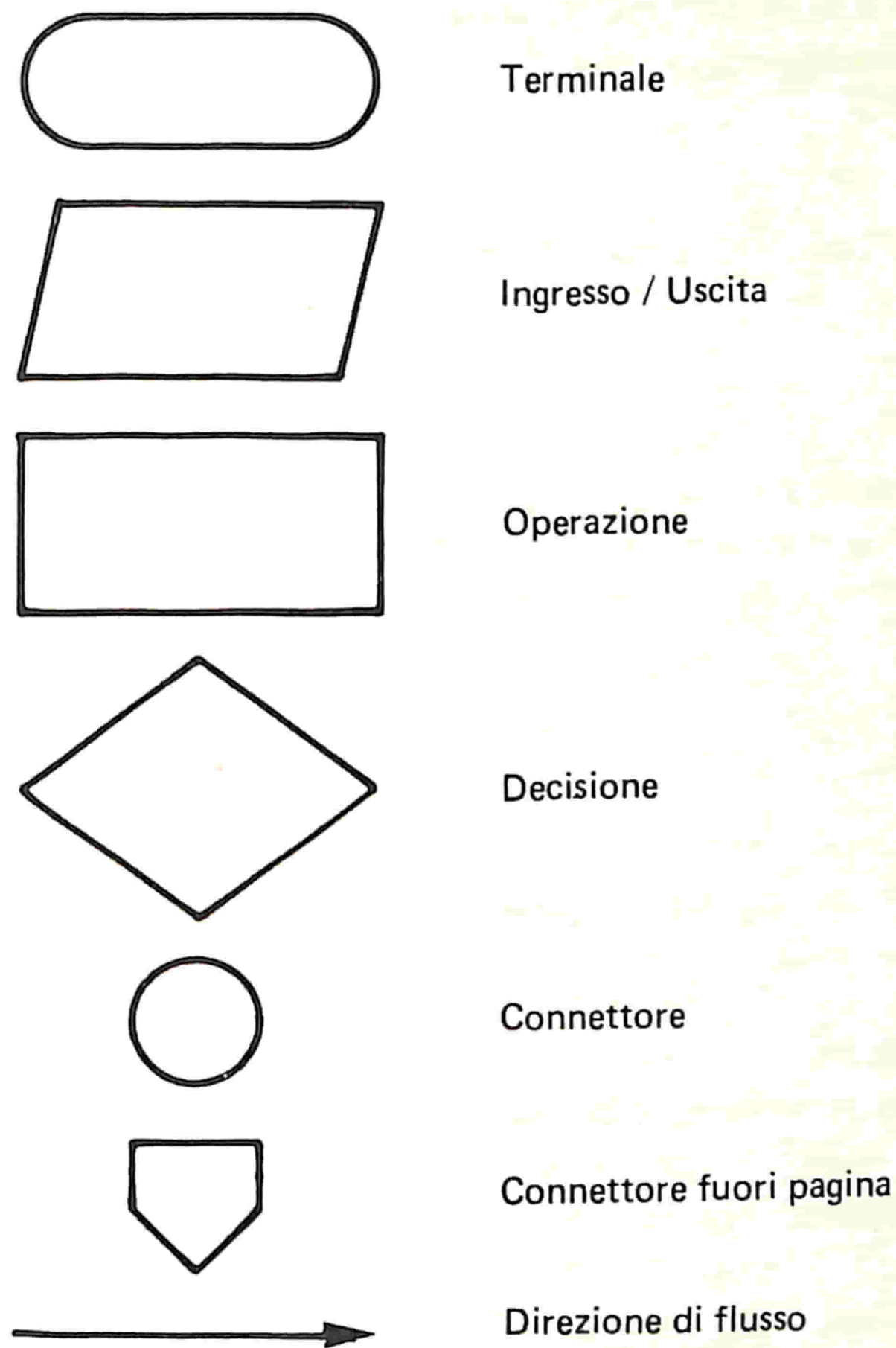


Fig. 3.1 Simboli usati nei diagrammi di flusso

Vediamo ora qualche applicazione.

Problema 1

Supponiamo di voler calcolare l'area di un cerchio, con la formula $A = \pi r^2$, per un valore qualsiasi del raggio r .

Le successive fasi del problema e il relativo diagramma di flusso sono i seguenti:

1. conoscere il valore numerico di r
2. calcolare l'area con la formula data
3. stampare i valori del raggio e dell'area
4. fine

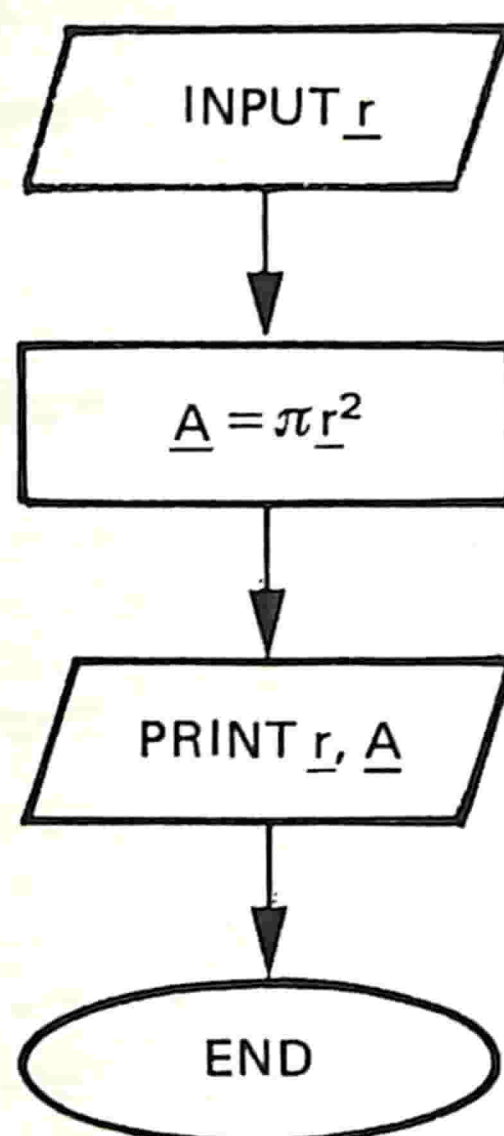


Fig. 3.2 Diagramma di flusso del problema 1

Problema 2

Calcolare le reazioni vincolari di una trave semplicemente appoggiata di luce qualsiasi sottoposta a un carico verticale concentrato in un punto qualsiasi (fig. 3.3).

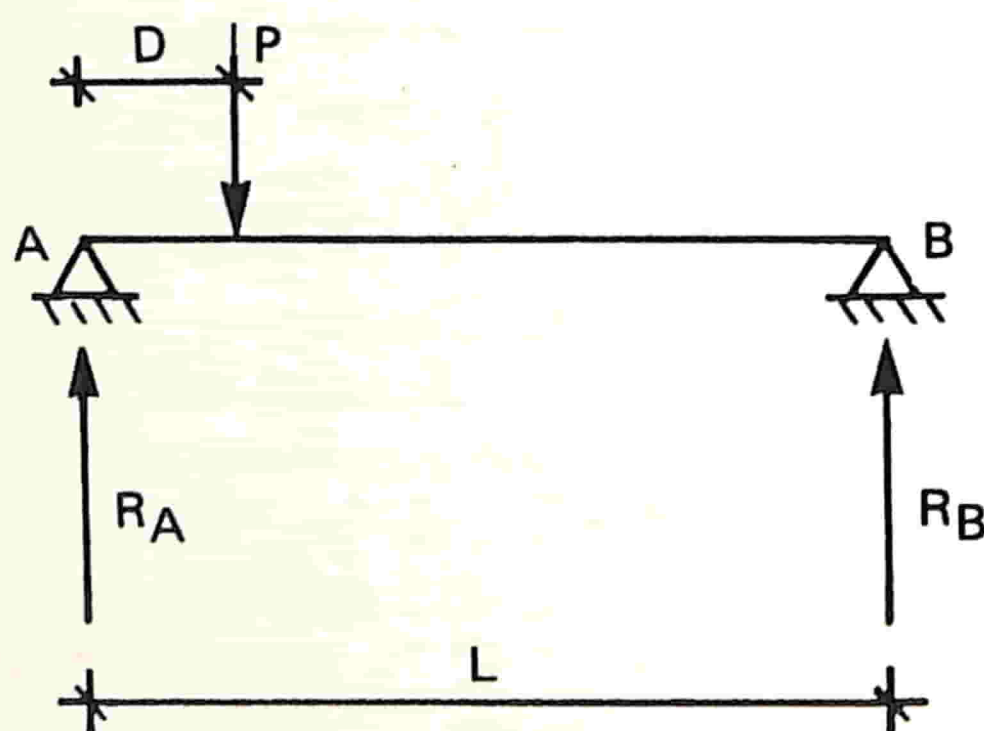


Fig. 3.3 Trave semplicemente appoggiata soggetta a un carico concentrato

Le relazioni che risolvono il problema sono (dalla scienza delle costruzioni):

$$R_B = \frac{P \cdot D}{L}; \quad R_A = P - R_B$$

Lo schema logico è il seguente:

1. immettere i valori della luce della trave L , del carico P , della distanza D del carico P dall'appoggio A ;
2. calcolare R_B con la formula data;
3. calcolare R_A con la formula data;
4. stampare i valori dei dati iniziali e dei risultati;
5. fine.

Il relativo diagramma di flusso è riportato in fig. 3.4.

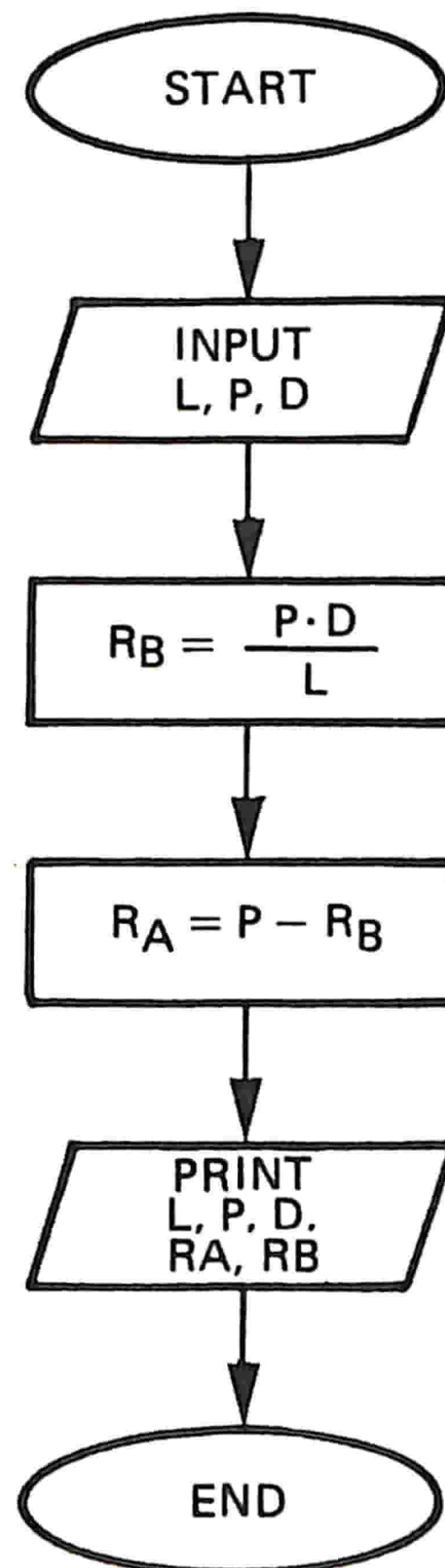


Fig. 3.4 Diagramma di flusso del problema 2: calcolo reazioni trave appoggiata per un carico concentrato

PROBLEMI DECISIONALI E SALTI

Gli esempi di programmi che abbiamo visto per i problemi 1 e 2 corrispondono a una elaborazione di tipo lineare: tutte le operazioni, cioè, si svolgono in maniera sequenziale, una dopo l'altra.

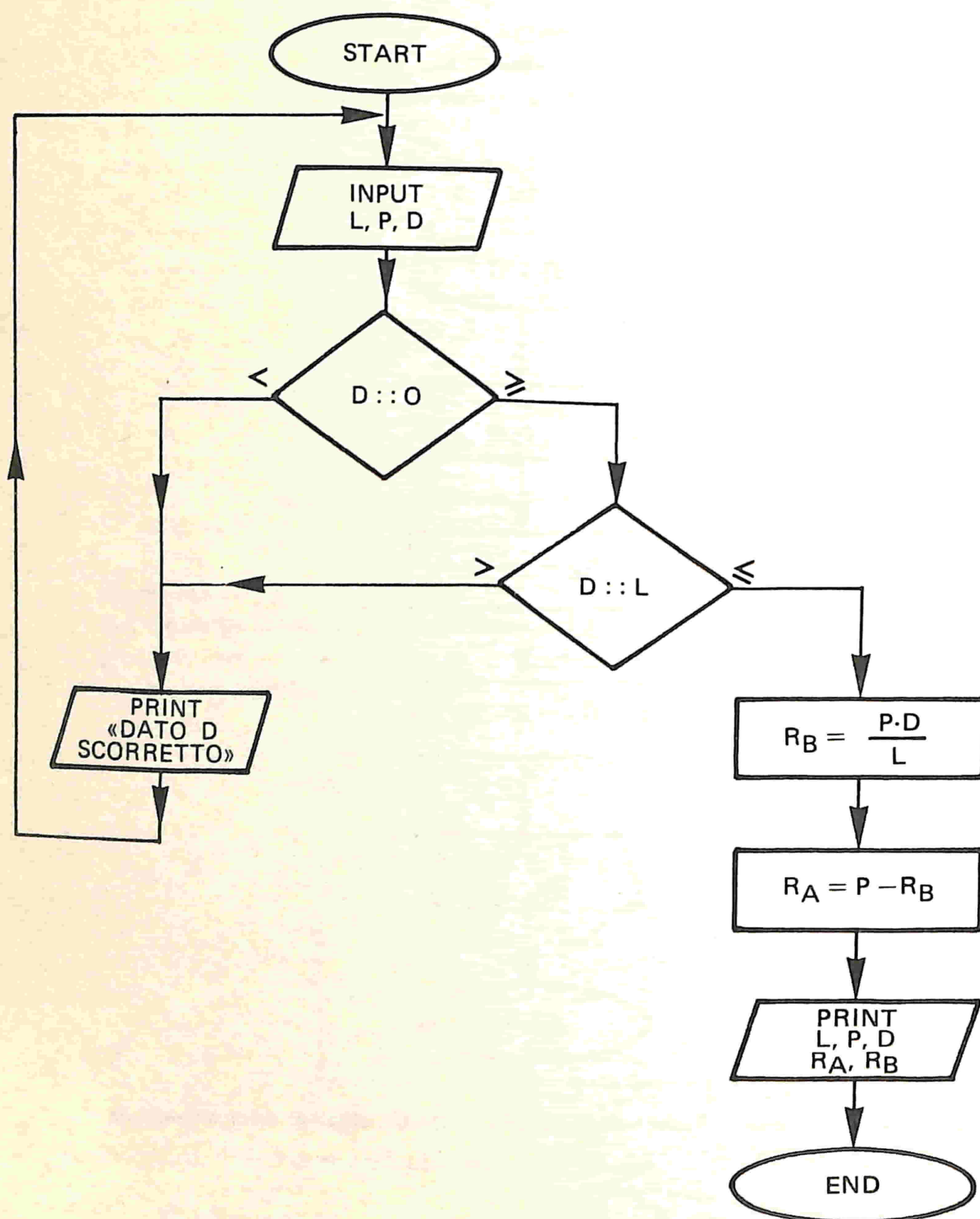


Fig. 3.5 Diagramma di flusso del problema 3: calcolo reazioni trave appoggiata per un carico concentrato, con controllo della distanza D

Quasi sempre nella programmazione ci troviamo invece di fronte a problemi in cui, a un certo punto del programma, bisogna prendere una decisione: se un certo dato è maggiore (o minore o diverso) rispetto a un altro, bisogna eseguire una certa serie di operazioni, in caso contrario si deve eseguire una diversa serie di operazioni.

Continuiamo, per esempio, lo studio del problema 2 relativo al calcolo delle reazioni di una trave isostatica. Mentre la luce L e il carico P possono assumere qualsiasi valore, il valore della distanza D deve essere compreso tra zero ed L : $0 \leq D \leq L$. Miglioriamo quindi il nostro programma inserendo un controllo del valore della distanza D . Se, per errore, il valore immesso per D è minore di zero o maggiore di L , facciamo stampare una scritta che ci avverta dell'errore. Se, al contrario, il valore di D è corretto, il programma proseguirà come nell'esempio precedente.

Problema 3

Come il problema 2, ma con controllo del valore della distanza D .

Tralasciamo la parte descrittiva e vediamo direttamente il diagramma di flusso in fig. 3.5.

Il programma inizia con l'immissione dei valori di L , P , D ; poi il valore di D viene confrontato (il simbolo $::$ significa confronto) con zero; se D è minore di zero, e quindi il valore non è accettabile, il programma prosegue dal lato sinistro del simbolo di decisione, stampa la frase **DATO D SCORRETTO** e ritorna all'inizio chiedendo di nuovo l'input dei dati. Lo stesso succede se al secondo confronto D risulta maggiore di L . Se invece il valore della distanza è compreso tra zero e L , come deve essere, il programma prosegue nel ramo di destra del diagramma con il calcolo e la stampa delle reazioni.

Il programma che abbiamo appena costruito non sarà eseguito necessariamente in maniera lineare, ma, per particolari valori di D , *salterà...* da un punto all'altro.

In due casi, dopo i due confronti, il salto avverrà se si verifica una certa condizione ($D < 0$ o $D > L$), perciò questo tipo di salto si chiama *salto condizionato* (*conditional jump*); in un altro caso, dopo l'istruzione di stampa **DATO D SCORRETTO**, il programma salta all'inizio senza alcuna condizione: questo tipo di connessione si chiama *salto incondizionato* (*unconditional jump*).

ITERAZIONE

Oltre ai salti condizionati e incondizionati, un'altra operazione viene usata costantemente nella programmazione: l'*iterazione* o *looping*. Es-

sa consiste nel far eseguire per un certo numero di volte una serie di operazioni finché non si verifichi una data condizione, dopo di che il programma riprende a scorrere in maniera lineare. Vediamo un esempio.

Problema 4

Stampare la tabellina pitagorica di un numero qualsiasi.

Fissiamo la lunghezza della tabellina a 10. Il diagramma di flusso è quello di fig. 3.6.

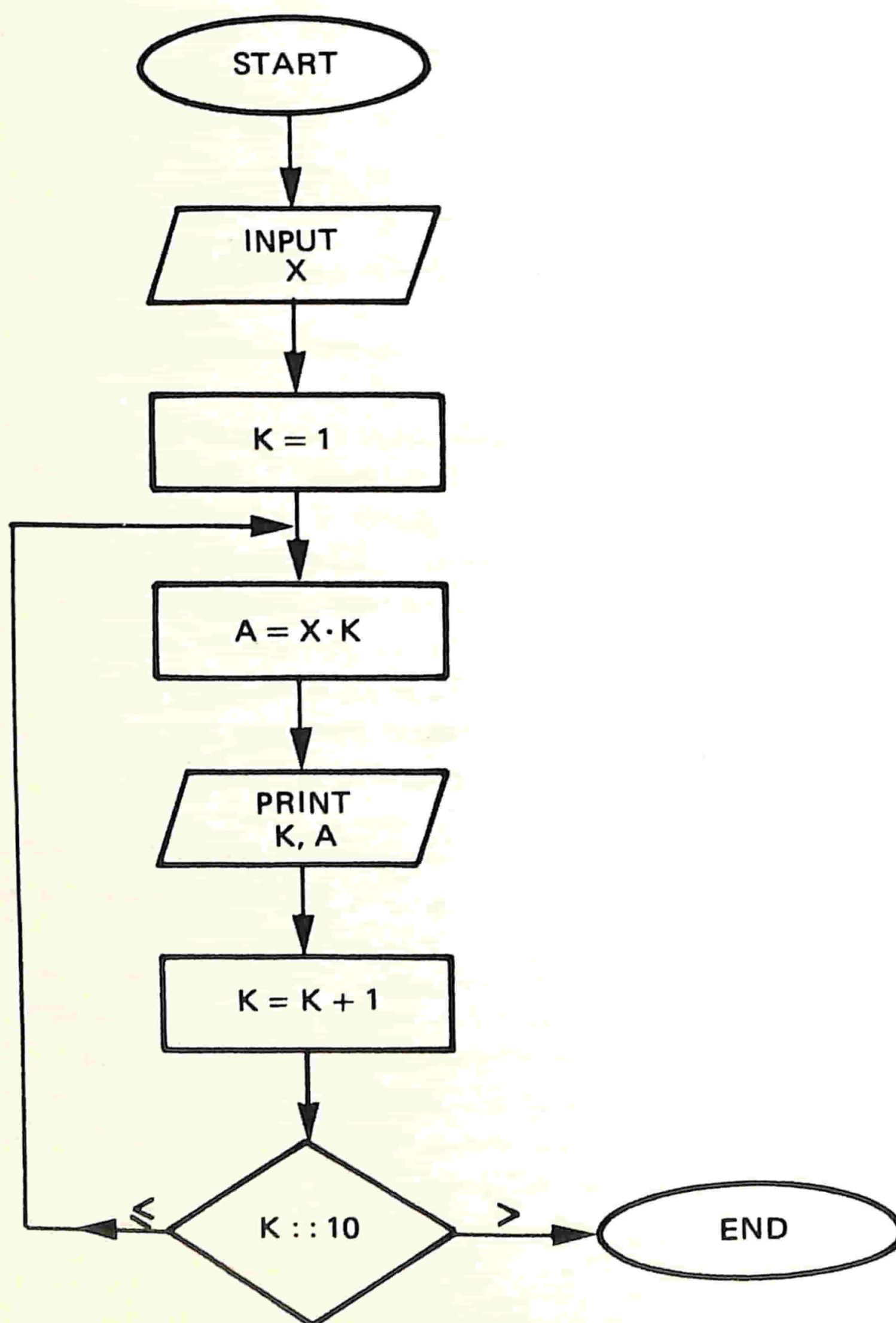


Fig. 3.6 Diagramma di flusso del problema 4: tabellina pitagorica di un numero qualsiasi

Il programma inizia con l'input del numero di cui vogliamo stampare la tabellina; tale valore viene assegnato alla variabile X ; poi viene posta uguale a 1 la variabile K che useremo come un contatore. Viene calcolato il valore di X per K , con $K = 1$, e tale valore assegnato alla variabile A . Si stampano i valori di K e di A . Il valore di K viene aumentato di 1; tale operazione, $K = K + 1$, se viene letta come un'espressione algebrica risulta poco chiara, in realtà significa: il nuovo valore di K è uguale al vecchio valore di K più 1.

Il nuovo valore di K (2) viene quindi confrontato con 10; se è minore o uguale a 10, come in questo caso, il programma prosegue saltando indietro come si vede nel diagramma. Viene quindi ripetuto il calcolo di X per K , vengono stampati i nuovi valori di K (2) e A ($X \cdot 2$), K viene incrementato di 1 e così via finché, quando K è uguale a 11, il programma esce dal ciclo (*loop*) e prosegue nel ramo di destra, terminando.

L'output in stampa sarà dato da due colonne di numeri la prima delle quali va da 1 a 10 (i successivi valori della variabile K) e l'altra riporta i prodotti dei vari valori di K per il numero X , cioè la tabellina pitagorica di X .

Come avrete capito, in questo programma abbiamo costruito un ciclo che fa ripetere per dieci volte lo stesso gruppo di operazioni, finché non si verifica la condizione $X > 10$.

Facciamo un altro esempio.

Problema 5

Dati cinque numeri, calcolarne la media.

Il diagramma di flusso del programma è in fig. 3.7.

Abbiamo costruito un ciclo che viene eseguito cinque volte: ogni volta ci viene chiesto un numero in ingresso e viene calcolata la somma progressiva dei vari numeri.

Vediamo più in dettaglio.

All'inizio poniamo uguale a 1 la variabile che funge da contatore, e uguale a zero la somma dei vari numeri; viene chiesto il valore del primo numero X , calcolata la sua somma con i numeri precedenti, nel primo passaggio quindi $S = 0 + X$. Poi K viene aumentato di 1, confrontato con 5 e, siccome $K = 2 < 5$, il programma salta indietro fino al punto precedente la richiesta di input.

Viene inserito il secondo numero, calcolata la sua somma con i numeri precedenti e così via finché dopo cinque passaggi all'interno del ciclo, cioè quando $K = 6$, il programma prosegue nel ramo di destra del simbolo di decisione.

Per calcolare la media M dei cinque numeri basta ora dividere la somma dei numeri, S , per 5; si stampa il risultato e il programma termina.

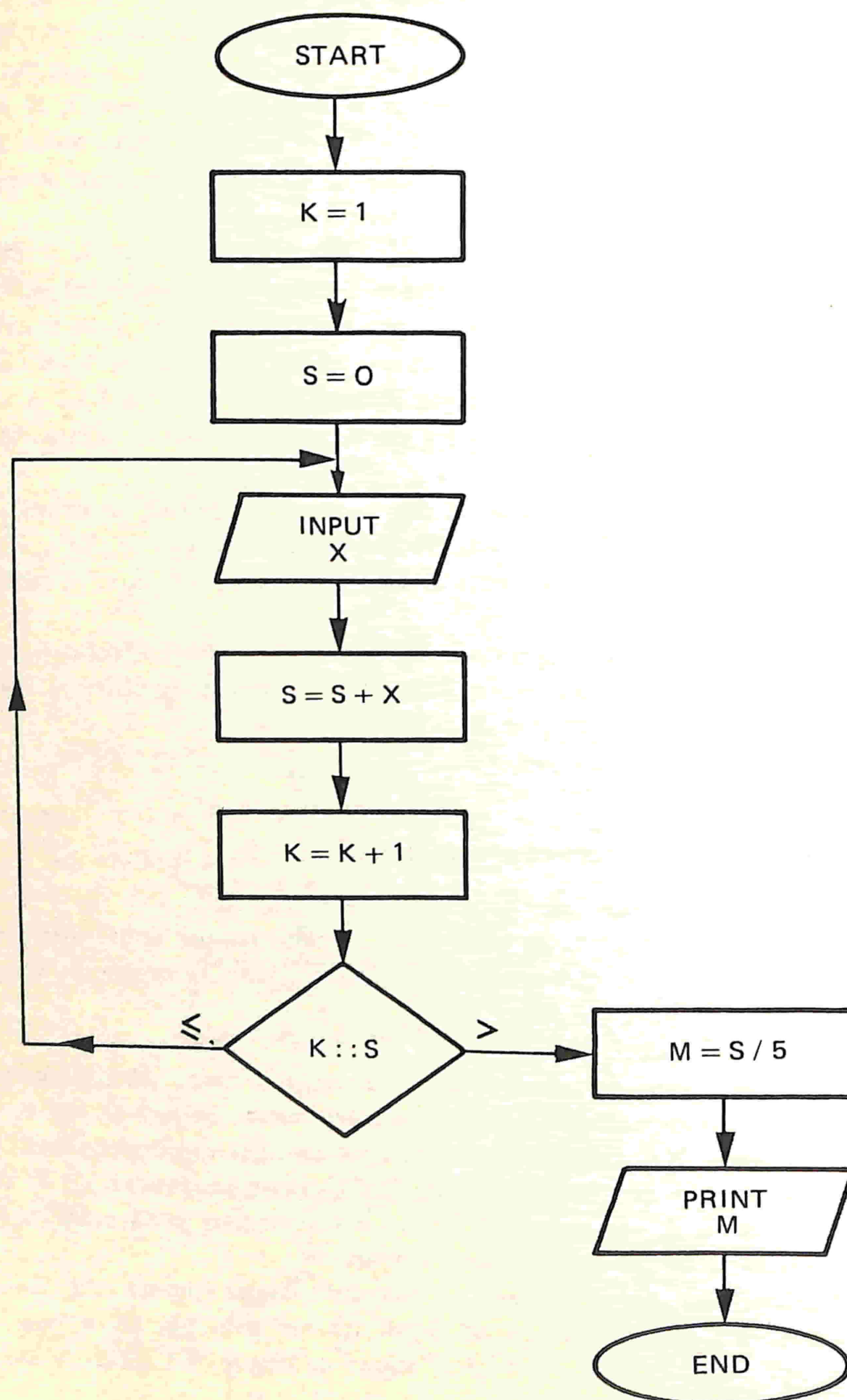


Fig. 3.7 Diagramma di flusso del problema 5: media di 5 numeri

CONCLUSIONE

Per riassumere questo capitolo sulla programmazione possiamo dire quindi che *l'idea fondamentale* che deve guidare chi costruisce un programma è di *organizzare il problema secondo una logica di tipo binario, dividendo il problema complessivo in una serie di problemi elementari e tenendo presente la possibilità di iterare gruppi di operazioni uguali.*

Prima di affrontare gli esercizi, voglio darvi un'ultima informazione sulla programmazione: i requisiti di un buon programmatore sono fantasia, pazienza e precisione.

Esercizi

Costruite i diagrammi di flusso dei problemi seguenti. Possibili soluzioni sono riportate in fondo al volume.

3.1. Pensate un programma che calcoli il volume di un parallelepipedo di dimensioni $B \times L \times H$.

3.2. Il salario di un muratore è di 15000 £/h per le prime 40 ore settimanali; le ore eccedenti vengono pagate al 50% in più come straordinario. Costruite un programma che, prendendo in ingresso il numero di ore lavorate da un muratore alla settimana, calcoli la relativa paga settimanale.

3.3. Inventate un programma che calcoli la media dei voti di una classe di N studenti. (Suggerimento: date come input N e costruite un ciclo che venga eseguito N volte.)

CAPITOLO QUATTRO

Impariamo il Basic risolvendo problemi di costruzioni

In questo capitolo, attraverso lo studio di un semplice problema di costruzioni, imparerete quella decina di istruzioni che costituiscono il nucleo del linguaggio Basic.

IL CALCOLO REAZIONI DI UNA TRAVE ISOSTATICA

Abbiamo già iniziato a studiare questo problema nel capitolo 3 relativo alla programmazione. Eravamo arrivati fino alla costruzione del diagramma di flusso relativo, riportato in fig. 3.4. Ora traduciamo questa rappresentazione grafica in Basic.

Come regola generale è comodo iniziare a scrivere un qualsiasi programma assegnando alla prima riga il numero 100 e alle successive numeri crescenti di 10 in 10. Questo ci permetterà in seguito, ove necessario, di inserire e aggiungere altre righe senza dover riscrivere continuamente tutti i numeri di linea.

Con riferimento ai simboli della fig. 3.3, il programma più semplice che traduce il nostro problema in Basic è il seguente:

```
100 INPUT L,P,D
110 RB=P*D/L
120 RA=P-RB
130 PRINT L,P,D
140 PRINT RA,RB
150 END
```

Digitate esattamente questo programma sul vostro computer, battendo alla fine di ciascuna linea il tasto RETURN. Il programma è ora nella

memoria del computer. Scrivendo il comando RUN, seguito da una battuta del tasto RETURN, il programma inizierà a girare.

Vedrete apparire sullo schermo un punto interrogativo: il computer attende che inseriate qualcosa dalla tastiera. Alla riga 100 infatti trova un comando di INPUT seguito dai nomi di tre variabili: il programma non prosegue finché non inserite tre valori numerici, separati l'uno dall'altro da virgole.

Scrivete i valori corrispondenti alla luce, al carico e alla distanza del carico dall'appoggio sinistro A, poi battete il tasto RETURN. Il computer assegnerà tali valori alle variabili chiamate, rispettivamente, L, P e D.

Le righe 110 e 120 assegnano alle variabili RA e RB i valori corrispondenti alle espressioni a destra del segno di =. Queste istruzioni, in Basic, si chiamano *istruzioni di assegnazione*.

Le righe 130 e 140 fanno stampare (PRINT) sul video i valori corrispondenti alle variabili L, P, D, spaziati di una certa quantità, e, alla riga successiva, i valori delle variabili RA e RB che costituiscono le soluzioni del problema.

La riga 150 indica al computer che il programma è terminato (END).

Avvertenza. Quando si scrive un programma al computer, specialmente se si è alle prime armi, è molto facile commettere degli errori. Difficilmente poi si riescono a prevedere all'inizio i casi particolari di dati per cui i risultati possono essere assurdi o dar luogo ad errori.

È quindi *indispensabile*, ogni volta che si scrive un programma, provarlo diverse volte, fargli cioè risolvere problemi di cui sono noti sia i dati iniziali che i risultati, in modo da controllarne il corretto funzionamento.

Vi consiglio di eseguire sempre questo tipo di verifica prima di considerare completata la stesura di un programma e di cominciare a usarlo per risolvere problemi reali.

MIGLIORIAMO L'INPUT E L'OUTPUT

Come per il progetto di una casa non esiste un'unica soluzione, ma esistono soluzioni più o meno funzionali, belle e brutte, così non c'è il programma perfetto, la soluzione unica.

Esistono però due tipi fondamentali di programmi: quelli che funzionano e producono risultati corretti e quelli che non funzionano o, peggio, producono risultati errati.

Il programma che abbiamo appena costruito funziona, cioè produce risultati esatti, ma, se l'avete fatto girare, avrete notato che ha diversi difetti. Per esempio le richieste di input sul video non sono chiare: potremmo facilmente confondere l'ordine in cui inserire i dati; nean-

che l'output è chiaro; mentre i valori di L e P possono essere qualsiasi, non sono accettabili dei valori della distanza D minori di zero o maggiori di L.

Scriviamo quindi un programma che elimini il primo difetto, cioè migliori le richieste di input, permettendoci di inserire i dati senza possibilità di errore.

Se avete il primo programma ancora in memoria, aggiungete le seguenti righe:

```
40 REM CALCOLO REAZIONI TRAVE APPOGGIATA PER CARICHI CONCENTRATI
50 PRINTCHR$(147)
60 PRINT"CALCOLO REAZIONI DI UNA TRAVE APPOGGIATA PER CARICO CONCENTRATO"
70 PRINT:PRINT
80 INPUT"LUCE DELLA TRAVE (IN M), L=";L
90 INPUT"VALORE DEL CARICO (IN KG), P=";P
100 INPUT"DISTANZA DI P DALL'APPOGGIO SINISTRO (IN M), D=";D
```

Digitate LIST e vedrete apparire il programma completo.

```
40 REM CALCOLO REAZIONI TRAVE APPOGGIATA PER CARICHI CONCENTRATI
50 PRINTCHR$(147)
60 PRINT"CALCOLO REAZIONI DI UNA TRAVE APPOGGIATA PER CARICO CONCENTRATO"
70 PRINT:PRINT
80 INPUT"LUCE DELLA TRAVE (IN M), L=";L
90 INPUT"VALORE DEL CARICO (IN KG), P=";P
100 INPUT"DISTANZA DI P DALL'APPOGGIO SINISTRO (IN M), D=";D
110 RB=P*D/L
120 RA=P-RB
130 PRINT L,P,D
140 PRINT RA,RB
150 END
```

Alla riga 40 troviamo una nuova istruzione: REM (da *remark*, nota). Si tratta di un'istruzione che non viene eseguita, ma serve da annotazione per chi scrive o legge il programma. Vi consiglio di usarla spesso, e comunque *sempre* all'inizio di un programma; a distanza di tempo infatti queste note sono comode per ricordare l'argomento del programma o di parti del programma particolarmente interessanti.

La riga 50 dà istruzione al computer di stampare la funzione CHR\$(147). Tale funzione, nella forma PRINT CHR\$(X), stampa il carattere corrispondente al valore X nel codice ASCII.

Se controllate nell'appendice C, vedrete che per $X = 147$ il corrispondente carattere ASCII è il tasto CLR/HOME shiftato. Quando il computer esegue la riga 110 quindi stampa un CLR/HOME shiftato, cioè cancella lo schermo e porta il cursore in alto a sinistra. Conviene sempre mettere all'inizio di un programma questa istruzione, in modo da cominciare l'esecuzione del programma con lo schermo libero da scritte precedenti.

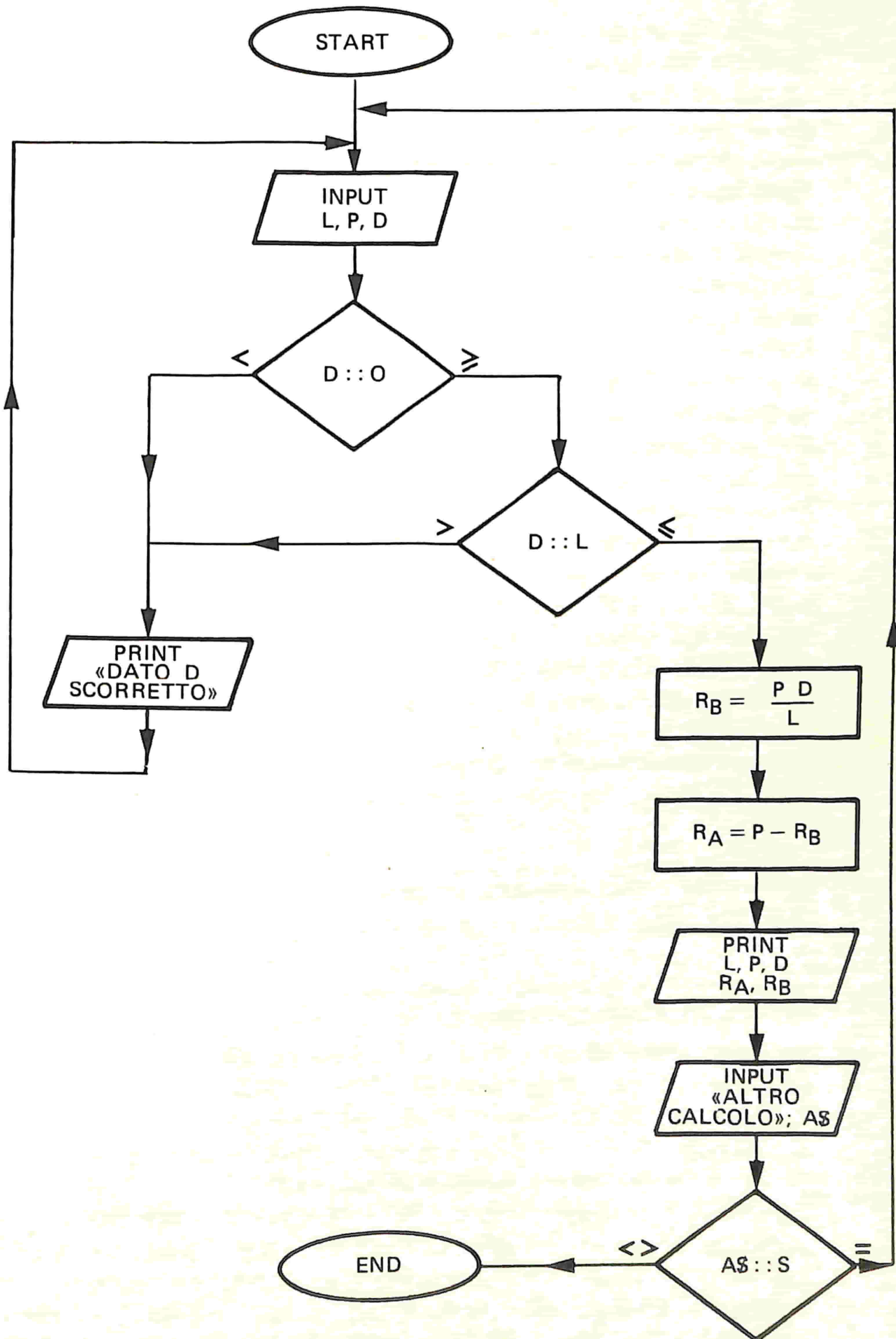


Fig. 4.1 Diagramma di flusso del programma 4.2: calcolo reazioni di una trave appoggiata soggetta a un carico concentrato

La riga 60 stampa la stringa (serie di caratteri alfabetici o numerici) racchiusa tra virgolette.

La riga 70 stampa due righe vuote, cioè sposta il cursore di due righe più in basso. Notate che i due punti (:) servono per separare due istruzioni successive scritte sulla stessa riga.

Nelle righe da 80 a 100 ritroviamo gli INPUT dell'esempio precedente, ma questa volta sul video compaiono prima le stringhe tra virgolette (per la riga 80, per esempio: LUCE DELLA TRAVE (IN M), L=), poi il punto interrogativo di richiesta dati: ora è molto difficile sbagliare ad introdurre i dati.

Eseguite un calcolo di prova e vedrete che, se l'INPUT è chiaro, non lo è ancora l'output; manca inoltre il controllo sul valore della distanza D che deve essere compresa tra zero e L.

Nel programma che segue, il cui flow-chart è riportato in fig. 4.1, sono aggiunte alcune righe che eliminano questi difetti.

Prima di scriverlo, cancellate il programma che avete in memoria con un comando NEW, e controllate con LIST se si è davvero cancellato. Oppure, se volete, prima di cancellare il vecchio programma, registratelo sulla vostra memoria di massa, cassetta o disco (vedi capitolo 2). Scrivete ora il nuovo programma, controllate se avete scritto tutto bene usando il comando LIST nelle sue varie possibilità (vedi appendice B), poi fatelo girare con dati che diano risultati noti.

```

100 REM CALCOLO REAZIONI TRAVE APPOGGIATA PER CARICHI CONCENTRATI
110 PRINTCHR$(147)
120 PRINT"CALCOLO REAZIONI DI UNA TRAVE APPOGGIATA PER CARICO CONCENTRATO"
130 PRINT:PRINT
140 INPUT"LUCE DELLA TRAVE (IN M), L=";L
150 INPUT"VALORE DEL CARICO (IN KG), P=";P
160 INPUT"DISTANZA DI P DALL'APPOGGIO SINISTRO (IN M), D=";D
170 IFD<0 THEN GOTO 300
180 IFD>L THEN GOTO 300
190 RB=P*D/L
200 RA=P-RB
210 PRINT:PRINT
220 PRINT"L="L"M", "P="P"KG", "D="D"M"
230 PRINT
240 PRINT"RA="RA"KG", "RB="RB"KG"
250 PRINT:PRINT
260 INPUT"ALTRO CALCOLO (S/N)";A$
270 IFA$="S" THEN GOTO 110
280 END
300 PRINT:PRINT:PRINT
310 PRINT"VALORE DELLA DISTANZA D SCORRETTO"
320 GOTO 130

```

Esempio

Per

$L = 5 \text{ M}$ $P = 10000 \text{ KG}$ $D = 2.5 \text{ M}$

$RA = 5000 \text{ KG}$ $RB = 5000 \text{ KG}$

Cambiando il valore di D in

$D = 1.5 \text{ M}$

$RA = 7000 \text{ KG}$ $RB = 3000 \text{ KG}$

Le righe da 100 a 160 sono uguali alle prime sette righe del programma precedente.

Alla linea 170 troviamo un'istruzione che richiede qualche spiegazione: l'istruzione IF... THEN... Questa istruzione serve per tradurre in Basic il simbolo di decisione o scelta binaria rappresentato nel diagramma di flusso. In italiano si potrebbe tradurre così: "se (IF) è vera la relazione scritta *allora* (THEN) fai quello che segue". Se invece la relazione è falsa il programma continua alla riga successiva. L'istruzione IF-THEN è molto importante e vi consiglio di studiarla bene nella scheda in appendice.

Alla riga 170 quindi il computer controlla se il valore della variabile D è minore di zero; se le cose stanno così, il programma continua eseguendo l'istruzione che trova dopo il THEN, cioè GO TO (GO TO = vai a) 300; il programma quindi *salta* alla riga 300. Il GO TO in questo caso è facoltativo: si può scrivere semplicemente THEN 300.

Qui stampa tre righe vuote, poi la scritta VALORE DELLA DISTANZA D SCORRETTO e a 320 trova il comando di tornare alla linea 130, dove chiede di nuovo i dati (vedete anche sul diagramma di flusso).

Se invece alla riga 170 il valore della distanza D si rivela esatto, cioè maggiore o uguale a zero, il programma prosegue alla riga 180. Viene effettuato l'altro controllo sul valore della distanza, e quindi il calcolo delle reazioni.

Dalla riga 210 alla 240 viene eseguita la stampa degli stessi output dell'esempio precedente, ma in maniera più chiara e leggibile, con le relative unità di misura.

Infine alla riga 260 il programma avanza un'ulteriore richiesta di input: se vogliamo effettuare o meno un altro calcolo. La risposta che darete sarà S per sì e un qualsiasi altro tasto alfanumerico per no.

Attenzione: finora abbiamo usato come dati solo valori numerici, per le variabili numeriche L, P, D; questa volta il nostro dato non è un numero, ma una lettera, cioè, in generale, una serie di caratteri alfanumerici, quindi una stringa. Come sapete le variabili che rappresentano stringhe sono definite con nomi che terminano con il simbolo di dollaro (\$). Chiamando la variabile A\$ ho quindi avvertito il computer di considerare ciò che viene presentato in ingresso come una stringa.

Nella linea 270 A\$ viene confrontata con la stringa S, se il confronto dà valore vero, cioè se vogliamo effettuare un altro calcolo, il programma continua con l'istruzione GO TO 110, altrimenti termina alla linea successiva.

IMPLEMENTIAMO IL CALCOLO DELLE REAZIONI

In questo paragrafo vedremo come sia possibile generalizzare il programma che abbiamo appena visto al caso di un numero qualsiasi di carichi concentrati.

Come ricorderete al capitolo 3 sulla programmazione, una delle possibilità offerte dal computer è quella di poter ripetere uno stesso gruppo di istruzioni per un certo numero di volte, finché non si verifichi una data condizione, dopo di che il programma riprende a scorrere in maniera lineare: è questa l'operazione di iterazione o, in inglese, di *looping*.

Per costruire dei cicli (*loop*) in Basic si usa la coppia di istruzioni FOR - NEXT.

A titolo di esempio vediamo un ciclo per l'input di N carichi concentrati:

```
100 FOR I=1 TO N
110 INPUT"VALORE DEL CARICO";P
120 INPUT"DISTANZA DA A";D
130 NEXT I
```

La riga 100 dice al computer: *per* (FOR) I che va da 1 a (TO) N ripeti il gruppo di istruzioni seguenti fino alla istruzione NEXT I (successivo I).

Supponendo ad esempio $N = 3$, il computer ci chiederà per tre volte di seguito

VALORE DEL CARICO ?

DISTANZA DA A?

La variabile I è un contatore che all'inizio vale 1. Quando il programma arriva alla riga 130, I viene incrementato di 1 e confrontato con N; se è minore o uguale a N il programma riprende dalla linea 100, se è maggiore di N il programma continua con la riga successiva alla 130.

Attenzione: se guardate bene il pezzo di programma appena scritto vedrete che c'è un problema: gli N valori diversi dei carichi (o dei bracci) vengono tutti assegnati alla variabile di nome P, ciascun nuovo valore sostituisce il precedente ed alla fine in P troveremo il valore dell'ultimo carico dato in ingresso.

Per risolvere questo problema bisogna far costruire al computer una tabella dei vari carichi chiamandoli P(1), P(2), ..., P(N).

<i>carichi</i>	<i>valore</i>
P(I)	in t
P(0)	3.5
P(1)	4.0
P(2)	...

Questa tabella in Basic si costruisce con un *vettore*: con la istruzione DIM P(N) si dice al computer di preparare (*dimensionare*) in memoria lo spazio per N + 1 variabili i cui nomi sono P(0), P(1), P(2), ..., P(N).

Questo tipo di variabili, raggruppate in un vettore, si chiamano variabili indiciate, cioè con indice (in inglese *array-variable* o *subscripted variable*).

Quindi il programma corretto per l'input di N carichi concentrati e delle loro distanze da un punto fisso è il seguente:

```

100 INPUT "QUANTI CARICHI";N
110 DIM P(N):DIM D(N)
120 FOR I=1 TO N
130 PRINT "INPUT DEL CARICO N°I"
140 INPUT "VALORE DEL CARICO (IN KG)";P(I)
150 INPUT "DISTANZA DA A (IN M)";D(I)
160 NEXT I

```

Vediamo ora un programma che calcola le reazioni di una trave appoggiata soggetta a un numero qualsiasi di carichi concentrati.

```

100 REM CALCOLO REAZIONI TRAVE APPOGGIATA PER N CARICHI CONCENTRATI
110 PRINT CHR$(147)
120 PRINT "CALCOLO REAZIONI TRAVE APPOGGIATA PER N CARICHI CONCENTRATI"
130 PRINT:PRINT
140 INPUT "LUCE TRAVE (IN M), L=";L
150 PRINT
160 REM INPUT CARICHI
170 INPUT "QUANTI CARICHI";N
180 DIM P(N),D(N)
185 PRINT:PRINT
190 FOR I=1 TO N
200 PRINT "INPUT CARICO N°I"
210 INPUT "VALORE DEL CARICO (IN KG)";P(I)
220 INPUT "DISTANZA DA A (IN M)";D(I)
230 PRINT
240 NEXT I
250 REM CALCOLO REAZIONI
260 RA=0:RB=0
270 FOR I=1 TO N
280 RB=RB+P(I)*D(I)/L
290 RA=RA+P(I)*(L-D(I))/L
300 NEXT I
310 REM VERIFICA SOMMATORIA FORZE VERTICALI
315 PP=0

```



```

320 FOR I=1 TO N
330 PP=PP+P(I)
340 NEXT I
350 Y=PP-RA-RB
360 IFABS(Y)>0.05 THEN GOTO 520
370 REM STAMPA INPUTS
380 PRINTCHR$(147)
390 PRINT"LUNGHEZZA TRAVE L="L
400 PRINT
410 FOR I=1 TO N
420 PRINT"P("I")="P(I)"KG", "D("I")="D(I)"M"
430 NEXT I
440 REM STAMPA OUTPUTS
450 PRINT
460 PRINT"RA="RA"KG", "RB="RB"KG"
470 PRINT:PRINT
480 REM ALTRO CALCOLO
490 INPUT"ALTRO CALCOLO (S/N)";A$
500 IFA$="S" THEN CLR:GOTO 110
510 END
520 REM SOMMATORIA FORZE VERTICALI <>0
530 PRINT:PRINT:PRINT
540 PRINT"VERIFICA ERRATA: RIPROCEDERE"

```

Dopo aver scritto e fatto girare per prova il programma ho inserito alcune REM per renderne più chiara la lettura.

Fino alla riga 240 credo non ci sia bisogno di ulteriori spiegazioni.

Alla riga 260 vengono azzerate le variabili RA e RB: non sarebbe strettamente necessario, ma è una garanzia di sicurezza in più quando si eseguono più calcoli successivi.

Dalla riga 270 alla 300 troviamo il ciclo di calcolo delle reazioni. Le due reazioni vengono calcolate per sovrapposizione degli effetti, ripetendo il calcolo delle reazioni per un singolo carico N volte, quant'è il numero dei carichi, e sommando i successivi valori.

Dalla riga 310 alla 360 viene effettuata per controllo la sommatoria delle forze verticali. Dapprima, da 320 a 340, con un ciclo vengono sommati tutti i carichi contenuti nel vettore P(N), assegnando il valore risultante alla variabile PP. Poi, alla linea 350, PP viene sommata algebricamente con le reazioni e infine, alla riga 360 viene effettuato il controllo del valore della sommatoria forze verticali.

Il computer calcola i valori numerici con la precisione di 10 cifre significative. In questo caso ho ritenuto sufficiente una approssimazione alla seconda cifra decimale, ho imposto quindi che se la sommatoria delle forze verticali Y è, in valore assoluto, maggiore di 0.05 i risultati non sono da considerare accettabili. La funzione ABS(X) calcola il valore assoluto dell'espressione o variabile contenuta tra parentesi.

Se la verifica non è soddisfatta il programma salta alla riga 520 dove viene stampata la frase: VERIFICA ERRATA: RIPROCEDERE.

Se al contrario la verifica è accettabile il programma prosegue con la stampa degli input.

Alla riga 380 ho fatto cancellare lo schermo per ripulirlo dalle scritte precedenti.

Gli input vengono stampati dalla riga 390 alla 430. Segue poi la stampa degli output e la richiesta di altro calcolo.

Notate che alla riga 500 è presente una nuova istruzione CLR (dal verbo *to clear*, liberare, cancellare). Se è richiesto un altro calcolo, se cioè $A\$ = "S"$, viene eseguita l'istruzione CLR, dopo di che il programma riprende dalla riga 110.

L'istruzione CLR azzerava tutte le variabili, sia numeriche che di stringa, e libera tutti gli spazi predisposti per i vettori: dopo la sua esecuzione è come se il programma girasse per la prima volta. È buona norma inserirla in un programma quando questo può venire usato per più calcoli successivi.

Esercizi

Scrivete i programmi in Basic che risolvono i seguenti problemi. Possibili soluzioni sono riportate in fondo al volume.

4.1. Calcolare il volume di un parallelepipedo di dimensioni $B \times L \times H$.

4.2. Il salario di un muratore è di 15000 £/h per le prime 40 ore settimanali; le ore eccedenti vengono pagate al 50% in più come straordinario. Costruite il programma che calcoli la paga settimanale di un muratore per un qualsiasi numero di ore lavorate.

4.3. Calcolare la media dei voti di una classe di N studenti.

Programmi completi

In questo capitolo sono riportati programmi che risolvono problemi di estimo, costruzioni e topografia. Sono programmi che possono essere costruiti con le istruzioni studiate al capitolo 4.

Viene introdotta solo una nuova istruzione Basic: l'istruzione ON ... GOTO Tale istruzione, spiegata nel programma 5.4, permette di saltare in diversi punti del programma a seconda del valore dell'espressione che segue la preposizione ON, permette cioè l'*estensione multipla*.

Per ogni programma vengono forniti il listato, la descrizione del funzionamento e una spiegazione accompagnata, quando necessario, dal relativo diagramma di flusso.

Per l'estimo sono riportati due programmi che servono, rispettivamente, a calcolare l'interesse composto maturato da un capitale in un certo numero di anni e il valore potenziale di un fondo a cui vengono apportate determinate migliorie.

I programmi di topografia sono relativi a una applicazione del teorema dei seni e al problema dell'intersezione in avanti.

Il primo dei due programmi di costruzioni permette di calcolare le reazioni di una trave appoggiata soggetta ad un numero qualsiasi di carichi concentrati e uniformemente distribuiti. Il secondo serve invece a determinare le sollecitazioni di taglio e momento flettente in diverse sezioni di una trave appoggiata soggetta a un numero qualsiasi di carichi concentrati.

Qualche consiglio

Provate a modificare e migliorare i programmi riportati per risolvere i vostri problemi specifici. Provate anche a costruire da soli programmi relativi ad altri problemi, anche non pertinenti al campo del geometra. Non scoraggiatevi alle prime difficoltà e ricordatevi che difficilmente un programma gira perfettamente al primo colpo; c'è sempre bisogno di qualche correzione e modifica.

PROGRAMMA 5.1: CALCOLO INTERESSE COMPOSTO

Obiettivo: calcolare l'interesse maturato da un dato capitale in un certo numero di anni; calcolare inoltre il montante, cioè il capitale finale dato dalla somma del capitale iniziale più gli interessi.

Esempio

Un capitale di £ 2000000 viene investito, per un periodo di 3 anni, in una attività economica che fornisce un interesse netto del 16% annuo. Calcolare gli interessi maturati ed il montante alla fine del periodo considerato.

I dati da inserire nel programma sono

SAGGIO DI INTERESSE (IN DECIMALI)	0.16
CAPITALE INIZIALE	2000000
NUMERO ANNI	3

I risultati sono

INTERESSE COMPOSTO	= 1121792
MONTANTE	= 3121792

Lista del programma

```

100 REM CALCOLO INTERESSE COMPOSTO
110 PRINTCHR$(147)
120 PRINT"CALCOLO INTERESSE COMPOSTO"
130 PRINT:PRINT
140 INPUT"SAGGIO DI INTERESSE (IN DECIMALI)";R
150 PRINT
160 INPUT"CAPITALE INIZIALE";C0
170 PRINT
180 INPUT"NUMERO ANNI";N
190 PRINT
200 Q=1+R
210 IC=C0*(Q^N-1)

```

```

220 M=C0*Q↑N
230 PRINT:PRINT
240 PRINT"INTERESSE COMPOSTO="IC
250 PRINT
260 PRINT"MONTANTE="M
270 PRINT:PRINT
280 INPUT"ALTRO CALCOLO (S/N)";A$
290 IFA$="S"THEN CLR:GOTO110
300 END

```

Descrizione. Il programma chiede in input il saggio di interesse (R), il capitale iniziale (C0) ed il numero di anni (N) per cui si vogliono calcolare gli interessi. Fornisce in uscita l'interesse composto e il montante.

Commento al programma. Fino alla riga 190 vengono eseguiti gli input. I calcoli sono svolti nelle righe 200–220 con le formule

$$Q = 1 + R$$

$$IC = C0 * (Q \uparrow N - 1)$$

$$M = C0 * Q \uparrow N$$

Seguono la stampa degli output e la richiesta di altro calcolo.

PROGRAMMA 5.2: VALORE POTENZIALE DI UN FONDO

Obiettivo: determinare qual è il valore potenziale di un fondo a cui vengano apportate determinate migliorie in un certo numero di anni.

Esempio

Un fondo fornisce attualmente un beneficio fondiario di £ 1000000. Sono state progettate opere ordinarie di miglioramento che saranno compiute in 5 anni e porteranno il fondo a produrre un beneficio ordinario e continuativo, dal sesto anno in poi, di £ 1300000.

Le opere progettate richiedono un costo annuo posticipato $C = 800000$ £. Essendo il saggio di interesse $R = 0.05$ il costo totale delle migliorie riferito all'anno zero risulta essere

$$\begin{aligned}
 K0 &= C * (Q \uparrow N - 1) / (R * Q \uparrow N) = \\
 &= 800000 * (1.05 \uparrow 5 - 1) / (0.05 \times 1.05 \uparrow 5) = \\
 &= 3464000 \text{ £}
 \end{aligned}$$

Si vuole determinare il valore potenziale del fondo all'attualità: I dati da inserire nel programma sono

BENEFICIO FONDIARIO ATTUALE	1000000
BENEFICIO FONDIARIO POTENZIALE	1300000
COSTO MIGLIORIE	3464000
SAGGIO DI INTERESSE (IN DECIMALI)	0.05
ANNI	5

Il risultato è

VALORE POTENZIALE DEL FONDO VPA 21237157

Nota: come unità di misura del capitale può essere usato, invece della lira, il milione di lire, ciò facilita, in questo caso, l'introduzione dei dati e la lettura dei risultati.

Lista del programma

```

100 REM VALORE POTENZIALE DI UN FONDO
110 PRINTCHR$(147)
120 PRINT"VALORE POTENZIALE DI UN FONDO"
130 PRINT:PRINT
140 INPUT"BENEFICIO FONDIARIO ATTUALE";BA
150 PRINT
160 INPUT"BENEFICIO FONDIARIO POTENZIALE";BP
170 PRINT
180 INPUT"COSTO MIGLIORIE";K0
190 PRINT
200 INPUT"SAGGIO DI INTERESSE (IN DECIMALI)";R
210 PRINT
220 INPUT"ANNI";N
230 PRINT
240 INPUT"I DATI SONO ESATTI? (S/N)";A$
250 IF A$<>"S" THEN PRINTCHR$(147):GOTO140
260 Q=1+R
270 VA=BP/R-(BP-BA)*(Q^N-1)/(R*Q^N)-K0
280 PRINT:PRINT
290 PRINT"VALORE POTENZIALE DEL FONDO VPA"VA
300 PRINT:PRINT
310 INPUT"ALTRO CALCOLO (S/N)";A$
320 IF A$="S" THEN 110
330 END

```

Descrizione. Il programma richiede in input il beneficio fondiario attuale (BA), il beneficio fondiario potenziale (BP), il costo delle migliorie (K0), il saggio di interesse (R) e il numero di anni per cui vengono effettuate le migliorie (N).

Visto che i dati sono in un certo numero ho previsto la possibilità di richiedere l'input se ci si accorge che uno dei dati è errato.

Seguono, come al solito, la stampa degli output e la richiesta di altro calcolo.

Commento al programma. La struttura del programma è identica a quella del programma precedente. I calcoli vengono eseguiti alle righe 260 e 270 con le formule ivi riportate.

PROGRAMMA 5.3: TEOREMA DEI SENI

Obiettivo: in un triangolo sono dati un lato ed i due angoli adiacenti, determinare gli altri due lati (fig. 5.1).

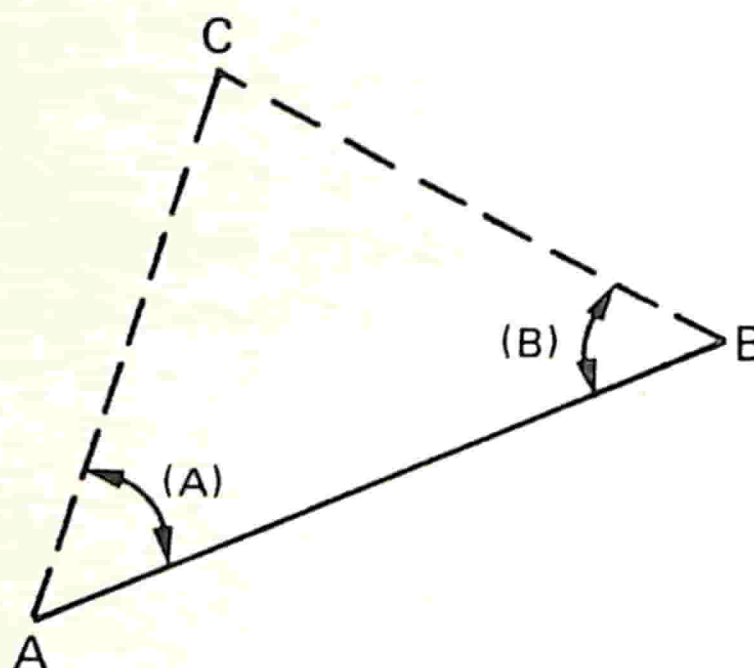


Fig. 5.1 Applicazione del teorema dei seni: noto il lato AB e i due angoli (A) e (B), determinare i lati AC e BC

Esempio

Un lato di un triangolo è lungo 5 cm, i due angoli adiacenti al lato valgono rispettivamente 30 e 60 gradi sessagesimali.

Determinare gli altri due lati del triangolo.

I dati da inserire nel programma sono

LUNGHEZZA DEL LATO AB	5
PRIMO ANGOLO (IN GR. SESSAGESIMALI)	30
SECONDO ANGOLO (IN GR. SESSAGESIMALI)	60

I risultati sono

SECONDO LATO AC = 4.33012702

TERZO LATO BC = 2.5

Lista del programma

```

100 REM TEOREMA DEI SENI
110 REM DATO UN LATO ED I DUE ANGOLI ADIACENTI DI UN TRIANGOLO QUALSIASI
120 REM CALCOLARE GLI ALTRI DUE LATI

```



```

130 PRINTCHR$(147)
140 PRINT"APPLICAZIONE DEL TEOREMA DEI SENI"
150 PRINT:PRINT
155 REM INPUT DATI
160 INPUT"LUNGHEZZA DEL LATO AB";AB
170 PRINT
180 INPUT"PRIMO ANGOLO (IN GR. SESSADECIMALI)";A
190 AR=A* $\pi$ /180
200 PRINT
210 INPUT"SECONDO ANGOLO (IN GR. SESSADECIMALI)";B
220 BR=B* $\pi$ /180
225 REM CALCOLO LATI
230 AC=AB*SIN(BR)/SIN(AR+BR)
240 BC=AB*SIN(AR)/SIN(AR+BR)
245 REM STAMPA OUTPUTS
250 PRINTCHR$(147)
260 PRINT"DATI INIZIALI"
270 PRINT
280 PRINT"LUNGHEZZA LATO AB="AB
290 PRINT"PRIMO ANGOLO"A
300 PRINT"SECONDO ANGOLO"B
310 PRINT:PRINT
320 PRINT"STAMPA RISULTATI"
330 PRINT
340 PRINT"SECONDO LATO AC="AC
350 PRINT"TERZO LATO BC="BC
360 PRINT:PRINT:PRINT
370 INPUT"ALTRO CALCOLO (S/N)";A$
380 IF A$="S" THEN 130
390 END

```

Descrizione. Gli input richiesti sono la lunghezza del lato (AB) e i valori dei due angoli in gradi sessagesimali ((A), (B)).

Eseguito il calcolo viene pulito lo schermo e sono stampati sia i dati iniziali, per controllo, sia i risultati: i due lati da determinare. Segue la richiesta di altro calcolo.

Commento al programma. Alle righe 190 e 220 viene effettuata la trasformazione degli angoli presentati in ingresso da gradi sessagesimali a radianti. Il computer infatti calcola le funzioni trigonometriche con argomenti in radianti.

Il calcolo dei due lati incogniti viene effettuato alle righe 230 e 240 applicando il teorema dei seni. La funzione SIN(X) calcola il seno della variabile o espressione X espressa in radianti (vedi appendice B).

PROGRAMMA 5.4: INTERSEZIONE IN AVANTI

Obiettivo: date le coordinate di due punti A e B e gli angoli misurati dalla base AB verso un terzo punto P, determinare le coordinate di P (fig. 5.2).

Nota: i vertici A, B, P, devono essere disposti in senso antiorario.

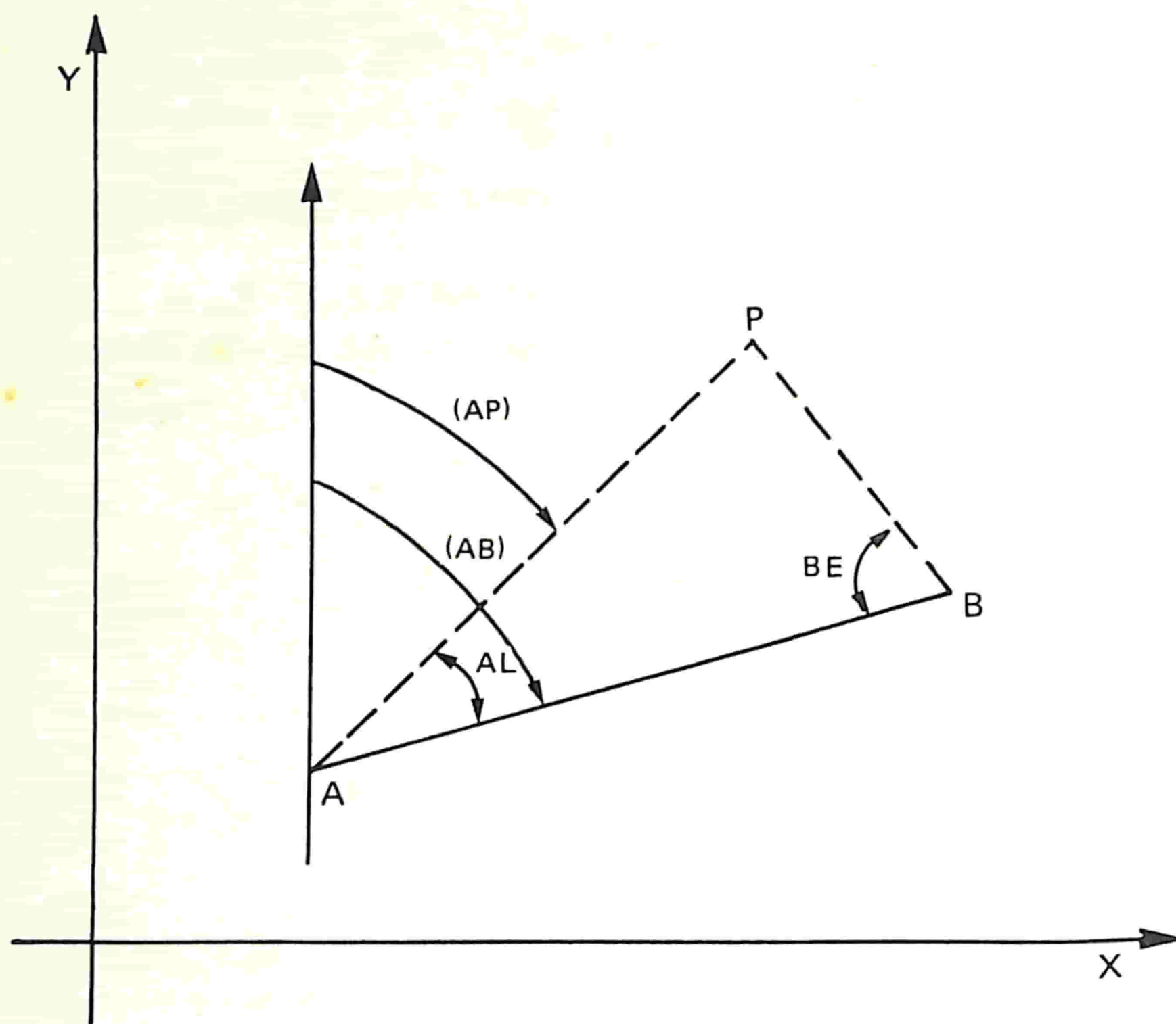


Fig. 5.2 Intersezione in avanti: note le coordinate dei punti A e B e gli angoli AL e BE, determinare le coordinate del punto P

Esempio

Le coordinate di due punti A e B sono, rispettivamente, 0,0 e 3,2. Gli angoli misurati dalla base AB verso un terzo punto P valgono rispettivamente $AL = 15$ e $BE = 90$ gradi centesimali.

Determinare le coordinate del punto P.

I dati da inserire nel programma sono

COORDINATE DEL PUNTO A(X,Y)	0,0
COORDINATE DEL PUNTO B(X,Y)	3,2
ANGOLO IN A (GR. CENT.)	15
ANGOLO IN B (GR. CENT.)	90

I risultati sono

ANGOLO DI DIREZIONE (AB) = 62.567

DISTANZA AP = 3.572

DISTANZA BP = .844

COORDINATE DEL PUNTO P

XP = 2.428 YP = 2.621

Lista del programma

```

100 REM INTERSEZIONE IN AVANTI
110 PRINTCHR$(147)
120 PRINT"INTERSEZIONE IN AVANTI"
130 PRINT:PRINT
140 INPUT"COORDINATE DEL PUNTO A (X,Y)";XA,YA
150 PRINT
160 INPUT"COORDINATE DEL PUNTO B (X,Y)";XB,YB
170 PRINT
180 INPUT"ANGOLO IN A (GR.CENT.)";K1
190 PRINT:AL=K1*π/200
200 INPUT"ANGOLO IN B (GR.CENT.)";K2
210 PRINT:BE=K2*π/200
220 INPUT"I DATI SONO ESATTI (S/N)";A$
230 IFA$<>"S"THEN110
240 IFXB-XA=0ANDYB-YA=0 THENPRINT"DATI ERRATI":GOTO130
245 REM INDIVIDUAZIONE DEL QUADRANTE
250 IFXB-XA>0ANDYB-YA>0THENQ=1:GOTO300
260 IFXB-XA>0ANDYB-YA<0THENQ=2:GOTO300
270 IFXB-XA<0ANDYB-YA<0THENQ=3:GOTO300
280 Q=4
290 REM CONTROLLO ESTREMI QUADRANTE (A π/2 E 3π/2)
300 IFYB-YA=0ANDQ=1THENA1=π/2:GOTO370
310 IFYB-YA=0ANDQ=3THENA1=3*π/2:GOTO370
320 REM CALCOLO COORDINATE DEL PUNTO P
330 A1=ATN((XB-XA)/(YB-YA))
340 ONQGOTO370,350,350,360
350 A1=A1+π:GOTO370
360 A1=A1+2*π:GOTO370
370 IFA1=0ORA1=π THENAB=ABS(YB-YA):GOTO390
380 AB=(XB-XA)/SIN(A1)
390 AP=AB*SIN(BE)/SIN(AL+BE)
400 BP=AB*SIN(AL)/SIN(AL+BE)
410 A2=A1-AL
420 XP=XA+AP*SIN(A2)
430 YP=YA+AP*COS(A2)
440 PRINTCHR$(147)
450 PRINT"STAMPA RISULTATI"
460 PRINT:PRINT
470 PRINT"ANGOLO DI DIREZIONE (AB)="INT(A1*1000*200/π+.5)/1000
480 PRINT
490 PRINT"DISTANZA AP="INT(AP*1000+.5)/1000
500 PRINT
510 PRINT"DISTANZA BP="INT(BP*1000+.5)/1000
520 PRINT
530 PRINT"COORDINATE DEL PUNTO P:"
540 PRINT:PRINT"XP="INT(XP*1000+.5)/1000,"YP="INT(YP*1000+.5)/1000
550 PRINT:PRINT
560 INPUT"ALTRO CALCOLO (S/N)";B$
570 IFB$="S"THEN CLR:GOTO110
580 END

```


Elenco delle variabili principali

XA, YA	coordinate di A
XB, YB	coordinate di B
K1	angolo in A (gradi centesimali)
AL	angolo in A (radianti)
K2	angolo in B (gradi centesimali)
BE	angolo in B (radianti)
Q	indicatore del quadrante
A1	angolo di direzione (AB)
AB	distanza AB
AP	distanza AP
BP	distanza BP
A2	angolo di direzione (AP)
XP, YP	coordinate di P

Descrizione. Il programma richiede in input le coordinate dei punti A e B e degli angoli AL e BE (fig. 5.2) espressi in gradi centesimali. Fornisce come output l'angolo di direzione (AB), le distanze AP e BP e le coordinate del punto P. Tutti gli output vengono forniti con l'approssimazione di tre cifre decimali.

Commento al programma. Il programma inizia con le richieste di input delle coordinate di A e B, nella forma X,Y, cioè ascissa, ordinata. Prosegue quindi alla riga 180 con la richiesta del valore dell'angolo AL (fig. 5.2); dato che il computer calcola le funzioni trigonometriche in radianti dobbiamo trasformare l'angolo AL, chiamato provvisoriamente K1, in radianti: il che viene fatto alla riga 190. Lo stesso avviene per l'angolo BE alla riga 210.

Vengono quindi effettuati una serie di controlli sui dati:

- alla riga 240 si controlla che i due punti A e B non siano coincidenti;
- dalla riga 245 alla 280 viene controllato il quadrante in cui operiamo, ed alla variabile Q viene assegnato un numero corrispondente al quadrante;
- dalla riga 290 alla 310 si controlla se siamo nelle posizioni limite del quadrante, a $\pi/2$ e $3\pi/2$.

Alla riga 330 inizia il calcolo delle coordinate del punto P che termina alla riga 430.

Una nuova istruzione Basic viene introdotta alla riga 340: l'istruzione ON ... GOTO

Tale istruzione permette di saltare in vari punti del programma a seconda del valore dell'espressione che segue la preposizione ON, viene perciò detta di *estensione multipla* (*multiple branch*). In particolare, se l'espressione che segue ON, nel nostro caso la variabile Q, vale zero, viene eseguita la riga successiva, cioè la 350. Se Q vale 1, 2, 3, 4 il programma salta (GOTO) rispettivamente a 370, 350, 350, 360, come specificato dalla serie di numeri che seguano il GOTO. Se Q vale più di 4 viene invece eseguita, come nel caso di $Q = 0$, la riga successiva, cioè la 350.

Tralascio la spiegazione dei calcoli fino alla riga 430 (conoscendo un po' di topografia e consultando l'appendice B per le funzioni SIN(X), COS(X) e ATN(X) li capirete da soli) per spiegarvi invece un altro punto interessante che troviamo alla riga 470.

Nell'istruzione PRINT, dopo la stringa "ANGOLO DI DIREZIONE (AB)", troviamo una lunga espressione: il risultato in stampa sarà il valore della variabile A1, cioè dell'angolo di direzione (AB), trasformato in gradi centesimali e approssimato con tre cifre decimali.

Vediamo in dettaglio: la funzione INT(X) (*INTEger* = numero intero) trasforma il valore di X nel più grande numero intero minore o uguale a X. Ad esempio, se X vale 4.3, -5.321, 6.99, INT(X) vale rispettivamente 4, -6, 6.

Nella pratica tecnica l'arrotondamento si esegue approssimando la penultima cifra significativa al valore superiore se l'ultima cifra è maggiore di 5, al valore che ha se l'ultima cifra è minore di 5.

Visto come opera la funzione INT(X), per approssimare alla terza cifra decimale moltiplichiamo quindi il valore della nostra variabile per 1000, aggiungiamo 0.5, calcoliamo l'INT del risultato e successivamente dividiamo per 1000.

La stessa approssimazione viene applicata a tutti gli altri valori che vengono stampati in uscita.

PROGRAMMA 5.5: REAZIONI TRAVI APPOGGIATE PER CARICHI QUALSIASI

Obiettivo: calcolare le reazioni di una trave semplicemente appoggiata soggetta ad un numero qualsiasi di carichi concentrati e di carichi uniformemente distribuiti.

Esempio

Una trave semplicemente appoggiata di luce $L = 5$ m è soggetta ad un carico distribuito su tutta la sua lunghezza di valore $q = 1500$ kg/m

e a due carichi concentrati di 2000 kg rispettivamente a distanza 1.5 m e 3.5 m dall'appoggio sinistro A.

Determinare le reazioni dei due appoggi.

I dati da inserire nel programma sono

LUCE TRAVE (IN M) L =	5
NUMERO CARICHI CONCENTRATI	2
INPUT CARICO N 1	
VALORE DEL CARICO (IN KG)	2000
VALORE DELLA DISTANZA (IN M)	1.5
INPUT CARICO N 2	
VALORE DEL CARICO (IN KG)	2000
VALORE DELLA DISTANZA (IN M)	3.5
NUMERO CARICHI DISTRIBUITI	1
VALORE CARICO DISTR. N 1 (IN KG/M)	1500
ASCISSA INIZIALE (IN M)	0
ASCISSA FINALE (IN M)	5

I risultati sono

REAZIONE IN A = 5750 KG

REAZIONE IN B = 5750 KG

Lista del programma

```

100 REM REAZIONI TRAVE APPOGGIATA PER CARICHI QUALSIASI
110 PRINTCHR$(147)
120 PRINT"CALCOLO REAZIONI TRAVE APPOGGIATA"
130 PRINT"PER CARICHI QUALSIASI"
135 PRINT:PRINT
140 INPUT"LUCE TRAVE (IN M) L=";L
150 PRINT:PRINT
160 INPUT"NUMERO CARICHI CONCENTRATI";N
170 NP=N+2
180 DIM P(NP),A(NP)
185 IFN=0THEN260
190 FORI=1TON
200 PRINT
210 PRINT"INPUT CARICO N" I
220 INPUT"VALORE DEL CARICO (IN KG)";P(I)
230 INPUT"VALORE DELLA DISTANZA (IN M)";A(I)
240 NEXTI
245 PRINT:PRINT
250 INPUT"I DATI SONO ESATTI (S/N)";A$
255 IFA$<>"S"THEN150
260 PRINTCHR$(147)
270 INPUT"NUMERO CARICHI DISTRIBUITI";C

```



```

275 IFC=0THEN380
280 DIM Q(C),X1(C),X2(C),W(C),BR(C)
290 FORI=1TOC
300 PRINT
310 PRINT"VALORE CARICO DISTR. N°I"(IN KG/M)":INPUTQ(I)
320 INPUT"ASCISSA INIZIALE (IN M)":X1(I)
330 INPUT"ASCISSA FINALE (IN M)":X2(I)
340 NEXTI
350 PRINT:PRINT
360 INPUT"I DATI SONO ESATTI (S/N)":B$
370 IF B$<>"S"THEN260
380 FORI=1TOC
390 W(I)=X2(I)-X1(I)
400 BR(I)=(X1(I)+X2(I))/2
410 NEXTI
420 RA=0:RB=0
430 FORI=1TON
440 RB=RB+P(I)*A(I)/L
450 RA=RA+P(I)*(L-A(I))/L
460 NEXTI
470 FORI=1TOC
480 RB=RB+Q(I)*W(I)*BR(I)/L
490 RA=RA+Q(I)*W(I)*(L-BR(I))/L
500 NEXTI
510 Y=0:P(NP-1)=-RA:P(NP)=-RB
520 A(NP-1)=0:A(NP)=L
530 FORI=1TONP
540 Y=Y+P(I)
550 NEXTI
560 FORI=1TOC
570 Y=Y+Q(I)*W(I)
580 NEXTI
590 IFABS(Y)>0.05THEN 700
600 PRINTCHR$(147)
610 PRINT"REAZIONE IN A="RA"KG"
620 PRINT
630 PRINT"REAZIONE IN B="RB"KG"
640 PRINT:PRINT
650 INPUT"ALTRO CALCOLO (S/N)":C$
660 IFC$="S"THENCLR:GOTO110
670 END
700 PRINT:PRINT
710 PRINT"VERIFICA REAZIONI ERRATA: RIPROCEDERE"
720 FORI=1TO2000:NEXTI
730 GOTO110

```

Elenco delle variabili principali

L	luce della trave
N	numero carichi concentrati
NP	numero carichi concentrati + reazioni
C	numero carichi distribuiti
RA	reazione in A
RB	reazione in B
Y	sommatoria forze verticali

Elenco dei vettori

P(NP)	carichi concentrati + reazioni
A(NP)	bracci dei carichi concentrati + reazioni
Q(C)	carichi distribuiti
X1(C)	ascisse iniziali carichi distribuiti
X2(C)	ascisse finali carichi distribuiti
W(C)	lunghezza carichi distribuiti
BR(C)	bracci carichi distribuiti

Descrizione. Il programma richiede in input la luce della trave, i valori dei vari carichi concentrati e le loro distanze dall'appoggio A (fig. 5.3). Chiede inoltre i valori dei carichi distribuiti con le relative ascisse di inizio e fine carico. Sono previste due richieste di controllo della correttezza dei dati, al termine dell'input dei carichi concentrati e dei carichi distribuiti.

In output vengono fornite le reazioni in A e in B.

Commento al programma. Per spiegarvi questo programma ha pensato che invece di tante parole fosse meglio darvi la lista delle variabili e dei vettori usati e i due diagrammi di flusso che ho disegnato prima di

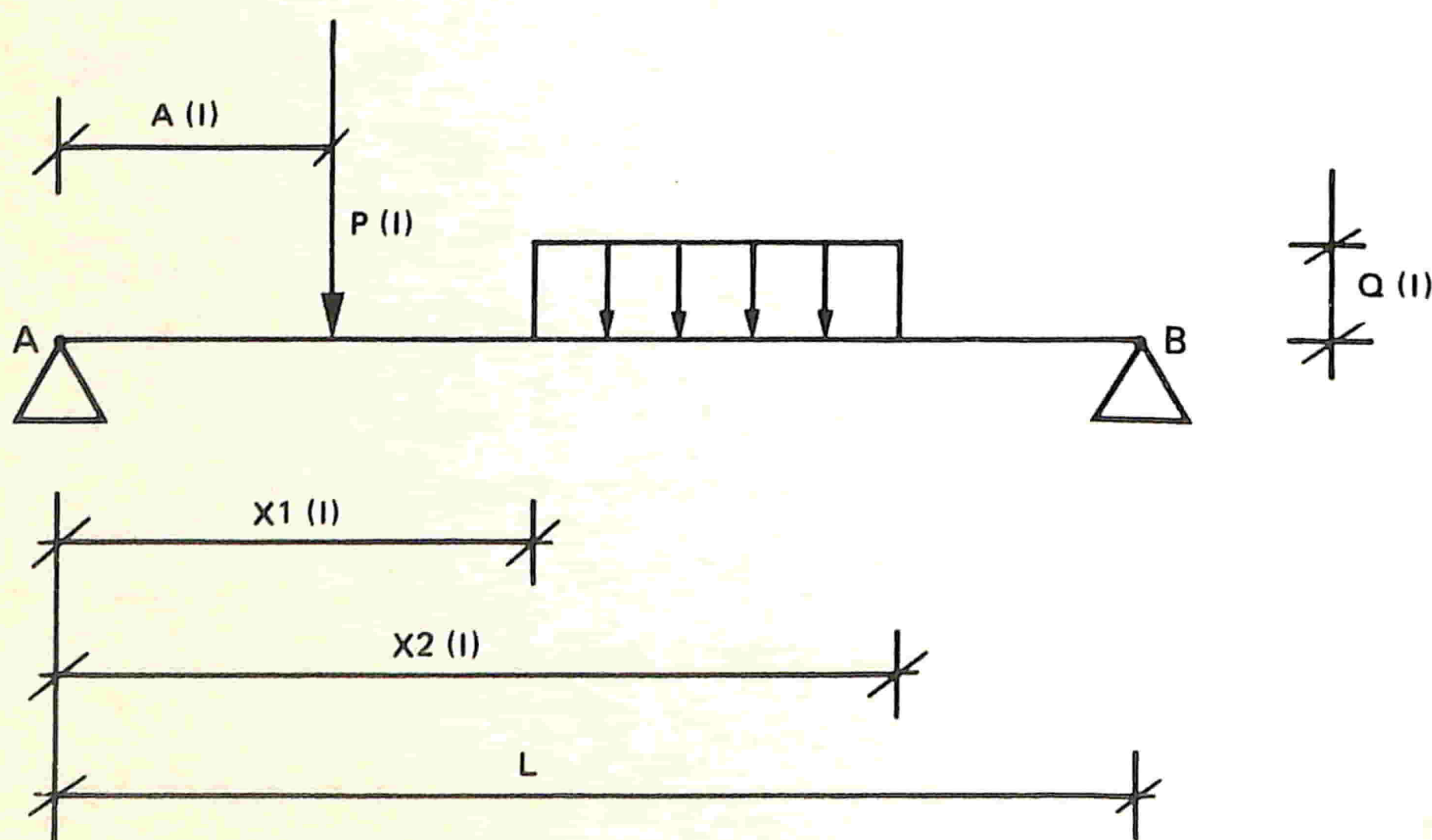


Fig. 5.3 Simboli usati nel programma 5.5: calcolo reazioni di una trave appoggiata per carichi qualsiasi

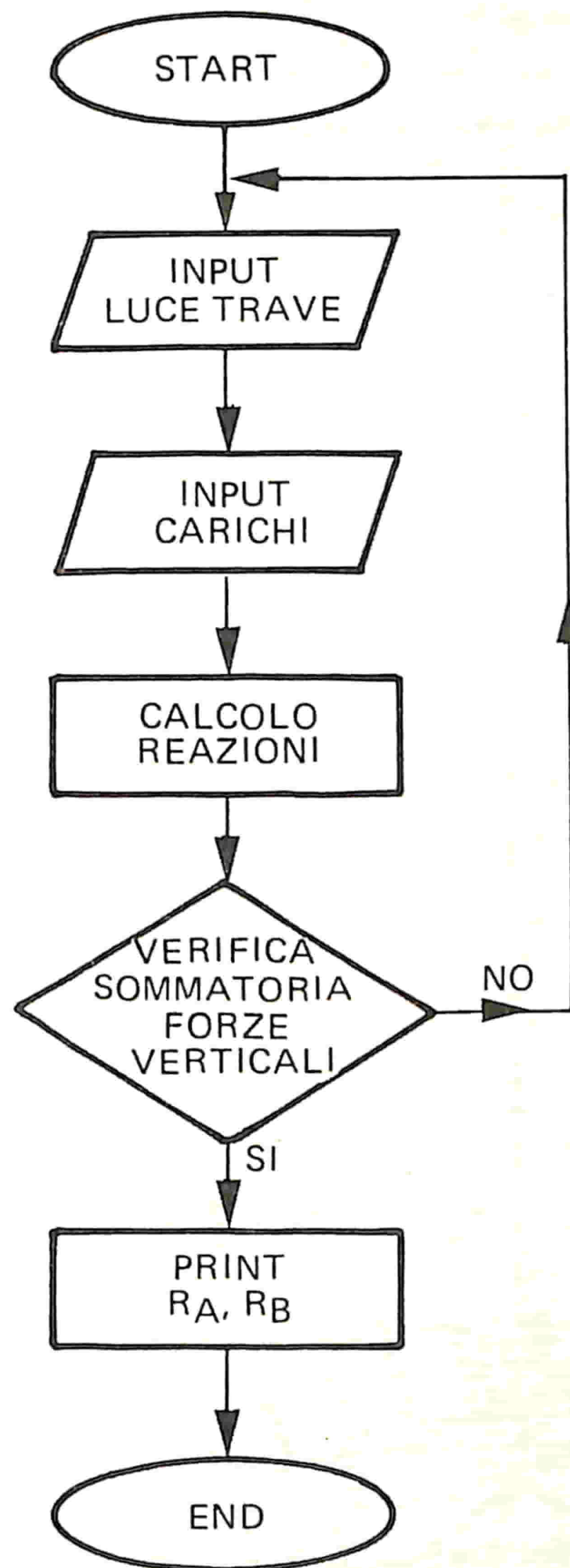


Fig. 5.4 Diagramma di flusso generale del programma 5.5

scrivere il programma. Il primo diagramma è a grandi blocchi e serve per inquadrare il problema (fig. 5.4).

Ogni blocco viene successivamente suddiviso in operazioni più semplici fino a ottenere il diagramma di flusso della fig. 5.5.

Questo procedimento pratico è quello di cui si parlava al capitolo 2 sulla programmazione, quando si diceva di “frammentare il problema principale in una successione di problemi elementari”.

Il grado di dettaglio del diagramma di flusso dipende dalla pratica del programmatore: per un programmatore poco esperto, un diagramma di flusso dettagliato significa meno errori nel programma.

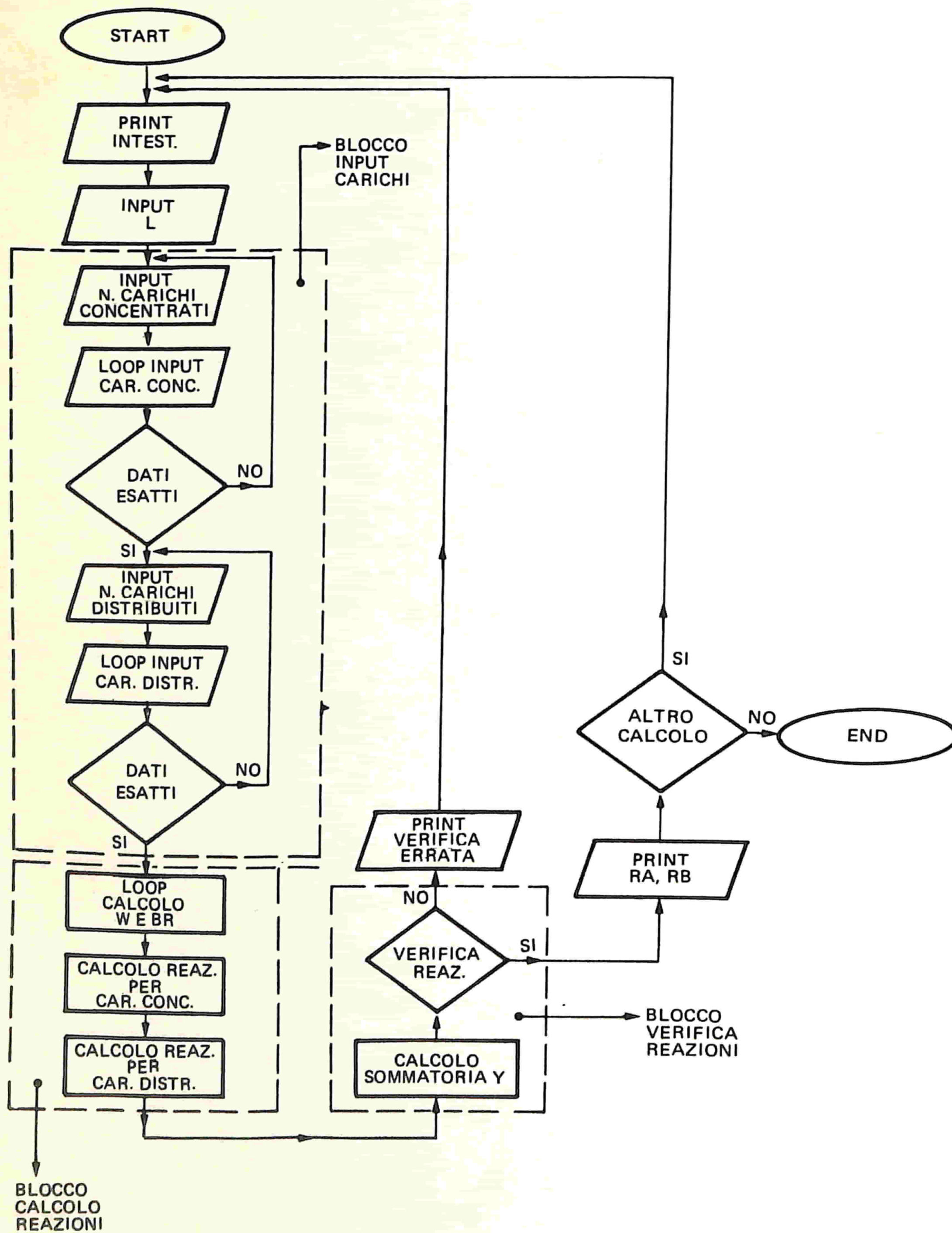


Fig. 5.5 Diagramma di flusso particolareggiato del programma 5.5

Credo che i due diagrammi di flusso e gli elenchi delle variabili e dei vettori siano una spiegazione sufficiente del programma: studiatelo attentamente e provate a implementarlo, per esempio, con il calcolo delle sollecitazioni di taglio e momento flettente in alcune sezioni e, successivamente, con il dimensionamento a flessione per materiali omogenei: legno o profilati in acciaio.

PROGRAMMA 5.6: CALCOLO SOLLECITAZIONI TRAVI APPOGGIATE

Obiettivo: determinare le sollecitazioni di taglio e momento flettente in alcune sezioni di una trave appoggiata soggetta ad un numero qualsiasi di carichi concentrati.

Esempio

Una trave semplicemente appoggiata di luce $L = 6$ m è soggetta all'azione di tre carichi concentrati uguali di valore $P = 3000$ kg posti rispettivamente a 1, 3, 5 m di distanza dall'appoggio sinistro A.

Calcolare lo sforzo di taglio ed il momento flettente agli appoggi e nelle sezioni sottostanti ai carichi.

I dati da inserire nel programma sono

LUCE TRAVE (IN M) $L =$	6
NUMERO CARICHI CONCENTRATI	3
INPUT CARICO N 1	
VALORE DEL CARICO (IN KG)	3000
VALORE DELLA DISTANZA (IN M)	1
INPUT N 2	
VALORE DEL CARICO (IN KG)	3000
VALORE DELLA DISTANZA (IN M)	3
INPUT CARICO N 3	
VALORE DEL CARICO (IN KG)	3000
VALORE DELLA DISTANZA (IN M)	5

I risultati sono

REAZIONE IN A $R_A = 4500$ KG

REAZIONE IN B $R_B = 4500$ KG

SOLLECITAZIONI $M(KGM)$ E $T(KG)$

SEZ. 1	M = 0	T = 4500
SEZ. 2	M = 4500	T = 1500
SEZ. 3	M = 7500	T = -1500
SEZ. 4	M = 4500	T = -4500
SEZ. 5	M = 0	T = 0

Lista del programma

```

100 REM CALCOLO SOLLECITAZIONI TRAVE APPOGGIATA
110 PRINTCHR$(147)
120 PRINT"CALCOLO SOLLECITAZIONI TAGLIO E MOMENTO FLETTENTE IN UNA TRAVE ";
125 PRINT"APPOGGIATA"
130 PRINT:PRINT
140 INPUT"LUCE TRAVE (IN M) L=";L
150 PRINT
160 INPUT"NUMERO CARICHI CONCENTRATI";N
170 NP=N+2
180 DIMP(NP),A(NP),T(NP),M(NP)
190 FOR I=2 TO N+1
200 PRINT
210 PRINT"INPUT CARICO N" I-1
220 INPUT"VALORE DEL CARICO (IN KG)";P(I)
230 INPUT"VALORE DELLA DISTANZA (IN M)";A(I)
240 NEXTI
250 PRINT:PRINT
260 INPUT "I DATI SONO ESATTI (S/N)";A$
270 IF A$ <> "S" THEN CLR:GOTO110
280 REM CALCOLO REAZIONI
290 RA=0:RB=0
300 FORI=2TONP
310 RB=RB+P(I)*A(I)/L
320 RA=RA+P(I)*(L-A(I))/L
330 NEXTI
340 REM VERIFICA SOMMATORIA FORZE VERTICALI
350 Y=0
360 P(1)=-RA:A(1)=0
370 P(N+2)=-RB:A(N+2)=L
380 FORI=1TONP
390 Y=Y+P(I)
400 NEXTI
410 IFABS(Y)>0.05THEN650
420 REM CALCOLO SOLLECITAZIONI
430 FORI=1TONP
440 DL=A(I)-A(I-1)
450 T(I)=T(I-1)-P(I)
460 M(I)=M(I-1)+T(I-1)*DL
470 NEXTI
480 REM STAMPA RISULTATI
490 PRINTCHR$(147)
500 PRINT"REAZIONE IN A RA="RA"KG"
510 PRINT"REAZIONE IN B RB="RB"KG"
520 PRINT:PRINT
530 PRINT"SOLLECITAZIONI M (KGM) E T (KG)"
540 PRINT
550 FORI=1TONP
560 PRINT"SEZ." I "M="M(I)" T="T(I)
580 NEXTI
590 PRINT
600 INPUT"ALTRI CALCOLI (S/N)";B$
610 IFB$="S"THEN CLR:GOTO110

```



```
620 END
650 REM SOMMATORIA FORZE VERTICALI <00
660 PRINT:PRINT:PRINT
670 PRINT"VERIFICA ERRATA: RIPROCEDERE"
680 GOTO620
```

Descrizione. Il programma prevede in input la luce della trave e il numero dei carichi concentrati (N). Successivamente vengono richiesti per N volte i valori dei carichi e delle rispettive distanze. Eseguiti il calcolo e la verifica delle reazioni, inizia il calcolo delle sollecitazioni T e M nelle sezioni corrispondenti ai carichi concentrati e alle reazioni.

Segue quindi la stampa delle reazioni e delle sollecitazioni nelle varie sezioni.

Nota: Il taglio viene calcolato nella sezione infinitesimamente a destra della posizione delle forze concentrate.

Commento al programma. Se avete studiato fino a questo punto del libro e svolto alcuni degli esercizi, non troverete eccessive difficoltà a capire da soli lo svolgimento del programma.

Se non ci riuscite, rileggetevi il capitolo 4 e il programma 5.5.

Terzo livello

In questa terza parte del libro viene insegnato come costruire programmi complessi, di livello professionale, con l'uso di tecniche di programmazione avanzate e della stampante.

Nel capitolo 6 vengono introdotte, applicandole a un programma di estimo, due utili tecniche di programmazione: il menù e la subroutine.

Nel capitolo 7 viene insegnato l'uso della stampante.

Il capitolo 8, infine, riporta alcuni programmi che risolvono problemi professionali di costruzioni e di topografia. Tali programmi possono essere costruiti, compresi, ed eventualmente, modificati e migliorati, dai lettori che hanno studiato le due parti precedenti del libro.

Tuttavia, chi è interessato solo all'uso dei programmi riportati può, anche senza aver letto la seconda parte del volume, copiarli sul proprio computer e usarli, seguendo le spiegazioni e gli esempi che sono forniti per ciascun programma.

L'hardware necessario per l'uso dei programmi di questo livello è composto da: un Commodore 64, o altri computer della Commodore, un registratore o floppy driver, un televisore o monitor TV e, facoltativamente, una stampante.

Il floppy driver compatibile con il Commodore 64 è il VIC 1541, mentre le stampanti sono diverse, la più comune delle quali è la MPS 801.

L'uso della stampante è, come ho detto, facoltativo; infatti i programmi sono realizzati in modo da funzionare anche senza stampante.

Se tuttavia l'uso che si vuol fare del computer è di tipo professionale, non si può fare a meno di conservare i risultati delle elaborazioni su carta e l'acquisto di una stampante diviene, prima o poi, indispensabile.

Menù e subroutine

In questo capitolo esamineremo un programma di estimo, il CALCOLO DELLE ANNUALITÀ, per la cui costruzione sono state usate due tecniche di programmazione molto utili: il cosiddetto *menù* e la *subroutine*.

Vediamo dapprima che cos'è un menù.

MENÙ

In molti problemi si presenta la necessità di dover scegliere tra diverse opzioni.

Per eseguire tale scelta faremo apparire sul video delle scritte indicanti le varie possibilità offerte dal programma, precedute da un numero di identificazione, un po' come il "menù" di un ristorante.

A seconda del numero scelto, il computer ci fornirà, continuando con le immagini gastronomiche, la pietanza desiderata, il programma cioè proseguirà con la parte da noi scelta.

Una maniera comoda per far eseguire l'operazione descritta, in un programma in Basic, è quella di usare una combinazione delle istruzioni GET e ON ... GOTO

L'istruzione GET (*to get* = prendere) consente di trasferire dati dalla tastiera, prendendo un carattere alla volta. GET è seguito da un nome di variabile, in genere una variabile di stringa, ed il carattere battuto sulla tastiera viene assegnato alla variabile.

Ad esempio, la riga

```
10 GET A$
```


prende il carattere battuto sulla tastiera e lo assegna alla variabile A\$. L'istruzione GET assomiglia all'istruzione INPUT, con la differenza che il programma non si interrompe per l'esecuzione di GET e può essere inserito, come dato, un solo carattere. Se non viene battuto alcun tasto alla variabile viene assegnato un valore nullo.

Abbiamo visto che GET viene eseguita automaticamente, senza che il programma si interrompa. Per interromperlo e darci il tempo di battere il tasto desiderato usiamo iterazioni di questo tipo

```
100 GET A$: IF A$ = "" THEN 100
```

finché non viene premuto un tasto, cioè finché la stringa A\$ è nulla, il programma non prosegue.

Conosciamo già invece l'istruzione ON X GOTO Tale istruzione permette l'estensione multipla, permette cioè, in relazione al valore dell'espressione o variabile X, di saltare in diversi punti del programma indicati dai numeri di riga che seguono il GOTO.

Per esempio, la riga

```
50 ON (K + 1) GOTO 100,400,800
```

calcola il valore dell'espressione $K + 1$ e, se il risultato è 1, il programma prosegue dalla prima riga dell'elenco, cioè la riga 100; analogamente se $K + 1$ vale 2 o 3 il programma prosegue con le righe 400 o 800.

Se il valore dell'espressione $K + 1$ è zero o maggiore del numero di righe dell'elenco, nel nostro caso 3, viene eseguita la riga di programma successiva alla 50.

Prima di passare alla spiegazione di come si costruisce un menù, dobbiamo aggiungere al nostro bagaglio di conoscenze un'ultima informazione: la funzione VAL(X\$).

Tale funzione converte la stringa X\$, se possibile, in un numero. Ad esempio, con il seguente programma

```
10 A$ = "10"
```

```
20 X = VAL(A$)
```

alla variabile numerica X viene assegnato il valore 10.

Possiamo ora costruire un menù. Esaminiamo il seguente programma.

```
100 REM MENU'
110 PRINTCHR$(147)
120 PRINT"PUOI SCEGLIERE TRA":PRINT
130 PRINT"1 SCRIVERE A"
```



```
140 PRINT"2  SCRIVERE B"  
150 PRINT"3  SCRIVERE C"  
160 GETA$:ON VAL(A$) GOTO 180,190,200  
170 GOTO160  
180 PRINT:PRINT"A":GOTO210  
190 PRINT:PRINT"B":GOTO210  
200 PRINT:PRINT"C"  
210 END
```

Le righe da 120 a 150 stampano sul video le diverse opzioni offerte dal menù, precedute da un numero di identificazione.

Il nocciolo del programma è costituito dalle righe 160 e 170. Alla riga 160 viene eseguita l'istruzione **GET**, viene cioè assegnata alla variabile **A\$** la stringa corrispondente al tasto premuto. La stringa **A\$** viene quindi trasformata in valore numerico con la funzione **VAL(A\$)**.

Se tale valore risulta essere 1, il programma prosegue alla riga 180. Qui viene stampato **A**, dopo di che il programma termina. Similmente, per **VAL(A\$)** uguale a 2 o 3 il programma prosegue alle righe 190 o 200.

Se invece il valore numerico di **A\$** è diverso da 1, 2 o 3, il programma prosegue con la riga successiva, cioè la 170, dove viene rimandato alla riga 160.

Quindi le uniche possibilità per far continuare il programma oltre le righe 160-170 sono di premere i tasti 1, 2 o 3, scegliere cioè una delle possibilità offerte dal menù.

SUBROUTINE

In programmi complessi capita spesso che si debba eseguire più volte uno stesso gruppo di operazioni. Ciò comporterebbe di dover riscrivere ogni volta la stessa serie di istruzioni, con spreco di tempo per il programmatore e di memoria del computer.

È utile in questi casi definire tali parti del programma come *sottoprogrammi* o *subroutine*, raggruppandole alla fine del programma principale.

Durante l'esecuzione del programma principale, quando abbiamo bisogno di eseguire la subroutine, inseriamo l'istruzione **GOSUB**, seguita dal numero di riga di inizio del sottoprogramma.

Ciascuna subroutine termina con una istruzione **RETURN**, che segnala al computer la fine del sottoprogramma e fa ritornare il controllo al programma principale riprendendone l'esecuzione dalla istruzione successiva a quella contenente il comando **GOSUB**.

Uno schema di programma con subroutine è il seguente:


```

100 REM PROGRAMMA PRINCIPALE      Inizio programma principale
.....
.....
.....
200 GOSUB 1000
210 .....
.....
.....
.....
300 GOSUB 1000
310 .....
.....
.....
.....
500 END                          Fine programma principale
1000 REM SUBROUTINE              Inizio subroutine
.....
.....
.....
1200 RETURN                      Fine subroutine

```

Il programma inizia alla riga 100 e continua normalmente fino alla riga 200 dove trova l'istruzione GOSUB 1000. Prosegue quindi dalla riga 1000 eseguendo la subroutine. Alla riga 1200 l'istruzione RETURN indica che il sottoprogramma è terminato, il computer ritorna quindi al programma principale eseguendo la riga 210 e le successive. Alla riga 300 viene eseguita di nuovo la subroutine, dopo di che il programma prosegue con la riga 310 per terminare alla riga 500.

CALCOLO DELLE ANNUALITÀ

Il programma che segue serve come esempio per l'uso del menù e delle subroutine e, nello stesso tempo, permette di risolvere una serie di problemi di matematica finanziaria.

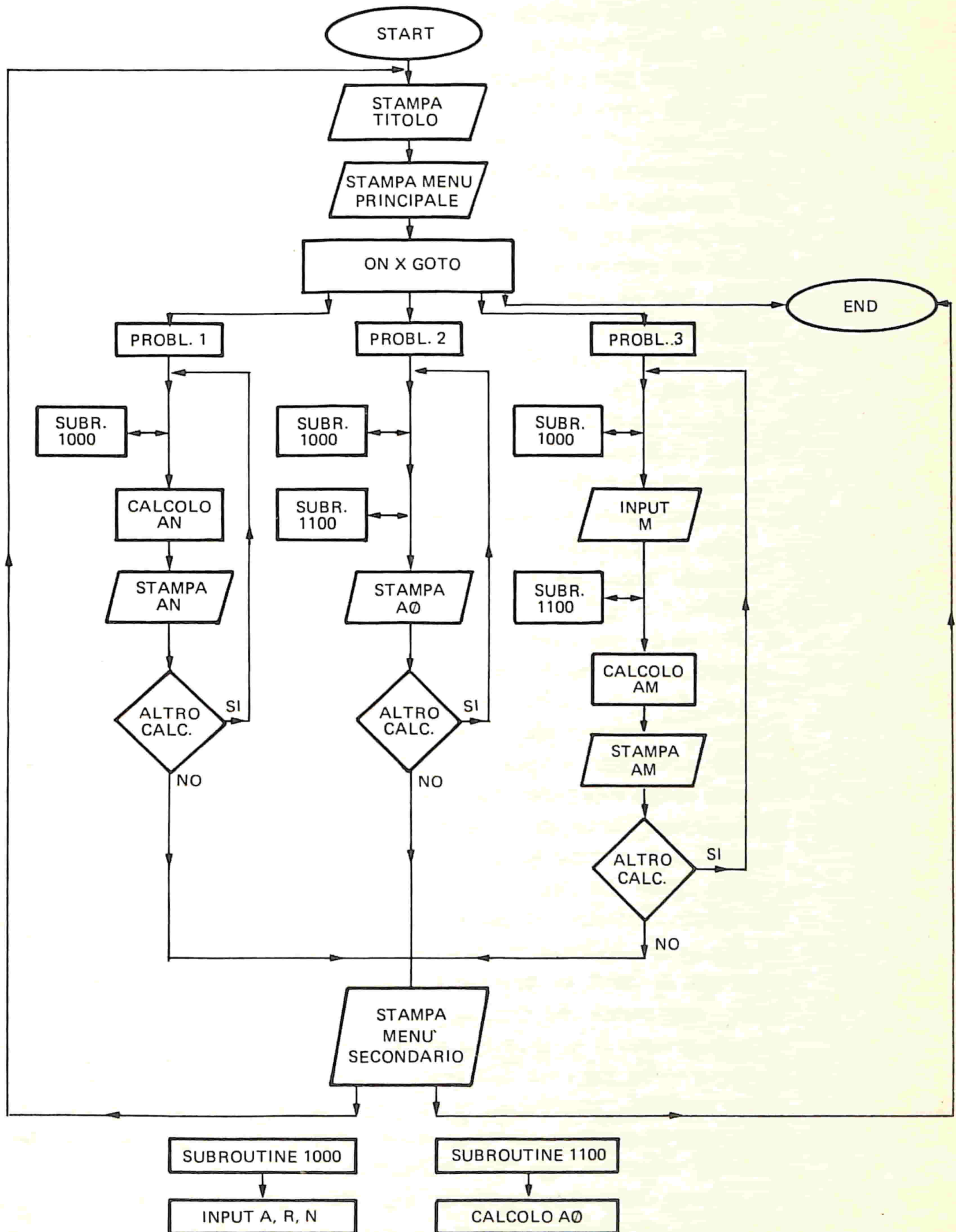


Fig. 6.1 Diagramma di flusso del programma calcolo annualità

Premetto alcuni richiami di estimo che ci permetteranno di comprendere meglio il problema che intendiamo risolvere.

Si intendono per *annualità* dei valori che si ripetono a intervalli regolari di un anno e che possono essere, per esempio, redditi, costi, crediti, quote di ammortamento, ecc.

Ci proponiamo di calcolare le *annualità costanti posticipate limitate*, cioè annualità che si ripetono con il medesimo ammontare (costanti), che si presentano alla fine di ogni anno (posticipate) e che si verificano per un numero limitato di anni (limitate).

Con il tipo di annualità in questione possono presentarsi tre diversi tipi di problemi:

1. *accumulazione finale*: quando le annualità sono riferite all'anno N , cioè alla fine del periodo; in questo caso i dati necessari per il calcolo sono: l'ammontare delle annualità (A), il saggio di interesse (R) ed il numero di anni per cui si verificano le annualità (N); la formula risolutiva è $AN = A \times (Q^N - 1)/R$ dove $Q = 1 + R$;
2. *accumulazione iniziale*: quando le annualità sono riferite all'anno zero del periodo esaminato; in questo caso i dati necessari per il calcolo sono gli stessi del caso 1. e la formula risolutiva è $A0 = A \times (Q^N - 1)/(R \times Q^N)$, con $Q = 1 + R$;
3. *accumulazione intermedia*: quando le annualità sono riferite ad un anno intermedio del periodo; in questo caso per i calcoli è necessario, oltre ai dati dei casi precedenti, l'anno intermedio (M) a cui vanno riferite le annualità; le formule risolutive sono $A0 = A \times (Q^N - 1)/(R \times Q^N)$ con $Q = 1 + R$, e $AM = A0 \times Q^M$.

Questa lunga spiegazione era necessaria per far notare come, nei tre casi, si tratti di problemi simili, in cui parte delle operazioni sono comuni. Ad esempio gli input di A , R e N , il calcolo di $Q = 1 + R$, il calcolo di $A0$ che ricorre nei casi 2 e 3.

Potremmo costruire tre diversi programmi che risolvano ciascuno uno dei tre problemi elencati. Ma, data la comunanza dell'argomento e la presenza di parti di problema uguali, sarà più pratico costruire un unico programma che risolva tutti e tre i casi.

Organizzeremo il programma in modo che all'inizio possa essere compiuta la scelta tra i tre differenti problemi. Le parti che sono comuni ai tre problemi: parte degli input e dei calcoli, verranno organizzate in sottoprogrammi posti alla fine del programma principale. Il diagramma di flusso relativo è riportato in fig. 6.1.

Lo schema del nostro programma sarà il seguente:

- stampa titolo del programma
- menù di scelta tra i problemi 1, 2, 3
- problema 1: accumulazione finale
- *GO* subroutine di input

calcolo *AN*
 stampa risultati
 richiesta di altro calcolo
GO TO menù secondario
 — problema 2: accumulazione iniziale
 GO subroutine di input
 GO subroutine di calcolo *A0*
 stampa risultati
 richiesta di altro calcolo
 GO TO menù secondario
 — problema 3: accumulazione intermedia
 GO subroutine di input
 input anno intermedio
 GO subroutine calcolo *A0*
 calcolo *AM*
 stampa risultati
 richiesta di altro calcolo
 GO TO menù secondario
 — *END* programma principale
 — subroutine di input
 — subroutine calcolo *A0*
 — menù secondario

Il menù secondario permette di ritornare al menù principale, per scegliere di lavorare su un altro dei tre problemi possibili, senza dover far ripartire il programma dall'inizio. Prevede infatti la scelta tra ritornare al menù principale e finire. Il programma completo è il seguente:

```

100 REM CALCOLO ANNUALITA'
110 PRINTCHR$(147)
120 PRINTTAB(10)"CALCOLO ANNUALITA'"
130 PRINTTAB(9)"POSTICIPATE LIMITATE"
140 PRINT:PRINT
150 REM MENU' PRINCIPALE
160 PRINT"1) ACCUMULAZIONE FINALE":PRINT
170 PRINT"2) ACCUMULAZIONE INIZIALE":PRINT
180 PRINT"3) ACCUMULAZIONE INTERMEDIA":PRINT
190 PRINT"4) FINE":PRINT
200 PRINT:PRINT
210 PRINT"SBATTI IL NUMERO RELATIVO AL TUO PROBLEMA"
220 A$="ANNUALITA' COSTANTI POSTICIPATE LIMITATE"
230 GETC$:ON VAL(C$) GOTO 250,350,450,999
240 GOTO230
250 REM PROBLEMA 1: ACCUMULAZIONE FINALE
260 PRINTCHR$(147)A$
270 PRINT:PRINTTAB(10)"ACCUMULAZIONE FINALE"
280 GOSUB1000
290 AN=A*(0.1N-1)/R
300 PRINT:PRINT
310 PRINT"ACCUMULAZIONE FINALE AN="INT(AN+.5)
320 PRINT
  
```



```

330 INPUT "ALTRO CALCOLO (S/N)"; D$: IF D$="S" THEN GOTO 250
340 GOTO 1200
350 REM PROBLEMA 2: ACCUMULAZIONE INIZIALE
360 PRINT CHR$(147); A$
370 PRINT: PRINT TAB(9) "ACCUMULAZIONE INIZIALE"
380 GOSUB 1000
390 GOSUB 1100
400 PRINT: PRINT
410 PRINT "ACCUMULAZIONE INIZIALE A0=" INT(A0+.5)
420 PRINT: INPUT "ALTRO CALCOLO (S/N)"; D$: IF D$="S" THEN GOTO 350
430 GOTO 1200
450 REM PROBLEMA 3: ACCUMULAZIONE INTERMEDIA
460 PRINT CHR$(147); A$
470 PRINT: PRINT TAB(8) "ACCUMULAZIONE INTERMEDIA"
480 GOSUB 1000
490 PRINT: INPUT "NUMERO ANNO INTERMEDIO"; M
500 GOSUB 1100
510 AM=A0*Q^M
520 PRINT: PRINT
530 PRINT "ACCUMULAZIONE INTERMEDIA ALL'ANNO" M
540 PRINT "AM=" INT(AM+.5)
550 PRINT: INPUT "ALTRO CALCOLO (S/N)"; D$: IF D$="S" THEN GOTO 450
560 GOTO 1200
999 END
1000 REM SUBROUTINE DI INPUT
1010 PRINT: PRINT
1020 INPUT "AMMONTARE DELL'ANNUALITA'"; A
1030 PRINT
1040 INPUT "SAGGIO DI INTERESSE (IN DECIMALI)"; R
1050 PRINT
1060 INPUT "NUMERO DI ANNI"; N
1070 Q=1+R
1080 RETURN
1100 REM SUBROUTINE CALCOLO A0
1110 A0=A*(Q^N-1)/(R*Q^N)
1120 RETURN
1200 REM MENU' SECONDARIO
1210 PRINT CHR$(147)
1220 PRINT: PRINT
1230 PRINT "1) MENU' PRINCIPALE"
1240 PRINT
1250 PRINT "2) FINE"
1260 PRINT: PRINT
1270 PRINT "BATTI IL NUMERO RELATIVO AL TUO PROBLEMA"
1280 GET B$: ON VAL(B$) GOTO 110,999
1290 GOTO 1280

```

Esempi di calcolo. Come prova del corretto funzionamento del programma eseguite i successivi esempi:

Problema 1: accumulazione finale

Si abbia da oggi diritto a un reddito di £ 200000, alla fine di ogni anno per 5 anni consecutivi; il saggio di interesse è $R = 0.10$. Determinare l'ammontare complessivo della somma esigibile alla fine del quinto anno.

I dati di ingresso sono quindi

$$A = 200000$$

$$R = 0.10$$

$$N = 5$$

l'ammontare della somma esigibile al termine del quinto anno è

$$AN = 1221020 \text{ L.}$$

Problema 2: accumulazione iniziale

Si sia contratto un debito per l'acquisto di una autovettura per cui si è debitori verso la ditta fornitrice, per 4 anni consecutivi, della somma annua di £ 2000000 pagabile alla fine di ogni anno a partire da oggi. Quanto bisogna pagare subito per liberarsi dell'intero debito, se il tasso di interesse è del 15%?

I dati sono:

$$A = 2000000$$

$$R = 0.15$$

$$N = 4$$

Il debito totale da pagare subito è

$$A_0 = 5709957 \text{ L}$$

Problema 3: accumulazione intermedia

Per eseguire un determinato lavoro, la cui durata è di 8 anni, si devono spendere alla fine di ogni anno £ 3000000. Supponendo che il tasso di interesse sia $R = 0.10$, si vuole conoscere la somma totale di tali spese riferita alla fine del quinto anno.

I dati sono

$$A = 3000000$$

$$R = 0.10$$

$$N = 8$$

$$M = 5$$

la somma totale delle spese riferita alla fine del quinto anno è

$$AM = 25775856 \text{ L.}$$

Commento al programma. Alla riga 120 troviamo subito qualcosa di nuovo: la funzione TAB. Tale funzione, nella forma PRINT TAB(X), fa stampare l'espressione che segue a partire dalla colonna X. Nel nostro caso la stringa che segue TAB(10) verrà quindi stampata a partire dalla colonna 10.

Le funzioni **TAB** sono state inserite nelle righe 120 e 130 per fare in modo che il titolo del programma risulti stampato sullo schermo in posizione centrale.

Sempre alla riga 120, all'inizio della stringa **CALCOLO ANNUALITÀ**, appare un simbolo che non è visibile sulla tastiera: una **R** in *reverse*. Tale simbolo indica l'operazione *reverse on* e si ottiene premendo contemporaneamente i tasti **CTRL** e **9**, dopo aver aperto le virgolette. Le funzioni di comando del cursore, di **CLR/HOME**, di **INST/DEL**, di *reverse on* e *reverse off* possono infatti essere programmate come dei normali tasti di carattere. Se cioè noi scriviamo, dopo un'istruzione **PRINT**, una stringa contenente tali funzioni, queste vengono "stampate" dal computer come se fossero normali tasti di carattere.

Nel nostro caso quindi, quando il computer inizia a stampare la stringa che compare alla riga 120, stampa il primo carattere che incontra: esegue cioè un *reverse on*, poi continua stampando gli altri caratteri che appariranno, appunto, in *reverse*.

Nel seguito del programma altre stringhe verranno stampate in *reverse*, per farle risaltare meglio sul video.

Come esercizio di uso dei comandi di cursore programmati, provate a sostituire, nella riga 110 al **CHR\$(147)** la seguente serie di tasti: virgolette, **CLR/HOME** shiftato, virgolette. Si otterrà lo stesso risultato, quello cioè di pulire lo schermo e portare il cursore in alto a sinistra. Ritorniamo al nostro programma, osservando che le righe da 160 a 210 fanno stampare il menù principale e la frase di richiesta di scelta. Le righe 230 e 240 completano il menù. Come abbiamo già visto, alla riga 230 con l'istruzione **GET** viene preso il primo carattere battuto sulla tastiera e tale carattere viene assegnato alla variabile **C\$**. Successivamente viene valutato il valore numerico di **C\$**; se tale valore è 1 il programma prosegue con la riga 250, cioè con il problema 1: accumulazione finale.

Analogamente se **VAL(C\$)** risulta essere rispettivamente 2, 3, o 4, il programma prosegue con le righe 350, 450 o 999, cioè con i problemi 2, 3 o eseguendo l'istruzione **END** e terminando.

Se il valore numerico di **C\$** risulta maggiore di 4 o zero (il che avviene se si batte lo 0 o un qualsiasi tasto non numerico) il programma prosegue alla riga successiva, cioè alla 240.

Alla riga 240 troviamo l'istruzione **GO TO 230** che ci riporta alla riga 230 con, di nuovo, la richiesta di **GET C\$**. È quindi possibile far proseguire il programma oltre le righe 230-240 solo se si premono i tasti 1, 2, 3 o 4, cioè se si sceglie una delle possibilità offerte dal menù.

Se si sceglie, ad esempio, il problema 1, il programma prosegue dalla riga 250, viene quindi, alla riga 260, pulito lo schermo e stampata la stringa **A\$**.

Alla precedente riga 220 A\$ è stata posta uguale ad ANNUALITÀ COSTANTI POSTICIPATE LIMITATE in *reverse*. Si è preferito usare tale artificio per non dover sempre riscrivere la stringa in questione che appare diverse volte nel seguito del programma.

Alla riga 280 il programma viene inviato alla subroutine 1000 (durante la costruzione del programma si è scelta come linea di inizio della subroutine il numero 1000, pensando che tale numero fosse abbastanza alto da non essere raggiunto dalla serie di istruzioni del programma principale).

Dalla riga 1000 alla 1060 vengono richiesti gli input dei dati ed alla 1070 viene calcolato il valore di Q. Alla riga 1080 l'istruzione RETURN indica che la subroutine è terminata.

Il controllo ritorna quindi al programma principale, con l'esecuzione della istruzione successiva al comando di eseguire la subroutine, cioè della riga 290.

Qui viene calcolato il valore di AN e tale valore viene, alla 310, stampato con approssimazione all'ultima cifra intera. Trattandosi di calcoli eseguiti, presumibilmente, in lire italiane, si è infatti pensato di non far stampare la parte decimale. Volendo si può modificare il programma in modo che approssimi alle 1000 lire, sostituendo a $\text{INT}(\text{AN}+.5)$ l'espressione $\text{INT}(\text{AN}/1000+.5) * 1000$.

Segue alla riga 330 la richiesta di ALTRO CALCOLO; rispondendo S a tale richiesta è possibile eseguire altri calcoli relativi al Problema 1.

Se invece si vogliono eseguire calcoli relativi a un altro dei problemi previsti nel programma oppure si vuole finire, basta rispondere alla richiesta di ALTRO CALCOLO con un qualsiasi tasto alfanumerico diverso da S. In tal modo il programma prosegue alla riga 340.

L'istruzione GO TO 1200 fa "saltare" il programma a tale riga, dove viene stampato il menù secondario.

Costruito alla stessa maniera del menù principale, il menù secondario prevede la scelta tra due sole possibilità: tornare al menù principale per scegliere un altro dei problemi previsti dal programma, o finire.

La restante parte del programma serve a risolvere i problemi 2 e 3.

Le righe da 350 a 430 risolvono il Problema 2 (accumulazione iniziale), mentre le righe da 450 a 560, risolvono il problema 3 (accumulazione intermedia).

Tali problemi sono trattati in maniera analoga al Problema 1, con la presenza di una ulteriore subroutine per il calcolo di A0 dalla riga 1100 alla 1120.

La stampante

Chi intende usare il computer come uno strumento professionale non può fare a meno di aggiungere all'hardware minimo, costituito dal computer, una memoria di massa (registratore o floppy driver) e un televisore o TV monitor, una stampante.

Operando con tale apparecchiatura è possibile infatti conservare i risultati delle elaborazioni e dei calcoli effettuati con il computer, su carta.

Una stampante inoltre amplifica le capacità di uso del vostro computer. Esistono infatti in commercio diversi programmi di *word processing* (elaborazione di testi), che trasformano il computer in una sofisticata macchina da scrivere.

Con un programma *word processor* caricato nel computer, è infatti possibile scrivere, o come si dice in termini tipografici "comporre", il testo, di una lettera, di una relazione o altro, sul video. Si può quindi correggerlo, modificarlo e, solo quando si è soddisfatti del risultato, stamparlo nel numero di copie desiderato. Il testo può poi naturalmente essere registrato su una memoria di massa, richiamato quando è necessario, eventualmente modificato, e stampato di nuovo senza doverlo ribattere.

È così possibile costruirsi un archivio di testi di uso frequente, come relazioni per il Genio Civile, computi metrici, parcelle e via dicendo. Le stampanti elettroniche comunemente usate sono del tipo ad aghi. I caratteri vengono costruiti con un insieme di piccoli punti, stampati da aghi, disposti in maniera da formare i vari caratteri alfanumerici.

Le stampanti compatibili con il Commodore 64 sono diverse, noi faremo riferimento a una delle più comuni: la MPS 801 della Commodore (fig. 7.1).

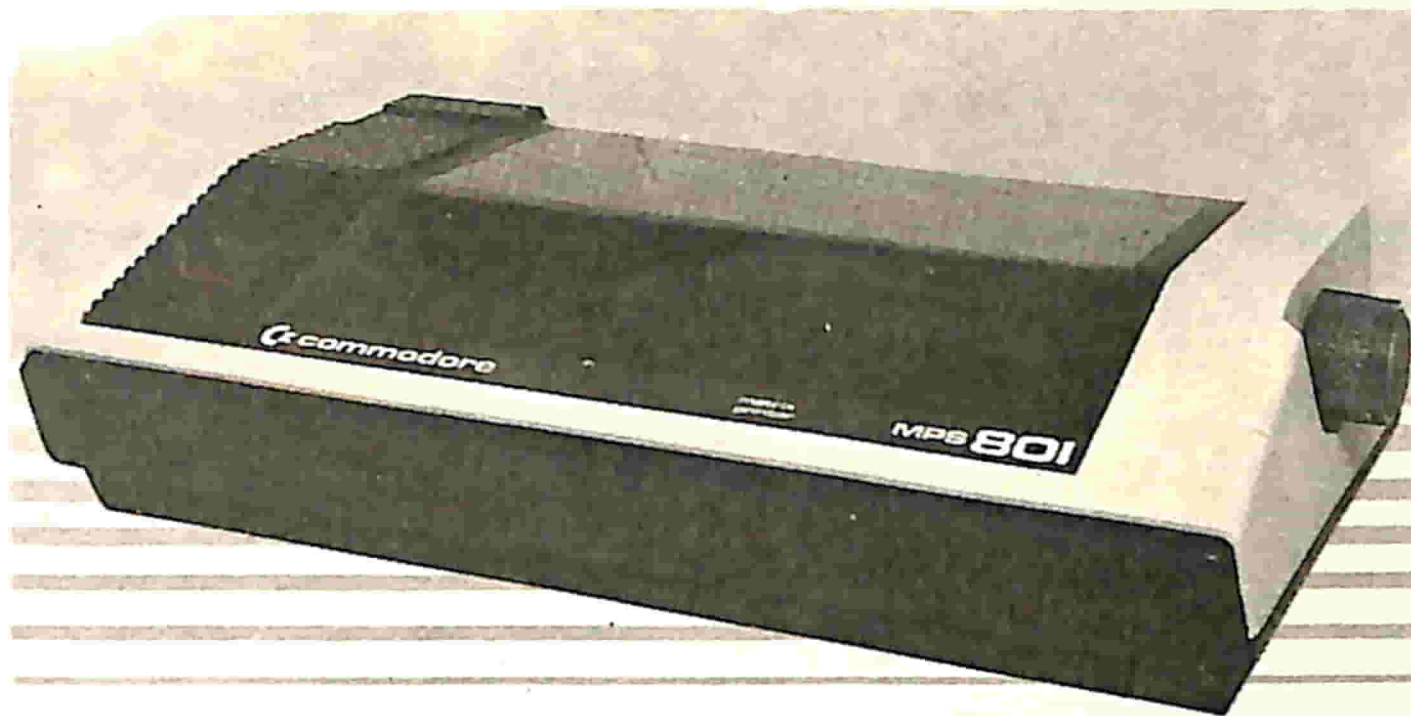


Fig. 7.1 La stampante MPS 801

L'installazione di tale apparecchiatura è molto semplice. Basta inserire la spina di alimentazione in una presa di corrente, collegare l'apposito cavo al computer o al floppy driver, inserire la carta e la stampante è pronta a funzionare. Maggiori informazioni sul funzionamento della stampante sono comunque riportate nel manuale di funzionamento della macchina.

La MPS 801 scrive i caratteri stampandoli in una matrice di punti 5×7 . La carta usata è quella a modulo continuo del formato 24×11 pollici, dove è possibile stampare 80 caratteri per riga.

DIRECT MODE

Cominciamo a usare la nostra stampante operando in *direct mode*, cioè direttamente dalla tastiera.

Vediamo come è possibile ottenere, per esempio, il listato di un programma stampato su carta.

Accendete le varie apparecchiature, cominciando dalle unità periferiche: video, floppy o registratore, stampante e, da ultimo, il computer. Caricate un programma da disco o da cassetta, oppure scrivete un programma direttamente alla tastiera.

A questo punto nella memoria del computer è presente il programma da voi desiderato. Per ottenerne il listato sul video basta digitare il comando **LIST**. Se il programma occupa più di 25 righe dello schermo, non sarà comunque possibile vederlo contemporaneamente tutto sul video.

Per ottenere la stampa del listato battete sulla tastiera

OPEN 1,4

CMD 1

digitate ora il comando

LIST

e la stampante comincerà a scrivere il listato del vostro programma. Quando appare la scritta

READY

scrivete

PRINT#1: CLOSE 1

e l'operazione è conclusa.

Analizziamo le varie operazioni che abbiamo compiuto.

L'istruzione OPEN (= apri), nel formato OPENfn,dn, serve per aprire un *file*. Per file si intende un insieme di dati, cioè una raccolta ordinata e autonoma di informazioni. L'istruzione OPEN è seguita dal numero di identificazione che diamo al file (fn = *file number*) che può essere un numero compreso tra 1 e 128. Dopo la virgola, va indicato il numero della apparecchiatura (dn = *device number*) a cui è indirizzato il file.

Generalmente il numero di identificazione dn per la stampante è 4; alcune stampanti sono tuttavia predisposte con un dn uguale a 5. Se l'istruzione nella forma sopra scritta non dovesse funzionare, consultate il manuale della vostra stampante per individuare l'esatto numero di identificazione.

Con l'istruzione OPEN 1,4, quindi, abbiamo aperto il file 1 verso l'apparecchiatura 4, cioè la stampante.

Il successivo comando CMD invia l'output, che normalmente andrebbe allo schermo, ad un altro dispositivo, nel nostro caso la stampante. Il comando CMD deve essere seguito dal numero del file su cui viene inviato l'output; deve essere lo stesso numero presente nell'istruzione OPEN, nel nostro caso 1.

A questo punto abbiamo quindi aperto un file verso la stampante ed abbiamo trasferito l'output dei dati su tale file.

Quando ora inviamo il comando LIST, il listato non apparirà quindi più sul video, ma sulla stampante.

Per riportare il computer nelle normali condizioni di funzionamento, al termine della operazione di stampa, dobbiamo scrivere l'istruzione PRINT#1, che disinserisce il comando CMD1, e dobbiamo chiudere il file che avevamo aperto. Per chiudere un file si usa l'istruzione CLOSE (= chiudi), seguita dal numero del file da chiudere, nel nostro caso 1.

Esercizi di scrittura

Oltre a eseguire operazioni di stampa di listati, possiamo anche scrivere direttamente sulla stampante, seguendo una serie di operazioni simili alle precedenti.

Scrivete

```
OPEN1,4:CMD1
```

ora la stampante è pronta a ricevere quello che vogliamo stampare.

Come per scrivere sullo schermo useremo l'istruzione PRINT, solo che ciò che normalmente compare sul video verrà ora stampato.

Per esempio, scrivete

```
PRINT"PROVA DI STAMPA N. 1"
```

verrà stampata la seguente riga

```
PROVA DI STAMPA N. 1
```

Scrivete ora

```
FOR I = 1 TO 5:PRINT I, I ↑ 2, SQR(I):NEXT I
```

verrà stampata la seguente lista

1	1	1
2	4	1.41421356
3	9	1.73205081
4	16	2
5	25	2.23606798

Per tornare al normale funzionamento scrivete

```
PRINT#1:CLOSE1
```

e il canale di comunicazione con la stampante verrà chiuso.

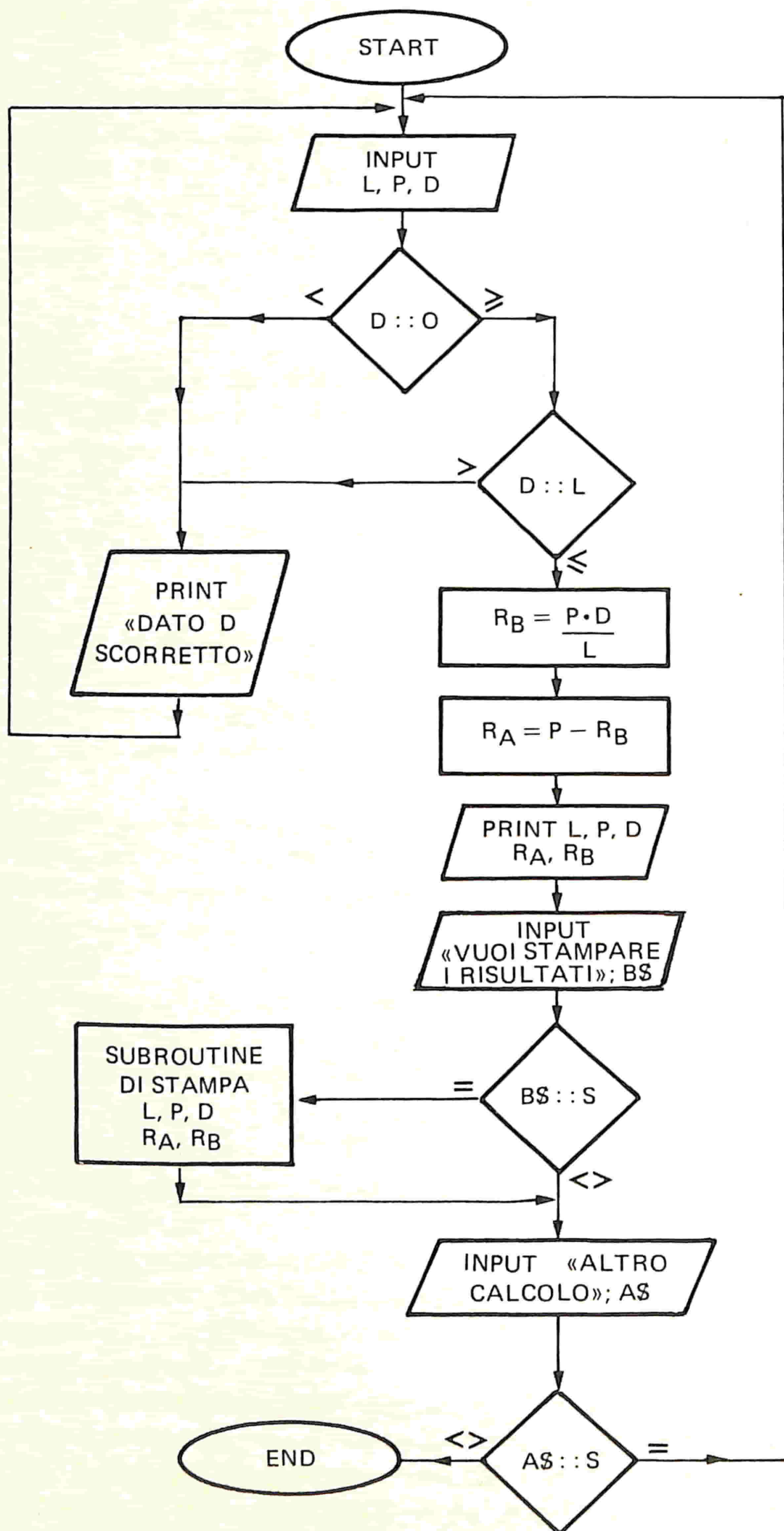


Fig. 7.2 Diagramma di flusso del programma di calcolo reazioni trave appoggiata soggetta a un carico concentrato con stampa dei risultati

PROGRAM MODE

L'uso della stampante in *program mode*, cioè inserito in un programma, è un'operazione abbastanza semplice. Vengono usate due istruzioni che già conosciamo, OPEN e CLOSE, e l'istruzione PRINT# che è praticamente l'equivalente dell'istruzione PRINT per scrivere sul video. Per comodità conviene raggruppare tutte le operazioni di stampa in una o più subroutine poste in fondo al programma.

Vediamo, ad esempio, come è possibile far stampare i risultati dei calcoli svolti con il programma CALCOLO REAZIONI TRAVE APPOGGIATA PER CARICHI CONCENTRATI che abbiamo costruito nel capitolo 4.

Confrontate il diagramma di flusso di tale programma, riportato in fig. 4.1, con quello della fig. 7.2.

Abbiamo inserito, prima della richiesta di ALTRO CALCOLO, la richiesta VUOI STAMPARE I RISULTATI. Se a tale richiesta viene risposto S (per sì), il programma prosegue con la subroutine di stampa, dopo di che riprende il flusso principale con la richiesta di ALTRO CALCOLO.

Nel programma che abbiamo già scritto al capitolo 4, basterà quindi aggiungere la seguente riga

```
255 INPUT "VUOI STAMPARE I RISULTATI (S/N)";B$: IF B$ = "S"  
THEN GOSUB 500
```

e scrivere la subroutine di stampa a partire dalla riga 500.

La subroutine di stampa è la seguente

```
500 REM SUBROUTINE DI STAMPA  
510 OPEN1,4  
520 PRINT#1,"CALCOLO REAZIONI TRAVE APPOGGIATA PER CARICHI CONCENTRATI"  
530 PRINT#1:PRINT#1  
540 PRINT#1,"L="L"M","P="P"KG","D="D"M"  
550 PRINT#1  
560 PRINT#1,"RA="RA"KG","RB="RB"KG"  
570 PRINT#1:PRINT#1  
580 CLOSE1  
590 RETURN
```

Commento al programma. Alla riga 510 viene aperto il file 1, sulla stampante (in numero di identificazione della stampante è 4).

Alla riga 520 troviamo l'istruzione PRINT#. Il formato di tale istruzione è

PRINT#fn,[lista delle espressioni]

Il numero *fn* (*fn* = *file number*) deve essere lo stesso del file aperto con l'istruzione **OPEN**, quindi nel nostro caso 1. Dopo la virgola segue la lista delle espressioni che si vogliono far stampare. Da questo punto di vista l'istruzione **PRINT#** funziona come l'istruzione **PRINT**, con le stesse regole di punteggiatura.

La riga 530 fa invece stampare due righe vuote.

Vengono successivamente stampati i valori dei dati di ingresso *L*, *P*, *D*, con le relative unità di misura. La virgola che separa le tre espressioni comporta, come per l'istruzione **PRINT**, che queste verranno stampate a una certa distanza una dall'altra; se le espressioni fossero separate da un punto e virgola (;) verrebbero stampate con due soli spazi tra l'una e l'altra.

Segue la stampa di una riga vuota e dei risultati dei calcoli: *RA* e *RB*, con le relative unità di misura. La riga 570 stampa poi altre due righe vuote.

Alla riga 580 troviamo l'istruzione **CLOSE**, seguita dal numero del file da chiudere, nel nostro caso 1. Quando è terminata l'operazione di stampa infatti bisogna chiudere il file su cui si è stampato.

Alla riga 590 l'istruzione **RETURN** indica che la subroutine è terminata e si ritorna al programma principale.

IL PROGRAMMA COMPLETO

Il listato completo del programma, in cui, oltre alla riga 255 ed alla subroutine 500, è stata modificata la **REM** alla riga 100, è il seguente.

```

100 REM CALCOLO REAZIONI TRAVE APPOGGIATA PER CARICHI CONCENTRATI, CON STAMPA
110 PRINTCHR$(147)
120 PRINT"CALCOLO REAZIONI DI UNA TRAVE APPOGGIATA PER CARICO CONCENTRATO"
130 PRINT:PRINT
140 INPUT"LUCE DELLA TRAVE (IN M), L=";L
150 INPUT"VALORE DEL CARICO (IN KG), P=";P
160 INPUT"DISTANZA DI P DALL'APPOGGIO SINISTRO (IN M), D=";D
170 IF D<0 THEN GOTO 300
180 IF D>L THEN GOTO 300
190 RB=P*D/L
200 RA=P-RB
210 PRINT:PRINT
220 PRINT"L="L"M", "P="P"KG", "D="D"M"
230 PRINT
240 PRINT"RA="RA"KG", "RB="RB"KG"
250 PRINT:PRINT
255 INPUT"VUOI STAMPARE I RISULTATI (S/N)";B$:IF B$="S" THEN GOSUB 500
260 INPUT"ALTRO CALCOLO (S/N)";A$
270 IF A$="S" THEN GOTO 110
280 END
300 PRINT:PRINT:PRINT
310 PRINT"VALORE DELLA DISTANZA D SCORRETTO"
320 GOTO 130
500 REM SUBROUTINE DI STAMPA

```



```
510 OPEN1,4
520 PRINT#1,"CALCOLO REAZIONI TRAVE APPOGGIATA PER CARICHI CONCENTRATI"
530 PRINT#1:PRINT#1
540 PRINT#1,"L="L"M","P="P"KG","D="D"M"
550 PRINT#1
560 PRINT#1,"RA="RA"KG","RB="RB"KG"
570 PRINT#1:PRINT#1
580 CLOSE1
590 RETURN
```

Come prova del corretto funzionamento del programma fatelo girare con i seguenti dati:

$L = 5 \text{ m}$, $P = 3500 \text{ kg}$, $D = 2 \text{ m}$

e

$L = 7.3 \text{ m}$, $P = 10200 \text{ kg}$, $D = 5.6 \text{ m}$

Il risultato in stampa sarà il seguente

CALCOLO REAZIONI TRAVE APPOGGIATA PER CARICHI CONCENTRATI

$L = 5 \text{ M}$ $P = 3500 \text{ KG}$ $D = 2 \text{ M}$

$RA = 2100 \text{ KG}$ $RB = 1400 \text{ KG}$

CALCOLO REAZIONI TRAVE APPOGGIATA PER CARICHI CONCENTRATI

$L = 7.3 \text{ M}$ $P = 10200 \text{ KG}$ $D = 5.6 \text{ M}$

$RA = 2375.34247 \text{ KG}$ $RB = 7824.65753 \text{ KG}$

Esercizi

Come esercizio vi proporrei di aggiungere, ai programmi che sono riportati nel libro, delle subroutine che facciamo stampare i risultati, cominciando dai programmi che avrete costruito risolvendo agli esercizi del capitolo 4.

In fondo al volume troverete delle soluzioni possibili.

Programmi completi

In questo capitolo sono riportati alcuni programmi che risolvono problemi di livello professionale di costruzioni e topografia: progetto e verifica di sezioni rettangolari in cemento armato, progetto e verifica di muri di sostegno a gravità, poligonale aperta.

Per ciascun programma vengono forniti il listato, la lista delle principali variabili, la descrizione del funzionamento ed esempi di prova per controllare il corretto funzionamento del programma.

Per costruire i programmi di questo capitolo sono state usate, oltre le istruzioni e funzioni Basic spiegate nel corso del libro, i comandi e le funzioni spiegati in questo paragrafo.

Comando DEF FN

Questo comando permette di definire (*to DEFine* = definire) con un breve nome un calcolo complesso come una funzione (*FuNction* = funzione). Quando una funzione viene usata spesso nel corso di un programma, l'uso di questo comando fa risparmiare tempo al programmatore, che non deve riscriverla ogni volta, e memoria nel computer.

Il formato del comando è

DEF FN <nome> (variabile) = <funzione>

Per esempio

10 DEF FNA(X) = 4 + 3 * X

definisce la funzione FNA della variabile X, la cui espressione è $4 + 3 * X$.

Il nome che segue FN può essere un qualsiasi nome di variabile Basic. Una volta definita la funzione, con una riga come la precedente, quando nel corso del programma vorremo far calcolare la funzione per un particolare valore della variabile X, richiameremo la funzione scrivendone il nome e inserendo al posto della variabile il valore desiderato.

Per esempio, la riga

```
100 PRINT FNA(5)
```

fa stampare il valore

$$4 + 3 * 5 = 19$$

la riga

```
200 K = SQR(FNA(4))
```

assegna alla variabile K il valore

$$\text{SQR}(4 + 3 * 4) = \text{SQR}(16) = 4$$

Funzione STR\$(X)

Questa funzione trasforma la variabile numerica X in una stringa. Cioè produce praticamente una stringa uguale alla versione stampata (PRINT) di X. Si può considerare, per le variabili numeriche, come l'inverso della funzione VAL(X\$).

La funzione STR\$(X), usata in connessione con la funzione LEN, viene utilizzata per la costruzione di tabelle, sia su video che su stampante.

Per esempio, il seguente programma

```
10 A = 4.321
```

```
20 PRINT STR$(A)
```

fa stampare la stringa

4.321

Funzione LEN(X\$)

Questa funzione dà il numero di caratteri della stringa X\$, compresi gli spazi e altri simboli (*LENgth* = lunghezza).

Per esempio, il programma

```
10 KT = 54.72
```

```
20 PRINT TAB(10-LEN(STR$(KT)))KT
```

fa stampare il valore di KT incolonnando l'ultima cifra a destra alla decima colonna.

Istruzioni del tipo della riga 20 sono usate per la costruzione di tabelle su video e su stampante.

PROGETTO DI SEZIONI RETTANGOLARI IN C.A.

Problema: progettare una o più sezioni rettangolari in cemento armato, sollecitate a flessione, ad armatura semplice o doppia.

```

100 REM PROGETTO DI SEZIONI RETTANGOLARI IN C.A. SOLLECITATE A MOMENTO FLETTENTE
110 REM ARMATURA SEMPLICE O DOPPIA
120 PRINTCHR$(147):N=15:SC=97.5:SA=2200
130 PRINTTAB(6)"PROGETTO SEZ. RETTANGOLARI":PRINT
140 PRINTTAB(15)"IN C.A."
150 SE=SE+1:PRINT
160 PRINT"SEZIONE N"SE
170 PRINT:PRINT
180 PRINT"INPUT DATI":PRINT
190 PRINT"RAPPORTO DI OMOG.      N= "N";:INPUT"|||||";N
200 PRINT"TENSIONE CONGLOMERATO  SC= "SC";:INPUT"|||||";SC
210 PRINT"TENSIONE ACCIAIO      SA= "SA";:INPUT"|||||";SA
220 PRINT:INPUT"LARGHEZZA B (IN CM)      B=";B
230 INPUT"RAPPORTO A2/A1      MU=";MU
240 IFMU=0THENC=0:GOTO260
250 INPUT"RAPPORTO H1/H      LA=";LA
260 PRINT:INPUT"MOMENTO FLETTENTE(IN KGCM)";M:IFM<0THENGOTO500
270 PRINT:INPUT"SI DATI SONO ESATTI (S/N) S||||";A$:IFA$<>"S"THENPRINT:GOTO180
280 PRINTCHR$(147):PRINT:PRINTTAB(14)"SEZIONE N"SE:PRINT:PRINT"DATI DI INPUT"
290 PRINT:PRINT"N="N,"SC="SC,"SA="SA,"B="B,"MU="MU,"LA="LA,"M="M
300 PRINT:PRINT:PRINT"RISULTATI"
310 G=N*SC/(N*SC+SA)
320 ZZ=1-G/3+(G/3-LA)*(G-LA)*MU/(1-G)
330 AL=SQR((1-(G-LA)*MU/(1-G))*2/(SC*G*ZZ))
340 H=AL*SQR(M/B):Y=G*H:H1=LA*H
350 BE=1/(AL*SA*ZZ)
360 A1=BE*SQR(M*B):A2=MU*A1
370 DEFFNA(X)=INT(X*100+.5)/100
380 PRINT:PRINT"ASSE NEUTRO  Y="FNA(Y)"CM"
390 PRINT:PRINT"ALTEZZA      H="FNA(H)"CM"
400 PRINT"ARMATURA      A1="FNA(A1)"CM12"

```



```

410 IFA2=0THEN440
420 PRINT"ARMATURA      A2="FNA(A2)"CM^2"
430 PRINT"COPRIFERRO    H1="FNA(H1)"CM"
440 PRINT:INPUT"VUOI LA STAMPA DEI RISULTATI (S/N)";C$:IFC$="S"THENGOSUB1000
450 PRINT:INPUT"ALTRO CALCOLO (S/N)";B$:IFB$="S"THENPRINTCHR$(147):GOTO130
460 END
500 PRINT:PRINT"    DATO ERRATO : M<0":PRINT:SE=SE-1:GOTO130
1000 OPEN1,4:PRINT#1:PRINT#1
1010 PRINT#1,TAB(15)"PROGETTO SEZIONI RETTANGOLARI IN C.A.":PRINT#1
1015 PRINT#1,TAB(17)"SOLLECITATE A MOMENTO FLETTENTE"
1020 PRINT#1:PRINT#1
1030 PRINT#1,"SEZIONE N="SE
1040 PRINT#1:PRINT#1
1050 PRINT#1,"DATI DI INPUT"
1060 PRINT#1:PRINT#1,"RAPPORTO DI OMOG.      N="N
1070 PRINT#1,"TENSIONE CONGLOMERATO    SC="SC"KG/CM^2"
1080 PRINT#1,"TENSIONE ACCIAIO        SA="SA"KG/CM^2"
1090 PRINT#1,"LARGHEZZA SEZIONE        B="B"CM"
1100 PRINT#1,"RAPPORTO A2/A1          MU="MU
1110 IFA2=0THENC=0:GOTO1130
1120 PRINT#1,"RAPPORTO H1/H          LA="LA
1130 PRINT#1,"MOMENTO FLETTENTE        M="M"KGCM
1140 PRINT#1:PRINT#1
1150 PRINT#1,"RISULTATI":PRINT#1
1160 PRINT#1,"ASSE NEUTRO          Y="FNA(Y)"CM":PRINT#1
1170 PRINT#1,"ALTEZZA            H="FNA(H)"CM"
1180 PRINT#1,"ARMATURA TESA      A1="FNA(A1)"CM^2"
1190 IFA2=0THEN1220
1200 PRINT#1,"ARMATURA COMPR. A2="FNA(A2)"CM^2"
1210 PRINT#1,"COPRIFERRO      H1="FNA(H1)"CM"
1220 PRINT#1:PRINT#1
1230 CLOSE1
1240 RETURN

```

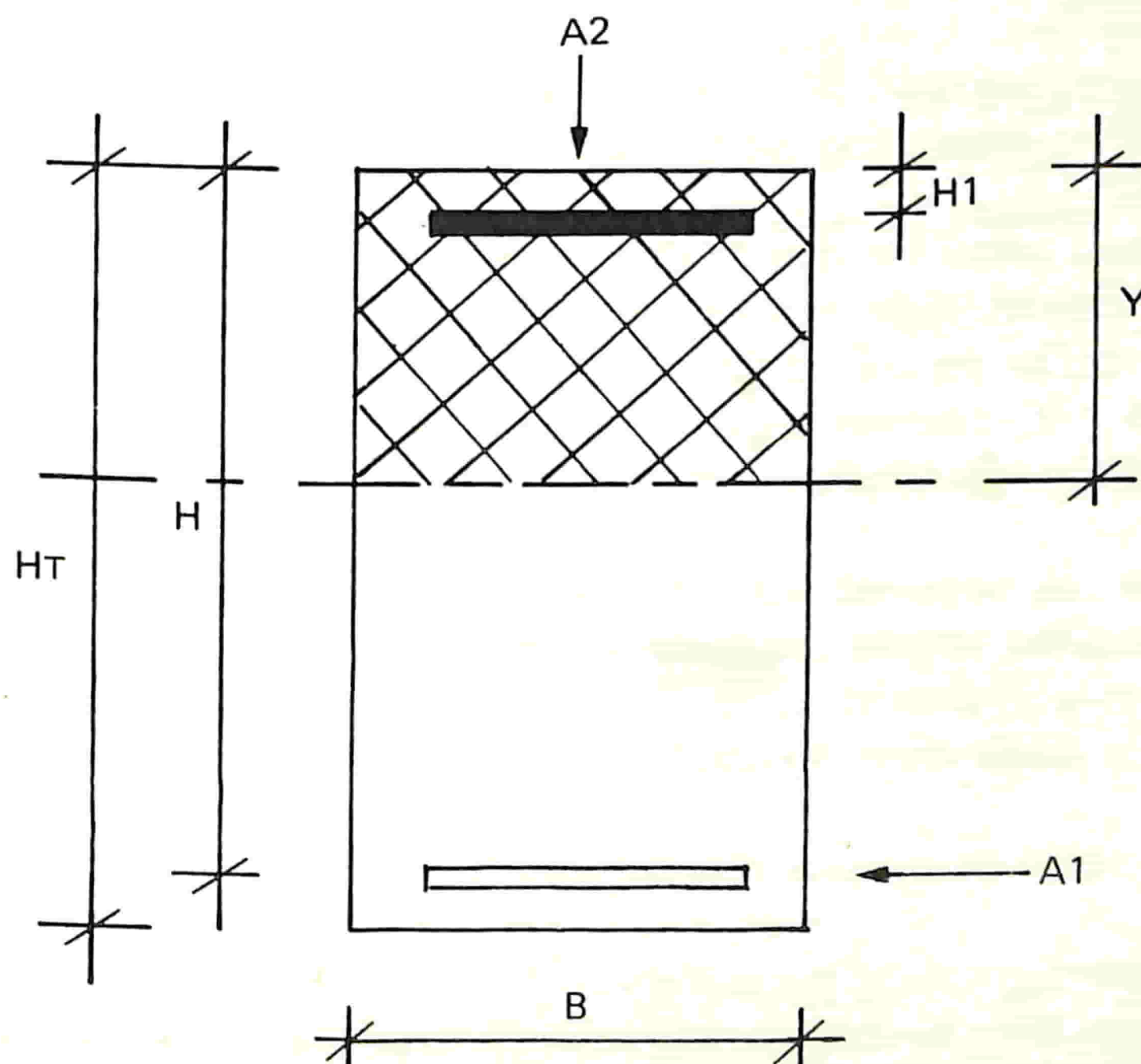


Fig. 8.1 Simboli usati nel programma di progetto di sezioni rettangolari in cemento armato inflesse

Elenco delle variabili principali.

- N = coefficiente di omogeneizzazione
 SC = tensione ammissibile calcestruzzo
 SA = tensione ammissibile acciaio
 B = larghezza sezione
 MU = rapporto A_2/A_1
 LA = rapporto H_1/H
 M = momento flettente
 H = altezza utile
 Y = posizione asse neutro
 H1 = copriferro armatura compressa (distanza tra il baricentro dell'armatura compressa e il lembo compresso)
 A1 = armatura tesa
 A2 = armatura compressa

Commento al programma

Con riferimento alle notazioni di fig. 8.1, il programma chiede come dati di ingresso

- coefficiente di omogeneizzazione N
- tensione ammissibile del conglomerato SC
- tensione ammissibile dell'acciaio SA

Per questi primi tre dati sono già previsti nel programma dei valori predeterminati, rispettivamente $N = 15$, $SC = 97.5 \text{ KG/CM} \uparrow 2$ (corrispondente ad un calcestruzzo con $RBK > = 300 \text{ kg/cm}^2$), $SA = 2200 \text{ KG/CM} \uparrow 2$ (corrispondente ad acciaio FEB38K controllato o FEB44K non controllato), ciò al fine di accelerare l'introduzione dei dati.

I valori che appaiono sul video possono comunque essere modificati con i dati del singolo problema, semplicemente scrivendo i nuovi valori al loro posto. Se si usano normalmente valori diversi per SC e SA, conviene sostituire tali valori direttamente nel programma alla riga 120.

Gli altri dati di ingresso richiesti sono

- la larghezza della sezione B
- il rapporto tra l'armatura compressa e quella tesa A_2/A_1 , MU
- il rapporto tra il copriferro H1 e l'altezza utile H, LA
- il momento flettente M che sollecita la sezione, espresso in kgcm

Avvertenza: il programma è predisposto per il calcolo di sezioni soggette a momento flettente positivo; se la sezione è sollecitata a momento negativo, basta considerare la fig. 8.1 capovolta e introdurre i dati di conseguenza.

I calcoli vengono eseguiti con le usuali formule (cfr. S. Di Pasquale e altri, *Costruzioni*, Le Monnier, Firenze).

Come output su video vengono forniti i dati di ingresso, per controllo, e i risultati:

- posizione dell'asse neutro Y
- altezza utile H
- armatura tesa $A1$
- armatura compressa $A2$
- copriferro $H1$.

Se il calcolo viene effettuato a semplice armatura gli ultimi due risultati non vengono stampati.

Se si ha a disposizione una stampante Commodore MPS 801, o altra compatibile con il Commodore 64, si può ottenere la stampa degli output rispondendo **S** alla richiesta **VUOI STAMPARE I RISULTATI**.

Se si eseguono più calcoli in successione, le sezioni vengono automaticamente numerate in maniera progressiva.

Per ottenere l'altezza totale della sezione HT , basta aggiungere all'altezza utile H il copriferro, secondo la normativa.

Tutti i risultati, su video e su stampante, vengono forniti con l'approssimazione di due cifre decimali. Se si vuole modificare tale precisione bisogna intervenire nel programma, modificando la funzione $FNA(X)$ definita alla riga 370.

Esempi di calcolo

Esempio 1

Si progetti la sezione di una trave rettangolare soggetta a momento flettente $M = 12000$ kgm, armata a doppia armatura. La larghezza della sezione è $B = 30$ cm, il calcestruzzo classe 300, l'acciaio FEB38K controllato, $N = 15$.

Soluzione

Assumiamo come rapporto tra l'armatura compressa e quella tesa $\mu = 0.5$, e, come rapporto tra il copriferro $H1$ e l'altezza utile H , $\lambda = 0.07$.

L'output sulla stampante è il seguente

2206=11057401102=111601120011011

SOLLECITATE A MOMENTO FLETTENTE

874011=11 1

01000012011

RAPPORTO DI OMOG.	N= 15
TENSIONE CONGLOMERATO	SC= 97.5 KG/CM ¹²
TENSIONE ACCIAIO	SA= 2200 KG/CM ¹²
LARGHEZZA SEZIONE	B= 30 CM
RAPPORTO A2/A1	MU= .5
RAPPORTO H1/H	LA= .07
MOMENTO FLETTENTE	M= 1200000 KGCM

205011111

ASSE NEUTRO	Y= 16.4 CM
ALTEZZA	H= 41.07 CM
ARMATURA TESA	A1= 15.02 CM ¹²
ARMATURA COMPR.	A2= 7.51 CM ¹²
COPRIFERRO	H1= 2.88 CM

Adottiamo come armatura tesa 8 Ø 16

$$A1 = 16.08 \text{ cm}^2$$

e come armatura compressa 4 Ø 16

$$A2 = 8.04 \text{ cm}^2$$

Per ottenere l'altezza totale HT aggiungiamo all'altezza utile H metà diametro Ø 16, più lo spessore della staffa, supponiamo Ø 10, più almeno due centimetri di copriferro:

$$HT = 41.07 + 0.8 + 1.0 + 2.0 = 44.87 \cong 45 \text{ cm}$$

La posizione del baricentro dell'armatura compressa sarà a

$$H1 = 2.0 + 1.0 + 0.8 = 3.8 \text{ cm}$$

dal brodo compresso.

Esempio 2

Si progetti una sezione a T, soggetta ad un momento flettente positivo $M = 24000 \text{ kgm}$, armata a semplice armatura. I dati geometrici della sezione sono $B = 60 \text{ cm}$, $B0 = 25 \text{ cm}$, $S = 20 \text{ cm}$.

I materiali sono: calcestruzzo classe 250, acciaio FEB44K non controllato; $N = 10$.

Soluzione

Supponiamo, in prima approssimazione, che l'asse neutro tagli le ali della sezione, per cui possiamo assimilare la sezione a T a una sezione rettangolare.

Introdotti i dati, l'output sulla stampante è il seguente

```

=====

```

SOLLECITATE A MOMENTO FLETTENTE

```

===== 1

```

```

=====

```

RAPPORTO DI OMOG.	N= 10
TENSIONE CONGLOMERATO	SC= 85 KG/CM ²
TENSIONE ACCIAIO	SA= 2200 KG/CM ²
LARGHEZZA SEZIONE	B= 60 CM
RAPPORTO A2/A1	MU= 0
MOMENTO FLETTENTE	M= 2400000 KGCM

```

=====

```

ASSE NEUTRO Y= 17 CM

ALTEZZA H= 61.02 CM

ARMATURA TESA A1= 19.71 CM²

Controlliamo la posizione dell'asse neutro:

$$Y = 17 < S = 20 \text{ cm}$$

l'asse neutro taglia le ali della sezione, il dimensionamento è quindi valido.

Come armatura adottiamo 8 Ø 18

$$A1 = 20.36 \text{ cm}^2$$

L'armatura sarà disposta su due file, dato il piccolo spessore della costola ($B0 = 25 \text{ cm}$). Per ottenere l'altezza totale HT dovremo quindi aggiungere all'altezza utile H metà spessore dell'interferro, l'altezza di una barra Ø 18, lo spessore della staffa, supponiamo Ø 10, e almeno 2 cm di copriferro; pertanto

$$HT = 61.02 + 1.0 + 1.8 + 1.0 + 2.0 = 66.82 \cong 70 \text{ cm}$$

VERIFICA DI SEZIONI RETTANGOLARI IN C.A.

Problema: verificare una o più sezioni rettangolari in cemento armato, ad armatura semplice o doppia, sollecitate a momento flettente.

Lista del programma

```

100 REM VERIFICA DI SEZIONI RETTANGOLARI IN C.A. SOLLECITATE A MOMENTO FLETTENTE
110 REM ARMATURA SEMPLICE O DOPPIA
120 PRINTCHR$(147):N=15:H1=3.5
130 PRINTTAB(6)"VERIFICA SEZ. RETTANGOLARI":PRINT
140 PRINTTAB(15)"IN C.A."
150 SE=SE+1:PRINT
160 PRINT"SEZIONE N"SE
170 PRINT:PRINT
180 PRINT"INPUT DATI":PRINT
190 INPUT"ALTEZZA (IN CM)          H=";H
200 INPUT"LARGHEZZA (IN CM)       B=";B
210 INPUT"ARMATURA A1 (IN CM^2)   A1=";A1
220 INPUT"ARMATURA A2 (IN CM^2)   A2=";A2
230 IFA1=0AND A2=0THEN500
240 INPUT"MOMENTO FLETTENTE(IN KGCM) M=";M:IFM<0THEN510
250 PRINT:IFA2=0THENH1=0:GOTO270
260 PRINT"COPRIFERRO (IN CM)      H1= "H1";INPUT"|||||";H1
270 PRINT"RAPPORTO DI OMOGENEIZZAZ. N= "N";INPUT"|||||";N
280 PRINT:PRINT:INPUT"MI DATI SONO ESATTI (S/N) S||||";A$
285 IFA$<"S"THENPRINT:GOTO180
290 PRINTCHR$(147):PRINT:PRINTTAB(14)"SEZIONE N"SE:PRINT:PRINT"DATI DI INPUT"
300 PRINT:PRINT"N="N,"H="H,"B="B,"A1="A1,"A2="A2,"H1="H1,"M="M
310 PRINT:PRINT"RISULTATI"
320 Y=N*(A1+A2)/B*(-1+SQRT(1+2*B*(A1*H+A2*H1)/(N*(A1+A2)^2)))
330 SC=M/(B*Y/2*(H-Y/3)+N*A2*(Y-H1)*(H-H1)/Y)
340 SA=N*SC*(H-Y)/Y
350 S1=N*SC*(Y-H1)/Y
360 DEFFNA(X)=INT(X*100+.5)/100
370 PRINT:PRINT"POSIZ. ASSE NEUTRO Y="FNA(Y)"CM"
380 PRINT:PRINT"TENS. CALCESTRUZZO SC="FNA(SC)"KG/CM^2"
390 PRINT:PRINT"TENS. ACCIAIO TESO SA="FNA(SA)"KG/CM^2"
392 IFA2=0THEN400
395 PRINT:PRINT"TENS. ACCIAIO COMP. S1="FNA(S1)"KG/CM^2"
400 PRINT:PRINT:INPUT"VUOI LA STAMPA DEI RISULTATI (S/N)";B$
405 IFB$="S"THENGOSUB1000
410 PRINT:INPUT"ALTRO CALCOLO (S/N)";C$:IFC$="S"THENPRINTCHR$(147):GOTO130
420 END
500 PRINT:PRINT"DATI ERRATI: A1=0,A2=0":PRINT:SE=SE-1:GOTO130
510 PRINT:PRINT"DATO ERRATO: M<0:?:SE=SE-1:GOTO130
1000 OPEN1,4:PRINT#1:PRINT#1
1010 PRINT#1,TAB(15)"VERIFICA SEZIONI RETTANGOLARI IN C.A.":PRINT#1
1020 PRINT#1,TAB(17)"SOLLECITATE A MOMENTO FLETTENTE"
1030 PRINT#1:PRINT#1
1040 PRINT#1,"SEZIONE N"SE
1050 PRINT#1:PRINT#1
1060 PRINT#1,"DATI DI INPUT":PRINT#1
1070 PRINT#1,"RAPPORTO DI OMOG. N="N
1080 IFA1=0THEN1100
1090 PRINT#1,"COPRIFERRO          H1="H1"CM"
1100 PRINT#1,"ALTEZZA             H="H"CM"
1110 PRINT#1,"LARGHEZZA          B="B"CM"
1120 PRINT#1,"ARMATURA TESA      A1="A1"CM^2"
1125 IFA2=0THEN1140
1130 PRINT#1,"ARMATURA COMPR.      A2="A2"CM^2"
1140 PRINT#1,"MOMENTO FLETTENTE    M="M"KGCM"

```



```
1150 PRINT#1:PRINT#1
1160 PRINT#1,"RISULTATI":PRINT#1
1170 PRINT#1,"POSIZ. ASSE NEUTRO      Y="FNA(Y)"CM":PRINT#1
1180 PRINT#1,"TENSIONE CALCESTRUZZO  SC="FNA(SC)"KG/CM12"
1190 PRINT#1,"TENSIONE ACCIAIO TESO   SA="FNA(SA)"KG/CM12"
1195 IFA2=0THEN1210
1200 PRINT#1,"TENSIONE ACCIAIO COMPR. S1="FNA(S1)"KG/CM12"
1210 PRINT#1:PRINT#1
1220 CLOSE1:RETURN
```

Elenco delle variabili principali

H = altezza utile
B = larghezza sezione
A1 = armatura tesa
A2 = armatura compressa
M = momento flettente
H1 = copriferro armatura compressa (distanza tra il baricentro dell'armatura compressa e il lembo compresso)
N = coefficiente di omogeneizzazione
Y = posizione asse neutro
SC = tensione massima conglomerato
SA = tensione massima acciaio teso
S1 = tensione massima acciaio compresso

Commento al programma

Con riferimento alle notazioni di fig. 8.1, il programma chiede come dati di ingresso:

- l'altezza utile H
- la larghezza della sezione B
- l'armatura tesa A1
- l'armatura compressa A2
- il momento flettente M (in kgcm)
- il copriferro H1
- il coefficiente di omogeneizzazione N.

Per gli ultimi due dati sono previsti nel programma dei valori predefiniti, rispettivamente $N = 15$ per il coefficiente di omogeneizzazione, e $H1 = 3.5$ cm per la distanza dal baricentro delle armature compresse al lembo superiore compresso, sinteticamente indicata come **COPRIFERRO**.

I valori in questione sono stati inseriti nel programma allo scopo di accelerare l'input dei dati. Possono tuttavia essere modificati, semplicemente scrivendo i nuovi valori al posto di quelli che appaiono sul video. Se si vuole invece modificarli permanentemente, si può operare direttamente sul programma, inserendo i nuovi valori alla riga 120.

Avvertenza: il programma è predisposto per il calcolo di sezioni soggette a momento flettente positivo; se si vogliono verificare sezioni soggette a momento negativo, basta considerare la fig. 8.1 come capovolta ed introdurre i dati di conseguenza.

I calcoli vengono eseguiti con le usuali formule (cfr. S. Di Pasquale e altri, *Costruzioni*, Le Monnier, Firenze).

Come output su video vengono stampati i dati di ingresso, per controllo, e i risultati:

- posizione dell'asse neutro Y
- tensione massima nel calcestruzzo SC
- tensione dell'acciaio teso SA
- tensione dell'acciaio compresso $S1$

Se la sezione è armata a semplice armatura, l'ultimo dato non viene stampato.

Se si ha a disposizione una stampante Commodore MPS 801, o altra compatibile con il Commodore 64, si può ottenere la stampa degli output su carta, rispondendo **S** alla richiesta **VUOI LA STAMPA DEI RISULTATI**.

Se vengono eseguiti calcoli per più sezioni in successione, il programma provvede automaticamente a numerare le varie sezioni in maniera progressiva.

Tutti gli output, sia su video sia su stampante, vengono forniti con l'approssimazione di due cifre decimali. Se si vuole modificare tale precisione, basta intervenire nel programma alla riga 360, modificando la funzione $FNA(X)$ ivi definita.

Esempi di calcolo

Esempio 1

Si verifichi la sezione di una trave rettangolare di dimensioni $B = 30$ cm, $HT = 45$ cm, soggetta a momento flettente positivo $M = 12000$ kgm. L'armatura tesa è costituita da 8 $\varnothing 16$ ($A1 = 16.08$ cm²) e quella compressa da 4 $\varnothing 16$ ($A2 = 8.04$ cm²).

I materiali sono calcestruzzo con $RBK > = 300$ kg/cm² e acciaio

FEB44K non controllato; il coefficiente di omogeneizzazione è $N = 15$. La staffatura è costituita da staffe $\varnothing 10$.

Soluzione

Si determina dapprima l'altezza utile della sezione, dal baricentro delle armature tese al lembo superiore compresso, sottraendo all'altezza totale HT il copriferro (2 cm), lo spessore della staffa $\varnothing 10$ e metà spessore della barra $\varnothing 16$:

$$H = 45 - 2.0 - 1.0 - 0.8 = 41.2 \text{ cm}$$

Analogamente si calcola la distanza dal lembo compresso del baricentro delle armature compresse $H1$, indicata nel programma come CO-PRIFERRO,

$$H1 = 2.0 + 1.0 + 0.8 = 3.8 \text{ cm}$$

Si inseriscono quindi i dati nel programma.
L'output in stampa è il seguente

```

PROGRAMMA PER IL CALCOLO DI SEZIONI RETTANGOLARI

```

```

SOLLECITATE A MOMENTO FLETTENTE

```

```

DATI DI INPUT

```

```

=====

```

```

RAPPORTO DI OMOG.  N= 15
COPRIFERRO          H1= 3.8 CM
ALTEZZA              H= 41.2 CM
LARGHEZZA            B= 30 CM
ARMATURA TESA        A1= 16.08 CM^2
ARMATURA COMPR.      A2= 8.04 CM^2
MOMENTO FLETTENTE    M= 1200000 KGCM

```

```

=====

```

```

POSIZ. ASSE NEUTRO    Y= 16.9 CM
TENSIONE CALCESTRUZZO SC= 95.92 KG/CM^2
TENSIONE ACCIAIO TESO  SA= 2069.44 KG/CM^2
TENSIONE ACCIAIO COMPR. SI= 1115.19 KG/CM^2

```

La tensione massima nel calcestruzzo risulta

$$SC = 95.92 < 97.5 \text{ kg/cm}^2, \text{ tensione ammissibile;}$$

la tensione nell'acciaio teso risulta

$$SA = 2069.44 < 2200 \text{ kg/cm}^2, \text{ tensione ammissibile.}$$

La verifica risulta quindi soddisfatta.

Esempio 2

Sia data una sezione a T di dimensioni $B = 60$ cm, $B_0 = 25$ cm, $S = 20$ cm e $HT = 70$ cm. La sezione, soggetta a momento flettente positivo $M = 24000$ kgm, è armata solamente nella parte inferiore con 8 $\varnothing 18$ ($A_1 = 20.36$ cm²) disposti su due file.

I materiali sono calcestruzzo con $R_{BK} \geq 250$ kg/cm² e acciaio FEB38K controllato; il rapporto di omogeneizzazione sia $N = 10$. Le staffe sono $\varnothing 10$.

Si chiede la verifica della sezione.

Soluzione

Si calcola dapprima l'altezza utile H , determinando la posizione del baricentro delle armature tese. All'altezza totale HT va sottratto il copriferro (2 cm), lo spessore della staffa (1 cm), lo spessore di una barra $\varnothing 18$ e metà dell'interferro tra le due file di barre (1 cm):

$$H = 70 - 2.0 - 1.0 - 1.8 - 1.0 = 64.2 \text{ cm.}$$

Si introducono quindi i dati nel programma.

L'output in stampa è il seguente

```

=====
SOLLECITATE A MOMENTO FLETTENTE

===== 1

=====

RAPPORTO DI OMOG.  N= 10
ALTEZZA             H= 64.2 CM
LARGHEZZA          B= 60 CM
ARMATURA TESA      A1= 20.36 CM^2
MOMENTO FLETTENTE  M= 2400000 KGCM

=====

POSIZ. ASSE NEUTRO      Y= 17.75 CM

TENSIONE CALCESTRUZZO  SC= 77.31 KG/CM^2
TENSIONE ACCIAIO TESO  SA= 2022.55 KG/CM^2

```

L'asse neutro è contenuto nello spessore delle ali:

$$Y = 17.75 < 20 \text{ cm}$$

quindi il calcolo come sezione rettangolare è valido.

La tensione massima nel calcestruzzo è

$SC = 77.31 < 85 \text{ kg/cm}^2$, tensione ammissibile;

la tensione nell'acciaio è

$SA = 2022.55 < 2200 \text{ kg/cm}^2$, tensione ammissibile.

La verifica risulta quindi soddisfatta.

MURI DI SOSTEGNO E GRAVITÀ

Problema: progettare e verificare un muro di sostegno a gravità a sezione rettangolare, soggetto ad una spinta orizzontale.

Lista del programma

```

100 REM PROGETTO DI MURI DI SOSTEGNO A GRAVITA' DI SEZIONE RETTANGOLARE
110 PRINTCHR$(147)
120 PRINT"PROGETTO DI MURI DI SOSTEGNO A GRAVITA'"
130 PRINTTAB(9)"A SEZIONE RETTANGOLARE"
140 PRINT:PRINT
150 PRINT"DATI DI PROGETTO"
160 PRINT:INPUT"ALTEZZA (M)"H=";H
170 PRINT:INPUT"SPINTA (KG)"S=";S
180 PRINT:INPUT"PESO SPECIFICO (KG/M^3)"GM=";GM
190 PRINT:INPUT"COEFF. SICUR. RIBALTAMENTO CR=";CR
200 PRINT:PRINT:INPUT"SI DATI SONO ESATTI (S/N) SIIIIII";AA$
210 IFAA$<>"S"THEN140
215 B=SQR(S*2*CR/GM/3)
220 IFINT(B*10)<INT(B*10+.5)THENB=(INT(B*10)+1)/10:GOTO230
225 B=(INT(B*10)+.5)/10
230 PRINTCHR$(147):PRINTTAB(10)"CALCOLO DI PROGETTO"
240 PRINT:PRINT:PRINT"DATI DI INGRESSO"
250 PRINT:PRINT"H="H"M", "S="S"KG"
260 PRINT:PRINT"CR="CR, "GM="GM"KG/M^3"
270 PRINT:PRINT:PRINT"SEZIONE DEL MURO"
280 PRINT:PRINTB"X"H"M"
290 PRINT:PRINT:PRINT"VUOI MODIFICARE LA BASE B: INPUT"IIIIIIII";B
300 PT=B*H*GM:MS=S*H/3:MR=PT*B/2
310 KR=MR/MS:IFKR<CRTHENWW=1:GOTO1000
320 PRINTCHR$(147):PRINTTAB(8)"VERIFICA AL RIBALTAMENTO":PRINT:PRINT
330 PRINT"H="H"M", "S="S"KG"
340 PRINT:PRINT"CR="CR, "GM="GM"KG/M^3"
350 PRINT:PRINT"B="B"M"
360 PRINT:PRINT:PRINT"MR/MS="INT(KR*100+.5)/100">="CR"VERIFICA SODDISFATTA"
365 PRINT:PRINT:PRINT"PER CONTINUARE BATTI UN TASTO"
370 GETA1$: IFA1$=""THEN370
380 PRINTCHR$(147):PRINTTAB(7)"VERIFICA ALLO SCORRIMENTO"
390 PRINT:PRINT:INPUT"COEFF. SICUR. A SCORRIMENTO CS=";CS
400 PRINT:PRINT:PRINT"TIPO DI TERRENO DI FONDAZIONE"
410 PRINT:PRINT"1) TERRENO SABBIOSO F=0.6"
420 PRINT:PRINT"2) TERRENO ORDINARIO COMPATTO ASCIUTTO F=0.5"
430 PRINT:PRINT"3) ARGILLA CON INTERPOSIZIONE DI UNO "
435 PRINTTAB(3)"STRATO DI GHIAIA E SABBIA DI ALMENO "
440 PRINTTAB(3)"10 CM F=0.35"

```



```

1630 PRINT#1,"RISULTATI":PRINT#1
1640 PRINT#1,"SEZIONE DEL MURO":PRINT#1
1650 PRINT#1,"ALTEZZA  H="H"M"
1660 PRINT#1,"BASE      B="B"M"
1670 PRINT#1:PRINT#1
1680 PRINT#1,"VERIFICA AL RIBALTAMENTO":PRINT#1
1690 PRINT#1,"MR/MS="INT(KR*1000+.5)/1000"> CR="CR"  VERIFICA SODDISFATTA"
1700 PRINT#1:PRINT#1,"VERIFICA ALLO SCORRIMENTO":PRINT#1
1710 PRINT#1,"PT*F/S="INT(KS*1000+.5)/1000"> CS="CS"  VERIFICA SODDISFATTA"
1720 PRINT#1:PRINT#1,"VERIFICA ALLO SCHIACCIAMENTO":PRINT#1
1730 IF ZZ=1 THEN PRINT#1,"SEZIONE PARZIALIZZATA: 3*U="3*INT(U*100+.5)/100"CM"
1740 PRINT#1
1750 IF ZZ=1 THEN PRINT#1,"KMIN="INT(K2*1000+.5)/1000"KG/CM^2":PRINT#1
1760 PRINT#1,"KMAX="INT(K1*1000+.5)/1000"<= KT="KT"KG/CM^2  ";
1770 PRINT#1,"VERIFICA SODDISFATTA"
1780 PRINT#1:PRINT#1
1790 CLOSE1
1800 RETURN

```

Elenco delle variabili principali

H = altezza
 S = spinta
 GM = peso specifico muro
 CR = coefficiente di sicurezza a ribaltamento
 B = base, in metri
 PT = peso del muro
 MS = momento spingente
 MR = momento resistente
 KR = rapporto MR/MS
 CS = coefficiente di sicurezza a scorrimento
 F = coefficiente attrito terreno
 KS = rapporto $PT \times F/S$
 KT = tensione ammissibile terreno
 U = distanza dello sforzo sulla base dal bordo esterno
 B1 = base, in cm
 E = eccentricità dello sforzo sulla base
 M = momento flettente alla base
 W = modulo di resistenza della base
 A = area della base
 K1 = tensione massima del terreno
 K2 = tensione minima del terreno

Commento al programma

Il programma permette di progettare e verificare muri di sostegno a gravità, di sezione rettangolare, soggetti a spinte orizzontali. La superficie del terreno è orizzontale e senza sovraccarico (fig. 8.2).

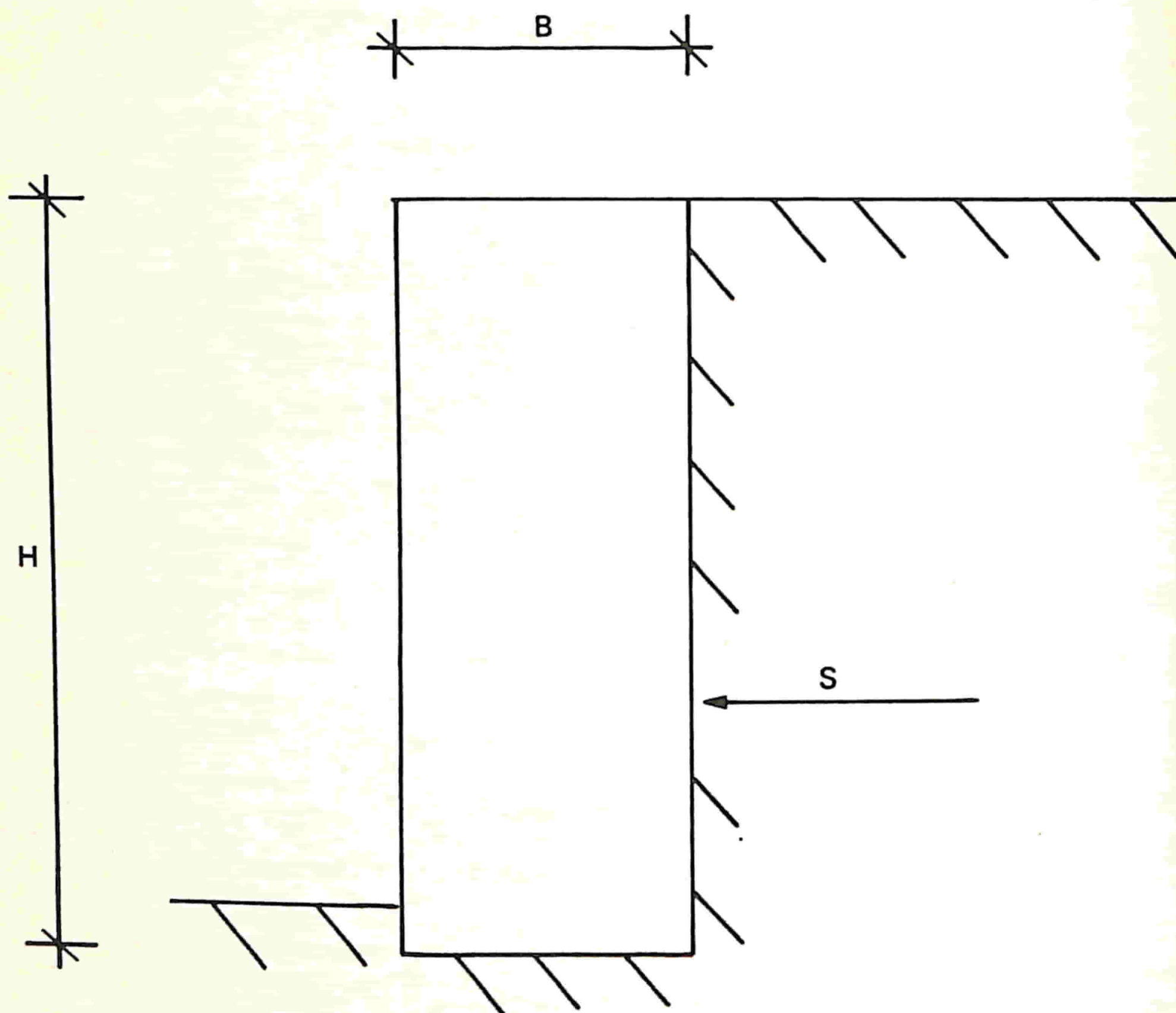


Fig. 8.2 Simboli usati nel programma di progetto di muri di sostegno a gravità

I dati e i calcoli sono riferiti alla lunghezza di 1 metro di muro. Come dati di ingresso sono richiesti

- l'altezza del muro H
- il valore della spinta S
- il peso specifico del muro GM
- il coefficiente di sicurezza a ribaltamento CR .

Terminato l'input dei dati, viene eseguito, per via analitica, il calcolo di progetto.

L'output su video riporta i dati di ingresso, per controllo, e, come risultato, la sezione del muro.

È prevista la possibilità di modificare la base B dimensionata con il calcolo. Per modificare la base (aumentandola) basta scrivere il nuovo valore al posto di quello che appare sul video. Se non si vuole modificare la base, basta battere il tasto RETURN.

Dopo tale operazione il programma continua eseguendo il calcolo di verifica a ribaltamento.

Sul video vengono stampati i dati di ingresso, compreso il valore di B , e il risultato del calcolo di verifica. Per continuare l'esecuzione del programma, battere un qualsiasi tasto.

Se la verifica a ribaltamento non è soddisfatta, viene stampato comunque sul video il risultato della verifica, insieme a un menù che prevede la scelta tra: modificare i dati iniziali, modificare solo la base B , o terminare l'esecuzione del programma.

Dopo la verifica al ribaltamento viene eseguita la verifica allo scorrimento. Vengono richiesti come dati:

- il coefficiente di sicurezza allo scorrimento CS
- il coefficiente di attrito del terreno F

L'introduzione di quest'ultimo dato viene fatta sotto forma di un menù in cui sono previsti i quattro tipi più comuni di terreno, con i relativi coefficienti di attrito (cfr. S. Di Pasquale e altri, *Costruzioni*, Le Monnier, Firenze).

Scelto il tipo di terreno, viene eseguito il calcolo di verifica allo scorrimento. Sul video vengono stampati i dati necessari per il calcolo di verifica ed il risultato della stessa.

Se la verifica non risulta soddisfatta, oltre al risultato del calcolo di verifica, viene stampato lo stesso menù visto in precedenza nel caso della verifica a ribaltamento. Si può quindi: modificare i dati iniziali, modificare la base B (e successivamente, se necessario, il coefficiente CS) o concludere l'esecuzione del programma.

Se la verifica a scorrimento è invece soddisfatta, premendo un tasto si passa all'ultima verifica: quella a schiacciamento. Come dato di ingresso viene richiesta

- la tensione ammissibile del terreno KT

Come è noto, nella verifica a schiacciamento, possono presentarsi due casi: nel primo lo sforzo normale sulla base del muro è interno al terzo medio (nocciolo centrale d'inerzia) (fig. 8.3.a), nel secondo lo sforzo normale è esterno al terzo medio (fig. 8.3.b) e la sezione si parzializza.

Nel primo caso il programma fornisce sul video i dati necessari per il calcolo di verifica a schiacciamento, i valori dell'eccentricità E e della distanza dal centro di pressione al bordo esterno del muro U . Vengono infine forniti i valori delle due tensioni del terreno: quella minima $KMIN$, al bordo interno, e quella massima $KMAX$, al bordo esterno del muro, confrontata con la tensione ammissibile del terreno KT .

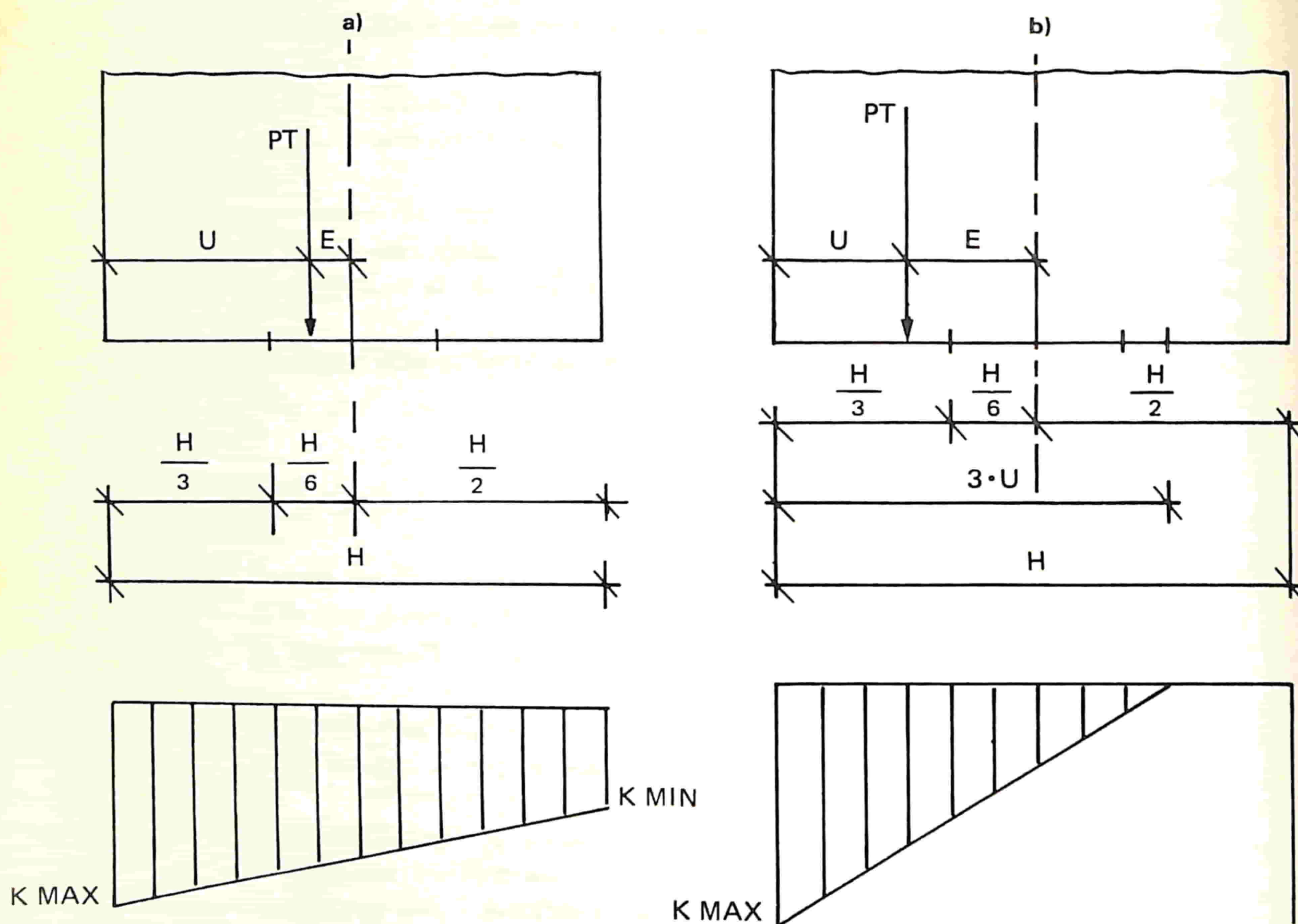


Fig. 8.3 Diagrammi delle pressioni del terreno alla base del muro: a) sezione non parzializzata, b) sezione parzializzata

Nel secondo caso, sezione parzializzata (fig. 8.3.b), il programma fornisce sul video la lunghezza della sezione dopo la parzializzazione $3 \cdot U$, tutti gli stessi dati del primo caso ed il solo valore della tensione massima K_{MAX} , confrontato con la tensione ammissibile del terreno K_T .

Se la verifica non risultasse soddisfatta, perché la sezione è eccessivamente parzializzata ($U < B/6$), o perché la tensione massima del terreno supera quella ammissibile, vengono stampati una scritta che avverte dell'errore e il solito menù che permette di modificare i dati iniziali o la base B .

Se, per ottenere che la verifica a scorrimento o a schiacciamento siano soddisfatte, si fosse costretti ad aumentare eccessivamente la base B , è consigliabile prevedere una fondazione al disotto del muro.

Quando le tre verifiche risultano tutte soddisfatte, appare la richiesta **VUOI LA STAMPA DEI RISULTATI**. Se si ha a disposizione una stam-

pante MPS 801, o un'altra compatibile con il Commodore 64, si può, rispondendo S a tale richiesta, ottenere la stampa dei risultati su carta.

L'output su stampante prevede la stampa di tutti i dati di input, del risultato del calcolo di progetto e delle tre verifiche.

Esempi di calcolo

Esempio 1

Progettare un muro di sostegno a gravità di sezione rettangolare, a sostegno di un terrapieno a superficie orizzontale senza sovraccarico, dell'altezza $H = 2.1$ m.

La spinta risulta essere $S = 1200$ kg, il materiale di costruzione del muro è calcestruzzo non armato ($GM = 2400$ kg/m³). Il terreno di fondazione è un terreno ordinario compatto asciutto, con tensione ammissibile a compressione $KT = 2$ kg/cm².

Soluzione

Si assume come coefficiente di sicurezza a ribaltamento $CR = 1.5$.

Si introducono i dati di progetto:

$$H = 2.1, S = 1200, GM = 2400, CR = 1.5$$

La base del muro risulta essere

$$B = 0.75 \text{ M}$$

che accettiamo come valida.

La verifica a ribaltamento risulta soddisfatta.

Assumiamo come coefficiente di sicurezza a scorrimento $CS = 1.3$.

Inseriamo tale valore e scegliamo il terreno n. 2, che corrisponde al nostro caso.

La verifica allo scorrimento risulta soddisfatta.

Inseriamo il dato relativo alla tensione ammissibile del terreno $KT = 2$.

La sezione risulta parzializzata, con

$$3 * U = 45.84 \text{ CM}$$

e la tensione massima sul terreno

$$K_{MAX} = 1.649 < 2 \text{ KG/CM} \uparrow 2.$$

Anche la verifica allo schiacciamento risulta così soddisfatta.
Chiediamo quindi la stampa dei risultati che risulta la seguente:

```

#####
#####

#####

ALTEZZA                H= 2.1 M
SPINTA                 S= 1200 KG
PESO SPEC. MURO        GM= 2400 KG/M3
COEFF.SICUR. RIBALTAMENTO CR= 1.5
COEFF.SICUR. SCORRIMENTO CS= 1.3
COEFF. ATTRITO TERRENO  F= .5
TENSIONE AMMISS. TERRENO KT= 2 KG/CM2

```

#####

SEZIONE DEL MURO

```

ALTEZZA  H= 2.1 M
BASE     B= .75 M

```

VERIFICA AL RIBALTAMENTO

```

MR/MS= 1.687 >= CR= 1.5 #####

```

VERIFICA ALLO SCORRIMENTO

```

PT*F/S= 1.575 >= CS= 1.3 #####

```

VERIFICA ALLO SCHIACCIAMENTO

```

KMAX= 1.649 <= KT= 2 KG/CM2 #####

```

Esempio 2

Si debba sostenere un terrapieno orizzontale di altezza $H = 3$ m con un muro di sostegno a gravità in calcestruzzo non armato ($GM = 2400$ kg/m³). La spinta risulta essere $S = 2400$ kg.

Il terreno di fondazione è un terreno ordinario compatto asciutto con tensione ammissibile a compressione $KT = 2$ kg/cm².

Si richiede il progetto e la verifica del muro.

Soluzione

Assunto come coefficiente di sicurezza a ribaltamento $CR = 1.5$, si introducono i dati nel programma

$H = 3$, $S = 2400$, $GM = 2400$, $CR = 1.5$

La base risulta essere

$B = 1.05$ M

valore che riteniamo accettabile.

La verifica a ribaltamento risulta soddisfatta.

Inserendo come valore per il coefficiente di sicurezza a scorrimento $CS = 1.3$ e scegliendo il terreno 2, come indicato nel problema, la verifica a scorrimento risulta soddisfatta.

Introdotta il valore della tensione ammissibile del terreno $KT = 2$, la verifica a schiacciamento non risulta invece soddisfatta. La tensione massima del terreno risulta infatti essere

$$K_{MAX} = 2.428 > KT = 2 \text{ kg/cm}^2$$

Decidiamo quindi di modificare la base B. Per operare tale scelta digitiamo 2.

Scriviamo il nuovo valore della base

$$B = 1.15$$

Le prime due verifiche risultano, evidentemente, soddisfatte.

Nella verifica a schiacciamento la sezione risulta parzializzata, con $3 \times U = 85.53 \text{ cm}$ e la massima tensione del terreno risulta

$$K_{MAX} = 1.936 < 2 \text{ kg/cm}^2$$

quindi, con il nuovo valore di B, la verifica risulta soddisfatta.

Chiediamo quindi la stampa dei risultati che risulta essere la seguente:

```
#####
```

```
#####
```

```
#####
```

ALTEZZA	H= 3 M
SPINTA	S= 2400 KG
PESO SPEC. MURO	GM= 2400 KG/M ³
COEFF. SICUR. RIBALTAMENTO	CR= 1.5
COEFF. SICUR. SCORRIMENTO	CS= 1.3
COEFF. ATTRITO TERRENO	F= .5
TENSIONE AMMISS. TERRENO	KT= 2 KG/CM ²

```
#####
```

SEZIONE DEL MURO

ALTEZZA	H= 3 M
BASE	B= 1.15 M

VERIFICA AL RIBALTAMENTO

MR/MS= 1.984 >= CR= 1.5 #####

VERIFICA ALLO SCORRIMENTO

$PT \cdot F/S = 1.725 \geq CS = 1.3$ *OK*

VERIFICA ALLO SCHIACCIAMENTO

$K_{MAX} = 1.936 \leq K_T = 2 \text{ KG/CM}^2$ *OK*

POLIGONALE APERTA PLANOALTIMETRICA

Problema: calcolo di una poligonale aperta planoaltimetrica di N vertici.

Lista del programma

```

100 REM POLIGONALE APERTA PLANOALTIMETRICA
110 PRINTCHR$(147)
120 PRINTTAB(2)"POLIGONALE APERTA PLANOALTIMETRICA"
130 PRINT:PRINT:INPUT"QUANTI SONO I VERTICI";N
140 DIM X(N),Y(N),H(N),AA(N),AZ(N),LS(N),LM(N),LI(N),L(N),D(N),XT(N),YT(N),DT(N)
150 DIM Z1(N),Z2(N)
160 PRINT:PRINT"COORDINATE PRIMO VERTICE"
170 INPUT"X=";X(1):INPUT"Y=";Y(1)
180 PRINT:INPUT"COSTANTE DISTANZIOMETRICA (50/100) 100";K:PRINTCHR$(147)
190 PRINTTAB(7)"INPUT DATI DI RILIEVO"
200 FORI=1TON-1
210 PRINT:PRINT"STAZIONE N°I:PRINT
220 INPUT"ALTEZZA STRUMENTALE";H(I)
230 PRINT:PRINT"PUNTO COLLIMATO" I+1
240 INPUT"ANGOLO AZIMUTALE (GR. CENTES.)";Z1(I):AA(I)=Z1(I)*π/200
250 INPUT"ANGOLO ZENITALE (GR. CENTES.)";Z2(I):AZ(I)=Z2(I)*π/200
260 PRINT:PRINT"LETTURE ALLA STADIA"
270 INPUT"LETTURA SUPERIORE";LS(I)
280 INPUT"LETTURA MEDIA";LM(I)
290 INPUT"LETTURA INFERIORE";LI(I)
300 PRINT:INPUT"I DATI SONO ESATTI (S/N) S";A$
310 IF A$<>"S" THEN 210
320 PRINTCHR$(147)
330 NEXTI
340 FORI=1TON-1
350 L(I)=K*(LS(I)-LI(I))*SIN(AZ(I))^2
360 NEXTI
370 FORI=2TON
380 X(I)=L(I-1)*SIN(AA(I-1))
390 Y(I)=L(I-1)*COS(AA(I-1))
400 D(I)=H(I-1)-LM(I-1)+L(I-1)/TAN(AZ(I-1))
410 NEXTI
420 FORI=1TON:XT(I)=0:YT(I)=0:DT(I)=0:FORJ=1TOI
430 XT(I)=XT(I)+X(J)
440 YT(I)=YT(I)+Y(J)
450 DT(I)=DT(I)+D(J)
460 NEXTJ:NEXTI
470 PRINTCHR$(147)
480 PRINTTAB(3)"COORDINATE E DISLIVELLI PARZIALI"
490 PRINT:PRINT
500 PRINT"N.VE.      X      Y      DELTA"
510 DEFFNA(G)=INT(G*1000+.5)/1000
520 PRINT:FORI=1TON

```



```

530 PRINTTAB(3-LEN(STR$(I)))I;
540 PRINTTAB(14-LEN(STR$(FNA(X(I))))FNA(X(I));
550 PRINTTAB(25-LEN(STR$(FNA(Y(I))))FNA(Y(I));
560 PRINTTAB(37-LEN(STR$(FNA(D(I))))FNA(D(I))
570 NEXTI
580 PRINT:PRINT"PER VEDERE I RISULTATI BATTI UN TASTO"
590 GETB$:IFB$=""THEN590
600 PRINTCHR$(147)
610 PRINTTAB(4)"COORDINATE E DISLIVELLI TOTALI"
620 PRINT:PRINT
630 PRINT"N.VE.      XT      YT      DELTA"
640 PRINT:FORI=1TON
650 PRINTTAB(3-LEN(STR$(I)))I;
660 PRINTTAB(14-LEN(STR$(FNA(XT(I))))FNA(XT(I));
670 PRINTTAB(25-LEN(STR$(FNA(YT(I))))FNA(YT(I));
680 PRINTTAB(37-LEN(STR$(FNA(DT(I))))FNA(DT(I))
690 NEXTI
700 IFXT(N)-X(1)>=0 AND YT(N)-Y(1)>=0THENQ=1:GOTO750
710 IFXT(N)-X(1)>=0 AND YT(N)-Y(1)<0THENQ=2:GOTO750
720 IFXT(N)-X(1)<0 AND YT(N)-Y(1)<=0THENQ=3:GOTO750
730 Q=4
750 IFYT(N)-Y(1)=0 AND Q=1THENA1=100:GOTO820
760 IFYT(N)-Y(1)=0 AND Q=3THENA1=300:GOTO820
770 A1=ATN((XT(N)-X(1))/(YT(N)-Y(1)))
780 AB=(XT(N)-X(1))/SIN(A1):A1=A1*200/π
790 ONQGOTO820,800,800,810
800 A1=A1+200:GOTO820
810 A1=A1+400:GOTO820
820 PRINT:PRINT
830 PRINT"ANGOLO DI DIREZIONE (AB)="A1
840 PRINT
850 PRINT"DISTANZA TOTALE AB="FNA(AB)
860 PRINT:INPUT"VUOI LA STAMPA DEI RISULTATI (S/N)";C$
870 IFC$="S"THEN GOSUB1000
880 PRINTCHR$(147)
890 PRINT:INPUT"ALTRO CALCOLO (S/N)";D$:IFD$="S"THEN CLR:GOTO110
900 END
1000 OPEN1,4:PRINT#1:PRINT#1
1010 PRINT#1,TAB(20)"POLIGONALE PLANOALTIMETRICA"
1020 PRINT#1:PRINT#1
1030 PRINT#1,"NUMERO DEI VERTICI"N
1040 PRINT#1,"COORDINATE PRIMO VERTICE: X(1)="X(1)", Y(1)="Y(1)"
1050 PRINT#1,"COSTANTE DISTANZIOMETRICA K="K
1060 PRINT#1:PRINT#1,"DATI DI RILIEVO"
1070 PRINT#1
1080 PRINT#1,"STAZ.    ALT.STR.    PUNTO    ANG.AZIM.    ANG.ZENIT.";
1085 PRINT#1,"          LETTURE ALLA STADIA"
1090 PRINT#1,"          COLL.TO"TAB(34)"SUP.    MED.    INF."
1100 PRINT#1
1110 FORI=1TON
1120 PRINT#1,TAB(3-LEN(STR$(I)))ITAB(7-LEN(STR$(FNA(H(I))))H(I);
1130 PRINT#1,TAB(11-LEN(STR$(I+1)))I+1TAB(13-LEN(STR$(FNA(Z1(I))))FNA(Z1(I));
1140 PRINT#1,TAB(14-LEN(STR$(FNA(Z2(I))))FNA(Z2(I));
1150 PRINT#1,TAB(10-LEN(STR$(LS(I))))LS(I);
1160 PRINT#1,TAB(7-LEN(STR$(LM(I))))LM(I)TAB(7-LEN(STR$(LI(I))))LI(I)
1170 NEXTI
1180 PRINT#1:PRINT#1
1190 PRINT#1,"COORDINATE E DISLIVELLI PARZIALI":PRINT#1
1200 PRINT#1,"      N. VERTICE      X      Y";
1210 PRINT#1,"      DELTA"
1220 PRINT#1
1230 FORI=1TON
1240 PRINT#1,TAB(10-LEN(STR$(I)))ITAB(20-LEN(STR$(FNA(X(I))))FNA(X(I));
1250 PRINT#1,TAB(20-LEN(STR$(FNA(Y(I))))FNA(Y(I));
1260 PRINT#1,TAB(20-LEN(STR$(FNA(D(I))))FNA(D(I))
1270 NEXTI
1280 PRINT#1:PRINT#1

```



```

1290 PRINT#1,"COORDINATE E DISLIVELLI TOTALI":PRINT#1
1300 PRINT#1,"      N. VERTICE          XT          YT";
1310 PRINT#1,"      DELTA"
1320 PRINT#1
1330 FOR I=1 TO N
1340 PRINT#1,TAB(10-LEN(STR$(I)))ITAB(20-LEN(STR$(FNA(XT(I))))FNA(XT(I));
1350 PRINT#1,TAB(20-LEN(STR$(FNA(YT(I))))FNA(YT(I));
1360 PRINT#1,TAB(20-LEN(STR$(FNA(DT(I))))FNA(DT(I))
1370 NEXT I
1380 PRINT#1:PRINT#1
1390 PRINT#1,"ANGOLO DI DIREZIONE (AB) = "A1
1400 PRINT#1
1410 PRINT#1,"DISTANZA TOTALE AB = "FNA(AB)
1420 CLOSE 1
1430 RETURN

```

Elenco delle variabili principali

N = numero vertici
 K = costante distanziometrica
 A1 = angolo di direzione (AB)
 AB = distanza totale

Elenco dei vettori

X(N) = ascisse stazioni
 Y(N) = ordinate stazioni
 H(N) = altezze strumentali
 Z1(N) = angoli azimutali in gradi centesimali
 Z2(N) = angoli zenitali in gradi centesimali
 AA(N) = angoli azimutali in radianti
 AZ(N) = angoli zenitali in radianti
 LS(N) = letture superiori alla stadia
 LM(N) = letture medie alla stadia
 LI(N) = letture inferiori alla stadia
 L(N) = lunghezze dei lati
 D(N) = dislivelli parziali
 XT(N) = ascisse totali
 YT(N) = ordinate totali
 DT(N) = dislivelli totali

Commento al programma

Con riferimento alla fig. 8.4, il programma chiede come primo dato di ingresso

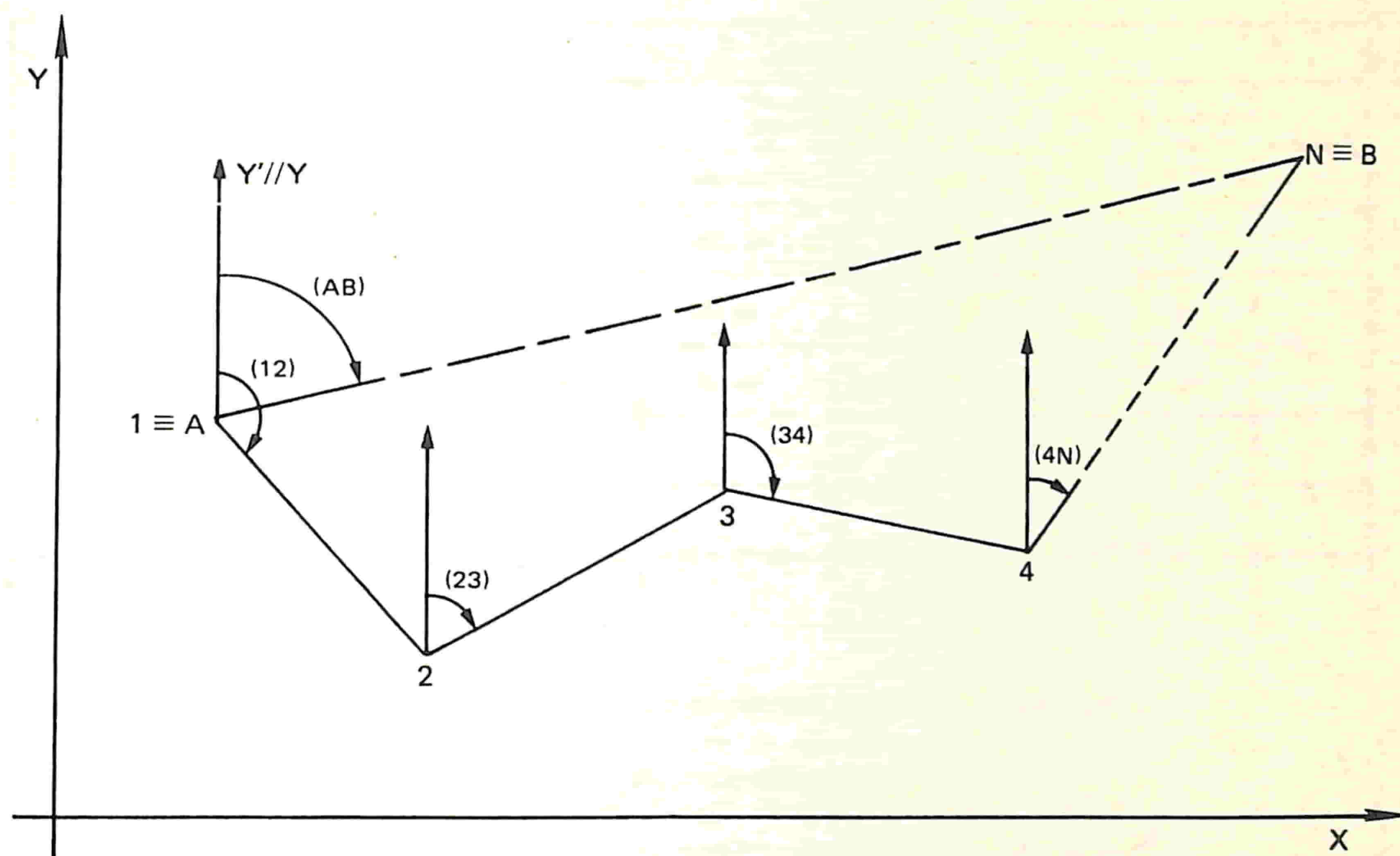


Fig. 8.4 Simboli usati nel programma di calcolo di poligoni aperte

— il numero dei vertici della poligonale N

richiede quindi, successivamente,

— le coordinate del primo vertice
 — la costante distanziometrica

Per questo ultimo dato è già previsto nel programma un valore predeterminato uguale a 100. Il valore che appare sul video può comunque essere modificato con il dato relativo al problema specifico, semplicemente scrivendo il nuovo valore al suo posto. Se il valore della costante distanziometrica è normalmente diverso da 100 (50), conviene sostituire direttamente tale valore nel programma alla riga 180.

Il programma richiede quindi per ogni stazione i seguenti dati:

— altezza strumentale (in metri)
 — angolo azimutale (in gradi centesimali gradazione oraria)
 — angolo zenitale (in gradi centesimali gradazione oraria)
 — le tre letture alla stadia (in metri)

L'input dei dati è facilitato dall'uso di un registro di campagna organizzato secondo lo schema di fig. 8.5.

REGISTRO DI CAMPAGNA								
Stazione	Altezza strum. (metri)	Punti collim.	Letture al cerchio (gr. centesimali)		Letture alla stadia (metri)			NOTE
			Azimutale	Zenitale	S.	M.	I.	
1	1.48	2	50.989	97.316	1.432	0.980	0.528	
2	1.48	3	351.370	99.167	1.972	1.496	1.020	
3	1.52	4	312.963	98.426	1.816	1.380	0.944	

Fig. 8.5 Registro di campagna dell'esempio di calcolo di poligonale aperta

I calcoli vengono eseguiti con le usuali formule (cfr. Bonfigli-Solaini, *Topografia*, Le Monnier, Firenze).

Come output su video vengono forniti le coordinate e i dislivelli parziali tra le varie stazioni e, successivamente, dopo aver premuto un tasto qualsiasi, le coordinate e dislivelli totali delle varie stazioni rispetto alla stazione iniziale.

Come ultimi risultati vengono forniti l'angolo di direzione (AB) e la distanza AB tra il primo e l'ultimo vertice.

Se si ha a disposizione una stampante Commodore MPS 801, o altra compatibile con il Commodore 64, si può ottenere la stampa degli output rispondendo **S** alla richiesta **VUOI LA STAMPA DEI RISULTATI**.

Tutti i risultati, sia su video che su stampante, ad eccezione dell'angolo di direzione (AB), vengono forniti con l'approssimazione di tre cifre decimali. Se si vuole modificare tale precisione bisogna intervenire nel programma modificando la funzione **FNA(G)** definita alla riga 510.

Esempio di calcolo

Per rilevare la poligonale aperta di quattro vertici 1-2-3-4, si è fatto stazione successivamente nei punti 1, 2 e 3 con un tacheometro a gradazione oraria e cannocchiale centralmente anallattico con costante distanziometrica $K = 100$.

I risultati delle misure sono riassunti nel registro di campagna di fig. 8.5.

Si richiede il calcolo della poligonale.

Soluzione

Si introducono il dato relativo al numero dei vertici

$$N = 4$$

le coordinate del primo vertice

$$X = 0$$

$$Y = 0$$

il valore della costante distanziometrica

$$K = 100$$

e successivamente i dati relativi ad ogni stazione, come da libretto di campagna.

L'output su stampante è il seguente

NUMERO DEI VERTICI 4

COORDINATE PRIMO VERTICE: X(1)= 0 , Y(1)= 0
 COSTANTE DISTANZIOMETRICA K= 100

STAZ. ALT. STR. PUNTO COLL. TO ANG. AZIM. ANG. ZENIT. LETTURE ALLA STADIA

STAZ.	ALT. STR.	PUNTO COLL. TO	ANG. AZIM.	ANG. ZENIT.	LETTURE SUP.	LETTURE MED.	LETTURE INF.
1	1.48	2	50.989	97.316	1.432	.98	.528
2	1.48	3	351.37	99.167	1.972	1.496	1.02
3	1.52	4	312.963	98.426	1.816	1.38	.944
4	0	5	0	0	0	0	0

N. VERTICE X Y DELTA

N. VERTICE	X	Y	DELTA
1	0	0	0
2	64.792	62.81	4.307
3	-65.841	68.738	1.23
4	-85.346	17.623	2.295

N. VERTICE XT YT DELTA

N. VERTICE	XT	YT	DELTA
1	0	0	0
2	64.792	62.81	4.307
3	-1.049	131.548	5.536
4	-86.395	149.17	7.831

(AB) = 366.579837

AB = 172.383

Soluzioni degli esercizi

ESERCIZIO 3.1

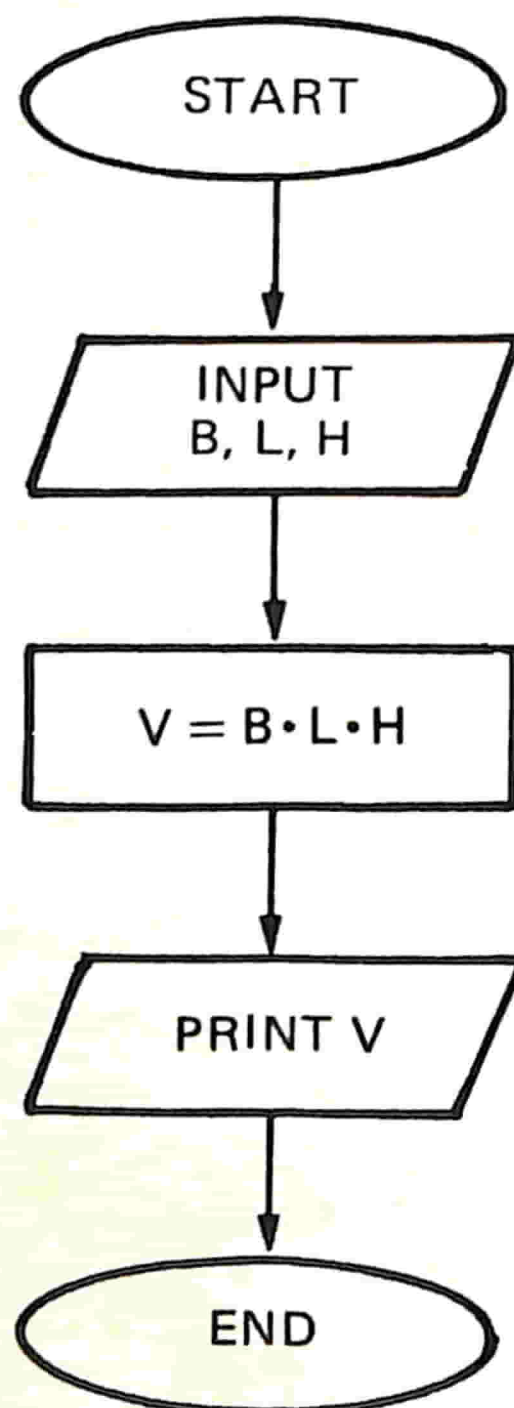


Fig. 9.1 Soluzione esercizio 3.1

ESERCIZIO 3.2

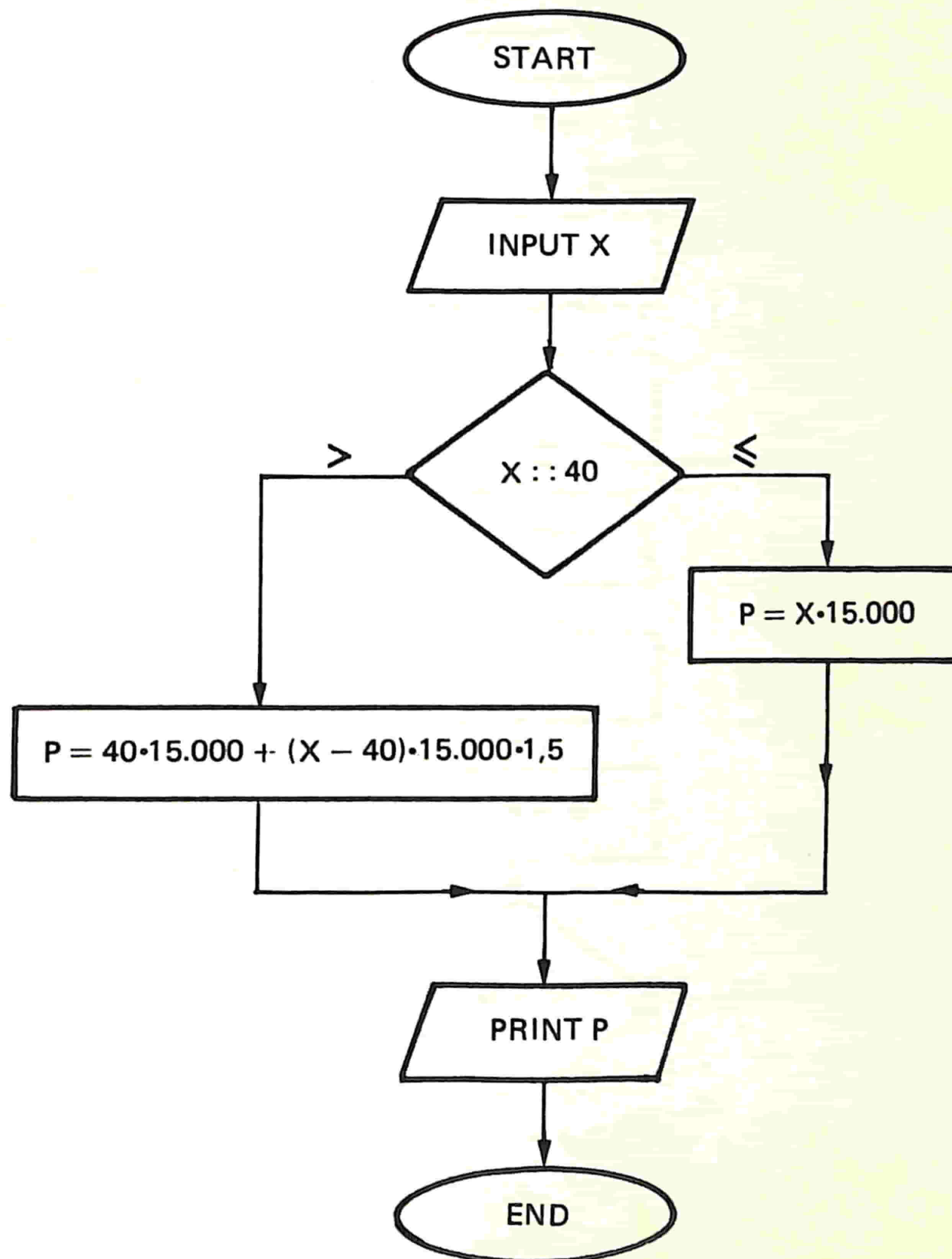


Fig. 9.2 Soluzione esercizio 3.2

ESERCIZIO 3.3

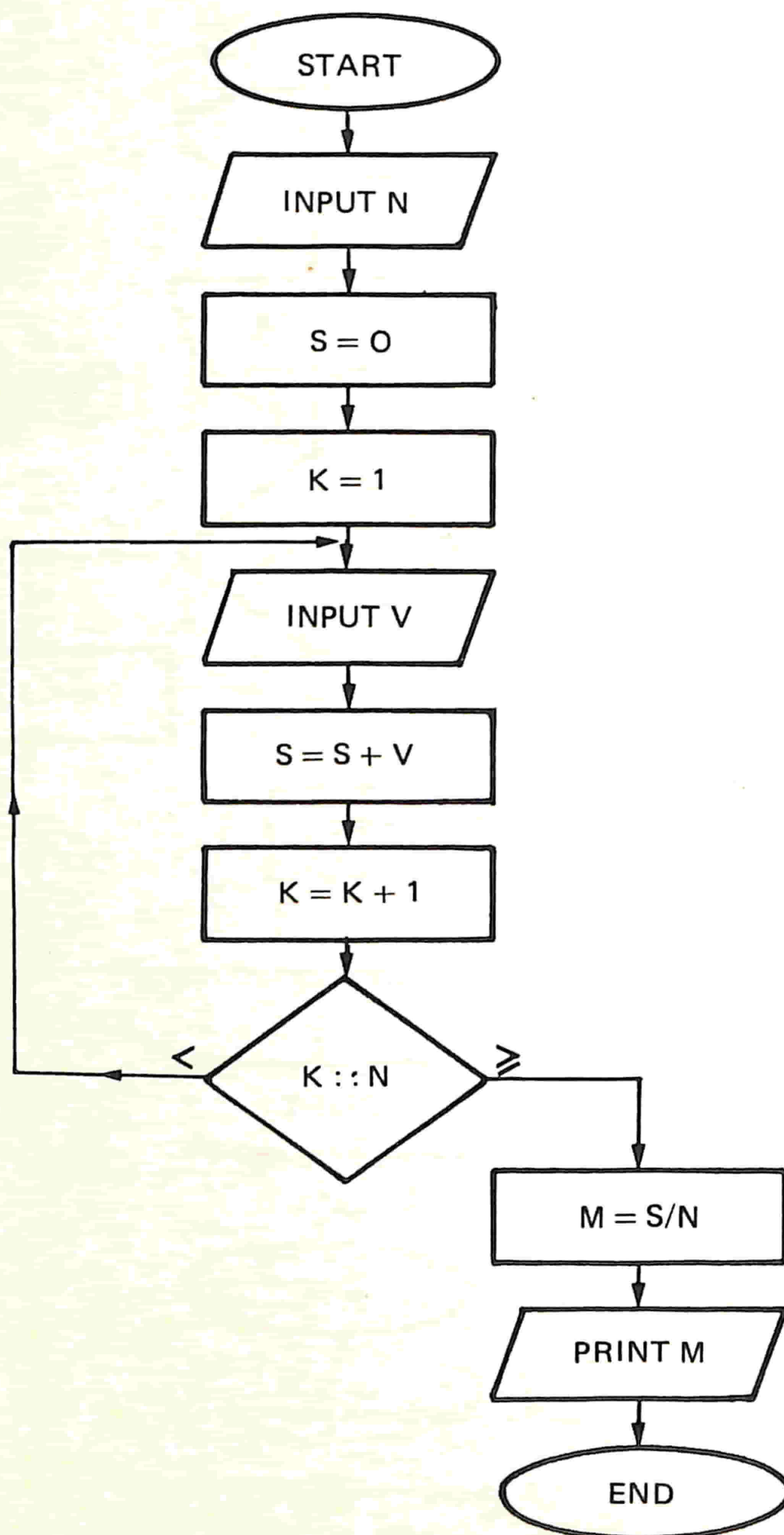


Fig. 9.3 Soluzione esercizio 3.3

ESERCIZIO 4.1

```

100 REM VOLUME DI UN PARALLELEPIPEDO
110 PRINTCHR$(147)
120 PRINT"VOLUME DI UN PARALLELEPIPEDO"
130 PRINT:PRINT
140 INPUT"DIMENSIONI (B,L,H)";B,L,H
150 V=B*L*H
160 PRINT
170 PRINT"IL VOLUME E' V="V
180 END

```

ESERCIZIO 4.2

```

100 REM PAGA SETTIMANALE MURATORE
110 PRINTCHR$(147)
120 PRINT"PAGA SETTIMANALE DI UN MURATORE"
130 PRINT:PRINT
140 INPUT"NUMERO ORE LAVORATE";X
150 IF X>40 THEN 180
160 P=X*15000
170 GOTO190
180 P=40*15000+(X-40)*15000*1.5
190 PRINT:PRINT
200 PRINT"PAGA SETTIMANALE £"P
210 END

```

ESERCIZIO 4.3

```

100 REM MEDIA VOTI DI UNA CLASSE
110 PRINTCHR$(147)
120 PRINT"MEDIA VOTI DI UNA CLASSE"
130 PRINT:PRINT
140 INPUT"NUMERO DEGLI STUDENTI";N
150 S=0
160 FORK=1TO N
170 PRINT
180 PRINT"VOTO STUDENTE N"K;
190 INPUT V
200 S=S+V
210 NEXTK
220 M=S/N
230 PRINT:PRINT
240 PRINT"LA MEDIA DEI VOTI E'"M
250 END

```

ESERCIZIO 7.1

```

100 REM VOLUME DI UN PARALLELEPIPEDO, CON STAMPA
110 PRINTCHR$(147)
120 PRINT"VOLUME DI UN PARALLELEPIPEDO"
130 PRINT:PRINT
140 INPUT"DIMENSIONI (B,L,H)";B,L,H
150 V=B*L*H
160 PRINT
170 PRINT"IL VOLUME E' V="V
172 PRINT:PRINT
174 INPUT"VUOI LA STAMPA DEI RISULTATI (S/N)";A$:IFA$="S"THENGOSUB200
180 END
200 REM SUBROUTINE DI STAMPA
210 OPEN1,4
220 PRINT#1,"VOLUME DI UN PARALLELEPIPEDO"

```

```

230 PRINT#1:PRINT#1
240 PRINT#1,"DIMENSIONI DEL SOLIDO"
250 PRINT#1,"B="B,"L="L,"H="H
260 PRINT#1
270 PRINT#1,"IL VOLUME E' V="V
280 PRINT#1
290 CLOSE1
300 RETURN

```

ESERCIZIO 7.2

```

100 REM PAGA SETTIMANALE MURATORE, CON STAMPA
110 PRINTCHR$(147)
120 PRINT"PAGA SETTIMANALE DI UN MURATORE"
130 PRINT:PRINT
140 INPUT"NUMERO ORE LAVORATE";X
150 IF X>40 THEN 180
160 P=X*15000
170 GOTO190
180 P=40*15000+(X-40)*15000*1.5
190 PRINT:PRINT
200 PRINT"PAGA SETTIMANALE £"P
202 PRINT:PRINT
205 INPUT"VUOI LA STAMPA DEI RISULTATI (S/N)";A$:IFA$="S"THENGOSUB300
210 END
300 REM SUBROUTINE DI STAMPA
310 OPEN4,4
320 PRINT#4,"PAGA SETTIMANALE MURATORE"
330 PRINT#4:PRINT#4
340 PRINT#4,"ORE LAVORATE N."X
350 PRINT#4
360 PRINT#4,"PAGA SETTIMANALE £"P
370 PRINT#4
380 CLOSE4
390 RETURN

```

ESERCIZIO 7.3

```

100 REM MEDIA VOTI DI UNA CLASSE, CON STAMPA
110 PRINTCHR$(147)
120 PRINT"MEDIA VOTI DI UNA CLASSE"
130 PRINT:PRINT
140 INPUT"NUMERO DEGLI STUDENTI";N
145 DIMV(N)
150 S=0
160 FORK=1TON
170 PRINT
180 PRINT"VOTO STUDENTE N"K;
190 INPUT V(K)
200 S=S+V(K)
210 NEXTK
220 M=S/N
230 PRINT:PRINT
240 PRINT"LA MEDIA DEI VOTI E'"M
243 PRINT:PRINT
245 INPUT"VUOI LA STAMPA DEI RISULTATI (S/N)";A$:IFA$="S"THENGOSUB300
250 END
300 REM SUBROUTINE DI STAMPA
310 OPEN1,4
320 PRINT#1,"MEDIA DEI VOTI DI UNA CLASSE"
330 PRINT#1:PRINT#1
340 PRINT#1,"STUDENTE N.", "VOTO"
350 PRINT#1
360 FORK=1TON

```



```
370 PRINT#1,TAB(4)K,TAB(5)V(K)
380 NEXTK
390 PRINT#1
400 PRINT#1,"LA MEDIA DEI VOTI E'"M
410 CLOSE1
420 RETURN
```

Glossario delle sigle e dei termini inglesi usati nel testo

And = e

ASCII = *American Standard Code for Information Interchange* (linguaggio standard americano per l'interscambio di informazioni)

Basic = *Beginners All Purpose Symbolic Instruction Code* (linguaggio simbolico per principianti adatto a tutti gli usi), come aggettivo: di base, elementare

Bit = unità minima di misura binaria dell'informazione

Branch = estensione, ramificazione

Byte = gruppo di 8 bit, unità di misura dell'informazione

Clear (to) = cancellare, chiarire

Close (to) = chiudere

Computer = elaboratore, macchina elaboratrice di dati

Conditional = condizionato

Define (to) = definire

Delete (to) = cancellare

Device = apparecchiatura

Down = giù

End = fine

Expression = formula, espressione

File = insieme di dati

Floating point = virgola mobile, decimale

Floppy disk = disco flessibile, speciale disco per registrazione di dati

Floppy driver = lettore/registratore di floppy disk

Flow-chart = diagramma di flusso, schema logico-grafico di un programma

For = per

Found = trovato
Free = libero
Function = funzione
Get (to) = prendere, avere, ottenere
Go to (to) = andare a
Hardware = insieme delle apparecchiature elettroniche, elettriche e meccaniche di un sistema di elaborazione dati
Home = casa, posizione in alto a sinistra sul video
If/then = se/allora
Input = dati di ingresso, immissione di dati
Integer = intero
Insert (to) = inserire
Jump = salto
Kbyte = 1024 byte, unità di misura dell'informazione
Length = lunghezza
Let (to) = permettere, lasciare, fare in modo che
List (to) = listare, elencare
Load (to) = caricare
Lock = serratura, bloccaggio
Loop = cappio, nodo scorsoio, anello, giro
Looping = iterazione
Mode = modo, maniera
New = nuovo
Next = successivo, seguente
Not = no, non
Off = fuori, spento
On = su, acceso
Open (to) = aprire
Output = dati di uscita
Plot (to) = tracciare diagrammi o disegni
Plotter = apparecchiatura per tracciare disegni
Press (to) = premere
Print (to) = stampare
Process (to) = trattare, elaborare
RAM = *Random Access Memory* (memoria ad accesso casuale)
Random = casuale
Read (to) = leggere
Ready = pronto
Remark = nota, annotazione
Return (to) = ritornare
Reverse = inverso, negativo
Right = destra
ROM = *Read Only Memory* (memoria di sola lettura)
Run (to) = correre

Save (to) = registrare, salvare
Search (to) = cercare
Shift (key) = tasto maiuscole
Syntax = sintassi
Software = programmi
Start (to) = partire
Subroutine = sottoprogramma
To = (preposizione) a
Tape = registratore
Unconditional = incondizionato
Word = parola

Appendici

Sintesi del Basic e delle tecniche di programmazione

In questa appendice sono riportate, in maniera sintetica, le regole del Basic nella versione Commodore V.2 descritte più ampiamente in diverse parti del testo. Vengono inoltre riassunte le principali tecniche di programmazione spiegate nel testo.

Variabili

Sono previsti quattro tipi di variabili:

- *Numeri reali* uno o due caratteri
 A, AB, K, A1, Z3
- *Numeri interi* uno o due caratteri seguiti dal segno %
 A%, K1%, CA%
- *Stringhe* uno o due caratteri seguiti dal segno \$
 A\$, K3\$, CA\$
- *Vettori o matrici* uno o due caratteri seguiti dalle dimensioni
 del vettore o matrice, incluse tra parentesi
 V(10), T\$(2,4), K2%(3,4,7)

Nota: vi sono tre nomi di variabili che non possono essere usati dall'utente: ST, TI, TI\$.

Operatori

— *Operatori aritmetici* (nell'ordine di precedenza in cui vengono eseguiti)

↑ elevamento a potenza
*/ moltiplicazione, divisione
+,- addizione, sottrazione

— *Operatori relazionali*

= uguale
<> diverso
< minore
> maggiore
<= minore o uguale
>= maggiore o uguale

— *Operatori logici* (nell'ordine di precedenza in cui vengono eseguiti)

NOT
AND
OR

Nota: per variare l'ordine di esecuzione delle varie operazioni di una espressione servirsi delle parentesi.

Operazioni con stringhe

Le stringhe possono essere concatenate con il segno

+

PROGRAMMAZIONE

Formato delle linee di programma

XXX COMANDO BASIC ... [RETURN]

Esempio: 1021 PRINT "REAZIONE RA="RA [RETURN]

[RETURN] sta per: batti il tasto contrassegnato RETURN.

Nota: è possibile scrivere più di una istruzione in una linea di programma, separando le varie istruzioni con il segno

Decisione e salti

— *Decisione e salto condizionato*

IF <espressione> <operatore relazionale> <espressione> THEN <espressione o numero di linea>

Esempio: IF X+2 > 0 THEN 500

Se la relazione è vera viene eseguita la parte a destra del THEN (in questo caso un salto alla riga 500), se è falsa viene eseguita la linea di programma successiva

— *Salto incondizionato*

GOTO <numero di linea>

Il programma salta alla linea di cui è indicato il numero

— *Salto multiplo*

ON X GOTO <elenco di numeri di linea>

Esempio: ON X GOTO 100,200,300

Se X vale 1 il programma salta alla prima linea dell'elenco (nell'esempio 100), se vale 2 alla seconda linea dell'elenco, ecc...

Iterazioni

FOR <variabile> = <x> TO <y> STEP <z>

.....

.....

NEXT <variabile>

Esempio: FOR I=1 TO 10 STEP 2

.....

.....

NEXT I

La serie di istruzioni comprese tra il FOR e il NEXT viene eseguita un certo numero di volte, finché il valore della variabile, inizialmente x , aumentando del passo (STEP) z ogni volta che il programma raggiunge il NEXT, non eguaglia il valore y .

Se viene omissa lo STEP $\langle z \rangle$, la variabile viene incrementata di 1 ogni volta che il programma raggiunge il NEXT.

Subroutine

GOSUB $\langle$ numero di linea $\rangle$

Il programma principale passa il controllo alla subroutine che inizia dalla linea il cui numero è specificato. Quando viene incontrato un

RETURN

che contrassegna la fine della subroutine, il programma continua dall'istruzione che segue il GOSUB più recente.

Nota: si può attivare una subroutine anche con l'istruzione di salto multiplo:

ON X GOSUB $\langle$ elenco di numeri di linea $\rangle$

Comandi e funzioni Basic

In questa appendice è riportato l'elenco completo dei comandi e funzioni del Commodore Basic V. 2. Per informazioni più approfondite consultare uno dei volumi indicati nella bibliografia.

COMANDI

CLOSE

Il comando **CLOSE** serve a chiudere i file aperti con l'istruzione **OPEN**.

Esempi

CLOSE 2	viene chiuso il file numero 2
CLOSE	vengono chiusi i file aperti

CLR

Il comando **CLR** cancella tutti i valori delle variabili dalla memoria. Viene eseguito automaticamente quando viene dato il comando **RUN**.

CMD

Il comando **CMD** invia l'output che normalmente andrebbe sullo schermo (con le istruzioni **PRINT** o **LIST**) ad un altro dispositivo o file precedentemente aperto.

Esempio

OPEN 1,4	viene aperto il file 1 sul dispositivo 4 (stampante)
CMD 1	l'output viene trasferito sul file 1
LIST	il listato del programma apparirà sulla stampante.
PRINT#1	l'output viene ritrasferito allo schermo
CLOSE 1	viene chiuso il file 1

CONT

Se l'esecuzione di un programma viene interrotta da un'istruzione **END** o **STOP**, o perché è stato premuto il tasto **RUN/STOP**, il programma può essere fatto ripartire eseguendo il comando **CONT** in modo diretto.

DATA

L'istruzione **DATA** è seguita da un elenco di dati, valori numerici o stringhe, separati da virgole, che vengono letti a seguito del comando **READ**.

I dati vengono letti in successione, ogni volta che si incontra una istruzione **READ**.

Le stringhe devono essere racchiuse tra virgolette se contengono virgole, punti e virgola, spazi, caratteri grafici.

Esempi

```
DATA 1,15,2,18,3,40
DATA 4,"PARTE 1",quattro
```

DEF FN

Oltre alle funzioni già predisposte (come **SIN**, **COS**) è possibile, nel corso di un programma definire specifiche funzioni di una variabile con il comando **DEF FN**.

Il formato dell'istruzione è il seguente

```
DEF FN <nome> <variabile> = <funzione>
```

Il nome può essere un qualsiasi nome di variabile Basic.

Esempio

```
20 DEF FN A1(X) = 4+X ↑ 3*SIN(X)
```

Nel corso del programma possiamo richiamare la funzione scrivendone il nome e inserendo al posto della variabile il valore desiderato.

Esempio

```
20 DEF FN A1(X)=4+3*X+2*X ↑ 2
40 PRINT FN A1(3)
```

viene stampato il valore
31

DIM

L'istruzione DIM serve per dimensionare vettori o matrici.

Esempio

```
110 DIM A(20)
```

END

Questa istruzione interrompe l'esecuzione del programma. Conviene sempre porla alla fine del programma principale per separarlo dalle subroutine.

FOR - NEXT

La coppia di istruzioni FOR e NEXT viene usata per far ripetere un certo gruppo di operazioni per un dato numero di volte. Il formato è il seguente

```
FOR <variabile> = <x> TO <y> STEP <z>
.....
.....
.....
NEXT <variabile>
```

Quando il programma esegue per la prima volta l'istruzione FOR la variabile viene posta uguale a x. Vengono quindi eseguite tutte le istruzioni comprese tra FOR e NEXT. Come viene raggiunta l'istruzione NEXT la variabile viene aumentata del passo (STEP) z e il programma riprende dall'istruzione FOR. L'esecuzione di questo ciclo si interrompe quando la variabile raggiunge il valore y. Se lo STEP non viene indicato è assunto uguale a 1.

Esempio

```
100 FOR I=1 TO 10
110 PRINT I
120 NEXT I
```

Il programma precedente stamperà sul video una lista di numeri da 1 a 10.

GET

L'istruzione GET consente di trasferire dati dalla tastiera, un carattere alla volta. GET è seguita da un nome di variabile, in genere una variabile stringa. Quando viene eseguita GET, il carattere che viene battuto è assegnato alla variabile. Se non viene battuto alcun carattere alla variabile viene assegnato un valore nullo.

Esempio

```
10 GET A$:IFA$=""THEN10
```

il programma non prosegue finché non viene premuto un tasto.

GET#

Questa istruzione permette di prendere, da un dispositivo o file precedentemente aperto, un carattere alla volta.

Esempio

```
GET#1,A$
```

immette nella memoria del computer il primo carattere del file 1.

GOSUB - RETURN

L'istruzione GOSUB è simile all'istruzione GOTO, fa infatti continuare l'esecuzione del programma dalla riga indicata dopo il GOSUB. Quando viene incontrata l'istruzione RETURN il programma ritorna alla riga immediatamente successiva a quella contenente il GOSUB.

Esempio

```
10 GOSUB 200
```

```
.....
```

```
200 .....
```

```
.....
```

```
250 RETURN
```

GOTO

L'istruzione GOTO, seguita da un numero di linea, fa proseguire il programma dal numero di linea specificato.

Esempio

```
50 GOTO400
```

IF - THEN

L'istruzione IF consente al computer di valutare una situazione ed in base al risultato scegliere tra due azioni da compiere.

Il formato dell'istruzione è

```
IF <condizione> THEN <azione>
```

La condizione in genere coinvolge uno degli operatori relazionali o logici.

Esempi

```
150 IF D>0 THEN 500
```

```
430 IF A$="S" THEN CLR:GOTO 130
```

INPUT

L'istruzione INPUT permette di ricevere dati dall'utente nel corso di un programma. Il formato è il seguente

```
INPUT "richiesta";<variabile>
```

È possibile inserire una richiesta che chiarisca cosa si attende dall'utente. Il dato inserito dall'utente tramite il tasto RETURN viene assegnato alla variabile. Se vengono richiesti più dati, questi vanno separati da virgole quando vengono battuti.

Esempi

```
120 INPUT"QUANTI CARICHI";N
```

```
250 INPUT"COORDINATE DI A (X,Y)";XA,YA
```

INPUT#

L'istruzione INPUT# è simile a INPUT, ma serve per prendere dati da un dispositivo o file precedentemente aperto (OPEN).

Esempio

```
10 INPUT#1,A
```

Verrà preso un dato numerico dal file 1.

LET

L'istruzione **LET** serve per assegnare un valore a una variabile. L'uso di **LET** è facoltativo, in questo libro è stato generalmente omesso.

Esempio

```
10 LET A=B+4  
10 A=B+4
```

Le due linee sono equivalenti, viene assegnato alla variabile **A** il valore della variabile **B** più 4.

LIST

Il comando **LIST** permette di ottenere il listato del programma presente in memoria. È possibile ottenere la visualizzazione dell'intero programma o soltanto di alcune linee.

Esempi

LIST	viene mostrato l'intero programma
LIST 100	viene mostrata solo la riga 100
LIST 100-200	vengono mostrate le linee da 100 a 200
LIST -100	vengono mostrate le linee dall'inizio fino a 100
LIST 100-	vengono mostrate le righe da 100 fino alla fine

LOAD

Il comando **LOAD** serve per caricare in memoria un programma presente su disco o su cassetta.

Per caricare un programma da cassetta scrivere

LOAD"nome del programma"

Per caricare un programma da disco scrivere

LOAD"nome del programma",8

NEW

Il comando **NEW** cancella il programma presente in memoria e azzerava tutte le variabili.

ON

L'istruzione **ON**, nelle forme

ON <espressione> **GOTO** <elenco di numeri di linea>
ON <espressione> **GOSUB** <elenco di numeri di linea>

Permette di far proseguire il programma da diversi punti, in relazione al valore di «espressione». Se tale valore è 1, viene eseguita la prima riga dell'elenco, se 2 la seconda, e così via. Se il valore dell'espressione è zero o maggiore del numero di linee dell'elenco, viene eseguita l'istruzione che segue ON.

Esempio

```
100 INPUT A
110 ON A GOTO 200,300,400
```

OPEN

L'istruzione OPEN permette al computer di collegarsi con i diversi dispositivi del sistema (registratore, disco, stampante, schermo) aprendo un file.

Nella forma

OPEN A,B

A è il numero del file, B il numero che indica il dispositivo. I numeri dei dispositivi sono

0	schermo
1	cassetta
4	(o 5) stampante
8	disco

Esempio

```
100 OPEN 1,4
```

apre il file 1 sulla stampante.

Dopo che un file su un dispositivo è stato aperto su di esso si possono scrivere (PRINT#) o estrarre dei dati (INPUT#, GET#).

Per maggiori dettagli vedere il *Manuale d'uso* del Commodore 64, in bibliografia.

POKE

L'istruzione POKE consente di inserire un determinato valore in una qualsiasi posizione di memoria della RAM.

Il formato dell'istruzione è il seguente

POKE N,M

dove N è un numero intero tra (per il Commodore 64) 0 e 65536 e rappresenta l'indirizzo della posizione di memoria che vogliamo modificare; M invece può essere compreso tra 0 e 255 e rappresenta ciò che vogliamo mettere nella memoria.

L'istruzione POKE è molto usata, insieme alla funzione PEEK, in programmi che gestiscono i colori, il suono, la grafica.

Per maggiori informazioni vedere il *Manuale d'uso* del Commodore 64, in bibliografia.

PRINT

L'istruzione PRINT permette di far apparire sul video delle informazioni.

Se le informazioni da stampare sono stringhe, esse vanno racchiuse tra virgolette. Quando si vogliono stampare più informazioni con la stessa istruzione PRINT, occorre separarle con virgole (se le vogliamo spaziare di una certa quantità) o con punto e virgola (se vogliamo che siano stampate di seguito).

Per controllare la posizione della stampa sul video vedere anche le funzioni TAB, SPC e POS.

Esempi

Il seguente programma

```
10 X=4
20 PRINT X+4;X-4,X*4;X ↑ 4
```

dà come output su video

```
8 0      16 256
```

Altro programma:

```
10 INPUT"UN NUMERO";X
20 PRINTX"AL QUADRATO VALE"X ↑ 2"E";
30 PRINTX"AL CUBO VALE"X ↑ 3
```

rispondendo 4 alla richiesta di INPUT, sul video compare la scritta.

```
4 AL QUADRATO VALE 16 E 4 AL CUBO VALE 64
```

PRINT#

È molto simile all'istruzione PRINT, serve per scrivere su un dispositivo o file precedentemente aperto.

Il formato è

PRINT#N,<lista di espressioni>

N è il numero del file su cui va stampata la lista di espressioni.

READ

Con questa istruzione è possibile assegnare a delle variabili i valori elencati in una istruzione DATA.

Per esempio, se un programma contiene la seguente istruzione

```
1000 DATA1,GENNAIO,2,FEBBRAIO
```

la riga

```
200 READ A,A$
```

assegnerà alla variabile A il valore 1, e alla variabile A\$ la stringa GENNAIO.

REM

L'istruzione REM non viene eseguita dal computer. Permette di aggiungere al programma delle note al fine di renderlo più chiaro e leggibile in listato.

Esempio

```
1000 REM SUBROUTINE DI CALCOLO DELLE REAZIONI
```

RESTORE

L'istruzione RESTORE viene usata solo in combinazione con le istruzioni READ e DATA. L'istruzione DATA contiene una lista di valori, l'istruzione READ associa i valori delle DATA, in ordine, a variabili. RESTORE (che va scritto da solo su una riga) fa ripartire la lettura dei successivi dati dal primo della lista contenuta nell'istruzione DATA.

RUN

Il comando RUN fa iniziare l'esecuzione del programma presente in memoria, cominciando dal numero di riga più basso.

Nella forma

```
RUN 100
```

fa iniziare l'esecuzione del programma a partire dalla riga 100.

SAVE

L'istruzione **SAVE** permette di registrare su una memoria di massa il programma presente in memoria.

Esempi

Per registrare un programma su cassette

SAVE"nome del programma"

Per registrare un programma su disco

SAVE"nome del programma"

Il nome del programma non può essere più lungo di 16 caratteri.

STOP

L'istruzione **STOP** inserita in un programma fa interrompere l'esecuzione del programma stesso e riporta il computer in modo diretto. È possibile riprendere l'esecuzione del programma con l'istruzione **CONT**.

SYS

Dopo che una subroutine in linguaggio macchina è stata posta in memoria a partire dalla locazione **A**, l'istruzione

SYS A

trasferisce il controllo del programma alla subroutine che inizia nella locazione **A**.

A è un valore o un'espressione numerica il cui risultato è compreso tra 0 e 65535 (per il Commodore 64).

VERIFY

L'istruzione **VERIFY** permette di verificare il programma presente in memoria con un programma registrato su disco o cassetta. Si usa appunto per verificare se l'operazione di registrazione (**SAVE**) è avvenuta correttamente.

Esempi

Per verificare un programma registrato su cassetta

VERIFY"nome del programma"

Per verificare un programma registrato su disco

VERIFY"nome del programma",8

WAIT

L'istruzione WAIT serve per sospendere l'esecuzione di un programma Basic finché i contenuti di una locazione di memoria cambiano in una maniera specifica.

Il formato è il seguente

WAITX,Y,Z

Dove X è la locazione di memoria, Y e Z due variabili.

Per maggiori informazioni vedere la *Guida per il programmatore*, in bibliografia.

FUNZIONI

Avvertenza: gli argomenti delle funzioni a seconda del tipo, sono così indicati:

X e Y indicano una qualsiasi espressione numerica

I e J indicano espressioni intere

X\$ e Y\$ indicano stringhe

ABS(X)

Dà il valore assoluto dell'espressione X.

Esempio

PRINTABS(2-3)

1

ASC(X\$)

Dà il codice ASCII del primo carattere della stringa X\$ (si veda l'Appendice C).

Esempio

PRINTASC("A")

65

ATN(X)

Dà il valore dell'angolo, misurato in radianti, la cui tangente è X.

Esempio

```
PRINTATN(1)  
.7953981636
```

CHR\$(I)

È l'inverso delle funzioni ASC e dà una stringa alfanumerica il cui codice ASCII è I.

Esempio

```
PRINTCHR$(65)  
A
```

COS(X)

Dà il valore del coseno di X con l'angolo X misurato in radianti.

Esempio

```
PRINTCOS( $\pi$ )  
-1
```

EXP(X)

Dà il valore della costante e (2.71828183) elevata alla potenza X.

Esempio

```
PRINTEXP(1)  
2.71828183
```

FRE(X)

Dà il numero di byte ancora disponibili in memoria; X può essere un numero qualsiasi.

INT(X)

Dà il valore del più grande numero intero minore o uguale a X.

Esempio

```
PRINTINT(2.34),INT(-2.34)
```

```
2          -3
```

LEFT\$(X\$,I)

Dà una stringa contenente gli I caratteri più a sinistra di X\$.

Esempio

```
PRINTLEFT$("PATRIZIO",3)  
PAT
```

```
LEN(X$)
```

Dà il numero dei caratteri (compresi gli spazi e altri simboli) di cui è composta la stringa X\$.

Esempio

```
PRINTLEN("PADOVA")  
6
```

```
LOG(X)
```

Dà il valore del logaritmo naturale (in base e) di X.

Esempio

```
PRINTLOG(EXP(1))  
1
```

```
MID$(X$,I,J)
```

Estrae dalla stringa X\$ J caratteri partendo dall'I-esimo carattere.

Esempio

```
PRINTMID$("ITALIA",3,3)  
ALI
```

```
PEEK(I)
```

Dà il contenuto di memoria della posizione i-esima, riportando un numero compreso tra 0 e 255. Viene usata spesso unitamente all'istruzione POKE.

```
POS(X)
```

Dà il numero della colonna dove è posizionato il cursore; X può essere un qualsiasi numero.

```
RIGHT$(X$,I)
```

Estrae dalla stringa X\$ gli I caratteri più a destra.

Esempio

```
PRINTRIGHT$("PATRIZIO",3)  
ZIO
```

RND(X)

Dà un numero casuale compreso tra 0 e 1. X può essere positivo, zero o negativo, ciò dà luogo a diverse serie di numeri casuali. Per maggiori informazioni vedere il *Manuale d'uso* del Commodore 64, in bibliografia.

SGN(X)

Dà il segno del numero X. I risultati possono essere +1, per X positivo, 0 per X=0, -1 per X negativo.

Esempio

```
PRINTSGN(-4)
```

```
-1
```

SIN(X)

Dà il valore del seno di X con X espresso in radianti.

Esempio

```
PRINTSIN(π/2)
```

```
1
```

SPC(I)

Se posta in una istruzione PRINT, fa stampare I spazi bianchi.

Esempio

```
PRINTSPC(8)"OTTO"  
      OTTO
```

SQR(X)

Dà il valore della radice quadrata di X, per X non negativo.

Esempio

```
PRINTSQR(2)
```

```
1.41421356
```

STATUS

Il valore della funzione STATUS permette di controllare se, dopo operazioni di input/output, si sono verificate certe condizioni. Per maggiori informazioni si veda la *Guida di riferimento per il programmatore*, in bibliografia.

PRINTSPC(8)"OTTO"
8 SPAZI OTTO

STR\$(X)

Dà una stringa corrispondente alla versione stampata (PRINT) di X.

Esempio

```
PRINTSTR$(3.148)
```

```
3.148
```

TAB(I)

Se usata in una istruzione PRINT, fa stampare il successivo elemento alla colonna X.

Esempio

```
PRINT1TAB(5)2
```

```
1      2
```

TAN(X)

Dà il valore della tangente di X, con X espresso in radianti.

Esempio

```
PRINTTAN( $\pi/4$ )
```

```
.999999999
```

TIME

All'accensione del computer parte un orologio che produce e conta "Jiffies", sessantesimi di secondo.

Il valore della funzione TIME (abbreviazione TI) in un certo istante è il numero dei sessantesimi di secondo registrati dall'orologio interno.

Esempio

```
PRINT TIME/60
```

Fornisce il numero di secondi trascorsi dall'accensione del computer.

TIME\$

La funzione TIME\$ (abbreviazione TI\$) fornisce una stringa nella forma "oommss", dove oo rappresentano le ore, mm i minuti e ss i secondi trascorsi dall'accensione del computer.

È possibile predisporre la funzione TIME\$ con un qualsiasi orario semplicemente ponendola uguale all'orario desiderato scritto nella forma "oommss".

Esempio

Per fissare la funzione `TIME$` alle 10 e trenta del mattino, basta scrivere:

```
TIME$="103000"
```

`USR(X)`

Questa funzione permette di passare da un programma Basic a un programma in linguaggio macchina. Il parametro `X` viene passato al programma in linguaggio macchina che fornirà il risultato dell'elaborazione al programma Basic.

Per maggiori informazioni sulla funzione `USR` e sul linguaggio macchina si veda la *Guida di riferimento per il programmatore*, in bibliografia.

`VAL(X$)`

La funzione `VAL` converte la stringa `X$` in un numero, se possibile. È la funzione inversa di `STR$`.

La stringa `X$` viene esaminata a partire da sinistra, se il primo carattere incontrato è diverso da `+`, `-`, `$` o una cifra, `VAL(X$)=0`.

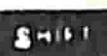
Esempio

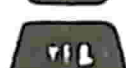
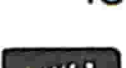
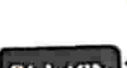
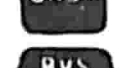
```
PRINTVAL("123A")  
123
```


Codici ASCII e CHR\$

Nella tabella che segue sono elencati i caratteri che vengono stampati sul video o eseguiti, se si tratta di comandi, quando si esegue l'istruzione

PRINT CHR\$(X)

STAMPA	CHR\$	STAMPA	CHR\$	STAMPA	CHR\$	STAMPA	CHR\$
	0		17	"	34	3	51
	1		18	#	35	4	52
	2		19	\$	36	5	53
	3		20	%	37	6	54
	4		21	&	38	7	55
	5		22	.	39	8	56
	6		23	(	40	9	57
	7		24	)	41	:	58
disattiva  	8		25	*	42	;	59
attiva  	9		26	+	43	<	60
	10		27	,	44	=	61
	11		28	-	45	>	62
	12		29	.	46	?	63
	13		30	/	47	@	64
	14		31	0	48	A	65
	15		32	1	49	B	66
	16	!	33	2	50	C	67

STAMPA	CHR\$	STAMPA	CHR\$	STAMPA	CHR\$	STAMPA	CHR\$
D	68		97		126		155
E	69		98		127		156
F	70		99		128		157
G	71		100		129		158
H	72		101		130		159
I	73		102		131		160
J	74		103		132		161
K	75		104	f1	133		162
L	76		105	f3	134		163
M	77		106	f5	135		164
N	78		107	f7	136		165
O	79		108	f2	137		166
P	80		109	f4	138		167
Q	81		110	f6	139		168
R	82		111	f8	140		169
S	83		112	 	141		170
T	84		113		142		171
U	85		114		143		172
V	86		115		144		173
W	87		116		145		174
X	88		117		146		175
Y	89		118		147		176
Z	90		119		148		177
[	91		120		149		178
£	92		121		150		179
]	93		122		151		180
↑	94		123		152		181
←	95		124		153		182
	96		125		154		183
	184		186		188		190
	185		187		189		191

CODICI
CODICI
CODICI

192-223
224-254
255

COME PER
COME PER
COME PER

96-127
160-190
126

Abbreviazioni per le parole chiavi del Basic

Co- mando	Abbrevia- zione	Simbolo sul video	Co- mando	Abbrevia- zione	Simbolo sul video
ABS	A SHIFT B	A	FOR	F SHIFT O	F
AND	A SHIFT N	A	FRE	F SHIFT R	F
ASC	A SHIFT S	A	GET	G SHIFT E	G
ATN	A SHIFT T	A	GET#	NESSUNO	GET#
CHR\$	C SHIFT H	C	GOSUB	GO SHIFT S	GO
CLOSE	CL SHIFT O	CL	GOTO	G SHIFT O	G
CLR	C SHIFT L	C	IF	NESSUNO	IF
CMD	C SHIFT M	C	INPUT	NESSUNO	INPUT
CONT	C SHIFT O	C	INPUT#	SHIFT N	I
COS	NESSUNO	COS	INT	NESSUNO	INT
DATA	D SHIFT A	D	LEFT\$	LE SHIFT F	LE
DEF	D SHIFT E	D	LEN	NESSUNO	LEN
DIM	D SHIFT I	D	LET	L SHIFT E	L
END	E SHIFT N	E	LIST	L SHIFT I	L
EXP	E SHIFT X	E	LOAD	L SHIFT O	L
FN	NESSUNO	FN	LOG	NESSUNO	LOG

Co- mando	Abbrevia- zione	Simbolo sul video	Co- mando	Abbrevia- zione	Simbolo sul video
MID\$	M SHIFT I	M 	SAVE	S SHIFT A	S 
NEW	NESSUNO	NEW	SGN	S SHIFT G	S 
NEXT	N SHIFT E	N 	SIN	S SHIFT I	S 
NOT	N SHIFT O	N 	SPC(	S SHIFT P	S 
ON	NESSUNO	ON	SQR	S SHIFT Q	S 
OPEN	O SHIFT P	O 	STATUS	ST	ST
OR	NESSUNO	OR	STEP	ST SHIFT E	ST 
PEEK	P SHIFT E	P 	STOP	S SHIFT T	S 
POKE	P SHIFT O	P 	STR\$	ST SHIFT R	ST 
POS	NESSUNO	POS	SYS	S SHIFT Y	S 
PRINT	?	?	TAB(	T SHIFT A	T 
PRINT#	P SHIFT R	P 	TAN	NESSUNO	TAN
READ	R SHIFT E	R 	THEN	T SHIFT H	T 
REM	NESSUNO	REM	TIME	TI	TI
RESTORE	RE SHIFT S	RE 	TIME\$	TI\$	TI\$
RETURN	RE SHIFT T	RE 	USR	U SHIFT S	U 
RIGHT\$	R SHIFT I	R 	VAL	V SHIFT A	V 
RND	R SHIFT N	R 	VERIFY	V SHIFT E	V 
RUN	R SHIFT U	R 	WAIT	W SHIFT A	W 

Messaggi di errore

Questa Appendice contiene un elenco completo dei messaggi di errore generati dal Commodore 64, con una descrizione delle relative cause.

BAD DATA (Dati errati). La stringa di dati è stata ricevuta da un file aperto ma il programma aspettava dati numerici.

BAD SUBSCRIPT (Indice errato). Il programma stava cercando di fare riferimento ad un elemento di una matrice il cui numero è al di fuori del campo specificato nell'istruzione **DIM**.

CAN'T CONTINUE (Non posso continuare). Il comando **CONT** non funziona sia perché il programma non è mai stato eseguito (**RUN**) oppure perché c'è stato un errore oppure perché è stata corretta una riga.

DEVICE NOT PRESENT (Dispositivo non presente). Il richiesto dispositivo di I/O non era disponibile per un'istruzione **OPEN**, **CLOSE**, **CMD**, **PRINT#**, **INPUT#** o **GET#**.

DIVISION BY ZERO (Divisione per zero). La divisione per zero è una incongruenza matematica e non è ammessa.

EXTRA IGNORED (Extra ignorati). Sono stati battuti troppi elementi di dati in risposta ad un'istruzione **INPUT**. Sono stati accertati soltanto i primi elementi.

FILE NOT FOUND (File non trovato). Se si stava cercando un file su nastro e si è trovato un marcatore di fine nastro. Se si cercava su disco, non esiste file con quel nome.

FILE NOT OPEN (File non aperto). Il file specificato in un'istruzione **CLOSE**, **CMD**, **PRINT#**, **INPUT#** o **GET#** deve essere per prima cosa aperto (**OPEN**).

FILE OPEN (File aperto). È stato compiuto un tentativo di aprire un file già aperto.

FORMULA TOO COMPLEX (Formula troppo complessa). L'espressione stringa che viene calcolata deve essere suddivisa in almeno due parti perché il sistema la possa elaborare.

ILLEGAL DIRECT (Illecito nel modo diretto). L'istruzione **INPUT** può essere usata soltanto nell'ambito di un programma e non nel modo diretto.

ILLEGAL QUANTITY (Quantità illecita). Un numero usato come argomento di una funzione o istruzione è fuori dal campo ammesso.

LOAD (Caricamento). C'è un problema con il programma sul nastro.

NEXT WITHOUT FOR (**NEXT** senza **FOR**). Ciò è provocato dalla nidificazione di iterazioni scorrette o dalla presenza di un nome variabile in un'istruzione **NEXT** che non corrisponde a quella nell'istruzione **FOR**.

NOT INPUT FILE (Non è un file di input). È stato compiuto un tentativo di **INPUT** o di **GET** su un file che era stato specificato solo per l'output.

NOT OUTPUT FILE (Non è un file di output). È stato compiuto un tentativo di **PRINT** dati su un file che era stato specificato solo per l'input.

OUT OF DATA (non più dati). È stata eseguita un'istruzione **READ** ma non c'erano più dati da leggere (**READ**) in un'istruzione **DATA**.

OUT OF MEMORY (Mancanza di memoria). Non c'è altra RAM disponibile per il programma o le variabili. Ciò può anche verificarsi quando sono stati nidificate troppe iterazioni **FOR** o quando sono in atto troppi **GOSUB**.

OVERFLOW (Superamento di capacità). Il risultato di un calcolo è maggiore del numero massimo ammesso che è $1.70141884E+38$.

REDIM'D ARRAY (Matrici ridimensionate). Una matrice può essere dimensionata (**DIM**) solo una volta. Se viene usata una variabile matrice prima che quella matrice sia dimensionata (**DIM**), viene eseguita un'operazione **DIM** automatica su quella matrice portando il numero di elementi a 10 e qualsiasi successiva **DIM** provoca questo errore.

REDO FROM START (Ripetere dall'inizio). Sono stati battuti dati di alfabetici durante un'istruzione **INPUT** quando si aspettavano dati numerici. Basta ribattere l'entrata corretta e il programma continua da solo.

RETURN WITHOUT GOSUB (**RETURN** senza **GOSUB**). Si è incontrata un'istruzione **RETURN** e non è stato emesso un comando **GOSUB**.

STRING TOO LONG (Stringa troppo lunga). Una stringa può contenere fino a 255 caratteri.

?SYNTAX ERROR (Errore di sintassi). Un'istruzione non riconosciuta dal Commodore 64. (Parentesi mancanti o in più, parole chiave scritte erroneamente ecc.).

TYPE MISMATCH (Errato abbinamento di tipo). Questo errore si verifica quando viene usato un numero in luogo di una stringa o viceversa.

UNDEF'D FUNCTION (Funzione non definita). È stato fatto riferimento ad una funzione definita dall'utente che non è mai stata definita usando l'istruzione **DEF FN**.

UNDEF'D STATEMENT (Istruzione non definita). È stato compiuto un tentativo di **GOTO** o **GOSUB** o di **RUN** relativamente a un numero di riga che non esiste.

VERIFY (Verifica). Il programma su nastro o su disco non corrisponde al programma correntemente in memoria.

Bibliografia

Vengono qui elencati alcuni testi, suddivisi per argomenti, con cui è possibile ampliare ed approfondire le conoscenze acquisite con il presente libro.

Programmazione e Basic

T. Dwyer e M. Crithfield, *Il Basic e il personal computer, uno: introduzione*, Franco Muzzio Editore, Padova, 1982, £ 12.000.

Semplice introduzione al Basic: applicazioni per diversi personal computer.

H.D. Peckham, *Imparate il Basic con il PET/CBM*, Franco Muzzio Editore, Padova, 1983, £ 12.000.

Introduzione al Commodore Basic, impostato secondo la tecnica "impara da solo", richiede la possibilità di usare un computer della Commodore; presenta moltissimi esercizi di graduale difficoltà.

B.S. Gottfried, *Programmare in Basic*, collana Schaum, Etas libri, Milano.

Corso di programmazione in Basic di livello medio-alto, richiede discrete conoscenze di matematica.

Commodore 64

J. e D. Willis, *Come usare il Commodore 64*, Franco Muzzio Editore, Padova, 1984, £ 18.000.

Semplice introduzione all'uso del Commodore 64.

Commodore 64, Manuale d'uso, Commodore Italiana, 1982.
Manuale fornito con la macchina.

F. Mosher e D. Schneider, *Handbook of Basic of the Commodore 64*, Robert J. Brady Co., Bowie, Maryland 20715, 1984.

Manuale chiaro e completo su tutti gli usi del Commodore 64, richiede conoscenze di inglese e di almeno uno dei precedenti libri sul Basic.

Commodore 64, Guida di riferimento per il programmatore, Commodore Italiana, 1983.

Manuale tecnico del Commodore 64, richiede discrete conoscenze di programmazione e di hardware.

Costruzioni

S. Di Pasquale e altri, *Costruzioni*, Le Monnier, Firenze.

D. Fois, *Corso di costruzioni*, Calderini, Bologna.

Estimo

I. Michieli, *Estimo generale*, Edagricole, Bologna.

Topografia

C. Bonfigli, L. Solaini, *Corso di topografia*, Le Monnier, Firenze.

Program sheet

Programma principale

100

Subroutine

1000



*Finito di stampare nel mese di aprile 1985
presso Lito Velox - Trento
Printed in Italy*

Siete interessati ai personal computer?

Su questo argomento, nella collana "il piacere del computer" sono stati pubblicati i seguenti titoli.

Troverete questi libri nelle principali librerie, oppure potete ordinarli direttamente alla casa editrice compilando la cartolina qui allegata

32 programmi con il PET

Il volume contiene 32 programmi in Basic, completamente documentati con listati, esecuzioni di prova, istruzioni per far girare il programma, suggerimenti per variazioni, ecc. 240 pagine, 16.000 lire, sigla PDC 1

Intervista sul personal computer, hardware

Seicento domande e risposte sul mondo dei personal computer. Questo primo volume, dedicato all'hardware, contiene una introduzione ai computer in generale e ai microprocessori in particolare. 240 pagine, 16.000 lire, sigla PDC 2

32 programmi con l'Apple

Il volume contiene 32 programmi in Basic, completamente documentati con listati, esecuzioni di prova, istruzioni per far girare il programma, suggerimenti per variazioni, ecc. 248 pagine, 16.000 lire, sigla PDC 3

Microsoft Basic

Un breve manuale di introduzione al Microsoft Basic, una evoluzione e specializzazione del Basic originale, che è di fatto lo standard del Basic per microcomputer 150 pagine, 12.000 lire, sigla PDC 4

Pascal

Scritto per coloro che non hanno esperienza di calcolatori o programmazione. Gli argomenti sono organizzati in modo che il lettore possa iniziare a programmare fin dall'inizio. 200 pagine, 12.000 lire, sigla PDC 5

32 programmi con il TRS-80

Trentadue programmi, completamente documentati, pronti per essere eseguiti su un TRS-80 modello I. Il volume comprende i listati, le spiegazioni e i consigli per ulteriori progetti. 240 pagine, 12.000 lire, sigla PDC 6

Intervista sul personal computer, software

In questo secondo volume, dedicato al software, altre centinaia di domande e risposte sul mondo dei personal computer. Contiene una introduzione alla programmazione, ai linguaggi assembler e a quelli evoluti. 200 pagine, 12.000 lire, sigla PDC 7

Imparate il Basic con il PET/CBM

Questo libro è stato progettato per essere utile a chiunque desideri imparare a programmare in Basic avendo a disposizione un PET. 250 pagine, 16.000 lire, sigla PDC 8

Il personal computer come professione

Il personal computer vi offre mille opportunità di lavoro: scrivere articoli o libri su computer; intraprendere un'attività di consulenza o organizzativa: il libro contiene numerosi consigli per la migliore riuscita. 112 pagine, 9.000 lire, sigla PDC 9

Te ne intendi di computer?

Lo scopo di questo libro è di aumentare il livello della vostra comprensione dei computer. Sapere cosa possono e cosa non possono fare, qual è il loro ruolo nella società, quali problemi creano. 144 pagine, 12.000 lire, sigla PDC 10

Il Basic e il personal computer, uno: introduzione

Il libro, scritto in tono amichevole e informale, non richiede precedenti esperienze con i computer. Vi è compresa una presentazione del Basic con decine di esempi dettagliati, che fanno diventare realtà le vostre idee. 190 pagine, 14.000 lire, sigla PDC 11

Imparate il linguaggio dell'Apple

Questo libro vi mette in grado di comprendere il linguaggio macchina dell'Apple partendo dagli esempi più semplici ed arrivando all'uso del miniassembler. 340 pagine, 15.000 lire, sigla PDC 12

Il Basic e il personal computer, due: applicazioni

Numerosi programmi in Basic adatti ad ogni personal computer. L'uso delle variabili alfanumeriche, gli algoritmi di sort, i giochi, l'arte con il computer sono esempi di programmi illustrati nel volume. 200 pagine, 14.000 lire, sigla PDC 13

Il manuale del CP/M

Questo volume è un'introduzione breve ma efficace alla tecnica e alla filosofia del CP/M. 120 pagine, 9.500 lire, sigla PDC 14

A scuola con il PET/CBM

Trenta programmi didattici per ogni tipo di PET/CBM matematica, fisica, statistica, ecc. 160 pagine, 13.000 lire, sigla PDC 15

Il manuale dell'Atom

L'edizione italiana del manuale originale di questo personal computer, ora importato in Italia. 300 pagine, 18.500 lire, sigla PDC 16

Il libro del Commodore VIC 20

Questo libro è inteso come supplemento al manuale della macchina, e presenta numerose caratteristiche del VIC. 156 pagine, 12.000 lire, sigla PDC 17.

Troverete questi libri nelle principali librerie, oppure potete ordinarli direttamente alla casa editrice compilando la cartolina qui allegata

SUL RETRO
ALTRE
INFORMAZIONI

Ho trovato questa cartolina nel libro
acquistato in

In questo spazio potete scrivere quello che pensate di questo libro.

- ☐ Desidero ricevere il vostro più recente catalogo
☐ Desidero ricevere i volumi qui indicati:

sigla	titolo	prezzo
totale		

- ☐ Pagherò al postino l'importo totale indicato + L. 1.000 quale contributo alle spese di spedizione
☐ Allego assegno o vaglia n. per l'importo totale indicato

Il debug nei personal computer

Il "debug", in informatica, è la messa a punto dei programmi, la ricerca e la correzione finale degli errori. Si tratta di una operazione di grande importanza, che si può effettuare in vari modi, tutti trattati in questo libro. 111 pagine, 15.000 lire, sigla PDC 18

Programmazione in Basic per l'uomo d'affari

Questo libro è per gli uomini d'affari che vogliono imparare ad usare il calcolatore e parte dalla convinzione che sia facile per un uomo d'affari imparare a programmare la gestione di una azienda. 256 pagine, 19.000 lire, sigla PDC 19

Imparate il Basic con lo ZX-81

È il microcomputer più venduto nel mondo. Ma necessita di software e documentazione: questo libro contiene 37 programmi che illustrano tutte le caratteristiche e le capacità della macchina. 132 pagine, 13.000 lire, sigla PDC 20

Dal Basic al Pascal

Questa guida rende facile passare dalla propria conoscenza del Basic ad una perfetta padronanza del Pascal, un linguaggio a 2 livelli che richiede uno spazio inferiore di memoria nel vostro computer. 88 pagine, 10.000 lire, sigla PDC 21

Imparate il Basic con il Texas TI 99/4A

Il Texas TI 99/4A vi può aiutare nell'apprendimento delle lingue, della matematica, tenere la contabilità della vostra famiglia. Basta conoscere il linguaggio giusto: questo libro ve lo insegna. 264 pagine, 22.000 lire, sigla PDC 22

A scuola con il Texas TI 99/4A

Scritto da un insegnante, questo è un libro di software per la scuola. Gli argomenti sono presentati in ordine di difficoltà crescente e sono tutti trattati negli studi medi inferiori e superiori. 208 pagine, 18.000 lire, sigla PDC 23

Come usare il Commodore 64

Il lettore troverà nel libro non solo una facile guida per imparare il Basic del Commodore 64, ma anche una fonte completa di informazioni sull'installazione, gli accessori, gli user's group ed il software disponibile. 128 pagine, 18.000 lire, sigla PDC 24

Imparate il Basic con lo Spectrum

Un libro per chi non ha esperienza di computer, e per chi è già esperto ma necessita di maggiori informazioni di quelle del manuale. Partendo dai principi di base, il lettore può arrivare alle più avanzate tecniche utilizzabili. 192 pagine, 19.000 lire, sigla PDC 25

A scuola con il Commodore 64

Scritto da due insegnanti, questo è un libro di software per la scuola. I programmi sono in Basic, e sono tra i pochi programmi originali, studiati e scritti in Italia e non tradotti. 172 pagine, 17.000 lire, sigla PDC 26

Imparate il Basic con l'IBM Personal Computer

153 capitoli e le 5 appendici, porteranno in breve anche coloro che non hanno esperienza del linguaggio dell'IBM ad apprezzare le moltissime capacità di questo personal. 372 pagine, 26.000 lire, sigla PDC 27

Introduzione al Lisp

Questo volume presenta gli elementi essenziali del Lisp in modo piano e accessibile, e riporta e commenta una sessantina di programmi e di routine immediatamente utilizzabili. 216 pagine, 19.000 lire, sigla PDC 28

Programmi in Basic per l'elettronica

Una raccolta di routine d'aiuto alla soluzione di problemi che frequentemente si incontrano, soprattutto in fase di progettazione e impegnano moltissimo tempo in calcoli di solito noiosi e ripetitivi. 138 pagine, 14.000 lire, sigla PDC 29

Il linguaggio macchina dello Spectrum

Sono fornite le routine più interessanti per realizzare divertenti scorrimenti sullo schermo, per rinumerare le linee di un programma, per ricerche veloci, per la grafica ad alta pubblicità. 152 pagine, 16.000 lire, sigla PDC 30

32 programmi per il VIC 20

I programmi riportati spaziano dai giochi alle applicazioni domestiche, dall'istruzione alla grafica, alla matematica. Sono completamente documentati, illustrati e discussi. 250 pagine, 18.000 lire, sigla PDC 31

Introduzione all'Apple Macintosh

Il volume si propone di far conoscere questa nuova macchina, chiarire l'impostazione di fondo, spiegare quali funzioni consente di assolvere, come affrontare i problemi tipici del personal computer. 126 pagine, 16.000 lire, sigla PDC 32

Grafica e suoni con il Commodore 64

Questo volume è una guida alle capacità grafiche e sonore del Commodore 64 che porta il lettore dai primi rudimenti alla conoscenza delle funzioni degli integrati VIC II e SID e della loro programmazione. 266 pagine, 22.000 lire, sigla PDC 33

Come usare l'IBM PCjr.

Per sapere tutto sul PCjr, per chi deve decidere se questo è il computer adatto alle sue esigenze, per chi lo ha acquistato e ora vuole imparare a sfruttarlo nel miglior modo. 140 pagine, 16.000 lire, sigla PDC 34

32 programmi con il Commodore 64

Programmi da usare subito, per far funzionare utilmente il proprio calcolatore, ma anche programmi per imparare a programmare. 244 pagine, 19.000 lire, sigla PDC 35

Matematica e geometria con il Commodore 64

Ogni programma è preceduto da spiegazioni e commenti che ne fanno un libro per imparare e non solo una raccolta di listati. 200 pagine, 19.000 lire, sigla PDC 36

ZX Spectrum: tecniche avanzate di linguaggio macchina

Consente agli esperti un uso più sofisticato del computer. Vi sono descritte routine particolarmente complesse e ognuna di queste è illustrata con disegni e diagrammi. 176 pagine, 18.000 lire, sigla PDC 37

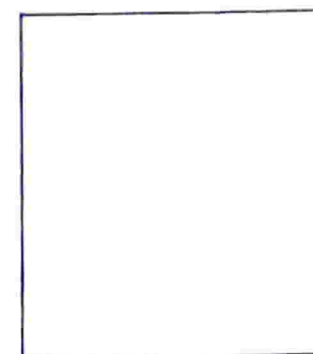
Chimica con il pocket computer

Per studenti e professionisti dediti alla chimica, propone calcoli e programmi su temi di termodinamica, chimica industriale, chimica analitica, cinetica e spettroscopia. 112 pagine, 14.000 lire, sigla PDC 38

Programmi in Basic per i geometri

Si rivolge agli aspiranti geometri e ai geometri professionisti, proponendo un uso intelligente e funzionale del computer per l'apprendimento e l'attività professionale. 212 pagine, 19.000 lire, sigla PDC 39

cedola di commissione libraria



franco muzzio editore

via makallè, 73

35138 padova

il formato della cartolina è conforme alle vigenti norme postali

cognome e nome

indirizzo

cap. località

(partita iva o codice fiscale)

firma

Questo libro si rivolge sia agli studenti allievi-geometri che ai geometri professionisti che sono interessati all'uso del computer per il proprio lavoro.

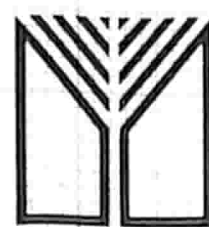
Offre una trattazione articolata su tre livelli di difficoltà: indica come usare programmi prodotti da altri e come apportarvi piccole modifiche; insegna i rudimenti del Basic attraverso applicazioni nel campo delle costruzioni; guida infine alla creazione di programmi complessi con uso della stampante.

Per tutti, l'autore fornisce una serie di programmi completi, immediatamente utilizzabili, di costruzioni, estimo e topografia.

Risparmiate tempo ed errori!

Tutti i programmi descritti in questo libro sono disponibili su un disco minifloppy, pronti per essere utilizzati.

Per l'ordine usate la cartolina inserita nel volume.



franco muzzio & c. editore

ISBN 88-7021-282-3

L. 19.000 (18.627)