Aurelio Ghersi

IL PERSONAL COMPUTER NEL CALCOLO DEGLI EDIFICI

Introduzione alla programmazione

CUEN

Pubblicato dalla CUEN
(Cooperativa Universitaria Editrice Napoletana)
P.le V. Tecchio, 80 - Facoltà di Ingegneria - 80125 NAPOLI
Tel. (39) (81) 610426 - 636667 - 7682337

1ª edizione Aprile 1987
2ª edizione Luglio 1988

Finito di stampare nel mese di Luglio 1988
presso la Poligrafica Marotta, per conto della CUEN

INDICE

CAPITOLO QUARTO - Il linguaggio BASIC

CAPITOLO QUINTO - Operazioni sulla memoria di massa

CAPITOLO SESTO - Ingresso dati con maschera

CAPITOLO SETTIMO - Fondamenti di grafica

CAPITOLO OTTAVO - Risoluzione di una trave continua

PREMESSA.

Appena una decina di anni fa il computer era uno strumento riservato a pochi, disponibile solo in ambienti universitari, grosse industrie e qualche raro studio professionale. L'evoluzione tecnica lo ha in breve trasformato in uno strumento di massa, rendendolo un componente essenziale ed ineliminabile della progettazione strutturale. I programmi disponibili in tale campo, inizialmente limitati alla risoluzione di schemi relativamente semplici, con dati geometrici e di carico totalmente definiti dall'utente, sono notevolmente cresciuti in complessità e completezza. I compiti e l'atteggiamento dell'ingegnere civile vanno quindi man mano cambiando. Scomparsa la fatica bruta del calcolo manuale, è aumentata l'importanza della sua preparazione teorica, indispensabile per valutare consapevolmente le ipotesi e le scelte del programmatore. Sempre fondamentale rimane l'esperienza pratica, necessaria per prevedere l'ordine di grandezza dei risultati ed effettuare un riscontro con i valori forniti dall'elaboratore. Forse meno essenziale, ma ugualmente importante, è anche la padronanza delle tecniche di programmazione, che consente di giudicare con più maturità i programmi fatti dagli altri, adattarli alle proprie esigenze o crearne di specifici per le necessità individuali.

A questa rapida evoluzione tecnica corrisponde una lenta trasformazione dell'insegnamento universitario. Il seminario "Il personal computer nel calcolo di edifici", da me tenuto a partire dal 1983/84, organizzato nell'ambito dei corsi di Complementi di Tecnica delle Costruzioni della Facoltà di Ingegneria di Napoli, vuole essere un piccolo contributo in questa direzione. I suoi contenuti sono andati via via crescendo e precisandosi, anche se l'assetto raggiunto non può ancora essere considerato definitivo. Gli argomenti trattati comprendono il progetto e la verifica di sezioni in c.a. presso e tensoinflesse, la risoluzione matriciale di insiemi di aste, la risoluzione iterativa di schemi intelaiati piani e spaziali a maglie rettangolari. L'interesse dimostrato dagli allievi è stato notevole, ma la loro contemporanea mancanza di basi di informatica ha reso necessario l'inserimento di una prima parte, di introduzione alla programmazione, con lo scopo di fornire quei concetti necessari per lo sviluppo di programmi nel campo dell'ingegneria civile.

Ho ritenuto che il trasporre gli argomenti delle lezioni in un insieme di testi potesse risultare utile agli allievi che seguono il seminario ed anche, più in generale, a studenti ed ingegneri che indipendentemente dall'università sentono la necessità di affrontare queste problematiche. Riorga-

nizzare e mettere per iscritto concetti esposti verbalmente è un lavoro più lungo e faticoso di quanto pensassi, il cui primo risultato è costituito dal presente volume, relativo alla parte iniziale del seminario. I restanti argomenti saranno contenuti in testi successivi, che spero vengano pubblicati con scadenze non troppo lunghe.

Parlare in generale di calcolatori o di programmazione non rientra nelle mie competenze specifiche. Lo spirito con il quale cerco di affrontare l'argomento non è quindi quello dell'esperto di elettronica ed informatica, ma solo quello di un utente e realizzatore di programmi che desidera comunicare ad altri le sue esperienze. Questa prima parte è quindi intessuta di considerazioni soggettive, spesso opinabili, e di deliberate semplificazioni (specie nel parlare della struttura dei calcolatori) che temo possano essere considerate come gravi imprecisioni dagli esperti in elettronica. Il mio obiettivo è quello di *informare* su una realtà (quella dei personal computers) in rapida evoluzione, ma soprattutto di *formare* una mentalità logica nell'approccio dei problemi. Molti ingegneri civili sono programmatori autodidatti, spesso formatisi lavorando con calcolatori di modeste capacità, e tendono a sprecare notevoli energie nel tentativo di diminuire l'ingombro di memoria o aumentare la velocità di esecuzione, a costo di ricorrere alle più contorte soluzioni. Il progresso continuo, che rende disponibili ogni giorno calcolatori più veloci e con maggiore capacità di memoria, và contro una tale impostazione. È invece fondamentale risolvere i problemi nella maniera più chiara e lineare possibile, seguendo i criteri di quella che è denominata "programmazione strutturata" e non è altro che un modo logico, schematico, di ordinare le proprie idee.

Un secondo punto che ritengo molto importante riguarda la codifica dei programmi. Nei personal computer si usa quasi sempre il linguaggio BASIC, che nonostante la tendenza ad una standardizzazione presenta tuttora notevoli differenze tra un modello e un altro. Chi lavora sempre su uno stesso calcolatore tende ad utilizzare al massimo le peculiarità del suo linguaggio, minimizzando la propria fatica immediata ma ottenendo programmi incompatibili con altri computer. In un periodo di continui cambiamenti è invece indispensabile poter trasferire con facilità i programmi realizzati, senza dover ogni volta ricominciare da capo. Nel testo cerco quindi di evidenziare le caratteristiche di base, comuni a tutti i dialetti BASIC; per le istruzioni più specifiche, non standardizzabili, faccio riferimento a due tra i linguaggi più usati, il BASIC HP ed il GWBASIC, cercando di creare nel lettore, col loro confronto reciproco, una mentalità meno ristretta.

I primi quattro capitoli del testo costituiscono la vera fase di "introduzione alla programmazione", sviluppando i concetti finora accennati. I restanti capitoli potrebbero definirsi di "programmazione avanzata", perchè affrontano problemi più complessi, di grande utilità pratica, ma che richiedono una maturazione di base.

Il primo riguarda la memoria di massa e le operazioni per prelevarvi ed immagazzinarvi programmi ed informazioni. Il conservare stabilmente i

dati diventa essenziale nel caso di programmi complessi, ad esempio per il calcolo di un edificio, per il quale possono essere necessarie più elaborazioni, con successive piccole modifiche, per ottimizzare la struttura.

Un altro problema fondamentale per chi lavora con personal computers è l'organizzazione dell'ingresso dati. Rimpiango la quantità di tempo da me sprecata per creare programmi distinti per l'input e per la correzione dei dati, solo per la non conoscenza di una impostazione di ingresso dati, mediante maschere, che avrebbe risolto in maniera ottimale tutti i miei problemi. Eppure, la mia ignoranza era parzialmente giustificata dalla totale assenza di tale argomento, almeno nei testi di programmazione più diffusi tra gli ingegneri civili. Spero di dare così almeno un piccolo contributo ad evitare che altri commettano i miei stessi errori.

Ultimo argomento trattato è la grafica, finora spesso trascurata, ma che va assumendo una importanza sempre crescente. Il mio obiettivo non è certo quello di fornirne una trattazione completa e approfondita, che richiederebbe un apposito volume, ma piuttosto di dare al lettore alcuni concetti essenziali e la padronanza di un minimo di istruzioni di base, svincolandolo dalle individualità dei diversi linguaggi.

Il capitolo conclusivo riporta un esempio completo, che racchiude tutte le problematiche esposte. Si è fatto riferimento alla risoluzione di uno schema di trave continua, problema dotato di una complessità non eccessiva, ma sufficiente a rendere l'idea delle difficoltà cui il lettore andrà incontro nelle proprie applicazioni.

Nel concludere questo lavoro, desidero ringraziare le persone che maggiormente hanno contribuito a stimolarmi ed aiutarmi. Innanzitutto, il prof. Michele Pagano, che mi ha proposto di tenere ai suoi studenti questo seminario e mi ha consigliato di fissare per iscritto gli schemi delle lezioni, ottenendo così un ampio materiale dalla cui rielaborazione deriva questo testo. In secondo luogo tutti gli amici, con i quali ho avuto modo di discutere gli argomenti quì esposti, ricevendo spesso utili suggerimenti. Sono grato in particolare ad Antonio Amendola, che mi ha fornito parte della bibliografia ed ha riletto con cura le bozze del sesto capitolo, ed a Michele Gagliardi ed Enzo Di Sarno, che hanno messo a mia disposizione i loro computer. Infine, ultima ma per me più importante, mia moglie Lia che, pur non amando troppo i calcolatori, ha letto con pazienza numerose parti del testo, suggerendomi alcuni opportuni ritocchi, e soprattutto mi ha costantemente incoraggiato, aiutandomi a superare momenti di stanchezza e sfiducia.

CAPITOLO PRIMO

STRUTTURA DEL PERSONAL COMPUTER

1.1 - Introduzione

Il continuo progresso tecnico ha messo a disposizione degli ingegneri strumenti di calcolo via via sempre più potenti. Si è infatti passati dal regolo calcolatore alle macchine calcolatrici, prima meccaniche e poi elettroniche, per giungere infine agli attuali calcolatori programmabili. Compito comune di tutte queste macchine, pur nella loro diversità, è la elaborazione di dati. Esse, cioè, ricevono dei dati di ingresso (fase definita, con termine inglese, *input*) e ricavano da essi dei risultati, o dati di uscita (in inglese *output*).

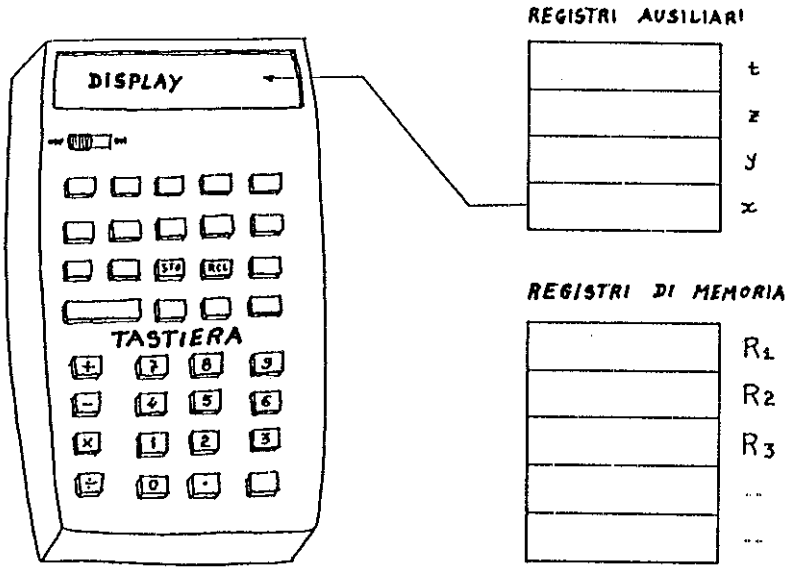
Il regolo è un calcolatore meccanico di tipo analogico. Esso infatti opera non direttamente su numeri, bensì su grandezze fisiche (lunghezze di segmenti) che si fanno corrispondere ai valori numerici con una scala logaritmica. Sfruttando le proprietà dei logaritmi si ottiene ad esempio il prodotto di due numeri come somma dei due segmenti ad essi corrispondenti.

Anche tra i calcolatori elettronici esistono macchine *analogiche*, nelle quali le grandezze "analoghe" a quelle su cui operare sono in genere tensioni e correnti elettriche. Esse trovano applicazione in campi diversi dall'ingegneria civile (per esempio nel controllo automatico di processi industriali continui) e non saranno quindi prese in esame in questo testo.

La maggior parte delle calcolatrici e dei calcolatori con i quali veniamo ormai continuamente a contatto sono invece elaboratori *numerici*, che operano su informazioni conservate in *registri* mediante una codifica convenzionale. Si vedrà in seguito cosa è fisicamente un registro e in che modo le informazioni sono contenute in esso. È però possibile prescindere dalla sua essenza fisica e definirlo come l'unità logica nella quale è memorizzata una informazione.

La calcolatrice (fig. 1) è sostanzialmente in grado di effettuare singole operazioni. Essa presenta in genere una tastiera per l'immissione di dati e di istruzioni operative ed un display per la visualizzazione dei risultati, che mostra il contenuto corrente di uno dei registri ausiliari riservati alle operazioni aritmetiche. È inoltre dotata di uno o più registri di memoria direttamente indirizzabili. Ciascun dato numerico viene inserito premendo i tasti corrispondenti alle cifre e al punto decimale. Esso è costruito, cifra dopo cifra, nel display, ma una volta terminata la sua immissione viene considerato come una unità non più scindibile. Con ap-

positi tasti lo si può ricopiare in altri registri di memoria, utilizzare per operazioni aritmetiche, e così via. Ad ogni tasto corrisponde una attività della calcolatrice, che viene svolta mediante appositi circuiti. Esiste pertanto una struttura fisica, tangibile (l'insieme dei circuiti) che consente di effettuare le singole operazioni. Essa è detta in inglese *hardware* (da hard = duro)



3.5 copia il numero 3.5, contenuto nel registro x, nel registro R1
STO R1 (il precedente contenuto di R1 viene perso)

RCL R1 il contenuto del registro R1 viene ricopiato nel registro R2
STO R2 (il contenuto di R1 non viene alterato)

PROGRAMMA = insieme dei codici corrispondenti ai tasti, nell'ordine in cui essi andrebbero premuti (*programma in linguaggio macchina*)

Fig 1

Il calcolatore programmabile consente invece elaborazioni molto più complesse; ad esempio, può ricevere in ingresso dati geometrici e di carico

relativi a un telaio e fornire in uscita le caratteristiche di sollecitazione nelle aste. Non esiste in questo caso un circuito che effettua direttamente l'elaborazione richiesta (ad esempio un circuito per la risoluzione dei telai). Essa è invece realizzata come successione di operazioni più semplici. L'insieme di istruzioni che indica al calcolatore quali operazioni effettuare, e con che ordine, è detta programma. Anche con una semplice calcolatrice è possibile effettuare elaborazione complesse, premendo in sequenza più tasti. Nel caso che essa sia programmabile, il programma è una successione di codici corrispondenti ai tasti che indica l'ordine con cui questi andrebbero premuti. Si parla in tal caso di programma in "linguaggio macchina", perchè a ciascuna istruzione corrisponde direttamente una operazione elementare. Si definiscono invece "linguaggi evoluti" quelli nei quali a ciascuna istruzione corrisponde una attività più complessa, cioè più operazioni elementari.

Il calcolatore programmabile è dotato in definitiva, in aggiunta alla parte hardware, di una struttura logica, non tangibile (l'insieme dei programmi) che lo rende capace di elaborazioni molto più generali di quelle consentite dal singolo circuito. Essa è detta in inglese *software* (da soft = soffice). Ciascun calcolatore è dotato di un insieme di programmi, realizzati da chi lo ha progettato, che fanno parte integrante del suo corredo e sono essenziali per la sua utilizzazione; essi costituiscono il "software di base" della macchina. L'insieme dei programmi scritti da chi usa la macchina è detto invece "software dell'utente".

CALCOLATORE
ELABORATORE ELETTRONICO DI DATI

dati di ingresso (INPUT) → CALCOLATORE HARDWARE + SOFTWARE → dati di uscita (OUTPUT)

CALCOLATORE **NUMERICO** — opera su informazioni contenute in registri

CALCOLATORE = HARDWARE + SOFTWARE

HARDWARE insieme dei circuiti che consentono operazioni elementari

SOFTWARE insieme dei programmi che rendono il calcolatore capace di elaborazioni complesse

SOFTWARE DI BASE — i programmi che fanno parte del corredo della macchina

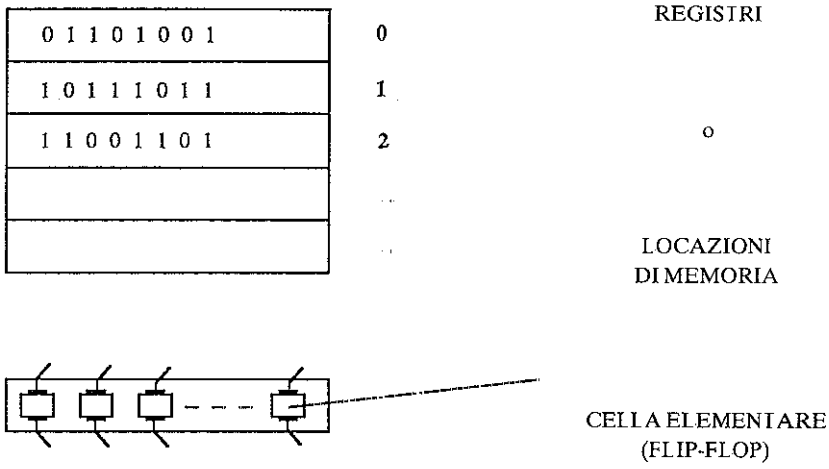
SOFTWARE DELL'UTENTE — i programmi scritti da chi usa il calcolatore

1.2 - Registri di memoria e informazioni.

Il registro (fig 2) è un insieme di celle elementari, cioè di elementi fisici che possono assumere due stati stabili (*multivibratori bistabili* o *flip-flop*) I due stati vengono indicati simbolicamente con le cifre 0 e 1, che sono le cifre utilizzate nella numerazione binaria. Per tale motivo il simbolo col quale si indica lo stato della cella elementare è detto BIT (dall'inglese BInary digiT = cifra binaria). Lo stato del registro è rappresentato mediante una sequenza di cifre binarie, cioè con un insieme di bit.

Il numero di celle che costituiscono il registro è detto "lunghezza", o numero di bit, del registro. Molto diffusi tra i personal computer sono i registri composti da otto celle elementari, cioè ad otto bit. In inglese un tale insieme di 8 bit è indicato col termine BYTE.

I registri di memoria di un calcolatore costituiscono un insieme ordinato. Ciascuno di essi può quindi essere individuato in maniera univoca mediante il suo numero d'ordine, che è detto "indirizzo" del registro.



BIT = BInary digiT (cifra binaria) simbolo col quale si indica lo stato della cella elementare
BYTE = insieme di 8 bit

Fig. 2

Un dato importante per valutare la quantità di informazioni che un calcolatore può contenere è il numero di registri presenti in esso, ovvero la sua "capacità di memoria". Per indicare sinteticamente tale numero si usa il termine K (kappa) per individuare $2^{10}=1024$ registri e il termine M (mega) per individuare $2^{20}=1048576$ registri. Si dice quindi, ad esempio, che un calcolatore ha una memoria di 64K quando esso possiede $64\times1024=65536$ registri. Attualmente si va affermando la tendenza a valutare la capacità di memoria con riferimento non al numero di registri ma al numero di bytes, per consentire un immediato confronto tra calcolatori aventi registri di differente lunghezza. Così ad esempio per un calcolatore dotato di 64K registri a 16 bit si dirà che esso ha una capacità di memoria di 128K bytes, perchè ogni registro equivale a 2 bytes.

Per chiarire come le informazioni vengono conservate nei registri del calcolatore è opportuno innanzitutto soffermarsi sul significato del termine "informazione". Si prendano ad esempio in esame le seguenti tre affermazioni:
a) per laurearsi in ingegneria occorre aver superato 29 esami;
b) ho un appuntamento alle ore 11.45;
c) ho un amico che si chiama Antonio
Le informazioni fornite da ciascuna frase sono caratterizzate innanzitutto da un *valore* (29, 11.45, Antonio). Inoltre da una frase all'altra cambia non solo il valore ma anche il *tipo* di informazione. Infatti "29" è un numero intero (non può essere un numero decimale, perchè non ha senso dire "... occorre aver superato 29.5 esami"). "11.45" non è un numero ma un orario, e come tale è sottoposto a leggi aritmetiche diverse da quelle dei numeri (per esempio 11.45+0.30=12.15 e non 11.75). "Antonio" infine è un insieme di caratteri, per il quale le operazioni aritmetiche non hanno proprio significato.

Una informazione può essere conservata in un registro, purchè se ne trasformi il valore in una sequenza di cifre binarie mediante una specifica convenzione (*codice*), che dipende dal suo tipo. Viceversa dalla sequenza binaria contenuta in un registro si può risalire al valore, purchè se ne conosca il tipo e quindi il criterio per la decodifica. Molto spesso il numero di cifre binarie necessario per rappresentare il valore di una informazione è maggiore del numero di bit di un registro. In tal caso verranno utilizzati più registri fisici consecutivi, trattati dal calcolatore come un insieme unitario; il loro numero è in genere strettamente correlato al tipo di informazione.

Valore e tipo non sono però sufficienti a costituire una informazione. Per esempio il numero intero 29, preso a se stante, non vuol dire niente. Esso diventa una informazione solo se associato al concetto "esami necessari per laurearsi", cioè se dotato di un *attributo*. In definitiva l'insieme di bit di un registro può essere interpretato come un valore quando se ne conosce il tipo ed assume significato di informazione per chi ne conosce l'attributo.

CALCOLATORE NUMERICO	= elaboratore di dati (INFORMAZIONI) contenuti in REGISTRI
REGISTRO	= insieme di celle elementari
lunghezza	– numero di celle che lo compongono
indirizzo	– numero d'ordine nell'insieme ordinato di registri
INFORMAZIONE caratterizzata da:	VALORE TIPO ATTRIBUTO

Il VALORE dell'informazione:
- è conservato in uno o più registri di memoria
- è codificato (cioè trasformato in una sequenza binaria) con un codice che dipende dal tipo

1.3 - I calcolatori elettronici.

Si è finora parlato, genericamente, di "calcolatore elettronico programmabile" In effetti, la gamma di calcolatori esistenti oggi sul mercato è molto ampia ed è opportuno premetterne una panoramica globale all'esame più dettagliato di quello che più specificamente è l'oggetto del presente testo, il personal computer.

Il progresso tecnico è talmente veloce che qualsiasi classificazione rischia di essere obsoleta già dopo pochi anni. Le distinzioni che si faranno di seguito non possono quindi essere ritenute di validità assoluta, ma solo il tentativo di descrivere l'immagine attuale di una realtà in movimento.

Al livello inferiore della scala si colloca la *calcolatrice portatile programmabile* Di aspetto pressocchè coincidente a quello dei modelli non programmabili, essa è dotata di tastiera e display ed ha registri ad 8 bit. I primi modelli possedevano poche centinaia di registri ed operavano solo in linguaggio macchina. Attualmente si è arrivati a capacità di memoria fino a 16K e si è reso disponibile un linguaggio più evoluto, il BASIC.

Il gradino successivo è costituito dal *micro computer* In genere non portatile, è dotato sempre di una tastiera per l'input, un video per la visualizzazione dei risultati ed una memoria di massa per conservare stabilmente programmi o dati. I registri sono spesso ad 8 bit, ma diventano sempre più diffusi i modelli con registri a 16 bit.

Principalmente in base alla destinazione d'uso, è possibile dividere i micro computers in due categorie: l'*home computer* e il *personal computer*. Il primo è nato come aiuto in casa (home = casa), per i conti della spesa, per i compiti degli studenti ed anche per i giochi La sua memoria è compresa tra 16K e 64K. È programmabile quasi esclusivamen-

te in BASIC ed ha un costo relativamente basso, soprattutto per la possibilità di utilizzare come video un comune televisore e come memoria di massa un generico registratore. Il secondo è destinato ad un uso più professionale. Prevede in genere una serie completa di unità periferiche specifiche (video, disco e stampante) e talvolta consente l'uso di più linguaggi (oltre al BASIC, anche il FORTRAN o il PASCAL). Ha una capacità di memoria più elevata, che può arrivare anche a 1M.

Al livello immediatamente superiore si colloca il *mini computer*, dotato di registri a 16 o 32 bit, con capacità di memoria e velocità di elaborazione ancor più elevata. La gamma di linguaggi disponibile è molto ampia ed anche le unità periferiche sono più numerose e sofisticate. Spesso è consentita la multiprogrammazione, cioè l'esecuzione contemporanea di più programmi mediante una ripartizione delle risorse del sistema (unità periferiche, unità aritmetica, eccetera), gestita automaticamente dal calcolatore.

I modelli più potenti sono infine i *main-frames*, cioè i calcolatori dei grossi centri di calcolo. Dotati di registri a 32, 36 o 64 bit, essi sono destinati a soddisfare contemporaneamente le esigenze di una molteplicità di utenti. Alla capacità di multiprogrammazione, si affianca spesso anche quella di multielaborazione, cioè il possesso di più unità aritmetiche in parallelo. Si va inoltre sempre più diffondendo la tendenza a creare collegamenti tra di essi, realizzando reti di calcolatori a livello nazionale ed internazionale.

CALCOLATRICE PORTATILE PROGRAMMABILE	memoria 1K - 16K linguaggio macchina (o BASIC)
MICRO COMPUTER	registri a 8 o 16 bit
- HOME COMPUTER	memoria 16K - 64K tastiera; normale televisore come video linguaggio BASIC
- PERSONAL COMPUTER	memoria 32K - 1M serie completa di unità periferiche linguaggio BASIC (anche FORTRAN, PASCAL)
MINI COMPUTER	registri a 16 o 32 bit
MAIN FRAME	registri a 32, 36 o 64 bit

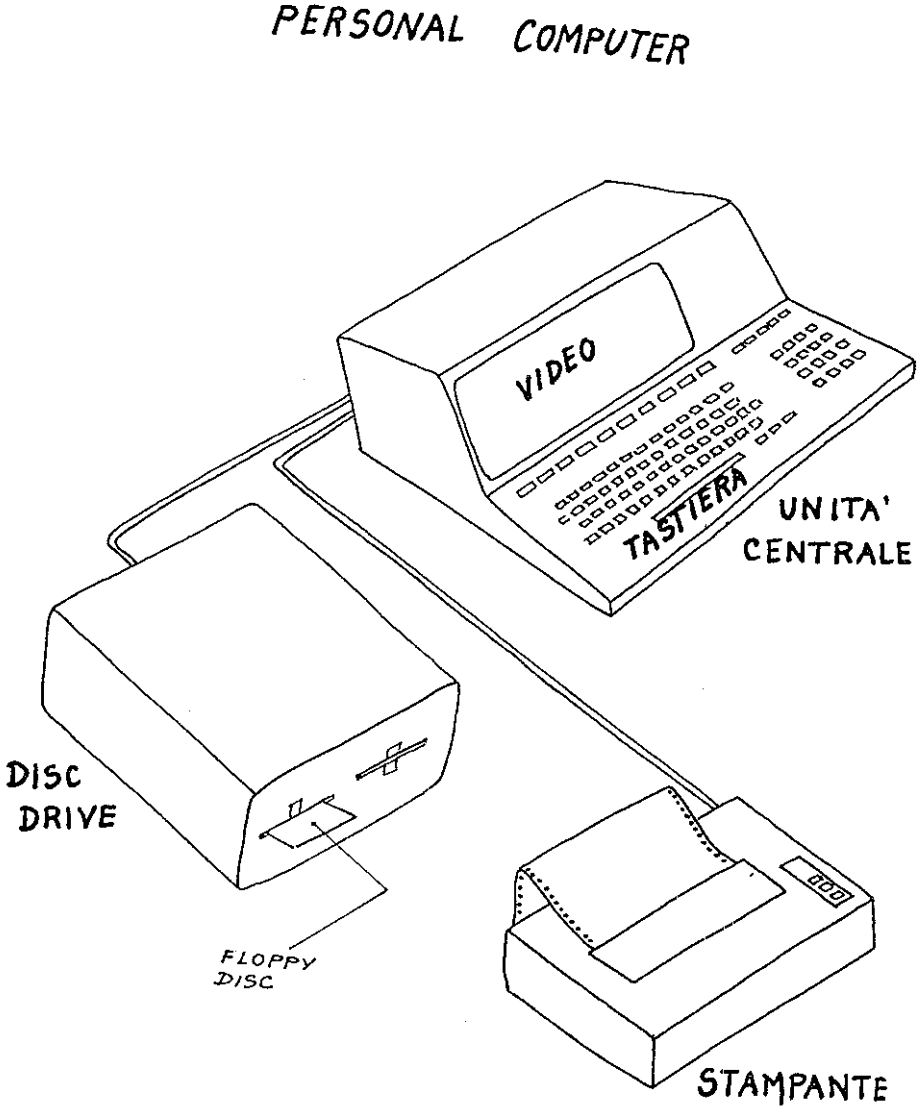

PERSONAL COMPUTER
VIDEO
TASTIERA
UNITA' CENTRALE
DISC DRIVE
FLOPPY DISC
STAMPANTE

Fig 3

1.4 - Il personal computer.

L'ingegnere può trovarsi ad utilizzare modelli di calcolatore che spaziano nell'intera gamma innanzi descritta La calcolatrice programmabile, ad esempio, può essere utilizzata per quei calcoli di massima che ci si trova a dover svolgere nelle situazioni più disparate, in cantiere o durante l'impostazione di un progetto Un main-frame diventa invece indispensabile per le elaborazioni di particolare complessità a volte richieste Ma lo strumento più usato è sicuramente il personal computer (fig 3), che affianca ad una potenza di elaborazione non indifferente un ingombro ed un costo non eccessivi.

Il personal, come del resto tutti i calcolatori, è costituito dall'unità centrale e da unità periferiche di ingresso, di uscita e memoria di massa

A) Unità centrale.

L'unità centrale, vero e proprio "cuore" del calcolatore, è composta dalla *CPU* e dalla *memoria centrale*

La CPU (*Central Processing Unit*, unità di governo e di calcolo) contiene un insieme di registri, di puntatori che individuano gli operandi, di indicatori di stato ed una unità logico-aritmetica Essa esamina il codice dell'operazione elementare da eseguire (per esempio, ricopiare in un suo registro il contenuto di un registro della memoria centrale del calcolatore, oppure sommare tra loro il contenuto di due suoi registri) ed attiva quindi i circuiti necessari per compiere tale operazione.

La memoria centrale è costituita dall'insieme ordinato dei registri di memoria. Lo stato dei registri è stabile solo finche essi sono sotto tensione: la mancanza di alimentazione elettrica provoca l'azzeramento della memoria e la perdita delle informazioni in essa contenute. È possibile accedere al singolo registro per operazioni di lettura e scrittura; per questo motivo la memoria è detta RAM (*Random Access Memory*, cioè memoria ad accesso casuale).

026 000 011 210 303 030 011 210
153 031 305 031 266 031 247 031
342 207 022 210 022 210 106 251
070 204 230 136 262 001 377 251
340 040 262 030 377 321 000 140
366 012 262 065 210 261 014 140
036 306 000 000 316 274 011 316
030 030 230 316 306 207 117 220
367 003 316 263 000 116 220 367

PICCOLA PARTE
DI UN
SISTEMA OPERATIVO

Fig 4

Una parte della memoria centrale contiene il *sistema operativo*, che è un programma in linguaggio macchina, cioè una sequenza di codici che corrispondono ad operazioni elementari (fig. 4). Esso rende possibile l'esecuzione di istruzioni operative più complesse e l'adozione di linguaggi evoluti. Pertanto la facilità d'uso del computer dipende in gran parte dalla qualità del suo sistema operativo.

In numerosi calcolatori il sistema operativo viene caricato in memoria da un disco. È possibile quindi utilizzarne versioni più aggiornate, o anche completamente differenti, semplicemente usando un diverso disco. In altri modelli il sistema operativo è invece residente in maniera stabile nella memoria centrale. Perchè ciò avvenga, i registri che lo contengono devono mantenere il loro stato anche a calcolatore spento. Inoltre deve essere possibile leggere il loro contenuto ma non modificarlo, per non alterare il sistema operativo. Si utilizza quindi un tipo particolare di memoria detto ROM (*Read Only Memory*, cioè memoria di sola lettura). In quesi casi non è necessario un caricamento iniziale; non è però possibile cambiare il sistema operativo se non sostituendo fisicamente la parte di memoria che lo contiene oppure aggiungendo altre ROM tali da superare ed annullare le sue istruzioni.

Il sistema operativo può essere molto complesso e quindi occupare un numero elevato di registri di memoria. È quindi importante conoscerne l'ingombro per valutare correttamente la capacità di memoria a disposizione dell'utente.

C P U	CENTRAL PROCESSING UNIT UNITA' DI GOVERNO E DI CALCOLO - contiene registri, puntatori ed una unità logico-aritmetica - consente e sovrintende l'esecuzione delle operazioni elementari
SISTEMA OPERATIVO	PROGRAMMA IN LINGUAGGIO MACCHINA - consente l'esecuzione di operazioni più complesse - controlla (o contiene in se) l'interprete BASIC

LA FACILITÀ D'USO DEL CALCOLATORE DIPENDE IN GRANDE MISURA DALLA QUALITÀ DEL SUO SISTEMA OPERATIVO

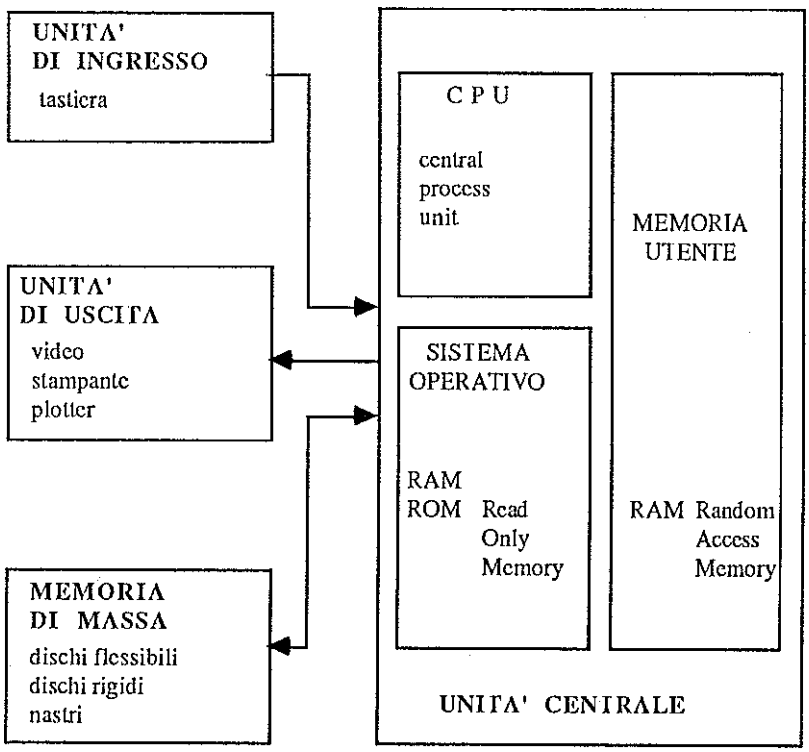
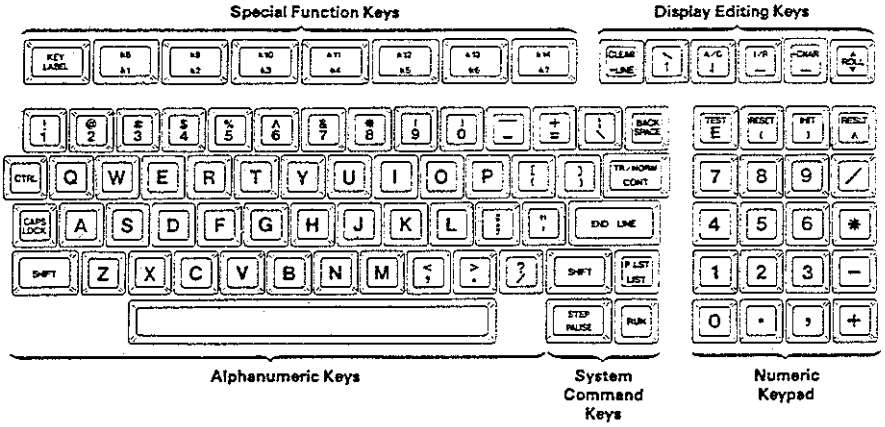
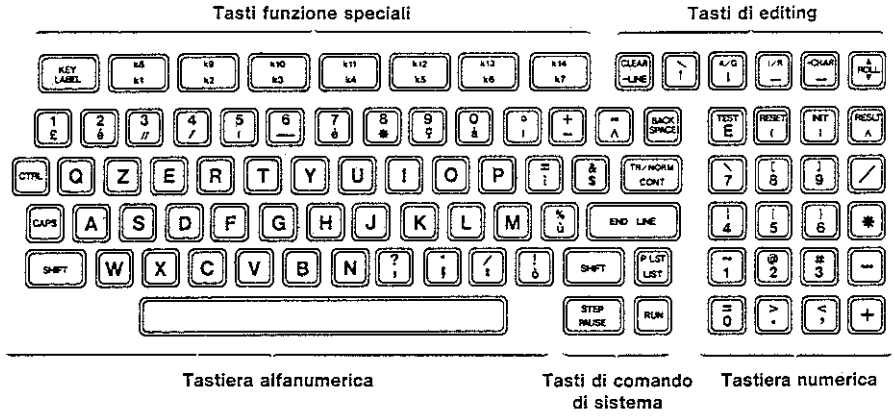

UNITA'
DI INGRESSO
tastiera
UNITA'
DI USCITA
video
stampante
plotter
MEMORIA
DI MASSA
dischi flessibili
dischi rigidi
nastri
C P U
central
process
unit
SISTEMA
OPERATIVO
RAM
ROM Read
Only
Memory
MEMORIA
UTENTE
RAM Random
Access
Memory
UNITA' CENTRALE

Fig 5



TASTIERA ASCII



TASTIERA ITALIANA

B) Unità di ingresso.

La tipica unità di ingresso del personal computer è la *tastiera* (fig. 6). Mediante essa vengono immessi nel calcolatore sia i programmi che i dati da elaborare. Essa è sostanzialmente simile alla tastiera di una comune macchina da scrivere. Contiene infatti tasti per i caratteri alfabetici e numerici, ai quali si aggiungono tasti di comando, per richiedere l'esecuzione di particolari compiti (per esempio interrompere l'elaborazione in corso).

Spesso per uno stesso modello di computer sono disponibili differenti tipi di tastiere, in cui la posizione ed il significato di alcuni tasti varia per soddisfare i requisiti linguistici delle diverse nazioni. Esiste quindi una "tastiera ASCII", che rispetta le indicazioni dell'*American Standard Code for Informations Interchange*, ed una "tastiera italiana", che contiene ad esempio le lettere accentate, non usate nella lingua inglese.

Esistono anche altre unità di ingresso, meno frequentemente usate: il *touch screen*, il *mouse*, la *tavoletta grafica*. Esse consentono sostanzialmente di comunicare al calcolatore, con differente grado di precisione, una posizione nel piano. Il *touch screen* (o "schermo sensibile al tocco") consente di indicare una posizione nello schermo semplicemente toccandolo; è in genere utilizzato per selezionare tra diverse alternative presentate su esso. Il *mouse* (cioè "topo", per l'aspetto piccolo e tondeggiante) è un oggetto il cui movimento viene messo in corrispondenza con quello di un puntatore sullo schermo; può essere utilizzato per scegliere tra più alternative, ma anche per tracciarvi disegni con appositi programmi di grafica. La *tavoletta grafica* consente infine di trasmettere al calcolatore un disegno, percorrendone le linee con una apposita *penna*.

C) Unità di uscita.

Numerose sono le unità di uscita, mediante le quali il computer fornisce all'utente il risultato delle sue elaborazioni. Sempre presenti sono il *video* e la *stampante*, ai quali può essere aggiunto anche il *plotter*.

Il video consente la visualizzazione di caratteri alfanumerici (lettere, cifre o simboli speciali). La posizione corrente, cioè il punto dello schermo nel quale andrebbe posto il successivo carattere, è individuata dal *cursore*, che è di solito un rettangolino lampeggiante. In molti calcolatori è possibile spostare il cursore, premendo appositi tasti, in qualsiasi punto dello schermo.

Oltre al mostrare i risultati di una elaborazione, uno dei compiti fondamentali del video è quello di aiutare nella fase di inserimento di dati o programmi. Infatti, tutto ciò che viene immesso nel calcolatore mediante tastiera viene in genere immediatamente ritrasmesso al video (e quindi visualizzato), consentendo così un controllo della correttezza dei tasti premuti.

In alcuni modelli il video ha anche capacità grafiche. Esso in tal caso

consente di tracciare linee che uniscano due suoi punti qualsiasi ed ottenere così diagrammi, disegni, eccetera

La stampante consente di imprimere su carta un insieme di caratteri, e quindi di riprodurre in maniera stabile i risultati di una elaborazione o il testo di un programma. Essa opera in maniera sequenziale, avanzando la carta dopo ogni riga di stampa. Non è quindi possibile tornare all'indietro, come invece accade per il cursore del video.

Le stampanti più comunemente utilizzate per un personal sono dotate di una testina mobile che compone ciascun carattere mediante un insieme di aghi, secondo una matrice di punti. Esse possono essere suddivise sostanzialmente in due tipi: stampanti a impatto e stampanti termiche. Le prime operano in maniera analoga alle comuni macchine da scrivere, in quanto il carattere viene impresso sulla carta mediante un nastro inchiostrato. I modelli più comuni hanno velocità di stampa compresa tra 80 e 160 caratteri per secondo. Le seconde, invece, utilizzano una carta speciale (detta carta termica), sensibile al calore, sulla quale i caratteri vengono impressi dalla testina proprio sfruttando questa caratteristica. Esse sono in genere più silenziose e veloci, ma le scritte tendono col tempo a perdere di chiarezza, specie se la carta rimane esposta a lungo al sole. Il costo della carta termica è inoltre maggiore rispetto a quello della carta comune.

Alcune stampanti hanno anche capacità grafiche, cioè consentono la copia su carta di un disegno visualizzato in un video grafico (*hard-copy* del video).

Il plotter permette di eseguire diagrammi o disegni direttamente su carta. Il risultato che esso fornisce è molto migliore di quello ottenuto mediante la hard-copy del video. Esso infatti opera con un pennino e traccia linee effettivamente continue, mentre la stampante riproduce le linee con un insieme discreto di punti. I modelli di plotter più economici consentono di lavorare su un foglio fisso, mentre i modelli più avanzati permettono l'uso di rotoli di carta, con possibilità di avanzamento e riavvolgimento, e quindi la realizzazione di disegni di dimensioni molto ampie

D) Memoria di massa.

Per conservare dati e programmi stabilmente, anche a calcolatore spento, si ricorre a supporti magnetici (*nastri* o *dischi*), sui quali l'informazione binaria è registrata da una testina che magnetizza un tratto del substrato magnetico.

Il nastro, nel caso dei personal computer, è racchiuso in una cassetta analoga a quelle utilizzate dai registratori. L'accesso alle informazioni è sequenziale, cioè occorre riavvolgere il nastro per arrivare al tratto nel quale sono conservate le informazioni che interessano. Ciò comporta tempi elevati per il prelievo delle informazioni, oltre ad una più facile usura del nastro stesso.

Molto più usato è il disco. Esso è in continua rotazione dentro l'unità che lo contiene (*disc drive*). La testina si sposta radialmente ed ha quindi accesso praticamente immediato a qualsiasi punto del disco. I dischi possono essere suddivisi in dischi flessibili (*floppy disk*) o dischi rigidi. I primi, molto economici, sono estraibili dal drive e possono essere conservati a costituire una biblioteca di dati o programmi. La loro capacità può variare da alcune centinaia di K fino ad oltre 1 M. I secondi sono invece fissi nel drive; la loro capacità di memoria è dell'ordine di parecchi M

CAPITOLO SECONDO

IL PROGRAMMA

Come si è già sottolineato in precedenza, ciò che rende il calcolatore capace di elaborazioni complesse non è tanto l'insieme dei suoi circuiti quanto quello dei programmi per esso realizzati.

Si può definire "programma" una sequenza di istruzioni, scritte in un linguaggio comprensibile al calcolatore, che descrivono come elaborare i dati per risolvere un problema.

È opportuno evidenziare i concetti insiti in questa definizione. Il programma è, innanzi tutto, una sequenza di istruzioni. Il computer può infatti svolgere un compito per volta; è quindi necessario individuare con chiarezza l'ordine col quale esso deve effettuare le operazioni.

In secondo luogo, ciascuna istruzione deve essere comprensibile al calcolatore. Ogni computer, per mezzo dei circuiti che lo compongono, può svolgere un insieme di operazioni elementari, individuate mediante un codice (linguaggio macchina) caratteristico del singolo modello. Per semplificare il lavoro del programmatore sono però disponibili dei programmi (interpreti o compilatori), contenuti o controllati dal sistema operativo, che consentono all'utente di adoperare istruzioni complesse, che vengono da loro trasformate nei codici delle equivalenti operazioni base. L'interprete opera durante l'esecuzione del programma, traducendone di volta in volta la singola istruzione. Il compilatore opera invece sul testo dell'intero programma, creandone una versione in linguaggio macchina, direttamente eseguibile, che può quindi fornire risultati molto più rapidamente. In particolare, il BASIC è normalmente un linguaggio "interpretato", ma esistono per molti calcolatori anche dei compilatori BASIC, utilizzati per velocizzare programmi già collaudati.

Col termine "linguaggio simbolico" si intende l'insieme dei nomi simbolici delle istruzioni in tal modo possibili. Oltre alla maggiore potenza espressiva, un ulteriore vantaggio rispetto al linguaggio macchina è quello di poter scrivere un programma indipendentemente dal modello di computer su cui operare. I linguaggi simbolici sono infatti più o meno standardizzati. Ad esempio per il FORTRAN, linguaggio scientifico molto usato soprattutto con i calcolatori di dimensioni medie e grandi, è universalmente utilizzato lo standard ASCII. Meno codificato è il BASIC,

per il quale esistono numerose versioni, che ne costituiscono quasi dei "dialetti" differenti.

Infine, la sequenza di istruzioni del programma deve indicare al calcolatore in che modo risolvere il problema. Deve perciò prevedere che vengano richieste in ingresso delle informazioni, da conservare in appositi registri di memoria individuati da nomi simbolici (variabili) Deve quindi indicare in che modo elaborare i valori di ingresso per ottenere quelle informazioni che costituiscono la soluzione del problema. Deve inoltre specificare sotto che forma, e a quale unità, vengano inviati in uscita i valori così ottenuti

Realizzare un programma non è un compito particolarmente difficile, ma richiede soprattutto molto ordine e chiarezza di idee. Un errore che commette comunemente chi ha a disposizione un personal computer è il mettersi a lavorare direttamente al calcolatore, lasciandosi guidare dall'ispirazione del momento È invece importante ragionare a tavolino, impostando lo schema del programma nel rispetto di regole strutturali ben precise e cercando di codificarlo in un linguaggio facilmente leggibile e comprensibile.

Nella realizzazione di un programma si possono individuare cinque fasi: analisi del problema, programmazione, codifica, controllo e documentazione.

A) Analisi del problema.

Come non si può spiegare ad altri un concetto che non si conosce, così non si può far trovare al calcolatore la risposta ad un quesito che non si è in grado di risolvere manualmente. Il primo passo è quindi sempre quello di analizzare il problema, ricercarne le possibili vie di soluzione ed individuare tra esse quella che si ritiene la più adatta al computer Nei casi più semplici l'analisi si riduce unicamente ad un lavoro di organizzazione logica di procedimenti di calcolo già noti e alla identificazione dei dati e dei risultati del problema, mentre nei casi complessi essa richiede una notevole ricerca bibliografica o la individuazione di nuove metodologie

B) Programmazione.

Il lavoro di programmazione consiste nel definire la struttura logica sequenziale del programma. Esso viene effettuato a livelli successivi, con grado di dettaglio via via crescente. Al fine di ottenere un programma di facile comprensione si preferisce utilizzare un numero limitato di strutture logiche, che sono descritte nel capitolo successivo.

Un modo di rappresentare graficamente la struttura del programma è costituito dal "diagramma di flusso" In esso ogni blocco di istruzioni è racchiuso in un rettangolo, mentre l'ordine di esecuzione dei blocchi è

indicato mediante linee orientate

C) Codifica.

Analisi del problema e programmazione sono fasi sostanzialmente generali, indipendenti dal calcolatore con cui si deve operare

Una volta individuata la struttura logica del programma, essa deve essere codificata, cioè tradotta in istruzioni di un linguaggio comprensibile allo specifico calcolatore che si vuole utilizzare per le applicazioni. Ogni linguaggio ha delle regole formali che devono essere rispettate rigorosamente Il linguaggio più comune tra i personal computer, al quale esclusivamente si fa riferimento in questo testo, è il BASIC.

D) Controllo.

Il programma, una volta codificato, è pronto per l'uso È però necessario controllare se le fasi precedenti sono state tutte svolte correttamente Si procederà quindi ad un certo numero di esecuzioni di prova, utilizzando esempi risolti manualmente o trovati nella bibliografia. Se il confronto ha esito negativo, occorre riesaminare il lavoro nell'ordine inverso (codifica, programmazione, analisi) La ricerca ed eliminazione degli errori è detta in inglese *debugging*, cioè spulciamento (indicando con bug, pulce, l'errore nel programma)

Poichè ogni programma di una certa complessità contiene in se numerose alternative, occorre effettuare un numero di prove tale da controllare tutti i casi possibili In particolare, può essere opportuno effettuare alcune esecuzioni con dati di ingresso completamente casuali. Un frequente errore di analisi e programmazione può infatti nascere da scelte fatte proprio perchè si conosce l'ordine di grandezza dei valori su cui si opera.

Quando le alternative sono troppo numerose, diventa impossibile effettuare tante prove da garantire assolutamente la correttezza del programma. Sarà opportuno in tal caso mantenerlo "sotto osservazione" per un certo tempo, cioè utilizzarlo sottoponendone i risultati ad una analisi critica costante

E) Documentazione.

Molto spesso, una volta accertato che il programma funziona correttamente, esso viene messo da parte, anche per lungo tempo Quando poi diventa necessario farne uso, o modificarlo per adattarlo a nuove esigenze, ci si trova di fronte a qualcosa di oscuro e incomprensibile, su cui non si sa come mettere mano

È quindi fondamentale completare la realizzazione di un programma

con una adeguata documentazione Essa deve essere divisa in due parti. Una, rivolta all'utente del programma, deve illustrare il procedimento risolutivo e le ipotesi su cui esso si basa, nonchè le modalità per utilizzarlo (quali dati fornire, e in che ordine) La seconda, rivolta al programmatore, deve chiarirne la struttura interna, indicare l'attributo e il tipo delle variabili utilizzate, e contenere ogni altra informazione che possa servire a renderlo più comprensibile.

PROGRAMMA:	**sequenza** di istruzioni
	scritte in un **linguaggio** comprensibile al calcolatore
	che descrivono come **elaborare i dati** per risolvere un problema

FASI NELLA STESURA DI UN PROGRAMMA:

1. **ANALISI DEL PROBLEMA**	Richiede conoscenza del problema e delle possibili vie di soluzione
2. **PROGRAMMAZIONE**	Definizione della struttura logica sequenziale
	Viene effettuata in fasi successive, con grado di dettaglio crescente
3. **CODIFICA**	Traduzione della struttura logica sequenziale in istruzioni di un linguaggio prescelto
4 **CONTROLLO**	Esecuzione di prova. Individuazione ed eliminazione di eventuali errori di codifica, di programmazione o di analisi
5 **DOCUMENTAZIONE**	Ipotesi del programma e modalità d'uso Struttura interna del programma

CAPITOLO TERZO

DIAGRAMMA DI FLUSSO E STRUTTURE LOGICHE DI PROGRAMMAZIONE

3.1 - Variabili

Definire la struttura logica sequenziale del programma, vuol dire indicare quali informazioni il calcolatore deve ricevere in ingresso, come le deve elaborare e cosa deve fornire in uscita come risultato.

Ogni informazione è caratterizzata da tipo, attributo e valore. Quest'ultimo è conservato, in forma codificata, in uno o più registri di memoria. Quando si opera in linguaggio macchina, questi devono essere indicati mediante il loro indirizzo. Se si utilizza un linguaggio evoluto, è invece possibile individuarli in maniera simbolica, prescindendo dall'indirizzo. Per indicare il nome simbolico scelto a tal fine si usa il termine "variabile". Esso verrà scelto, rispettando i criteri formali inposti dal linguaggio, usualmente in modo da richiamare alla mente l'attributo della informazione. Rifacendosi al primo degli esempi trattati nel paragrafo 1.2, per indicare i registri che contengono "il numero di esami necessari per laurearsi" si potrà scegliere il nome "ESAMI". Così, dicendo che la variabile ESAMI vale 29 si intende che lo stato dei registri, posti all'indirizzo associato al nome ESAMI, rappresenta il numero 29 nel codice di tipo "numero intero".

Da un punto di vista puramente logico, si indicherà con "variabile" qualsiasi informazione il cui valore possa cambiare (e quindi richiede una apposita area di memoria per conservarlo). Si indicherà invece con "costante" ogni informazione il cui valore è definito in maniera fissa, invariabile.

ELABORAZIONE DI INFORMAZIONI	
Il valore dell'informazione è conservato in registri di memoria	
In LINGUAGGIO MACCHINA:	ogni registro è individuato dal suo NUMERO D'ORDINE
Nei LINGUAGGI EVOLUTI:	i registri che contengono il valore dell'informazione sono individuati da un NOME SIMBOLICO (in genere scelto in modo da ricordare l'attributo)

VARIABILE =	nome simbolico che individua i registri di memoria che contengono il valore dell'informazione

3.2 - Elementi del diagramma di flusso

Il diagramma di flusso è una rappresentazione grafica simbolica della struttura logica sequenziale del programma. I simboli utilizzabili in esso sono numerosi, e non sempre rigorosamente formalizzati. Si è preferito adottarne nel seguito il minimo indispensabile.

Le istruzioni operative, singole o più spesso a gruppi, sono racchiuse in figure geometriche. Queste sono connesse con linee orientate, con una freccia che indica il verso di percorrenza e quindi l'ordine di esecuzione delle istruzioni. La sequenza "normale" di blocchi è dall'alto verso il basso. La freccia viene quindi omessa nei casi in cui non può sorgere dubbio sul verso

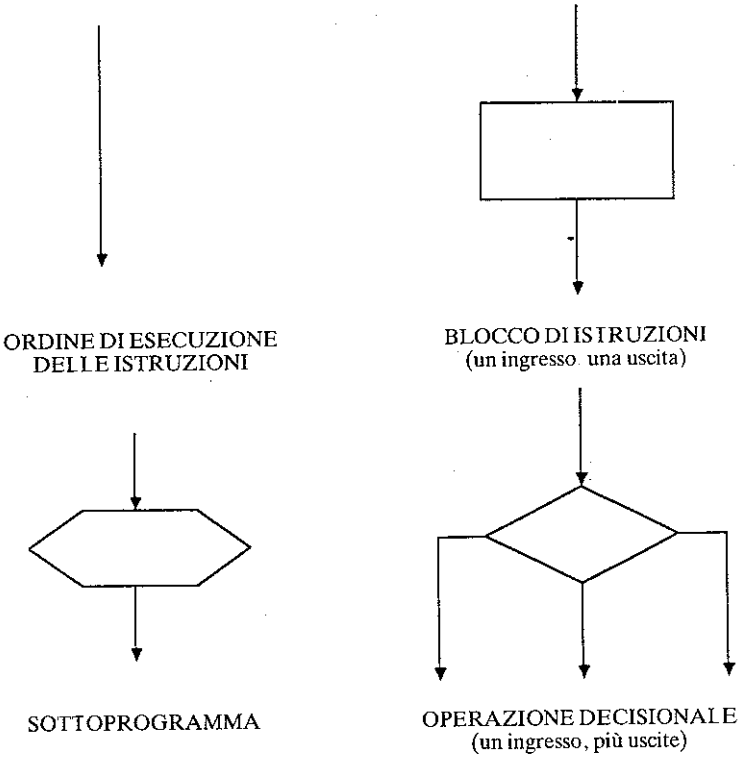


Fig. 7

Per distinguere i diversi tipi di operazioni si utilizzano tre figure geometriche Il rettangolo indica una operazione generica (di ingresso, di uscita, di calcolô) Un esagono allungato indica invece il riferimento ad una procedura predefinita (sottoprogramma). Entrambe tali figure presentano soltanto due linee di connessione, una in ingresso ed una in uscita. Un rombo indica infine le operazioni decisionali. In tal caso si ha ancora un solo ingresso ma due (o più) uscite.

DIAGRAMMA DI FLUSSO:
Rappresentazione grafica simbolica della struttura logica sequenziale del programma (cioè della sequenza di istruzioni)

3.3 - Sequenza continua di blocchi.

La forma più immediata di diagramma di flusso è costituita da una sequenza continua di blocchi di istruzioni, rappresentata graficamente da più rettangoli posti l'uno sopra l'altro e collegati da una linea verticale (fig. 8).

Un tale schema si ritrova non solo nei programmi più semplici, ma anche nei livelli di minor dettaglio dei programmi più complessi. Ad esempio è consueto individuare in ogni programma tre blocchi sequenziali: ingresso dei dati, elaborazione, uscita dei risultati.

Si riporta di seguito l'analisi ed il diagramma di flusso relativi a tre semplici problemi: risoluzione di una equazione di primo grado, risoluzione di un sistema lineare di due equazioni, verifica a flessione di una sezione in c.a. a doppia armatura

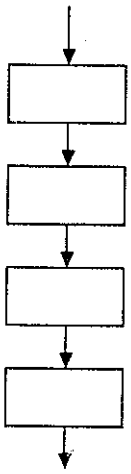


Fig. 8

A) Risoluzione di una equazione di primo grado.

Una equazione di primo grado può essere scritta, in forma simbolica, come:

$$a\,x + b = 0$$

L'analisi del problema è in questo caso immediata, perchè sappiamo che la soluzione è fornita dalla formula:

$$x = -\,b / a$$

Le informazioni che devono essere manipolate, e quindi le variabili che si utilizzano, sono tre e vengono descritte nelle righe seguenti

attributo:	tipo:	nome simbolico:
coefficiente della x	numero reale	A
termine noto	numero reale	B
incognita	numero reale	X

I nomi prescelti, che rispettano le regole formali del linguaggio BASIC che verranno indicate nel capitolo successivo, richiamano immediatamente i simboli utilizzati nell'analisi teorica del problema. Le due variabili A e B costituiscono i dati di partenza e possono quindi essere chiamate "variabili di ingresso". La variabile X rappresenta invece il risultato cercato ed è pertanto una "variabile di uscita"

Il diagramma di flusso (fig. 9) è in questo caso costituito da tre blocchi, che rappresentano le fasi di input, elaborazione ed output Si noti la presenza di un quarto blocco contenente la parola fine, cioè una istruzione che indica al calcolatore che l'esecuzione è terminata

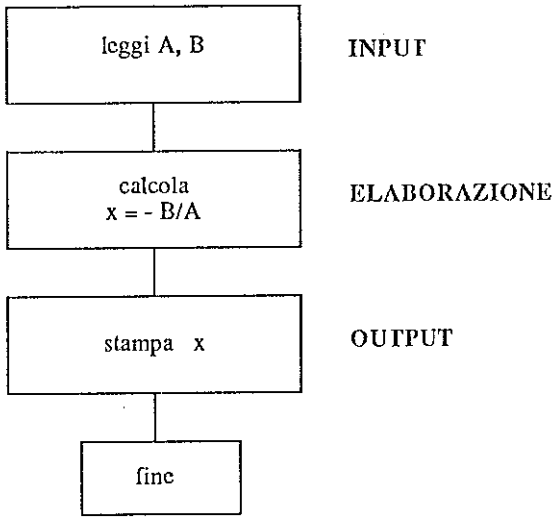


Fig 9

B) Risoluzione di un sistema lineare di equazioni (2 equazioni, 2 incognite)

Un sistema lineare di due equazioni in due incognite può essere scritto, in forma simbolica, come:

$$a\,x + b\,y = c$$
$$d\,x + e\,y = f$$

In questo caso la regola di Kramer ci fornisce la soluzione:

$$x = (c e - f b) / v$$

$$y = (a f - c d) / v$$

dove

$$v = a e - b d$$

Le variabili che si utilizzano nel problema sono pertanto:

attributo:	tipo:	nomi simbolici:
coefficienti delle incognite	numero reale	A B D E
termini noti	numero reale	C F
incognite	numero reale	X Y
v = a e - b d	numero reale	V

Anche in questo caso, le variabili A B C D E F possono essere chiamate "variabili di ingresso" e le X Y "variabili di uscita" La variabile V, che rappresenta una quantità calcolata durante l'esecuzione del programma ma che non deve essere fornita come risultato verrà invece denominata "variabile interna al programma".

Il diagramma di flusso (fig. 10) contiene sequenzialmente un blocco per la fase di input, tre per l'elaborazione ed uno per l'output.

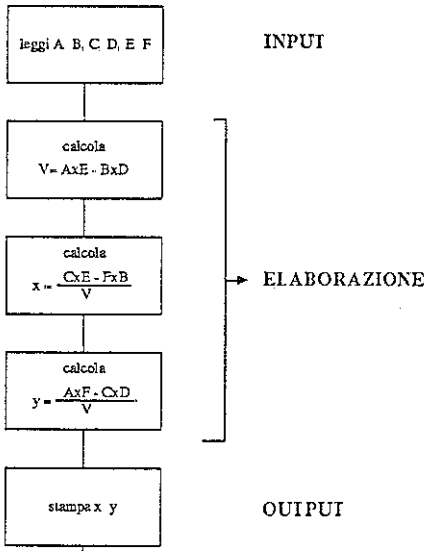


Fig. 10

C) Verifica a flessione - sezione rettangolare in c.a. a doppia armatura.

La geometria della sezione è descritta mediante la base B, il copriferro d, l'altezza utile h, l'area di ferro teso A_f e compresso A'_f ed il rapporto dei moduli elastici dell'acciaio e del calcestruzzo $n=E_f/E_c$. La caratteristica di sollecitazione su essa agente è il momento flettente M.

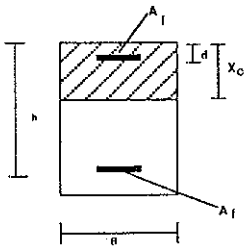


Fig 11

Dalla Tecnica delle Costruzioni conosciamo le formule che forniscono la posizione dell'asse neutro x_c e le tensioni massime in acciaio e calcestruzzo σ_c e σ_f:

$$x_c = \frac{n\,(A_f + A'_f)}{B}\left(-1 + \sqrt{1 + \frac{2\,B\,(A_f\,h + A'_f\,d)}{n\,(A_f + A'_f)^2}}\right)$$

$$\sigma_c = \frac{M}{\frac{B\,x_c}{2}\left(h - \frac{x_c}{3}\right) + \frac{n\,A'_f}{x_c}(x_c - d)(h - d)} \quad ; \quad \sigma_f = n\,\sigma_c\,\frac{h - x_c}{x_c}$$

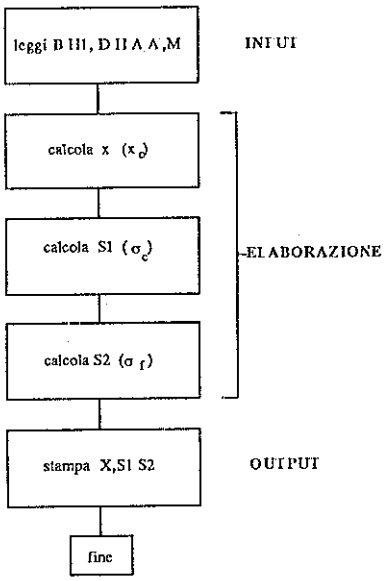


Fig 12

Si utilizzano le variabili di ingresso B, H1, D, N, A, A1, M, e le variabili di uscita X, S1, S2, tutte di tipo numero reale Il nome simbolico scelto mostra immediatamente l'attributo di ciascuna variabile.

Il diagramma di flusso è riportato in figura 12.

3.4 - Alterazione della sequenza.

La sequenza normalmente seguita è quella di elencazione delle istruzioni (nella rappresentazione grafica, dall'alto verso il basso) L'istruzione che impone di eseguire un'operazione diversa da quella immediatamente successiva è detta "salto". Si parla di salto incondizionato quando questa imposizione è assoluta, senza condizioni. Graficamente esso è rappresentato da una linea che porta ad un blocco non sottostante al precedente, cioè che si ricollega ad un altro punto della sequenza (fig 13).

Una diversa alterazione della sequenza è consentito dalle operazioni decisionali Come si è detto esse presentano due (o più) uscite, dipendenti dal realizzarsi, o meno, di una determinata condizione. Si può realizzare in tal modo un salto condizionato, cioè un salto che viene eseguito solo in casi prefissati. Graficamente, si rappresenta il rombo (operazione decisionale) con una uscita che rispetta la sequenza "normale" ed un'altra che indica il salto.

Infine, anche il ricorso ad un sottoprogramma equivale ad una alterazione della sequenza Si intende con sottoprogramma (in inglese *subroutine*) un insieme ben definito di operazioni, con un inizio ed una fine. Utilizzare il sottoprogramma vuol dire abbandonare una sequenza saltando ad eseguire la prima delle sue istruzioni, proseguire con esse fino all'ultima e poi tornare alla istruzione immediatamente successiva nella sequenza originaria. Si noti che, a differenza del semplice salto, si ha in tal caso sempre necessariamente anche un ritorno all'indietro

Un esempio di applicazione dei salti e delle operazioni decisionali è fornito dalla risoluzione di una equazione di secondo grado. Questa può essere scritta, in forma simbolica, come:

$$a\,x^2 + b\,x + c = 0$$

Per essa, come si sa, esistono radici reali solo quando il discriminante $\Delta = b^2 - 4\,ac$ è maggiore o uguale a zero In tal caso le soluzioni sono fornite dalla formula:

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}$$

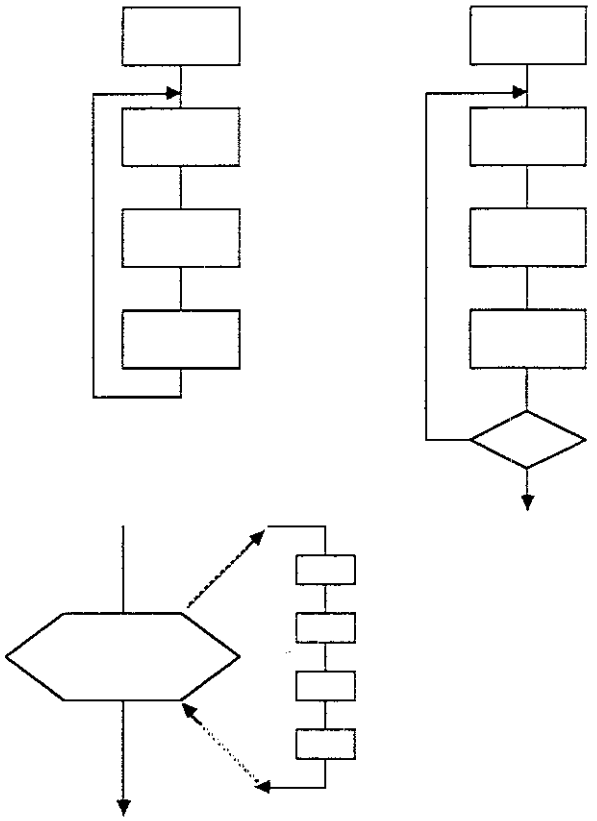


Fig. 13

Per risolvere il problema occorrerà pertanto prima calcolare il valore del discriminante e poi, a seconda che esso sia, o no, maggiore o uguale a zero, applicare la formula per il calcolo delle radici oppure stampare una segnalazione di errore (fig 14).

Si utilizzano le variabili di ingresso A B C (coefficienti dell'equazione), la variabile interna D (discriminante) e le variabili di uscita X1 X2 (incognite del problema) Esse sono tutte del tipo numero reale

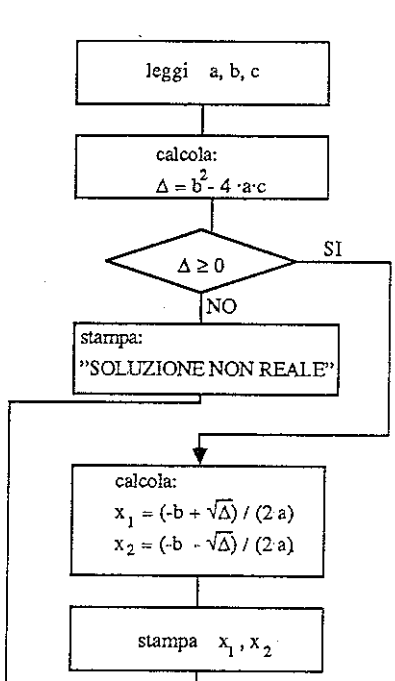


Fig. 14

3.5 - Tecniche di programmazione

L'uso dei salti è spesso molto utile nella realizzazione di programmi, in particolare per minimizzare il numero di istruzioni necessarie o il tempo di esecuzione. Un buon programmatore non deve però preoccuparsi di ciò, almeno nella fase iniziale di impostazione del lavoro. L'ottimizzazione dei tempi e dell'ingombro di memoria deve essere semmai solo la fase conclusiva, di perfezionamento di un prodotto già ben funzionante.

La caratteristica principale di un programma di buona qualità è la sua chiarezza e leggibilità. L'uso eccessivo e non disciplinato di salti contrasta con queste esigenze. Essi rendono il programma "ingarbugliato" (fig. 15) e diventa difficile prevedere tutte le possibili alternative che si vengono a creare durante la esecuzione e verificarle esaurientemente mediante esempi. In caso di errore, è più problematico individuarne la causa o addi-

rittura la parte di programma che lo racchiude Infine, diventa difficile e rischioso modificarne un qualsiasi gruppo di istruzioni, a causa delle possibili conseguenze su parti del programma diverse, ma a questo collegate

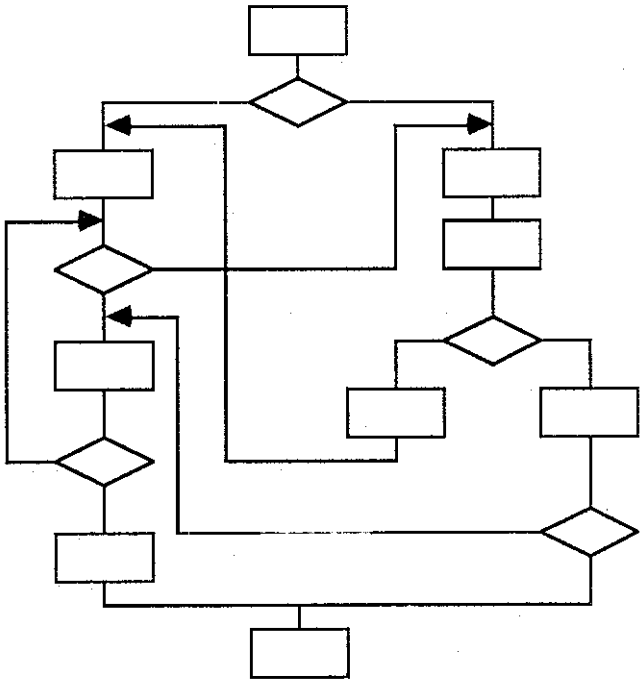


Fig. 15

Per procedere in maniera corretta è quindi necessario seguire rigidamente dei criteri, anche se ciò può apparire limitante.

Innanzitutto, è importante affrontare il problema con successivi gradi di dettaglio, cioè non pensare fin dall'inizio a tutti i particolari, ma cominciare con una impostazione generale su grandi linee ed andare poi man mano definendo ciascun punto in fasi successive.

Per ciascun livello di dettaglio occorre descrivere in maniera chiara la struttura logica, in genere mediante un diagramma di flusso a blocchi. In tale impostazione bisogna rinunciare all'uso dei salti ed utilizzare esclusivamente alcune strutture logiche, tutte caratterizzate dall'avere un solo ingresso ed una sola uscita.

Completata una fase, si passa alla successiva, nella quale ciascuno dei blocchi

precedentemente individuati viene a sua volta analizzato e descritto con gli stessi criteri innanzi esposti.

Questo modo di procedere viene detto "programmazione strutturata", con riferimento all'imposizione di adottare esclusivamente alcune ben definite strutture logiche.

3.6 - Strutture logiche di programmazione.

Le strutture logiche ammesse possono raggrupparsi in tre categorie: strutture sequenziali, strutture condizionali o selettive, strutture di ciclo

3.6.1 - Strutture sequenziali.

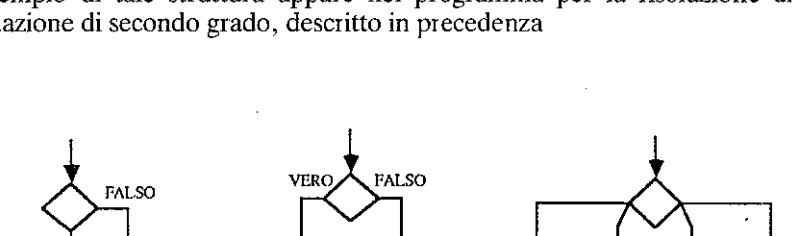
Si intende con tale termine una sequenza continua di blocchi di istruzioni, ovvero di strutture logiche. È la struttura di programmazione più semplice, ed è già stata descritta ed utilizzata negli esempi del paragrafo 3.3.

3.6.2 - Strutture condizionali o selettive.

Sono strutture che consentono di scegliere tra due o più alternative (fig. 16).

La possibilità di eseguire, o no, un blocco di istruzioni in base al verificarsi di una specifica condizione è descritta dalla struttura "IF ... THEN ...", traducibile in italiano con: "SE si verifica la condizione ALLORA esegui il blocco"

La scelta tra due blocchi di istruzioni alternativi è rappresentata con la struttura "IF ... THEN ... ELSE ...", cioè: "SE si verifica la condizione ALLORA esegui il primo blocco ALTRIMENTI esegui il secondo blocco". Un esempio di tale struttura appare nel programma per la risoluzione di un'equazione di secondo grado, descritto in precedenza



IF THEN IF THEN ELSE CASE OF

Fig 16

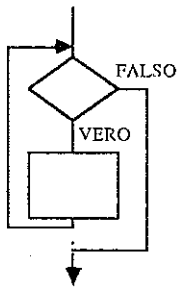
La scelta fra più blocchi di istruzioni è individuata mediante la struttura "CASE .. OF ...". Essa consiste in una espressione, detta selettore, ed una lista di blocchi di istruzioni, ciascuno identificato da un'etichetta che rappresenta un possibile valore del selettore. Il valore corrente del selettore indica così quale blocco eseguire.

Questa struttura viene più spesso utilizzata in una forma semplificata, nella quale il selettore è un numero intero, che può assumere i valori tra 1 ed n. In tal caso essa ha il significato: "esegui il CASO k TRA blocco (caso) 1, blocco 2, .. blocco n".

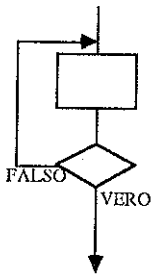
3.6.3 - Strutture di ciclo.

Si ha un ciclo quando un blocco di istruzioni viene eseguito più volte. Il ciclo viene detto iterativo quando non è definito a priori quante volte ripetere l'esecuzione. Esempio di ciò è la risoluzione iterativa di un telaio, nella quale non si conosce, all'inizio, quante operazioni effettuare per raggiungere la convergenza. Quando invece il numero di volte è noto a priori il ciclo è detto ripetitivo o enumerativo.

Sono strutture di ciclo tipicamente iterative la "WHILE .. DO ..." e la "REPEAT .. UNTIL ..." (fig. 17).



WHILE .. DO



REPEAT .. UNTIL

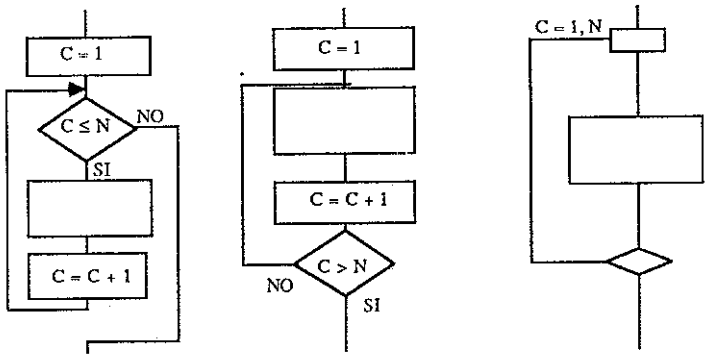
Fig. 17

La prima è traducibile con "FINCHÈ è vera la condizione ESEGUI il blocco di istruzioni". In essa il controllo sul verificarsi della condizione è premesso all'esecuzione del blocco. Se fin dall'inizio la condizione non è verificata, il blocco di istruzioni non viene mai eseguito.

La seconda equivale a "RIPETI il blocco di istruzioni FINCHÈ NON diventa vera la condizione". In essa quindi, al contrario della precedente, il ciclo viene interrotto quando la condizione diventa vera. Inoltre il controllo è posto dopo il blocco di istruzioni, che viene pertanto eseguito sempre almeno una volta, anche se la condizione è vera fin dall'inizio.

Per evidenziare la differenza tra le due strutture, si pensi di applicarle alla risoluzione di un telaio a nodi fissi mediante operazioni di equilibramento dei nodi. Utilizzando la prima si direbbe "FINCHÈ i nodi sono squilibrati ESEGUI le operazioni di nodo". Con la seconda, invece, "RIPETI le operazioni di nodo FINCHÈ NON si è raggiunto l'equilibrio nei nodi"

I cicli ripetitivi richiedono l'esistenza di un contatore, cioè di una variabile interna utilizzata per contare quante volte viene ripetuta l'esecuzione del blocco di istruzioni. Il contatore viene inizializzato, cioè gli si assegna un valore di partenza, prima di cominciare le ripetizioni. Dopo ciascuna esecuzione del blocco di istruzioni esso viene incrementato di una quantità costante (*passo di incremento*, in inglese *step*). La ripetizione termina quando il contatore supera un valore finale imposto.



WHILE DO REPEAT UNTIL FOR DO

CICLO RIPETITIVO variabili: C contatore
N numero di ripetizioni richiesto

Fig 18

È possibile realizzare cicli ripetitivi sia con la struttura "WHILE .. DO ..." che

con la "REPEAT ... UNTIL ...", utilizzando come condizione il confronto tra valore corrente e valore finale del contatore. I corrispondenti diagrammi di flusso sono mostrati in figura 18. Esemplificativamente si è assunto un valore iniziale ed un passo di incremento unitari; il valore finale N indica in tal caso il numero di ripetizioni richieste. Si noti che, mentre per N>0 entrambi gli schemi sono equivalenti, per N=0 solo la struttura "WHILE ... DO ..." consente effettivamente di eseguire zero volte (cioè non eseguire affatto) il blocco di istruzioni.

Per maggiore sintesi e chiarezza espressiva, si utilizza però usualmente per i cicli ripetitivi una struttura specifica, la "FOR ... DO ...". Essa equivale a "PER il contatore che va da un valore iniziale ad un valore finale con un passo definito ESEGUI il blocco di istruzioni". Nella sua rappresentazione grafica, l'inizio e la fine del ciclo sono indicati schematicamente con un piccolo rettangolo ed un piccolo rombo che stanno a ricordare le istruzioni di incremento e di confronto. La linea che torna dalla fine all'inizio indica il ripetersi dell'esecuzione del blocco di istruzioni. A fianco ad essa è indicato il contatore ed i tre valori caratteristici del ciclo (iniziale, finale e passo di incremento, quest'ultimo in genere esplicitato solo se diverso dall'unità).

CICLO = Sequenza di istruzioni ripetuta più volte

CICLO ITERATIVO	Quando il numero di volte non è noto a priori
CICLO RIPETITIVO	Quando il numero di volte è fissato a priori
CONTATORE	Variabile interna usata per contare quante volte viene ripetuto il blocco di istruzioni.

Si riportano di seguito due esempi di problemi che richiedono strutture di ciclo: tabellazione di una funzione e calcolo del punteggio base per la laurea. In entrambi i casi il ciclo è ripetitivo, ma si sono utilizzate rispettivamente la struttura "WHILE ... DO ..." e la "FOR ... DO ..."

A) Tabellazione di una funzione

Si indica genericamente con $y=f(x)$ l'espressione analitica di una funzione della sola variabile x, rappresentata graficamente in figura 19. Si ipotizza inoltre che la funzione sia definita per tutti i valori di x compresi nell'intervallo a–b. Tabellare la funzione vuol dire calcolare e stampare le coppie di valori $x,f(x)$ facendo variare la x da a fino a b con un passo dx.

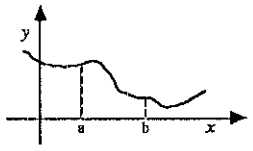


Fig. 19

L'espressione analitica della funzione deve ovviamente essere definita, ad esempio $f(x)=-7x^3+\sqrt{(x^2+10)}$ Sono invece variabili di ingresso del problema, cioè dati da assegnare, gli estremi dell'intervallo ed il passo di incremento, per i quali si possono utilizzare i nomi A B D Sono variabili di uscita le quantità x ed $f(x)$, che si indicano rispettivamente con X e Y

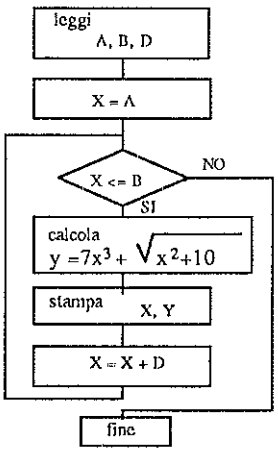


Fig 20

Il ciclo di calcolo e stampa può essere realizzato mediante la struttura "WHILE .. DO . ." (fig 20) Occorre assegnare inizialmente ad X il valore A, e poi, *finchè* X è minore o uguale a B, *eseguire* il calcolo di Y, la stampa dei valori di X ed Y e l'incremento della variabile X della quantità D

B) Calcolo del punteggio base per la laurea.

All'esame di laurea il voto è assegnato incrementando, o in casi eccezionali decrementando, un punteggio base determinato trasformando in base 110 la media dei voti dei singoli esami (che sono assegnati in trentesimi). Per valutare questo punteggio occorre quindi calcolare la media dei voti (somma dei voti diviso il numero degli esami) e moltiplicarla per 110/30 (fig. 21).

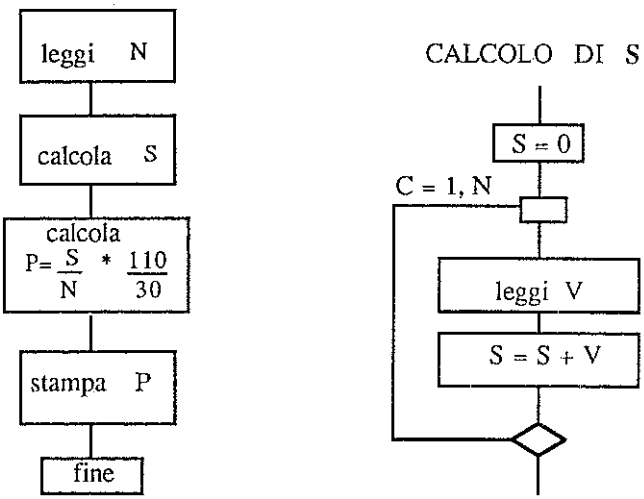


Fig. 21

La somma è calcolata aggiungendo ciclicamente il valore di ciascun voto V ad una variabile interna S, inizialmente azzerata. Il ciclo è ripetuto tante volte quanti sono gli esami (variabile N) ed è controllato dal contatore C. La variabile d'uscita, che contiene il punteggio cercato, è denominata P.

3.7 - Variabili di comodo.

Negli esempi finora illustrati si sono utilizzate variabili dette "di ingresso", "di uscita" ed "interne", tutte contenenti informazioni strettamente legate al problema in esame. È però frequente l'uso di variabili "di comodo" che assumono valori convenzionali fissati dal programmatore Esse hanno soprattutto funzione di controllo, poichè servono per ricordare che si sono verificati determinati eventi nel corso dell'elaborazione e consentire in base a ciò la scelta, in un momento successivo, tra diverse sequenze di istruzioni. La figura 22 ne mostra schematicamente una possibile utilizzazione

Si riportano di seguito due esempi nei quali si è fatto ricorso a variabili di comodo Il primo, determinazione del punto di nullo di una funzione $f(x)$, ovvero soluzione dell'equazione $f(x)=0$, è sviluppato attraverso livelli successivi, fino al grado di massimo dettaglio. Per il secondo, che propone

un video-game, il "tiro al piattello", si è riportata solo la fase generale di impostazione

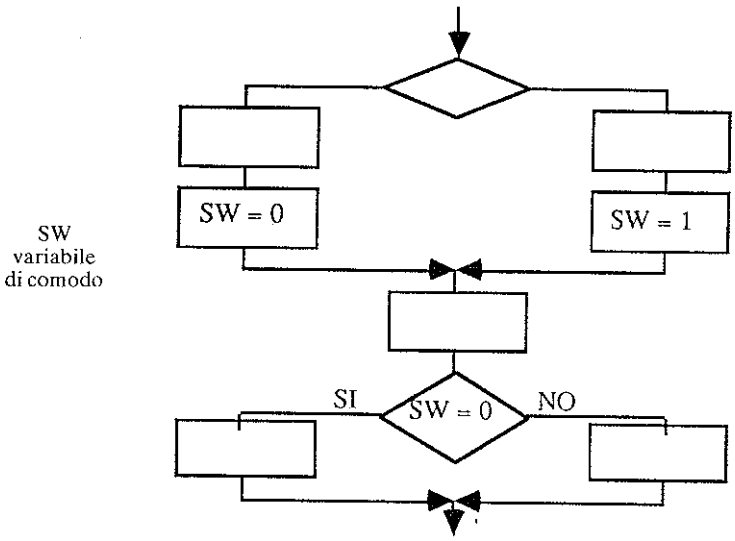


Fig 22

A) Punto di nullo di una funzione.

Si indica genericamente con $y=f_{(x)}$ l'espressione analitica di una funzione della sola variabile x, rappresentata graficamente in figura 23. Si ipotizza inoltre che la funzione sia definita e continua per tutti i valori di x compresi nell'intervallo chiuso a–b. Se $f_{(a)}$ ed $f_{(b)}$ hanno segno opposto, la funzione ha nell'intervallo almeno un punto di nullo, cioè esiste almeno un valore di x per il quale $f_{(x)}=0$

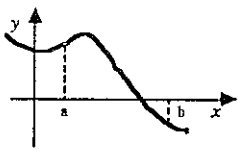


Fig. 23

Tra i numerosi metodi iterativi che consentono di individuare tale punto di nullo, si prende in esame quello detto "metodo del dimezzamento". Esso procede restringendo via via l'intervallo entro cui cercare il punto, con successive fasi di dimezzamento (fig. 24). In una generica fase, occorre determinare il valore di $f_{(x)}$ nel punto medio dell'intervallo corrente. In base a tale valore si può definire un nuovo intervallo, che sarà

la metà destra (o la sinistra) di quello precedente Il procedimento termina quando l'ampiezza dell'intervallo è sufficientemente ridotta (minore di un valore ε_1 prefissato), oppure se nel punto medio di un intervallo la funzione assume un valore sufficientemente prossimo a zero (in valore assoluto, minore di un ε_2 prefissato).

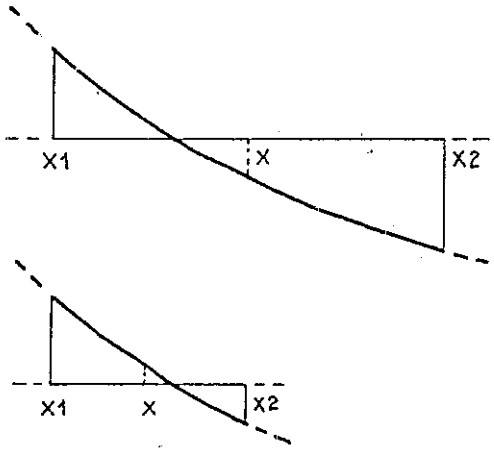


Fig 24

Il problema si presenta un pò più complesso, rispetto agli esempi finora trattati. È bene quindi affrontarlo logicamente con successivi gradi di dettaglio

Si parte da una analisi molto generale (fig 25), i cui blocchi contengono istruzioni inequivocabili dal punto di vista logico ma non ancora immediatamente operative. La fase iniziale prevede la lettura dei dati, la definizione dell'intervallo di partenza e un controllo dei valori di $f_{(a)}$ ed $f_{(b)}$ per accertarsi che siano di segno opposto. Il ciclo è realizzato mediante una struttura "WHILE . DO .. " che impone sostanzialmente: *finchè* occorre ancora cercare il punto di nullo *esegui* il dimezzamento dell'intervallo. La fase finale è costituita dalla stampa del risultato trovato.

Data la generalità di questa prima analisi, non è stato necessario fare esplicitamente riferimento alle informazioni da manipolare e definire i nomi delle variabili. Ciò diventa indispensabile procedendo nei successivi livelli di dettaglio. Si utilizzerà pertanto la seguente simbologia:

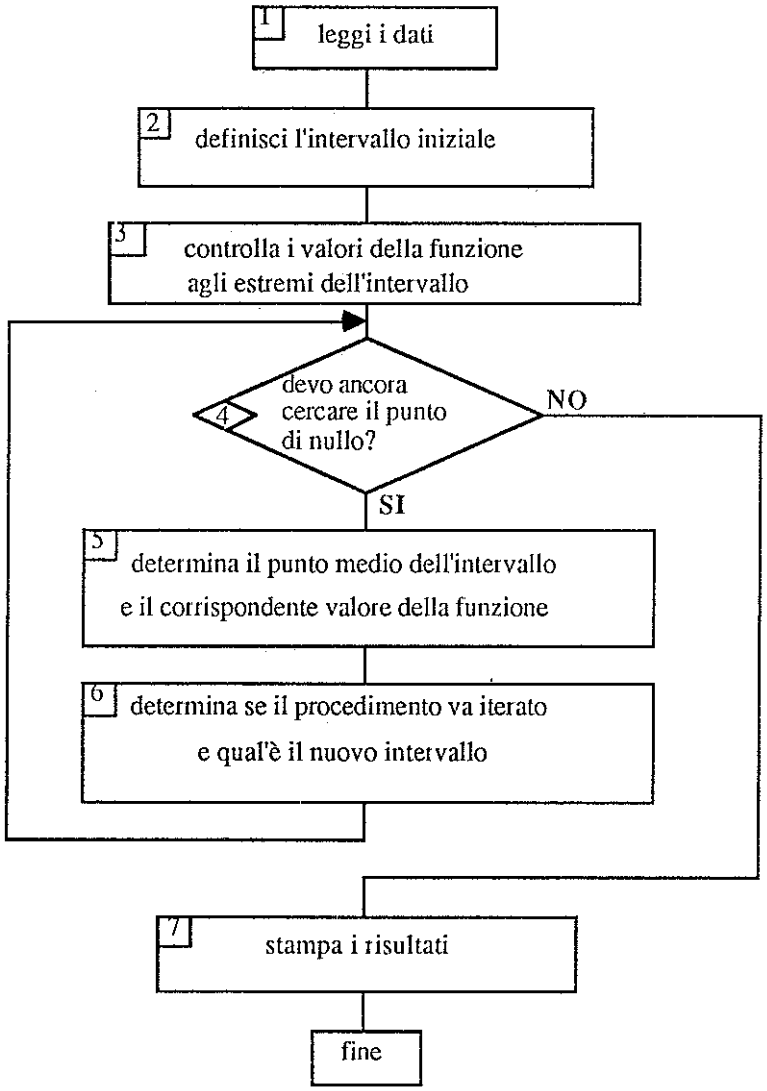

1
leggi i dati
2
definisci l'intervallo iniziale
3
controlla i valori della funzione
agli estremi dell'intervallo
4
devo ancora
cercare il punto
di nullo?
NO
SI
5
determina il punto medio dell'intervallo
e il corrispondente valore della funzione
6
determina se il procedimento va iterato
e qual'è il nuovo intervallo
7
stampa i risultati
fine

Fig 25

A B	estremi dell'intervallo di partenza;
X1 X2	estremi dell'intervallo generico;
Y1 Y2	valori assunti dalla funzione in corrispondenza ad X1 ed X2;
X	punto medio dell'intervallo X1–X2 (che al termine delle iterazioni rappresenta il valore cercato);
Y	valore assunto dalla funzione in corrispondenza ad X;
E1	approssimazione ammissibile per $f_{(x)}$;
E2	ampiezza minima dell'intervallo.

Nel procedere allo sviluppo del programma, ci si rende conto che in più momenti è utile conoscere o poter indicare a che punto sia la ricerca. Si può usare a tal fine una variabile di comodo, indicata con la lettera R, assumendo convenzionalmente i seguenti significati:

R = 0	occorre ancora cercare il punto di nullo;
R = 1	il punto di nullo è stato trovato;
R = 2	il punto di nullo non esiste ($f_{(a)}$ ed $f_{(b)}$ hanno lo stesso segno)

Le figure 26 e 27 mostrano, per ognuno dei blocchi precedenti, lo sviluppo in dettaglio. In alcuni casi si tratta, banalmente, di esplicitare le indicazioni già fornite, inserendovi i nomi delle variabili. In altri, invece, si evidenzia l'approfondimento dell'analisi. Ad esempio il blocco 3, che controlla i valori della funzione agli estremi dell'intervallo, presenta una struttura "CASE . . OF . . " che prevede quattro possibilità: che la funzione si annulli all'estremo a o b oppure che $f_{(a)}$ ed $f_{(b)}$ siano concordi o discordi. Il blocco 6, che determina se il procedimento va iterato e qual'è il nuovo intervallo, presenta una struttura "IF . . THEN . . ELSE . . ", che a sua volta racchiude altre strutture. Si noti che in questo caso è necessario un ulteriore approfondimento dell'analisi per descrivere i blocchi 6a, 6b e 6c.

Un esame complessivo evidenzia l'utilità della variabile R. Il suo valore viene inizialmente definito nel blocco 3, in base ai valori della funzione in a e b. Viene utilizzata come variabile di controllo per il ciclo "WHILE . . DO . . " (blocco 4). È usata nel blocco 6, per indicare che $f_{(x)}$ è quasi nulla o che l'intervallo è molto ristretto. Infine, nel blocco 7, consente di stampare il valore trovato o una segnalazione di errore, a seconda della situazione riscontrata in precedenza.

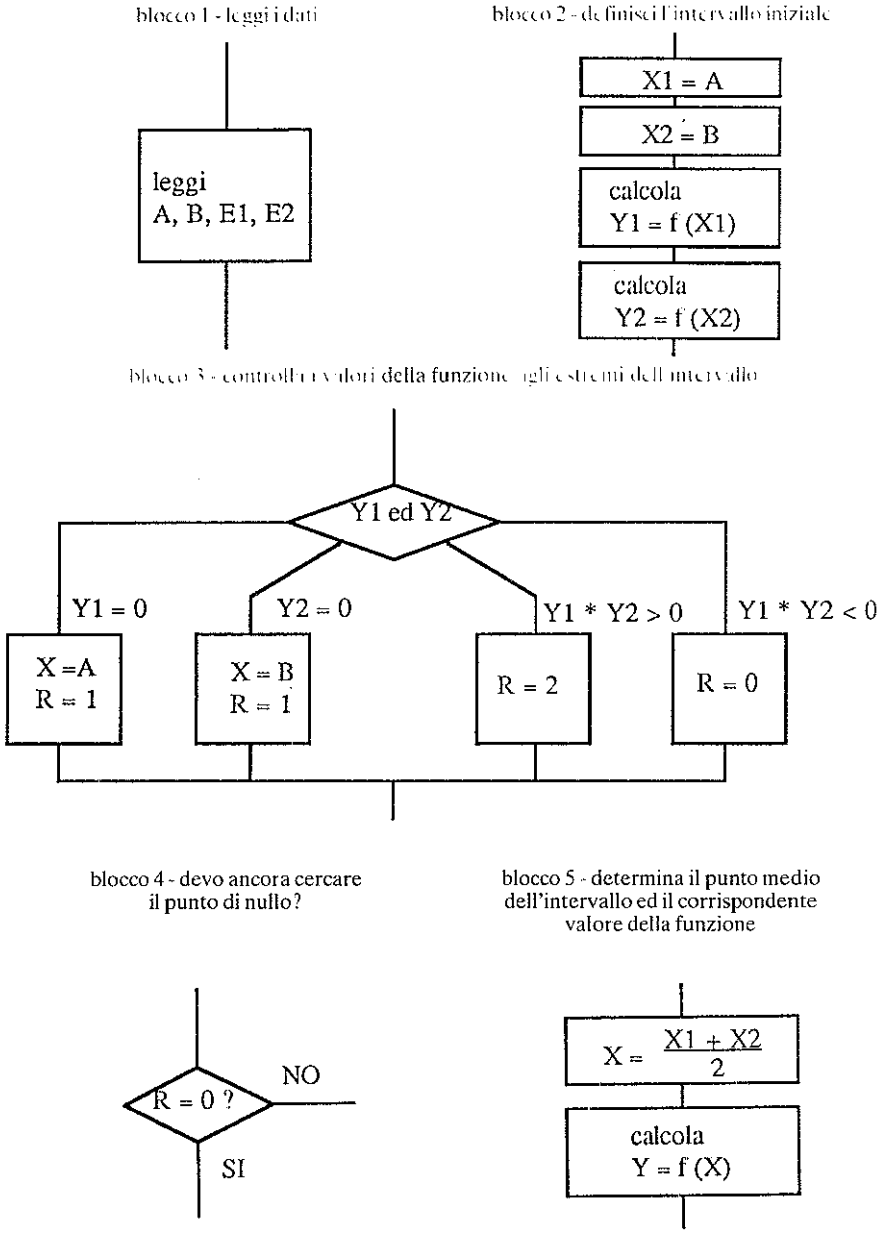
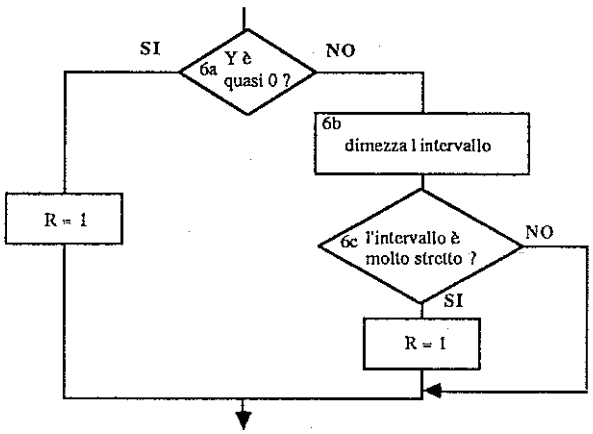
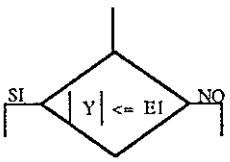

blocco 1 - leggi i dati
leggi
A, B, E1, E2
blocco 2 - definisci l'intervallo iniziale
X1 = A
X2 = B
calcola
Y1 = f (X1)
calcola
Y2 = f (X2)
blocco 3 - controlla i valori della funzione agli estremi dell'intervallo
Y1 ed Y2
Y1 = 0
Y2 = 0
Y1 * Y2 > 0
Y1 * Y2 < 0
X =A
R = 1
X = B
R = 1
R = 2
R = 0
blocco 4 - devo ancora cercare il punto di nullo?
R = 0 ?
NO
SI
blocco 5 - determina il punto medio dell'intervallo ed il corrispondente valore della funzione
X = X1 + X2 / 2
calcola
Y = f (X)

Fig. 26

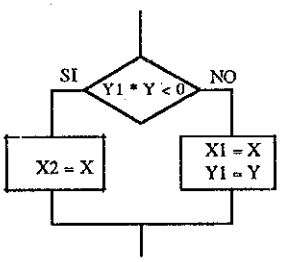
blocco 6 - determina se il procedimento va iterato e qual'è il nuovo intervallo



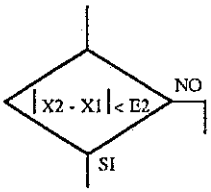
blocco 6a
Y è quasi 0?



blocco 6b
dimezza l'intervallo



blocco 6c
l'intervallo è molto stretto?



blocco 7 - stampa i risultati

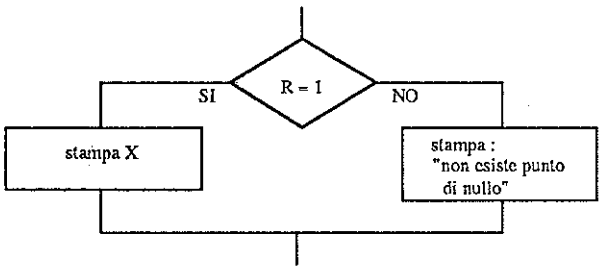


Fig 27

B) Tiro al piattello.

Si pensi di realizzare un video-game che simuli una gara di tiro al piattello Sullo schermo deve comparire il fucile del concorrente, che può essere fatto ruotare per prendere la mira prima di sparare il colpo. Obiettivo del giocatore è colpire i piattelli che attraversano lo schermo, uno alla volta, con traiettorie sempre diverse (fig 28).

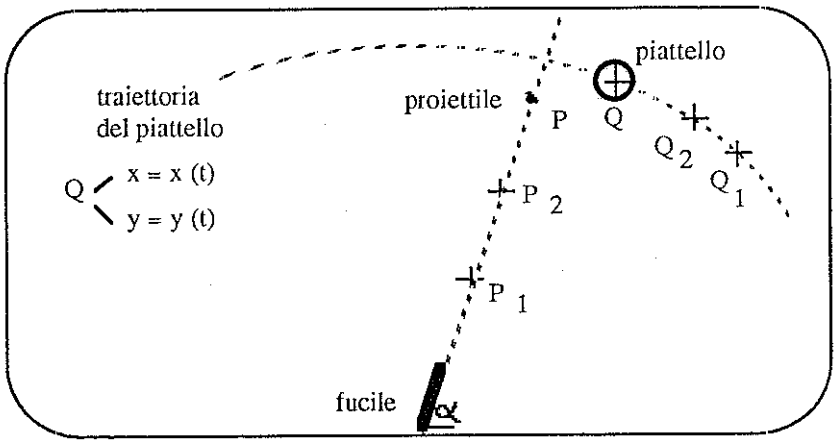


Fig. 28

Si prende in esame solo una fase del gioco, quella in cui il proiettile sparato ed il piattello si muovono nel video (con traiettoria rettilinea il primo, curvilinea il secondo), finchè il piattello viene colpito, oppure l'uno o l'altro escono dallo schermo

Il diagramma di flusso di figura 29 mostra il livello più generale di analisi. Il tempo viene diviso in intervalli dT, sufficientemente piccoli. In ciascuno di essi, il proiettile compie il percorso P1–P2 mentre il piattello compie il percorso Q1–Q2 (assimilabile per semplicità ad un segmento). Si calcolano pertanto tali posizioni e si determina se esiste un istante nel quale il proiettile colpisce il bersaglio, cioè in cui la distanza tra i punti P e Q in movimento diventa inferiore al raggio del piattello. Se ciò avviene, si controlla che il punto di impatto sia interno alla zona visualizzata nello schermo. In caso contrario, una analoga verifica è effettuata sulla posizione dei due elementi all'istante finale dell'intervallo. A seconda del risultato di questi controlli, si farà vedere il piattello colpito che va in mille pezzi, oppure proiettile e bersaglio nella nuova posizione raggiunta

Risulta vantaggioso memorizzare queste informazioni in due variabili di comodo, C ed F, che assumono convenzionalmente i seguenti significati:

C = 0 il bersaglio non è stato colpito (o è stato colpito, ma al di fuori dello schermo);
C = 1 il bersaglio è stato colpito;
F = 0 la fase non è terminata;
F = 1 la fase è terminata (il bersaglio è stato colpito, oppure piattello o proiettile sono usciti dallo schermo)

Le operazioni innanzi descritte sono racchiuse in un ciclo iterativo, realizzato con una struttura "REPEAT . UNTIL .", che impone di *ripetere* l'incremento del tempo e l'esame delle posizioni mutue di proiettile e piattello, *finchè non* si è giunti alla fine della fase (individuata dal valore 1 della variabile F)

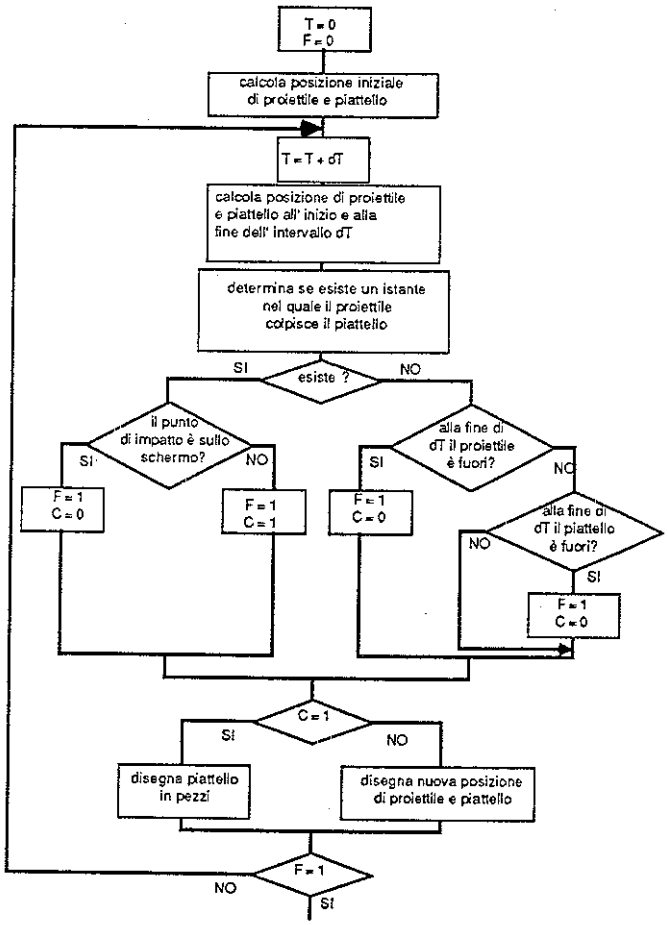


Fig. 29

3.8 - Variabili con indice

Come si è già evidenziato, ogni informazione è caratterizzata da tipo, attributo e valore. Il registro, o il gruppo di registri consecutivi, che ne contiene il valore codificato è identificato, nei linguaggi evoluti, da un nome simbolico detto variabile.

Spesso occorre trattare con un insieme di informazioni omogenee, cioè dotate dello stesso tipo ed attributo ma con valori diversi. Un tale insieme è in genere ordinato, ed ogni informazione ha in esso una posizione ben precisa. In questo caso è preferibile conservarne i valori sequenzialmente, in gruppi consecutivi di registri di memoria. Il gruppo di registri che contiene il singolo valore viene individuato in base alla sua posizione nell'insieme, mediante un "indice". L'intero insieme di registri che contiene tutti i valori è individuato da un unico nome simbolico. Esso è indicato col termine "array" o "variabile con indice", per distinguerlo da quello di una informazione singola, che verrà d'ora in poi denominato "variabile semplice"

In alcuni insiemi di informazioni può essere preferibile individuare il singolo valore mediante più indici. Un caso tipico è costituito dagli elementi di una matrice, ciascuno dei quali è identificabile mediante il numero d'ordine della riga e della colonna cui appartiene. Un array viene allora detto "monodimensionale" quando per localizzarne gli elementi si utilizza un solo indice, "pluridimensionale" quando se ne usano più di uno. Questa distinzione non ha ovviamente riscontro nella struttura del calcolatore. I registri di memoria costituiscono un insieme sequenziale e sono quindi individuati da un unico numero d'ordine. È compito del programma che implementa il linguaggio in uso mettere in relazione il valore degli indici alla posizione effettiva dei registri che contengono l'informazione.

Poichè i registri di memoria in cui sono conservati i valori devono essere consecutivi, è necessario che al calcolatore venga segnalato il numero massimo di valori previsto. Tale operazione viene detta "dimensionamento" dell'array e deve sempre essere effettuata prima di utilizzarne un qualunque elemento.

INSIEME DI INFORMAZIONI OMOGENEE

caratterizzate da	unico TIPO unico ATTRIBUTO numerosi VALORI
L'insieme dei VALORI delle informazioni:	- è conservato ordinatamente in più gruppi di registri consecutivi. - è codificato con un codice che dipende dal tipo.

Nei LINGUAGGI EVOLUTI:	- l'insieme globale dei registri è individuato da un unico NOME SIMBOLICO - il gruppo dei registri che contiene il singolo valore viene individuato in base alla sua posizione nell'insieme, mediante un INDICE.
ARRAY (o VARIABILE CON INDICE)	nome simbolico che individua i registri di memoria che contengono un insieme ordinato di valori di informazioni omogenee
INDICE	individua la posizione di un singolo valore in un insieme di valori di informazioni omogenee.

ARRAY **MONODIMENSIONALE** – unico indice

ARRAY **PLURIDIMENSIONALE** – più indici

Quando si vuole utilizzare un array, occorre segnalare preliminarmente al calcolatore il numero di informazioni che si vogliono conservare nell'array (DIMENSIONARE L'ARRAY)

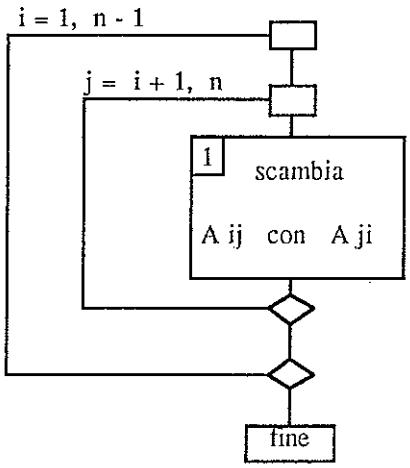
Si riportano di seguito due esempi di problemi che richiedono l'uso di variabili con indice: trasposizione di una matrice quadrata e prodotto di due matrici.

A) Trasposizione di una matrice quadrata.

Sia A una matrice quadrata di ordine n, cioè con n righe ed n colonne. Come già evidenziato, essa può essere considerata un array bidimensionale, il cui generico elemento può essere indicato con A_{ij}, dove i e j sono numeri interi compresi tra 1 ed n. La sua trasposta si ottiene scambiando tra loro gli elementi simmetrici rispetto alla sua diagonale, cioè invertendo A_{ij} con A_{ji}. Ovviamente, gli elementi diagonali rimangono invariati.

Se si vuole effettuare la trasposizione memorizzando la trasposta al posto della matrice di partenza, ciò può essere realizzato con un doppio ciclo (fig. 30). Il primo passa in rassegna tutte le righe, dalla prima alla penultima (l'indice i varia da 1 a n−1). Il secondo individua, per la riga generica i, tutti gli elementi posti alla destra della diagonale (A_{ij}, con l'indice j che varia da i+1 ad n) e ne scambia il valore con gli elementi simmetrici (A_{ji}).

In un secondo livello di dettaglio è esaminata l'operazione di scambio, che richiede una variabile interna V "di appoggio" per conservare temporaneamente il valore di A_{ij}, per evitare che venga perso quando si memorizza in A_{ij} il valore contenuto in A_{ji}.



blocco 1 – scambia A_{ij} con A_{ji}

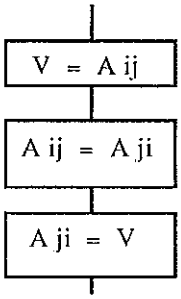


Fig 30

B) Prodotto di due matrici.

Sia A una matrice di m righe ed n colonne e B una matrice di n righe e p colonne. Il prodotto matricale A × B da come risultato una matrice C di m righe e p colonne, il cui generico elemento C_{ij} è ottenuto come somma dei prodotti degli elementi della riga i di A per i corrispondenti della colonna j di B:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

Per valutare C occorre un doppio ciclo (fig. 31) che passi in rassegna tutti gli elementi (facendo variare l'indice i da 1 a m e l'indice j da 1 a p). In un secondo livello di dettaglio è esaminato il calcolo del valore di C_{ij}. Esso viene inizialmente azzerato, e poi incrementato della quantità $A_{ik}\ B_{kj}$ all'interno di un ciclo che passa in rassegna tutti gli n elementi della riga i di A e della colonna j di B (cioè con l'indice k che varia da 1 a n).

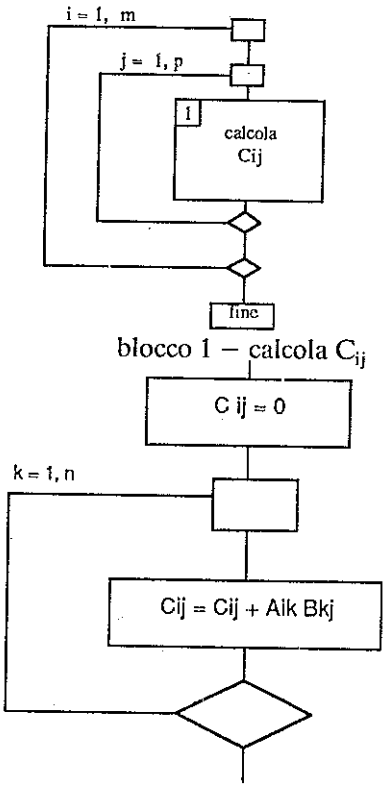


Fig. 31

CAPITOLO QUARTO

IL LINGUAGGIO BASIC

4.1 - Introduzione

Numerosi sono i linguaggi sviluppati per i calcolatori. Ognuno di essi ha regole sintattiche proprie ed implementa una particolare serie di istruzioni, cioè è dotato di un insieme di "parole chiave" ciascuna delle quali indica al calcolatore una sequenza di operazioni elementari da effettuare.

Per le applicazione scientifiche su mini computer e su main-frame il linguaggio più usato è il FORTRAN (da *FORmula TRANslation*, cioè traduzione di formule), sviluppato nel 1954 dall'IBM e successivamente più volte migliorato. Suo scopo principale è quello di consentire la scrittura di formule comunque complesse e quindi una agevole esecuzione di elaborazioni numeriche. È pure importante, anche se meno utilizzato, il PASCAL, caratterizzato soprattutto dalla presenza di istruzioni che consentono un uso naturale delle tecniche di programmazione strutturata. Per le applicazioni commerciali è invece adoperato il COBOL (da *COmmon Business Oriented Language*, cioè linguaggio orientato ai lavori commerciali), particolarmente indirizzato alla gestione di grosse quantità di dati, alfabetici o numerici (ad esempio nomi, cognomi, date).

Sui micro computers è adoperato quasi esclusivamente il BASIC. Esso è stato sviluppato per la prima volta nel 1963 con l'obiettivo di fornire un linguaggio semplice ma generale, adatto alla gestione di programmi interattivi. Il suo nome deriva da *Beginner's All-purpose Symbolic Instruction Code*, cioè linguaggio simbolico di uso generale per principianti.

Da allora esso ha subito successive rielaborazioni da parte delle case costruttrici di micro computers, senza giungere ancora, a differenza degli altri linguaggi citati, ad una piena standardizzazione. Esistono quindi oggi numerose versioni del BASIC, che presentano differenze spesso anche sostanziali. L'American National Standard Institute (ANSI) ha codificato i requisiti minimi che un linguaggio deve soddisfare per essere definito propriamente "BASIC". Questo minimo comune verrà in seguito indicato come *BASIC ANSI* o *BASIC standard*. Operare attenendosi strettamente ad esso è però molto limitante. Si evidenzieranno quindi alcune caratteristiche di due linguaggi BASIC più potenti. Il primo è il *Technical BASIC* dei calcolatori della serie 80 della Hewlett-Packard (che si indicherà sinteticamente con BASIC HP). Il secondo è il BASIC sviluppato dalla Microsoft (*GWBASIC*), adottato dai calcolatori che utilizzano il sistema operativo MS–DOS (Olivetti, IBM, HP Vectra, ecc.), che è attualmente

il linguaggio più diffuso tra i personal computer. Nella scrittura di programmi non si utilizzeranno a pieno le potenzialità di questi linguaggi, ma ci si atterrà ad uno standard tale da facilitare la compatibilità anche con altri BASIC meno potenti.

4.2 - Individuazione della sequenza.

Si è definito "programma" una sequenza di istruzioni codificate in un linguaggio. È quindi di fondamentale importanza definire una regola generale per individuare operativamente tale sequenza, cioè l'ordine di esecuzione.

Un programma in BASIC è costituito da un insieme di "linee di programma". Il BASIC standard prevede che ciascuna linea contenga una sola istruzione. Sia il BASIC HP che il GWBASIC consentono invece la presenza di più istruzioni per linea, separate rispettivamente dal simbolo ` @ ¨ e dal simbolo ":".

Ogni linea di programma è contrassegnata con un numero intero positivo. La regola generale prevede che esse vengano eseguite sequenzialmente, da quella col numero più basso a quella col numero più alto. Questa impostazione è particolarmente consona alla natura interattiva del personal. L'ordine di esecuzione è infatti svincolato dall'ordine di immissione delle istruzioni. In un qualsiasi momento è possibile inserire una nuova linea di programma tra due già esistenti, contraddistinte da numeri non consecutivi, semplicemente assegnando ad essa un numero compreso tra questi. Proprio per consentire successivi inserimenti è opportuno lasciare un intervallo nella numerazione di linee di programma consecutive. Si usano quindi in genere numeri che vanno di 10 in 10.

Il massimo numero che può essere assegnato ad una linea di programma dipende dal linguaggio che si utilizza. Ad esempio, nel GWBASIC esso è pari a 65529, cioè quasi 2^{16}.

4.3 - Istruzioni di commento.

Per migliorare la leggibilità di un programma è possibile inserire in esso delle frasi di commento. Il BASIC standard utilizza a tal fine la istruzione **"REM"** (dall'inglese *remark*, cioè "nota"). Il testo che segue la parola "REM" non viene preso in considerazione dal calcolatore in fase di esecuzione. Un esempio di commento è fornito dalla prima tra le linee di programma che seguono

```
10 REM INGRESSO DATI – si leggono i valori di A, B, C
20 READ A,B,C
```

Il BASIC HP ed il GWBASIC consentono di usare al posto della parola "REM" rispettivamente un punto esclamativo "!" e un apice "'". Questo

simbolo può essere posto anche in coda ad una istruzione operativa. Si può quindi avere, ad esempio:

```
20 READ A,B,C ! si leggono i valori di A  B  C
```

Un moderato uso dei commenti è indubbiamente positivo, perchè consente di individuare con facilità le diverse parti di un programma.

Come caso limite, i commenti potrebbero essere tanto estesi da costituire, almeno in parte, la documentazione del programma. Ciò può essere accettato per un programma a finalità didattiche; è invece sconsigliabile in un programma operativo, a causa dell'ingombro di memoria che essi comporterebbero.

4.4 - Costanti e variabili.

Il BASIC ammette più tipi di informazioni: informazioni numeriche, cioè numeri interi o reali, ed informazioni alfanumeriche.

Un numero intero è un insieme di più cifre, eventualmente precedute dal segno I valori ammessi costituiscono un insieme non molto ampio (da −99999 a +99999 nel BASIC HP, da −32768 a +32767 nel GWBASIC).

Per numero reale si intende più propriamente un numero decimale, cioè un insieme di segno, cifre ed il punto decimale, eventualmente seguiti da un esponente, che indica una potenza di dieci per la quale il numero è moltiplicato (per esempio, −5.32537E+3 equivale a −5325.37). Il numero massimo di cifre significative ed il campo di variazione dell'esponente è dipendente dal linguaggio (nel BASIC HP le cifre significative ammesse sono 12, e l'esponente può andare da −499 a +499, nel GWBASIC le cifre sono 7 e l'esponente da −38 a +38). Nei BASIC più potenti è possibile in alternativa la memorizzazione con un numero di cifre significative differente (precisione ridotta, o SHORT, con solo 5 cifre significative per il BASIC HP; doppia precisione con 16 cifre significative per il GWBASIC)

Una informazione alfanumerica è costituita da un insieme di caratteri (lettere, numeri ed altri simboli), che vengono rappresentati racchiusi tra virgolette (ad esempio "29 Novembre 1984", "Antonio", "−5325.37"). Per indicare un valore alfanumerico si usa spesso il termine stringa (dall'inglese *string*). Nel BASIC HP la lunghezza massima (cioè il numero massimo di caratteri) di una stringa è normalmente assunta pari a 18 caratteri, ma può essere estesa fino a 65530; nel GWBASIC è invece pari a 255 caratteri

Si intende per costante una informazione il cui valore è definito esplicitamente nel testo del programma e non è quindi modificabile durante l'esecuzione

Si indica invece col termine variabile un nome simbolico che individua il gruppo di registri che contengono il valore di una informazione (variabile

semplice) o più informazioni omogenee (array).

Ciascun linguaggio impone delle regole da rispettare nella scelta dei nomi delle variabili. Nel BASIC ANSI il nome di una variabile numerica può essere costituito da una lettera, oppure da una lettera ed una cifra. Sono ad esempio nomi ammissibili A e B7. Il nome di una variabile alfanumerica può essere costituito da una lettera, o una lettera ed una cifra, seguiti dal simbolo dollaro "$" (ad esempio A$ e B7$).

Le regole indicate valgono sia per le variabili semplici che per gli array. Ovviamente in questo secondo caso per individuarne uno specifico elemento il nome deve essere seguito dall'indice, racchiuso tra parentesi. L'indice può essere costituito anche da una espressione numerica. Sono quindi elementi di un array A(5), B7(5*K−12), A$(J,2). Nell'ambito di un programma è possibile utilizzare lo stesso nome per una variabile semplice e per un array. Non è invece possibile usare lo stesso simbolo per array con un differente numero di indici.

Le regole del BASIC ANSI rendono ammissibili solo un numero limitato di nomi. Ciò può comportare difficoltà nella realizzazione di programmi complessi, con numerose variabili. Inoltre rende quasi impossibile ricollegare mnemonicamente il nome di una variabile al suo attributo. Sia il BASIC HP che il GWBASIC consentono quindi di utilizzare, anzichè solo una lettera ed una cifra, un insieme di lettere e cifre molto ampio (fino a 32 o 40 caratteri), con l'unica condizione che il carattere iniziale sia una lettera.

Nei programmi sviluppati in seguito si adotterà uno standard intermedio, considerando ammissibili, oltre a quelli consentiti dal BASIC ANSI, anche nomi formati da due lettere.

TIPI DI INFORMAZIONE

NUMERICA	NUMERO INTERO NUMERO REALE (decimale) possibile distinzione tra semplice e doppia precisione
ALFANUMERICA	INSIEME DI CARATTERI (stringa)
COSTANTE	quantità che ha un valore definito esplicitamente, non alterabile durante l'esecuzione del programma
VARIABILE	nome simbolico che individua i registri di memoria che contengono il valore di una singola informazione o di più informazioni omogenee.

NOMI AMMISSIBILI PER LE VARIABILI

BASIC standard:

VARIABILE NUMERICA	una lettera, oppure una lettera ed una cifra
VARIABILE ALFANUMERICA	una lettera, oppure una lettera ed una cifra, seguiti dal simbolo "$"
ARRAY	come per le variabili semplici, ma seguito da un indice

È possibile utilizzare uno stesso nome per una variabile semplice ed un array.

BASIC HP e GWBASIC:

Lettere e cifre, fino ad un massimo di 32 o 40 caratteri
- Il primo carattere deve essere una lettera
- I nomi delle variabili alfanumeriche devono terminare con il simbolo "$"

Standard adottato negli esempi:

Una lettera, una lettera ed una cifra, due lettere (seguite da "$" se la variabile è alfanumerica)

4.5 - Istruzioni di definizione di tipo e dimensionamento.

Perchè il calcolatore possa memorizzare o elaborare i valori di informazioni è necessario che ne conosca il tipo. La distinzione tra variabili numeriche ed alfanumeriche è automatica, in base alla assenza o presenza del simbolo $ in coda al loro nome. In mancanza di ulteriori precisazioni, tutte le variabili numeriche vengono considerate reali. Per indicare che una variabile è di tipo intero, o reale con un numero di cifre significative diverso, occorre usare specifiche istruzioni.

Nel BASIC HP le istruzioni "**INTEGER ...**" e "**SHORT ...**" contengono l'elenco delle variabili rispettivamente di tipo intero o reale a precisione ridotta. Ad esempio, le linee di programma:

```
10 INTEGER A A1
20 SHORT B7
```

indicano che le variabili A ed A1 sono di tipo intero e la variabile B7 è reale a precisione ridotta.

Nel GWBASIC la definizione di tipo, intero o reale a doppia precisione, è possibile per tutte le variabili i cui nomi iniziano con una stessa lettera, mediante le istruzioni "**DEFINT ...**" e "**DEFDBL ...**". Ad esempio, le linee di programma:

```
10 DEFINT A,C
20 DEFDBL B
```

indicano che tutte le variabili il cui nome inizia per A o per C sono di tipo intero e tutte quelle che iniziano per B sono reali a doppia precisione

Nel GWBASIC è inoltre possibile definire il tipo della singola variabile aggiungendo in coda al suo nome il simbolo "%" per indicare che essa è di tipo intero ed il simbolo "#" per indicare che è a doppia precisione.

Una ulteriore necessità per il calcolatore è quella di conoscere quali variabili siano semplici e quali con indice. Per riservare a quest'ultime un'area di memoria sufficientemente ampia, occorre inoltre definire il numero di indici ed il valore minimo e massimo che ciascuno di essi può assumere.

Il valore minimo dell'indice è 0 oppure 1 a seconda della versione di BASIC utilizzata. Sia il BASIC HP che il GWBASIC assumono implicitamente come limite inferiore il valore 0. È però possibile definire come limite inferiore il valore 1 mediante l'istruzione "**OPTION BASE 1**".

Per definire quanti siano gli indici e quale sia il loro valore massimo si usa l'istruzione "**DIM** ...". Ad esempio, la linea di programma:

```
10 DIM C0(15),D$(6,12)
```

indica che la variabile C0 è un array monodimensionale, contenente 16 elementi (con l'indice da 0 a 15), e la variabile D$ è un array bidimensionale, con 7×13 elementi. Se si è utilizzata la "OPTION BASE 1" il numero di elementi sarà invece rispettivamente 15 e 6×12.

Le indicazioni anzidette devono essere fornite al calcolatore prima di utilizzare gli array interessati. È quindi prassi comune riportare tali istruzioni proprio all'inizio di ciascun programma.

La definizione di array è possibile anche in maniera implicita, semplicemente usando il nome della variabile seguito dall'indice, racchiuso tra parentesi. Così, se in una istruzione compare il termine F1(J,K) il calcolatore identifica la variabile F1 come array bidimensionale. Il massimo numero di elementi è però in tal caso definito automaticamente dal computer, che pone come limite superiore a ciascun indice il valore 10.

Sia il BASIC ANSI che il BASIC HP accettano solo array mono o bidimensionali. Il GWBASIC accetta invece un numero di indici praticamente illimitato (fino a 255).

Nel BASIC HP nasce un analogo problema di dimensionamento anche per le variabili alfanumeriche. Esso infatti assume implicitamente che ciascuna stringa contenga non più di 18 caratteri. Per assegnare una lunghezza massima diversa si utilizza ancora l'istruzione "DIM " indicando tra parentesi quadre il numero di caratteri richiesto. Così, la linea di programma:

```
10 DIM C$[500] V$(30)[6]
```

indica che la variabile semplice C$ può contenere 500 caratteri e che l'ar-

ray V$ è costituito da 31 elementi (o 30, a seconda della OPTION BASE), ciascuno dei quali ha fino a 6 caratteri

Questo problema non esiste invece nel GWBASIC, per il quale la massima lunghezza di ogni stringa è invariabilmente pari a 255 caratteri.

DEFINIZIONE DI TIPO

nome-variabile$ variabile di tipo alfanumerico

BASIC HP:

Se non definite in maniera diversa, tutte le variabili numeriche sono di tipo reale a precisione massima

INTEGER nome-variabile-1, nome-variabile-2 ..

le variabili elencate sono definite di tipo intero

SHORT nome-variabile-1, nome-variabile-2 ..

le variabili elencate sono definite di tipo reale a precisione ridotta

GWBASIC:

Se non definite in maniera diversa, tutte le variabili numeriche sono di tipo reale a precisione semplice

DEFINT lettera-1, lettera-2

le variabili i cui nomi iniziano con le lettere elencate sono definite di tipo intero

DEFDBL lettera-1, lettera-2 ..

le variabili i cui nomi iniziano con le lettere elencate sono definite di tipo reale a precisione doppia.

nome-variabile% variabile di tipo intero
nome-variabile# variabile di tipo reale a doppia precisione

DIMENSIONAMENTO

Se non definito espressamente, il minimo valore
che può assumere l'indice di un array è pari a 0

OPTION BASE 1 definisce come limite inferiore il valore 1

Se non definito espressamente, il massimo valore
che può assumere l'indice di un array è pari a 10

DIM nome-array-1 (indice-max) nome-array-2 (indice-max)

le variabili elencate sono array, i cui indici
possono assumere al massimo i valori indicati

BASIC HP:

INTEGER nome-array-1 (indice-max), nome-array-2 (indice-max)

SHORT nome-array-1 (indice-max) nome-array-2 (indice-max)

le variabili elencate sono array di tipo intero (o di tipo reale a precisione ridotta) i cui indici possono assumere al massimo i valori indicati

Se non definito espressamente, il massimo
numero di caratteri di una stringa è pari a 18

DIM nome-stringa-1 [caratteri] nome-stringa-2 [caratteri]

le variabili alfanumeriche elencate hanno
il numero massimo di caratteri indicato

4.6 - Istruzioni di assegnazione e calcolo

4.6.1 - Assegnazione di valore

Assegnare un valore ad una variabile semplice, o ad un elemento di un array, significa memorizzare il valore, codificato in una sequenza binaria, nei registri di memoria individuati dal nome simbolico della variabile e dall'eventuale indice.

Nel linguaggio BASIC standard l'istruzione che impone ciò è costituita dalla parola chiave "LET", seguita dal nome della variabile, dal simbolo "=" e dal valore da memorizzare La parola "LET" è ridondante, poichè basta il simbolo di uguaglianza per individuare l'istruzione Col tempo essa è quindi scomparsa dall'uso, anche se è tuttora ammessa per compatibilità col BASIC standard.

Il valore da assegnare può essere una costante, il valore di una variabile o il risultato di una espressione. Possibili istruzioni di assegnazione sono riportati nelle linee di programma che seguono

```
10 A=7.25
20 B=F2
30 C=C+1
40 A$="CIAO"
50 B$=F2$
60 C$=A$&B$   !  BASIC HP
```

Si noti che l'istruzione di assegnazione non è una espressione di uguaglianza algebrica, statica, ma una operazione dinamica che comporta prima la determinazione del valore del secondo membro e poi la memorizzazione di tale valore. Ha quindi senso l'istruzione C=C+1 (cioè "incrementa di 1 il valore contenuto in C") che è invece priva di senso in algebra.

ASSEGNAZIONE DI VALORE AD UNA VARIABILE

memorizzazione del valore di una informazione nei registri di memoria individuati dal suo nome simbolico

(LET)	nome-variabile	=	costante variabile espressione

4.6.2 - Espressione aritmetica.

L'espressione aritmetica è un insieme di costanti e variabili numeriche, operatori aritmetici, funzioni matematiche, in base alle quali il calcolatore effettua una elaborazione, ottenendo come risultato un valore numerico

Gli operatori aritmetici, cioè i simboli che individuano le operazioni aritmetiche, sono:

+	somma;
−	sottrazione;
*	moltiplicazione;
/	divisione;
^	elevazione a potenza.

Il BASIC HP ed il GWBASIC accettano inoltre come operatori aritmetici:

\	divisione intera, cioè parte intera del quoziente di una divisione (il BASIC HP utilizza per tale operatore anche la parola "DIV");
MOD	modulo, cioè resto di una divisione.

Così ad esempio l'espressione 19 \ 6 fornisce il risultato 3, mentre 19 MOD 6 fornisce il risultato 1, perchè dividendo 19 per 6 si ottiene 3 col resto di 1.

Col termine "funzione" si intende una prescrizione su come manipolare uno o più valori per ottenere un singolo valore come risultato. Essa è individuata mediante un nome simbolico, seguito dai parametri su cui operare, racchiusi tra parentesi e separati da virgole.

Sono esempi di funzioni matematiche:

ABS(X)	fornisce il valore assoluto di X;
INT(X)	fornisce il più grande intero minore o uguale ad X;
SGN(X)	individua il segno di X, cioè fornisce 1, 0, −1 a seconda che X sia positivo, nullo o negativo;
SQR(X)	fornisce la radice quadrata di X;
MAX(X,Y)	fornisce il maggiore tra X ed Y (solo nel BASIC HP);
MIN(X,Y)	fornisce il minore tra X ed Y (solo nel BASIC HP).

Esistono inoltre funzioni che forniscono un valore numerico ma hanno parametri alfanumerici. Ad esempio:

LEN(X$)	fornisce il numero di caratteri (ovvero la lunghezza, in inglese *lenght*) della stringa X$;
VAL(X$)	fornisce il valore numerico della stringa X$;
NUM(X$)	nel BASIC HP fornisce il codice corrispondente al primo carattere della stringa X$; nel GWBASIC lo stesso risultato è fornito dalla funzione ASC(X$).

Pertanto la funzione LEN("−23.491") fornisce il valore 7, mentre VAL("−23.491") fornisce il valore numerico −23.491

I parametri delle funzioni possono a loro volta essere costituiti da espressioni aritmetiche o alfanumeriche. Così ad esempio SQR(4.5*3−29/4) fornisce il valore 2.5 e SGN(VAL("−23.491")) fornisce il valore −1.

4.6.3 - Espressione alfanumerica.

L'espressione alfanumerica è un insieme di costanti, variabili, operatori e funzioni alfanumeriche.

Il concetto di operatore o di funzione alfanumerica è analogo a quello illustrato per il caso numerico. Non vi è però per essi una standardizzazione, e in alcuni casi lo stesso compito è affidato ad un operatore o ad una sola funzione, a seconda del linguaggio. Se ne riportano nel seguito solo alcuni esempi.

A) Concatenazione di due stringhe.

Questa operazione fornisce una stringa contenente in sequenza i caratteri della prima e della seconda stringa di partenza. Nel BASIC HP l'operatore corrispondente è il simbolo "&"; nel GWBASIC il simbolo "+". Esso può essere applicato sia a costanti che a variabili. Così, ad esempio, nel primo linguaggio l'operazione "AUTO"&"MOBILE" fornisce come risultato la stringa "AUTOMOBILE".

B) Estrazione di una sottostringa.

La stringa che in tal modo si ottiene è parte della stringa iniziale, cioè contiene tutti i caratteri di essa, compresi tra una posizione iniziale ed una finale. Nel BASIC HP ciò viene svolto dall'operatore "[,]", dove tra parentesi quadra sono indicate le due posizioni, iniziale e finale. Esso può essere applicato solo a variabili. Cosi, ad esempio, se A$="AUTOMOBILE", A$[3,6] fornisce il valore "TOMO". Nel GWBASIC lo stesso risultato è fornito dalla funzione "**MID$**(X$,I,J)", che ha come parametri, nell'ordine, la stringa di partenza, la posizione iniziale ed il numero di caratteri della sottostringa. Per ottenere lo stesso risultato dell'esempio precedente si scriverà quindi MID$(A$,3,4).

C) Trasformazione di un numero in stringa.

Nel BASIC HP esiste la funzione **VAL$**(X) che fornisce una stringa corrispondente al numero X. Così ad esempio VAL$(351.24) ha come risultato "351.24". Lo stesso effetto è ottenuto nel GWBASIC mediante la funzione **STR$**(X).

D) Passaggio da un codice al carattere corrispondente.

Sia nel BASIC HP che nel GWBASIC esiste la funzione **CHR$**(X) che fornisce una stringa contenente il carattere che corrisponde al codice X. Il valore numerico X deve essere compreso tra 0 e 255.

4.6.4 - Espressione logica.

L'espressione logica è un insieme di espressioni aritmetiche, o alfanumeriche, operatori relazionale ed operatori logici. Essa fornisce come risultato il valore 0 (falso) o il valore 1 (vero).

Gli operatori relazionali controllano l'esistenza di una relazione tra due espressioni dello stesso tipo. Essi sono:

=	uguale;
<	minore;
>	maggiore;
<=	minore o uguale;
>=	maggiore o uguale;
<>	diverso (il BASIC HP accetta per esso anche il simbolo "#").

Così si ha ad esempio:

3<2.5	0 (falso);
12/3>=3.6	1 (vero);
"AUTO"<>"MOBILE"	1 (vero);
"AUTO"<"MOBILE"	1 (vero) perchè "AUTO" precede "MOBILE" in ordine alfabetico

Gli operatori logici sono in genere applicati ad espressioni di relazione e forniscono come risultato 0 (falso) o 1 (vero), in base alla verità o falsità di esse

Operatori applicati a due espressioni:

AND	fornisce come risultato 1 se entrambe le espressioni sono vere;
OR	fornisce come risultato 1 se almeno una delle espressioni è vera;
EXOR	(XOR nel GWBASIC) fornisce come risultato 1 se solo una delle espressioni è vera (non tutte e due)

Operatore applicato ad una sola espressione:

NOT	fornisce come risultato 1 se essa è falsa

Così, ad esempio:

(3>2.5) AND ("AUTO"<>"MOBILE")	1(vero)	perchè entrambe le espressioni sono vere;
(3>2.5) AND ("AUTO">"MOBILE")	0(falso)	perchè la seconda espressione non è vera;
(3>2.5) OR ("AUTO">"MOBILE")	1(vero)	perchè almeno una espressione (la prima) è vera;
(3>2.5) EXOR ("AUTO">"MOBILE")	1(vero)	perchè solo la prima espressione è vera;
(3>2.5) EXOR ("AUTO"<>"MOBILE")	0(falso)	perchè sono vere entrambe le espressioni (e non solo una);
NOT (3>2.5)	0(falso)	perchè l'espressione è vera.

Si noti che in alcuni linguaggi, come ad esempio il GWBASIC, per simboleggiare "vero" si usa il valore −1 anzichè 1.

4.6.5 - Gerarchia degli operatori

Nell'eseguire una elaborazione complessa, con più operatori, il calcolatore rispetta la loro "gerarchia", cioè un ben preciso ordine di precedenza; solo a parità di livello gerarchico esegue le operazioni nell'ordine con il quale vengono presentate (da sinistra verso destra). Per imporre un diverso ordine si possono utilizzare le parentesi. Ciò è opportuno in particolare per gli operatori relazionali e logici, che possono presentare gerarchie diverse a seconda del linguaggio

La gerarchia presente nel BASIC HP è la seguente:

( )	parentesi;
^	elevazione a potenza;
NOT	operatore logico NOT;
* / MOD DIV	moltiplicazione e divisione;
+ −	somma e sottrazione;
= < > <= >= <>	operatori relazionali;
AND	operatore logico AND;
OR EXOR	operatori logici OR EXOR

4.6.6 - Esempi

È riportata di seguito la codifica di tre programmi già analizzati in precedenza nel paragrafo 3.3: risoluzione di una equazione di primo grado, risoluzione di un sistema lineare di due equazioni in due incognite e verifica di sezione rettangolare in c.a. soggetta a flessione semplice. Dopo ciascuno di essi è riportato il risultato fornito dalla esecuzione (su HP-87)

A) Risoluzione di una equazione di primo grado

```
 10 REM RISOLUZIONE DI UNA EQUAZIONE DI PRIMO GRADO
 20 REM
 30 REM Equazione :    a * x + b = 0
 40 REM
 50 REM Variabili : A    coefficiente della incognita
 60 REM             B    termine noto
 70 REM             X    incognita
 80 REM
100 READ A,B
110 X=-(B/A)
120 PRINT "Risultato :  X
130 STOP
140 DATA 3 2 10

Risultato :        -3 125
```

B) Risoluzione di un sistema lineare di equazioni (2 equazioni, 2 incognite)

```
 10 REM RISOLUZIONE DI UN SISTEMA LINEARE DI EQUAZIONI
 20 REM          ( 2 equazioni  2 incognite )
 30 REM
 40 REM Sistema di equazioni: A * X + B * Y = C
 50 REM                       D * X + E * Y = F
 60 REM
 70 REM Variabili : A B C D E F        coefficienti delle equazioni
 80 REM             V                  variabile interna
 90 REM             X,Y                incognite
100 REM
200 READ A B C,D E F
210 V=A*E-B*D
220 X=(C*E-F*B)/V
230 Y=(A*F-C*D)/V
240 PRINT "Risultati :",X Y
250 STOP
260 DATA 1,2,3,4,5,6
```

Risultati: −1 2

C) Verifica a flessione - sezione rettangolare in c.a. a doppia armatura

```
 10 REM          VERIFICA A FLESSIONE
 20 REM     Sezione rettangolare in c a. a doppia armatura
 30 REM
 40 REM Variabili: B     base della sezione [cm]
 50 REM            H1    altezza utile della sezione [cm]
 60 REM            D     copriferro [cm]
 70 REM            N     rapporto Ef/Ec
 80 REM            A1    area di ferro teso [cm2]
 90 REM            A2    area di ferro compresso [cm2]
100 REM            A     area totale di ferro [cm2]
110 REM            M     momento flettente [kgcm]
120 REM            X     posizione dell'asse neutro [cm]
130 REM            S1    tensione nel calcestruzzo [kg/cm2]
140 REM            S2    tensione nell'armatura [kg/cm2]
150 REM
200 READ B,H1,D,N,A1,A2,M
210 A=A1+A2
220 X=N*A/B*(-1+SQR (1+2*B*(A1*H1+A2*D)/N/A^2))
230 S1=M/(B*X/2*(H1-X/3)+N*A2/X*(X-D)*(H1-D))
240 S2=N*S1*(H1-X)/X
250 PRINT "Risultati :"
260 PRINT X,S1,S2
270 STOP
280 DATA 30,57,3,15,9,0,850000
```

Risultati :
18.5922064775 59 9944134763 1859 04753808

4.7 - Istruzioni di salto.

4.7.1 - Salto incondizionato.

Come già spiegato in precedenza, si intende per salto incondizionato l'imposizione assoluta (senza condizioni) di abbandonare la usuale sequenza, saltando ad una diversa linea di programma e proseguendo poi da essa

Nel linguaggio BASIC l'istruzione che impone ciò è costituita dalla parola chiave "**GOTO**" (dall'inglese *go to*, cioè "vai a"), seguita dal numero che individua la linea di programma che si deve passare ad eseguire.

Si prendano ad esempio in esame le seguenti linee di programma:

```
10 A=0
20 GOTO 70
30 .
   .
70 A=A+1
80 B$='ESEMPIO"
90
```

Il calcolatore esegue l'istruzione 10; poi incontra il comando di salto e passa ad eseguire l'istruzione 70; prosegue quindi con le istruzioni 80, 90, ecc.

Il salto può essere indirizzato ad una qualsiasi linea di programma, che preceda o segua l'istruzione di salto, e persino alla stessa linea che la contiene (anche se ciò bloccherebbe il programma in un ciclo senza fine) Si consiglia, comunque, di far sempre riferimento ad istruzioni operative, e non ad istruzioni di commento, che potrebbero in una fase successiva essere ritenute superflue ed eliminate.

Le istruzioni di salto incondizionato verranno di seguito utilizzate solo per realizzare le strutture logiche non direttamente previste dal BASIC.

4.7.2 - Salto ad un sottoprogramma.

Anche il ricorso ad un sottoprogramma richiede un salto, cioè il passare ad eseguire una diversa sequenza di istruzioni. In questo caso, però, il calcolatore memorizza l'indirizzo di partenza, in modo da poter tornare alla sequenza principale quando è terminata l'esecuzione delle istruzioni del sottoprogramma.

Nel linguaggio BASIC l'istruzione che impone questo salto è costituita dalla parola chiave "**GOSUB**" (dall'inglese *go to subroutine*, cioè vai al sottoprogramma), seguita dall'indirizzo della prima istruzione del sottoprogramma L'istruzione "**RETURN**" indica invece la fine della sequenza secondaria ed impone il ritorno a quella principale, cioè il salto all'istru-

zione successiva al GOSUB

Si prendano ad esempio in esame le seguenti linee di programma:

```
10 A=0
20 GOSUB 70
30
40
..
60 STOP
70 A=A+1
80 B$="ESEMPIO
90 RETURN
```

Il calcolatore esegue l'istruzione 10; poi incontra il comando di salto e passa ad eseguire l'istruzione 70; prosegue con l'istruzione 80; incontra quindi il comando RETURN e torna alla sequenza principale, eseguendo le istruzioni 30, 40, ecc.

È importante ricordare che una sequenza, considerata come sottoprogramma, non deve essere poi ripercorsa come parte della sequenza principale L'istruzione RETURN in tal caso non avrebbe significato e verrebbe considerata erronea, causando l'interruzione dell'esecuzione Si avrà pertanto cura di porre i sottoprogrammi in coda al programma principale, assegnando alle loro linee numeri maggiori di quelli riservati alla sequenza principale.

SALTO INCONDIZIONATO

GOTO n

vai ad eseguire l'istruzione contenuta nella linea di programma contrassegnata con il numero n ;
prosegui poi con le istruzioni successive a quella linea

RICORSO AD UN SOTTOPROGRAMMA

GOSUB n

vai ad eseguire l istruzione contenuta nella linea di programma contrassegnata con il numero n ;
prosegui poi con le istruzioni successive a quella linea, fino ad incontrare l'istruzione "RETURN"

RETURN

vai ad eseguire l istruzione contenuta nella linea di programma immediatamente successiva al "GOSUB n" ;
prosegui poi con le istruzioni successive a quella linea

4.8 - Istruzioni decisionali

4.8.1 - Salto condizionato

La fondamentale istruzione decisionale, comune a tutti i linguaggi BASIC, è il salto condizionato, cioè l'istruzione che impone di passare alla esecuzione di un'altra parte di programma se si verifica una assegnata condizione. Essa si presenta nel seguente modo:

IF *espressione-logica* **THEN** *n*

dove n è il numero della linea di programma a cui bisogna saltare se l'espressione logica fornisce come risultato 1 (vero)

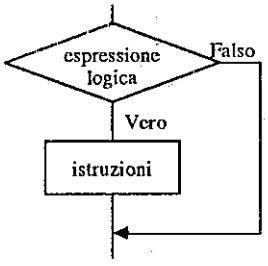
4.8.2 - Struttura logica IF ... THEN ...

La struttura logica "IF ... THEN ..." può essere realizzata utilizzando l'istruzione di salto condizionato, come mostrato in figura 32. Si noti che le istruzioni da eseguire se la condizione è vera sono poste immediatamente dopo essa, e quindi il salto deve essere effettuato se la condizione *non* è verificata (cioè se è vera la condizione opposta "NOT espressione").

Quasi tutti i BASIC consentono inoltre la realizzazione di tale struttura direttamente con l'istruzione:

IF *espressione-logica* **THEN** *istruzione*

Mediante essa è però possibile eseguire condizionatamente solo una istruzione, e non una sequenza di più istruzioni. Il BASIC HP ed il GWBASIC consentono di indicare dopo il "THEN" più istruzioni, ma sempre in numero limitato.



IF ... THEN ...

```
... IF NOT espressione THEN n
. . . . . .
. . . . . istruzioni
. . . . . .
n . . . . . .
```

Fig. 32

4.8.3 - Struttura logica IF ... THEN ... ELSE ...

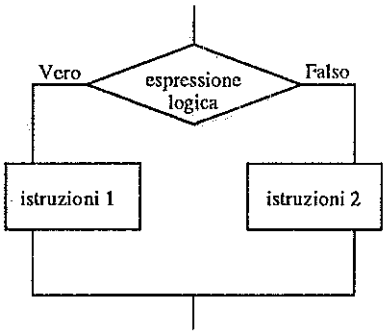
La struttura logica "IF ... THEN ... ELSE ..." può essere realizzata uti-

lizzando le istruzioni per il salto condizionato ed incondizionato, come mostrato in figura 33. Anche in questo caso, per conservare l'ordine dei due bocchi alternativi il salto viene effettuato quando la condizione *non* è verificata.

Quasi tutti i BASIC consentono inoltre la realizzazione di tale struttura direttamente con l'istruzione:

IF *espressione-logica* **THEN** *istruzione-1* **ELSE** *istruzione-2*

Mediante essa è però possibile l'alternativa tra due istruzioni, e non tra due sequenze di più istruzioni. Il BASIC HP ed il GWBASIC consentono di indicare sia dopo THEN che dopo ELSE più istruzioni, ma sempre in numero limitato




```
   IF NOT espressione THEN k
   ...
                              istruz 1
   ...
   GOTO n
k  ...
                              istruz 2
   ...
n  ...
```

IF THEN ELSE

Fig 33

4.8.4 - Struttura logica CASE ... OF ...

La struttura logica "CASE ... OF ..." (fig. 34) può essere realizzata usando più istruzioni di salto condizionato ed incondizionato, come mostrato nelle righe seguenti.

```
  IF espressione = 1 THEN n1
  IF espressione = 2 THEN n2
  IF espressione = 3 THEN n3
  IF espressione = 4 THEN n4
  PRINT "CASO NON ESISTENTE"
  STOP
n1 ...
   ...                        istruz 1
  GOTO k
n2 ...
   ...                        istruz 2
```

```
  . GOTO k
n3  ....  .
  .. ..... . .          istruz 3
  .. GOTO k
n4  .  .
  .. ..... .            istruz 4
  .. GOTO k
k  .  .  .  ...  .  .
```

Lo stesso risultato può essere ottenuto con l'istruzione:

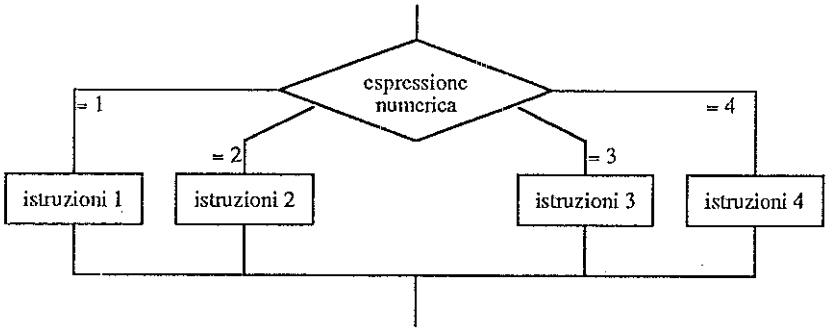
ON *espressione-numerica* **GOTO** *n1, n2, n3* ...

che presenta dopo il GOTO l'elenco delle linee di programma alle quali bisogna saltare a seconda del valore dell'espressione

Si può infine utilizzare anche l'istruzione:

ON *espressione-numerica* **GOSUB** *n1, n2, n3*

Essa consente di far eseguire, in base al valore dell'espressione, uno tra più sottoprogrammi alternativi, e quindi tra più blocchi di istruzioni.



CASE . OF ..

Fig 34

4.8.5 - Esempio.

Si riporta di seguito la codifica di un programma per la risoluzione di una equazione di secondo grado, già analizzato nel paragrafo 3 4. Esso utilizza la struttura logica "IF ... THEN ... ELSE ..." e richiede quindi l'uso di salti condizionati. Dopo il listato sono riportati i risultati ottenuti

```
 10 REM RISOLUZIONE DI UNA EQUAZIONE DI SECONDO GRADO
 20 REM
 30 REM Equazione :      A * X^2 + B * X + C = 0
 40 REM
 50 REM Variabili: A B C     coefficienti dell equazione
 60 REM            D         Delta = B^2-4*A*C
 70 REM            X1 X2     valori dell'incognita
 80 REM
200 READ A B C
210 D=B^2-4*A*C
220 IF D>= 0 THEN 250
230 PRINT "SOLUZIONE NON REALE
240 GOTO 280
250 X1=(-B+SQR (D))/(2*A)
260 X2=(-B-SQR (D))/(2*A)
270 PRINT "Risultati :",X1,X2
280 STOP
290 DATA 3.2 -4
```

Risultati: .868517091822 -1.53518375849

ISTRUZIONI DECISIONALI

IF espressione-logica **THEN** n

se l'espressione è vera salta alla linea di programma n:
in caso contrario prosegui normalmente in sequenza

IF espressione-logica **THEN** istruzione

se l'espressione è vera esegui l'istruzione; in ogni caso prosegui con l'istruzione contenuta nella linea successiva

IF espressione-logica **THEN** istruzione-1 **ELSE** istruzione-2

se l'espressione è vera esegui l'istruzione-1, altrimenti esegui l'istruzione-2;
in ogni caso prosegui con l'istruzione contenuta nella linea successiva

ON espressione-aritmetica **GOTO** n1, n2, n3

salta ad una delle linee n1, n2, n3 in base al valore della espressione

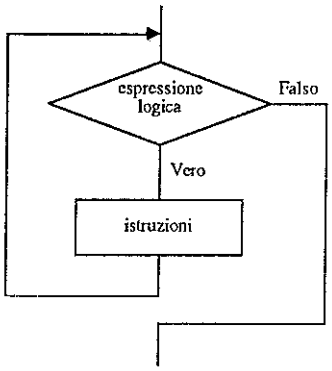
ON espressione-aritmetica **GOSUB** n1, n2, n3

in base al valore dell'espressione, vai ad eseguire uno dei sottoprogrammi che cominciano alle linee n1, n2, n3 ; quando incontri "RETURN" torna alla linea successiva

4.9 - Istruzioni di ciclo

4.9.1 - Strutture logica WHILE ... DO ... e REPEAT ... UNTIL ...

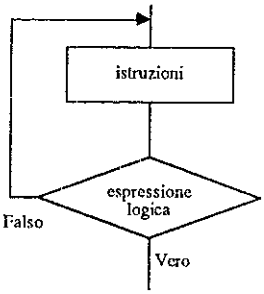
Il BASIC standard e il BASIC HP non posseggono istruzioni che consentano di definire direttamente tali strutture. Occorre quindi ricorrere ad istruzioni di salto condizionato ed incondizionato, come mostrato in figura 35 e in figura 36.



WHILE ... DO

```
k   IF NOT espressione THEN n
        istruzioni
    GOTO k
n
```

Fig. 35



REPEAT ... UNTIL

```
n
        istruzioni
    IF NOT espressione THEN n
```

Fig. 36

Nel GWBASIC la struttura "WHILE . . DO . ." può essere realizzata direttamente mediante le istruzioni "**WHILE** . ." e "**WEND**" (While END, cioè fine del WHILE). L'istruzione "WHILE . . " indica l'inizio del ciclo ed è seguita dall'espressione logica. Finchè tale espressione fornisce come risultato "vero", vengono eseguite le istruzioni che seguono, fino alla WEND.

Si riporta di seguito la codifica di due programmi già esaminati nei paragrafi 3 6 e 3 7, che presentano strutture "WHILE . . DO . . ": tabellazione e determinazione del punto di nullo di una funzione. Al testo dei programmi seguono i relativi risultati

A) Tabellazione di una funzione.

```
 10 REM       TABELLAZIONE DI UNA FUNZIONE
 20 REM
 30 REM Funzione :          Y = -7 * X^3 + SQR (X^2 + 10)
 40 REM
 50 REM Variabili: A B      estremi dell intervallo
 60 REM            D        passo di incremento
 70 REM            X        ascissa corrente
 80 REM            Y        valore di f(X)
 90 REM
200 READ A,B,D
210 X=A
220 IF X>B THEN 270
230 Y=-(7*X^3)+ SQR (X^2+10)
240 PRINT X,Y
250 X=X+D
260 GOTO 220
270 STOP
280 DATA 0,1, 1
```

Operando su un calcolatore HP-87 si ottengono i seguenti risultati:

```
0                3  16227766017
 1               3 . 15685840391
 2               3  11259590355
 3               2  98747603485
 4               2  7394754901
 5               2  32656211872
 6               1 . 70669538789
 7                 83782694814
 8              - . 32209871394
 9              -1  81514355545
1               -3 . 68337520964
```

Usando le istruzioni "WHILE . ." e "WEND" del GWBASIC, le linee dal numero 220 al 260 possono essere sostituite da:

```
220 WHILE X <= B
230 Y=-(7*X^3)+SQR (X^2+10)
240 PRINT X Y
250 X=X+D
260 WEND
```

Operando, con o senza queste variazioni, su un calcolatore che utilizza il linguaggio GWBASIC, si ottengono i seguenti risultati:

```
0                3 .162278
 .1              3 .156859
 .2              3  112596
 .3              2  987476
 .4              2  739476
 .5              2  326562
 .6              1 .706695
 .7000001         .8378265
 .8000001       - .3220997
 .9000001       -1  815145
```

Confrontando questi valori con quelli forniti dall'HP-87, si nota innanzitutto la differenza del numero di cifre significative adoperate dall'elaboratore. È evidenziata però anche un altra particolarità, che in questo caso provoca un errore piuttosto insidioso perchè non facilmente prevedibile. Nell'effettuare l'incremento della variabile di controllo, aggiungendo 0.1 a 0.6 si è ottenuto il valore 0.7000001 anzichè 0.7. Una tale differenza è normalmente irrilevante ai fini pratici, ma nel caso in esame diventa fatale per il controllo del ciclo. Infatti, continuando nell'incremento, il computer passa ai valori 0.8000001 e 0.9000001, fornendo il valore corrispondente della funzione. Il successivo incremento assegna alla variabile X un valore superiore, anche se di pochissimo, al limite 1, ed il ciclo viene quindi interrotto, omettendo il calcolo e la stampa dell'ultimo valore.

Utilizzando il calcolatore, è bene ricordare sempre che i cosiddetti numeri reali sono in realtà per l'elaboratore dei numeri decimali con un numero massimo di cifre prefissato. Le operazioni su di essi comportano in generale un errore di arrotondamento, che è tanto maggiore quanto più piccolo è il numero di cifre significative e che può propagarsi (cioè aumentare) quando si effettua un numero notevole di calcoli. Ad esempio, nella risoluzione di strutture con metodi matriciali la correttezza del risultato può essere anche sensibilmente inficiata se si opera con una precisione ridotta.

Nel caso in esame si evidenzia però anche un altro aspetto di questo problema. La codifica di un numero reale in un insieme di bit può avvenire sostanzialmente in due maniere diverse. Si può rispettare la impostazione decimale, facendo corrispondere a ciascuna cifra un insieme di bit (per l'esattezza 4); oppure trasformare l'intero numero nel corrispondente in binario. L'HP-87 segue la prima via, mentre il GWBASIC la seconda. La scelta è rilevante si fini dell'ingombro di memoria. Ad esempio, per memorizzare un numero con 7 cifre significative rispettando la notazione

decimale occorrono 28 bit; trasformandolo interamente in binario ne sono sufficienti 24, perchè 2^{24} vale 16777216, che è maggiore di 10^7. La seconda impostazione ha però un inconveniente: la codifica è esatta per i numeri interi, diventa però approssimata per i decimali, causando gli effetti sopra segnalati.

Utilizzando variabili reali come contatori per il controllo di un ciclo si corre sempre il rischio di incappare in errori del genere. La impostazione preferibile è quella di utilizzare come indici del ciclo solamente numeri interi. Un'alternativa ugualmente valida ai fini pratici può essere quella di incrementare fittiziamente di pochissimo il valore del limite finale (ad esempio, nel caso in esame, portandolo a 1.000001).

B) Ricerca del punto di nullo di una funzione.

La codifica segue scrupolosamente i diagrammi di flusso mostrati nelle figure 25, 26 e 27. La determinazione del valore della funzione, richiesta in più punti del programma, è effettuata in un unico sottoprogramma (linee 2000 e 2010). In tal modo, se si vuole determinare il punto di nullo di un'altra funzione basta semplicemente modificare la linea 2000, inserendovi la nuova espressione analitica. Lo stesso risultato si potrebbe ottenere usando, al posto di un sottoprogramma, una funzione definita dall'utente mediante le istruzioni "**FN**....", che per brevità si è preferito non trattare nel presente testo.

```
 10 REM   RICERCA DEL PUNTO DI NULLO DI UNA FUNZIONE
 20 REM
 30 REM Funzione:     Y = -7 * X^3 + SQR (X^2 + 10)
 40 REM
 50 REM Variabili: A ,B      estremi dell'intervallo iniziale
 60 REM            E1        approssimazione ammissibile per f(x)
 70 REM            E2        ampiezza minima dell'intervallo
 80 REM            X1 X2     estremi dell'intervallo generico
 90 REM            Y1,Y2     valori di f(X1) ed f(X2)
100 REM            X         punto medio dell intervallo
110 REM            Y         valore di f(X)
120 REM            R         variabile di comodo :
130 REM                      R=0   cercare punto di nullo
140 REM                      R=1   punto di nullo trovato
150 REM                      R=2   punto di nullo inesistente
1000 READ A,B,E1,E2
1010 X=A
1020 GOSUB 2000
1030 X1=X
1040 Y1=Y
1050 X=B
1060 GOSUB 2000
1070 X2=X
1080 Y2=Y
1090 IF Y1=0 THEN 1150
```

```
1090 IF Y1=0 THEN 1150
1100 IF Y2=0 THEN 1180
1110 IF Y1*Y2>0 THEN 1210
1120 IF Y1*Y2<0 THEN 1230
1130 PRINT 'CASO NON ESISTENTE'
1140 STOP
1150 X=A
1160 R=1
1170 GOTO 1250
1180 X=B
1190 R=1
1200 GOTO 1250
1210 R=2
1220 GOTO 1250
1230 R=0
1240 GOTO 1250
1250 IF R=0 THEN 1380
1260 X=(X1+X2)/2
1270 GOSUB 2000
1280 IF ABS (Y)>E1 THEN 1310
1290 R=1
1300 GOTO 1370
1310 IF Y1*Y>= 0 THEN 1340
1320 X2=X
1330 GOTO 1360
1340 X1=X
1350 Y1=Y
1360 IF X2-X1<=E2 THEN R=1
1370 GOTO 1250
1380 IF R#1 THEN 1410
1390 PRINT "Punto di nullo : ",X
1400 GOTO 1420
1410 PRINT "Non esiste punto di nullo
1420 STOP
1430 DATA 0 1, 001, 001
1990 REM-----------------------------
2000 Y=-(7*X^3)+SQR (X^2+10)
2010 RETURN
```

Punto di nullo: 7744140625

4.9.2 - Struttura logica FOR .... DO ....

Questa struttura può essere realizzata in BASIC mediante due istruzioni, che definiscono rispettivamente l'inizio e la fine del ciclo:

FOR *nome-variabile* = *espr.-1* **IO** *espr.-2* **SIEP** *espr -3*

....

NEXI *nome-variabile*

Con essa si impone al calcolatore di eseguire le istruzioni comprese tra

FOR e NEXT più volte, utilizzando come contatore la variabile il cui nome è specificato, facendola variare dal valore fornito dall'espressione 1 a quello fornito dall'espressione 2, col passo indicato dall'espressione 3. Se l'ultima parte dell'istruzione (STEP *espressione-3*) è omessa, viene implicitamente assunto un passo di incremento pari ad 1. Si ha così ad esempio:

```
10 F=1                          ! Calcola e stampa il fattoriale dei numeri
20 FOR N=1 TO 20                ! compresi tra 1 e 20
30 F=F*N                        ! N, contatore del ciclo, viene fatto variare
40 PRINT F                      ! da 1 a 20 con passo unitario
50 NEXT N                       ! La variabile F contiene il valore del
60 STOP                         ! fattoriale
```

Le istruzioni "FOR ..." e "NEXT ..." vengono in genere realizzate come fossero cicli "WHILE ... DO ...". Si prenda per esempio in esame il seguente programma:

```
10 K=0
20 FOR N=1 TO K
30      PRINT " Il valore corrente del contatore N è ";N
40 NEXT N
50 STOP
```

Il calcolatore dovrebbe effettuare all'inizio del ciclo il confronto tra il valore dell'indice (1) ed il limite superiore (K, che vale 0) Non essendo verificata la condizione $1<0$, le istruzioni interne al ciclo non dovrebbero mai essere eseguite, cioè il programma non dovrebbe stampare alcun risultato.

Vi sono però dei linguaggi che realizzano le istruzioni "FOR ... NEXT ..." come fossero cicli "REPEAT ... UNTIL ...". Il confronto viene in tal caso effettuato dopo aver eseguito le istruzioni interne al ciclo, che verrebbero quindi eseguite sempre almeno una volta. Il programma innanzi scritto fornirebbe quindi come risultato:

```
Il valore corrente del contatore N è 1
```

Vi è infine anche qualche linguaggio che, quando il valore della seconda espressione è minore di quello della prima ed il passo non è specificato, assume automaticamente per quest'ultimo il valore di -1 (decremento unitario). In tal caso i risultati forniti dal solito programma sarebbero:

```
Il valore corrente del contatore N è 1
Il valore corrente del contatore N è 0
```

È pertanto opportuno innanzitutto controllare quale sia l'effettivo comportamento del linguaggio del calcolatore che si ha a disposizione (magari proprio facendo eseguire questo programma di prova). Si consiglia inoltre, per garantire la compatibilità dei propri programmi con altri linguaggi, di premettere a tutti i cicli che possono presentare un tale

problema un controllo (istruzione "IF ... THEN ...") che li faccia eventualmente saltare Nel programma di prima basterebbe aggiungere l'istruzione:

```
15 IF K<1 THEN 50
```

Si riporta di seguito la codifica di programmi, esaminati nei paragrafi 3 6 e 3.8, che contengono cicli "FOR ... DO ...". In essi le istruzioni interne al ciclo sono state scritte spostate verso destra, rispetto alle istruzioni "FOR" e "NEXT", per meglio evidenziarle

A) Calcolo del punteggio base per la laurea.

```
 10 READ N
 20 S=0
 30 FOR C=1 TO N
 40      READ V
 50      S=S+V
 60 NEXT C
 70 P=S/N*110/30
 80 PRINT  Punteggio base per la laurea : ";P
 90 STOP
100 DATA 19
110 DATA 27 30 28 30 27 26,30,30,24,26,24,30,27,28,28,30 27 28 30
```

```
Punteggio base per la laurea:      102 280701754
```

B) Trasposizione di una matrice quadrata.

```
1000 FOR I=1 TO N-1                 ! Sottoprogramma per la trasposizione di una
1010     FOR J=I+1 TO N             ! matrice quadrata
1020          V=A(I J)              ! Le variabili I e J indicano il valore della
1030          A(I,J)=A(J,I)         ! riga e della colonna dell'elemento dello
1040          A(J,I)=V              ! array A
1050     NEXT J
1060 NEXT I
1070 RETURN
```

C) Prodotto di due matrici.

```
1000 FOR I=1 TO M                   ! Sottoprogramma per il prodotto di due
1010     FOR J=1 TO P               ! matrici A e B
1020          C(I,J)=0
1030          FOR K=1 TO N
1040               C(I,J)=C(I,J)+A(I K)*B(K J)
1050          NEXT K
1060     NEXT J
1070 NEXT I
1080 RETURN
```

4.10 - Istruzioni di ingresso dati

4.10.1 - Generalità

Col termine "ingresso dati" o "input" si intende l'assegnazione di valore alle variabili di ingresso, cioè la quantizzazione delle informazioni di cui il calcolatore ha bisogno come punto di partenza per le sue elaborazioni

Dal punto di vista operativo, se ne possono distinguere due diverse modalità: assegnazione dei dati all'interno del programma o assegnazione in maniera interattiva, conversazionale, durante l'esecuzione.

La prima situazione può essere realizzata in BASIC mediante l'istruzione "READ", che consente di assegnare alle variabili un valore letto direttamente nel programma. In alternativa, un valore di ingresso può essere definito esplicitamente mediante l'istruzione di assegnazione Si ricorre però a quest'ultima solo in casi particolari, quale quello di informazioni che si considerano variabili per mantenere la generalità del programma, ma che in genere hanno sempre uno stesso valore Si pensi, ad esempio, al coefficiente di omogeneizzazione n=Ef/Ec, che in base alla attuale normativa assume il valore 15, ma in passato poteva essere preso pari a 10

Per ottenere invece un ingresso dati interattivo esiste l'istruzione "INPUT", che interrompe l'esecuzione per chiederne il valore direttamente all'utilizzatore del programma. Infine, è possibile realizzare un input conversazionale ottimale per l'utente, ma più complesso per il programmatore, mediante istruzioni che controllano tastiera e video. Di esso, denominato "input a maschera", si tratterà diffusamente nel capitolo 6

4.10.2 - Requisiti ottimali

Per l'utente di un programma, l'ingresso dati è una fase di importanza fondamentale, lunga e faticosa quando la quantità di informazioni da fornire è notevole È quindi indispensabile analizzare i problemi che in essa si possono incontrare, per definire i requisiti che l'input ottimale dovrebbe soddisfare.

A) Compattezza della fase di ingresso dati

Una tendenza che spesso si incontra, specie tra programmatori autodidatti che si sono formati lavorando con calcolatori molto limitati, è quella di cercare ad ogni costo la minimizzazione dei tempi di esecuzione e dell'ingombro di memoria. Da ciò deriva, ad esempio, l'inframmezzare istruzioni di input con elaborazioni, magari per non memorizzare direttamente tutti i dati di ingresso ma solo alcune grandezze da essi derivate

Si è invece già sottolineato come la caratteristica principale di un buon programma sia la sua chiarezza e leggibilità. Per ottenere ciò è importante distinguere le diverse fasi logiche ed individuare blocchi compatti di ingresso dati. Questo diventa poi particolarmente importante nel caso di grossi programmi con input interattivo. Si pensi ad esempio ad un programma per il calcolo di telai, che può richiedere anche decine di minuti per la elaborazione. Se i dati relativi a ciascuna condizione di carico devono essere forniti man mano che gli schemi vengono risolti, occorre restare vicino al calcolatore, in attesa delle nuove richieste. Se invece tutti i dati vengono definiti all'inizio, una volta che l'elaborazione è cominciata si può abbandonare il computer e dedicarsi ad altre attività.

La compattezza della fase di ingresso dati è comunque opportuna, indipendentemente dal tipo di input (conversazionale o no) e dalle istruzioni di ingresso che si utilizzano per la codifica. Deve essere quindi prevista fin dalla fase di programmazione.

B) Chiarezza e semplicità di input.

Molti errori nell'uso di programmi nascono da una assegnazione erronea dei dati. È quindi di importanza fondamentale che le modalità di input siano definite in maniera tale da non consentire interpretazioni scorrette. Può essere utile a ciò un "manuale d'uso", ma è preferibile che il programma stesso fornisca chiaramente tutte le indicazioni necessarie, indicando per ogni informazione richiesta il suo attributo e le eventuali unità di misura imposte. Inoltre, l'ingresso dati deve essere commisurato alle capacità dell'utente. Non deve, ad esempio, richiedere un intervento diretto sul programma a chi non ha nozioni base di programmazione.

C) Possibilità di correggere errori nei dati.

Capita a volte di accorgersi, prima ancora di aver completato l'input, di aver assegnato un valore errato ad una informazione già definita. Quando i dati sono pochi, ricominciare dall'inizio non è un problema; nel caso di programmi complessi è invece di fondamentale importanza il poter modificare con facilità qualsiasi valore già assegnato.

D) Possibilità di riutilizzare i dati.

Solo nei casi più semplici si può considerare definitivamente risolto un problema una volta completata l'esecuzione ed ottenuti i risultati. Si pensi al calcolo di un telaio, già richiamato come esempio. I valori ottenuti devono essere analizzati, prima di tutto per riscontrare la correttezza dei dati di ingresso e poi per verificare l'ammissibilità delle caratteristiche di sollecitazione così determinate. Può quindi essere necessario ripetere

l'elaborazione modificando solo una piccola parte dei dati È evidente il vantaggio che si ottiene se si possono riutilizzare i valori usati in precedenza, apportando solo le correzioni necessarie, anzichè riassegnarli tutti quanti.

Una nuova elaborazione può essere richiesta immediatamente dopo la precedente, cioè quando programma e dati sono ancora presenti nella memoria centrale del calcolatore, oppure a distanza di tempo In questo secondo caso diventa indispensabile conservare stabilmente i dati su memoria di massa.

4.10.3 - Lettura dati all'interno del programma.

In BASIC è possibile creare all'interno di un programma un insieme ordinato di valori, dal quale attingere per definire i dati di ingresso L'istruzione che consente ciò è costituita dalla parola "**DATA**" (che in inglese vuol dire "dati"), seguita da un elenco di valori In uno stesso programma si possono avere numerose istruzioni DATA; i valori in esse presenti vengono considerati sequenzialmente, in base al numero della linea di programma che li contiene. Nel calcolatore esiste un puntatore, cioè un particolare registro di memoria il cui contenuto indica ("punta a") un elemento di tale insieme, il primo disponibile

L'istruzione che consente di assegnare in ingresso valori dell'insieme così definito è individuata dalla parola "**READ**" (che in inglese vuol dire "leggi"), seguita dall'elenco delle variabili i cui valori devono essere letti. Essa impone al calcolatore di assegnare sequenzialmente a ciascuna variabile il primo valore disponibile nell'insieme. Ogni volta che un valore viene assegnato, il puntatore passa ad indicare quello immediatamente successivo.

Si considerino, ad esempio, le seguenti linee di programma:

```
10 READ A,B,C$
20 READ D E$,F$
30 DATA 5,-2,ANTONIO,7
40 DATA "14 NOVEMBRE",6 "2 3"
```

L'insieme di dati, definito dalle istruzioni 30 e 40, è costituito dai sette valori: 5, −2, "ANTONIO", 7, "14 NOVEMBRE", 6, "2,3".

Si noti che, nell'istruzione DATA, un gruppo di caratteri è considerato valore alfanumerico anche se non è espressamente racchiuso tra virgolette.

Inizialmente il puntatore indica il primo di essi, il valore numerico 5. Eseguendo l'istruzione 10, il calcolatore pone A=5, B=−2, C$="ANTONIO" ed il puntatore passa ad indicare il valore 7. Perchè l'istruzione sia eseguita correttamente il tipo della variabile e del valore corrispondente devono coincidere

Eseguendo poi l'istruzione 20 il calcolatore pone D = 7, E$ = '14 NOVEMBRE", F$ = "6". Per quest'ultimo, si noti che, essendo la variabile F$ alfanumerica, il valore assegnatole è l'insieme di caratteri "6" e non il valore numerico 6. Il puntatore è passato ad indicare l'ultimo valore dell'elenco ("2,3"), che rimane disponibile per ulteriori operazioni di ingresso.

Come si nota nell'esempio fatto, la corrispondenza è creata tra l'insieme di variabili elencate nelle istruzioni READ e l'intero insieme di valori contenuti nei DATA. È però comodo, per maggiore chiarezza, raggruppare i valori nelle istruzioni DATA con criteri logici, in genere creando una corrispondenza diretta tra singole istruzioni READ e DATA. Così, sarebbe stato preferibile, anche se identico ai fini pratici, scrivere le stesse linee di programma in maniera diversa, facendo idealmente corrispondere le linee 10 e 30 e le linee 20 e 40:

```
10 READ A,B,C$
20 READ D,E$,F$
30 DATA 5,-2,ANTONIO
40 DATA 7,"14 NOVEMBRE" 6
50 DATA '2,3'
```

Il puntatore ai valori dei dati può essere riposizionato in maniera tale da indicare il primo elemento dell'insieme. L'istruzione che consente ciò è costituita dalla parola "**RESTORE**" (che vuol dire "ripristina"). Nel BASIC HP e nel GWBASIC essa può essere anche seguita dal numero di una linea di programma che contenga dati; si indica in tal modo di riposizionare il puntatore in maniera tale da indicare il primo valore contenuto in quella riga. Così, per esempio, con riferimento alle linee di programma appena scritte, eseguendo l'istruzione:

```
60 RESTORE
```

il puntatore torna ad indicare il valore numerico 5; eseguendo poi l'istruzione:

```
70 RESTORE 40
```

esso passa ad indicare il valore numerico 7.

DATA	valore-1, valore-2 i valori elencati vanno a far parte di un insieme, da cui possono essere prelevati per quantizzare le variabili di ingresso; un puntatore indica il primo elemento dell'insieme disponibile per le operazioni di lettura dati
READ	variabile-1, variabile-2 a ciascuna variabile dell'elenco viene assegnato un valore tratto sequenzialmente dall'insieme costituito con le istruzioni DATA

RESTORE

riposiziona il puntatore in maniera tale da indicare il primo elemento dell'insieme

RESTORE n (BASIC HP e GWBASIC)

riposiziona il puntatore in maniera tale da indicare il primo valore contenuto nell'istruzione DATA posta alla linea n

4.10.4 - Lettura dati interattiva

In BASIC è possibile un ingresso dati interattivo, cioè l'inserimento di valori per le variabili di ingresso direttamente durante l'esecuzione del programma. L'istruzione che permette ciò è costituita dalla parola "**INPUT**", seguita dall'elenco delle variabili il cui valore deve essere definito. Quando il calcolatore la incontra durante l'esecuzione, esso visualizza un punto interrogativo e rimane in attesa. L'utente può scrivere sul video il valore (o i valori, separati da virgole) da assegnare. Egli indicherà poi al calcolatore di riprendere l'elaborazione, premendo un opportuno tasto ("END LINE" o "RETURN" a seconda dei calcolatori). I valori forniti in tal modo devono essere in numero pari a quelli che compaiono nell'istruzione INPUT. Il computer li assegnerà alle variabili, nell'ordine con cui compaiono, in maniera analoga a quanto avviene con l'istruzione READ.

Il GWBASIC consente di inserire tra la parola INPUT e l'elenco delle variabili una stringa, seguita da ";" Essa viene visualizzata prima del punto interrogativo; pertanto la si utilizza in genere per fornire indicazioni atte a chiarire quali valori devono essere immessi. Ad esempio, nell'eseguire la linea di programma:

```
50 INPUT " Qual è il valore del momento flettente (in kgm) "; M
```

il calcolatore visualizza:

```
Qual è il valore del momento flettente (in kgm) ?
```

e resta in attesa del valore da assegnare alla variabile M, che rappresenta il momento flettente.

Lo stesso risultato può essere ottenuto premettendo all'istruzione INPUT una istruzione di output su video. Ad esempio:

```
40 PRINT " Qual è il valore del momento flettente (in kgm) ";
50 INPUT M
```

INPUT variabile-1 variabile-2

sul video compare un punto interrogativo per indicare la richiesta di dati;
i valori da assegnare alle variabili devono essere trasmessi da tastiera; occorre cioè scriverli sul video, separati da virgole, ed indicare al calcolatore di proseguire, premendo un apposito tasto ("END LINE" o "RETURN");
il calcolatore assegna i valori alle variabili in maniera analoga all'istruzione READ, e riprende l'elaborazione

INPUT stringa ; variabile-1 variabile-2 ... (GWBASIC)

sul video compare la stringa seguita da un punto interrogativo; si prosegue quindi analogamente a quanto sopra definito

4.10.5 - Confronto tra READ e INPUT.

Entrambe le modalità di ingresso analizzate, ottenibili con l'istruzione READ e con la INPUT, rispondono solo parzialmente ai requisiti elencati.

Nella lettura di dati all'interno del programma, il fatto che i valori siano presentati in un elenco non è, di per se, un modo per favorirne la comprensione. Le linee dei DATA possono però essere accompagnate da istruzioni di commento che indichino tutto ciò che si ritiene necessario per conferire ad esse sufficiente chiarezza. L'impostazione non è comunque adatta al generico utente, perchè comporta in effetti un intervento sul programma. Inoltre essa non è proprio utilizzabile nel caso di programmi "protetti", per i quali non è possibile modificare o listare le istruzioni.

Correggere eventuali errori nei dati è invece molto facile. Basta infatti richiamare la linea di programma che contiene il valore errato e modificarlo, così come si opera su una qualsiasi linea di programma.

I dati possono essere immediatamente riutilizzati, perchè non vengono persi nell'esecuzione. È inoltre possibile conservarli su memoria di massa per un uso futuro, memorizzando in essa l'intero programma con i dati contenuti.

La lettura di dati interattiva con l'istruzione INPUT si presenta molto più chiara e di semplice uso. Infatti è possibile illustrare ampiamente ogni valore richiesto, premettendo a ciascuna istruzione di ingresso una opportuna uscita sul video. Inoltre essa non comporta alcun intervento sul programma, ma solo il rispondere a domande che il computer pone; è quindi adatta anche all'uso da parte di personale non specializzato.

Non è invece possibile correggere eventuali errori nei dati. Ciascun valore, una volta trasmesso, viene accettato dal calcolatore, che prosegue l'elaborazione e non consente di tornare indietro.

Neanche la riutilizzazione immediata dei dati è direttamente possibile

Infatti i loro valori sono ancora presenti nella memoria centrale al termine della esecuzione, ma vengono azzerati quando si inizia una nuova elaborazione

In base a quanto esposto, l'istruzione INPUT appare adatta a programmi che richiedono una quantità modesta di dati L'istruzione READ è invece senz'altro preferibile per programmi più complessi, realizzati per uso proprio. Essa è inoltre adatta ai fini didattici, come la realizzazione degli esempi utilizzati nel presente testo, perchè mostra esplicitamente il valore assegnato ai dati in una esecuzione di prova.

Nel realizzare programmi che comportano una notevole mole di dati di ingresso e sono rivolti ad utenti generici, è possibile utilizzare ancora l'istruzione INPUT Per consentire correzioni e rielaborazioni occorrerebbe però realizzare di volta in volta uno specifico programma, che richieda espressamente all'utente quali modifiche effettuare Ciò innanzitutto comporterebbe una notevole fatica per il programmatore, e in secondo luogo richiederebbe all'utente l'acquisizione di due differenti tecniche, per l'immissione dei dati e per la loro correzione. Il ricorso all'input a maschera si presenta invece molto più agevole per il programmatore, una volta che ci si sia impadroniti di tale tecnica, e consente all'utente di affrontare allo stesso modo immissione o correzione dei dati

INGRESSO DATI CON "READ"

Chiarezza e semplicità d'uso:

- può essere reso comprensibile con frasi di commento;
- non adatto al generico utente perchè è un intervento sul programma;
- non utilizzabile in programmi protetti.

Correzione di errori nei dati:

- possibile

Riutilizzazione dei dati:

- riutilizzazione immediata possibile;
- riutilizzazione a lunga scadenza possibile se il programma viene memorizzato con tutti i dati

INGRESSO DATI CON "INPUT"

Chiarezza e semplicità d'uso:

- può essere reso ben comprensibile con opportune scritte;
- è facilmente utilizzabile da qualsiasi utente.

Correzione di errori nei dati:

- una volta trasmesso, un valore non è più modificabile

Riutilizzazione dei dati:

- i dati precedenti vengono dimenticati all'inizio di una nuova elaborazione.

4.11 - Istruzioni di uscita

4.11.1 - Generalità

Col termine "uscita" o "output" si intende il trasmettere informazioni, numeriche o alfanumeriche, dalla memoria centrale del calcolatore ad una unità di uscita (video, stampante), che le presenta su un apposito supporto (lo schermo, un foglio di carta). Anche la trasmissione di informazioni grafiche (a un plotter o ad un video grafico) è una operazione di uscita, ma esistono per essa delle apposite istruzioni, che verranno esaminate in seguito.

Usualmente, le informazioni in uscita vengono presentate in maniera sequenziale. Ogni nuovo valore viene, cioè, visualizzato o stampato in una posizione, dello schermo o della carta, che segue quella occupata dal valore precedente. Per tutti i calcolatori è anche possibile, almeno sul video, una uscita in posizioni indipendenti dall'ordine di esecuzione (non sequenziale). Un esempio di ciò lo si ha nella creazione di maschere per l'input. Le relative istruzioni non sono standardizzate e non vengono pertanto illustrate in questo paragrafo.

Il valore di una informazione può essere presentato in uscita così come è memorizzato (per esempio, nel caso di un numero reale, con tutte le cifre decimali che possiede) o in maniera diversa (con un numero prefissato di cifre). In questo secondo caso il valore viene detto "formattato" (dall'inglese *format*, cioè "formato"). La formattazione viene utilizzata per organizzare in maniera più chiara e leggibile i risultati di una elaborazione.

4.11.2 - Output non formattato

Nel BASIC ANSI, l'istruzione che consente di visualizzare valori numerici o alfanumerici è costituita dalla parola "**PRINT**" (cioè "stampa"), seguita da un elenco di costanti, variabili o espressioni, separate da virgole o punti e virgola. Il calcolatore visualizza i valori forniti esplicitamente, o calcolati come risultato delle espressioni, con una spaziatura che dipende dal simbolo usato come separatore nell'elenco. Il simbolo "," indica una spaziatura estesa; la riga è idealmente divisa in più campi, ed il valore viene allineato a sinistra nel primo campo libero (il numero e l'ampiezza dei campi dipende dal modello di computer). Il simbolo ";" indica invece una spaziatura compatta; il valore viene posto immediatamente a fianco del precedente (ogni valore numerico viene preceduto da uno spazio bianco se positivo o dal segno meno se negativo, e viene seguito da uno spazio bianco). Se l'elenco termina con "," o ";" i valori forniti da una successiva istruzione PRINT vengono messi in coda ai precedenti, sulla stessa riga. Ad esempio, le linee di programma:

```
10 PRINT 2,5+2^4,"AUTO","MOBILE"
20 PRINT -2;
30 PRINT -(5+2^4);"AUTO";"MOBILE"
```

causano la visualizzazione di:

```
2             21            AUTO          MOBILE
-2 -21 AUTOMOBILE
```

Nell'elenco che segue PRINT può essere compresa anche una funzione particolare, la funzione "**TAB( )**", che si utilizza per l'incolonnamento dei valori in uscita. Il suo parametro indica infatti la posizione del primo carattere di un valore da stampare. Ad esempio:

```
10 PRINT TAB(5);2;TAB(40);5+2^4;"AUTO";TAB(10);"MOBILE"
```

causa la visualizzazone di:

```
    2                                   21 AUTO
       MOBILE
```

Si noti che l'ultimo valore, "MOBILE", per essere posto a partire dalla colonna 10 è stato automaticamente riportato nella riga successiva.

In alcuni calcolatori, ad esempio il Commodore 64, la funzione TAB( ) indica invece quanti spazi lasciare tra un dato e l'altro; questo effetto è ottenuto in GWBASIC mediante la funzione "**SPC( )**".

Per ottenere l'uscita su stampante anzichè su video si usa, a seconda del tipo di BASIC, la stessa istruzione PRINT o istruzioni perfettamente analoghe a questa.

Il BASIC HP effettua una distinzione logica tra due tipi di unità: "printer" (cioè "stampante") e "CRT" (da *Cathode Ray Tube*, cioè "tubo a raggi catodici" e quindi "video") L'istruzione PRINT, già definita, indica una uscita sull'unità definita come printer. Si usa invece l'istruzione "**DISP**" (dal termine inglese *display*, cioè "visualizza"), perfettamente analoga alla precedente, per indicare una uscita sull'unità definita come CRT.

La denominazione printer e CRT non richiede però necessariamente una corrispondenza con le unità stampante e video. Esistono infatti due istruzioni, "**PRINTER IS**" e "**CRT IS**", che consentono di definire il codice dell'unità printer e dell'unità CRT. Ad esempio, i codici usati per il modello HP-85 sono:

1	per indicare il video;
2	per indicare la stampante termica;
701	per indicare una stampante meccanica.

Una qualunque unità di uscita può essere definita come printer o CRT. Per esempio, l'istruzione "PRINTER IS 1" definisce come unità printer il video. In tal modo l'istruzione PRINT fornirà un output su di esso e non sulla stampante.

Le istruzioni PRINTER IS e CRT IS consentono anche di indicare il numero di caratteri contenuti in ciascuna riga dell'unità di uscita. Così, l'istruzione "PRINTER IS 701,132" definisce come printer la stampante meccanica ed indica che in essa ciascuna riga può contenere 132 caratteri.

Anche il GWBASIC distingue stampante da video. L'istruzione PRINT indica una uscita sul video, mentre l'istruzione "**LPRINT**" indica l'uscita sulla stampante. A differenza del BASIC HP, questa distinzione è effettiva, cioè non è possibile utilizzare PRINT per uscita sulla stampante o viceversa

Per indicare il numero di caratteri contenuti in ciascuna riga del video o della stampante, si usano rispettivamente le istruzioni "**WIDTH**" (cioè "ampiezza" della riga) e "**WIDTH LPRINT**". Così, "WIDTH LPRINT 132" indica che ogni riga della stampante può contenere 132 caratteri.

USCITA DATI NON FORMATTATA

PRINT elenco grandezze

i valori indicati esplicitamente o calcolati vengono inviati alla unità di uscita

I singoli termini dell'elenco possono essere separati da:

, indica spaziatura estesa - il valore è allineato a sinistra nel primo campo libero;

; indica spaziatura compatta - il valore è posto immediatamente a fianco del precedente

TAB (espressione numerica)

Il valore fornito dall'espressione numerica (arrotondato) indica la posizione del primo carattere del valore da stampare

BASIC HP:

PRINT elenco grandezze

indica l'uscita su unità printer

DISP elenco grandezze

indica l'uscita su unità CRT

PRINTER IS codice-unità, numero-caratteri

indica qual è l'unità printer (stampante) e quanti caratteri possono essere contenuti in una sua riga

CRT IS codice-unità, numero-caratteri

indica qual è l'unità CRT (video) e quanti caratteri possono essere contenuti in una sua riga.

GWBASIC:

PRINT elenco grandezze

indica l'uscita sul video

LPRINT elenco grandezze

indica l'uscita sulla stampante

WIDTH numero-caratteri

indica quanti caratteri possono essere contenuti in una riga del video

WIDTH LPRINT numero-caratteri

indica quanti caratteri possono essere contenuti in una riga della stampante

4.11.3 - Sottoprogramma per la formattazione.

Il BASIC standard non prevede specifiche istruzioni per la formattazione dei valori di uscita. È però possibile realizzare un sottoprogramma di validità generale che, dato un qualsiasi valore numerico, ne effettui l'arrotondamento con un numero prefissato di cifre decimali. Per consentire l'esatto incolonnamento del valore in uscita, esso deve inoltre determinarne il numero di caratteri (cifre o segno) presenti prima del punto decimale

Per poter scrivere questo sottoprogramma, occorre innanzitutto analizzare le operazioni da effettuare per ottenere il risultato voluto

I valori da fornire in ingresso al sottoprogramma possono essere indicati mediante i seguenti nomi simbolici:

U numero da formattare;
N numero di cifre decimali richieste

In BASIC non esiste una funzione che effettui l'arrotondamento di un valore, ma esiste una funzione (INT) che ne determina la parte intera, cioè lo approssima per difetto all'unità. Per arrotondare un valore U all'unità, basta prendere la parte intera del numero incrementato di 0 5: INT(U+ 5) Se lo si vuole arrotondare con N cifre decimali, occorre pri-

ma moltiplicarlo per 10^N, arrotondarlo all'unità e infine dividerlo nuovamente per 10^N : INT(U * 10^N + .5) / 10^N. Per esempio, sia U=25.36892 e N=2. Moltiplicando il valore per 100 (10^2) si ottiene 2536.892; aggiungendo 0.5 si ha 2537.392, la cui parte intera è 2537; dividendo per 100 si ottiene infine il valore richiesto: 25.37.

Per determinare il numero di cifre intere di un numero V, maggiore dell'unità, se ne può calcolare il logaritmo decimale; nel BASIC HP ciò è possibile con la funzione LGT: LGT(V). Il numero cercato è pari alla parte intera del valore così ottenuto, incrementata di 1: INT(LGT(V))+1. Ad esempio, se V=25.37 il suo logaritmo vale 1.4043205; la sua parte intera è 1, ed incrementando si ottiene il valore 2. Se il numero V è negativo, basta effettuare l'operazione sul suo valore assoluto: INT(LGT(ABS(V)))+1. Molti linguaggi non posseggono la funzione LGT, ma consentono di ottenere lo stesso risultato mediante il logaritmo naturale LOG. Per le proprietà dei logaritmi si ha infatti: LGT(x)=LOG(x)/LOG(10). Il numero cercato sarà quindi fornito dall'espressione INT(LOG(ABS(V))/LOG(10))+1.

Per i numeri inferiori in valore assoluto all'unità questo calcolo non è necessario; essi sono infatti usualmente rappresentati dal calcolatore senza cifre intere (nemmeno lo zero iniziale). Il valore 0 è invece rappresentato sempre così come appare, cioè con una cifra intera.

Poichè il BASIC premette a ciascun valore uno spazio bianco se esso è positivo, o il segno meno se negativo, il numero di caratteri presenti prima del punto decimale sarà pari al numero di cifre intere più 1.

Le operazioni esaminate consentono pertanto di determinare, come valori di uscita del sottoprogramma, le seguenti grandezze:

V valore di U arrotondato con N cifre decimali;
I numero di caratteri (cifre o segno) presenti prima del punto decimale.

Si riporta di seguito la codifica del sottoprogramma descritto e la sua applicazione all'esempio, già esaminato, di tabellazione di una funzione. I risultati ottenuti in tal modo possono essere confrontati con quelli riportati nel paragrafo 4.9.1, non formattati

```
 10 REM          TABELLAZIONE DI UNA FUNZIONE
 20 REM
 30 REM Funzione:     Y = -7 * X^3 + SQR (X^2 + 10)
 40 REM
 50 REM Variabili: A,B    estremi dell intervallo
 60 REM            D      passo di incremento
 70 REM            X      ascissa corrente
 80 REM            Y      valore di f(X)
 90 REM
200 READ A,B,D
210 X=A
```

```
220 IF X>B THEN 270
230 Y=-(7*X^3)+SQR (X^2+10)
231 U=Y
232 N=5
233 GOSUB 1000
234 V1=V
235 I1=I
236 U=X
237 N=2
238 GOSUB 1000
240 PRINT TAB (5-I);V;TAB (25-I1);V1
250 X=X+D
260 GOTO 220
270 STOP
280 DATA 0,1, 1
800 REM
810 REM        SOTTOPROGRAMMA PER LA FORMATTAZIONE
820 REM
830 REM Variabili          U    numero da formattare
840 REM di ingresso:       N    numero di cifre richieste
850 REM                         dopo la virgola
860 REM
870 REM Variabili          V    valore di U arrotondato
880 REM di uscita:              con N cifre decimali
890 REM                    I    numero di cifre di V
900 REM                         prima della virgola
910 REM
1000 V=INT (U*10^N+.5)/10^N
1010 IF V=0 THEN 1040
1020 IF ABS (V)<1 THEN 1060
1030 GOTO 1080
1040 I=2
1050 GOTO 1090
1060 I=1
1070 GOTO 1090
1080 I=INT (LGT (ABS (V)))+2
1090 RETURN
```

```
0                3  16228
   1             3  15686
   2             3  1126
   3             2  98748
   4             2  73948
   5             2  32656
   6             1  7067
   7                83783
   8             -  3221
   9            - 1 81514
1               - 3 68338
```

Come si può notare, l'arrotondamento e l'incolonnamento del punto decimale sono stati realizzati correttamente. Il numero di cifre decimali stampate non è però sempre pari a quanto richiesto, poichè eventuali zeri

finali sono omessi. Inoltre, in alcuni calcolatori la divisione per 10^N può, in casi particolari, fornire un risultato approssimato (per esempio 25.370000001 anzichè 25.37), vanificando il lavoro di arrotondamento.

Può quindi essere preferibile operare in maniera diversa, trasformando il valore INT(U*10^N+.5) in una stringa V$ ed inserendo espressamente il punto decimale nella posizione corretta. Per far ciò, si possono memorizzare gli ultimi N caratteri in una stringa D$ (il nome simbolico D$ richiama l'attributo "cifre decimali") e la parte restante in I$ ("parte intera"). Se le cifre sono meno delle cifre decimali richieste, la parte intera conterrà il segno e la cifra zero, mentre alla parte decimale verranno aggiunti gli zeri necessari per il completamento. Ad esempio, se U=.002536892 ed N=6, INT(U*10^N+.5) vale 2537. Sarà quindi I$="0" e D$="002537".

Si riporta di seguito la codifica di questa versione del sottoprogramma ed i risultati ottenuti utilizzandolo nella tabellazione di una funzione.

```
 10 REM                    TABELLAZIONE DI UNA FUNZIONE
 20 REM
 30 REM Funzione:  Y = -7 * X^3 + SQR (X^2 + 10)
 40 REM
 50 REM Variabili: A B     estremi dell intervallo
 60 REM            D       passo di incremento
 70 REM            X       ascissa corrente
 80 REM            Y       valore di f(X)
 90 REM
200 READ A B D
210 X=A
220 IF X>B THEN 270
230 Y=-(7*X^3)+SQR (X^2+10)
231 U=Y
232 N=5
233 GOSUB 1000
234 U1$=U$
235 I1=I
236 U=X
237 N=2
238 GOSUB 1000
240 PRINT TAB (5-I);U$;TAB (25-I1);U1$
250 X=X+D
260 GOTO 220
270 STOP
280 DATA 0 1 .1
700 REM
710 REM SOTTOPROGRAMMA PER LA FORMATTAZIONE (con stringhe)
720 REM
730 REM Variabili          U     numero da formattare;
740 REM di ingresso:       N     numero di cifre richieste dopo la
750 REM                          virgola
760 REM
770 REM Variabili          U$    stringa che contiene il valore di U
780 REM di uscita:               arrotondato con N cifre decimali;
790 REM                    I     numero di cifre di V prima della
```

```
800 REM                          virgola
810 REM
820 REM Variabili        V       valore arrotondato del numero
830 REM interne:                 moltiplicato per 10^N;
840 REM                  V$      stringa che contiene V senza segno;
850 REM                  V0      lunghezza di V$;
860 REM                  I$      stringa che contiene la parte
870 REM                          intera di U$, col segno;
880 REM                  D$      stringa che contiene la parte
890 REM                          decimale di U$;
900 REM                  D0      lunghezza di D$;
910 REM                  C       contatore
920 REM
1000 V=INT (U*10^N+ .5) !                  in GWBASIC :
1010 V$=VAL$ (ABS (V)) !                   V$= STR$(ABS(V))
1020 V0=LEN (V$)
1030 IF V<0 THEN 1060
1040 I$=" "
1050 GOTO 1070
1060 I$="-"
1070 IF V0<= N THEN 1110
1080 I$=I$&V$[1,V0-N] !                    I$=I$+LEFT$(V$ V0-N)
1090 D$=V$[V0-N+1,V0] !                    D$=RIGHT$(V$,N)
1100 GOTO 1130
1110 I$=I$&"0" !                           I$=I$+"0"
1120 D$=V$
1130 I=LEN (I$)
1140 D0=LEN (D$)
1150 U$=I$& " " !                          U$=I$+ " "
1160 IF D0>= N THEN 1200
1170 FOR C=1 TO N-D0
1180 U$=U$& "0" !                          U$=U$+"0"
1190 NEXT C
1200 U$=U$&D$ !                            U$=U$+D$
1210 RETURN
```

0 00	3 16228
0 10	3 15686
0 20	3 11260
0 30	2 98748
0 40	2 73948
0 50	2 32656
0 60	1 70670
0 70	0 83783
0 80	-0 32210
0 90	-1 81514
1 00	-3 68338

4.11.4 - Output formattato.

Sia il BASIC HP che il GWBASIC presentano istruzioni che consentono una diretta formattazione dei valori in uscita. Nonostante le evidenti analogie, le differenze sono tali che è preferibile esaminare separatamente i due linguaggi.

A) BASIC HP.

L'istruzione di formattazione è costituita dalle parole "**PRINT USING**", o "**DISP USING**", seguite da una stringa di formattazione, da un ";" e dall'elenco di valori o espressioni da stampare. I valori dell'elenco possono essere separati indifferentemente da virgole o punti e virgola, senza che ciò abbia effetto sulla spaziatura dell'output, che è totalmente controllata dalla stringa.

La stringa di formattazione contiene una o più indicazioni di formato, separate da "," o da "/"; quest'ultima indica il passaggio ad una nuova riga (andare a capo). Ciascuna indicazione di formato è composta da caratteri che forniscono convenzionalmente istruzioni su come manipolare il valore.

I caratteri di più frequente uso sono:

D indica la posizione di una cifra;
Z indica la posizione di una cifra a sinistra del punto decimale; se tale cifra non è presente, viene stampata la cifra 0 (zero);
. indica la posizione del punto decimale;
E indica che il valore numerico deve essere presentato in notazione esponenziale;
A indica la posizione di un carattere;
X indica uno spazio bianco tra due valori;
() indica che lo stesso formato è ripetuto per più valori consecutivi.

Così ad esempio con le istruzioni:

```
10 PRINT USING "DDDD.DDE,DDDD.DD,DDDZ DD";- 4648,- 4648,- 4648
20 PRINT USING  AAA,XX,DDD.DD,X,AA";"M =",12 24,"tm"
30 PRINT USING  AAAAAAAAAA,3(XXX,DDZ DDD)";"Risultati:",32 427,- 2, 4758
```

si ottiene:

```
-464 80E-003    - 46    -0 46
M =    12 24 tm

Risultati:        32 427      -0 200      0 476
```

Per indicare più caratteri uguali consecutivi se ne può scrivere uno solo, premettendogli una cifra che indica il numero di ripetizioni. Le istruzioni precedenti possono quindi essere scritte:

```
10 PRINT USING "4D.2DE,4D.2D,3DZ.2D";-4648,-4648,-4648
20 PRINT USING "3A,2X,3D.2D,X,2A";"M =";12.24;"tm"
30 PRINT USING "10A,3(3X,2DZ.3D)";"Risultati:",32.427,-2,4758
```

La stringa di formattazione può anche essere definita su una differente linea di programma, mediante l'istruzione "**IMAGE**". In tal caso, dopo la parola "PRINT USING" si indicherà il numero della linea che contiene la stringa. Le seguenti linee di programma equivalgono quindi alle istruzioni precedenti:

```
10 PRINT USING 60;-.4648,-.4648,-4648
20 PRINT USING 40;"M =";12.24,"tm"
30 PRINT USING 50;"Risultati:";32.427,-2,4758
40 IMAGE 3A,2X,3D.2D,X,2A
50 IMAGE 10A,3(3X,2DZ.3D)
60 IMAGE 4D.2DE,4D.2D,3DZ.2D
```

B) GWBASIC

L'istruzione di formattazione, "**PRINT USING**" o "**LPRINT USING**", è complessivamente analoga a quella del BASIC HP. I valori dell'elenco devono essere separati da punto e virgola, ed è possibile usare tale simbolo alla fine di un elenco di valori per indicare che gli output successivi devono essere messi in coda a questi e non su una nuova riga. Differenti sono invece i caratteri che forniscono le indicazioni su come manipolare i valori. Quelli di uso più frequente sono:

#	indica la posizione di una cifra; viene stampata sempre almeno una cifra intera (eventualmente uno zero);
.	indica la posizione del punto decimale;
^^^^	indica che il valore numerico deve essere presentato in notazione esponenziale;
!	indica di stampare solo un carattere della stringa;
\ \	indica quanti caratteri della stringa devono essere stampati (tanti quanti sono gli spazi bianchi tra le sbarre più due);
spazio bianco	indica uno spazio bianco tra due valori

Così, gli stessi risultati dell'esempio visto per il BASIC HP si possono ottenere con le seguenti istruzioni:

```
10 PRINT USING "####.##";-4648;
12 PRINT USING "####.##";-4648;-4648
20 PRINT USING "\ \";"M =";
22 PRINT USING "#### ##";12.24;
24 PRINT USING "\ \";"tm"
30 PRINT USING "\ \";"Risultati:";
32 PRINT USING "#### ##";32.427;-2;4758
```

4.12 - Istruzione di fine dell'esecuzione.

È necessario ordinare esplicitamente al calcolatore di interrompere l'esecuzione quando non vi sono ulteriori elaborazioni da svolgere Ciò avviene mediante l'istruzione "**STOP**" Essa può essere posta in un qualunque punto del programma, e può essere presente in più punti diversi (per esempio, se si effettua un controllo dell'attendibilità dei valori di ingresso, può essere inserita per evitare il passaggio alla fase di calcolo) Nella programmazione strutturata, che prevede una sequenza di blocchi ad un ingresso ed una uscita, è però preferibile che ve ne sia una sola, a conclusione dell'ultimo blocco nell'ordine logico di esecuzione

CAPITOLO QUINTO

OPERAZIONI SULLA MEMORIA DI MASSA

5.1 - Struttura del supporto magnetico

Tra i supporti magnetici nei quali conservare stabilmente informazioni, i più usati dai personal computer sono i dischi flessibili, e ad essi si farà quindi implicitamente riferimento nel seguito. Molti dei concetti che si esporranno sono comunque di validità generale, estendibili sia ai dischi rigidi che ai nastri.

Il disco flessibile è un disco di plastica la cui superficie è rivestita di un ossido di ferro magnetico. Le dimensioni più usuali del suo diametro sono 5.125 pollici (130 mm) o, più raramente, 3.5 pollici (89 mm).

I dati sono caricati sotto forma di cifre binarie, rappresentate sul disco da inversioni di flusso magnetico. Essi vengono letti o memorizzati dalle testine dell'unità disco. Si parla di unità disco (e di dischi flessibili) a singola faccia ("SS", dall'inglese *single side*) oppure a doppia faccia ("DS", cioè *double side*), in base alla possibilità di memorizzare informazioni su una sola faccia del disco o su entrambe. Si distingue inoltre tra dischetti a singola densità ("SD", *single density*), doppia densità ("DD", *double density*) o alta densità ("HD", *high density*), in base alla quantità di informazioni che possono essere conservate per unità di superficie.

Lo strato magnetico di un disco vergine, cioè mai usato, si presenta uniforme, indifferenziato. Perchè esso possa venire usato occorre crearvi una organizzazione interna, cioè una suddivisione in blocchi, mediante una operazione detta "formattazione" o "inizializzazione". La lettura o la memorizzazione di dati da parte della testina non viene infatti effettuata operando sul singolo bit o byte, bensì a blocchi di dimensione fissa, che vengono detti "record fisici". Il termine "record" indica in generale un insieme di informazioni registrate su un supporto. La parola "fisico" è usata per indicare che tali blocchi hanno una corrispondenza effettiva, strutturale, nel disco. In contrapposizione a questo, si userà in seguito il termine "record logico" per indicare un insieme di informazioni di dimensione definita dal programmatore indipendentemente dalla struttura fisica del supporto.

L'organizzazione interna dei dischi varia da computer a computer. Ad esempio, i dischi flessibili utilizzati dai calcolatori HP-86 e HP-87 sono suddivisi in 35 "tracce" o "piste" concentriche e in 16 "settori" (fig. 37), in modo da ottenere $35 \times 16 = 560$ record fisici per faccia, ciascuno dei quali

contiene 256 bytes. La loro capacità complessiva è quindi 560×2×256=286720 bytes. Completamente differente è la formattazione effettuata dall'MS-DOS (*MicroSoft Disk Operating System*), sistema operativo usato con il GWBASIC. Esso infatti consente un massimo di 40 tracce e 9 settori, cioè 360 record fisici (ciascuno di 512 bytes) per faccia, per un totale di 368640 bytes. Di conseguenza i dischi formattati HP non vengono proprio letti dai calcolatori IBM, e viceversa.

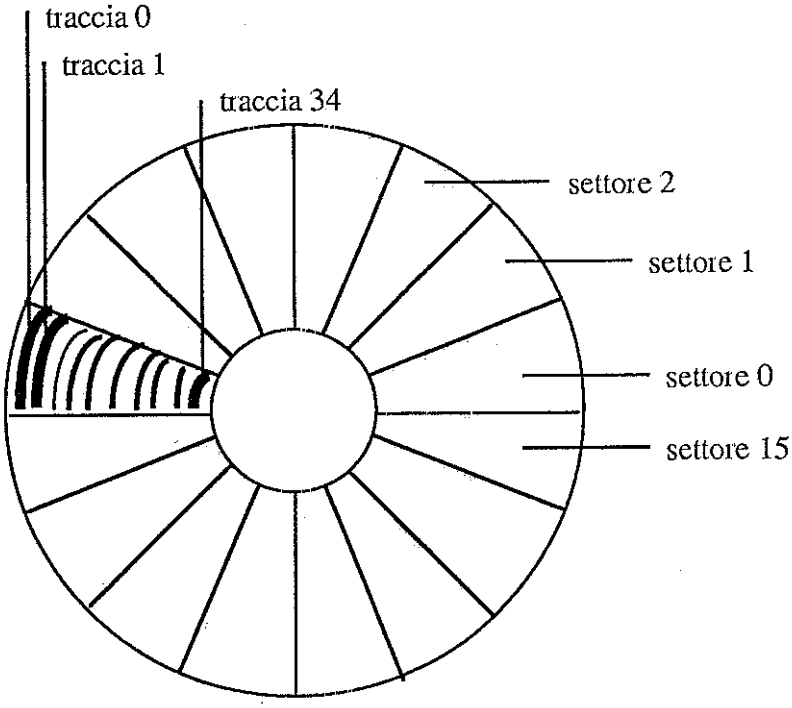


Fig. 37

RECORD	insieme unitario di informazioni registrate su un supporto magnetico (disco o nastro)
RECORD FISICO	dipende dalla struttura del supporto (per esempio nei dischi flessibili HP è composto da 256 bytes)
RECORD LOGICO	è definito dal programmatore, indipendentemente dalla struttura fisica del supporto
FORMATTAZIONE o **INIZIALIZZAZIONE**	definizione della organizzazione interna del disco mediante la sua suddivisione in record fisici.

5.2 - I files.

Si definisce "file" un insieme omogeneo di informazioni registrato su un supporto, in una sequenza ordinata di records. Esso può venire manipolato dall'utente per mezzo di comandi logici, anche ignorando i dettagli su come l'informazione sia fisicamente organizzata sull'unità di memorizzazione.

Un file è denominato "program file" (file di programma) quando contiene programmi scritti in linguaggio simbolico (nei personal computer quasi sempre in BASIC, raramente in ASSEMBLER o FORTRAN) o in linguaggio macchina (cioè direttamente eseguibili). Un insieme di files di questo tipo viene denominato "libreria di programmi".

In alternativa, esso viene chiamato "data file" (file dati) quando contiene dati, eventualmente organizzati in blocchi (records logici). Un insieme di files dati e di procedure di accesso alle informazioni in essi contenute viene indicato col termine inglese "data base", cioè "base di dati".

Ciascun file è individuato mediante un nome simbolico. Nei personal HP questo può essere costituito da un massimo di 6 o 10 caratteri, a seconda del modello; l'MS-DOS consente invece nomi di 8 caratteri. Il nome di un file viene in genere scelto in modo da richiamare alla mente il suo contenuto. Così, ad esempio, un file che contenga un programma per la risoluzione di travi continue può essere identificato con il nome "TRAVI".

Una seconda informazione necessaria per caratterizzare un file è il tipo di contenuto (dati, programmi in BASIC, ecc.). Nei calcolatori HP questo è individuato direttamente dal sistema operativo, in base al genere di istruzione usata per la memorizzazione. Nell'MS-DOS è invece definito mediante una *estensione* del nome, ottenuta aggiungendo ad esso il simbolo " " e tre caratteri. Alcune estensioni hanno un significato standard, automaticamente riconosciuto dall'MS-DOS. Ad esempio, BAS indica un programma in BASIC, COM un file di comandi, EXE un programma immediatamente eseguibile. Un programma in BASIC per la risoluzione di travi continue potrà pertanto essere denominato "TRAVI.BAS". In altri casi, l'estensione può essere scelta arbitrariamente dall'utente, in modo da richiamarne il contenuto (ad esempio TXT per testi, DAT per dati).

Tutte le informazioni necessarie al sistema operativo per individuare i files memorizzati in un disco flessibile (nome, tipo, nonchè dimensione ed indirizzo), sono contenute in un gruppo di records del disco denominato "directory" o "catalogo dei files" e possono essere visualizzate anche dall'utente.

Nei personal computer HP ciò è ottenuto con l'istruzione "**CAT**". Le informazioni in tal modo visualizzate possono essere per esempio, nel caso di un HP-87:

```
[ Volume ]: TEXT10
  Name          Type          Bytes          Recs
  TRAVI         PROG            256            40
  datTRAVI      DATA            256            12
  PCintr6       DATA           1683            38
                NULL            256             4
  UTIL/1        BPGM            256             6
  VERIFICA      PROG            256            14
```

La prima informazione è il *volume* del disco, cioè un nome simbolico (nell'esempio "TEXT10") che individua l'intero disco flessibile.

Per ogni file sono poi riportate quattro indicazioni:

Name	nome del file (costituito da un massimo di 10 caratteri);
Type	tipo del file, cioè del suo contenuto; esso può essere: PROG se contiene un programma in BASIC; BPGM se contiene un programma binario; DATA se contiene dati; NULL se è vuoto;
Bytes	numero di bytes contenuti in ciascun record del file;
Recs	numero di records, fisici o logici, del file

L'MS-DOS consente un risultato analogo con l'istruzione "**DIR**". Si ottiene ad esempio:

```
 Volume in drive A is TEXT10
 Directory of A:\

TRAVI    BAS     10217     9-02-86   6:41p
DATTRAVI DAT      2941    10-09-86  10:22a
PCINTR6  TXT     63825    10-09-86   2:45p
UTIL/1   BIN      1518    10-15-86   7:31p
VERIFICA BAS      3426    10-16-86   4:20p
    5 File(s)     254756 bytes free
```

Le informazioni base, contenute nelle prime tre colonne, sono analoghe a quelle dell'HP: nome, tipo, dimensione complessiva del file (in bytes). Viene inoltre indicata la data e l'ora in cui ciascun file è stato generato o aggiornato per l'ultima volta, nonchè l'indicazione della quantità di memoria ancora disponibile sul disco

Un risultato più sintetico è fornito dall'istruzione "**FILES**" del GWBASIC, che fa visualizzare solamente nome ed estensione dei files:

```
A:\
TRAVI.BAS        DATTRAVI.DAT     PCINTR6.TXT      UTIL/1.BIN
VERIFICA.BAS
  254756 Bytes free
```

A parte le differenze puramente formali, è importante sottolineare alcune differenze sostanziali che si riscontrano tra l'ambiente HP e quello IBM (MS-DOS)

Innanzitutto, nei calcolatori HP serie 80 il sistema operativo è residente in memoria, su ROM, e contiene, come parte inscindibile, il linguaggio BASIC Non vi sono pertanto distinzioni tra istruzioni del sistema operativo ed istruzioni del BASIC HP.

L'MS-DOS, invece, è residente su disco e può, a richiesta, richiamare e mandare in esecuzione il GWBASIC, che è un programma in linguaggio macchina, anch'esso su disco I due "ambienti", MS-DOS e GWBASIC, sono quindi separati, e ciascuno di essi ha un proprio distinto insieme di comandi, che, a volte, possono dare effetti analoghi (come nel caso delle istruzioni DIR dell'MS-DOS e FILES del GWBASIC)

In secondo luogo, il sistema operativo HP considera i files come costituiti da una sequenza di record ordinata sulla base dell'organizzazione effettuata nella formattazione I files, disposti uno appresso all'altro, hanno necessariamente dimensioni fisse. Se infatti se ne volesse espandere uno, occorrerebbe utilizzare i records successivi, che sono già occupati da un altro file Inoltre, se un file viene cancellato, lo spazio prima occupato non scompare, ma rimane disponibile per altri file come file vuoto, cioè di tipo NULL. L'eliminazione di questi spazi può essere ottenuta mediante l'istruzione "**PACK**" che compatta le informazioni andando a ricopiarle da un record ad un altro.

L'MS-DOS, invece, non si preoccupa di assegnare ad un file un insieme sequenziale di record, poichè possiede una struttura a puntatori che consente di indicare quale record segue "logicamente" un altro, indipendentemente dall'ordine fisico. Questa organizzazione è più complessa, ma indubbiamente più efficace per l'utente, perchè consente l'automatica espansione o la totale cancellazione di files

FILE	insieme omogeneo di informazioni registrato su un supporto in una sequenza ordinata di record
PROGRAM FILE	file che contiene programmi scritti in linguaggio simbolico o in linguaggio macchina
LIBRERIA DI PROGRAMMI	un insieme di program files
DATA FILE	file che contiene dati.
DATA BASE	insieme di data files e di procedure di accesso alle informazioni in essi contenute

DIRECTORY gruppo di records del disco che contiene informazioni relative ai files in esso memorizzati

NOME DEL FILE	è costituito da un insieme di caratteri (fino a 6 o 10 negli HP fino a 8 per l'MS-DOS)
TIPO DEL FILE	nei calcolatori HP è definito automaticamente dal sistema operativo; nell'MS-DOS è indicato da una estensione del nome (esempio: nome.BAS)

VISUALIZZAZIONE DELLA DIRECTORY

nel BASIC HP:	istruzione "CAT";
nell'MS-DOS:	istruzione DIR";
nel GWBASIC:	istruzione FILES

5.3 - Operazioni comuni a tutti i tipi di file

Le operazioni che possono essere effettuate su tutti i files, indipendentemente dal loro tipo, sono due: cancellazione e cambiamento di nome.

Nel BASIC HP la prima operazione viene effettuata mediante il comando "**PURGE** *nome-file*" La seconda con "**RENAME** *vecchio-nome* **TO** *nuovo-nome*" Pertanto eseguendo le istruzioni:

```
PURGE "datTRAVI"
RENAME "PCintr6" TO "Cap 6"
```

e visualizzando quindi la directory col comando CAT, per vederne l'effetto, si ottiene:

```
[ Volume ]: TEXT10
  Name          Type          Bytes          Recs
  TRAVI         PROG            256            40
                NULL            256            12
  Cap 6         DATA           1683            38
                NULL            256             4
  UTIL/1        BPGM            256             6
  VERIFICA      PROG            256            14
```

Nel GWBASIC le istruzioni equivalenti sono "**KILL** *nome-file*" e "**NAME** *vecchio-nome* **AS** *nuovo-nome*". Pertanto con le istruzioni:

```
KILL "DATTRAVI.DAT"
NAME "PCINTR6.TXT" AS "CAP-6.TXT
FILES
```

si ottiene:

```
A:
TRAVI.BAS       CAP-6.TXT       UTIL/1.BIN      VERIFICA.BAS
     259828 Bytes free
```

Cancellazione di un file

BASIC HP	PURGE nome-file
GWBASIC	KILL nome-file

Cambio di nome a un file

BASIC HP	RENAME vecchio-nome TO nuovo-nome
GWBASIC	NAME vecchio-nome AS nuovo-nome

5.4 - Operazioni su file di programma

Tre sono le operazioni fondamentali che possono essere effettuate su files di programmi.

Innanzitutto, la memorizzazione del programma, cioè il ricopiarlo dalla memoria del calcolatore, che non viene modificata, alla memoria di massa. L'istruzione che lo impone è costituita nel BASIC-HP dalla parola "**STORE**", cioè "immagazzina", seguita dal nome che si vuole assegnare al file. Nel GWBASIC la parola chiave è invece "**SAVE**", cioè "salva" Se già esiste nel disco un file di programma con lo stesso nome, questo viene cancellato e sostituito col programma che si memorizza

L'operazione inversa è il caricamento del programma, cioè il ricopiarlo dalla memoria di massa alla memoria del calcolatore. Se in questa c'è già un programma, esso viene cancellato prima di caricare il nuovo, insieme agli eventuali dati presenti. L'istruzione che lo impone è costituita dalla parola "**LOAD**" (cioè "carica"), seguita dal nome del file.

Infine, una operazione utile soprattutto nell'affrontare problemi molto complessi è il concatenamento di file di programmi, ottenuto mediante l'istruzione "**CHAIN** *nome-file*". Essa consiste nel caricare e mandare immediatamente in esecuzione il programma col nome indicato In tal modo, un problema che richiederebbe un programma molto esteso ed un eccessivo ingombro della memoria centrale può essere risolto mediante più programmi, di dimensioni minori, che si richiamano l'un l'altro Ci potrà, ad esempio, essere un primo programma che legge i dati, uno o più programmi successivi che li elaborano ed un programma finale che stampa i risultati.

Il concatenamento, come il caricamento, comporta l'annullamento di tutte le informazioni presenti nella memoria centrale del calcolatore. È però possibile indicare quali dati devono essere salvati, perchè comuni ai programmi concatenati, elencando i nomi delle variabili nell'istruzione "**COM**" del BASIC-HP (o "**COMMON**" del GWBASIC) Un'alternativa, ugualmente possibile ed in alcuni casi preferibile, consiste nel conservare su memoria di massa le informazioni da salvare, per richiamarle in momenti e da programmi successivi

Memorizzazione di un programma

BASIC HP	STORE nome-file
GWBASIC	SAVE nome-file

Caricamento di un programma

LOAD nome-file

Concatenamento di un programma

CHAIN nome-file

Per indicare che alcune variabili sono comuni a più programmi

BASIC HP	COM elenco-variabili
GWBASIC	COMMON elenco-variabili

5.5 - I files dati

Il BASIC prevede l'uso di due tipi di file dati: sequenziali e ad accesso diretto (o random).

Un file è detto sequenziale quando per operare su di esso è necessario rispettare l'ordine dei record fisici che lo compongono. Per leggere una informazione contenuta ad esempio nel dodicesimo record, occorre quindi aver già letto gli undici record precedenti. I valori sono contenuti nel file in maniera compatta, uno appresso all'altro, con un ingombro complessivo che è la somma dei loro ingombri individuali. Una loro distinzione in blocchi logici non è essenziale; come si vedrà in seguito, è possibile nel GWBASIC mentre non esiste proprio nel BASIC HP.

In un file ad accesso diretto, al contrario, i dati sono raggruppati in record logici, tutti di una stessa lunghezza, definita dal programmatore. Nel caso di valori di ingombro variabile, la dimensione del record sarà commisurata al massimo ingombro necessario, e vi potrà quindi essere uno spreco di spazio per quei valori che ne richiederebbero una quantità minore. Ogni record ha un numero di identificazione, ed è quindi possibile accedere alle informazioni contenute in uno di essi senza dover passare prima per i record precedenti.

Entrambi i tipi di file presentano vantaggi e svantaggi. Il tipo da usare dipende dalle esigenze pratiche del programma su cui si sta lavorando.

Indipendentemente dalle richieste dell'utente, che può voler esaminare anche un singolo dato, le informazioni vengono sempre prelevate dal disco a blocchi, che vengono ricopiati in un'area di memoria centrale detta "buffer", che funge da deposito temporaneo. Analogamente, quando si da il comando di scrivere qualcosa sul disco, l'informazione viene prima posta nel buffer e solo in un momento successivo effettivamente trascritta, insieme alle altre contenutevi, nella memoria di massa.

Il calcolatore gestisce più buffer, cioè può essere "collegato" contemporaneamente con più file dati. Il BASIC HP ne prevede 10, mentre nel GWBASIC il loro numero può essere definito nel caricare l'interprete GWBASIC (se non specificato, viene assunto pari a 3)

Per poter leggere o scrivere su un file dati, la prima operazione da effettuare è dunque la sua "apertura", cioè il definire quale sia il buffer da utilizzare. Nel BASIC HP ciò si ottiene con l'istruzione:

ASSIGN# *numero-buffer* **TO** *nome-file*

che letteralmente vuol dire: ASSEGNA il buffer avente il numero indicato AL file dati di cui si riporta il nome. Nel GWBASIC l'istruzione corrispondente è:

OPEN *modo* ,# *numero-buffer* , *nome-file* , *lunghezza-record*

Essa contiene, oltre al numero del buffer e al nome del file, anche l'indicazione del modo di operare (lettura, scrittura o ad accesso diretto), che verrà analizzato più dettagliatamente nei paragrafi successivi Contiene inoltre, solo nel caso di file ad accesso diretto, la lunghezza dei record logici che lo compongono.

Se il file che si vuole utilizzare è nuovo, cioè non esiste nella directory del disco, in GWBASIC è la stessa operazione di apertura che ne provoca automaticamente l'inserimento del nome nel catalogo. Operando col BASIC HP, invece, a causa della struttura rigida che in esso hanno i file, occorre definire preliminarmente il numero di record da cui esso è composto e la lunghezza degli eventuali record logici. A tal fine si utilizza l'istruzione:

CREATE *nome-file* , *numero-records* , *bytes-per-record*

Una volta terminate le operazioni su un file, è necessario effettuarne la "chiusura", cioè rilasciare il buffer assegnatogli. La chiusura è fondamentale soprattutto quando si è in fase di scrittura, perchè impone in tal caso che il contenuto del buffer venga ricopiato nel file prima del suo rilascio; se invece si spegne il calcolatore mentre un file è ancora aperto, le ultime informazioni ad esso inviate non vengono ricopiate e sono quindi perse. Dopo averla effettuata, è possibile riutilizzare lo stesso buffer per altri file o, nel GWBASIC, riaprire lo stesso file con modalità diversa. Le istruzioni di chiusura, sostanzialmente analoghe nel BASIC HP e nel GWBASIC, sono rispettivamente:

ASSIGN# *numero-buffer* **TO** *

CLOSE # *numero-buffer*

FILE SEQUENZIALE	quando è possibile effettuare operazioni di lettura o scrittura su un record solo dopo aver operato su tutti i records che lo precedono.
FILE AD ACCESSO DIRETTO (o RANDOM)	quando è possibile effettuare operazioni di lettura o scrittura su un record indipendentemente dai records che lo precedono

BUFFER	area della memoria centrale utilizzata come deposito temporaneo nelle operazioni di input e output con la memoria di massa

Apertura di un file dati

```
BASIC HP    ASSIGN# numero-buffer TO nome-file
GWBASIC     OPEN modo , # num-buffer , nome-file , lungh-record
```

Creazione di un file dati

```
BASIC HP    CREATE nome-file, numero-records, lunghezza-record
```

Chiusura di un file dati

```
BASIC HP    ASSIGN#numero-buffer TO *
GWBASIC     CLOSE #numero-buffer
```

5.6 - Operazioni su file sequenziali

Le differenze esistenti tra BASIC HP e GWBASIC nella gestione di file sequenziali non sono eccessive, ma è comunque preferibile esaminare separatamente i due linguaggi

A) BASIC HP

Nell'aprire un file sequenziale non è necessario specificare quale tipo di operazione (lettura o scrittura) si vuole effettuare su esso In ogni momento, infatti, il computer individua la posizione corrente all'interno del file per mezzo di un puntatore, ed a partire da questa effettua la successiva operazione di input o output richiesta. La distinzione tra queste due modalità deve però ovviamente essere mantenuta dal programmatore.

I valori delle informazioni, numeriche o alfanumeriche, sono conservati su disco con una codifica che consente al calcolatore di distinguerli l'uno dall'altro e individuarne il tipo.

La scrittura di dati è ottenuta con l'istruzione:

PRINT# *numero-buffer* , *elenco-grandezze*

L'elenco può contenere costanti, variabili o espressioni, separate tra loro da virgole. I valori così forniti vengono memorizzati in sequenza; in coda all'ultimo viene messa l'indicazione di fine del file

La lettura è ottenuta con l'istruzione:

READ# *numero-buffer* , *elenco-variabili*

I valori vengono prelevati dal disco e conservati nelle variabili elencate. In caso di discordanza di tipo tra un valore e la variabile corrispondente, l'operazione viene interrotta e l'errore segnalato

B) GWBASIC

Come già evidenziato nel paragrafo precedente, nell'aprire un file occorre indicare anche il modo di operare su esso Nei file sequenziali sono ammesse tre modalità:

"I" input – ingresso dati, cioè lettura dal disco;
"O" output – uscita dati, cioè scrittura sul disco;
"A" append – aggiunta dati, cioè scrittura sul disco in coda ai valori preesistenti.

I valori delle informazioni, numeriche od alfanumeriche, sono conservati su disco come insieme di caratteri. Essi possono essere memorizzati con l'istruzione:

WRITE # *numero-buffer , elenco-grandezze*

Il gruppo di valori memorizzati con un'unica istruzione è considerato come un record logico, e separato con un apposito simbolo dal record precedente. A loro volta, i singoli valori vengono considerati come campi del record e separati mutuamente mediante virgole. L'immagine che si ottiene sul disco è analoga a quella fornita da un'istruzione DATA Ad esempio, le istruzioni:

```
OPEN "O , #1, " PROVA DAT
WRITE #1, "valori di prova",4.254.3 5*5^2–18 44
WRITE #1  421 4958 "13 ottobre"
CLOSE #1
```

memorizzeranno nel file "PROVA.DAT", associato al buffer 1:

```
'valori di prova", 4.254  69 06
421.4958, '13 ottobre"
```

La distinzione logica in campi e record non ha però grande importanza applicativa. Infatti l'istruzione di lettura:

INPUT # *numero-buffer , elenco-variabili*

preleva i valori dal file in maniera consecutiva, perfettamente analoga a quella dei calcolatori HP. L'unica differenza che il GWBASIC presenta consiste nell'assenza di un riscontro tra tipo di valore e tipo di variabile.

La memorizzazione di valori è possibile anche con l'istruzione "**PRINT#**", che non separa i valori con virgolette e virgole. Essa può rivelarsi molto utile in applicazioni avanzate, ma richiede maggior attenzione per evitare errori in fase di lettura. Non verrà quindi utilizzata negli esempi di questo testo

BASIC HP

PRINT# numero-buffer ; elenco-grandezze

i valori indicati esplicitamente o calcolati vengono memorizzati nel buffer, codificati e con indicazione del tipo.

READ# numero-buffer ; elenco-variabili

a ciascuna variabile dell'elenco viene assegnato un valore, prelevato dal buffer; viene effettuato un riscontro tra tipo di variabile e di valore

GWBASIC

WRITE# numero-buffer , elenco-grandezze

i valori indicati esplicitamente o calcolati vengono memorizzati nel buffer, come insieme di caratteri, separati da virgole

INPUT# numero-buffer , elenco-variabili

a ciascuna variabile dell'elenco viene assegnato un valore, prelevato dal buffer; non viene effettuato un riscontro tra tipo di variabile e di valore.

5.7 - Operazioni su file ad accesso diretto.

La gestione di file ad accesso diretto da parte del BASIC HP e del GWBASIC si presenta sostanzialmente differente. I due linguaggi sono quindi esaminati separatamente.

A) BASIC HP.

Nell'aprire un file ad accesso diretto non è necessario specificare la lunghezza dei suoi record logici Essa infatti è fissa e già definita al calcolatore al momento della creazione del file (vedi paragrafo 5.5).

Le modalità operative sono sostanzialmente analoghe a quelle già descritte nel paragrafo precedente. Infatti, nell'ambito del singolo record logico le informazioni sono conservate in maniera sequenziale.

La scrittura di dati è ottenuta con l'istruzione:

PRINT# *num-buffer* , *num-record* ; *elenco-grandezze*

I valori così forniti vengono memorizzati in sequenza a partire dall'inizio del record specificato; in coda all'ultimo viene messa l'indicazione di fine del record.

La lettura è ottenuta con l'istruzione:

READ# *num-buffer* , *num-record* ; *elenco-variabili*

I valori vengono prelevati dal disco a partire dall'inizio del record indicato e conservati nelle variabili elencate, effettuando un riscontro del tipo.

B) GWBASIC

Il fatto che un file sia ad accesso diretto viene segnalato al calcolatore al momento della sua apertura, indicando la modalità "R", cioè random. In tale sede occorre anche definire la lunghezza dei suoi record, che sarà ovviamente sempre la stessa ogni volta che il file viene utilizzato. La lunghezza massima dei record è, per default, 128 bytes. Se è necessario usare una dimensione maggiore, ciò deve essere specificato nel caricare il GWBASIC.

Ciascun record è costituito da un insieme di caratteri e deve essere diviso in campi aventi lunghezza e nome specificati con l'istruzione:

FIELD # numero-buffer , lungh-1 **AS** *var-1* , *lungh-2* **AS** *var-2* ...

Ad esempio, le istruzioni:

```
OPEN "R" #1 "ELENCO",60
FIELD #1 20 AS COGN$, 20 AS NOME$, 8 AS MATR$
```

indicano che il file "ELENCO" è costituito da record di 60 caratteri. Di essi, i primi 20 costituiscono la variabile COGN$ (cognome), quelli da 21 a 40 la variabile NOME$ (nome) e quelli da 41 a 48 MATR$ (numero di matricola). Gli ultimi 12 caratteri non sono stati definiti; dimensionare con abbondanza i record comporta uno spreco di spazio, ma è un accorgimento che si usa quando si pensa di dover aggiungere in futuro ulteriori informazioni in ciascuno di essi.

Le variabili alfanumeriche di campo definite mediante l'istruzione FIELD differiscono dalle altre variabili, perchè non possono essere direttamente utilizzate in istruzioni di ingresso o di assegnazione. Per conservare in esse dei valori, si utilizzano le istruzioni:

LSET *var-campo* = *espressione-alfanumerica*

RSET *var-campo* = *espressione-alfanumerica*

Il risultato dell'espressione alfanumerica è posto nel campo indicato. Se la sua lunghezza è minore di quella del campo, viene allineato a sinistra (LSET, dove L indica *left*, cioè sinistra) oppure a destra (RSET, con R per *right*, destra); la restante parte del campo è riempita con spazi bianchi.

Valori numerici devono essere convertiti in alfanumerici prima di essere memorizzati in un campo. A tal fine si può ricorrere alla funzione STR$, descritta nel paragrafo 4.6. Più frequentemente, si utilizzano le funzioni MKI$, MKS$, MKD$ che trasformano rispettivamente numeri interi, in singola o doppia precisione in stringhe di 2, 4 o 8 caratteri. Per riconvertire la stringa in numero si usano le funzioni inverse: VAL, CVI, CVS, CVD.

Scrittura e lettura di un record sono ottenute con le istruzioni:

PUT # *numero-buffer , numero-record*

GET # *numero-buffer , numero-record*

Quest'ultima assegna automaticamente alle variabili di campo i valori letti nel record indicato

BASIC HP

PRINT # numero-buffer , numero-record ; elenco-grandezze

i valori indicati vengono memorizzati sequenzialmente nel record

READ # numero-buffer , numero-record ; elenco-variabili

a ciascuna variabile dell'elenco viene assegnato un valore, prelevato sequenzialmente a partire dall'inizio del record

GWBASIC

FIELD # numero-buffer , lungh-1 AS var-1

indica lunghezza e nome dei campi di cui è costituito ciascun record del file associato al buffer

LSET campo = espressione-alfanumerica
RSET campo = espressione-alfanumerica

pone il risultato dell'espressione alfanumerica nel campo allineandolo rispettivamente a sinistra o a destra

PUT # numero-buffer , numero-record

memorizza i campi nel record indicato

GET # numero-buffer , numero-record

preleva dal record indicato il valore dei campi

5.8 - Esempio: fusione di due elenchi ordinati

5.8.1 - Analisi del problema e diagramma di flusso.

Si pensi di avere due elenchi contenenti, in ordine alfabetico, i nominativi degli studenti che seguono due corsi universitari, A e B. Queste informazioni possono essere riportate in due file sequenziali, indicati rispettivamente col nome "CORSO-A" e "CORSO-B". Obiettivo del programma che si vuol realizzare è la costituzione di un terzo elenco, che contenga in maniera ordinata tutti i nominativi dei due corsi. Si indica il file che contiene questo elenco col nome "CORSI-AB".

Anzichè esaminare direttamente la soluzione, proviamo a seguire lo sviluppo logico che porta progressivamente al risultato finale Sulla scorta dei criteri più volte illustrati, si inizia l'analisi descrivendo a grandi linee quello che appare il modo migliore per raggiungere l'obiettivo. L'impostazione iniziale viene successivamente affinata e dettagliata, eliminando nel far ciò gli eventuali difetti o imprecisioni via via emersi

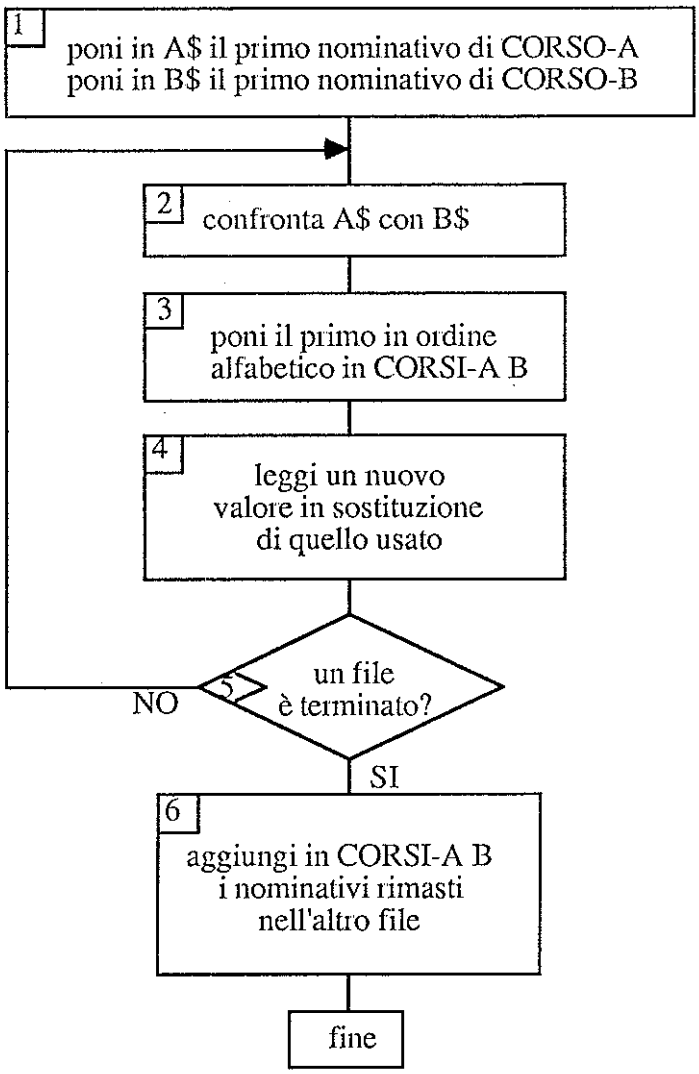


Fig. 38

Come ottenere dunque l'elenco cercato? Per cominciare, si può prelevare il primo nominativo del CORSO-A (che indicheremo con A$), ed il primo del CORSO-B (B$). Confrontandoli, si può determinare quale sia il primo in assoluto (per esempio A$), e memorizzarlo in CORSI-AB. Una volta fatto ciò, questo valore non interessa più. In sua sostituzione occorre prelevarne un altro dallo stesso file che lo conteneva (nell'esempio, CORSO-A). Si può quindi *ripetere* il confronto e la memorizzazione di valori, *finchè non* ci si accorge di essere giunti alla fine di uno dei file. A questo punto, i nominativi dell'altro file non ancora utilizzati andranno direttamente aggiunti in CORSI-AB. Questa impostazione, ancora molto generale, è rappresentato graficamente nel diagramma di flusso di figura 38.

Il confronto tra A$ e B$ può essere effettuato semplicemente mediante l'espressione logica A$<B$, se ipotizziamo che queste stringhe siano composte da due campi di dimensioni fisse (ad esempio 20 caratteri ciascuno) contenenti rispettivamente il cognome e il nome.

I due blocchi successivi, memorizzazione in CORSI-AB e lettura di un nuovo valore, sono funzioni del risultato di questo confronto. Occorrerebbe quindi definire meglio la procedura, introducendo una struttura IF... THEN... ELSE... (fig. 39). I blocchi 3a-3b e 4a-4b che in essa compaiono, sostanzialmente analoghi, possono essere unificati se si modifica leggermente l'impostazione iniziale. Individuando infatti gli elenchi di partenza con i numeri 1 e 2 (al posto delle lettere A e B) ed i nominativi con N$(1) ed N$(2) (al posto di A$ e B$), ed usando una variabile M per indicare quale dei due nominativi precede l'altro in ordine alfabetico, si ottiene il diagramma di figura 40.

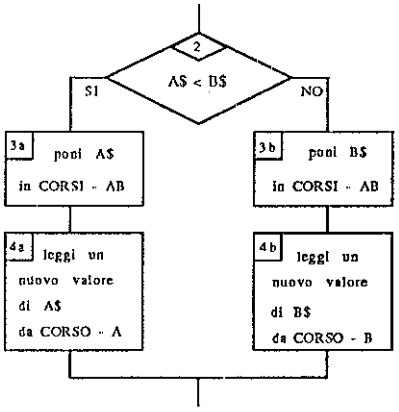


Fig 39

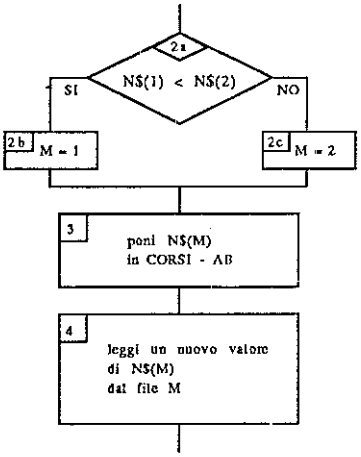


Fig 40

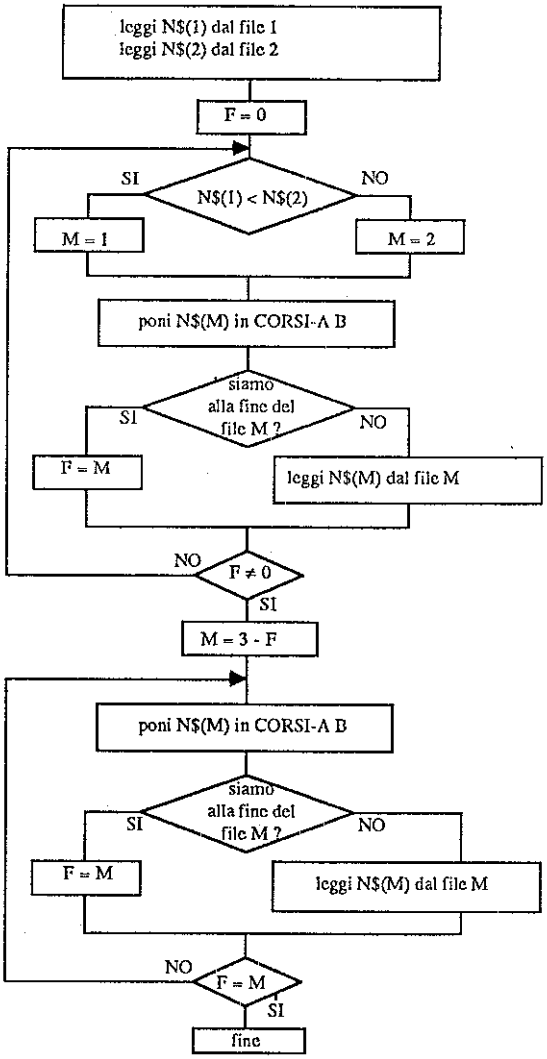

leggi N$(1) dal file 1
leggi N$(2) dal file 2
F = 0
SI
N$(1) < N$(2)
NO
M = 1
M = 2
poni N$(M) in CORSI-A B
siamo alla fine del file M ?
SI
NO
F = M
leggi N$(M) dal file M
NO
F ≠ 0
SI
M = 3 - F
poni N$(M) in CORSI-A B
siamo alla fine del file M ?
SI
NO
F = M
leggi N$(M) dal file M
NO
F = M
SI
fine

Fig 41

Un altro punto da precisare riguarda la lettura di un nuovo valore per N$(M) ed il controllo del file M per individuarne l'eventuale fine Per l'esattezza, occorre prima chiedersi se si è giunti alla fine del file (ed esistono in BASIC istruzioni che lo consentono), e poi, solo in caso contrario, leggere il nuovo valore Si può utilizzare una variabile di comodo, F, col seguente significato:

F = 0 in entrambi i file ci sono valori da utilizzare;
F = 1 nel file 1 non ci sono più valori da utilizzare;
F = 2 nel file 2 non ci sono più valori da utilizzare

Essa viene utilizzata per il controllo del ciclo (blocco 5) Inoltre, quando questo è terminato l'espressione numerica (3−F) indica in quale file ci sono ancora valori non esaminati La loro memorizzazione nel file CORSI-AB (blocco 6) è realizzata mediante un ulteriore ciclo REPEAT UNTIL

Si ottiene in questo modo il diagramma di flusso complessivo più dettagliato, rappresentato in figura 41

5.8.2 - Codifica in BASIC HP

Operando in BASIC HP, si può individuare la fine di un file dati sfruttando l'istruzione "ON ERROR " Normalmente, se nel leggere un valore dal disco il computer incontra la fine del file l'esecuzione viene interrotta e l'errore segnalato (*Error 71, EOF, end of file*). È però consentito indicare al calcolatore come comportarsi in presenza di errori L'istruzione "**ON ERROR GOSUB** *n*" prescrive che, da quando essa viene incontrata e finchè non viene annullata con l'istruzione "**OFF ERROR**", al verificarsi di un errore il computer salti al sottoprogramma che inizia alla linea n Nel caso in esame, il sottoprogramma controlla che l'errore riscontrato (ERRN) sia proprio il numero 71, ed in tal caso pone F=M L'esecuzione prosegue poi con la linea successiva a quella in cui si era verificato l'errore

La codifica completa del programma è riportata nelle righe seguenti:

```
 10 OPTION BASE 1
 20 DIM N$(2)[40]
 30 ASSIGN# 1 TO  CORSO-A
 40 ASSIGN# 2 TO "CORSO-B"
 50 ASSIGN# 3 TO  CORSI-AB'
 60 READ# 1 ; N$(1)
 70 READ# 2 ; N$(2)
 80 F=0
 90 ON ERROR GOSUB 250
100 IF N$(1)<N$(2) THEN M=1 ELSE M=2
110 PRINT# 3 ; N$(M)
120 READ# M ; N$(M)
130 IF F=0 THEN 100
140 M=3−F
```

```
150 PRINT# 3 ; N$(M)
160 READ# M ; N$(M)
170 IF F<>M THEN 150
180 ASSIGN# 1 TO *
190 ASSIGN# 2 TO *
200 ASSIGN# 3 TO *
210 STOP
220 REM
230 REM      Sottoprogramma che individua la fine del file
240 REM
250 IF ERRN =71 THEN 280
260 PRINT "ERRORE"
270 STOP
280 F=M
290 RETURN
```

5.8.3 - Codifica in GWBASIC

In GWBASIC la fine di un file può essere individuata mediante la funzione "**EOF(*n*)**" che fornisce il valore "vero" quando si è raggiunta la fine del file associato al buffer n

La codifica completa del programma è riportata nelle righe seguenti:

```
 10 OPTION BASE 1
 20 DIM N$(2)
 30 OPEN "I",#1  "CORSO-A"
 40 OPEN  I ,#2,  CORSO-B"
 50 OPEN  O',#3, "CORSI-AB
 60 INPUT#1,N$(1)
 70 INPUT#2 N$(2)
 80 F=0
100 IF N$(1)<N$(2) THEN M=1 ELSE M=2
110 WRITE#3,N$(M)
120 IF EOF(M) THEN F=M ELSE INPUT#M N$(M)
130 IF F=0 THEN 100
140 M=3-F
150 WRITE#3 N$(M)
160 IF EOF(M) THEN F=M ELSE INPUT#M N$(M)
170 IF F<>M THEN 150
180 CLOSE#1
190 CLOSE#2
200 CLOSE#3
210 STOP
```

CAPITOLO SESTO

INGRESSO DATI CON MASCHERA.

6.1 - Generalità.

Nella vita quotidiana è ormai molto diffuso l'uso di moduli prestampati per comunicare informazioni (ad esempio in indagini statistiche, nella richiesta di certificati, nella denuncia dei redditi). Essi possono essere costituiti da una o più pagine, in ciascuna delle quali vi è un insieme di scritte che chiarisce l'argomento trattato e indica quali informazioni occorre fornire; vi sono inoltre delle zone prefissate, che indicheremo col termine "campi", nelle quali scrivere le informazioni richieste

L'ingresso dati con maschera è la diretta trasposizione di questa modalità al calcolatore. La pagina è in questo caso lo schermo e la maschera è costituita dall'insieme di scritte e di campi (fig. 42) Se l'unità video ha capacità grafiche, i campi possono essere evidenziati riquadrandoli con linee In un video alfanumerico che consente l'uso di più colori, o di diverse tonalità di grigio, è più semplice individuarli cambiando il colore di fondo Se invece il video prevede solamente il bianco e il nero, sarà preferibile riempirli inizialmente con caratteri di default; ad esempio, nella figura è usato a tal fine il simbolo " . "

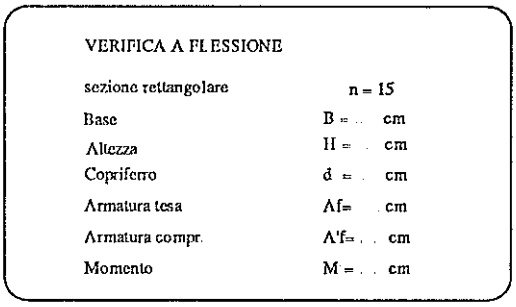


Fig 42

Questa modalità di ingresso dati risponde pienamente ai requisiti per un input ottimale analizzati nel paragrafo 4 10. L'insieme di scritte può essere tale da definire in maniera inequivocabile le informazioni richieste.

L'immissione del valore in un campo è semplice, perchè avviene mediante la tastiera con modalità analoghe a quelle richieste dall'istruzione INPUT. A differenza di quest'ultima, è possibile spostarsi da un campo all'altro, e da una pagina all'altra, premendo tasti convenzionalmente adibiti a questo scopo. Si può in tal modo effettuare la correzione immediata di valori inseriti erroneamente. Infine, la stessa maschera può essere utilizzata, oltre che per assegnare nuovi valori, anche per variare informazioni conservate su memoria di massa, semplicemente inserendo nei campi presentati all'utente i valori prelevati dal disco al posto dei caratteri di default

Per poter utilizzare maschere nell'ingresso dati, bisogna preparare un insieme di sottoprogrammi che gestiscono in maniera standard, valida per tutti i programmi da implementare, i diversi aspetti dell'input: definizione di scritte e campi della maschera, immissione di valori nel singolo campo, passaggio da un campo all'altro nella pagina, o da una pagina all'altra.

Per realizzare questi sottoprogrammi occorre usare istruzioni per la gestione del video e della tastiera, che non fanno parte del BASIC standard, ma sono presenti in tutti i calcolatori, sia pure con nomi e caratteristiche che possono variare da un modello all'altro.

Nei computer HP, serie 80, esse non fanno parte del linguaggio residente in memoria; nei modelli HP-86/87 vengono implementate dal programma binario "UTIL/1", fornito dalla casa costruttrice insieme al computer.

Il GWBASIC contiene invece direttamente le istruzioni necessarie. I loro effetti possono però variare leggermente in funzione dell'hardware del modello, ed è quindi opportuno effettuare dei riscontri concreti prima di utilizzarle. Nello scrivere gli esempi riportati nel testo, esse sono state verificate sui computer Olivetti M24 e IDE Best, entrambi con video monocromatico.

6.2 - Codici dei caratteri alfanumerici.

Come già più volte precisato, il valore di ogni informazione viene conservato nei registri di memoria in forma codificata, con codici dipendenti dal tipo. Per una completa gestione di informazioni alfanumeriche, occorrono precise informazioni su come avviene questa codifica.

Nei personal computer ogni carattere viene memorizzato in un byte, cioè in 8 bit. Esso è quindi rappresentato con un numero binario ad 8 cifre (da "00000000" a "11111111"), corrispondente ad un numero intero compreso tra 0 e 255 (fig. 43).

I codici da 0 a 31 sono riservati alla trasmissione di informazioni particolari o indicazioni operative tra unità diverse; ad esempio, il codice 12 rappresenta per la stampante il comando di salto pagina. Se inviati al video, vengono visualizzati con simboli non standard, che variano da un computer all'altro.

I codici da 32 a 127 rappresentano gli effettivi caratteri alfanumerici, visualizzabili e stampabili. Esiste per essi uno standard ASCII, che è rispettato, a meno di qualche simbolo poco usuale, da tutti i computer.

		EQUIVALENT FORMS			EQUIVALENT FORMS			EQUIVALENT FORMS			EQUIVALENT FORMS	
Mnemonic	Char.	Binary	Dec	Char.	Binary	Dec	Char.	Binary	Dec	Char.	Binary	Dec
NUL	@^{c}	00000000	0	SPACE	00100000	32	@	01000000	64	`	01100000	96
SOH	A^{c}	00000001	1	!	00100001	33	A	01000001	65	a	01100001	97
STX	B^{c}	00000010	2	"	00100010	34	B	01000010	66	b	01100010	98
ETX	C^{c}	00000011	3	#	00100011	35	C	01000011	67	c	01100011	99
EOT	D^{c}	00000100	4	$	00100100	36	D	01000100	68	d	01100100	100
ENQ	E^{c}	00000101	5	%	00100101	37	E	01000101	69	e	01100101	101
ACK	F^{c}	00000110	6	&	00100110	38	F	01000110	70	f	01100110	102
BEL	G^{c}	00000111	7	'	00100111	39	G	01000111	71	g	01100111	103
BS	H^{c}	00001000	8	(	00101000	40	H	01001000	72	h	01101000	104
HT	I^{c}	00001001	9	)	00101001	41	I	01001001	73	i	01101001	105
LF	J^{c}	00001010	10	*	00101010	42	J	01001010	74	j	01101010	106
VT	K^{c}	00001011	11	+	00101011	43	K	01001011	75	k	01101011	107
FF	L^{c}	00001100	12	,	00101100	44	L	01001100	76	l	01101100	108
CR	M^{c}	00001101	13	-	00101101	45	M	01001101	77	m	01101101	109
SO	N^{c}	00001110	14	.	00101110	46	N	01001110	78	n	01101110	110
SI	O^{c}	00001111	15	/	00101111	47	O	01001111	79	o	01101111	111
DLE	P^{c}	00010000	16	0	00110000	48	P	01010000	80	p	01110000	112
DC1	Q^{c}	00010001	17	1	00110001	49	Q	01010001	81	q	01110001	113
DC2	R^{c}	00010010	18	2	00110010	50	R	01010010	82	r	01110010	114
DC3	S^{c}	00010011	19	3	00110011	51	S	01010011	83	s	01110011	115
DC4	T^{c}	00010100	20	4	00110100	52	T	01010100	84	t	01110100	116
NAK	U^{c}	00010101	21	5	00110101	53	U	01010101	85	u	01110101	117
SYN	V^{c}	00010110	22	6	00110110	54	V	01010110	86	v	01110110	118
ETB	W^{c}	00010111	23	7	00110111	55	W	01010111	87	w	01110111	119
CAN	X^{c}	00011000	24	8	00111000	56	X	01011000	88	x	01111000	120
EM	Y^{c}	00011001	25	9	00111001	57	Y	01011001	89	y	01111001	121
SUB	Z^{c}	00011010	26	:	00111010	58	Z	01011010	90	z	01111010	122
ESC	[^{c}	00011011	27	;	00111011	59	[	01011011	91	{	01111011	123
FS	\$^{c}$	00011100	28	<	00111100	60	\	01011100	92	|	01111100	124
GS	]^{c}	00011101	29	=	00111101	61	]	01011101	93	}	01111101	125
RS	^^{c}	00011110	30	>	00111110	62	^	01011110	94	~	01111110	126
US	_^{c}	00011111	31	?	00111111	63	_	01011111	95		01111111	127

Fig. 43

I codici da 128 a 255 (cioè quelli col primo bit pari ad 1) non presentano alcuna standardizzazione. Nel BASIC HP vengono usati per rappresenta-

re sul video gli stessi caratteri corrispondenti al codice col primo bit nullo, con l'aggiunta del simbolo del cursore. Nell'HP-85 questo è costituito da un trattino posto sotto il carattere; negli HP-86/87 viene invece visualizzato il carattere su sfondo invertito (scuro su fondo chiaro).

Nel GWBASIC questi codici rappresentano invece lettere accentate ed altri simboli particolari. La presenza o assenza del cursore in una posizione del video costituisce una informazione distinta dal codice del carattere

L'istruzione HP "NUM(X$)" e l'analoga GWBASIC "ASC(X$)" forniscono il codice del primo carattere della variabile X$ L'istruzione inversa, "CHR$(X)" fornisce il carattere avente come codice X

6.3 - Gestione del video.

Si è già accennato, proprio all'inizio del testo, che ciò che compare nel display di una calcolatrice non è altro che il contenuto corrente di uno specifico registro ausiliario. In maniera del tutto analoga, l'insieme dei caratteri visualizzati in un certo istante nello schermo di un personal computer mostra il contenuto di un gruppo di registri dedicati a questo compito.

Lo schermo dell'HP-86/87 contiene 16 o 24 righe ed 80 colonne. Esse sono individuate rispettivamente con i numeri da 0 a 15 o 23 e da 0 a 79. Nell'HP-85 vi sono solo 16 righe e 32 colonne.

Il BASIC HP utilizza un byte per ciascun carattere visualizzato. L'area di memoria riservata a tale scopo è capace di contenere fino a 204 righe. Sullo schermo appare quindi l'immagine di una sua parte. L'intero contenuto può essere fatto scorrere sul video, come fosse un nastro continuo, utilizzando un tasto a ciò adibito (ROLL). Questa caratteristica consente, ad esempio, di rivedere il contenuto di righe ormai scomparse dal video, che in altri calcolatori vanno invece perdute. Essa potrebbe essere utilizzata nell'input con maschera per avere a disposizione pagine di dati molto più ampie dello schermo. Si è però preferito rinunciarvi, perchè incompatibile con altri calcolatori.

In GWBASIC lo schermo contiene 25 righe e 40 od 80 colonne, numerate rispettivamente da 1 a 25 e da 1 a 40 o 80. La venticinquesima riga contiene normalmente indicazioni sul significato dei tasti funzione. Per poterla utilizzare da programma, è necessario eliminare tali indicazioni con l'istruzione "**KEY OFF**".

Il GWBASIC, che consente l'adozione di schermi policromatici, utilizza due byte per ciscun carattere visualizzato. Il primo contiene il codice del carattere. Del secondo, 5 bit sono usati per definire il colore (o la tonalità di grigio) del carattere, altri 3 per il colore dello sfondo. La definizione di questi colori è ottenuta mediante l'istruzione:

COLOR *colore-carattere , colore-sfondo*

Il colore del carattere deve essere un numero compreso tra 0 e 31. I valori da 16 a 31 rappresentano gli stessi colori definiti con i numeri da 0 a 15, ma con il

carattere lampeggiante. Per lo sfondo si possono utilizzare valori compresi tra 0 e 7. Una immagine di tutte le possibili combinazioni è ottenuta mediante il seguente programma:

```
10 FOR I=0 TO 31
20      FOR J=0 TO 7
30          COLOR I,J
40          PRINT "COLOR";I;" ";J
50      NEXT J
60      PRINT
70 NEXT I
80 COLOR 7,0
90 STOP
```

I colori automaticamente definiti dal calcolatore all'accensione sono indicati dai codici 7 e 0 (carattere bianco su sfondo nero, se il video non è a colori). Nella maschera di input si useranno in seguito codici scelti in modo da evidenziare i campi ma mantenere ben visibili i caratteri in essi disposti. La scelta dipende necessariamente dal modello di computer; per l'M24 sono sembrati preferibili i codici 0 e 3 (carattere nero su sfondo chiaro) e per l'IDE Best i codici 15 e 1 (carattere bianco su sfondo grigio), ma ovviamente tale scelta potrà essere modificata dal lettore.

Anche in GWBASIC l'area di memoria riservata al video è più ampia di quanto strettamente necessario. A differenza che negli HP, essa è strutturata non come nastro continuo, ma come pagine distinte, intercambiabili (4 pagine di 80 colonne, oppure 8 pagine di 40 colonne ciscuna). Si è però preferito non sfruttare questa caratteristica nell'input a maschera per garantire la compatibilità con altre versioni del BASIC.

Per una completa gestione del video, occorrono istruzioni che consentano lo svolgimento di cinque operazioni: posizionamento del cursore, visualizzazione o annullamento del suo simbolo, scrittura di caratteri nel video in una posizione definita, lettura di caratteri dal video, cancellazione dello schermo. Esse verranno esaminate separatamente per i due linguaggi, BASIC HP e GWBASIC.

A) BASIC HP.

Come già accennato nel paragrafo precedente, il primo bit del codice di un carattere viene utilizzato per indicare la presenza o assenza del simbolo del cursore. Esso può quindi essere fatto comparire o scomparire in una qualsiasi posizione dello schermo semplicemente variando di 128 il codice memorizzato nel registro corrispondente. Quando si è in modalità *insert* (inserimento di un carattere in mezzo ad altri) viene visualizzato un doppio cursore, cioè il suo simbolo è ripetuto anche nel carattere immediatamente a sinistra.

L'effettiva posizione del cursore è definita mediante un registro puntatore. Finchè la gestione del video è interamente a carico del sistema operativo, la corrispondenza tra simbolo e posizione reale è garantita da esso. Nella gestione di maschere per l'input, tale compito è invece affidato al programma stesso.

L'istruzione del programma binario "UTIL/1" che consente il posizionamento del cursore è costituita dalla parola "**AWRITE**" seguita dal numero della riga e della colonna, separati da una virgola. Con questa istruzione il cursore viene posto nella posizione richiesta ma non viene visualizzato

Il nome della istruzione, "AWRITE", deriva dalle parole inglesi *alpha* e *write*; la prima è un riferimento al video alfanumerico, la seconda significa "scrivi" Infatti, aggiungendo in coda ad essa una espressione alfanumerica, separata dal numero della colonna mediante una virgola, se ne impone la visualizzazione del risultato a partire dalla posizione indicata. In tal caso il cursore rimane in corrispondenza del primo carattere.

L'istruzione che permette la lettura dal video è invece costituita dalla parola "**AREAD**" (da *alpha* e *read* cioè "leggi"), seguita dal nome di una variabile alfanumerica. Vengono letti tanti caratteri quanti ne sono consentiti dalla lunghezza massima della variabile, a partire dalla posizione corrente del cursore.

Infine, la cancellazione dello schermo può essere ottenuta mediante l'istruzione del BASIC HP "**CLEAR**" (parola che significa "libera" o "vuota" lo schermo).

B) GWBASIC.

In GWBASIC la presenza del simbolo del cursore è sempre collegata alla sua reale posizione nello schermo. Questa viene definita mediante l'istruzione:

LOCATE *n-riga, n-colonna, cursore, inizio, fine*

I primi due parametri indicano la riga e la colonna nella quale si vuole porre il cursore Il terzo può assumere il valore 0 oppure 1, per indicare rispettivamente che il cursore deve essere invisibile oppure visibile Gli ultimi due parametri indicano la dimensione del cursore (linea iniziale e finale che lo compongono), che può variare a seconda del modello di computer e della presenza di schede per la grafica; nell'Olivetti M24 la dimensione normale del cursore è definita dai parametri 6,7; in modalità *insert* (inserimento di caratteri) la sua dimensione è invece definita da 3,7.

La scrittura di caratteri viene effettuata con l'istruzione PRINT, già nota, che pone caratteri sullo schermo a partire dalla posizione del cursore; questo viene spostato man mano che le scritte vengono visualizzate.

L'individuazione del codice di un carattere dello schermo è possibile median-

te la funzione "**SCREEN**(*n-riga,n-colonna*)". Essa può fornire anche il colore del carattere e del suo sfondo, se vi si aggiunge un terzo parametro di valore diverso da 0. Questa istruzione non varia la posizione del cursore.

Infine, la cancellazione dello schermo è ottenuta mediante l'istruzione "**CLS**" (da *clear* e *screen*).

BASIC HP - programma binario "UTIL/1"

AWRITE n-riga, n-colonna

il cursore viene posizionato nella riga e colonna indicata, senza essere visualizzato.

Codice di un carattere variato di 128

viene fatto comparire o scomparire il simbolo del cursore.

AWRITE n-riga, n-colonna espr-alfanum

il risultato dell'espressione alfanumerica viene visualizzato a partire dalla posizione indicata

AREAD variabile-alfanumerica

la variabile alfanumerica viene riempita con caratteri letti dallo schermo a partire dalla posizione corrente del cursore

CLEAR

il contenuto dello schermo viene cancellato.

GWBASIC

LOCATE n-riga, n-colonna, cursore, inizio, fine

Il cursore viene posizionato nella riga e colonna indicata; se il terzo parametro è diverso da 0, il cursore è visualizzato, con dimensioni definite dagli ultimi due parametri

PRINT

stampa di valori sullo schermo, a partire dalla posizione del cursore (vedi paragrafo 4.11)

SCREEN (n-riga, n-colonna)

fornisce il codice del carattere posto nella posizione indicata.

CLS

il contenuto dello schermo viene cancellato

6.4 - Codici dei tasti.

Ogni tasto della tastiera (o ogni combinazione ottenuta premendo i tasti "SHIFT" o "CTRL" contemporaneamente ad un altro tasto) è individuato mediante un codice costituito da uno o più byte. I tasti che corrispondono a caratteri alfanumerici, o a comandi per la stampante o altre unità, hanno come codice quello del carattere stesso (quindi un numero compreso tra 1 e 127).

I tasti operativi hanno invece codici convenzionali. Nel BASIC HP, che usa sempre un solo byte per ogni tasto, essi sono numeri compresi tra 128 e 255. Nel GWBASIC sono invece costituiti da due byte, dei quali il primo ha sempre il valore 0, oppure un solo byte di valore inferiore a 32.

In figura 44 sono riportati i codici corrispondenti ai tasti che verranno utilizzati nell'input a maschera

TASTO PREMUTO	CODICE HP-86/87	CODICE GWBASIC	SIGNIFICATO NELL'INPUT A MASCHERA
carattere alfanumerico	da 32 a 127		Il carattere viene inserito nel campo nella posizione corrente del cursore. Il cursore viene spostato verso destra nel campo
→	170	0 77	Il cursore viene spostato verso destra nel campo
←	159	0 75	Il cursore viene spostato verso sinistra nel campo
I/R INS	158	 0 82	Attiva / disattiva la modalità *insert*
−CHR DEL	136	 0 83	Cancella il carattere posto in corrispondenza al cursore
BACKSPACE	153	8	Cancella il carattere che precede il cursore.
	34	34	Copia nel campo il contenuto del campo precedente (opzionalmente: poi passa al campo successivo)
ENDLINE ENTER	154	 13	Comando di uscita dal campo: 1 - passa al campo successivo
↑	163	0 72	Comando di uscita dal campo: 2 - passa al campo precedente
↖ HOME	152	 0 71	Comando di uscita dal campo: 3 - passa al primo campo
KEY LABEL PG DN	150	 0 81	Comando di uscita dal campo: 4 - passa alla pagina successiva
Shift KEY LABEL PG UP	96	 0 73	Comando di uscita dal campo: 5 - passa alla pagina precedente

Fig. 44

6.5 - Gestione della tastiera.

Normalmente, quando si preme un qualsiasi tasto il sistema operativo del calcolatore riceve questa informazione e decide cosa fare in conseguenza: se il tasto corrisponde ad un comando (per esempio, interrompere una elaborazione in corso), questa operazione viene eseguita; se invece corrisponde ad un carattere alfanumerico, esso viene visualizzato nella posizione corrente sul video.

Con apposite istruzioni è invece possibile cedere l'autorità decisionale al programma in esecuzione. In tal modo, appena un tasto viene premuto il codice ad esso corrispondente viene memorizzato in un registro di memoria, a disposizione del programma. Lo si potrà così esaminare e decidere in base ad esso quali attività compiere.

A) BASIC HP.

Il programma binario "UTIL/1" consente, con l'istruzione "**TAKE KEYBOARD**" (parole che in inglese significano "prendi la tastiera"), di creare un buffer, cioè un insieme di registri di transito, ed anteporsi al sistema operativo nella gestione dei tasti premuti. Dal momento che questa istruzione è stata eseguita, ogni volta che si preme un tasto il carattere corrispondente al suo codice viene posto nel buffer, in coda agli altri caratteri già presenti.

L'istruzione opposta è "**RELEASE KEYBOARD**" (cioè "rilascia la tastiera") Essa annulla il buffer e cede nuovamente al sistema operativo il compito di interpretare i tasti.

Per conoscere all'interno del programma quale tasto è stato premuto si usa l'istruzione:

variabile-alfanumerica = **KEY$**

Essa estrae dal buffer il primo carattere disponibile (diminuendo quindi di una unità il numero di caratteri in esso conservati) e lo memorizza nella variabile alfanumerica Se il buffer è vuoto, pone nella variabile una stringa nulla ""

Un esempio di utilizzazione delle istruzioni descritte è fornito da un programma che consente di stampare il codice corrispondente a ciascun tasto premuto. Per ottenere ciò, bisogna innanzitutto creare il buffer e anteporsi al sistema operativo. Occorre poi *ripetere* una serie di operazioni (esaminare quale tasto è stato premuto e stamparne il codice) *finchè non* viene premuto un tasto che indica la fine (ad esempio la lettera "F"). Quando ciò è avvenuto, si cede al sistema operativo il compito di interpretare i tasti e si termina l'esecuzione. Per esaminare quale tasto è stato premuto, non è sufficiente prelevare un valore dal buffer. Occorre *ripetere* tale operazione *finchè non* si è ottenuto un valore diverso dalla stringa nulla.

Si noti come, nell'analizzare il problema e descrivere come risolverlo, si

sono utilizzate espressamente due strutture logiche "REPEAT ... UNTIL ...". La codifica relativa è riportata di seguito

```
10 TAKE KEYBOARD
20 V$=KEY$
30 IF V$="" THEN 20
40 PRINT NUM(V$)
50 IF V$<>"F" THEN 20
60 RELEASE KEYBOARD
70 STOP
```

L'uso di un buffer consente di premere più tasti in sequenza, anche molto rapidamente, senza interferire con le altre operazioni richieste (nell'esempio, la stampa del codice)

B) GWBASIC

Il GWBASIC crea automaticamente un buffer nel quale porre i codici corrispondenti ai tasti premuti. L'istruzione che consente di prelevare un valore dal buffer si presenta simile a quella implementata per gli HP-86/87:

variabile-alfanumerica = **INKEY$**

In questo caso, però, ad un tasto possono corrispondere anche due bytes di codice, cioè una variabile di due caratteri. Il programma che visualizza il codice dei tasti deve essere quindi così modificato:

```
10 V$=INKEY$
20 IF V$="" THEN 10
30 FOR V=1 TO LEN(V$)
40 PRINT ASC(MID$(V$ V 1));
50 NEXT V
60 PRINT
70 IF V$<>"F" THEN 10
80 STOP
```

BASIC HP – programma binario 'UTIL/1'

TAKE KEYBOARD
RELEASE KEYBOARD

crea / annulla un buffer che contiene i codici corrispondenti ai tasti premuti

variabile-numerica = KEY$

preleva dal buffer il primo valore disponibile e pone nella variabile alfanumerica il carattere ad esso corrispondente

GWBASIC

variabile-alfanumerica = INKEY$

preleva dal buffer il primo valore disponibile e pone nella variabile alfanumerica il carattere, o i caratteri, ad esso corrispondente

6.6 - Gestione di un campo dati

6.6.1 - Impostazione generale

Per consentire all'utente la massima semplicità d'uso, è importante che la immissione di dati in un singolo campo avvenga con le stesse modalità cui egli è abituato dall'istruzione INPUT.

Il cursore evidenzia la posizione corrente nel campo (fig. 45). Ogni volta che viene premuto un tasto corrispondente ad un carattere alfanumerico, questo è visualizzato in tale posizione, al posto del carattere preesistente, ed il cursore viene avanzato verso destra. Il campo dati ha però dimensioni ben definite. Quando esso è stato riempito, il cursore viene a trovarsi al di fuori, nella prima posizione disponibile alla sua destra. A questo punto, non ha più senso aggiungervi ulteriori caratteri; se ciò viene tentato, occorre segnalare l'errore e lasciarne inalterato il contenuto.

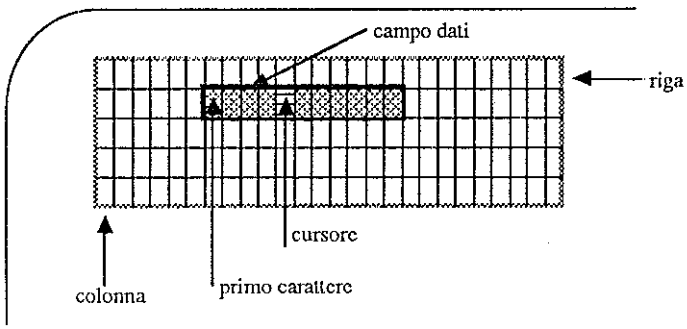


Fig. 45

Nel normale funzionamento della tastiera, alcuni tasti sono di aiuto per la immediata correzione di errori di battitura I tasti contrassegnati con una freccia consentono il movimento del cursore senza modificare il contenuto dello schermo Nell'ambito di un campo dati sono ammissibili spostamenti verso destra e sinistra, finchè il cursore non ha raggiunto un estremo del campo. Due tasti sono in genere adibiti alla cancellazione di caratteri: il tasto *backspace*, che sposta il cursore verso sinistra e cancella il carattere contenuto nella nuova posizione, e il tasto *delete*, che compatta un testo, eliminando il carattere situato nella posizione corrente del cursore e richiamando verso sinistra i caratteri ad esso successivi. Un tasto, infine, permette l'attivazione della modalità *insert* Quando questa è in vigore, premendo il tasto corrispondente a un carattere alfanumerico esso viene inserito nella posizione corrente del cursore dopo aver spostato verso destra i caratteri presenti a partire da quella posizione. Il simbolo del cursore viene variato per indicare che si è in tale modalità; il BASIC HP utilizza a tal fine un doppio cursore, mentre il GWBASIC varia le dimensioni del simbolo usuale La modalità *insert* viene disattivata quando si preme nuovamente il tasto usato per attivarla oppure i tasti *backspace* e *delete* (nel GWBASIC anche premendo i tasti che spostano il cursore)

Nei problemi di ingegneria si ha spesso a che fare con valori che si ripetono Si è quindi ritenuto comodo dedicare un tasto al compito di ricopiare nel campo in esame il contenuto del campo precedente Questa funzione non trova riscontro nell'istruzione INPUT. Si è quindi scelto per indicarla il tasto individuato dal simbolo *virgolette*, che viene comunemente utilizzato nella scrittura per omettere una parola indicando che essa è uguale a quella riportata nella riga precedente

Una volta completata l'immissione del valore nel campo, il passaggio al dato successivo deve essere ottenuto, come nell'istruzione INPUT, mediante il tasto ENDLINE (o ENTER, nel GWBASIC). Caratteristica dell'ingresso dati con maschera è però il potersi spostare con facilità da un campo all'altro, o da una pagina all'altra Occorrerà quindi definire altri tasti che rappresentino convenzionalmente il passaggio ad un campo diverso da quello successivo Le diverse possibilità verranno prese in esame in seguito. Ai fini dell'uscita dal singolo campo è sufficiente avere a disposizione un elenco ordinato dei codici dei tasti che impongono di terminare le operazioni nel campo ed individuare se il tasto premuto corrisponde ad uno di essi In caso affermativo, il numero d'ordine del codice individuato costituisce un valore di uscita che sarà usato dal sottoprogramma che gestisce l'insieme dei campi di una pagina

Quando si usa il tasto *virgolette* innanzi descritto, dopo aver immesso il valore si dovrà in genere passare ad un altro campo; può quindi essere comodo associare ad esso anche la funzione di spostarsi automaticamente al campo successivo

Si è infine ritenuto più comodo per l'utente imporre che, prima di abbandonare il campo, la sua parte destra, a partire dal cursore, venga riempita di spazi bianchi. Una tale scelta è ovviamente soggettiva, e può essere eliminata da chi

non la ritenga opportuna. Questa operazione non deve però essere effettuata quando il cursore è nella posizione iniziale del campo, perchè ciò impedirebbe un rapido spostamento, senza modifiche, da un campo all'altro.

6.6.2 - Variabili utilizzate.

Nel definire i nomi delle variabili interne ad un sottoprogramma, è sempre bene evitare la possibilità di coincidenza con nomi di variabili del programma principale che lo richiama. Volendo realizzare, come nel caso in esame, un sottoprogramma di validità generale, cioè che possa essere usato da più programmi, occorre una regola semplice ma inequivocabile per evitare tali coincidenze. Si è pertanto deciso di assegnare a tutte le variabili utilizzate in questo e negli altri sottoprogrammi che in seguito si espongono nomi che inizino con la lettera U, ed evitare tale lettera nei programmi principali. La scelta è caduta sulla U perchè quasi nessun ente fisico trattato nei problemi di ingegneria ha nome che inizia con essa.

Nel sottoprogramma per la gestione del singolo campo si sono utilizzate le seguenti variabili:

Variabili di ingresso:

UR	riga in cui è situato il campo;
UC	colonna del primo carattere del campo;
UL	lunghezza (cioè numero di caratteri) del campo;
UP$	contenuto del campo precedente;
UF()	codici dei tasti di comando d'uscita dal campo

Variabile d'uscita:

UF	numero d'ordine del tasto d'uscita premuto; UF=0 indica che non è stato ancora premuto un tasto di comando d'uscita

Variabili interne:

UP	posizione del cursore nel campo; è un numero intero compreso tra 0 ed UL;
UI	indica se si è in modalità "insert" (UI=1) oppure no (UI=0);
UK$	carattere (o caratteri) corrispondenti al codice del tasto premuto;
UK	codice del tasto premuto; nel caso del GWBASIC, si è usato un codice convenzionale per quei tasti cui corrispondono due bytes;
UC$	variabile ausiliaria utilizzata per la lettura di un carattere dallo schermo;
UV	variabile ausiliaria utilizzata come contatore nel ciclo di confronto tra codice del tasto premuto e codici dei comandi d'uscita, nel riempire di spazi bianchi la parte destra del campo, nello spostare insiemi di caratteri nel campo.

6.6.3 - Descrizione del sottoprogramma

La figura 46 mostra lo schema base del sottoprogramma. All'inizio (blocco 1), occorre indicare che il cursore occupa la posizione iniziale nel campo, che la modalità *insert* non è attivata e che non si è ancora giunti al termine dell'inserimento di dati nel campo. Bisogna poi *ripetere* una sequenza di operazioni (blocchi 2-5), *finchè non* si riscontra che è stato premuto un tasto di uscita. Nell'ordine, si visualizza il cursore nella posizione corrente, si aspetta che sia premuto un tasto, si fa scomparire il simbolo del cursore, ed infine si esamina il codice del tasto premuto e si eseguono le operazioni ad esso connesse.

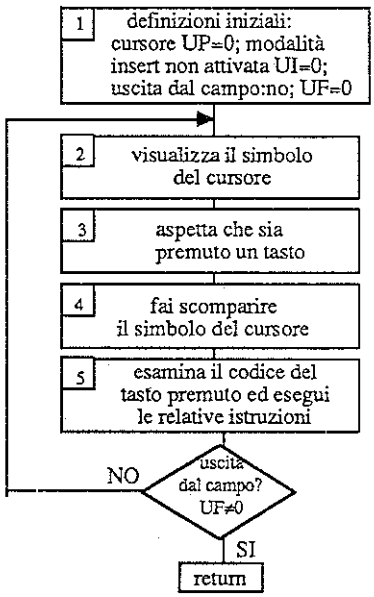


Fig 46

I blocchi 2 e 4 sono stati realizzati mediante sottoprogrammi. Nel BASIC HP, per visualizzare o annullare il cursore occorre leggere il carattere posto nella posizione corrente, variarne il codice di 128 e riscriverlo sullo schermo. Se si è in modalità *insert*, la stessa operazione deve essere effettuata anche sul carattere posto nella posizione immediatamente precedente. L'incremento di 128 fa comparire il simbolo di curso-

re dove esso è assente e scomparire dove è presente. È sufficiente quindi un unico sottoprogramma per entrambi le funzioni (linee a partire da 9800). Nel GWBASIC occorrono invece due distinti gruppi di istruzioni (da 9800 e da 9850) che utilizzano il comando LOCATE. Nel visualizzare il cursore, le sue dimensioni saranno maggiori o minori, secondo che sia o no attiva la modalità *insert*

Il blocco 3 prevede un ciclo "REPEAT ... UNTIL ..." che termina quando viene premuto un tasto. Il codice ad esso corrispondente viene memorizzato nella variabile UK. In GWBASIC ad alcuni tasti corrispondono due bytes, il primo dei quali ha codice 0 Per semplicità operativa, si è preferito ricondurre anche questi ad un unico codice, stabilendo convenzionalmente di assegnare ad ogni sequenza di tale tipo un valore numerico pari al codice del secondo byte cambiato di segno. Così ad esempio al tasto END (codici 0 79) si è associato il valore −79.

Il blocco 5 presenta una struttura "CASE ... OF ...". Un selettore di caso indirizza, in base al codice, ad uno tra i possibili blocchi operativi Se il tasto premuto non corrisponde ad una operazione prevista, viene emesso un suono bitonale (sottoprogramma 9900) che indica l'errore Tutti i blocchi terminano con il salto alla linea 9700, che chiude il ciclo base con l'esame della variabile UF (blocco 6).

Per realizzare le funzioni individuate come necessarie nel paragrafo 6.6 1, si sono previsti 8 blocchi di istruzioni, che si esaminano dettagliatamente quì di seguito con lo stesso ordine nel quale sono stati allora descritti.

A) Carattere alfanumerico (linee da 9300).

Se il cursore è alla fine del campo (UP=UL) non è possibile inserire un altro carattere; si richiama quindi il sottoprogramma che emette il segnale di errore. In caso contrario, il carattere viene visualizzato sullo schermo. Prima di farlo, se è attiva la modalità *insert* occorre spostare verso destra i caratteri, leggendoli e riscrivendoli ciclicamente ad uno ad uno, a partire dal penultimo (l'ultimo deve necessariamente essere perso) fino a quello corrispondente al cursore. La posizione di questo viene infine variata (UP=UP+1).

B) Avanzamento del cursore (linee da 9400).

Se il cursore è alla fine del campo, non è possibile spostarlo ulteriormente verso destra. In caso contrario, si varia la posizione incrementando UP.

Nel GWBASIC lo spostamento annulla la modalità *insert*. La variabile UI viene pertanto posta pari a 0.

C) Arretramento del cursore (linee da 9450).

Se il cursore è all'inizio del campo (UP=0), non è possibile spostarlo verso sinistra. In caso contrario, la posizione è variata decrementando UP.

D) Backspace (linee da 9500)

L'operazione è possibile solo se il cursore non è all'inizio del campo Nel BASIC HP il tasto backspace comporta lo spostamento del cursore verso sinistra e la sostituzione del carattere esistente in questa posizione con uno spazio bianco. Nel GWBASIC l'arretramento del cursore è accoppiato con un compattamento del testo. I caratteri vengono ad uno ad uno letti e riscritti, arretrati di una posizione. Alla fine del campo viene posto uno spazio bianco. Per entrambi i linguaggi viene annullata la modalità *insert*.

E) Delete (linee da 9550).

L'operazione è possibile solo se il cursore non è a fine campo. Il compattamento effettuato è simile a quella descritto per il backspace del GWBASIC. Il carattere eliminato è però quello corrispondente al cursore, e la posizione di questo non viene variata.

F) Tasto insert (linee da 9650).

Per attivare o disattivare la modalità è sufficiente invertire il valore della variabile UI (UI=1−UI).

G) Virgolette (linee da 9250)

Si è imposto che questa operazione venga effettuata solo se la lunghezza del campo precedente (e quindi della variabile UP$) è uguale a quella del campo in esame. In tal caso il contenuto di UP$ viene visualizzato e la posizione del cursore riportata all'estremo sinistro (UP=0). Se si vuole il passaggio automatico al campo successivo, si porrà poi UF=1.

H) Uscita dal campo (linee da 9200).

Il numero d'ordine del tasto di uscita individuato nella fase di selezione viene memorizzato in UF. Se il cursore non è all'inizio o alla fine del campo, si procede al riempimento della parte destra di esso con spazi bianchi.

6.6.4 - Codifica in BASIC HP.

```
8980 !
8985 ! GESTIONE DEL SINGOLO CAMPO DATI
8990 !
8995 !          definizione valori iniziali :
9000 UP=0
9005 UI=0
9010 UF=0
9045 !          visualizza il cursore :
9050 GOSUB 9800
9055 !          aspetta che sia premuto un tasto :
9060 WAIT 200
9065 UK$=KEY$
```

```
9070 IF UK$= " THEN 9065
9075 UK=NUM (UK$)
9080 !              fa scomparire il cursore :
9085 GOSUB 9800
9095 !              individua il tasto premuto :
9100 FOR UV=1 TO 5
9105        IF UK=UF(UV) THEN 9200
9110 NEXT UV
9115 IF UK=34 THEN 9250
9120 IF UK>31 AND UK<128 THEN 9300
9125 IF UK=170 THEN 9400
9130 IF UK=159 THEN 9450
9135 IF UK=153 THEN 9500
9140 IF UK=136 THEN 9550
9145 IF UK=158 THEN 9650
9150 GOSUB 9900
9155 GOTO 9700
9190 !
9195 !              uscita dal campo :
9200 UF=UV
9205 IF UP=0 OR UP=UL THEN 9700
9210 FOR UV=UP TO UL-1
9215        AWRITE UR,UC+UV,' '
9220 NEXT UV
9225 GOTO 9700
9240 !
9245 !              virgolette :
9250 IF LEN (UP$)#UL THEN GOSUB 9900 @GOTO 9700
9255 AWRITE UR,UC,UP$
9260 UP=0
9265 UI=0
9270 UF=1 ! se si vuole passare automaticamente al campo successivo
9275 GOTO 9700
9290 !
9295 !              carattere :
9300 IF UP=UL THEN GOSUB 9900 @ GOTO 9700
9305 IF UI=0 THEN 9335
9310 FOR UV=UL-2 TO UP STEP -1
9315        AWRITE UR UC+UV
9320        AREAD UC$
9325        AWRITE UR UC+UV+1,UC$
9330 NEXT UV
9335 AWRITE UR UC+UP UK$
9340 UP=UP+1
9345 GOTO 9700
9390 !
9395 !              avanza il cursore :
9400 IF UP=UL THEN GOSUB 9900   ELSE UP=UP+1
9405 GOTO 9700
9440 !
9445 !              indietreggia il cursore :
9450 IF UP=0 THEN GOSUB 9900   ELSE UP=UP-1
9455 GOTO 9700
9490 !
```

```
9495 !              backspace :
9500 IF UP=0 THEN GOSUB 990   @  GOTO 9700
9505 UP=UP-1
9510 AWRITE UR UC+UP
9515 UI=0
9520 GOTO 9700
9540 !
9545 !              delete :
9550 IF UP=UL THEN GOSUB 990   @  GOTO 9700
9555 FOR UV=UP TO UL-2
9560        AWRITE UR,UC+UV+1
9565        AREAD UC$
9570        AWRITE UR UC+UV UC$
9575 NEXT UV
9580 AWRITE UR,UC+UL-1   "
9585 UI=0
9590 GOTO 9700
9640 !
9645 !              insert :
9650 UI=1-UI
9655 GOTO 9700
9690 !
9695 !              fine operazioni sul tasto :
9700 IF UF=0 THEN 9050
9705 RETURN
9790 !
9795 !              visualizza/annulla il cursore :
9800 AWRITE UR UC+UP
9805 AREAD UC$
9810 AWRITE UR,UC+UP,CHR$ (NUM (UC$)+128)
9815 IF UI=0 THEN RETURN
9820 AWRITE UR UC+UP-1
9825 AREAD UC$
9830 AWRITE UR UC+UP-1,CHR$ (NUM (UC$)+128)
9835 RETURN
9890 !
9895 !              segnalazione di errore :
9900 BEEP 122 50
9905 BEEP 149,50
9910 RETURN
```

6.6.5 - Codifica in GWBASIC.

```
8980
8985  GESTIONE DEL SINGOLO CAMPO DATI
8990
8995              definizione valori iniziali :
9000 UP=0
9005 UI=0
9010 UF=0
9015 COLOR 0 3
9045              visualizza il cursore :
9050 GOSUB 9800
```

```
9055            aspetta che sia premuto un tasto :
9060 UK$=INKEY$
9065 IF UK$= '' THEN 9060
9070 UK=ASC (UK$)
9075 IF UK=0 THEN UK=-ASC (RIGHT$ (UK$,1))
9080            fa scomparire il cursore :
9085 GOSUB 9850
9095            individua il tasto premuto :
9100 FOR UV=1 TO 5
9105      IF UK=UF(UV) THEN 9200
9110 NEXT UV
9115 IF UK=34 THEN 9250
9120 IF UK>31 AND UK<128 THEN 9300
9125 IF UK=-77 THEN 9400
9130 IF UK=-75 THEN 9450
9135 IF UK=8 THEN 9500
9140 IF UK=-83 THEN 9550
9145 IF UK=-82 THEN 9650
9150 GOSUB 9900
9155 GOTO 9700
9190 '
9195            uscita dal campo :
9200 UF=UV
9205 IF UP=0 OR UP=UL THEN 9700
9210 FOR UV=UP TO UL-1
9215      LOCATE UR UC+UV
9220      PRINT ' "
9225 NEXT UV
9230 GOTO 9700
9240 '
9245            virgolette :
9250 IF LEN (UP$)<>UL THEN GOSUB 9900 : GOTO 9700
9255 LOCATE UR UC
9260 PRINT UP$
9265 UP=0
9270 UI=0
9280 UF=1    se si vuole passare automaticamente al campo successivo
9285 GOTO 9700
9290
9295            carattere :
9300 IF UP=UL THEN GOSUB 9900 : GOTO 9700
9305 IF UI=0 THEN 9335
9310 FOR UV=UL-2 TO UP STEP -1
9315      UC$=CHR$ (SCREEN (UR UC+UV))
9320      LOCATE UR UC+UV+1
9325      PRINT UC$
9330 NEXT UV
9335 LOCATE UR,UC+UP
9340 PRINT UK$
9345 UP=UP+1
9350 GOTO 9700
9390 '
9395            avanza il cursore :
9400 IF UP=UL THEN GOSUB 9900 ELSE UP=UP+1
```

```
9405 UI=0
9410 GOTO 9700
9440 '
9445              indietreggia il cursore :
9450 IF UP=0 THEN GOSUB 9900 ELSE UP=UP-1
9455 UI=0
9460 GOTO 9700
9490 '
9495              backspace :
9500 IF UP=0 THEN GOSUB 9900 : GOTO 9700
9502 FOR UV=UP-1 TO UL-2
9504      UC$=CHR$ (SCREEN (UR,UC+UV+1))
9506      LOCATE UR UC+UV
9508      PRINT UC$
9510 NEXT UV
9515 LOCATE UR UC+UL-1
9520 PRINT "
9525 UP=UP-1
9530 UI=0
9535 GOTO 9700
9540 '
9545              delete :
9550 IF UP=UL THEN GOSUB 9900 : GOTO 9700
9555 FOR UV=UP TO UL-2
9560      UC$=CHR$ (SCREEN (UR UC+UV+1))
9565      LOCATE UR UC+UV
9570      PRINT UC$
9575 NEXT UV
9580 LOCATE UR UC+UL-1
9585 PRINT "
9590 UI=0
9595 GOTO 9700
9640 '
9645              insert :
9650 UI=1-UI
9655 GOTO 9700
9690 '
9695              fine operazioni sul tasto :
9700 IF UF=0 THEN 9050
9705 COLOR 7,0
9710 RETURN
9790 '
9795              visualizza il cursore :
9800 IF UI=0 THEN LOCATE UR UC+UP,1 6 7 ELSE LOCATE UR,UC+UP 1 3 7
9805 RETURN
9840 '
9845              annulla il cursore :
9850 LOCATE UR UC+UP,0
9855 RETURN
9890 '
9895              segnalazione di errore :
9900 SOUND 415 35 2 18
9905 SOUND 349 23 2 64
9910 RETURN
```

6.7 - Gestione di più campi dati

6.7.1 - Impostazione generale

Per gestire più campi, è necessario individuare un insieme di tasti mediante i quali indicare il passaggio da un campo all'altro o da una pagina all'altra.

Si è ritenuto sufficiente implementare tre modalità di spostamento nell'ambito della singola pagina: al campo successivo, al campo precedente, al primo campo Gli estremi iniziale e finale dell'insieme di campi sono stati considerati collegati, cioè si è definito "campo successivo all'ultimo" il primo campo, e viceversa

Il passaggio al campo successivo deve essere ottenuto, per analogia con l'istruzione INPUT, mediante il tasto ENDLINE (ENTER nel GWBASIC). Per le altre due funzioni non è possibile individuare una corrispondenza con l'istruzione INPUT. Si è pertanto deciso di utilizzare il tasto contrassegnato da una freccia rivolta verso l'alto, che normalmente sposta il cursore alla riga immediatamente superiore, per indicare il passaggio al campo precedente ed il tasto con una freccia inclinata (o tasto HOME nel GWBASIC), che usualmente sposta il cursore all'estremo superiore sinistro dello schermo, per indicare il passaggio al primo campo della pagina.

Si sono inoltre individuate due modalità di spostamento ad altre pagine: alla pagina successiva o alla pagina precedente. Esse sono state associate convenzionalmente nel BASIC HP ai tasti "KEY LABEL" e "shift KEY LABEL", nel GWBASIC ai tasti "PG DN" (*page down*) e "PG UP" (*page up*).

Il sottoprogramma dovrà quindi definire di volta in volta le grandezze necessarie alla routine che gestisce l'immissione dati nel singolo campo, richiamarla e poi esaminare il codice di uscita premuto per individuare a quale campo o pagina passare. In molti casi è opportuno aggungere un controllo immediato sul valore inserito nel campo (ad esempio per verificare se esso è numerico o alfanumerico). Una generalizzazione di questo controllo appare notevolmente complessa; si preferisce pertanto fare riferimento ad un sottoprogramma esterno che potrà essere scritto in base alle esigenze connesse a ciascun insieme di dati.

6.7.2 - Variabili utilizzate

Nel sottoprogramma per la gestione dei campi di una pagina si sono utilizzate le seguenti variabili:

Variabili di ingresso:

UN	numero di campi della pagina;
UR( )	array che contiene la riga in cui è situato ciascun campo;
UC( )	array che contiene la colonna del primo carattere di ciascun campo;

UL( ) array che contiene la lunghezza di ciascun campo;
UM distanza tra due campi logicamente consecutivi

Variabile d'uscita:
UU indicatore di uscita dalla pagina;
UU=0 indica che non si deve ancora uscire dalla pagina;
UU=1 indica che bisogna passare alla pagina successiva;
UU=2 indica che bisogna passare alla pagina precedente.

Variabili interne:
UJ indice che individua il numero d'ordine del campo in esame;
UG indice che individua il numero d'ordine del campo che precede logicamente quello in esame;
UP$ contenuto del campo logicamente precedente;
UR riga in cui è situtato il campo in esame;
UC colonna del primo carattere del campo in esame;
UL lunghezza del campo in esame;
UF numero d'ordine del tasto di uscita dal singolo campo premuto;
UC$ variabile ausiliaria utilizzata per la lettura di un carattere dallo schermo;
UD$ variabile ausiliaria utilizzata per la lettura del contenuto di un campo dallo schermo

6.7.3 - Descrizione del sottoprogramma

La figura 47 mostra lo schema base del sottoprogramma. All'inizio (blocco 1), occorre indicare che il campo in esame è il primo della pagina e che non si è ancora giunti al termine dell'inserimento dati nella pagina corrente. Bisogna poi *ripetere* una sequenza di operazioni (blocchi 2-6), *finchè non* è stato dato il comando di fine pagina. Nell'ordine, si legge il valore del campo precedente, si definiscono riga, colonna iniziale e lunghezza del campo in esame, si richiama il sottoprogramma che gestisce l'immissione di dati in esso, poi quello che consente un eventuale controllo del valore inserito, ed infine si esamina il numero d'ordine del tasto di uscita dal campo premuto e si eseguono le operazioni ad esso connesse

Il blocco 2 richiede innanzitutto la definizione di quale sia il campo cui far riferimento nel caso di valori che si ripetono (da assegnare col tasto *virgolette*). Di solito si tratta del campo immediatamente precedente, sulla base dell'ordine definito. Si pensi però ad una tabella di dati, ad esempio un insieme di coordinate x y z. Esse vengono assegnate nell'ordine x1, y1, z1, x2, y2, z2 ... ma ripetitività sarà in genere tra grandezze omonime (x1, x2 ...) e quindi con riferimento al campo *m* volte precedente (3 nell'esempio) in campo in esame. Il numero d'ordine UG del campo "logicamente" precedente è pertanto definito in funzione del valore UM che indica la distanza tra campi corrispondenti.

Occorre poi leggere dal video il contenuto di tale campo. Poichè una operazione di tale genere è necessaria anche in altre fasi, si è utilizzato un sottoprogramma (linee a partire da 8800) che richiede in ingresso il numero del campo UG e fornisce come uscita il suo contenuto UD$.

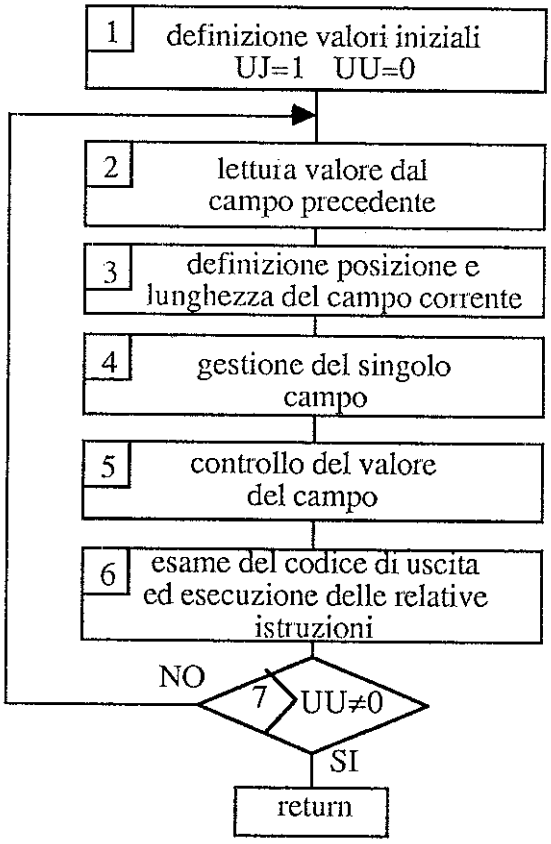


Fig. 47

Il blocco 3 preleva riga, colonna iniziale e lunghezza del campo dagli ar-

ray che contengono l'insieme di tali valori, in base all'indice UJ

I blocchi 4 e 5 sono costituiti rispettivamente dal sottoprogramma di gestione del singolo campo, esaminato nel paragrafo precedente, e da un sottoprogramma per il controllo dei valori immessi nei campi, che potrà essere definito di volta in volta (linee da 6000). Quest'ultimo può modificare il numero d'ordine del tasto di uscita dal campo UF, azzerandolo per indicare che il valore immesso non è corretto ed occorre pertanto assegnarlo nuovamente.

Il blocco 6 presenta una struttura "CASE ... OF " Il selettore di caso è costituita dalla variabile UF, che può presentare valori compresi tra 0 e 5, cui corrispondono i blocchi logici descritti di seguito

A) Ritorna sullo stesso campo (UF=0).

È la situazione che si verifica se è riscontrato un errore nel valore immesso nel campo L'indice del campo corrente rimane inalterato

B) Passa al campo successivo (UF=1; linee da 8200).

Occorre incrementare l'indice UJ. Poichè il campo successivo all'ultimo deve essere il primo, si usa l'espressione UJ MOD UN + 1 che fornisce il valore UJ+1 se UJ è minore di UN, il valore 1 quando UJ=UN

C) Passa al campo precedente (UF=2; linee da 8300)

Occorre decrementare l'indice UJ. Poichè il campo precedente al primo deve essere l'ultimo, si usa l'espressione (UJ+UN−2) MOD UN+1 che fornisce il valore UJ−1 se UJ è maggiore di 1, il valore UN se UJ=1.

D) Passa al primo campo della pagina (UF=3; linee da 8400).

L'indice del campo viene posto pari ad 1

E) Passa alla pagina successiva (UF=4; linee da 8500).

Pone l'indicatore di fine pagina UU pari ad 1.

F) Passa alla pagina precedente (UF=5; linee da 8600)

Pone l'indicatore di fine pagina UU pari a −1

6.7.4 - Codifica in BASIC HP.

```
7980 !
7985 !  GESTIONE DI PIU CAMPI
7990 !
7995 !              definisce valori iniziali :
8000 TAKE KEYBOARD
8005 UJ=1
8010 UU=0
8035 !              preleva il valore del campo precedente :
8040 UG=(UJ+UN-UM-1) MOD UN+1
8045 GOSUB 8800
```

```
8050 UP$=UD$
8055 !              definisce riga  colonna  lunghezza del campo in esame :
8060 UR=UR(UJ)
8065 UC=UC(UJ)
8070 UL=UL(UJ)
8075 !              gestione del singolo campo :
8080 GOSUB 9000
8085 !              controllo del valore del campo :
8090 GOSUB 6000
8095 !              esamina codice uscita singolo campo :
8100 IF UF=0 THEN 8700
8105 ON UF GOTO 8200,8300,8400 8500 8600
8190 !
8195 !              passa al campo successivo :
8200 UJ=UJ MOD UN+1
8205 GOTO 8700
8290 !
8295 !              passa al campo precedente :
8300 UJ=(UJ+UN-2) MOD UN+1
8305 GOTO 8700
8390 !
8395 !              passa al primo campo della pagina :
8400 UJ=1
8405 GOTO 8700
8490 !
8495 !              passa alla pagina successiva :
8500 UU=1
8505 GOTO 8700
8590 !
8595 !              passa alla pagina precedente :
8600 UU=-1
8605 GOTO 8700
8690 !
8695 !              fine individuazione campo successivo :
8700 IF UU=0 THEN 8040
8705 RELEASE KEYBOARD
8710 RETURN
8780 !
8785 !              LETTURA DI UNA STRINGA DAL VIDEO
8790 !
8800 UD$=    !                          per una maggiore velocità di esecuzione
8805 FOR UV=0 TO UL(UG)-1 !             modificare così il sottoprogramma:
8810       AWRITE UR(UG) UC(UG)+UV !
8815       AREAD UC$ !                  8800 UD$=
8820       UD$=UD$&UC$ !                8805 AWRITE UR(UG), UC(UG)
8825 NEXT UV !                          8810 AREAD UD$[1,UL(UG)]
8830 RETURN !                           8815 RETURN
```

6.7.5 - Codifica in GWBASIC

```
7980
7985              GESTIONE DI PIU CAMPI
7990
```

```
7995            definisce valori iniziali :
8000 UJ=1
8005 UU=0
8035            preleva il valore del campo precedente :
8040 UG=(UJ+UN-UM-1) MOD UN+1
8045 GOSUB 8800
8050 UP$=UD$
8055            definisce riga  colonna  lunghezza del campo in esame :
8060 UR=UR(UJ)
8065 UC=UC(UJ)
8070 UL=UL(UJ)
8075            gestione del singolo campo :
8080 GOSUB 9000
8085            controllo del valore del campo :
8090 GOSUB 6000
8095            esamina codice uscita singolo campo :
8100 IF UF=0 THEN 8700
8105 ON UF GOTO 8200,8300,8400,8500,8600
8190
8195            passa al campo successivo :
8200 UJ=UJ MOD UN+1
8205 GOTO 8700
8290 '
8295            passa al campo precedente :
8300 UJ=(UJ+UN-2) MOD UN+1
8305 GOTO 8700
8390
8395            passa al primo campo della pagina :
8400 UJ=1
8405 GOTO 8700
8490 '
8495            passa alla pagina successiva :
8500 UU=1
8505 GOTO 8700
8590 '
8595            passa alla pagina precedente :
8600 UU=-1
8605 GOTO 8700
8690 '
8695            fine individuazione campo successivo :
8700 IF UU=0 THEN 8040
8705 RETURN
8780 '
8785          LETTURA DI UNA STRINGA DAL VIDEO
8790
8800 UD$=''
8805 FOR UV=0 TO UL(UG)-1
8810     UC$=CHR$(SCREEN(UR(UG),UC(UG)+UV))
8815     UD$=UD$+UC$
8820     NEXT UV
8825 RETURN
```

6.8 - Gestione di più pagine

La gestione di un gruppo di pagine può essere impostata in due maniere differenti Quella cui si è fatto implicitamente riferimento parlando di "pagina precedente" e "pagina successiva" considera le pagine come un insieme da percorrere in sequenza La sua struttura logica è riportata in figura 48. Si sono utilizzate le variabili UH ed UZ per indicare rispettivamente il numero d'ordine della pagina in esame ed il numero totale di pagine. Il valore di UH viene inizialmente posto pari ad 1 Dopo ciò, si *ripete* una serie di operazioni (immissione dati nella pagina corrente ed individuazione della nuova pagina in base al valore di uscita UU) *finchè non* si è superata l'ultima pagina.

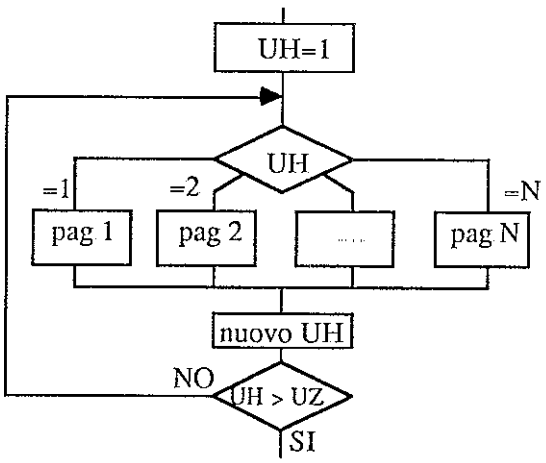


Fig 48

In alternativa, si può prevedere un menù che contenga l'elenco delle pagine, più una opzione che indichi la fine dell'esame del gruppo di pagine. Mediante esso si può scegliere di volta in volta a quale pagina passare La struttura logica (fig 49) è ancora realizzata con un ciclo che *ripete* la presentazione del menù e l'immissione di dati nella pagina prescelta *finchè non* si riscontra l'indicazione di fine. Con questa impostazione non è più necessaria una doppia uscita dalla pagina (in avanti o all'indietro) perchè si torna in ogni caso al menù.

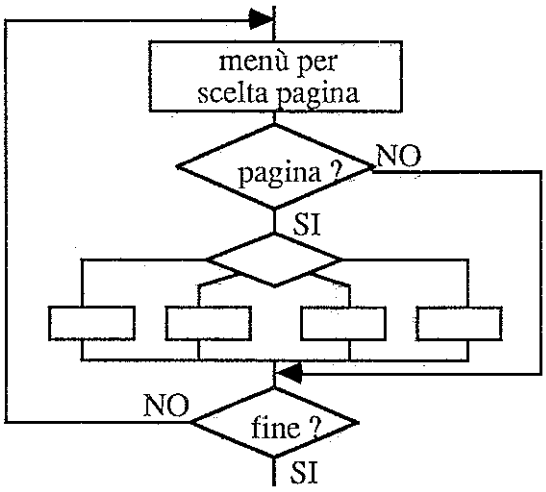


Fig. 49

L'impostazione sequenziale è particolarmente adatta a quei casi nei quali è necessario fornire valori a tutte le pagine; un tipico esempio è costituito dai dati di ingresso per un calcolo strutturale, che non può essere effettuato se manca qualche informazione. L'impostazione selettiva è invece comoda nei casi in cui è sufficiente intervenire solo su alcune pagine di dati; questa situazione si riscontra ad esempio nella gestione di grossi archivi, che può richiedere la modifica di una limitata parte delle informazioni in essi contenute. Nei programmi più complessi si possono creare strutture di gestione delle pagine molto elaborate, che racchiudono entrambe le impostazioni descritte; ad esempio, si può prevedere un menù principale che rinvia a più menù secondari, ciascuno dei quali a sua volta indirizza a blocchi di pagine da percorrere sequenzialmente.

6.9 - Altri sottoprogrammi di utilità generale.

6.9.1 - Descrizione e modalità d'uso dei sottoprogrammi.

Per la gestione di ciascuna pagina, occorre preliminarmente definire numero, posizione e dimensione dei campi, nonchè le scritte esplicative. È poi necessario evidenziare i campi sullo schermo ed eventualmente met-

tervi dei valori iniziali Dopo aver richiamato il sottoprogramma di immissione dati bisogna infine leggere i valori dai campi per assegnarli alle variabili del programma principale Tutte queste funzioni possono essere agevolate da un insieme di sottoprogrammi, che si analizzano di seguito.

A) Lettura posizione e dimensione campi.

Legge da un insieme di istruzioni DATA il numero di campi della pagina e, per ciascuno di essi, riga, colonna d'inizio, lunghezza. È indispensabile riposizionare il puntatore ogni volta che si devono riutilizzare i DATA, e quindi definirne per ogni pagina l'inizio con l'istruzione RESTORE.

Modalità d'uso:

```
        RESTORE n
        GOSUB 7000
n       DATA num-campi
        DATA riga-1  col-1, lungh-1, riga-2, col-2, lungh-2
```

B) Lettura posizione e dimensione campi - per tabelle.

Col termine "tabella" si intende un insieme di campi disposto secondo una matrice bidimensionale. Un esempio già citato è costituito da un gruppo di coordinate x y z. Usualmente, il numero di righe può essere una variabile (il cui valore deve ovviamente essere stato definito prima di arrivare alla pagina in esame), mentre il numero di campi per riga è fisso (tre, nell'esempio)

Il sottoprogramma richiede come valore di ingresso UT, numero totale di righe di campi. Esso legge da istruzioni DATA il numero di campi per riga, la riga dello schermo nella quale posizionare la prima riga di campi, la colonna d'inizio e la lunghezza di ciascuno dei campi di una riga. In base a queste informazioni determina posizione e dimensione di tutti i campi della tabella.

Modalità d'uso:

```
        RESTORE n
        UT= num-righe
        GOSUB 7100
n       DATA campi-per-riga, riga-inizio
        DATA col-1  lungh-1  col-2  lungh-2
```

C) Lettura e visualizzazione scritte.

Legge da un insieme di istruzioni DATA il numero di scritte e, per ciascuna di esse, riga, colonna d'inizio e testo. Le scritte lette vengono visualizzate nello schermo.

Modalità d'uso:

```
..      RESTORE n
..      GOSUB 7200
n       DATA num-scritte
        DATA riga-1 col-1 scritta-1, riga-2 col-2 scritta-2
```

D) Lettura e visualizzazione scritte ripetitive - per tabelle.

Nella maschere per tabelle vi sono spesso scritte che si ripetono identiche per tutte le righe da cui esse sono composte. Il sottoprogramma richiede come dato d'ingresso il numero di righe UT. Esso legge la posizione d'inizio (riga e colonna) della scritta ed il suo testo e la visualizza in UT righe consecutive.

Modalità d'uso:

```
..      RESTORE n
        UT= num-righe
        GOSUB 7300
n       DATA riga colonna scritta
```

E) Visualizzazione indici - per tabelle.

Un'informazione che spesso compare nelle maschere per tabelle è l'indice di ciascuna riga. Gli indici sono normalmente numeri consecutivi. Il sottoprogramma richede come dati d'ingresso il numero di righe UT e l'indice della prima riga UA. Legge da istruzioni DATA riga e colonna d'inizio e quindi calcola e visualizza gli indici

Modalità d'uso:

```
..      RESTORE n
..      UT= num-righe
        UA= primo-indice
        GOSUB 7400
n       DATA riga, colonna
```

F) Visualizzazione fondo campo.

È necessario avere già definito posizione e lunghezza dei campi della pagina. Il sottoprogramma visualizza il carattere di fondo, mettendo così in risalto la posizione dei campi. Si ricorda che nel BASIC HP, che non consente l'uso di differenti tonalità, i campi sono evidenziati col simbolo " . . "; ciò deve però essere fatto solo per i campi che non contengono un valore iniziale

Modalità d'uso:

```
..      GOSUB 7500
```

G) Lettura di un valore da un campo del video.

Il sottoprogramma richiede come dato d'ingresso il numero d'ordine del campo UJ. Esso legge dal video il contenuto del campo e lo pone nella variabile alfanumerica UD$; ne pone inoltre il valore nella variabile numerica UD. È possibile mantenere inalterato il valore di UJ (subroutine 7600) oppure incrementarlo automaticamente di una unità (subroutine 7650); questa seconda possibilità si presenta molto comoda quando si vuole leggere il valore di più campi consecutivi.

Modalità d'uso:

```
        UJ= num-ord-campo
        GOSUB 7600          non incrementa UJ
oppure:
        UJ= num-ord-campo
        GOSUB 7650          incrementa UJ
```

H) Scrittura di un valore su un campo del video.

Occorre definire preliminarmente il numero d'ordine del campo UJ ed il valore da visualizzare, UD se numerico, UD$ se alfanumerico. A seconda del tipo di valore si userà rispettivamente la subroutine 7700 o 7725. Nel BASIC HP è anche utile poter riempire il campo col simbolo di default " " (subroutine 7750). Il valore di UJ viene sempre incrementato.

Modalità d'uso:

```
        UJ= num-ord-campo
        UD= val-numerico
        GOSUB 7700          per valore numerico
oppure:
        UJ= num-ord-campo
        UD$= val-alfanumerico
        GOSUB 7725          per valore alfanumerico
oppure:
        UJ= num-ord-campo
        GOSUB 7750          per simbolo " " (solo BASIC HP)
```

I) Dimensionamento variabili e definizione codici d'uscita.

Dimensiona le variabili con indice utilizzate nelle subroutine descritte in questo capitolo e definisce i codici d'uscita. Deve pertanto essere sempre richiamato dal programma principale prima di iniziare l'uso della maschera. Il valore massimo dell'indice per UR, UC ed UL (linea 7800) deve essere commisurato al massimo numero di campi previsto per una pagina.

Modalità d'uso:

```
GOSUB 7800
```

6.9.2 - Codifica in BASIC HP

```
6980 !
6985 !          LETTURA POSIZIONE E DIMENSIONE CAMPI
6990 !
7000 READ UN
7005 FOR UJ=1 TO UN
7010     READ UR(UJ) UC(UJ) UL(UJ)
7015 NEXT UJ
7020 UM=1
7025 RETURN
7080 !
7085 !          LETTURA POSIZIONE E DIMENSIONE CAMPI per tabelle
7090 !
7100 READ UM,UR
7105 UN=UT*UM
7110 FOR UV=1 TO UM
7115     READ UC,UL
7120     FOR UJ=0 TO UN-UM STEP UM
7125         UR(UJ+UV)=UR+UJ/UM
7130         UC(UJ+UV)=UC
7135         UL(UJ+UV)=UL
7140     NEXT UJ
7145 NEXT UV
7150 RETURN
7180 !
7185 !          LETTURA E VISUALIZZAZIONE SCRITTE
7190 !
7200 READ US
7205 FOR UV=1 TO US
7210     READ UR UC US$
7215     AWRITE UR UC US$
7220 NEXT UV
7225 RETURN
7280 !
7285 !          LETTURA E VISUALIZZAZIONE SCRITTE RIPETITIVE per tabelle
7290 !
7300 READ UR,UC,US$
7305 FOR UV=0 TO UT-1
7310     AWRITE UR+UV UC US$
7315 NEXT UV
7320 RETURN
7380 !
7385 !          VISUALIZZAZIONE INDICI per tabelle
7390 !
7400 READ UR,UC
7405 FOR UV=0 TO UT-1
7410     AWRITE UR+UV UC VAL$ (UA+UV)
```

```
7415 NEXT UV
7420 RETURN
7480 !
7485 !              VISUALIZZAZIONE FONDO CAMPO
7490 !
7500 FOR UJ=1 TO UN
7505       FOR UV=0 TO UL(UJ)-1
7510             AWRITE UR(UJ) UC(UJ)+UV ''
7515       NEXT UV
7520 NEXT UJ
7525 RETURN
7580 !
7585 !              LETTURA DI UN VALORE DA UN CAMPO DEL VIDEO
7590 !
7600 UG=UJ
7605 GOSUB 8800
7610 UD=0
7615 ON ERROR GOTO 7625
7620 UD=VAL (UD$)
7625 OFF ERROR
7630 RETURN
7645 !
7650 GOSUB 7600
7655 UJ=UJ+1
7660 RETURN
7680 !
7685 !              SCRITTURA DI UN VALORE SU UN CAMPO DEL VIDEO
7690 !
7700 UD$=VAL$ (UD)
7705 FOR UV=LEN (UD$)+1 TO UL(UJ)
7710       UD$=UD$&' '
7715 NEXT UV
7720 !
7725 AWRITE UR(UJ),UC(UJ),UD$
7730 UJ=UJ+1
7735 RETURN
7745 !
7750 UD$=
7755 FOR UV=1 TO UL(UJ)
7760       UD$=UD$&' '
7765 NEXT UV
7770 GOTO 7725
7780 !
7785 !              DIMENSIONAMENTO VARIABILI E DEFINIZIONE CODICI DI USCITA
7790 !
7800 INTEGER UF(5),UR(60),UC(60),UL(60)
7805 DIM UC$[1],UD$[80],UP$[80],US$[80]
7810 UF(1)=154
7815 UF(2)=163
7820 UF(3)=152
7825 UF(4)=150
7830 UF(5)=96
7835 RETURN
```

6.9.3 - Codifica in GWBASIC

```
6980
6985              LETTURA POSIZIONE E DIMENSIONE CAMPI
6990
7000 READ UN
7005 FOR UJ=1 TO UN
7010      READ UR(UJ),UC(UJ),UL(UJ)
7015 NEXT UJ
7020 UM=1
7025 RETURN
7080 '
7085              LETTURA POSIZIONE E DIMENSIONE CAMPI per tabelle
7090
7100 READ UM,UR
7105 UN=UT*UM
7110 FOR UV=1 TO UM
7115      READ UC,UL
7120      FOR UJ=0 TO UN-UM STEP UM
7125           UR(UJ+UV)=UR+UJ/UM
7130           UC(UJ+UV)=UC
7135           UL(UJ+UV)=UL
7140      NEXT UJ
7145 NEXT UV
7150 RETURN
7180 '
7185              LETTURA E VISUALIZZAZIONE SCRITTE
7190
7200 READ US
7205 FOR UV=1 TO US
7210      READ UR,UC US$
7215      LOCATE UR,UC
7220      PRINT US$
7225 NEXT UV
7230 RETURN
7280 '
7285              LETTURA E VISUALIZZAZIONE SCRITTE RIPETITIVE per tabelle
7290
7300 READ UR,UC,US$
7305 FOR UV=0 TO UT-1
7310      LOCATE UR+UV,UC
7315      PRINT US$
7320 NEXT UV
7325 RETURN
7380 '
7385              VISUALIZZAZIONE INDICI per tabelle
7390
7400 READ UR,UC
7405 FOR UV=0 TO UT-1
7410      LOCATE UR+UV UC
7415      PRINT UA+UV
7420 NEXT UV
7425 RETURN
7480 '
```

```
7485              VISUALIZZAZIONE FONDO CAMPO
7490
7500 COLOR 0,3
7505 FOR UJ=1 TO UN
7510      FOR UV=0 TO UL(UJ)-1
7515           LOCATE UR(UJ) UC(UJ)+UV
7520           PRINT "
7525      NEXT UV
7530 NEXT UJ
7535 COLOR 7 0
7540 RETURN
7580 '
7585              LETTURA DI UN VALORE DA UN CAMPO DEL VIDEO
7590
7600 UG=UJ
7605 GOSUB 8800
7610 UD=VAL (UD$)
7615 RETURN
7645 '
7650 GOSUB 7600
7655 UJ=UJ+1
7660 RETURN
7680 '
7685              SCRITTURA DI UN VALORE SU UN CAMPO DEL VIDEO
7690 '
7700 UD$=STR$(UD)
7705 IF LEFT$(UD$,1)="   THEN UD$=RIGHT$(UD$ LEN(UD$)-1)
7710 FOR UV=LEN(UD$)+1 TO UL(UJ) : UD$=UD$+" " : NEXT UV
7720 '
7725 LOCATE UR(UJ),UC(UJ)
7730 COLOR 0,3
7735 PRINT UD$
7740 COLOR 7,0
7745 UJ=UJ+1
7750 RETURN
7780 '
7785              DIMENSIONAMENTO VARIABILI E DEFINIZIONE CODICI DI USCITA
7790
7800 DIM UF(5),UR(60),UC(60) UL(60)
7805 UF(1)=13
7810 UF(2)=-72
7815 UF(3)=-71
7820 UF(4)=-81
7825 UF(5)=-73
7830 RETURN
```

6.10 - Esempio: verifica a flessione

6.10.1 - Descrizione del programma

Un esempio piuttosto semplice è costituito dalla utilizzazione dell'ingresso

dati con maschera in un programma per la verifica di una sezione rettangolare con doppia armatura soggetta a flessione semplice. L'analisi di tale problema e la relativa codifica sono riportate nei paragrafi 3.3 e 4.6. La struttura del programma prevede quattro fasi: definizione della maschera, immissione dei dati, elaborazione, visualizzazione dei risultati. Per consentire l'effettuazione di più verifiche, le ultime tre fasi sono state inserite in un ciclo. Così, dopo la visualizzazione dei risultati l'elaborazione si arresta, ma premendo il tasto "CONT" ("F5" nel GWBASIC) riprende con la cancellazione dei risultati precedenti ed il ritorno alla fase di immissione dati. Si noti che in questo modo i valori precedentemente utilizzati sono ancora presenti nei campi; è pertanto possibile modificarli anzichè riassegnarli nuovamente.

Le istruzioni necessarie per la definizione della maschera sono raggruppate in un sottoprogramma (linee a partire da 1000). Il puntatore ai DATA viene posizionato in modo da individuare la linea 1500, a partire dalla quale sono riportate le informazioni necessarie per i sottoprogrammi 7000, 7200 e 7500, descritti nel paragrafo precedente. I valori utilizzati per righe e colonne nella versione in GWBASIC sono superiori di una unità rispetto a quelli del BASIC HP, perchè nei due linguaggi tali elementi vengono numerati a partire rispettivamente da 1 e da 0.

Le istruzioni per l'ingresso dati sono contenute nel sottoprogramma che inizia alla linea 2000. Viene richiamata la subroutine 8000 (immissione dati nei campi) e poi, per ciascun campo, la 7650 (che ne preleva il valore, da assegnare alle variabili del programma principale).

Si è definito anche il sottoprogramma per il controllo dei valori immessi nei campi (linee da 6000). Per una corretta esecuzione del calcolo, è necessario che le grandezze B, H, A_f siano maggiori di 0, che d sia minore di H (cioè del valore del campo precedente) e che A'_f ed M non siano negativi. Se la condizione non è verificata, viene segnalato l'errore e si pone UF=0 per indicare che non si può passare ad un altro campo.

6.10.2 - Codifica in BASIC HP

```
 10 !          VERIFICA A FLESSIONE SEMPLICE
 20 !          sezione rettangolare a doppia armatura
 30 !
 80 !          PROGRAMMA PRINCIPALE
 90 !
100 OPTION BASE 1
110 GOSUB 7800
120 GOSUB 1000
490 !
500 GOSUB 2000
580 !
590 !          calcolo
600 N0=15
610 M=M*100
620 H1=H-D
```

```
630 A=A1+A2
640 X=N0*A/B*(-1+SQR (1+2*B*(A1*H1+A2*D)/N0/A^2))
650 S1=M/(B*X/2*(H1-X/3)+N0*A2/X*(X-D)*(H1-D))
660 S2=N0*S1*(H1-X)/X
670 !              visualizzazione risultati
680 AWRITE 10,0
690 PRINT USING  4A,3DZ DD.3A  ;  X = ' X   cm
700 AWRITE 11,0
710 PRINT USING  9A 5DZ DD.7A  ; "sigma c =  ,S1   kg/cm2"
720 AWRITE 12,0
730 PRINT USING "9A,5DZ DD,7A" ;  sigma f =" S2," kg/cm2
790 !
800 AWRITE 14.0  Per continuare premi [CONT]
810 PAUSE
820 !              cancellazione risultati
830 FOR V=10 TO 14
840       AWRITE V 0,"
850 NEXT V
860 !
870 GOTO 500
980 !
990 !              DEFINIZIONE MASCHERA
1000 CLEAR
1010 RESTORE 1500
1020 GOSUB 7000
1030 GOSUB 7200
1040 GOSUB 7500
1050 RETURN
1490 !
1500 DATA 6
1510 DATA 3,23,5 4,23,5,5,23 5 6 23,5,7,23 5 8,23,5
1520 DATA 8
1530 DATA 0,0 VERIFICA A FLESSIONE
1540 DATA 1 0 sezione rettangolare          n=15
1550 DATA 3,0,Base              B =         cm
1560 DATA 4,0 Altezza           H =         cm
1570 DATA 5,0 Copriferro        d =         cm
1580 DATA 6,0 Armatura tesa     Af =        cm2
1590 DATA 7 0,Armatura compr    A'f=        cm2
1600 DATA 8,0 Momento           M =         kgm
1980 !
1990 !              INGRESSO DATI CON MASCHERA
2000 GOSUB 8000
2010 UJ=1
2020 GOSUB 765  @     UD
2030 GOSUB 765  @     UD
2040 GOSUB 765  @     UD
2050 GOSUB 765  @     UD
2060 GOSUB 765  @     UD
2070 GOSUB 765  @     UD
2080 RETURN
5980 !
5985 !              CONTROLLO DEL VALORE DEI CAMPI
5990 !
```

```
6000 GOSUB 7600
6010 ON UJ GOTO 6100 6100,6150,6100,6200 6200
6090 !
6100 IF UD<= 0 THEN GOSUB 6500
6110 GOTO 6300
6140 !
6150 IF UD>= VAL (UP$) THEN GOSUB 6500
6160 GOTO 6300
6190 !
6200 IF UD<0 THEN GOSUB 6500
6210 GOTO 6300
6290 !
6300 RETURN
6490 !
6500 AWRITE 14,0, 'ERRORE - valore non ammissibile
6510 BEEP
6520 WAIT 1000
6530 AWRITE 14,0 "
6540 UF=0
6550 RETURN
```

6.10.3 - Codifica in GWBASIC

```
 10            VERIFICA A FLESSIONE SEMPLICE
 20            sezione rettangolare a doppia armatura
 30
 80            PROGRAMMA PRINCIPALE
 90
100 OPTION BASE 1
110 GOSUB 7800
120 GOSUB 1000
490 '
500 GOSUB 2000
580 '
590            calcolo
600 N0=15
610 M=M*100
620 H1=H-D
630 A=A1+A2
640 X=N0*A/B*(-1+SQR (1+2*B*(A1*H1+A2*D)/N0/A^2))
650 S1=M/(B*X/2*(H1-X/3)+N0*A2/X,(X-D)*(H1-D))
660 S2=N0*S1*(H1-X)/X
670            visualizzazione risultati
680 LOCATE 11,1
690 PRINT USING "X = ####.## cm"; X
700 LOCATE 12,1
710 PRINT USING "sigma c = ###### ## kg/cm2"; S1
720 LOCATE 13,1
730 PRINT USING "sigma f = ######.## kg/cm2"; S2
790 '
800 LOCATE 15,1
810 PRINT "Per continuare premi CONT (F5)
820 STOP
```

```
 830              cancellazione risultati
 840 FOR V=11 TO 18
 850       LOCATE V 1
 860       PRINT '
 870 NEXT V
 880 '
 890 GOTO 500
 980 '
 990              DEFINIZIONE MASCHERA
1000 CLS
1010 RESTORE 1500
1020 GOSUB 7000
1030 GOSUB 7200
1040 GOSUB 7500
1050 RETURN
1490 '
1500 DATA 6
1510 DATA 4,24,5,5,24 5 6 24 5,7,24,5,8 24,5,9,24,5
1520 DATA 8
1530 DATA 1,1,VERIFICA A FLESSIONE
1540 DATA 2 1 sezione rettangolare          n=15
1550 DATA 4,1,Base              B =         cm
1560 DATA 5,1.Altezza           H =         cm
1570 DATA 6,1,Copriferro        d =         cm
1580 DATA 7 1 Armatura tesa     Af =        cm2
1590 DATA 8,1,Armatura compr    A'f=        cm2
1600 DATA 9,1 Momento           M =         kgm
1980 '
1990              INGRESSO DATI CON MASCHERA
2000 GOSUB 8000
2010 UJ=1
2020 GOSUB 7650 : B=UD
2030 GOSUB 7650 : H=UD
2040 GOSUB 7650 : D=UD
2050 GOSUB 7650 : A1=UD
2060 GOSUB 7650 : A2=UD
2070 GOSUB 7650 : M=UD
2080 RETURN
5980 '
5985              CONTROLLO DEL VALORE DEI CAMPI
5990
6000 GOSUB 7600
6010 ON UJ GOTO 6100.6100,6150 6100,6200,6200
6090 '
6100 IF UD<= 0 THEN GOSUB 6500
6110 GOTO 6300
6140 '
6150 IF UD>= VAL(UP$) THEN GOSUB 6500
6160 GOTO 6300
6190 '
6200 IF UD<0 THEN GOSUB 6500
6210 GOTO 6300
6290 '
6300 RETURN
```

```
6490
6500 LOCATE 15,1
6510 PRINT "ERRORE – valore non ammissibile
6520 BEEP
6530 FOR UV=1 TO 1500 : NEXT UV
6540 LOCATE 15,1
6550 PRINT "
6560 UF=0
6570 RETURN
```

6.11 - Esempio: tabella di dati (coordinate nello spazio)

6.11.1 - Descrizione del programma

L'organizzazione dell'ingresso dati con maschera si presenta un pò più complessa, rispetto al caso precedente, quando si devono gestire più pagine di informazioni. Si prende in esame a titolo esemplificativo un programma che consente di definire le coordinate di un insieme di punti rispetto ad un sistema di riferimento x,y,z. Un simile programma può costituire uno dei blocchi di input necessari per la risoluzione di uno schema tridimensionale di telaio. Per agevolare l'utente, deve essere possibile sia assegnare nuovi valori, sia modificare valori conservati in precedenza sulla memoria di massa. I dati inseriti devono essere memorizzati in un file, per poterli utilizzare in seguito.

Il programma principale presenta una struttura analoga a quella già mostrata in figura 48. Un blocco di istruzioni (linee da 400 a 560) indirizza alle diverse pagine, individuate dall'indice UH; per la definizione della maschera e l'immissione di valori è previsto un diverso sottoprogramma per ciascuna di esse. Il blocco finale (a partire dalla linea 600) effettua la memorizzazione su disco dei valori assegnati. Se il programma è parte di una sequenza, l'istruzione conclusiva "STOP" può essere sostituita dal concatenamento al programma successivo "CHAIN *nome-file*".

La pagina 1 richiede le prime informazioni che occorre fornire: la scelta se modificare valori già memorizzati, e in caso affermativo il nome del file che li contiene. Nella stessa pagina va inserito anche il nome del file in cui devono essere conservati alla fine i valori delle coordinate. La subroutine 1000 gestisce l'immissione di questi dati (linee 1000–1100 e DATA 1500–1550) e, se richiesto, la lettura dei valori da modificare (linee 1130–1200): numero di punti e coordinate di ciascuno di essi.

La pagina 2 (subroutine 2000) è dedicata esclusivamente alla definizione del numero di punti. Questa informazione non può essere accorpata alla pagina precedente, perchè in caso di modifica di dati memorizzati il suo valore iniziale deve essere già stato letto sul file indicato nella pagina 1. Non può inoltre essere riunita alle informazioni successive (coordinate dei punti) perchè il loro numero (e quindi la maschera e i campi) dipende da essa

L'ultimo tipo di pagina è utilizzato per l'assegnazione delle coordinate. Lo schermo ha dimensioni limitate ed è pertanto necessario dividere i punti in più gruppi. La pagina 3 conterrà i punti da 1 a 20, la 4 quelli da 21 a 40, e così via. Le pagine sono formalmente identiche e vengono quindi gestite dallo stesso sottoprogramma (linee a partire da 3000) Si sono indicati con I1 e I2 il numero d'ordine del primo e dell'ultimo punto della pagina, e tali valori sono di volta in volta calcolati in funzione di UH (linee 3020–3030). Poichè le coordinate costituiscono una tabella, si sono utilizzati per la individuazione dei campi e la visualizzazione delle scritte della maschera i sottoprogrammi 7100, 7200, 7300, 7400

Ogni volta che si gestisce una sequenza di pagine, che può essere percorsa andando sia in avanti che all'indietro, è necessario sapere in ogni momento quali valori siano già stati assegnati Nel visualizzare la maschera di ciascuna pagina è infatti necessario decidere se riempire i campi con valori già definiti o con simboli di default. Nel programma si sono utilizzate le variabili VH ed I3 per indicare rispettivamente l'ultima pagina esaminata e l'ultimo punto di cui sono definite le coordinate.

Infine, il controllo dei valori (subroutine 6000) è limitato a solo due variabili. Si verifica che il primo campo della pagina 1 (richiesta se si vuole modificare dati memorizzati) contenga 'SI' oppure 'NO', respingendo altre risposte. Si verifica inoltre che l'unico campo della pagina 2 (numero dei punti) rientri in un intervallo ammissibile (dal valore minimo 1 ad un massimo che si è posto pari a 100, perchè tale è la dimensione massima degli array X, Y, Z).

6.11.2 - Variabili utilizzate.

Variabili da definire nella prima pagina:

R$	contiene "SI" o "NO" per indicare se si vuole, o no, modificare valori memorizzati su disco;
FI$	nome del file che contiene valori da modificare;
FO$	nome del file in cui conservare i valori assegnati.

Variabile da definire nella seconda pagina:

N	numero di punti.

Variabili da definire nelle altre pagine:

X( )	array che contiene le coordinate x;
Y( )	array che contiene le coordinate y;
Z( )	array che contiene le coordinate z.

Variabili di ingresso o uscita dei sottoprogrammi standard della maschera:

UA, UD, UD$, UF, UI, UT, UU.

Altre variabili:

I	indice che individua il punto generico;
I1	numero d'ordine del primo punto della pagina in esame;
I2	numero d'ordine dell'ultimo punto della pagina in esame;
I3	numero d'ordine dell'ultimo punto di cui sono state già definite le coordinate;
UH	numero d'ordine della pagina in esame;
UZ	numero totale di pagine;
VH	numero d'ordine dell'ultima pagina di cui sono già stati definiti i valori.

6.11.3 - Codifica in BASIC HP.

```
 10 !            LETTURA COORDINATE NELLO SPAZIO
 20 !
 80 !            PROGRAMMA PRINCIPALE
 90 !
100 OPTION BASE 1
110 PAGESIZE 24
120 DIM X(100) Y(100),Z(100)
190 !
200 GOSUB 7800
210 VH=0
220 I3=0
230 UZ=3
240 UH=1
390 !           scelta tra le pagine
400 IF UH=1 THEN 450
410 IF UH=2 THEN 480
420 GOTO 510
440 !
450 GOSUB 1000
460 GOTO 540
470 !
480 GOSUB 2000
490 GOTO 540
500 !
510 GOSUB 3000
520 GOTO 540
530 !
540 UH=UH+UU
550 IF UH=0 THEN UH=1
560 IF UH<= UZ THEN 400
590 !           memorizzazione valori su disco
600 ASSIGN# 1 TO FO$
610 PRINT# 1 ; N
620 FOR I=1 TO N
630       PRINT# 1 ; X(I),Y(I) Z(I)
640 NEXT I
650 ASSIGN# 1 TO *
660 CLEAR
```

```
 670 !              fine
 680 PRINT "FINE ELABORAZIONE
 690 PAGESIZE 16 !                  solo per HP-87 (che normalmente visualizza 16 righe)
 700 STOP
 980 !
 990 !              PRIMA PAGINA - indicazioni sui files
1000 CLEAR
1010 RESTORE 1500
1020 GOSUB 7000
1030 GOSUB 7200
1040 GOSUB 7500
1050 GOSUB 8000
1060 UJ=1
1070 GOSUB 7650 @ RS=UD$
1080 GOSUB 7650 @ FIS=UD$
1090 GOSUB 7650 @ FO$=UD$
1100 VH=1
1110 IF R$ <> "SI" THEN 1220
1120 !              lettura valori da disco
1130 ASSIGN# 1 TO FI$
1140 READ# 1 ; N
1150 FOR I=1 TO N
1160       READ# 1 ; X(I),Y(I) Z(I)
1170 NEXT I
1180 ASSIGN# 1 TO *
1190 VH=INT ((N+19)/20)+2
1200 I3=N
1210 !
1220 RETURN
1490 !
1500 DATA 3
1510 DATA 1,50,2,3,50,10 7 50 10
1520 DATA 3
1530 DATA 1,1,modifica di valori memorizzati?   (SI o NO)
1540 DATA 3 1.se SI :   in quale file?
1550 DATA 7,1,in quale file conservare i valori?
1980 !
1990 !              SECONDA PAGINA - numero di punti
2000 CLEAR
2010 RESTORE 2500
2020 GOSUB 7000
2030 GOSUB 7200
2040 UJ=1
2050 IF UH<= VH THEN UD= N @ GOSUB 7700 ELSE GOSUB 7750
2060 GOSUB 8000
2070 UJ=1
2080 GOSUB 7650 @ N=UD
2090 UZ=INT ((N+19)/20)+2
2100 IF VH<UH THEN VH=UH
2110 RETURN
2490 !
2500 DATA 1
2510 DATA 1,50,3
2520 DATA 1
```

```
2530 DATA 1,20 numero totale di punti
2980 !
2990 !              ALTRE PAGINE - coordinate dei punti
3000 CLEAR
3010 RESTORE 3500
3020 I1=(UH-3)*20+1
3030 I2=MIN (N,(UH-2)*20)
3040 UT=I2-I1+1
3050 GOSUB 7100
3060 GOSUB 7200
3070 GOSUB 7300
3080 UA=I1
3090 GOSUB 7400
3100 !              visualizzazione valori precedenti
3110 UJ=1
3120 FOR I=I1 TO I2
3130       IF I>I3 THEN 3180
3140             UD=X(I) @ GOSUB 7700
3150             UD=Y(I) @ GOSUB 7700
3160             UD=Z(I) @ GOSUB 7700
3170       GOTO 3220
3180             GOSUB 7750
3190             GOSUB 7750
3200             GOSUB 7750
3210       GOTO 3220
3220 NEXT I
3230 !              assegnazione nuovi valori
3240 GOSUB 8000
3250 UJ=1
3260 FOR I=I1 TO I2
3270       GOSUB 7650 @ X(I)=UD
3280       GOSUB 7650 @ Y(I)=UD
3290       GOSUB 7650 @ Z(I)=UD
3300 NEXT I
3310 IF VH<UH THEN VH=UH
3320 IF I3<I2 THEN I3=I2
3330 RETURN
3490 !
3500 DATA 3,3
3510 DATA 20,8,40,8,60,8
3520 DATA 4
3530 DATA 1,1,punto,1,20,X,1,40,Y,1,60,Z
3540 DATA 3,29,m                    m                    m
3550 DATA 3,1
5980 !
5985 !              CONTROLLO DEL VALORE DEI CAMPI
5990 !
6000 IF UH=1 THEN 6100
6005 IF UH=2 THEN 6200
6010 GOTO 6300
6095 !
6100 IF UJ<> 1 THEN 6115
6105       GOSUB 7600
6110       IF UD$ <> "SI" AND UD$ <> "NO  THEN GOSUB 6500
```

```
6115 GOTO 6300
6190 !
6200 GOSUB 7600
6205 IF UD<= 0 OR UD>100 THEN GOSUB 6550
6210 GOTO 6300
6295 !
6300 RETURN
6495 !
6500 AWRITE 23,1 "Devi rispondere SI o NO
6505 GOTO 6555
6545 !
6550 AWRITE 23,1 "Valore non ammissibile"
6555 BEEP
6560 WAIT 1000
6565 AWRITE 23,1,
6570 UF=0
6575 RETURN
```

6.11.4 - Codifica in GWBASIC

```
 10            LETTURA COORDINATE NELLO SPAZIO
 20
 80            PROGRAMMA PRINCIPALE
 90'
100 OPTION BASE 1
110 KEY OFF
120 DIM X(100),Y(100),Z(100)
190'
200 GOSUB 7800
210 VH=0
220 I3=0
230 UZ=3
240 UH=1
390            scelta tra le pagine
400 IF UH=1 THEN 450
410 IF UH=2 THEN 480
420 GOTO 510
440'
450 GOSUB 1000
460 GOTO 540
470'
480 GOSUB 2000
490 GOTO 540
500'
510 GOSUB 3000
520 GOTO 540
530'
540 UH=UH+UU
550 IF UH=0 THEN UH=1
560 IF UH<= UZ THEN 400
590'           memorizzazione valori su disco
600 OPEN "O" #1,FO$
610 WRITE #1, N
```

```
 620 FOR I=1 TO N
 630       WRITE #1, X(I),Y(I) Z(I)
 640 NEXT I
 650 CLOSE #1
 660 CLS
 670               fine
 680 PRINT "FINE ELABORAZIONE
 690 KEY ON
 700 STOP
 980 '
 990               PRIMA PAGINA - indicazioni sui files
1000 CLS
1010 RESTORE 1500
1020 GOSUB 7000
1030 GOSUB 7200
1040 GOSUB 7500
1050 GOSUB 8000
1060 UJ=1
1070 GOSUB 7650 : R$=UD$
1080 GOSUB 7650 : FI$=UD$
1090 GOSUB 7650 : FO$=UD$
1100 VH=1
1110 IF R$ <>  SI ' THEN 1220
1120               lettura valori da disco
1130 OPEN "I",#1,FI$
1140 INPUT #1, N
1150 FOR I=1 TO N
1160       INPUT #1, X(I),Y(I),Z(I)
1170 NEXT I
1180 CLOSE #1
1190 VH=INT ((N+19)/20)+2
1200 I3=N
1210 '
1220 RETURN
1490 '
1500 DATA 3
1510 DATA 2,51,2,4,51,10,8,51,10
1520 DATA 3
1530 DATA 2,2,modifica di valori memorizzati ? (SI o NO)
1540 DATA 4 2 "se SI : in quale file ?"
1550 DATA 8,2 in quale file conservare i valori ?
1980 '
1990               SECONDA PAGINA - numero di punti
2000 CLS
2010 RESTORE 2500
2020 GOSUB 7000
2030 GOSUB 7200
2040 GOSUB 7500
2050 UJ=1
2060 IF UH<= VH THEN UD=N : GOSUB 7700
2070 GOSUB 8000
2080 UJ=1
2090 GOSUB 7650 : N=UD
2100 UZ=INT ((N+19)/20)+2
```

```
2110 IF VH<UH THEN VH=UH
2120 RETURN
2490 '
2500 DATA 1
2510 DATA 2,51 3
2520 DATA 1
2530 DATA 2,21,numero totale di punti
2980 '
2990            ALTRE PAGINE - coordinate dei punti
3000 CLS
3010 RESTORE 3500
3020 I1=(UH-3)*20+1
3030 I2=(UH-2)*20 : IF I2>N THEN I2=N
3040 UT=I2-I1+1
3050 GOSUB 7100
3060 GOSUB 7200
3070 GOSUB 7300
3080 UA=I1
3090 GOSUB 7400
3100 GOSUB 7500
3110 '          visualizzazione valori precedenti
3120 UJ=1
3130 FOR I=I1 TO I2
3140     IF I>I3 THEN 3180
3150         UD=X(I) : GOSUB 7700
3160         UD=Y(I) : GOSUB 7700
3170         UD=Z(I) : GOSUB 7700
3180 NEXT I
3190 '          assegnazione nuovi valori
3200 GOSUB 8000
3210 UJ=1
3220 FOR I=I1 TO I2
3230     GOSUB 7650 : X(I)=UD
3240     GOSUB 7650 : Y(I)=UD
3250     GOSUB 7650 : Z(I)=UD
3260 NEXT I
3270 IF VH<UH THEN VH=UH
3280 IF I3<I2 THEN I3=I2
3290 RETURN
3490 '
3500 DATA 3,4
3510 DATA 21,8,41,8,61,8
3520 DATA 4
3530 DATA 2,2,nodo,2,21,X,2,41,Y,2,61,Z
3540 DATA 4,30,m                m                m
3550 DATA 4,2
5980 '
5985            CONTROLLO DEL VALORE DEI CAMPI
5990
6000 IF UH=1 THEN 6100
6005 IF UH=2 THEN 6200
6010 GOTO 6300
6095 '
6100 IF UJ<> 1 THEN 6115
```

```
6105        GOSUB 7600
6110        IF UD$ <> "SI" AND UD$ <> "NO" THEN GOSUB 6500
6115 GOTO 6300
6190 '
6200 GOSUB 7600
6205 IF UD<= 0 OR UD>100 THEN GOSUB 6550
6210 GOTO 6300
6295 '
6300 RETURN
6495 '
6500 LOCATE 24,2
6505 PRINT "Devi rispondere SI o NO";
6510 GOTO 6560
6545
6550 LOCATE 24,2
6555 PRINT "Valore non ammissibile " ;
6560 BEEP
6565 FOR UV=1 TO 1500: NEXT UV
6570 LOCATE 24,2
6575 PRINT "
6580 UF=0
6585 RETURN
```

CAPITOLO SETTIMO

FONDAMENTI DI GRAFICA

7.1 - Introduzione

Le applicazioni grafiche dei calcolatori stanno diventando sempre più comuni in molteplici settori, grazie al perfezionamento dell'hardware e in particolare al progresso nel campo dei circuiti elettronici, che ha reso disponibile a costi moderati le grandi capacità di memoria necessarie per l'immagazzinamento di informazioni grafiche. Anche nel campo dell'ingegneria civile non è troppo lontano il momento in cui la classica figura del disegnatore, chino sul suo tavolo da disegno e armato di pennini a china e lamette per cancellare, verrà sostituita da quella di un tecnico che ottiene gli stessi risultati attraverso la tastiera di un computer.

Date le molteplici possibilità di impiego, un approccio generale e onnicomprensivo alla grafica richiederebbe un intero volume. D'altro canto molte applicazioni, quali la generazione di figure in movimento sullo schermo o l'uso del colore, sono lontane dalle necessità dell'ingegnere civile. Si è quindi preferito effettuare una stretta selezione e rivolgere l'attenzione esclusivamente al problema del tracciamento (sullo schermo, ma in maniera analoga anche su un plotter) di grafici, diagrammi o in generale insiemi di linee. Si è inoltre cercato di individuare il minimo numero di istruzioni necessarie a ciò, tra la molteplicità ridondante che BASIC HP e GWBASIC presentano

Per questo tipo di applicazioni è possibile un approccio logico, quasi del tutto svincolato dalle caratteristiche fisiche del personal computer su cui si lavora. Nell'affrontare la grafica su video è però necessaria una premessa per illustrare le caratteristiche strutturali che la rendono possibile.

La maggior parte degli schermi si basa sulla tecnologia del tubo a raggi catodici, analoga a quella dei televisori. Nel tubo a raggi catodici un fascio di elettroni colpisce uno schermo rivestito di fosfori, che emette luce con una intensità dipendente dall'energia cinetica degli elettroni. Poichè l'emissione di luce si affievolisce rapidamente, l'intera immagine viene ridisegnata parecchie volte ogni secondo, basandosi su una rappresentazione di essa immagazzinata in un'area di memoria a ciò dedicata. Il fascio di elettroni percorre lo schermo secondo una modalità detta "a scansione sistematica orizzontale", cioè tracciandovi un insieme di righe, ciascuna delle quali costituita da una molteplicità di puntini che possono avere una luminosità minore o maggiore (da scuro a chiaro, con eventuali tonalità intermedie).

Nel capitolo precedente si sono già descritte le caratteristiche di uno schermo alfanumerico, cioè destinato alla visualizzazione di caratteri. Nello schermo dell'HP-86/87 l'immagine è costituita da 640×240 punti (il primo numero si riferisce alla quantità misurata in orizzontale, cioè al numero di punti per riga) Ogni carattere è composto mediante una matrice di 8×15 punti se le righe di testo visualizzate sono 16, o 8×10 se le righe sono 24 (80 caratteri per 8 punti corrispondono a un totale di 640 punti, così come 16 righe per 15 punti, o 24 per 10, corrispondono a un totale di 240 punti). Nell'M24 lo schermo è costituito da 640×400 punti e ciascun carattere da 8×16 punti. La quantità di memoria necessaria per conservare le informazioni alfanumeriche visualizzate è molto minore del numero di punti necessari per rappresentarle. In base a quanto detto, nel BASIC HP basta infatti un byte ogni 120 o 80 punti, nel GWBASIC due bytes ogni 128 punti.

Quando si opera in modalità grafica bisogna individuare la luminosità o il colore del singolo punto. Più propriamente si denomina "pixel" (*picture element*, cioè "elemento di immagine") l'unità minima il cui stato è conservato nella memoria del calcolatore. Essa costituisce un "punto grafico" individualmente controllabile dal software, che viene visualizzato o con un singolo punto o con un insieme di più punti contigui La capacità di memoria necessaria per immagazzinare queste informazioni è notevolmente superiore rispetto a quella richiesta dalla modalità alfanumerica Per ciascun pixel di uno schermo monocromatico occorre almeno un bit per indicare se esso è acceso o spento. Per ogni pixel di uno schermo policromatico occorrono invece più bit per indicarne il colore.

L'area di memoria riservata alla grafica ha dimensioni limitate (spesso 16 o 32 kbytes) e non sempre consente di sfruttare al massimo la potenzialità dello schermo. Nel BASIC HP le istruzioni "**GRAPH**" e "**GRAPHALL**" consentono di definire le modalità omonime, che utilizzano rispettivamente 400×240 e 544×240 punti. Nell'M24 è possibile avere 640×400 pixel solo in bianco e nero. Per ottenere immagini a colori occorre accorpare più punti. L'effetto visivo che così si ottiene è meno preciso, meno nitido o, come si suol dire, ha una minore risoluzione rispetto al caso precedente. L'istruzione "**SCREEN** *n*" consente di definire la modalità alfanumerica (quando n è uguale a zero) o grafica (n maggiore di zero) Più precisamente, il valore n=1 definisce la modalità grafica di minore risoluzione, che prevede 320×200 pixel, ciascuno dei quali corrisponde a 4 puntini dello schermo; n=2 individua una risoluzione maggiore (640×200 pixel), mentre la risoluzione massima (640×400 pixel), ammessa solo da alcuni calcolatori, corrisponde ai valori di n maggiori o uguali a 3.

La qualità di un'immagine dipende dalla risoluzione, ma anche dalla dimensione del video, poichè a parità di pixel essa appare tanto meno nitida quanto più grande è lo schermo Nell'HP-87 in modalità GRAPH si hanno 400×240 pixel per una superficie di 12.2×7.5 cm, cioè una densità di 1024 pixel per centimetro quadro. Nell'HP-86, che ha un monitor di 12 pollici, lo stesso numero di pixel occupa una superficie di 14.4×15 cm, con

una densità nettamente minore (445 pixel/cm^2) e quindi una immagine peggiore, pur essendo uguale la quantità di informazioni usate per rappresentarla. L'M24 con la massima risoluzione raggiunge un risultato intermedio: 640×400 pixel in una superficie di 21.2×16 cm, cioè circa 755 pixel/cm^2. Nell'IDE Best la massima risoluzione è invece di 640×200 pixel in una superficie di 19×12.5 cm cioè circa 540 pixel/cm^2

Pixel	punto grafico cioè unità minima individualmente controllabile dal software
BASIC HP	
GRAPH	modalità grafica normale (400×240 pixel)
GRAPHALL	modalità grafica espansa (544×240 pixel)
GWBASIC	
SCREEN 0	modalità alfanumerica (testo)
SCREEN 1	modalità grafica a bassa risoluzione (320×200 pixel)
SCREEN 2	modalità grafica a media risoluzione (640×200 pixel)
SCREEN n (n>2)	modalità grafica ad alta risoluzione (640×400 pixel)

7.2 - Sistema fisico e sistema logico di riferimento.

Ciascun video grafico ha dei limiti fisici, che delimitano l'area entro cui è possibile la creazione di immagini. In essa ciascun punto è individuato mediante le sue coordinate rispetto ad un sistema si riferimento (fig 50) Nell'HP-87 l'origine di questo coincide con l'estremo inferiore sinistro dello schermo; l'asse x è orizzontale ed orientato verso destra, l'asse y verticale orientato verso l'alto. Come unità di misura si usa il millimetro, oppure una unità grafica pari ad un centesimo del lato minore del video Nel GWBASIC l'origine è invece posta nell'estremo superiore sinistro e gli assi x ed y sono orientati rispettivamente verso destra e verso il basso. L'unità di misura è in questo caso il pixel

In numerosi casi è però preferibile confinare il disegno in un'area ristretta dello schermo, definendo dei limiti logici per la graficizzazione. Quest'area è usualmente indicata col termine "finestra" La finestra di partenza, cui inizialmente il calcolatore fa riferimento, è ovviamente coincidente con l'intero schermo fisico Nel BASIC HP questi limiti logici vengono definiti mediante l'istruzione "**LIMIT** *xmin, xmax, ymin, ymax*". In essa *xmin, ymin* e *xmax, ymax* rappresentano le coordinate dell'estremo inferiore sinistro e superiore destro della finestra, misurate in millimetri. La corrispondente istruzione del GWBASIC è "**VIEW** *(xmin,ymin)−(xmax,ymax)*", con le coordinate espresse in pixel e riferite agli estremi superiore sinistro e inferiore destro.

Anche nell'ambito di una finestra i punti vengono individuati mediante le loro coordinate, espresse in unità grafiche o in pixel. È però possibile svincolarsi da queste grandezze e utilizzare un sistema di riferimento logi-

co, definito dall'utente. Nel BASIC HP e nel GWBASIC le istruzioni, diverse nel nome ma non nella sostanza, sono rispettivamente "**SCALE** *xmin, xmax, ymin, ymax*" e "**WINDOW** (*xmin, ymin*)–(*xmax, ymax*)". Esse indicano i valori minimo e massimo, assunti dalle coordinate logiche agli estremi della finestra. Se *xmin*<*xmax* e *ymin*<*ymax* gli assi x ed y sono orientati rispettivamente verso destra e verso l'alto, come usualmente avviene in un riferimento cartesiano. Nel seguito si utilizzerà sempre tale tipo di riferimento, in modo da prescindere dalle caratteristiche fisiche dello schermo

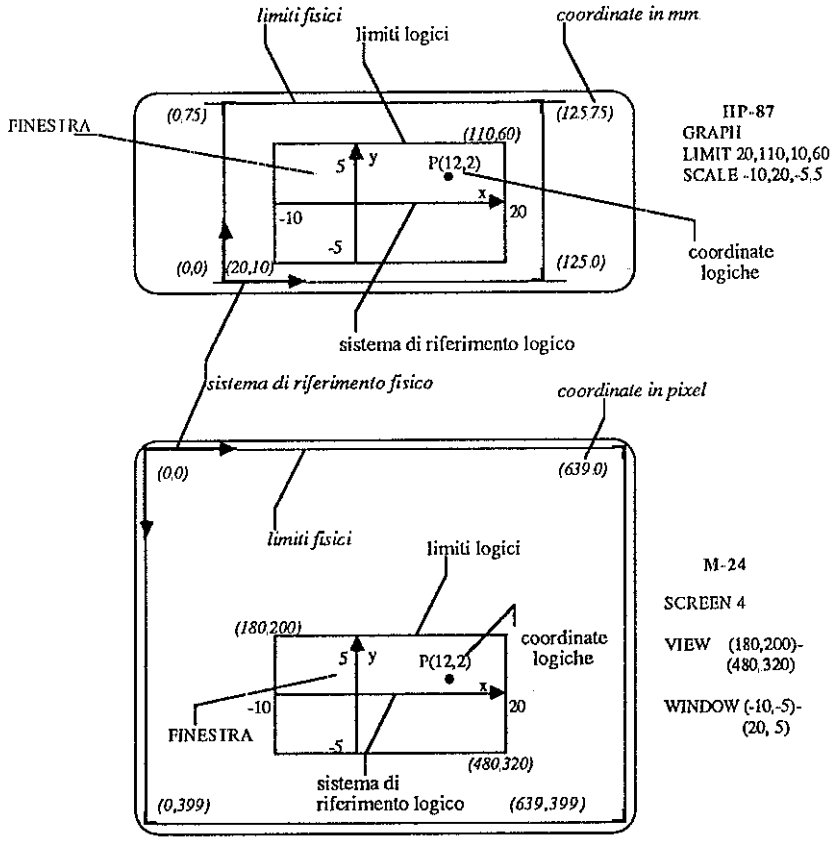


Fig. 50

Limiti fisici	delimitano l'area entro cui è fisicamente possibile la creazione di immagini
Limiti logici	individuano una finestra nello schermo, cioè un'area ristretta entro cui si vuole racchiudere l'immagine
Sistema di riferimento fisico	utilizza come unità di misura grandezze legate alla struttura dello schermo (millimetri, unità grafiche, pixel)
Sistema di riferimento logico	ha origine ed unità di misura definite dall'utente, in base alle sue esigenze e indipendentemente dalla struttura fisica dello schermo

Individuazione di una finestra

BASIC HP	LIMIT xmin,xmax,ymin,ymax
GWBASIC	VIEW (xmin,ymin)–(xmax,ymax)

Definizione di coordinate logiche

BASIC HP	SCALE xmin,xmax,ymin,ymax
GWBASIC	WINDOW (xmin,ymin)–(xmax,ymax)

7.3 - Tracciamento di linee

Analogamente a quanto avviene in modalità alfanumerica, anche in modalità grafica esiste un puntatore che individua l'ultimo punto cui si è fatto riferimento con istruzioni grafiche. Nel BASIC HP si può spostare il puntatore in una posizione di coordinate *x* ed *y*, senza tracciare linee, con l'istruzione "**MOVE** *x,y*". Per tracciare un segmento dalla posizione corrente fino ad un punto *x,y* si utilizza invece l'istruzione "**DRAW** *x,y*". Di conseguenza, per tracciare una linea isolata dal punto *x1,y1* al punto *x2,y2* occorre utilizzare in sequenza le due istruzioni "**MOVE** *x1,y1*" e "**DRAW** *x2,y2*". Quando si vuole tracciare un insieme di segmenti consecutivi, ciascuno che inizia alla fine del precedente, è invece sufficiente utilizzare una istruzione MOVE per posizionarsi all'inizio della spezzata ed una sequenza di istruzioni DRAW per disegnarla. Ad esempio, per tracciare un triangolo rettangolo con un vertice nell'origine e cateti di lunghezza 7.5 e 4, si scriverà:

```
MOVE 0,0
DRAW 7.5,0
DRAW 0,4
DRAW 0,0
```

Nel GWBASIC non è necessaria una istruzione corrispondente alla MOVE, perchè il tracciamento di linee è effettuato con l'istruzione "**LINE** (*x1,y1*)–(*x2,y2*)" che racchiude le informazioni su entrambi gli estremi del segmento. Le coordinate del primo punto possono essere omesse, se esso è quello correntemente individuato dal puntatore, ottenendo

la forma abbreviata "LINE $-(x2,y2)$", equivalente alla DRAW del BASIC HP Per ottenere lo stesso triangolo innanzi descritto si useranno i comandi:

```
LINE (0,0)-(7 5,0)
LINE -(0,4)
LINE -(0,0)
```

Lo spostamento del puntatore ed il tracciamento di linee possono essere effettuati utilizzando anche coordinate relative, misurate rispetto all'ultimo punto individuato. In BASIC HP si utilizzano a tal fine le istruzioni "**IMOVE** *dx,dy*" e "**IDRAW** *dx,dy*", nelle quali *dx* e *dy* indicano di quanto occorre incrementare (la lettera iniziale "I" stà proprio per "incremento") le coordinate del punto corrente per ottenere il punto di arrivo. Ad esempio, i comandi:

```
MOVE 7.5 4
IDRAW 2,-3
```

impongono il tracciamento di una linea dal punto (7.5,4) al punto (9.5,1).

Analogo effetto si ottiene in GWBASIC inserendo la parola "**STEP**" nell'istruzione LINE. STEP può essere riferito al primo punto, al secondo o ad entrambi. Così, i comandi:

```
LINE (7.5,4)-STEP(2,-3)
LINE STEP(1,4)-(5 0)
LINE STEP(-1 -1)-STEP(3,4)
```

provocano, se eseguiti consecutivamente, il tracciamento di tre linee: la prima dal punto (7.5,4) al punto (9 5,1); la seconda da (10.5,5) a (5,0); la terza da (4,1) a (7,5).

Normalmente nel tracciare un segmento tutti i pixel dello schermo che si trovano lungo tale allineamento vengono accesi, visualizzando così l'immagine della linea. Sia nel BASIC HP che nel GWBASIC sono però possibili altre tre alternative. I punti possono essere tutti spenti, cioè si può tracciare un segmento dello stesso colore dello sfondo (e in tal modo è possibile cancellare linee già disegnate). Oppure si può cambiare lo stato luminoso dei punti, accendendo quelli spenti e viceversa O, infine, variare la posizione del puntatore senza modificare lo stato dei punti della linea, rendendo i comandi DRAW e LINE equivalenti a MOVE. Nel BASIC HP ciò è possibile mediante l'istruzione "**PEN** *colore*" che indica che tutte le successive linee devono essere tracciate con la modalità *colore*. Nel GWBASIC l'indicazione è invece riferita alla singola linea ed è ottenuta aggiungendo un parametro al comando LINE: "**LINE** $(x1,y1)-(x2,y2)$,*colore*" Di seguito si riportano i valori possibili per il parametro *colore*

BASIC HP:	GWBASIC:	significato:
1	1	accende i punti
−1	0	spegne i punti
−2	3	inverte lo stato
0	2	non varia lo stato

Utilizzando in GWBASIC schermi policromatici, il valore del parametro *colore* indica invece il colore col quale deve essere tracciata la linea.

Un'altra possibilità che entrambi i linguaggi offrono è quella di tracciare linee non continue (tratteggiate, punteggiate, a tratto e punto). Nel BASIC HP esistono otto tipi di linee predefinite, e l'istruzione "**LINE TYPE** *k*" indica quale deve essere utilizzata. Ad esempio, "LINE TYPE 3" impone che le linee vengano tracciate con puntini separati. Ogni prescrizione vale fin quando non viene definito un nuovo tipo di linea.

Nel GWBASIC l'indicazione è invece riferita sempre ad un singolo segmento ed è effettuata aggiungendo un ulteriore parametro all'istruzione LINE: "**LINE** *(x1,y1)−(x2,y2),,,stile*". In questo caso occorre definire una maschera di 16 bit, per ciascuno dei quali il valore 0 corrisponde ad un punto spento ed il valore 1 ad un punto acceso (o, più propriamente, un punto per il quale si interviene sullo stato luminoso in base al parametro *colore* già citato). Ad esempio, una linea punteggiata è definita dalla maschera *1010101010101010*, cioè dal valore esadecimale *AAAA* (il sistema di numerazione esadecimale, ovvero a base 16, ha come cifre *0 1 2 3 4 5 6 7 8 9 A B C D E F*, e pertanto *A* corrisponde al decimale *10* cioè al binario *1010*). Essa viene quindi tracciata con il comando "LINE *(x1,y1)−(x2,y2),,,&HAAAA*", in cui il prefisso *&H* indica che i successivi caratteri devono essere considerati cifre esadecimali.

BASIC HP

MOVE x,y
: sposta il puntatore fino al punto (x,y) senza tracciare linee

DRAW x,y
: traccia una linea fino al punto (x,y)

IMOVE dx,dy
: sposta il puntatore di una quantità dx,dy senza tracciare linee

IDRAW dx,dy
: traccia una linea fino al punto ottenuto incrementando le coordinate correnti di una quantità dx,dy

PEN colore
: indica come modificare lo stato luminoso dei punti nel tracciare una linea

LINE TYPE k
: indica che tipo di linea tracciare.

GWBASIC

LINE (x1,y1)-(x2,y2)
traccia una linea dal punto (x1,y1) al punto (x2,y2);
se il primo punto non è indicato, si intende riferirsi al punto corrente

STEP(dx,dy) al posto di (x1,y1) o (x2,y2) nell'istruzione LINE
indica che il punto deve essere ottenuto incrementando le coordinate del punto corrente della quantità dx,dy

LINE (x1,y1),(x2,y2),colore,,stile
il parametro *colore* indica come modificare lo stato luminoso dei punti nel tracciare la linea;
il parametro *stile* definisce il tipo di linea mediante un insieme di 16 bit

7.4 - Esempio: disegno di un istogramma

7.4.1 - Descrizione del programma e variabili utilizzate

L'istogramma è un particolare tipo di diagramma, in cui i valori della funzione, relativi a successivi intervalli della variabile, sono rappresentati mediante segmenti o rettangoli. Si suppone di voler mostrare graficamente il numero di allievi iscritti ad un corso in un insieme di anni consecutivi e quanti di essi abbiano sostenuto l'esame nei dodici mesi successivi alla fine delle lezioni. Il valore relativo a ciascun anno sarà rappresentato con un rettangolo, la cui altezza corrisponde, in una opportuna scala, al numero di iscritti. Il quantitativo di esami svolti può essere mostrato tratteggiando la parte inferiore fino all'altezza ad esso corrispondente.

Nel programma sono state utilizzate le seguenti variabili:

- N numero totale di anni;
- J numero d'ordine dell'anno generico;
- I( ) array che contiene il numero di iscritti al corso per ciascun anno;
- E( ) array che contiene il numero di esami fatti dagli studenti nel periodo definito per ciascun anno;
- M massimo numero di iscritti in un anno;
- J1,J2 variabili utilizzate nel tratteggiare la parte inferiore di ogni rettangolo

Il lavoro da svolgere per ottenere il risultato voluto può essere distinto in tre fasi: lettura dei dati, definizione della scala del disegno, graficizzazione dell'istogramma

La prima di esse non richiede alcun chiarimento. Per la seconda, occorre definire in che modo devono essere rappresentati i valori. Gli anni possono essere riportati lungo l'asse delle x, riservando uno spazio unitario a ciascun rettangolo. Le ascisse andranno quindi dal valore 0 al valo-

re N (fig. 51). Sulle ordinate è riportato il numero degli studenti. Il valore minimo è quindi 0, mentre il massimo deve essere uguale al più alto tra i valori da rappresentare. Per la definizione della scala del disegno occorre quindi preliminarmente passare in rassegna tutti i valori del numero di iscritti, per determinarne il maggiore

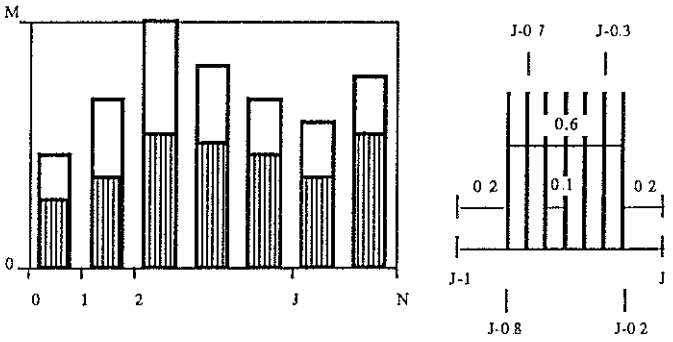


Fig. 51

La fase di graficizzazione richiede la ripetizione di una serie di operazioni, uguali per tutti gli anni. Si disegna prima il rettangolo corrispondente al numero di iscritti; la dimensione di base è stata posta pari a 0.6, per mantenere i rettangoli reciprocamente distanziati. Si passa poi alla rappresentazione del numero degli esami, delimitando il rettangolo superiormente in corrispondenza a tale valore e realizzando il tratteggio mediante linee verticali, che per il generico rettangolo J corrispondono ad ascisse variabili da J−0 7 a J−0.3 con passo 0 1. Il modo più immediato per ottenere queste linee è l'uso di un ciclo, il cui indice J1 vari tra tali estremi col passo indicato. Se si opera in questo modo in GWBASIC, si può notare ancora una volta l'errore già segnalato nel paragrafo 4.9: una linea (l'ultima del primo rettangolo) non viene disegnata. Come già evidenziato, la causa di ciò è il tipo di codifica utilizzato in GWBASIC per i numeri decimali, che comporta nelle operazioni approssimazioni molto piccole, in genere trascurabili, ma che possono diventare determinanti nel controllo di un ciclo. Per ovviare a ciò si è utilizzato come contatore un numero intero, che varia da −7 a −3, calcolando in funzione di esso il valore della ascissa.

Nei paragrafi seguenti è riportata la codifica del programma in BASIC HP e in GWBASIC. Nel primo caso, il passaggio alla modalità grafica è auto-

matico quando si incontrano in esecuzione istruzioni di tale genere Nel secondo è invece necessario definire preliminarmente la modalità con l'istruzione SCREEN.

7.4.2 - Codifica in BASIC HP.

```
100 !              DISEGNO DI UN ISTOGRAMMA
110 !
120 DIM I(20) E(20)
130 !              legge i dati
140 READ N
150 FOR J=1 TO N
160        READ I(J) E(J)
170 NEXT J
180 !              individua il valore massimo
190 M=0
200 FOR J=1 TO N
210        IF I(J)>M THEN M=I(J)
220 NEXT J
230 !              definisce il sistema logico di coordinate
240 SCALE 0 N 0,M
250 !              pulisce lo schermo
260 GCLEAR
270 !              inizio ciclo di graficizzazione
280 FOR J=1 TO N
290 !              rappresenta il numero degli iscritti
300        MOVE J- 8 0
310        DRAW J- 8,I(J)
320        DRAW J- 2,I(J)
330        DRAW J- 2,0
340        DRAW J- 8,0
350 !              rappresenta il numero degli esami
360        MOVE J- 8,E(J)
370        DRAW J- 2,E(J)
380        FOR J1=J-.7 TO J- 3 STEP -1
390             MOVE J1,0
400             DRAW J1 E(J)
410        NEXT J1
420 !              fine disegno J
430 NEXT J
440 !              fine
450 STOP
460 !              DATI
470 DATA 8
480 DATA 110,63,135,71,144,68,112,55 109,61,132,60,140,75 108,72
```

7.4.3 - Codifica in GWBASIC.

```
100              DISEGNO DI UN ISTOGRAMMA
110 '
120 DIM I(20) E(20)
130              legge i dati
```

```
140 READ N
150 FOR J=1 TO N
160       READ I(J) E(J)
170 NEXT J
180               individua il valore massimo
190 M=0
200 FOR J=1 TO N
210       IF I(J)>M THEN M=I(J)
220 NEXT J
230               definisce il sistema logico di coordinate
240 WINDOW (0 0)-(N,M)
250               pulisce lo schermo
260 CLS
270               inizio ciclo di graficizzazione
280 FOR J=1 TO N
290               rappresenta il numero degli iscritti
310       LINE (J- 8,0)-(J- 8,I( J))
320       LINE -(J- 2,I(J))
330       LINE -(J- 2 0)
340       LINE -(J- 8,0)
350               rappresenta il numero degli esami
370       LINE (J- 8,E(J))-(J- 2 E(J))
380       FOR J2=-7 TO -3
390           J1=J+J2*.1
400           LINE (J1,0)-(J1 E(J))
410       NEXT J2
420               fine disegno J
430 NEXT J
440               fine
450 STOP
460               DATI
470 DATA 8
480 DATA 110,63,135,71,144 68,112,55,109,61,132,60 140,75,108,72
```

7.5 - Esempio: diagramma di una funzione

7.5.1 - Descrizione del problema e variabili utilizzate

Un esempio ripetutamente utilizzato nei capitoli 3 e 4 è stato quello della tabellazione di una funzione. I valori allora ottenuti possono ora essere mostrati sotto forma di diagramma.

In tutti i problemi di questo genere sono possibili due impostazioni diverse. La prima consiste nel diagrammare i valori man mano che essi vengono calcolati. La scala del disegno deve in questo caso essere definita prima del calcolo, ed occorre pertanto conoscere almeno l'ordine di grandezza dei valori che si determineranno. La seconda consiste invece nell'effettuare prima tutte le elaborazioni numeriche, memorizzandone i risultati in un array, e nel passare solo in un momento successivo alla loro graficizzazione. È in tal modo possibile delegare al calcolatore l'individuazione dei valori minimi e massimi e la scelta di una appropriata scala di

disegno. Si ha però lo svantaggio di un maggior impegno di memoria.

Nell'esempio in esame si è preferita la prima via, inserendo tra i dati di ingresso informazioni relative all'intervallo entro cui sono comprese le ascisse e le ordinate. Si sono utilizzate le seguenti variabili:

A,B	estremi dell'intervallo entro cui diagrammare la funzione;
D	passo di incremento;
X1,X2	estremi dell'intervallo delle ascisse;
Y1,Y2	estremi dell'intervallo delle ordinate;
X	ascissa corrente;
Y	valore di $f_{(x)}$

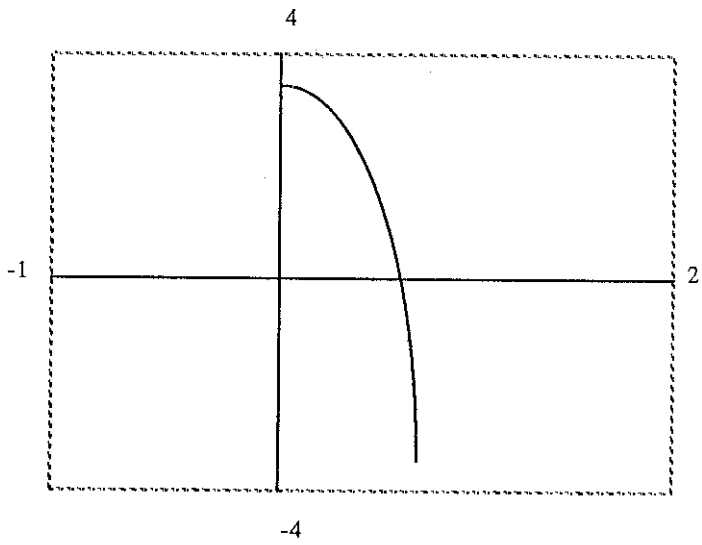


Fig 52

I blocchi operativi del programma sono individuati e descritti mediante frasi di commento. Per alcuni di essi (disegno di un riquadro, tracciamento degli assi coordinati) si potrebbero utilizzare direttamente istruzioni più potenti del BASIC HP e del GWBASIC, che per semplicità si è preferito non descrivere in questo testo.

Il ciclo di calcolo e disegno è analogo a quello utilizzato nella tabellazione. Si noti però che nel tracciare il diagramma è necessario posizionare il puntatore nel primo punto e poi tracciare linee di congiunzione con i punti successivi, creando quindi una differenziazione tra i punti che non esisteva in quel programma. Nella codifica per l'M24 il valore del secondo estremo, B, è stato posto pari a 1.00001 anzichè 1, per ovviare ai già citati problemi di controllo del ciclo. Il disegno ottenuto è mostrato in figura 52

7.5.2 - Codifica in BASIC HP.

```
 10 !                                    DIAGRAMMA DI UNA FUNZIONE
 20 !
 90 !            legge i dati
100 READ A,B,D
110 READ X1,X2,Y1,Y2
120 !            definisce le coordinate logiche di riferimento
130 SCALE X1,X2,Y1,Y2
140 !            riquadra lo schermo (o la finestra) con linea punteggiata
150 LINE TYPE 3
160 MOVE X1,Y1
170 DRAW X1,Y2
180 DRAW X2,Y2
190 DRAW X2 Y1
200 DRAW X1,Y1
210 !            disegna gli assi con linea continua
220 LINE TYPE 1
230 MOVE X1,0
240 DRAW X2,0
250 MOVE 0,Y1
260 DRAW 0 Y2
270 !            calcola le coordinate del primo punto
280 X=A
290 GOSUB 500
300 !            posiziona il puntatore sul primo punto
310 MOVE X,Y
320 !            passa al secondo punto
330 X=X+D
340 !            ciclo di calcolo e graficizzazione
350 IF X>B THEN 410
360 GOSUB 500
370 DRAW X Y
380 X=X+D
390 GOTO 350
400 !            fine ciclo
410 STOP
420 DATA 0,1,.1
430 DATA -1,2,-4,4
490 !            sottoprogramma che calcola f(x)
500 Y=-(7*X^3)+SQR (X^2+10)
510 RETURN
```

7.5.3 - Codifica in GWBASIC.

```
 10                                     DIAGRAMMA DI UNA FUNZIONE
 20 '
 90              legge i dati
100 READ A,B,D
110 READ X1,X2,Y1,Y2
120             definisce le coordinate logiche di riferimento
130 WINDOW (X1,Y1)-(X2,Y2)
```

```
140             riquadra lo schermo (o la finestra) con linea punteggiata
170 LINE (X1,Y1)-(X1,Y2),,,&H8888
180 LINE -(X2,Y2),,,&H8888
190 LINE -(X2 Y1)  &H8888
200 LINE -(X1,Y1),,,&H8888
210             disegna gli assi con linea continua
240 LINE (X1,0)-(X2,0)
260 LINE (0,Y1)-(0 Y2)
270 '           calcola le coordinate del primo punto
280 X=A
290 GOSUB 500
300             posiziona il puntatore sul primo punto
310 LINE -(X,Y),2
320             passa al secondo punto
330 X=X+D
340 '           ciclo di calcolo e graficizzazione
350 IF X>B THEN 410
360 GOSUB 500
370 LINE -(X Y)
380 X=X+D
390 GOTO 350
400             fine ciclo
410 STOP
420 DATA 0,1 00001, 1
430 DATA -1 2,-4,4
490 '           sottoprogramma che calcola f(x)
500 Y=-(7*X^3)+SQR (X^2+10)
510 RETURN
```

7.6 - Sistema di riferimento monometrico.

Esaminando la figura 52 si nota che un segmento unitario staccato sull'asse delle ascisse ha una lunghezza nettamente differente da un analogo segmento posto su quello delle ordinate. Ciò deriva ovviamente dal fatto che i valori minimi e massimi delle coordinate sono stati assegnati senza alcun riferimento alle reali dimensioni dell'area grafica su cui si è operato. L'effetto distorcente risulterebbe evidente se si disegnasse una figura regolare: un quadrato apparirebbe come un rettangolo, una circonferenza come un'ellisse. In molti casi è quindi necessario adottare un sistema di riferimento monometrico, cioè in cui le unità di misura per gli assi x ed y abbiano la stessa lunghezza.

Anzichè cercare di volta in volta gli opportuni valori per la definizione del sistema logico di coordinate, è preferibile realizzare un sottoprogramma che svolga tale compito in maniera generale. In esso si utilizzano le seguenti variabili:

variabili di ingresso:

X1,X2 valore minimo e massimo delle ascisse dei punti da rappresentare;
Y1,Y2 valore minimo e massimo delle ordinate dei punti da rappresentare;

variabili interne:

FX,FY dimensioni reali dell'area grafica, finestra o schermo in metri;
SC fattore di scala;
LX,LY dimensioni in scala dell'area grafica;
XC,YC coordinate del punto centrale della figura da rappresentare (pari alla media tra X1,X2 e tra Y1,Y2);

variabili di uscita:

X1,X2 valore minimo e massimo delle ascisse logiche dell'area grafica, nella scala determinata;
Y1,Y2 valore minimo e massimo delle ordinate logiche dell'area grafica, nella scala determinata

La scala di rappresentazione è in generale il rapporto tra la dimensione di un oggetto e quella della sua immagine (se rappresento il lato di una stanza, lungo 420 cm, con un segmento di 8 4 cm il disegno è in scala 1 a 50, dove il fattore di scala 50 è pari a 420/8 4). Per far rientrare esattamente tutte le ascisse nell'area grafica occorrerebbe un fattore di scala pari a (X2−X1)/FX; analogamente per le ordinate occorrerebbe (Y2−Y1)/FY. Per la rappresentazione dell'intero disegno si deve pertanto adottare un fattore pari al massimo tra tali due valori

Le dimensioni in scala dell'area grafica, LX ed LY, sono ottenute come prodotto delle dimensioni fisiche per il fattore di scala. È quindi facile definire i valori limite da assegnare alle coordinate. Per mantenere il disegno centrato occorre determinare le coordinate del punto centrale della figura da rappresentare, ed aggiungere e togliere ad esse la metà di LX ed LY. I valori di uscita così ottenuti vengono per brevità conservati nelle stesse variabili X1, X2, Y1, Y2.

Si riporta di seguito la codifica del sottoprogramma.

```
 990 REM       definizione scala per rappresentazione monometrica
1000 FX= 125
1010 FY= 075
1020 SC=(X2−X1)/FX
1030 IF SC<(Y2−Y1)/FY THEN SC=(Y2−Y1)/FY
1040 LX=FX*SC
1050 LY=FY*SC
1060 XC=(X1+X2)/2
1070 YC=(Y1+Y2)/2
1080 X1=XC−LX/2
1090 X2=XC+LX/2
1100 Y1=YC−LY/2
1110 Y2=YC+LY/2
1120 RETURN
```

In figura 53 si confrontano i diagrammi ottenuti per l'esempio precedente con e senza l'uso del sottoprogramma. Per evidenziare le differenze, lungo gli assi coordinati sono stati riportati dei trattini con passo unitario.

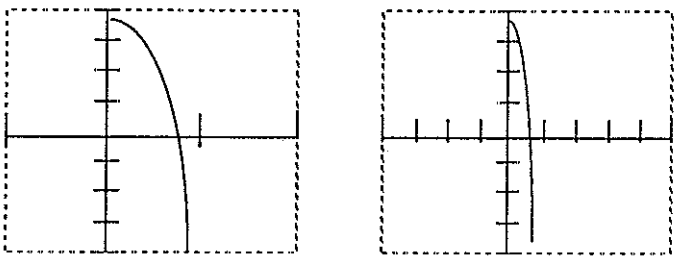


Fig 53

Le dimensioni indicate nelle linee 1000 e 1010 si riferiscono allo schermo dell'HP-87 in modalità GRAPH; per altri calcolatori o differenti modalità si possono utilizzare i valori indicati alla fine del paragrafo 7.1 Se l'area grafica è solo parte dell'intero video, esse andranno ridotte in proporzione.

Si noti che, qualora si voglia effettuare una rappresentazione grafica in una scala prefissata (ad esempio 1 a 200), si può far ricorso allo stesso sottoprogramma purchè si sostituiscano le linee 1020-1030 con una assegnazione di valore a SC (ad esempio con l'istruzione "1020 SC=200").

7.7 - Esempio: schema di un telaio piano.

7.7.1 - Descrizione del problema e variabili utilizzate.

Un problema delicato ed oneroso che si incontra in genere nell'uso di programmi di calcolo strutturale è quello del controllo dei dati. La grafica può venire incontro all'utente, fornendogli un mezzo di esame più immediato. Ad esempio, nella risoluzione di telai generici, cioè con aste comunque disposte nello spazio o nel piano, la visualizzazione dello schema geometrico può evidenziare errori grossolani nella definizione delle coordinate dei nodi o degli estremi delle aste. Il caso del telaio spaziale presenta una certa complessità, perchè occorre passare da un'immagine tridimensionale alla sua proiezione sullo schermo piano, sfruttando formule di trasformazione dipendenti dal tipo di rappresentazione (assonometrica o prospettica). Il modo di procedere è però del tutto simile a quello richiesto per un telaio piano, che si mostra esemplificativamente in questo paragrafo.

In un programma per la visualizzazione di schemi intelaiati piani si possono individuare, analogamente agli altri programmi finora descritti, tre fasi: lettura dei dati, definizione della scala, rappresentazione grafica

In questo caso i dati sono costituiti dal numero e dalle coordinate dei nodi, dal numero di aste e dai loro estremi. Nell'esempio i loro valori sono riportati in linee di DATA; più in generale potranno essere letti mediante maschere o dalla memoria di massa, inserendo questo programma come routine in un complesso più vasto di programmi strutturali

La seconda fase richiede innanzitutto la determinazione dei valori limite delle coordinate, mediante un ciclo che passa in rassegna tutti i nodi. Questi valori vengono poi utilizzati dal sottoprogramma descritto nel paragrafo precedente, che definisce la scala più opportuna per una rappresentazione monometrica.

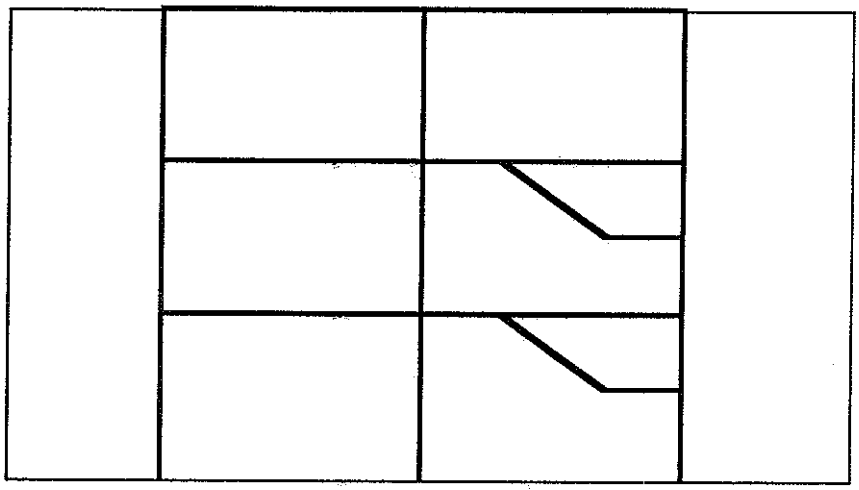


Fig 54

Nella terza fase, infine, vengono disegnate ciclicamente tutte le aste della struttura, tracciando una linea che congiunge il primo estremo, E1(J) per la generica asta J, col secondo estremo, E2(J). In figura 54 è mostrato il risultato ottenuto per un telaio a tre piani dotato di travi a ginocchio.

Nel programma si sono utilizzate le seguenti variabili:

N numero di nodi;
I numero d'ordine del nodo generico;
X( ) array che contiene le ascisse dei nodi;
Y( ) array che contiene le ordinate dei nodi;

M numero di aste;
J numero d'ordine dell'asta generica;
E1( ) array che contiene il numero d'ordine del primo estremo di ogni asta;
E2( ) array che contiene il numero d'ordine del secondo estremo di ogni asta;
X1,X2 valori limite di X;
Y1,Y2 valori limite di Y

7.7.2 - Codifica in BASIC HP.

```
 10!             SCHEMA GEOMETRICO DI UN TELAIO PIANO
 20!
 30 DIM X(100),Y(100),E1(100),E2(100)
 90!             legge i dati
100 READ N
110 FOR I=1 TO N
120       READ X(I),Y(I)
130 NEXT I
140 READ M
150 FOR J=1 TO M
160       READ E1(J),E2(J)
170 NEXT J
180!            determina i valori limite di X ed Y
190 X1=X(1)
200 X2=X(1)
210 Y1=Y(1)
220 Y2=Y(1)
230 FOR I=2 TO N
240       X1=MIN (X1,X(I))
250       X2=MAX (X2,X(I))
260       Y1=MIN (Y1,Y(I))
270       Y2=MAX (Y2,Y(I))
280 NEXT I
290!            individua la scala del disegno
300 GOSUB 1000
310 SCALE X1,X2,Y1,Y2
320!            ciclo di graficizzazione
330 FOR J=1 TO M
340       MOVE X(E1(J)),Y(E1(J))
350       DRAW X(E2(J)),Y(E2(J))
360 NEXT J
370!            fine
380 STOP
390!            dati
400 DATA 18
410 DATA 0,0,5,0,11 0 9.5 1 6,11,1.6,0,3.2,5,3 2,6.5 3 2,11,3.2
420 DATA 9.5 4.8,11,4.8,0,6.4,5,6 4,6.5,6 4,11,6.4,0,9.6,5 9.6,11,9.6
430 DATA 23
440 DATA 1 6,2,7,3,5,5,9,6,12,7,13,9,11 11,15,12,16,13,17,15,18
450 DATA 6,7,7 8 8,9,8,4,4,5 12 13,13,14,14 15,14,10,10,11 16,17,17 18
```

Al programma deve essere aggiunta la subroutine 1000, già descritta.

7.7.3 - Codifica in GWBASIC

```
 10              SCHEMA GEOMETRICO DI UN TELAIO PIANO
 20
 30 DIM X(100),Y(100) E1(100),E2(100)
 90 '            legge i dati
100 READ N
110 FOR I=1 TO N
120       READ X(I) Y(I)
130 NEXT I
140 READ M
150 FOR J=1 TO M
160       READ E1(J) E2(J)
170 NEXT J
180 '           determina i valori limite di X ed Y
190 X1=X(1)
200 X2=X(1)
210 Y1=Y(1)
220 Y2=Y(1)
230 FOR I=2 TO N
240       IF X1>X(I) THEN X1=X(I)
250       IF X2<X(I) THEN X2=X(I)
260       IF Y1>Y(I) THEN Y1=Y(I)
270       IF Y2<Y(I) THEN Y2=Y(I)
280 NEXT I
290             individua la scala del disegno
300 GOSUB 1000
310 WINDOW (X1,Y1)-(X2,Y2)
320 '           ciclo di graficizzazione
330 FOR J=1 TO M
350       LINE (X(E1(J)),Y(E1(J)))-(X(E2(J)),Y(E2(J)))
360 NEXT J
370             fine
380 STOP
390 '           dati
400 DATA 18
410 DATA 0 0 5 0 11 0 9 5,1 6 11,1.6,0,3.2,5,3 2,6 5 3.2 11,3.2
420 DATA 9.5,4 8,11,4 8,0,6 4 5 6.4,6 5,6.4,11,6 4 0,9 6,5,9 6,11,9.6
430 DATA 23
440 DATA 1 6.2,7,3,5,5,9,6,12.7 13,9,11,11,15,12 16,13,17,15,18
450 DATA 6 7 7 8 8,9,8,4,4,5,12 13,13,14,14,15 14 10 10,11,16,17,17 18
```

Al programma deve essere aggiunta la subroutine 1000, già descritta.

7.8 - Inserimento di scritte nel disegno

La presenza di scritte esplicative è spesso necessaria per la completezza di un disegno. Ad esempio, la presenza del numero d'ordine in corrispon-

denza a ciascun nodo rende più semplice l'esame di uno schema piano di telaio. Oppure, l'aggiunta del valore in prossimità dei punti di massimo rende più leggibile un diagramma

Nei casi citati, le scritte devono essere inserite in punti individuati in base al sistema logico di riferimento. Il BASIC HP contiene una vasta gamma di comandi che facilitano questo compito. L'istruzione base è costituita dalla parola "**LABEL**" (che in inglese vuol dire "etichetta") seguita da una espressione alfanumerica. Il valore dell'espressione è visualizzato in corrispondenza della posizione corrente del puntatore grafico. Altre istruzioni, per le quali si rinvia ai manuali del computer, consentono di definire la grandezza e l'inclinazione della scritta

Nel GWBASIC non esistono invece istruzioni specifiche. Possono essere utilizzati particolari comandi che consentono di definire lo stato di un insieme di punti dello schermo, creandosi un proprio alfabeto grafico. Si possono in tal modo realizzare scritte di dimensione diversa o inclinate rispetto all'orizzontale. In alternativa, molto più semplicemente ma con risultati di minore qualità, si può usare l'istruzione LOCATE per posizionare il puntatore alfanumerico, che è distinto da quello grafico, e la PRINT per scrivere il valore voluto. In questo caso, se si vuole visualizzare una scritta in un punto definito mediante le sue coordinate logiche è necessario trasformare queste in indicazioni di riga e colonna. Come in tutti i casi ripetitivi, conviene demandare anche questa operazione ad un sottoprogramma, che di seguito si descrive

Si sono utilizzate per esso le seguenti variabili:

variabili di ingresso:

SC	modalità grafica;
P1,Q1	coordinate in pixel dell'estremo superiore sinistro della finestra;
P2,Q2	coordinate in pixel dell'estremo inferiore destro della finestra;
X1,X2	estremi logici delle ascisse;
Y1,Y2	estremi logici delle ordinate;
X,Y	coordinate logiche del punto in cui si vuol porre la scritta;

variabili interne:

PC	numero di pixel occupati in orizzontale da un carattere;
QC	numero di pixel occupati in verticale da un carattere;
CM	numero massimo di colonne;
P,Q	coordinate in pixel del punto in cui si vuol porre la scritta;

variabili di uscita:

R,C	riga e colonna d'inizio della scritta.

Esaminiamo ora in dettaglio le operazioni che bisogna compiere. Prima di tutto occorre trasformare le coordinate logiche in coordinate in pixel. L'intervallo delle ascisse X1−X2 corrisponde all'intervallo in pixel P1−P2. Sfruttando tale proporzionalità, l'ascissa cercata è calcolata in funzione di quella logica mediante l'espressione $P = P1 + (X-X1)/(X2-X1) \times (P2-P1)$.

In maniera analoga, ma tenendo conto del differente orientamento tra asse fisico e logico delle ordinate, si determina l'altra coordinata in pixel.

Da questi valori si può risalire al numero d'ordine di riga e colonna. Infatti, se un carattere occupa in orizzontale 8 pixel, il primo di una riga corrisponde ai pixel da 0 a 7, il secondo a quelli da 8 a 15, e così via. Il passaggio dalla posizione in pixel al numero d'ordine è ottenuto semplicemente dividendola per la dimensione in pixel del carattere ed incrementando di 1 la parte intera del risultato.

Si riporta di seguito la codifica in GWBASIC del sottoprogramma, insieme ad un programma esemplificativo che richiede ciclicamente posizione e testo da visualizzare.

```
 10              PROGRAMMA PRINCIPALE
 20
 90              lettura dati
100 READ SC
110 READ P1 P2 Q1,Q2
120 READ X1,X2,Y1,Y2
130 SCREEN SC
140 KEY OFF
150 CLS
160             definizione finestra e coordinate logiche
170 VIEW (P1 Q1)-(P2,Q2)
180 WINDOW (X1,Y1)-(X2,Y2)
190             disegno contorno e assi coordinati
200 LINE (X1,Y1)-(X1,Y2),,,&H8888
210 LINE -(X2 Y2),,,&H8888
220 LINE -(X2,Y1),,,&H8888
230 LINE -(X1,Y1),,,&H8888
240 LINE (X1,0)-(X2,0)
250 LINE (0,Y1)-(0,Y2)
260             ciclo :
270 LOCATE 1,1
280 PRINT "X,Y,scritta';
290 INPUT X,Y,A$
300 GOSUB 1500
310 LOCATE R C
320 PRINT A$
330 GOTO 270
490             dati
500 DATA 2
510 DATA 231,639,0,335
520 DATA -25,25,-10 10
1470
1480            SCRITTURA DI CARATTERI SU UN DISEGNO
1490            individuazione di riga e colonna
1500 IF SC=1 THEN PC=8 : QC=8 : CM=40
1510 IF SC=2 THEN PC=8 : QC=8 : CM=80
1520 IF SC>2 THEN PC=8 : QC=16: CM=80
1530 P=P1+(X-X1)/(X2-X1)*(P2-P1)
1540 Q=Q1+(Y2-Y)/(Y2-Y2)*(Q2-Q1)
1550 R=INT(Q/QC)+1
```

```
1560 C=INT(P/PC)+1
1570 IF R<1 THEN R=1
1580 IF R>25 THEN R=25
1590 IF C<1 THEN C=1
1600 IF C>CM THEN C=CM
1610 RETURN
```

CAPITOLO OTTAVO

RISOLUZIONE DI UNA TRAVE CONTINUA

8.1 - Introduzione

A conclusione del testo, si è voluto riportare un esempio completo, che racchiudesse tutte le problematiche esposte. Si è fatto quindi riferimento alla risoluzione di uno schema di trave continua, problema sicuramente noto al lettore e dotato di una complessità non eccessiva, ma sufficiente a rendere l'idea delle difficoltà cui il programmatore andrà incontro. L'approccio segue lo schema indicato nel secondo capitolo. L'analisi del problema e la definizione della struttura logica sequenziale del programma vengono effettuate in fasi successive, con grado di dettaglio crescente. Nei paragrafi 8.2 e 8.3 se ne mostrano le linee generali, approfondite poi nei paragrafi da 8.4 a 8.8. La codifica è riportata nei paragrafi 8.9 e 8.10, e la risoluzione di esempi di controllo nel paragrafo 8.11. L'intero capitolo costituisce una esauriente documentazione del programma, sia con riferimento all'utente che al programmatore.

Lo scopo che mi sono prefisso è ovviamente la illustrazione di un approccio metodologico e non la creazione di un programma esauriente dal punto di vista professionale. Lascio quindi al lettore il compito di ulteriori sviluppi, quali la valutazione dell'inviluppo di più schemi di carico o la determinazione automatica dell'area di ferro necessaria.

8.2 - Oggetto del calcolo

Nell'attività professionale dell'ingegnere civile si incontrano numerosi elementi per i quali si può adottare lo schema di trave continua. Tipici esempi sono costituiti dai solai degli edifici, le solette da ponte, le travi di edifici soggetti prevalentemente a carichi verticali. Nell'automatizzarne il calcolo è possibile seguire due differenti impostazioni: creare un programma generale, adatto a tutte le situazioni, che risolva lo schema astratto di trave continua; oppure creare programmi orientati allo specifico elemento, ad esempio al calcolo di solai. In questo secondo caso l'input può risultare semplificato e l'output più completo (ad esempio riportando anche l'entità delle fasce piene e semipiene necessarie per il solaio).

Le due impostazioni sono entrambe valide ed ugualmente indispensabili per il professionista, che dovrebbe abituarsi ad usare una vasta gam-

ma di programmi, generali e specifici, commisurati alle esigenze della singola situazione Per i fini didattici che ci si propone nel testo, si è ritenuto più idoneo seguire esclusivamente la prima via. Oggetto del calcolo è quindi lo schema geometrico di trave continua, cioè un insieme lineare di aste (travi alla De Saint Venant) rigidamente collegate tra loro agli estremi, in corrispondenza di punti esternamente vincolati Si è però preferito limitarne la generalità introducendo alcune ipotesi semplificative, in relazione sia alla geometria che ai carichi

Lo schema geometrico previsto ha pertanto vincoli esterni costituiti da appoggi fissi; è così esclusa la possibilità di cedimenti elastici o anelastici o di vincoli di incastro. La sezione di ogni trave è costante per tutta la campata; si è ipotizzato che essa abbia forma a T, o come caso particolare rettangolare, e che il momento d'inerzia venga calcolato automaticamente dal computer. Nello schema non sono presenti sbalzi alle estremità; se ne può però tener conto con facilità, applicando come carico ai nodi l'azione flettente da essi trasmessa.

La risoluzione viene effettuata per una singola condizione di carico, costituita esclusivamente da carichi uniformemente distribuiti sulle campate e momenti concentrati nei nodi. I primi sono considerati positivi se diretti verso il basso, i secondi positivi se orari

Le caratteristiche generali dello schema geometrico e di carico sono riportate in figura 55.

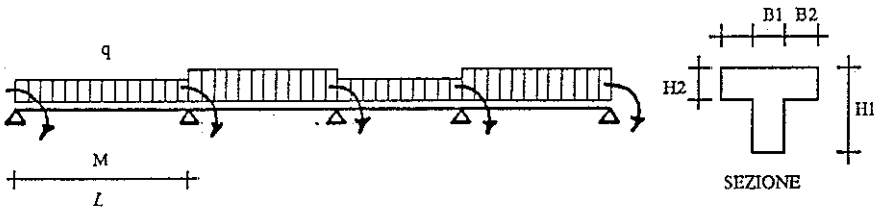


Fig. 55

8.3 Obiettivi ed impostazione generale del programma.

Dall'analisi delle diverse modalità di ingresso dati, si è individuata come impostazione ottimale quella che prevede l'utilizzazione di maschere, e quindi dei concetti e dei sottoprogrammi descritti nel capitolo 6 Per consentire l'uso dei dati in successive elaborazioni, è necessario poter disporre nei campi della maschera valori iniziali prelevati dal disco,

nonchè poter memorizzare su esso i dati dopo averli assegnati La riutilizzazione immediata di valori può invece essere consentita strutturando il programma con un ciclo che preveda al termine delle operazioni di uscita il ritorno alla fase di input

La risoluzione dello schema può essere effettuata mediante numerosi procedimenti. Tra essi si è preferito il metodo degli spostamenti, che assume come incognite le rotazioni dei nodi, pervenendo alla scrittura diretta del sistema di equazioni di equilibrio dei nodi. Per la risoluzione del sistema si utilizzerà il procedimento di riduzione di Gauss-Jordan, che si presenta in questo caso sotto una forma particolarmente semplice.

Una particolare cura deve infine essere posta nelle fase di output, affinchè ciò che essa produce sia effettivamente un aiuto chiaro ed inequivocabile per il professionista. Una prima scelta che in ogni programma occorre effettuare è quella dell'unità cui indirizzare l'uscita. Tale scelta è legata sia alla quantità di informazioni che si avranno, sia all'uso che se ne deve fare. Ad esempio, la verifica di una sezione fornisce come risultato un numero minimo di valori, che molto spesso devono essere esaminati all'istante per prendere decisioni; è quindi preferibile visualizzarli sullo schermo Nel calcolo di una trave, invece, le informazioni che si otterranno sono un pò più complesse ed è pertanto opportuno farle stampare su carta.

Fig 56

Esaminiamo con più attenzione cosa occorre riportare nel tabulato di uscita. È importante tener presente che esso verrà in genere conservato ed utilizzato anche a distanza di tempo, e potrà essere allegato a relazioni progettuali e presentato a persone che non conoscono il programma che lo ha prodotto. Deve quindi avere una sua forma autosufficiente ed autoesplicativa. Deve innanzitutto evidenziare l'argomento, le ipotesi di base ed eventualmente anche l'autore del programma. Deve inoltre contenere una intestazione che faccia capire a che cosa (quale pratica, o altro) si riferisce il calcolo. È inoltre necessario che siano in esso chiaramente indicati tutti i dati forniti all'elaboratore, per consentirne in qualsiasi momento il controllo. I risultati devono infine essere esposti in maniera chiara e completa, tale da consentire ogni possibile riscontro. Nel caso della trave continua, si riporteranno oltre ai momenti flettenti e tagli anche le rotazioni; esse infatti, pur non essendo di grande interesse per il progettista, costituiscono le reali incognite del problema, la cui conoscenza è necessaria se si vuole verificare la rigorosità dei valori delle caratteristiche di sollecitazione.

Un aspetto non indispensabile ma comunque utile per il progettista può essere quello della diagrammazione dei risultati. Anche in questo caso l'uscita dovrebbe essere su carta, mediante un plotter o la hard-copy del video. Non si è però voluto scendere in dettagli di questo genere, maggiormente legati alle caratteristiche della singola macchina, e ci si è pertanto limitati alla visualizzazione dei diagrammi

Sulla base di quanto finora esposto, è possibile delineare la struttura logica di base del programma (fig. 56). Essa è fondamentalmente una struttura di ciclo "REPEAT ... UNTIL ... " che impone la ripetizione di una sequenza di blocchi. Questi costituiscono le diverse fasi della procedura: ingresso dati, loro memorizzazione sul disco, risoluzione dello schema, stampa dei dati, stampa dei risultati, visualizzazione dei diagrammi. Ciascun blocco viene preso dettagliatamente in esame nei paragrafi successivi.

8.4 -Ingresso dati.

8.4.1 - Divisione in pagine e organizzazione complessiva.

I dati da fornire vanno inseriti in più pagine, individuate in modo da riunire le informazioni in gruppi omogenei

La pagina 1 (codificata nella subroutine 1000) richiede le indicazioni necessarie per il prelievo e l'immagazzinamento di valori sulla memoria di massa: la scelta se effettuare tali operazioni ed il nome dei files da utilizzare. La lettura dei dati dal disco (se richiesta) segue immediatamente l'immissione di queste informazioni.

La pagina 2 (subroutine 2000) è dedicata ad alcuni dati generali, relativi

allo schema da calcolare: numero di campate, modulo elastico del materiale, intestazione da apporre al lavoro.

La pagina 3 (subroutine 3000) consente la piena definizione dello schema geometrico; richiede infatti la luce e le dimensioni della sezione di ciascuna campata, cioè una tabella di valori che può essere organizzata con i criteri già descritti nel paragrafo 6.11.

Le pagine 4 e 5 (subroutine 4000 e 5000) sono dedicate allo schema di carico, contenendo rispettivamente i carichi distribuiti sulle aste ed i momenti concentrati sugli appoggi. Entrambe sono grandezze con indice, ma non è agevole gestirle in un'unica pagina perchè le seconde sono in numero differente dalle prime (una in più); data la unitarietà concettuale, si è comunque preferito farle comparire nelle due metà di una stessa videata.

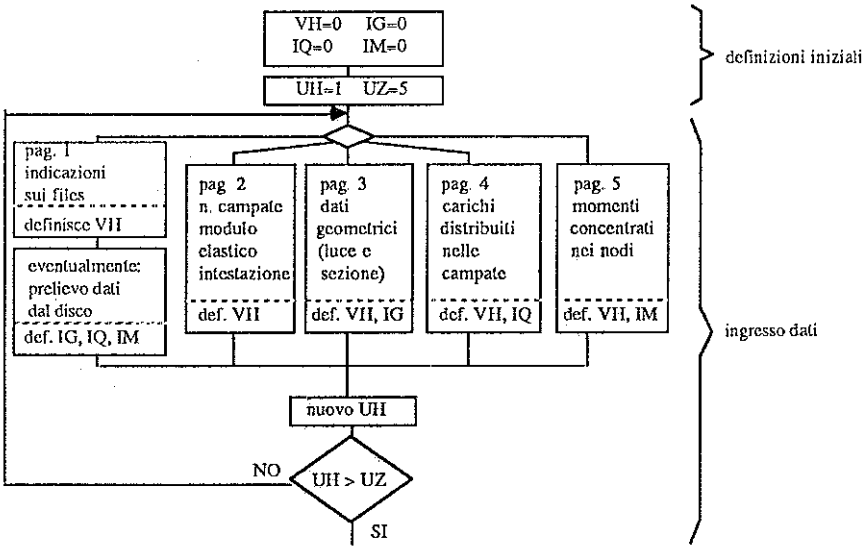


Fig 57

Le pagine sono considerate un insieme sequenziale, gestito secondo la

struttura logica mostrata in figura 57 e codificata nelle linee di programma 200-590. La variabile UH indica il numero d'ordine della pagina in esame, in base al quale si sceglie quale blocco di istruzioni di input eseguire. Essa è anche la variabile di controllo del ciclo "REPEAT ... UNTIL ...", mentre UZ rappresenta il numero totale di pagine, cioè il limite superato il quale si ha il termine della fase di ingresso dati.

Poichè la sequenza di pagine può essere percorsa andando sia in avanti che all'indietro, è necessario sapere in ogni momento quali valori siano già stati assegnati per decidere se riempire i campi con tali valori o con simboli di default. Si usa la variabile VH per indicare fino a quale pagina si è giunti nell'assegnazione di dati. Questa indicazione è sufficiente per le pagine che contengono un numero fisso di valori, ma non per quelle i cui campi sono funzione del numero di campate (numero che può essere modificato nella pagina 2). Si utilizzano quindi altre tre variabili, IG, IQ ed IM, per indicare rispettivamente per quante campate si sono definiti i dati geometrici o i carichi distribuiti e per quanti appoggi si sono assegnati i momenti concentrati.

8.4.2 - Controllo del valore dei campi.

Il sottoprogramma per il controllo dei valori immessi nei campi (linee a partire da 6000) presenta una struttura selettiva che indirizza, in base al numero d'ordine della pagina e del campo, al test da effettuare. Nella prima pagina, si verifica che il primo e terzo campo contengano "SI" oppure "NO" in risposta alle domande presentate. Nella seconda pagina si controlla che il numero di campate sia positivo e non superiore ad un limite massimo, scelto pari a 16 per consentire la visualizzazione di ogni insieme di dati in una sola pagina; si verifica inoltre che al modulo di elasticità sia assegnato un valore maggiore di zero. Nella terza pagina, si controlla che non vi siano valori negativi; per la luce e le prime due dimensioni della sezione si richiede che il valore sia anche diverso da zero. Nessun limite è invece posto ai valori dei carichi (quarta e quinta pagina).

8.4.3 - Unità di misura.

Nel preparare le scritte illustrative per l'ingresso dati, nonchè nello stampare dati e risultati, occorre decidere se tali valori debbano essere riferiti a particolari unità di misura. Alcuni programmi per consentire maggiore generalità preferiscono mantenersi svincolati da qualsiasi sistema di misura, e richiedono soltanto che i dati vengano forniti in unità tra loro coerenti. Ritengo però che tale impostazione vanifichi almeno in parte la ricerca di massima chiarezza che è stata posta tra i requisiti fondamentali per limitare la possibilità di errore dell'utente. Considero pertanto necessaria la precisa definizione delle unità di misura da utilizzare nel programma.

Il sistema di misura attualmente adottato in Italia praticamente dalla totalità degli ingegneri civili è il "sistema tecnico", che considera come grandezze fisiche fondamentali la lunghezza, la forza e il tempo. Unità di misura della lunghezza è il metro (m), ma è frequente anche l'uso di un suo sottomultiplo, il centimetro (cm); ad esempio le tensioni sono normalmente indicate in kg/cm^2. Unità di misura della forza è il chilogrammo peso (kg$_p$, o meno correttamente kg); spesso è anche adoperato un suo multiplo, la tonnellata (t). Unità di misura del tempo è il secondo (s).

Io trovo scomodo rinunciare all'uso di multipli e sottomultipli. Sono perciò abituato, nel realizzare programmi, ad utilizzare sempre le stesse grandezze nella fase di calcolo (in genere metro, tonnellata e secondo), ma accettare in ingresso e in uscita grandezze differenti (come il kg/cm^2), operando automaticamente le conversioni necessarie.

A livello internazionale il sistema tecnico è ormai superato, e la stessa normativa italiana si va adeguando al nuovo standard ("sistema internazionale"). Appare pertanto necessario, anche se doloroso, un graduale abbandono del vecchio sistema. Il sistema di misura internazionale considera come grandezze fisiche fondamentali la lunghezza, la massa e il tempo, le cui unità di misura sono il metro (m), il chilogrammo massa (kg) e il secondo (s). La forza è quindi una grandezza derivata, misurata in newton (N) o col suo multiplo chilonewton (kN). 1 N equivale a circa 0.1 kg$_p$ (cioè un etto) ed 1 kN a circa 100 kg$_p$.

Anche ricorrendo a tale sistema, trovo preferibile adottare nel calcolo un insieme base di grandezze (m, kN, s), accettando in ingresso ed uscita multipli e sottomultipli (ad esempio, le tensioni misurate in N/mm^2).

Quale dei due sistemi di misura scegliere dunque, in questa fase di transizione? Ho ritenuto preferibile consentire l'uso di entrambi, introducendo nel programma la variabile MI, che vale 1 o 2 per indicare il primo o il secondo sistema. Le scritte relative ad unità di misura sono state quindi sdoppiate, e vengono selezionate in base a tale valore.

8.4.4 - Variabili utilizzate.

Variabili da definire nella prima pagina:

- I$ contiene "SI" o "NO" per indicare se si vuole, o no, modificare valori memorizzati sul disco;
- FI$ nome del file che contiene i valori da modificare;
- O$ contiene "SI" o "NO" per indicare se si vuole, o no, memorizzare i dati sul disco;
- FO$ nome del file nel quale memorizzare i dati.

Variabili da definire nella seconda pagina:

- IZ numero di campate;
- EL modulo di elasticità (kg/cm^2 o N/mm^2);
- IT$ intestazione del calcolo

Variabili da definire nella terza pagina:

DL() array che contiene la luce delle campate;
B1() array che contiene la larghezza dell'anima della sezione a T (cm);
H1() array che contiene l'altezza totale della sezione a T (cm);
B2() array che contiene la sporgenza dell'ala della sezione a T (cm);
H2() array che contiene lo spessore dell'ala della sezione a T (cm)

Variabili da definire nella quarta pagina:

Q() array che contiene i carichi uniformemente distribuiti sulle campate (t/m o kN/m).

Variabili da definire nella quinta pagina:

MN() array che contiene i momenti concentrati sugli appoggi (tm o kNm)

Variabili di ingresso o uscita dai sottoprogrammi standard della maschera:

UA, UD, UD$, UF, UJ, UT, UU

Altre variabili:

I indice che individua la campata o l'appoggio generico;
IG numero di campate per le quali si sono assegnati i dati geometrici;
IQ numero di campate per le quali si sono assegnati i carichi distribuiti;
IM numero di appoggi per i quali si sono assegnati i momenti concentrati;
MI sistema di misura utilizzato: MI=1 sistema tecnico
MI=2 sistema internazionale;
UH numero d'ordine della pagina in esame;
UZ numero totale di pagine;
VH numero d'ordine dell'ultima pagina di cui sono già stati definiti i valori.

8.5 - Memorizzazione dei dati sul disco.

Per ripetere una stessa elaborazione a distanza di tempo, eventualmente con qualche modifica, è necessario conservare nella memoria di massa tutte le informazioni definite nelle pagine 2, 3, 4 e 5. Poichè questi valori devono sempre essere utilizzati in blocco, sia nel prelevarli che nell'immagazzinarli su disco, è preferibile adoperare un file sequenziale. Le istruzioni del sottoprogramma 10000, che consente la memorizzazione, non presentano alcuna particolarità: viene assegnato un buffer di transito, vengono scritti i dati con lo stesso ordine col quale saranno poi letti (nella subroutine 1000), e viene infine rilasciato il buffer. Si ricorda che, se si opera con calcolatori HP serie 80, il file su cui si vogliono archiviare i valori deve essere già stato definito con l'istruzione CREATE.

8.6 - Risoluzione dello schema.

8.6.1 - Scrittura del sistema di equazioni.

Per determinare le caratteristiche di sollecitazione e le componenti di movimento della trave continua, che è una struttura iperstatica, si utilizza il metodo degli spostamenti. Si fa riferimento ad uno schema principale ottenuto impedendo la rotazione dei nodi (cioè dei punti di estremità delle aste, posti in corrispondenza degli appoggi) e si assumono come incognite le rotazioni impedite. Tra le infinite soluzioni congruenti si individua quindi l'unica che sia anche equilibrata, scrivendo e risolvendo il sistema di equazioni che impongono l'equilibrio tra i momenti concentrati nei nodi, le azioni che insorgono nello schema principale in assenza di rotazioni (momenti d'incastro perfetto) e le azioni conseguenti alle rotazioni incognite

Nella trattazione teorica che segue si utilizza la seguente simbologia:

E	modulo di elasticità del materiale;
I	momento d'inerzia della sezione;
l	luce della campata;
w'	indice di rigidezza della trave: $w' = \frac{E\,I}{l}$;
φ	rotazione del nodo;
M	momento flettente trasmesso dal nodo all'estremo della trave (positivo se orario);
$\overline{M}$	momento d'incastro perfetto;
M_N	momento concentrato nel nodo.

Si usa il pedice S o D per indicare il riferimento all'estremo sinistro o destro della campata ed il pedice i, i−1, i+1, per indicare il numero d'ordine della campata o dell'appoggio.

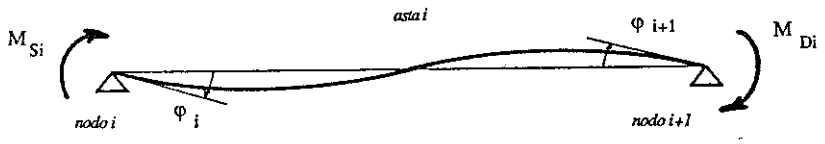


Fig 58

Per la generica asta i (fig. 58) l'azione flettente trasmessa dai nodi agli estremi della trave è fornita dalle seguenti espressioni:

$$M_{Si} = \overline{M}_{Si} + 4\,w'_i\,\varphi_i + 2\,w'_i\,\varphi_{i+1}$$

$$M_{Di} = \overline{M}_{Di} + 4\,w'_i\,\varphi_{i+1} + 2\,w'_i\,\varphi_i$$

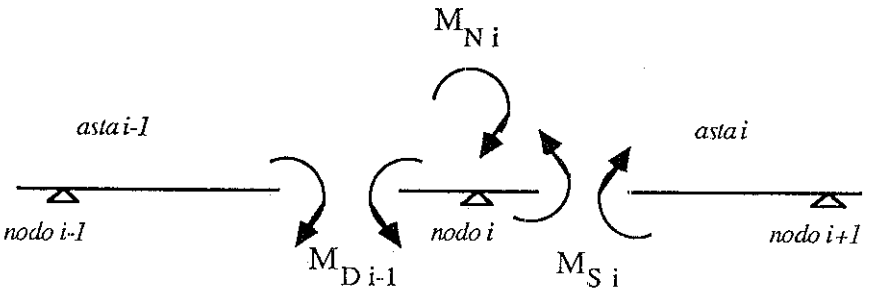


Fig. 59

L'equilibrio alla rotazione del generico nodo i (fig. 59) è espresso dalla relazione:

$$M_{D\,i-1} + M_{Si} = M_{Ni}$$

Esprimendo in essa il momento in funzione della rotazione dei nodi si ha:

$$\overline{M}_{D\,i-1} + 4\,w'_{i-1}\,\varphi_i + 2\,w'_{i-1}\,\varphi_{i-1} + \overline{M}_{Si} + 4\,w'_i\,\varphi_i + 2\,w'_i\,\varphi_{i+1} = M_{Ni}$$

ovvero:

$$2\,w'_{i-1}\,\varphi_{i-1} + (4\,w'_{i-1} + 4\,w'_i)\,\varphi_i + 2\,w'_i\,\varphi_{i+1} = M_{Ni} - \overline{M}_{D\,i-1} - \overline{M}_{Si}$$

che si può scrivere:

$$b_{i-1}\,\varphi_{i-1} + a_i\,\varphi_i + b_i\,\varphi_{i+1} = c_i$$

avendo posto:

$$a_i = 4\,w'_{i-1} + 4\,w'_i\,;$$

$$b_i = 2\,w'_i\,;$$

$$c_i = M_{Ni} - \overline{M}_{D\,i-1} - \overline{M}_{Si}$$

Applicando questa espressione a tutti gli n nodi, si ottiene il sistema di n equazioni in n incognite:

$$\begin{aligned} a_1\,\varphi_1 + b_1\,\varphi_2 &= c_1 \\ b_1\,\varphi_1 + a_2\,\varphi_2 + b_2\,\varphi_3 &= c_2 \end{aligned}$$

$$b_2 \varphi_2 + a_3 \varphi_3 + b_3 \varphi_4 = c_3$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$b_{n-2} \varphi_{n-2} + a_{n-1} \varphi_{n-1} + b_{n-1} \varphi_n = c_{n-1}$$

$$b_{n-1} \varphi_{n-1} + a_n \varphi_n = c_n$$

8.6.2 - Risoluzione del sistema di equazioni.

La risoluzione di un sistema di equazioni lineari è un problema di analisi numerica che si incontra molto di frequente, anche in campi diversi dall'ingegneria civile. Numerosi sono i procedimenti proposti, che risultano più o meno idonei secondo le differenti situazioni. Nel caso in esame, si preferisce utilizzare il metodo di riduzione di Gauss-Jordan. Esso prevede due fasi. Inizialmente si trasforma ciascuna equazione, dalla seconda fino all'ultima, combinandola linearmente con la precedente in maniera tale da annullare, nella generica equazione i, i coefficienti delle prime i−1 incognite. Si ottiene in tal modo un sistema di equazioni triangolare, cioè un sistema per il quale la matrice dei coefficienti delle incognite presenta valori nulli al di sotto della diagonale principale. Nella seconda fase si procede alla determinazione del valore delle incognite, partendo dall'ultima equazione (in cui ora è rimasta una sola incognita) fino alla prima

Il sistema di equazioni che si è scritto per imporre l'equilibrio dei nodi della trave continua presenta una particolarità. La matrice dei coefficienti delle incognite è tridiagonale, cioè ha termini tutti nulli, ad eccezione di quelli della diagonale principale e delle due diagonali ad essa immediatamente adiacenti. Il procedimento di riduzione si presenta in tal caso in forma particolarmente semplice.

Esaminiamo innanzitutto la prima fase, con riferimento all'equazione i (supponendo quindi di aver già operato sulle equazioni precedenti). Le equazioni i−1 (già ridotta) e i si scrivono:

$$\begin{aligned} a_{i-1} \varphi_{i-1} + b_{i-1} \varphi_i &= c_{i-1} \\ b_{i-1} \varphi_{i-1} + a_i \varphi_i + b_i \varphi_{i+1} &= c_i \end{aligned}$$

Sommando all'equazione i l'equazione i−1, con tutti i termini moltiplicati per $-b_{i-1}/a_{i-1}$, essa diventa:

$$a'_i \varphi_i + b_i \varphi_{i+1} = c'_i$$

con:

$$a'_i = a_i - \frac{b^2_{i-1}}{a_{i-1}}$$

$$c'_i = c_i - \frac{b_{i-1}\, c_{i-1}}{a_{i-1}}$$

I nuovi coefficienti a'_i e c'_i sostituiscono completamente i vecchi a_i e c_i e sono ad essi equivalenti come significato; di seguito verranno quindi indicati senza apice.

Il sistema di equazioni al termine della prima fase diventa pertanto:

$$\begin{array}{lr}
a_1 \varphi_1 + b_1 \varphi_2 & = c_1 \\
\quad a_2 \varphi_2 + b_2 \varphi_3 & = c_2 \\
\quad\quad a_3 \varphi_3 + b_3 \varphi_4 & = c_3 \\
\quad\quad\quad \ldots\ldots\ldots & \ldots \\
\quad\quad\quad\quad a_{n-1} \varphi_{n-1} + b_{n-1} \varphi_n & = c_{n-1} \\
\quad\quad\quad\quad\quad a_n \varphi_n & = c_n
\end{array}$$

Esaminiamo ora la seconda fase, con riferimento all'equazione i (supponendo quindi di aver già operato sulle equazioni successive). Le equazioni i e i+1 (già ridotta) si scrivono:

$$\begin{array}{lr}
a_i \varphi_i + b_i \varphi_{i+1} & = c_i \\
\varphi_{i+1} & = c_{i+1}
\end{array}$$

Sostituendo nell'equazione i il valore di φ_{i+1} fornito dall'equazione i+1 e dividendo per a_i si ottiene:

$$\varphi_i = c'_i \qquad \text{con} \qquad c'_i = \frac{c_i - b_i\, c_{i+1}}{a_i}$$

Anche in questo caso, si può omettere l'apice del termine c'_i.

Al termine della seconda fase il sistema di equazioni è stato risolto, perchè presenta ormai l'aspetto:

$$\begin{array}{lr}
\varphi_1 & = c_1 \\
\quad \varphi_2 & = c_2 \\
\quad\quad \varphi_3 & = c_3 \\
\quad\quad\quad \ldots\ldots & \ldots \\
\quad\quad\quad\quad \varphi_{n-1} & = c_{n-1} \\
\quad\quad\quad\quad\quad \varphi_n & = c_n
\end{array}$$

8.6.3 - Descrizione del sottoprogramma

Il sottoprogramma che effettua la risoluzione dello schema segue l'impostazione sin quì descritta. Innanzitutto (linee 11000-11120) calcola momento d'inerzia ed indice di rigidezza delle aste, dopo aver convertito i

valori del modulo elastico e delle dimensioni della sezione nelle unità di misura principali (m e t oppure m e kN). Determina poi (linee 11140-11170) i momenti d'incastro perfetto dovuti al carico uniforme, memorizzandoli negli array MS ed MD. Nel caso si voglia modificare la tipologia di carico, ad esempio introducendo forze concentrate o carichi triangolari, è sufficiente adattare solo questo blocco di istruzioni, perchè il resto del sottoprogramma utilizza i valori di MS ed MD, prescindendo dal tipo di carico che li ha prodotti. Il successivo gruppo di istruzioni (linee 11190-11350) calcola i coefficienti delle incognite ed i termini noti ed effettua le due fasi per la soluzione del sistema di equazioni. Viene infine determinato (linee 11380-11430) anche il valore delle caratteristiche di sollecitazione (momento flettente e taglio) agli estremi delle travi, utilizzando per esse la convenzione dei segni della Scienza delle Costruzioni.

8.6.4 - Variabili utilizzate.

E	modulo di elasticità del materiale (t/m^2 o kN/m^2);
B1	larghezza dell'anima della sezione (m);
H1	altezza totale della sezione (m);
B2	sporgenza dell'ala della sezione (m);
H2	spessore dell'ala della sezione (m);
A	area della sezione (m^2);
S	momento statico della sezione rispetto al bordo superiore (m^3);
X	distanza del baricentro della sezione dal bordo superiore (m);
ZI()	array che contiene i momenti d'inerzia delle sezioni (m^4);
W()	array che contiene gli indici di rigidezza delle aste (tm o kNm);
MS()	array che contiene il momento d'incastro perfetto all'estremo sinistro delle aste; dopo la risoluzione del sistema, in esso viene conservato il valore del momento flettente all'estemo sinistro delle aste (tm o kNm);
MD()	array che contiene momento d'incastro perfetto e poi momento flettente all'estremo destro dell'asta (tm o kNm);
TS()	array che contiene i valori del taglio all'estremo sinistro delle aste (t o kN);
TD()	array che contiene i valori del taglio all'estremo destro delle aste (t o kN);
A()	array che contiene i valori dei coefficienti a_i;
B()	array che contiene i valori dei coefficienti b_i;
C()	array che contiene i valori dei coefficienti c_i; dopo la risoluzione del sistema, essi rappresentano il valore delle rotazioni dei nodi.

8.7 - Stampa dei dati e dei risultati.

Nell'organizzare il tabulato di uscita si sono seguiti i criteri già definiti nel paragrafo 8.3. Operativamente, la stampa viene effettuata mediante due sottoprogrammi.

Il primo (subroutine 12000) fornisce una sintetica descrizione del programma ed una panoramica completa dei dati assegnati per il calcolo. Viene stampata innanzitutto l'intestazione del lavoro. Segue un breve elenco delle ipotesi di base del programma e delle convenzioni dei segni adottate. Vengono poi riportati i dati geometrici, luci delle campate e dimensioni delle sezioni, ai quali è aggiunto anche, per ulteriore riscontro, il valore del momento d'inerzia delle sezioni determinato dall'elaboratore. Dopo questi è indicato il valore del modulo di elasticità del materiale ed infine quello dei carichi applicati sullo schema. I carichi distribuiti sulle campate ed i momenti concentrati sugli appoggi sono riportati in due colonne parallele, in analogia a come sono visualizzati nella fase di ingresso dati.

Il secondo sottoprogramma (linee a partire da 13000) fornisce i risultati del calcolo. Vengono innanzitutto stampati i valori delle rotazioni, che costituiscono le incognite del problema nel metodo degli spostamenti. Si passa poi alle caratteristiche di sollecitazione. Per esse si è adottata la convenzione dei segni della Scienza delle Costruzioni: il momento flettente è considerato positivo se tende le fibre inferiori della trave; il taglio positivo se, come azione sulla faccia di normale uscente diretta verso destra, è diretto verso il basso. Per consentire un più agevole uso dei risultati, si sono calcolati e stampati i valori delle caratteristiche di sollecitazione in più punti di ciascuna trave, ottenuti dividendo la campata in 5 parti uguali. Il taglio ed il momento flettente nella generica sezione a distanza x dall'estremo sinistro sono forniti dalle espressioni:

$$T(x) = T_s - q\,x$$
$$M(x) = M_s + T_s\,x - q\,x^2/2$$

Nella stampa dei valori di ingresso ed uscita, si è utilizzata la notazione esponenziale solo per le rotazioni, che sono rappresentate da numeri molto piccoli e con una più ampia gamma di ordini di grandezza. In tutti gli altri casi si è preferita la usuale notazione decimale, con un numero prefissato di cifre prima e dopo la virgola. Si hanno in questo modo maggiori vincoli, perchè i valori sono riportati in maniera corretta ed esauriente solo se rispettano l'ordine di grandezza previsto; ritengo però che il tabulato risulti così più facilmente leggibile dall'utente. Ogni valore è sempre seguito dall'indicazione della relativa unità di misura. Avendo consentito l'uso di due diversi sistemi di misura, si è utilizzata la variabile MI per selezionare tra codici di formattazione alternativi (contenuti in istruzioni IMAGE nel BASIC HP o memorizzati nelle stringhe A$ e B$ nel GWBASIC).

Nel calcolare e stampare le caratteristiche di sollecitazione lungo la campata

si sono utilizzate, oltre a quelle già definite nelle fasi precedenti, le seguenti variabili:

J	indice che individua il numero d'ordine del punto in esame;
X()	array che contiene l'ascissa dei punti, cioè la distanza dall'estremo sinistro della campata (m);
T()	array che contiene il valore del taglio nel punto generico (t o kN);
M()	array che contiene il valore del momento nel punto generico (tm o kNm).

8.8 - Diagramma del momento flettente e del taglio.

Un primo problema da affrontare nel sottoprogramma che consente la visualizzazione dei diagrammi di momento flettente e taglio (linee da 14000) è quello della definizione della scala del disegno.

Si sono indicate con FX ed FY le dimensioni reali, in metri, dell'area grafica riservata a ciascun diagramma. Nel programma sviluppato si è ipotizzato che essa coincida con l'intero schermo, visualizzando quindi separatamente i due disegni; se invece si preferisce vederli contemporaneamente, basta far riferimento a due distinte finestre nel video.

La dimensione orizzontale massima del disegno è pari alla lunghezza totale LT della trave continua (o meglio a qualcosa in più di questa, se si vuol disegnare anche il simbolo dell'appoggio). Per ottenere un disegno più ampio possibile essa dovrà coincidere col lato dell'area grafica. Si sono pertanto definiti come limiti per le ascisse i valori $-0.02 \times LT$ e $+1.02 \times LT$.

La scala delle ordinate deve essere definita in funzione del massimo (in valore assoluto) della caratteristica da diagrammare. Nel caso del taglio, per i tipi di carico previsti esso assume il valore massimo TX sicuramente all'estremo di una campata. Nel cercare il massimo momento MX, occorre invece tener conto anche del momento positivo in campata. Al posto di questo si è per semplicità utilizzato il valore $ql^2/8$, sicuramente maggiore. Utilizzando per il diagramma tutta l'altezza dell'area grafica, si rischia di ottenere un disegno poco gradevole esteticamente, perchè tutto stirato in verticale. Si è quindi deciso di fare in modo che le dimensioni reali di MX e TX siano in un preciso rapporto con la dimensione reale LM della massima campata. Tale rapporto è stato fissato pari rispettivamente a circa 0.5 e 0.25, valori prescelti in base a considerazioni estetiche (indubbiamente soggettive) e dopo qualche tentativo. I limiti delle ordinate sono quindi stati posti pari a +MM e −MM per il momento flettente ed a +TM e −TM per il taglio, dove:

$$MM = MX \times \frac{LT}{LM} \times \frac{FY}{FX}$$

e

$$IM = 2 \times TX \times \frac{LT}{LM} \times \frac{FY}{FX}$$

Si noti che nel BASIC HP è possibile assegnare all'estremo inferiore un valore positivo e a quello superiore un valore negativo, in modo da riportare al di sotto della linea d'asse i valori positivi delle caratteristiche di sollecitazione Nel GWBASIC lo stesso effetto può essere ottenuto solo cambiando il segno ai valori da diagrammare.

Definita la scala, si procede a disegnare lo schema geometrico della trave. Poichè questa operazione deve essere effettuata due volte, la si è codificata in un sottoprogramma separato (linee da 14800).

I valori del momento da diagrammare vengono calcolati con la stessa formula indicata nel paragrafo precedente; per approssimare meglio l'andamento curvilineo ogni campata è stata divisa in 20 intervalli. Il diagramma del taglio è invece lineare in ciascuna campata, ed è quindi costruito semplicemente unendo i punti che ne rappresentano i valori alle estremità.

Oltre a quelle definite in precedenza, si sono utilizzate le seguenti variabili:

FX,FY	dimensioni reali dell'area grafica, finestra o schermo (m);
LM	lunghezza massima delle campate (m);
LT	lunghezza totale della trave continua (m);
MX	valore massimo del momento flettente (tm o kNm);
TX	valore massimo del taglio (t o kN);
V	momento in campata $ql^2/8$;
MM	valore limite del momento flettente, nella definizione della scala delle ordinate (tm o kNm);
TM	valore limite del taglio, nella definizione della scala delle ordinate (t o kN);
HA	valore in scala, corrispondente all'altezza del simbolo grafico di appoggio (stesse unità di misura del momento o del taglio);
XA	distanza dell'estremo sinistro della generica campata dal primo appoggio (m);
J	indice che individua il numero d'ordine del punto in esame;
X	ascissa del punto generico, cioè distanza dall'estremo sinistro della campata (m);
M	momento flettente nel punto generico (tm o kNm);
T	taglio nel punto generico (t o kN).

8.9 - Codifica in BASIC HP

```
10 !          RISOLUZIONE DI UNA TRAVE CONTINUA
20 !
```

```
 80 !            PROGRAMMA PRINCIPALE
 90 !
100 PAGESIZE 24
110 PRINTER IS 701
120 OPTION BASE 1
130 DIM IT$[60],DL(16),B1(16),H1(16),B2(16),H2(16),Q(16),MN(17)
140 DIM ZI(16),W(16),MS(16),MD(16),TS(16),TD(16)
150 DIM A(17),B(17),C(17),X(5),M(5),T(5)
190 !
200 MI=1 !      unità di misura:      MI=1    sistema tecnico
210 !                                 MI=2    sistema internazionale
280 !
290 !           definizioni iniziali per ingresso dati
300 GOSUB 7800
310 VH=0
320 IG=0
330 IQ=0
340 IM=0
350 UZ=5
360 UH=1
390 !           ingresso dati
400 ON UH GOTO 420,450,480,510 540
410 !
420 GOSUB 1000
430 GOTO 570
440 !
450 GOSUB 2000
460 GOTO 570
470 !
480 GOSUB 3000
490 GOTO 570
500 !
510 GOSUB 4000
520 GOTO 570
530 !
540 GOSUB 5000
550 GOTO 570
560 !
570 UH=UH+UU
580 IF UH=0 THEN UH=1
590 IF UH<= UZ THEN 400
600 !           memorizzazione valori su disco
610 IF O$="SI" THEN GOSUB 10000
620 !           risoluzione dello schema
630 GOSUB 11000
640 !           stampa dati
650 GOSUB 12000
660 !           stampa risultati
670 GOSUB 13000
680 !           visualizzazione diagrammi
690 GOSUB 14000
700 !           richiesta ripetizione calcolo
710 CLEAR
720 DISP "Vuoi ripetere il calcolo (SI O NO)";
```

```
 730 INPUT R$
 740 IF R$="SI" THEN UH=2 @ GOTO 400
 750 !              fine
 760 CLEAR
 770 PAGESIZE 16
 780 DISP "FINE ELABORAZIONE"
 790 STOP
 980 !
 990 !              PRIMA PAGINA - indicazioni sui files
1000 CLEAR
1010 RESTORE 1500
1020 GOSUB 7000
1030 GOSUB 7200
1040 GOSUB 7500
1050 GOSUB 8000
1060 UJ=1
1070 GOSUB 7650 @ S$=UD$
1080 GOSUB 7650 @ I$=UD$
1090 GOSUB 7650 @ O$=UD$
1100 GOSUB 7650 @ FO$=UD$
1110 VH=1
1120 IF I$ <> "SI" THEN 1280
1130 !              lettura valori da disco
1140 ASSIGN# 1 TO FI$
1150 READ# 1 ; IT$,IZ,EL
1160 FOR I=1 TO IZ
1170       READ# 1 ; DL(I) B1(I) H1(I),B2(I),H2(I),Q(I)
1180 NEXT I
1190 FOR I=1 TO IZ+1
1200       READ# 1 ; MN(I)
1210 NEXT I
1220 ASSIGN# 1 TO *
1230 VH=5
1240 IG=IZ
1250 IQ=IZ
1260 IM=IZ+1
1270 !
1280 RETURN
1490 !
1500 DATA 4
1510 DATA 1 50 2 3 50 10,7 50,2,9,50,10
1520 DATA 5
1530 DATA 1,1,vuoi modificare valori memorizzati ? (SI o NO)
1540 DATA 3,1,se SI : in quale file ?
1550 DATA 7 1 vuoi memorizzare i dati sul disco ? (SI o NO)
1560 DATA 9,1,se SI : in quale file ?
1570 DATA 23 75 pag.1
1980 !
1990 !              SECONDA PAGINA - numero di campate
2000 CLEAR
2010 RESTORE 2500
2020 GOSUB 7000
2030 GOSUB 7200
2040 IF MI=1 THEN RESTORE 2550 ELSE RESTORE 2560
```

```
2050 GOSUB 7200
2060 IF VH<UH THEN 2140
2070 !              visualizzazione valori precedenti
2080        UJ=1
2090        UD=IZ @ GOSUB 7700
2100        UD=EL @ GOSUB 7700
2110        UD$=IT$ @ GOSUB 7725
2120 GOTO 2170
2130 !              visualizzazione valori di default
2140        GOSUB 7500
2150 GOTO 2170
2160 !              assegnazione nuovi valori
2170 GOSUB 8000
2180 UJ=1
2190 GOSUB 765( @ IZ=UD
2200 GOSUB 765  @ EL=UD
2210 GOSUB 765J @ T$=UD$
2220 IF VH<UH THEN VH=UH
2230 !
2240 RETURN
2490 !
2500 DATA 3
2510 DATA 1,25,2,4,25,7,9,15.60
2520 DATA 4
2530 DATA 1,1,numero di campate,4,1,modulo di elasticità 9 1 intestazione
2540 DATA 23,75,pag 2
2550 DATA 1,4,33,kg/cm2
2560 DATA 1 4,33,N/mm2
2980 !
2990 !              TERZA PAGINA - dati geometrici
3000 CLEAR
3010 RESTORE 3500
3020 UT=IZ
3030 GOSUB 7100
3040 GOSUB 7200
3050 GOSUB 7300
3060 UA=1
3070 GOSUB 7400
3080 !              visualizzazione valori precedenti o di default
3090 UJ=1
3100 FOR I=1 TO IZ
3110        IF I>IG THEN 3180
3120              UD=DL(I  @ GOSUB 7700
3130              UD=B1(I  @ GOSUB 7700
3140              UD=H1(I  @ GOSUB 7700
3150              UD=B2(I  @ GOSUB 7700
3160              UD=H2(I  @ GOSUB 7700
3170        GOTO 3240
3180              GOSUB 7750
3190              GOSUB 7750
3200              GOSUB 7750
3210              GOSUB 7750
3220              GOSUB 7750
3230        GOTO 3240
```

```
3240 NEXT I
3250 !              assegnazione nuovi valori
3260 GOSUB 8000
3270 UJ=1
3280 FOR I=1 TO IZ
3290       GOSUB 7650 @ DL(I)=UD
3300       GOSUB 7650 @ B1(I)=UD
3310       GOSUB 7650 @ H1(I)=UD
3320       GOSUB 7650 @ B2(I)=UD
3330       GOSUB 7650 @ H2(I)=UD
3340 NEXT I
3350 IF VH<UH THEN VH=UH
3360 IF IG<IZ THEN IG=IZ
3370 !
3380 RETURN
3490 !
3500 DATA 5 4
3510 DATA 14,7.31,5,44,5,57,5 70 5
3520 DATA 8
3530 DATA 0 25,D A T I      G E O M E T R I C I
3540 DATA 2,1,campata 2 15 luce 2 32,B1 2,45,H1 2 58 B2 2 71 H2
3550 DATA 23.75,pag 3
3560 DATA 4,22,m          cm          cm          cm          cm
3570 DATA 4 4
3980 !
3990 !              QUARTA PAGINA - carichi distribuiti
4000 IF UU=1 THEN CLEAR
4010 RESTORE 4500
4020 UT=IZ
4030 GOSUB 7100
4040 GOSUB 7200
4050 UA=1
4060 GOSUB 7400
4070 IF MI=1 THEN RESTORE 4570 ELSE RESTORE 4580
4080 GOSUB 7300
4090 !              visualizzazione valori precedenti o di default
4100 UJ=1
4110 FOR I=1 TO IZ
4120       IF I>IQ THEN 4150
4130             UD=Q(I) @ GOSUB 7700
4140       GOTO 4170
4150             GOSUB 7750
4160       GOTO 4170
4170 NEXT I
4180 !              assegnazione nuovi valori
4190 GOSUB 8000
4200 UJ=1
4210 FOR I=1 TO IZ
4220       GOSUB 7650 @ Q(I)=UD
4230 NEXT I
4240 IF VH<UH THEN VH=UH
4250 IF IQ<IZ THEN IQ=IZ
4260 !
4270 RETURN
```

```
4490 !
4500 DATA 1,5
4510 DATA 15,6
4520 DATA 5
4530 DATA 0,32,C A R I C H I
4540 DATA 2 1,campata,2,13,carico uniforme,23,34 pag 4
4550 DATA 3,11,"+ se verso il basso"
4560 DATA 5 4
4570 DATA 5 22,t/m
4580 DATA 5 22 kN/m
4980 !
4990 !              QUINTA PAGINA - momenti concentrati sugli appoggi
5000 RESTORE 5500
5010 UT=IZ+1
5020 GOSUB 7100
5030 GOSUB 7200
5040 UA=1
5050 GOSUB 7400
5060 IF MI=1 THEN RESTORE 5560 ELSE RESTORE 5570
5070 GOSUB 7300
5080 !              visualizzazione valori precedenti o di default
5090 UJ=1
5100 FOR I=1 TO IZ+1
5110       IF I>IM THEN 5140
5120             UD=MN(I) @ GOSUB 7700
5130       GOTO 5160
5140             GOSUB 7750
5150       GOTO 5160
5160 NEXT I
5170 !              assegnazione nuovi valori
5180 GOSUB 8000
5190 UJ=1
5200 FOR I=1 TO IZ+1
5210       GOSUB 7650 @ MN(I)=UD
5220 NEXT I
5230 IF VH<UH THEN VH=UH
5240 IF IM<IZ+1 THEN IM=IZ+1
5250 !
5260 RETURN
5490 !
5500 DATA 1,5
5510 DATA 56,6
5520 DATA 4
5530 DATA 2,41,appoggio,2,51 momento concentrato,23 74 pag 5
5540 DATA 3,55 "+ se orario"
5550 DATA 5,45
5560 DATA 5 63,tm
5570 DATA 5,63,kNm
5980 !
5985 !              CONTROLLO DEL VALORE DEI CAMPI
5990 !
6000 ON UH GOTO 6100,6200 6300 6600,6600
6090 !
6095 ! - - - - pag 1
```

```
6100 IF UJ<> 1 AND UJ<> 3 THEN 6115
6105      GOSUB 7600
6110      IF UD$ <> "SI" AND UD$ <>  NO" THEN GOSUB 6700
6115 GOTO 6600
6190 !
6195 ! ---- pag.2
6200 ON UJ GOTO 6220,6240,6600
6215 !
6220 GOSUB 7600
6225 IF UD<= 0 OR UD>16 THEN GOSUB 6750
6230 GOTO 6600
6235 !
6240 GOSUB 7600
6245 IF UD<= 0 THEN GOSUB 6750
6250 GOTO 6600
6290 !
6295 ! ---- pag.3
6300 GOSUB 7600
6305 ON (UJ-1) MOD 5+1 GOTO 6320,6320,6320,6340 6340
6315 !
6320 IF UD<= 0 THEN GOSUB 6750
6325 GOTO 6600
6335 !
6340 IF UD<0 THEN GOSUB 6750
6345 GOTO 6600
6595 !
6600 RETURN
6695 !
6700 AWRITE 23,1   Devi rispondere SI o NO'
6705 GOTO 6755
6745 !
6750 AWRITE 23,1, 'Valore non ammissibile "
6755 BEEP
6760 WAIT 1000
6765 AWRITE 23 1 "
6770 UF=0
6775 RETURN
```

Vanno quì inseriti i sottoprogrammi per la gestione della maschera (linee 6980–9910), già listati nel capitolo 6. Occorre solo modificare il dimensionamento degli array UR, UC, UL, i cui indici devono poter arrivare al valore 90:

```
7800 INTEGER UF(5),UR(90),UC(90),UL(90)
```

```
 9980 !
 9990 !          MEMORIZZAZIONE VALORI SU DISCO
10000 ASSIGN# 1 TO FO$
10010 PRINT# 1 ; IT$ IZ EL
10020 FOR I=1 TO IZ
10030     PRINT# 1 ; DL(I) B1(I) H1(I) B2(I) H2(I),Q(I)
```

```
10040 NEXT I
10050 FOR I=1 TO IZ+1
10060     PRINT# 1 ; MN(I)
10070 NEXT I
10080 ASSIGN# 1 TO *
10090 !
10100 RETURN
10960 !
10970 !           RISOLUZIONE DELLO SCHEMA
10980 !
10990 !           adatta EL alle unità di misura di calcolo (t/m2 o kN/m2)
11000 IF MI=1 THEN E=EL*10 ELSE E=EL*1000
11010 !           calcola il momento d'inerzia e l'indice di rigidezza
11020 FOR I=1 TO IZ
11030     B1=B1(I)/100
11040     H1=H1(I)/100
11050     B2=B2(I)/100
11060     H2=H2(I)/100
11070     A=B1*H1+2*B2*H2
11080     S=B1*H1^2/2+B2*H2^2
11090     X=S/A
11100     ZI(I)=B1*H1^3/3+2*B2*H2^3/3-A*X^2
11110     W(I)=E*ZI(I)/DL(I)
11120 NEXT I
11130 !           calcola i momenti d incastro perfetto
11140 FOR I=1 TO IZ
11150     MD(I)=Q(I)*DL(I)^2/12
11160     MS(I)=-(MD(I))
11170 NEXT I
11180 !           risoluzione dello schema
11190 A(1)=4*W(1)
11200 B(1)=2*W(1)
11210 C(1)=MN(1)-MS(1)
11220 FOR I=2 TO IZ+1
11230     A(I)=4*W(I-1)
11240     C(I)=MN(I)-MD(I-1)
11250     IF I=IZ+1 THEN 11290
11260          A(I)=A(I)+4*W(I)
11270          B(I)=2*W(I)
11280          C(I)=C(I)-MS(I)
11290     A(I)=A(I)-B(I-1)^2/A(I-1)
11300     C(I)=C(I)-B(I-1)*C(I- 1)/A(I-1)
11310 NEXT I
11320 C(IZ+1)=C(IZ+1)/A(IZ+1)
11330 FOR I=IZ TO 1 STEP -1
11340     C(I)=(C(I)-B(I)*C(I+1))/A(I)
11350 NEXT I
11360 !
11370 !           determina momento e taglio agli estremi delle travi
11380 FOR I=1 TO IZ
11390     MS(I)=MS(I)+(4*C(I)+2*C(I+1))*W(I)
11400     MD(I)=-(MD(I)+(4* C(I+1)+2*C(I))*W(I))
11410     TS(I)=(MD(I)-MS(I))/DL(I)+Q(I)*DL(I)/2
11420     TD(I)=(MD(I)-MS(I))/DL(I)-- Q(I)*DL(I)/2
```

```
11430 NEXT I
11440 !
11450 RETURN
11960 !
11970 !          STAMPA DATI
11980 !
11990 !          stampa intestazione lavoro
12000 PRINT TAB (11);IT$
12010 PRINT
12020 PRINT
12030 !          stampa descrizione sintetica del programma
12040 PRINT "RISOLUZIONE DI UNO SCHEMA DI TRAVE CONTINUA '
12050 PRINT
12060 PRINT "Schema geometrico:          aste rettilinee a sezione costante, rettangolare o a T
12070 PRINT  Vincoli esterni:            appoggi fissi '
12080 PRINT "Tipi di carico ammessi:     carichi uniformemente distribuiti sulle travi (positivi
12090 PRINT "                                 se diretti verso il basso);"
12100 PRINT "                            momenti concentrati sugli appoggi (positivi se orari) '
12110 PRINT
12120 PRINT "Convenzione dei segni per i risultati :"
12130 PRINT "       rotazione:           positiva se oraria;"
12140 PRINT "       momento flettente:   positivo se tende le fibre inferiori;"
12150 PRINT "       taglio:              positivo se diretto verso il basso, come azione sulla"
12160 PRINT '                                 faccia di normale uscente verso destra '
12170 PRINT
12180 PRINT
12190 !          stampa dati geometrici ed elastici
12200 PRINT"DATI GEOMETRICI ED ELASTICI"
12210 PRINT
12220 PRINT"campata  luce    larghezza   altezza   sporgenza   spessore   momento"
12230 PRINT"                 anima       totale    ala         ala        d'inerzia"
12240 IMAGE 2X,2D,6X,DDZ DD " m",4(3X,3DZ.D " cm") 2X DZ 6D " m4"
12250 FOR I=1 TO IZ
12260     PRINT USING 12240 ; I,DL(I),B1(I),H1(I),B2(I),H2(I),ZI(I)
12270 NEXT I
12280 PRINT
12290 IMAGE "Modulo di elasticità        E =  7D  kg/cm2 '
12300 IMAGE "Modulo di elasticità        E = ',6D,' N/mm2'
12310 IF MI=1 THEN PRINT USING 12290 ; EL ELSE PRINT USING 12300 ; EL
12320 PRINT
12330 PRINT
12340 PRINT "CARICHI UNIFORMI ';TAB (41);' MOMENTI CONCENTRATI"
12350 PRINT
12360 PRINT 'campata      carico ';TAB (41);"appoggio      momento"
12370 IMAGE 2X,2D,7X,2DZ DD," t/m ' 20X,2D 8X 2DZ DD  tm '
12380 IMAGE 2X,2D,7X,3DZ D," kN/m ',20X,2D,8X,3DZ D," kNm
12390 FOR I=1 TO IZ
12400     IF MI=1 THEN PRINT USING 12370 ; I,Q(I),I,MN(I)
12410     IF MI<> 1 THEN PRINT USING 12380 ; I Q(I) I MN(I)
12420 NEXT I
12430 IMAGE 42X 2D 8X 2DZ DD " tm "
12440 IMAGE 42X,2D,8X,3DZ.D," kNm
12450 IF MI=1 THEN PRINT USING 12430 ; I,MN(I) ELSE PRINT USING 12440 ; I MN(I)
12460 PRINT
```

```
12470 PRINT
12480 !
12490 RETURN
12960 !
12970 !            STAMPA RISULTATI
12980 !
12990 !            stampa rotazioni
13000 PRINT "ROTAZIONE DEI NODI
13010 PRINT
13020 PRINT          nodo     rotazione '
13030 FOR I=1 TO IZ+1
13040     PRINT USING "4X,2D,5X,MZ 3DE" ; I,C(I)
13050 NEXT I
13060 PRINT
13070 !            stampa caratteristiche di sollecitazione
13080 PRINT "CARATTERISTICHE DI SOLLECITAZIONE
13090 PRINT
13100 IMAGE  ascissa",6(3X,2DZ DD," m  )
13110 IMAGE  momento",6(3X,2DZ DD, ' tm')
13120 IMAGE "momento",6(2X,3DZ.D," kNm ')
13130 IMAGE  'taglio ",6(3X,2DZ DD," t ')
13140 IMAGE  taglio " 6(2X,3DZ D," kN ')
13150 FOR I=1 TO IZ
13160     PRINT "        Campata ";I
13170     PRINT
13180     FOR J=1 TO 5
13190          X(J)=J*DL(I)/5
13200          T(J)=TS(I)-Q(I)*X(J)
13210          M(J)=MS(I)+TS(I)*X( J)- Q(I)*X( J)^2/2
13220     NEXT J
13230     PRINT USING 13100 ; 0,X(1),X(2),X(3) X(4),X(5)
13240     IF MI=1 THEN PRINT USING 13110 ; MS(I),M(1),M(2),M(3),M(4),M(5)
13250     IF MI<> 1 THEN PRINT USING 13120 ; MS(I),M(1),M(2),M(3),M(4),M(5)
13260     IF MI=1 THEN PRINT USING 13130 ; TS(I),T(1) T(2),T(3) T(4) T(5)
13270     IF MI<> 1 THEN PRINT USING 13140 ; TS(I),T(1),T(2),T(3),T(4),T(5)
13280     PRINT
13290 NEXT I
13300 !
13310 RETURN
13960 !
13970 !            VISUALIZZAZIONE DIAGRAMMI
13980 !
13990 !            dimensioni reali schermo
14000 FX= 125
14010 FY= 075
14020 !            calcola luce massima e lunghezza totale
14030 LM=0
14040 LT=0
14050 FOR I=1 TO IZ
14060     IF LM<DL(I) THEN LM=DL(I)
14070     LT=LT+DL(I)
14080 NEXT I
14090 !            calcola momento e taglio massimo
14100 MX=0
```

```
14110 TX=0
14120 FOR I=1 TO IZ
14130     IF MX<ABS (MS(I)) THEN MX=ABS (MS(I))
14140     IF MX<ABS (MD(I)) THEN MX=ABS (MD(I))
14150     V=ABS (Q(I)*DL(I)^2/8)
14160     IF MX<V THEN MX=V
14170     IF TX<ABS (TS(I)) THEN TX=ABS (TS(I))
14180     IF TX<ABS (TD(I)) THEN TX=ABS (TD(I))
14190 NEXT I
14200 MM=MX*LT/LM*FY/FX @ IF MM<MX THEN MM=MX
14210 TM=2*TX*LT/LM*FY/FX @ IF TM<TX THEN TM=TX
14220 !           definisce la scala del disegno del momento flettente
14230 SCALE -( 02*LT) 1 02*LT,MM,-MM
14240 !           disegna lo schema geometrico
14250 HA=MM/25
14260 GOSUB 14800
14270 !           disegna il diagramma del momento
14280 XA=0
14290 MOVE XA,0
14300 FOR I=1 TO IZ
14310     DRAW XA,MS(I)
14320     FOR J=1 TO 20
14330         X=DL(I)*J/20
14340         M=MS(I)+TS(I)*X-Q(I)*X^2/2
14350         DRAW XA+X M
14360     NEXT J
14370     XA=XA+DL(I)
14380 NEXT I
14390 DRAW XA,0
14400 WAIT 5000
14410 !           definisce la scala del disegno del taglio
14420 SCALE -( 02*LT),1 02*LT,TM,-TM
14430 !           disegna lo schema geometrico
14440 HA=TM/25
14450 GOSUB 14800
14460 !           disegna il diagramma del taglio
14470 XA=0
14480 MOVE XA,0
14490 FOR I=1 TO IZ
14500     DRAW XA,TS(I)
14510     DRAW XA+DL(I) TD(I)
14520     XA=XA+DL(I)
14530 NEXT I
14540 DRAW XA,0
14550 WAIT 5000
14560 !
14570 RETURN
14790 !           disegna lo schema geometrico
14800 GCLEAR
14810 MOVE 0,0
14820 DRAW LT,0
14830 XA=0
14840 FOR I=0 TO IZ
14850     IF I>0 THEN XA=XA+DL(I)
```

```
14860     MOVE XA,0
14870     DRAW XA-LT/50,HA
14880     DRAW XA+LT/50,HA
14890     DRAW XA,0
14900 NEXT I
14910 RETURN
99980 !
99990 !          trave continua
99999 ! A.G. 22-11-86
```

8.10 - Codifica in GWBASIC

```
 10          RISOLUZIONE DI UNA TRAVE CONTINUA
 20
 80          PROGRAMMA PRINCIPALE
 90
100 KEY OFF
120 OPTION BASE 1
130 DIM DL(16),B1(16),H1(16),B2(16),H2(16),Q(16),MN(17)
140 DIM ZI(16),W(16),MS(16),MD(16),TS(16),TD(16)
150 DIM A(17),B(17),C(17),X(5),M(5),T(5)
190 '
200 MI=1              unità di misura :  MI=1   sistema tecnico
210 '                                    MI=2   sistema internazionale
280
290 '          definizioni iniziali per ingresso dati
300 GOSUB 7800
310 VH=0
320 IG=0
330 IQ=0
340 IM=0
350 UZ=5
360 UH=1
390 '          ingresso dati
400 ON UH GOTO 420,450,480,510,540
410 '
420 GOSUB 1000
430 GOTO 570
440 '
450 GOSUB 2000
460 GOTO 570
470 '
480 GOSUB 3000
490 GOTO 570
500 '
510 GOSUB 4000
520 GOTO 570
530 '
540 GOSUB 5000
550 GOTO 570
560 '
570 UH=UH+UU
```

```
580 IF UH=0 THEN UH=1
590 IF UH<= UZ THEN 400
600             memorizzazione valori su disco
610 IF O$= "SI" THEN GOSUB 10000
620             risoluzione dello schema
630 GOSUB 11000
640             stampa dati
650 GOSUB 12000
660             stampa risultati
670 GOSUB 13000
680             visualizzazione diagrammi
690 GOSUB 14000
700             richiesta ripetizione calcolo
710 CLS
720 PRINT "Vuoi ripetere il calcolo (SI o NO) ";
730 INPUT R$
740 IF R$="SI" THEN UH=2 : GOTO 400
750             fine
760 CLS
770 KEY ON
780 PRINT "FINE ELABORAZIONE"
790 STOP
980 '
990             PRIMA PAGINA - indicazioni sui files
1000 CLS
1010 RESTORE 1500
1020 GOSUB 7000
1030 GOSUB 7200
1040 GOSUB 7500
1050 GOSUB 8000
1060 UJ=1
1070 GOSUB 7650 : I$=UD$
1080 GOSUB 7650 : FI$=UD$
1090 GOSUB 7650 : O$=UD$
1100 GOSUB 7650 : FO$=UD$
1110 VH=1
1120 IF I$ <> "SI" THEN 1280
1130            lettura valori da disco
1140 OPEN "I ",#1,FI$
1150 INPUT #1, IT$,IZ,EL
1160 FOR I=1 TO IZ
1170      INPUT #1 DL(I),B1(I),H1(I) B2(I) H2(I),Q(I)
1180 NEXT I
1190 FOR I=1 TO IZ+1
1200      INPUT #1 MN(I)
1210 NEXT I
1220 CLOSE #1
1230 VH=5
1240 IG=IZ
1250 IQ=IZ
1260 IM=IZ+1
1270 '
1280 RETURN
1490 '
```

```
1500 DATA 4
1510 DATA 2,51,2,4,51,10,8,51,2,10,51,10
1520 DATA 5
1530 DATA 2,2,vuoi modificare valori memorizzati ? (SI o NO)
1540 DATA 4,2,se SI :          in quale file ?
1550 DATA 8,2,vuoi memorizzare i dati sul disco ? (SI o NO)
1560 DATA 10,2,se SI :         in quale file ?
1570 DATA 25,76,pag 1
1980 '
1990 '          SECONDA PAGINA - numero di campate
2000 CLS
2010 RESTORE 2500
2020 GOSUB 7000
2030 GOSUB 7200
2040 IF MI=1 THEN RESTORE 2550 ELSE RESTORE 2560
2050 GOSUB 7200
2060 GOSUB 7500
2070 IF VH<UH THEN 2140
2080 '          visualizzazione valori precedenti
2090     UJ=1
2100     UD=IZ : GOSUB 7700
2110     UD=EL : GOSUB 7700
2120     UD$=IT$ : GOSUB 7725
2130 '          assegnazione nuovi valori
2140 GOSUB 8000
2150 UJ=1
2160 GOSUB 7650 : IZ=UD
2170 GOSUB 7650 : EL=UD
2180 GOSUB 7650 : IT$=UD$
2190 IF VH<UH THEN VH=UH
2200 '
2210 RETURN
2490 '
2500 DATA 3
2510 DATA 2,26,2,5,26,7,10,16,60
2520 DATA 4
2530 DATA 2,2,numero di campate 5,2 modulo di elasticità 10 2 intestazione
2540 DATA 25,76,pag 2
2550 DATA 1,5,34,kg/cm2
2560 DATA 1 5,34,N/mm2
2980 '
2990 '          TERZA PAGINA - dati geometrici
3000 CLS
3010 RESTORE 3500
3020 UT=IZ
3030 GOSUB 7100
3040 GOSUB 7200
3050 GOSUB 7300
3060 UA=1
3070 GOSUB 7400
3080 GOSUB 7500
3090 '          visualizzazione valori precedenti o di default
3100 UJ=1
3110 FOR I=1 TO IZ
```

```
3120      IF I>IG THEN 3180
3130          UD=DL(I) : GOSUB 7700
3140          UD=B1(I) : GOSUB 7700
3150          UD=H1(I) : GOSUB 7700
3160          UD=B2(I) : GOSUB 7700
3170          UD=H2(I) : GOSUB 7700
3180 NEXT I
3190 '              assegnazione nuovi valori
3200 GOSUB 8000
3210 UJ=1
3220 FOR I=1 TO IZ
3230      GOSUB 7650 : DL(I)=UD
3240      GOSUB 7650 : B1(I)=UD
3250      GOSUB 7650 : H1(I)=UD
3260      GOSUB 7650 : B2(I)=UD
3270      GOSUB 7650 : H2(I)=UD
3280 NEXT I
3290 IF VH<UH THEN VH=UH
3300 IF IG<IZ THEN IG=IZ
3310 '
3320 RETURN
3490 '
3500 DATA 5,5
3510 DATA 15,7,32,5,45,5,58 5,71 5
3520 DATA 8
3530 DATA 1,26,D A T I G E O M E T R I C I
3540 DATA 3,2,campata,3,16 luce 3 33 B1 3,46,H1 3 59 B2,3,72,H2 3550 DATA 25,76,pag 3
3560 DATA 5,23,m          cm          cm          cm          cm
3570 DATA 5,5
3980 '
3990 '              QUARTA PAGINA - carichi distribuiti
4000 IF UU=1 THEN CLS
4010 RESTORE 4500
4020 UT=IZ
4030 GOSUB 7100
4040 GOSUB 7200
4050 UA=1
4060 GOSUB 7400
4070 IF MI=1 THEN RESTORE 4570 ELSE RESTORE 4580
4080 GOSUB 7300
4090 GOSUB 7500
4100              visualizzazione valori precedenti o di default
4110 UJ=1
4120 FOR I=1 TO IZ
4130      IF I>IQ THEN 4150
4140          UD=Q(I) : GOSUB 7700
4150 NEXT I
4160 '            assegnazione nuovi valori
4170 GOSUB 8000
4180 UJ=1
4190 FOR I=1 TO IZ
4200      GOSUB 7650 : Q(I)=UD
4210 NEXT I
4220 IF VH<UH THEN VH=UH
```

```
4230 IF IQ<IZ THEN IQ=IZ
4240 '
4250 RETURN
4490 '
4500 DATA 1,6
4510 DATA 16,6
4520 DATA 5
4530 DATA 1,33, C A R I C H I
4540 DATA 3,2,campata,3,14,carico uniforme 25 35,pag 4
4550 DATA 4,12,"+ se verso il basso"
4560 DATA 6 5
4570 DATA 6,23,t/m
4580 DATA 6,23 kN/m
4980 '
4990 '              QUINTA PAGINA - momenti concentrati sugli appoggi
5000 RESTORE 5500
5010 UT=IZ+1
5020 GOSUB 7100
5030 GOSUB 7200
5040 UA=1
5050 GOSUB 7400
5060 IF MI=1 THEN RESTORE 5560 ELSE RESTORE 5570
5070 GOSUB 7300
5080 GOSUB 7500
5090              visualizzazione valori precedenti o di default
5100 UJ=1
5110 FOR I=1 TO IZ+1
5120      IF I>IM THEN 5140
5130            UD=MN(I) : GOSUB 7700
5140 NEXT I
5150 '            assegnazione nuovi valori
5160 GOSUB 8000
5170 UJ=1
5180 FOR I=1 TO IZ+1
5190      GOSUB 7650 : MN(I)=UD
5200 NEXT I
5210 IF VH<UH THEN VH=UH
5220 IF IM<IZ+1 THEN IM=IZ+1
5230 '
5240 RETURN
5490 '
5500 DATA 1,6
5510 DATA 57 6
5520 DATA 4
5530 DATA 3 42,appoggio,3,52,momento concentrato,25,75,pag 5
5540 DATA 4,56,"+ se orario"
5550 DATA 6 46
5560 DATA 6,64,tm
5570 DATA 6 64 kNm
5980 '
5985              CONTROLLO DEL VALORE DEI CAMPI
5990
6000 ON UH GOTO 6100 6200 6300 6600 6600
6090 '
```

```
6095  ----  pag.1
6100 IF UJ<> 1 AND UJ<> 3 THEN 6115
6105       GOSUB 7600
6110       IF UD$ <> "SI" AND UD$ <> "NO  THEN GOSUB 6700
6115 GOTO 6600
6190 ' '
6195 ' ----  pag.2
6200 ON UJ GOTO 6220,6240,6600
6215 '
6220 GOSUB 7600
6225 IF UD<= 0 OR UD>16 THEN GOSUB 6750
6230 GOTO 6600
6235 '
6240 GOSUB 7600
6245 IF UD<= 0 THEN GOSUB 6750
6250 GOTO 6600
6290 '
6295  ----  pag.3
6300 GOSUB 7600
6305 ON (UJ-1) MOD 5+1 GOTO 6320,6320,6320,6340,6340
6315 '
6320 IF UD<= 0 THEN GOSUB 6750
6325 GOTO 6600
6335 '
6340 IF UD<0 THEN GOSUB 6750
6345 GOTO 6600
6595 '
6600 RETURN
6695 '
6700 LOCATE 25,2 : PRINT "Devi rispondere SI o NO"
6705 GOTO 6755
6745 '
6750 LOCATE 25,2 : PRINT "Valore non ammissibile "
6755 BEEP
6760 FOR UV=1 TO 1500 : NEXT UV
6765 LOCATE 25,2 : PRINT "
6770 UF=0
6775 RETURN
```

Vanno quì inseriti i sottoprogrammi per la gestione della maschera (linee 6980-9910), già listati nel capitolo 6. Occorre solo modificare il dimensionamento degli array UR, UC, UL, i cui indici devono poter arrivare al valore 90:

```
7800 DIM UF(5),UR(90),UC(90),UL(90)
```

```
9980
9990 '              MEMORIZZAZIONE VALORI SU DISCO
10000 OPEN  O",#1 FO$
```

```
10010 WRITE #1, IT$,IZ,EL
10020 FOR I=1 TO IZ
10030     WRITE #1, DL(I),B1(I),H1(I),B2(I),H2(I),Q(I)
10040 NEXT I
10050 FOR I=1 TO IZ+1
10060     WRITE #1, MN(I)
10070 NEXT I
10080 CLOSE #1
10090 '
10100 RETURN
10960 '
10970             RISOLUZIONE DELLO SCHEMA
10980
10990             adatta EL alle unità di misura di calcolo (t/m2 O kN/m2)
11000 IF MI=1 THEN E=EL*10 ELSE E=EL*1000
11010             calcola il momento d'inerzia e l'indice di rigidezza
11020 FOR I=1 TO IZ
11030     B1=B1(I)/100
11040     H1=H1(I)/100
11050     B2=B2(I)/100
11060     H2=H2(I)/100
11070     A=B1*H1+2*B2*H2
11080     S=B1*H1^2/2+B2*H2^2
11090     X=S/A
11100     ZI(I)=B1*H1^3/3+2*B2*H2^3/3-A*X^2
11110     W(I)=E*ZI(I)/DL(I)
11120 NEXT I
11130             calcola i momenti d'incastro perfetto
11140 FOR I=1 TO IZ
11150     MD(I)=Q(I)*DL(I)^2/12
11160     MS(I)=-(MD(I))
11170 NEXT I
11180             risoluzione dello schema
11190 A(1)=4*W(1)
11200 B(1)=2*W(1)
11210 C(1)=MN(1)-MS(1)
11220 FOR I=2 TO IZ+1
11230     A(I)=4*W(I-1)
11240     C(I)=MN(I)-MD(I-1)
11250     IF I=IZ+1 THEN 11290
11260         A(I)=A(I)+4*W(I)
11270         B(I)=2*W(I)
11280         C(I)=C(I)-MS(I)
11290     A(I)=A(I)-B(I-1)^2/A(I-1)
11300     C(I)=C(I)-B(I-1)*C(I-1)/A(I-1)
11310 NEXT I
11320 C(IZ+1)=C(IZ+1)/A(IZ+1)
11330 FOR I=IZ TO 1 STEP -1
11340     C(I)=(C(I)-B(I)*C(I+1))/A(I)
11350 NEXT I
11360 '
11370             determina momento e taglio agli estremi delle travi
11380 FOR I=1 TO IZ
11390     MS(I)=MS(I)+(4*C(I)+2*C(I+1))*W(I)
```

```
11400     MD(I)=-(MD(I)+(4*C(I+1)+2*C(I))*W(I))
11410     TS(I)=(MD(I)-MS(I))/DL(I)+Q(I)*DL(I)/2
11420     TD(I)=(MD(I)-MS(I))/DL(I)- Q(I)*DL(I)/2
11430 NEXT I
11440 '
11450 RETURN
11960 '
11970             STAMPA DATI
11980
11990             stampa intestazione lavoro
12000 LPRINT TAB (11);IT$
12010 LPRINT
12020 LPRINT
12030             stampa descrizione sintetica del programma
12040 LPRINT "RISOLUZIONE DI UNO SCHEMA DI TRAVE CONTINUA"
12050 LPRINT
12060 LPRINT "Schema geometrico:          aste rettilinee a sezione costante  retta ngolare o a T
12070 LPRINT "Vincoli esterni:            appoggi fissi "
12080 LPRINT " Tipi di carico ammessi:    carichi uniformemente distribuiti sulle travi (positivi"
12090 LPRINT "                                se diretti verso il basso);"
12100 LPRINT "                            momenti concentrati sugli appoggi (positivi se orari)
12110 LPRINT
12120 LPRINT  Convenzione dei segni per i risultati :"
12130 LPRINT "      rotazione:            positiva se oraria;
12140 LPRINT '      momento flettente:    positivo se tende le fibre inferiori;"
12150 LPRINT '      taglio:               positivo se diretto verso il basso, come azione sulla
12160 LPRINT '                                faccia di normale uscente verso destra."
12170 LPRINT
12180 LPRINT
12190 '           stampa dati geometrici ed elastici
12200 LPRINT "DATI GEOMETRICI ED ELASTICI"
12210 LPRINT
12220 LPRINT "campata     luce     larghezza  altezza   sporgenza   spessore momento
12230 LPRINT "                     anima      totale    ala         ala       d'inerzia"
12240 A$="   ##   ### ## m   #### # cm   #### # cm   #### # cm   #### # cm
##.###### m4"
12250 FOR I=1 TO IZ
12260     LPRINT USING A$ ; I,DL(I) B1(I) H1(I) B2(I),H2(I),ZI(I)
12270 NEXT I
12280 LPRINT
12290 IF MI=1 THEN A$="Modulo di elasticità       E = ####### kg/cm2"
12300 IF MI<>1 THEN A$="Modulo di elasticità      E = ###### N/mm2"
12310 LPRINT USING A$ ; EL
12320 LPRINT
12330 LPRINT
12340 LPRINT "CARICHI UNIFORMI";TAB (41); MOMENTI CONCENTRATI"
12350 LPRINT
12360 LPRINT  campata       carico";TAB (41); appoggio       momento"
12370 IF MI=1 THEN A$=" ##      ### ## t/m          ##       ### ## tm"
12380 IF MI<>1 THEN A$=" ##     ####.# kN/m         ##       ####.#kNm"
12390 FOR I=1 TO IZ
12400     LPRINT USING A$ ; I Q(I) I,MN(I)
12410 NEXT I
12420 IF MI=1 THEN A$=                            ##       ### ## tm'
```

```
12430 IF MI<>1 THEN A$= '                                    ##        ####.# kNm'
12440 LPRINT USING A$ ; I,MN(I)
12450 LPRINT
12460 LPRINT
12470 '
12480 RETURN
12960 '
12970             STAMPA RISULTATI
12980
12990             stampa rotazioni
13000 LPRINT " ROTAZIONE DEI NODI"
13010 LPRINT
13020 LPRINT "     nodo          rotazione
13030 FOR I=1 TO IZ+1
13040     LPRINT USING "     ##          +#.###^^^^  ; I,C(I)
13050 NEXT I
13060 LPRINT
13070             stampa caratteristiche di sollecitazione
13080 LPRINT "CARATTERISTICHE DI SOLLECITAZIONE
13090 LPRINT
13100 IF MI=1 THEN A$=          ### ##tm  ELSE A$="          ####.# kNm'
13110 IF MI=1 THEN B$="         ### ##t   ELSE B$=           ####.# kN '
13120 FOR I=1 TO IZ
13130     LPRINT " Campata  ;I
13140     LPRINT
13150     FOR J=1 TO 5
13160         X(J)=J*DL(I)/5
13170         T(J)=TS(I)-Q(I)*X(J)
13180         M(J)=MS(I)+TS(I)*X(J)- Q(I)*X(J)^2/2
13190     NEXT J
13200     LPRINT "ascissa";
13210     LPRINT USING "     ### ## m " ; 0 X(1) X(2) X(3),X(4),X(5)
13220     LPRINT "momento  ;
13230     LPRINT USING A$ ; MS(I),M(1),M(2) M(3) M(4) M(5)
13240     LPRINT "taglio ";
13250     LPRINT USING B$ ; TS(I) T(1) T(2),T(3) T(4),T(5)
13260     LPRINT
13270 NEXT I
13280 '
13290 RETURN
13960 '
13970             VISUALIZZAZIONE DIAGRAMMI
13980
13990             dimensioni reali schermo
14000 SCREEN 2
14005 FX=.212
14010 FY=.16
14020 '           calcola luce massima e lunghezza totale
14030 LM=0
14040 LT=0
14050 FOR I=1 TO IZ
14060     IF LM<DL(I) THEN LM=DL(I)
14070     LT=LT+DL(I)
14080 NEXT I
```

```
14090           calcola momento e taglio massimo
14100 MX=0
14110 TX=0
14120 FOR I=1 TO IZ
14130     IF MX<ABS (MS(I)) THEN MX=ABS (MS(I))
14140     IF MX<ABS (MD(I)) THEN MX=ABS (MD(I))
14150     V=ABS (Q(I)*DL(I)^2/8)
14160     IF MX<V THEN MX=V
14170     IF TX<ABS (TS(I)) THEN TX=ABS (TS(I))
14180     IF TX<ABS (TD(I)) THEN TX=ABS (TD(I))
14190 NEXT I
14200 MM=MX*LT/LM*FY/FX : IF MM<MX THEN MM=MX
14210 TM=2*TX*LT/LM*FY/FX : IF TM<TX THEN TM=TX
14220           definisce la scala del disegno del momento flettente
14230 WINDOW (- 02*LT,-MM)-(1 02*LT,MM)
14240           disegna lo schema geometrico
14250 HA=MM/25
14260 GOSUB 14800
14270           disegna il diagramma del momento
14280 XA=0
14290 LINE -(XA,0),2
14300 FOR I=1 TO IZ
14310     LINE -(XA,-MS(I))
14320     FOR J=1 TO 20
14330         X=DL(I)*J/20
14340         M=MS(I)+TS(I)*X-Q(I)*X^2/2
14350         LINE -(XA+X,-M)
14360     NEXT J
14370     XA=XA+DL(I)
14380 NEXT I
14390 LINE -(XA,0)
14400 FOR V=1 TO 7500 : NEXT V
14410           definisce la scala del disegno del taglio
14420 WINDOW (- 02*LT,-TM)-(1 02*LT,TM)
14430           disegna lo schema geometrico
14440 HA=TM/25
14450 GOSUB 14800
14460           disegna il diagramma del taglio
14470 XA=0
14480 LINE -(XA,0),2
14490 FOR I=1 TO IZ
14500     LINE -(XA,TS(I))
14510     LINE -(XA+DL(I),TD(I))
14520     XA=XA+DL(I)
14530 NEXT I
14540 LINE -(XA,0)
14550 FOR V=1 TO 7500 : NEXT V
14560 '
14570 SCREEN 0
14580 RETURN
14790 '         disegna lo schema geometrico
14800 CLS
14810 LINE (0,0)-(LT,0)
14820 XA=0
```

```
14830 FOR I=0 TO IZ
14840     IF I>0 THEN XA=XA+DL(I)
14850     LINE (XA,0)-(XA-LT/50 -HA)
14860     LINE -(XA+LT/50 -HA)
14870     LINE -(XA 0)
14880 NEXT I
14890 RETURN
19980 '
19990 '          trave continua
19999  A G  22-11-86
```

8.11 - Esempi di controllo.

Per quanta cura si sia messa nel realizzare il programma, è sempre possibile essere incorsi in qualche errore. È pertanto necessario effettuare un certo numero di esecuzioni di prova, controllando l'esattezza dei risultati forniti dall'elaboratore. Si possono utilizzare a tal fine esempi tratti da libri, purchè si sia certi che le ipotesi e l'impostazione del calcolo coincida. Oppure, specie quando il problema non è troppo complesso, si può effettuare il calcolo a mano. Io ho seguito questa seconda via, utilizzando due esempi che erano stati in precedenza preparati per fini didattici e risolti manualmente con molta accuratezza.

Il primo esempio si riferisce ad una trave di un edificio in cemento armato. Essa presenta 5 campate, delle quali quattro emergenti ed una a spessore ed è caricata esclusivamente da carichi distribuiti. Si riportano di seguito dati e risultati ottenuti. I diagrammi delle caratteristiche di sollecitazione sono contenuti nella figura 60

TRAVE 103 - sovraccarico su tutte le campate

RISOLUZIONE DI UNO SCHEMA DI TRAVE CONTINUA

Schema geometrico:	aste rettilinee a sezione costante rettangolare o a T
Vincoli esterni:	appoggi fissi.
Tipi di carico ammessi:	carichi uniformemente distribuiti sulle travi (positivi se diretti verso il basso); momenti concentrati sugli appoggi (positivi se orari)
Convenzione dei segni per i risultati :	
rotazione:	positiva se oraria;
momento flettente:	positivo se tende le fibre inferiori;
taglio:	positivo se diretto verso il basso, come azione sulla faccia di normale uscente verso destra

DATI GEOMETRICI ED ELASTICI

campata	luce	larghezza anima	altezza totale	sporgenza ala	spessore ala	momento d'inerzia
1	5 20 m	30 0 cm	60 0 cm	25 0 cm	22 0 cm	0 008308 m4
2	5 80 m	30 0 cm	60 0 cm	25 0 cm	22 0 cm	0 008308 m4
3	2 70 m	30 0 cm	60 0 cm	25 0 cm	22 0 cm	0 008308 m4
4	4 00 m	100 0 cm	22 0 cm	0 0 cm	0 0 cm	0 000887 m4
5	4 50 m	30 0 cm	60 0 cm	25 0 cm	22 0 cm	0 008308 m4

Modulo di elasticità E = 250000 kg/cm^2

CARICHI UNIFORMI

campata	carico
1	4 97 t/m
2	4 78 t/m
3	3 08 t/m
4	4 58 t/m
5	4 78 t/m

MOMENTI CONCENTRATI

appoggio	momento
1	0 00 tm
2	0 00 tm
3	0 00 tm
4	0 00 tm
5	0 00 tm
1	0 00 tm

ROTAZIONE DEI NODI

nodo	rotazione
1	7 186E−004
2	−3 537E−005
3	−3 446E−004
4	2 772E−004
5	3 539E−004
6	−6 138E−004

CARATTERISTICHE DI SOLLECITAZIONE

Campata 1

ascissa	0	00 m	1	04 m	2	08 m	3	12 m	4	16 m	5	20 m
momento	0	00 tm	7	48 tm	9	58 tm	6	30 tm	−2	35 tm	−16	37 tm
taglio	9	77 t	4	60 t	−0	56 t	−5	73 t	−10	90 t	−16	07 t

Campata 2

ascissa	0	00 m	1	16 m	2	32 m	3	48 m	4	64 m	5	80 m
momento	−16	37 tm	−1	88 tm	6	19 tm	7	82 tm	3	02 tm	−8	21 tm
taglio	15	27 t	9	72 t	4	18 t	−1	36 t	−6	91 t	−12	45 t

Campata 3

ascissa	0	00 m	0	54 m	1	08 m	1	62 m	2	16 m	2	70 m
momento	−8	21 tm	−5	79 tm	−4	27 tm	−3	65 tm	−3	93 tm	−5	10 tm
taglio	5	31 t	3	65 t	1	98 t	0	32 t	−1	34 t	−3	01 t

Campata 4

ascissa	0 00 m	0 80 m	1 60 m	2 40 m	3 20 m	4 00 m
momento	−5 10 tm	0 34 tm	2 85 tm	2 43 tm	−0 92 tm	−7 20 tm
taglio	8 64 t	4 97 t	1 31 t	−2 36 t	−6 02 t	−9 68 t

Campata 5

ascissa	0 00 m	0 90 m	1 80 m	2 70 m	3 60 m	4 50 m
momento	−7 20 tm	1 98 tm	7 30 tm	8 74 tm	6 30 tm	0 00 tm
taglio	12 35 t	8 05 t	3 75 t	−0 55 t	−4 85 t	−9 16 t

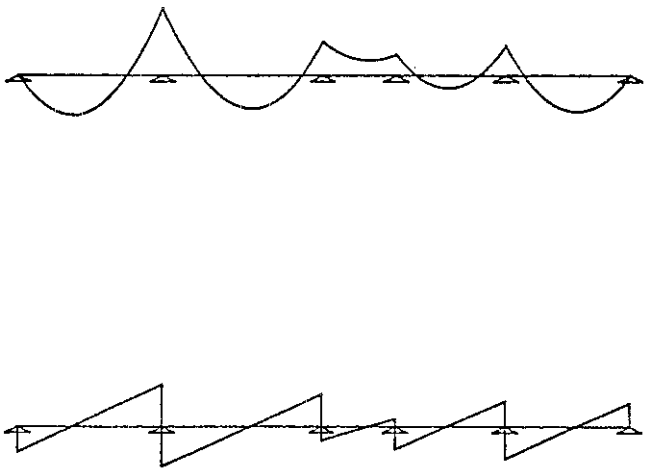


Fig 60

Il secondo esempio si riferisce ad un solaio di un edificio in cemento armato. Essa presenta 2 campate, ovviamente di uguale sezione; poichè esiste uno sbalzo al suo estremo destro, è caricato anche con un momento concentrato. I dati sono stati in questo caso indicati nel sistema di misura internazionale, per verificare anche da questo aspetto la correttezza del programma. Si riportano di seguito dati e risultati ottenuti. I diagrammi delle caratteristiche di sollecitazione sono contenuti nella figura 61.

SOLAIO 152 - sovraccarico sulla prima campata e sullo sbalzo

RISOLUZIONE DI UNO SCHEMA DI TRAVE CONTINUA

Schema geometrico: aste rettilinee a sezione costante rettangolare o a T
Vincoli esterni: appoggi fissi
Tipi di carico ammessi: carichi uniformemente distribuiti sulle travi (positivi se diretti verso il basso);
momenti concentrati sugli appoggi (positivi se orari).

Convenzione dei segni per i risultati :
rotazione: positiva se oraria;
momento flettente: positivo se tende le fibre inferiori;
taglio: positivo se diretto verso il basso, come azione sulla faccia di normale uscente verso destra

DATI GEOMETRICI ED ELASTICI

campata	luce	larghezza anima	altezza totale	sporgenza ala	spessore ala	momento d'inerzia
1	5.00 m	20.0 cm	22.0 cm	40.0 cm	4.0 cm	0.000332 m4
2	6.00 m	20.0 cm	22.0 cm	40.0 cm	4.0 cm	0.000332 m4

Modulo di elasticità E = 25000 N/mm^2

CARICHI UNIFORMI

campata	carico
1	6.9 kN/m
2	4.9 kN/m

MOMENTI CONCENTRATI

appoggio	momento
1	0.0 kNm
2	0.0 kNm
3	11.2 kNm

ROTAZIONE DEI NODI

nodo	rotazione
1	2.446E-003
2	-5.603E-004
3	-3.528E-004

CARATTERISTICHE DI SOLLECITAZIONE

Campata 1

ascissa	0.00 m	1.00 m	2.00 m	3.00 m	4.00 m	5.00 m
momento	0.0 kNm	10.0 kNm	13.2 kNm	9.4 kNm	-1.2 kNm	-18.8 kNm
taglio	13.5 kN	6.6 kN	-0.3 kN	-7.2 kN	-14.1 kN	-21.0 kN

Campata 2

ascissa	0.00 m	1.20 m	2.40 m	3.60 m	4.80 m	6.00 m
momento	-18.8 kNm	-3.1 kNm	5.4 kNm	6.9 kNm	1.4 kNm	-11.2 kNm
taglio	16.0 kN	10.1 kN	4.2 kN	-1.7 kN	-7.6 kN	-13.4 kN

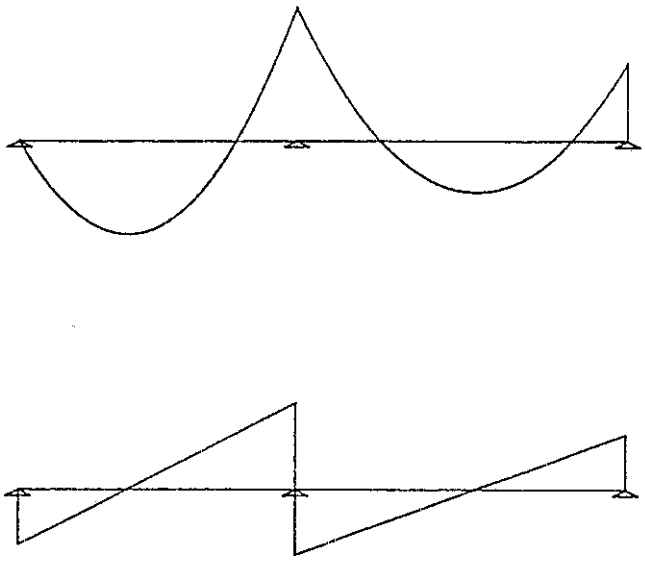


Fig 61

APPENDICE

Nel dischetto allegato al testo sono contenuti tutti i programmi in esso esemplificativamente illustrati. Si riporta nel seguito per ciascuno di essi il nome, il numero del paragrafo in cui è analizzato ed una sintetica descrizione del contenuto.

nome	paragrafo	contenuto
EQUAZ1	4.6 6	risoluzione equazione primo grado
SISTEQ	4 6 6	risoluzione sistema lineare di 2 equazioni in 2 incognite
VERFLE1	4 6 6	verifica a flessione per sezione rettangolare in c.a. a doppia armatura
EQUAZ2	4 8.5	risoluzione equazione di secondo grado.
TABFUNZ1	4.9 1	tabellazione di una funzione
NULFUNZ	4 9 1	punto di nullo di una funzione
LAUREA	4 9 2	punteggio base per laurea
TRASPMAT	4 9.2	trasposizione matrice
PRODMAT	4 9.2	prodotto di matrici
TABFUNZ2	4 11 3	tabellazione di una funzione (con formattazione 1)
TABFUNZ3	4.11.3	tabellazione di una funzione (con formattazione 2)
ELENCHI	5 8	fusione di due elenchi ordinati
CORSO-A	5 8	file dati per fusione elenchi
CORSO-B	5 8	file dati per fusione elenchi
COLOR	6 3	effetto dell'istruzione "COLOR"
TASTI	6.5	codice dei tasti
MASCHERA	6 6,6.7,6 9	sottoprogrammi per ingresso dati con maschera.
VERFLE2	6 10	verifica a flessione per sezione rettangolare in c.a. a doppia armatura.
COORDIN	6 11	tabella di dati (coordinate nello spazio).
XYZ	6 11	file dati coordinate nello spazio
ISTOGR	7 4 *	disegno di un istogramma.
DIAGFUN	7 5 *	diagramma di una funzione
SCALA	7 6	sottoprogramma per la definizione di scala per rappresentazione monometrica
SCHEMA	7 7 *	schema di un telaio piano
SCRITTE	7 8	inserimento di scritte nella pagina grafica
TRAVECON.	8	risoluzione di una trave continua
TRAVE	8 11	dati per schema di trave
SOLAIO	8 11	dati per schema di solaio

* effettuare "KEY OFF" e "SCREEN n" prima di mandare in esecuzione il programma

BIBLIOGRAFIA.

1. B. Fadini, C. Savy, *Calcolatori elettronici*, Istituto Editoriale del Mezzogiorno, 1971.

2. *Enciclopedia della scienza e della tecnica, voci "Calcolatore" ed altre*, Mondadori, 1980

3. L. Sansone, *Appunti di tecniche di programmazione*, CUEN, 1975.

4. M. Capurso, *Introduzione al calcolo automatico delle strutture, cap. 1*, Cremonese, 1977

5. D. Clarke, *Computer aided structural design, cap. 1*, John Wiley & sons, 1978

6. H. Peckham, W. Ellis, E. Lodi, *Programmazione strutturata in Basic*, McGraw-Hill, 1986.

7. *HP-85 Owner's manual and programming guide*, Hewlett-Packard, 1979.

8. *HP-85 Advanced Programming ROM manual*, Hewlett-Packard.

9. *HP-86B Manuale introduttivo*, Hewlett-Packard, 1982

10. *HP-87 Operating and BASIC programming manual*, Hewlett-Packard, 1981

11. *Unità a disco flessibile Manuale d'uso*, Hewlett-Packard, 1980

12. V. Wolverton, *Il libro dell'MS-DOS*, Mondadori, 1985

13. P. Hoffman, T. Nicoloff, *Il manuale MS-DOS*, McGraw-Hill, 1986

14. *BASIC-80 Reference Manual*, Microsoft, 1979

15. *GWBASIC manual*, Ericsson, 1984.

16. W. Ettlin, G. Solberg, *Il GWBASIC per Personal Computer Olivetti*, McGraw-Hill, 1986

17. G. Bianchi, *Impariamo a programmare con M24*, Zanichelli, 1986.

18. A. Van Dam, *Software per la grafica*, Le Scienze, novembre 1984

19 J. Scott, *Computergraphia*, Gruppo Editoriale Jackson, 1985.

INDICE ANALITICO