Spediz. in abbonamento postale GR. II/70 L. 2.000

(...)

33 CORSO PRATICO COL COMPUTER

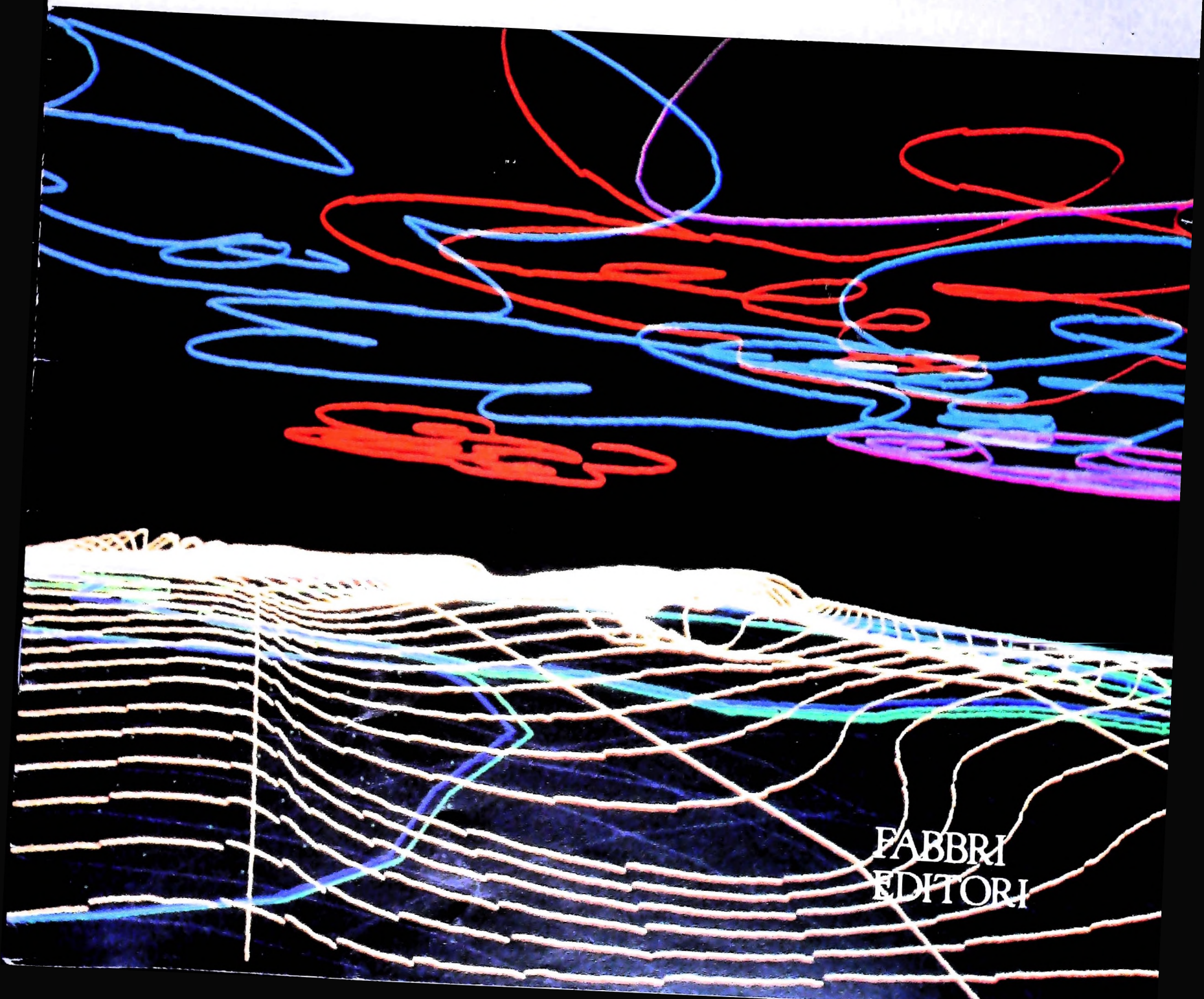
421859

è una iniziativa
FABBRI EDITORI
in collaborazione con
BANCO DI ROMA
e **OLIVETTI**

F4 F5 F6 F7

BATTERY LOW

diretto da GIANNI DEGLI ANTONI



FABBRI EDITORI

Direttore dell'opera
GIANNI DEGLI ANTONI

Comitato Scientifico
GIANNI DEGLI ANTONI
Docente di Teoria dell'Informazione, Direttore dell'Istituto di Cibernetica dell'Università degli Studi di Milano

UMBERTO ECO
Ordinario di Semiotica presso l'Università di Bologna

MARIO ITALIANI
Ordinario di Teoria e Applicazione delle Macchine Calcolatrici presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

MARCO MAIOCCHI
Professore Incaricato di Teoria e Applicazione delle Macchine Calcolatrici presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

DANIELE MARINI
Ricercatore universitario presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

Curatori di rubriche
ADRIANO DE LUCA (Professore di Architettura dei Calcolatori all'Università Autonoma Metropolitana di Città del Messico), GOFFREDO HAUS, MARCO MAIOCCHI, DANIELE MARINI, GIANCARLO MAURI, CLAUDIO PARMELLI, ENNIO PROVERA

Testi
CLAUDIO PARMELLI, ETTORE DECIO, Eidos (BRUNO MOTTA, CARMINE STRAGAPEDE), Etnoteam (ADRIANA BICEGO)

Tavole
Logical Studio Communication
Il Corso di Programmazione e BASIC è stato realizzato da Etnoteam S.p.A., Milano
Computergrafica è stato realizzato da Eidos, S.c.r.L, Milano
Usare il Computer è stato realizzato in collaborazione con PARSEC S.N.C. - Milano

Direttore Editoriale
ORSOLA FENGHI

Redazione
CARLA VERGANI
LOGICAL STUDIO COMMUNICATION

Art Director
CESARE BARONI

Impaginazione
BRUNO DE CHECCHI
PAOLA ROZZA

Programmazione Editoriale
ROSANNA ZERBARINI
GIOVANNA BREGGÈ

Segretarie di Redazione
RENATA FRIGOLI
LUCIA MONTANARI

IL LINGUAGGIO PASCAL (II)

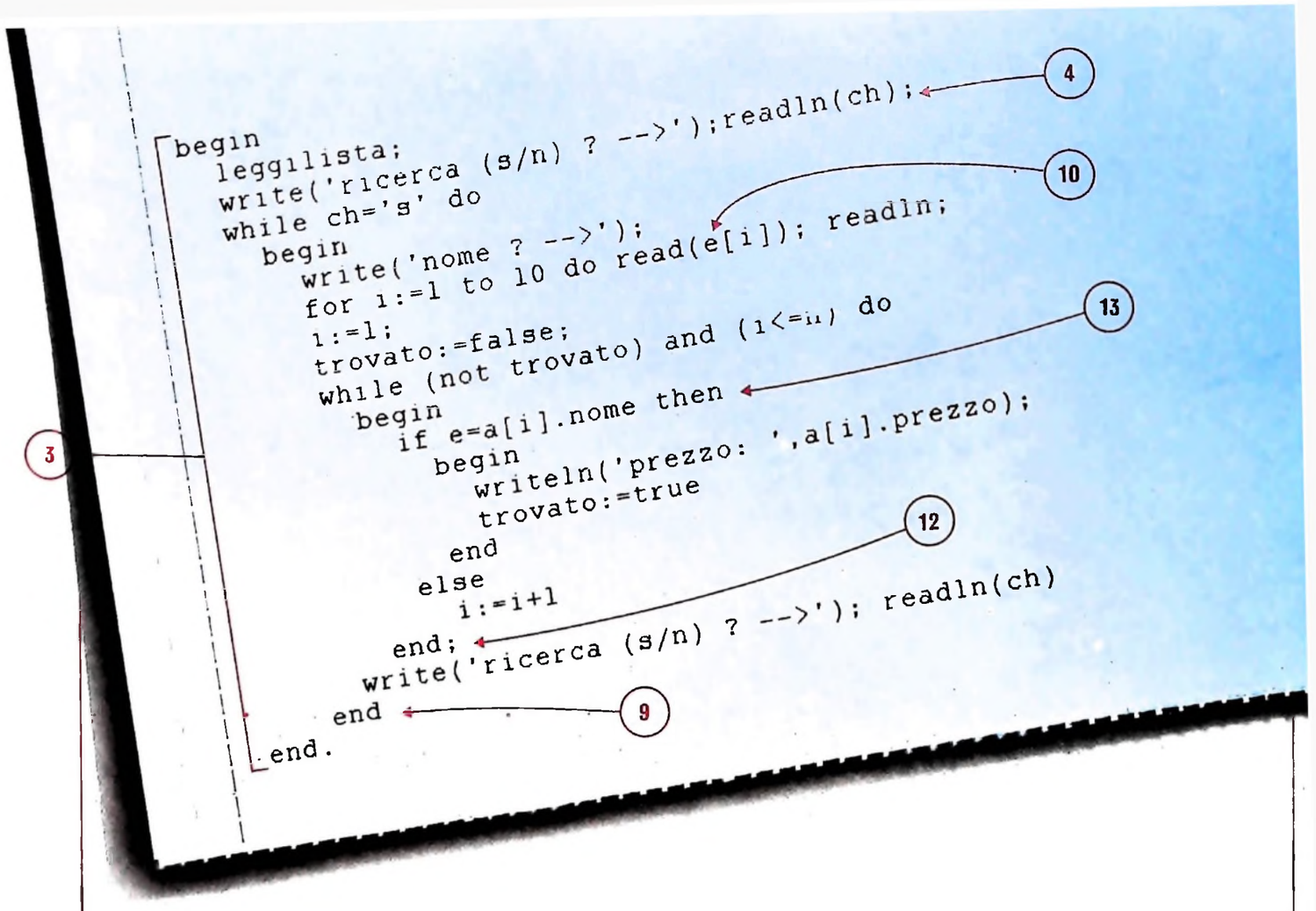
Continuiamo la nostra carrellata generale sul linguaggio Pascal, osservando un esempio di programma.

Si tratta dello stesso programma già presentato nel caso del linguaggio COBOL, cioè di un'applicazione gestionale (il Pascal si propone come linguaggio "general purpose", cioè adatto a tutte le applicazioni, sia di calcolo scientifico tecnico, sia per la costruzione di software di "base", sia per applicazioni gestionali), che legge un listino prezzi e permette in seguito la ricerca del prezzo di un prodotto, partendo dal suo nome.
Anche in questo caso il programma è molto rozzo e semplificato, ma serve al nostro scopo, quello di mostrare le caratte-

Esempio di programma, in linguaggio Pascal, per una gestione commerciale: con tale programma è possibile individuare immediatamente il prezzo di un prodotto partendo dal nome.

```
program search(input,output);
type prodotto=record
                nome:array[1..10] of char;
                prezzo:integer
              end;
     listino=array[1..100] of prodotto;
var a:listino;
    i,n:integer;
    ch:char;
    e:array[1..10]of char;
    trovato:boolean;
procedure leggilista;
var i,j:integer;
begin
  write('quanti elementi ? -->');
  readln(n);
  for i:=1 to n do
    begin
      for j:=1 to 10 do
        read(a[i].nome[j]);
      read(a[i].prezzo);readln
    end
end;
```

▶ segue



ristiche del linguaggio.
Il programma è illustrato qui sotto e nella pagina seguente.
L'ordine in cui le varie informazioni sono presentate non è quello in cui sono state pensate e scritte: infatti il linguaggio richiede che ogni simbolo (tipo, variabile, sottoprogramma ecc.) sia stato definito prima di essere usato (in genere, nei linguaggi di programmazione non è così: si pensi al GOTO BASIC, che può saltare a un'istruzione successiva, cioè a un numero di linea che non è ancora comparso nel programma): questa semplice limitazione permette di avere compilatori molto efficienti, ma presuppone di imparare a leggere i programmi osservandone la struttura.
Dunque, il nostro programma presenta:

- un insieme di definizioni e dichiarazioni (blocco 1)
- una PROCEDURE, cioè un sottoprogramma (blocco 2)
- la parte di istruzioni del programma (blocco 3).

Nel programma compaiono diversi elementi lessicali:
- parole riservate (come TYPE, VAR, PROCEDURE, IF, WHILE ecc.), cioè parole che hanno un preciso significato per il linguaggio, e che non possono essere usate dall'utente con un significato diverso
- parole definite dall'utente (come i nomi dei tipi, delle procedure, delle variabili)
- vari simboli aritmetici (come +, —, x ecc.) o di separazione (come il ";"), o altri ancora che impareremo a conoscere con più dettaglio.

Le varie istruzioni terminano con un ";" oppure con la parola riservata END; ciò permette di scrivere le istruzioni in qualunque posizione: all'inizio della riga, oppure usando una tabulazione che riveli la struttura del programma (come nel blocco 3) (si parla in tali casi di "indentazione", con una trasposizione del termine inglese "indentation"), oppure una dietro l'altra (come nel punto 4) o spezzata su più righe.

Iniziamo la lettura

Il programma presenta inizialmente una dichiarazione di un tipo di dati di nome "prodotto" (punto 5); cioè il programmatore ha scelto di definire un "modello" di struttura di dati per rappresentare un "prodotto"; si tratta di un record (cioè un insieme di elementi di tipo diverso, ma strettamente correlati dal punto di vista logico), i cui componenti sono:
- "nome", cioè il nome del prodotto, definito come array di 10 caratteri, cioè una stringa di 10 caratteri;
- "prezzo", cioè il prezzo del prodotto, definito come intero.

Inoltre viene definito un altro tipo denominato "listino" (punto 6), che è costituito da un array di 100 elementi, ciascuno dei quali ha la struttura di un tipo di prodotto.

Si noti l'estrema vicinanza al linguaggio naturale nell'espressione di questi concetti e all'eccezionale rilevanza della scelta dei nomi per facilitare la lettura del programma.
In questo modo, infatti, un "listino" è considerato un elenco ordinato di 100 "prodotti", per ciascuno dei quali è specificato il "nome" e il "prezzo".
La dichiarazione della variabile "a" causerà quindi l'allocazione di un array di 100 elementi, ciascuno dei quali avrà lo spazio per contenere 10 caratteri e un intero.
Leggiamo ora la parte di algoritmo (blocco 3), che inizia con la parola chiave BEGIN e termina con una END. All'interno di questa "coppia di parentesi" si svolge il programma:

- il richiamo di un sottoprogramma "leggilista"; senza andare a vedere con dettaglio il sottoprogramma in questo istante, possiamo senz'altro supporre, sulla base del nome, che si tratti di una procedura che legge tutti i nomi e i prezzi dei prodotti, e li inserisce nella tabella a;
- quindi troviamo un'istruzione di scrittura ("write" sta per "scrivi una riga") che chiede se si vuole effettuare o meno una ricerca, e indica che ci si attende una risposta del tipo "s" o "n"; subito dopo l'istruzione "readln" legge il carattere fornito ("s" o "n") nella variabile "ch" che è stata dichiarata (punto 8) "tipo carattere";
- quindi incontriamo una struttura di controllo WHILE che specifica che fintantoché la variabile ch ha valore "s" devono essere eseguite le istruzioni "racchiuse" tra il BEGIN che segue e la END corrispondente (punto 9); si noti che essa è preceduta da una nuova domanda sul dover effettuare nuovamente una ricerca, che permette all'utente di richiedere o meno l'iterazione della ricerca;
- la parte da iterare comprende:
  - la stampa della richiesta del nome del prodotto, seguita da un'iterazione enumerativa (punto 10), che fa variare una variabile "i" da 1 a 10, leggendo ogni volta un carattere del nome, che viene inserito nella variabile "e" (si notino le dichiarazioni di "i" e di "e" al punto 11); il seguente richiamo "readln" permette di andare a capo dopo aver inserito il nome dell'elemento; si osservi che la parte da iterare, essendo costituita da una sola istruzione, non è stata racchiusa tra BEGIN e END: ciò è necessario solo nel caso in cui più istruzioni debbano essere iterate;
  - quindi inizia la ricerca con l'inizializzazione della variabile "i" a 1, della variabile booleana "trovato" a "false" e con l'iterazione di un insieme di istruzioni fintantoché "i" si mantiene entro i limiti della tabella (si noti la variabile "n", che è definita in 11, e che deve evidentemente essere posizionata nell'ambito della procedura "leggilista") e nel contempo la variabile "trovato" assume il valore "false"; il blocco che deve essere iterato è quello racchiuso tra il BEGIN che segue e la END corrispondente (punto 12);
  - nell'ambito di questo blocco viene effettuato il confronto con il nome cercato, ed eventualmente visualizzato il prezzo corrispondente e posizionata la variabile "trovato"; si noti che il confronto dei nomi (punto 13) non è effettuato controllando carattere per carattere, ma direttamente l'array di 10 caratteri come un unico campo, cioè con un livello di astrazione superiore.

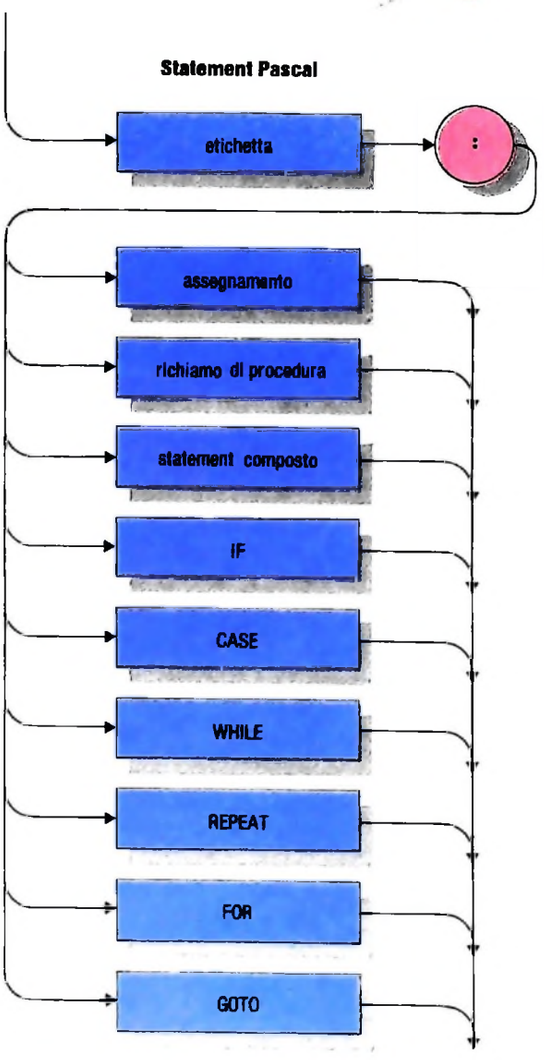
Passiamo ora a esaminare la procedura "leggilista". Essa chiede il numero di elementi da inserire in "a" (punto 14) e quindi ne carica nome e prezzo. Si notino le variabili locali alla procedura (punto 15) e il fatto che le dichiarazioni nel programma principale di "a" e "n" ne garantiscono l'accessibilità del sottoprogramma.

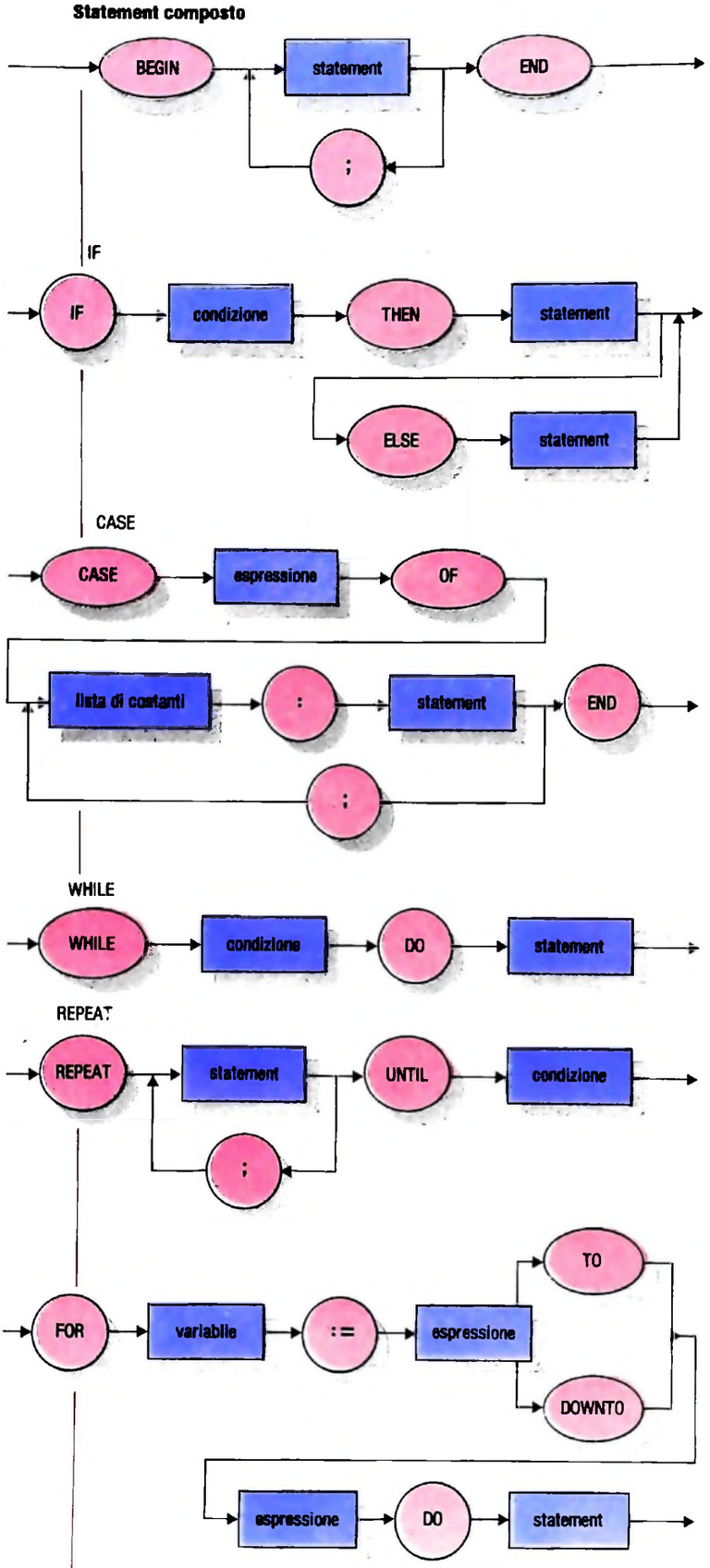
Le strutture di controllo in Pascal

Il Pascal mette dunque a disposizione strutture di controllo tipiche della programmazione strutturata e in particolare:

- IF..THEN..ELSE..
- CASE..
- WHILE..DO..
- REPEAT..UNTIL..
- FOR..

Le prime due riguardano le selezioni, le seconde sono due iterazioni e l'ultima è un'iterazione enumerativa.





Le varie caratteristiche delle strutture di controllo sono ben evidenziate dalla sintassi del linguaggio, di cui riportiamo un frammento parziale, mediante il formalismo delle "carte sintattiche".

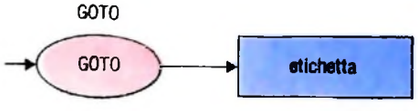
Da esse si vede come uno "statement" Pascal (figura della pagina precedente) è un'istruzione o un insieme di istruzioni connesse da una struttura di controllo: una possibile istruzione è l'assegnamento o il richiamo di una procedura; aggregati di istruzioni sono lo "statement composto" (figura a lato), che corrisponde alla struttura di controllo della "sequenza", o le strutture di controllo precedentemente indicate.

La sintassi della singola struttura di controllo è esplicitata, cosicché è possibile vedere, per esempio, che un IF.. THEN.. ELSE.. prevede la clausola ELSE come opzionale e seleziona l'esecuzione di uno e un solo "statement" (che a sua volta può essere una sequenza, un IF.. THEN.. ELSE, o altra struttura più complessa).

Si noti la forma dell'iterazione enumerativa FOR, che prevede due clausole differenti TO e DOWNTO: la prima presuppone un "passo" di iterazione pari a 1, la seconda prevede un passo "all'indietro" pari a -1; non sono possibili in Pascal iterazioni enumerative con passi differenti.

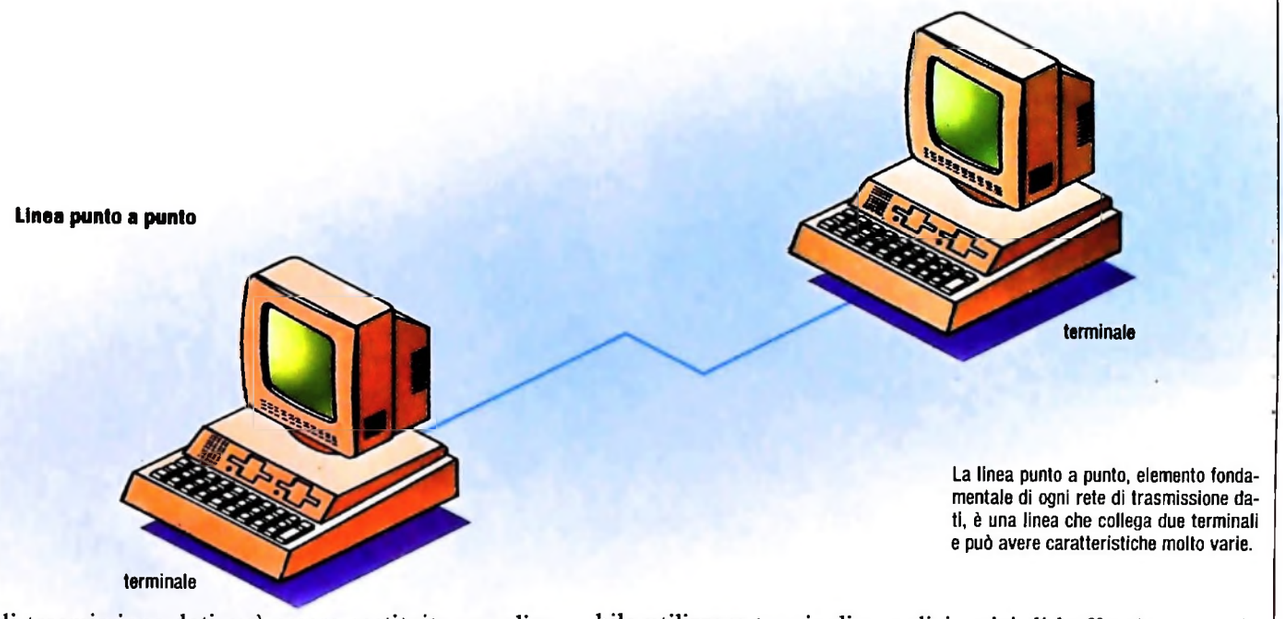
Si osservi la disponibilità di un'istruzione GOTO, che permette di saltare un'"etichetta" che sia stata anteposta a uno "statement" (come si vede dalla sintassi, ciascun "statement" può essere eventualmente preceduto da un'"etichetta", cioè una costante numerica, che la identifichi per essere destinazione di salti); ciò non è in contrasto con i principi della programmazione strutturata, ma è piuttosto legato alla necessità di avere uno strumento potente per risolvere casi gravi: tale istruzione viene infatti generalmente usata per "abbandonare" drasticamente un programma (o un sottoprogramma) a fronte di condizioni anomale trovate durante l'esecuzione e non recuperabili (come per esempio dati inconsistenti forniti al programma o istruzioni che richiedono una lettura o scrittura su file, senza che questi siano disponibili, e così via).

Si osservi, poi, che l'istruzione GOTO non è affatto di semplice esecuzione nell'ambito di un programma Pascal: infatti, supponendo di abbandonare un sottoprogramma per saltare a un'etichetta del programma principale, mentre una siffatta istruzione risulta in un linguaggio come il BASIC un semplice trasferimento del controllo, in Pascal si deve avere l'effetto di mantenere congruenza tra gli ambienti delle variabili, deallocando quella parte di STACK che conteneva le variabili locali del sottoprogramma che si sta abbandonando.



ELEMENTI DELLE RETI E LORO CONFIGURAZIONI

Un nuovo modo di comunicare nato dall'integrazione delle tecniche di telecomunicazione, dell'informatica e delle banche dati.

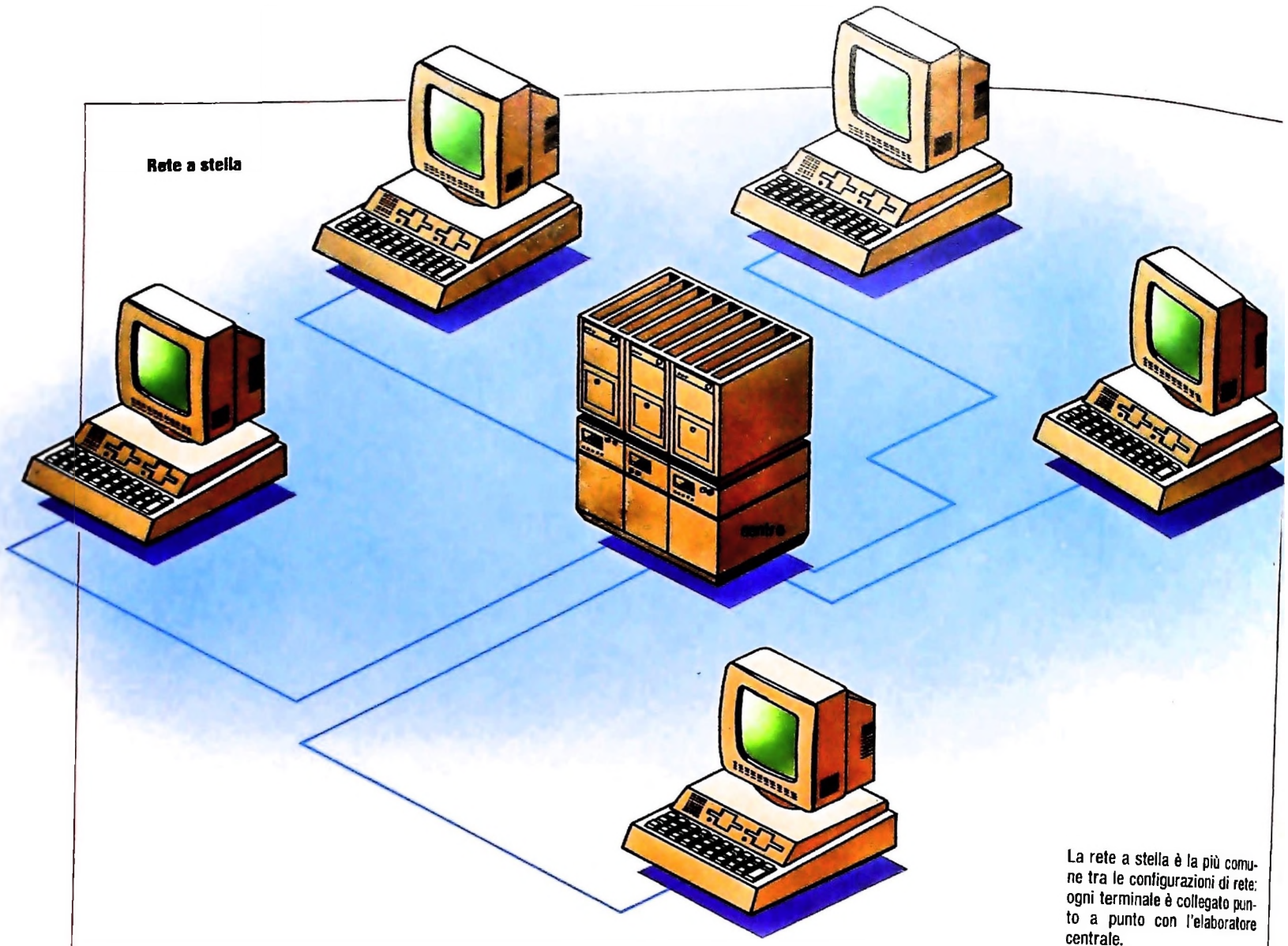


La linea punto a punto, elemento fondamentale di ogni rete di trasmissione dati, è una linea che collega due terminali e può avere caratteristiche molto varie.

Una rete di trasmissione dati può essere costituita semplicemente da terminali, linee ed elaboratori, oppure può essere articolata in un sistema molto complesso.

Linee per la trasmissione dati

A) Linee punto a punto
La linea *punto a punto* che, qualunque sia la sua lunghezza, collega due terminali, è l'elemento fondamentale della rete di trasmissione dati o comunicazione (figura sopra).
Tale linea può essere a una via, half-duplex o full-duplex, e può operare in modo sincrono o asincrono. La configurazione di rete più comune è la *rete a stella* (figura di pagina 514) che è caratterizzata dal fatto che ogni terminale è collegato punto a punto con l'elaboratore centrale.
B) Linee multidrop (o multipoint)
Dato il costo abbastanza elevato delle linee, si tenta di ottimizzare il modo in cui esse vengono impiegate.
La figura di pagina 515 presenta una configurazione molto frequente che impiega *linee multidrop*. Tale linea, detta anche *multipoint*, è una linea di trasmissione dati alla quale sono collegati due o più terminali.
Sulle linee a bassa velocità di questa configurazione è possibile utilizzare terminali semplici, privi di buffer (o memoria di transito), ma con tale configurazione si preferisce più spesso ricorrere a linee relativamente veloci e impiegare terminali bufferizzati. Ciò è dovuto al fatto che il terminale bufferizzato impiega in modo efficiente la linea di trasmissione durante la trasmissione stessa e non impegna la linea mentre l'operatore prepara il messaggio; ciò consente di ripartire la capacità della linea tra i diversi terminali.
Esaminando la figura di pagina 515, si può notare che non deve verificarsi il caso in cui due o più terminali trasmettano contemporaneamente, perché i dati spediti si incontrerebbero divenendo indecifrabili.
Le linee multidrop sono particolarmente adatte per quelle applicazioni in cui ogni terminale trasmette in modo intermittente e non deve utilizzare costantemente la linea. In questo caso si riesce a fare un uso delle linee della rete più efficiente di quanto non si riuscirebbe invece se ogni terminale fosse collegato all'elaboratore centrale con linee punto a punto. Infatti, in questa situazione, tali linee rimarrebbero inutilizzate per la maggior parte del tempo con un costo troppo elevato.
Il numero di terminali che possono essere collegati a una linea multidrop può variare considerevolmente da sistema a sistema ed è determinato da alcuni fattori quali: limitazioni



La rete a stella è la più comune tra le configurazioni di rete: ogni terminale è collegato punto a punto con l'elaboratore centrale.

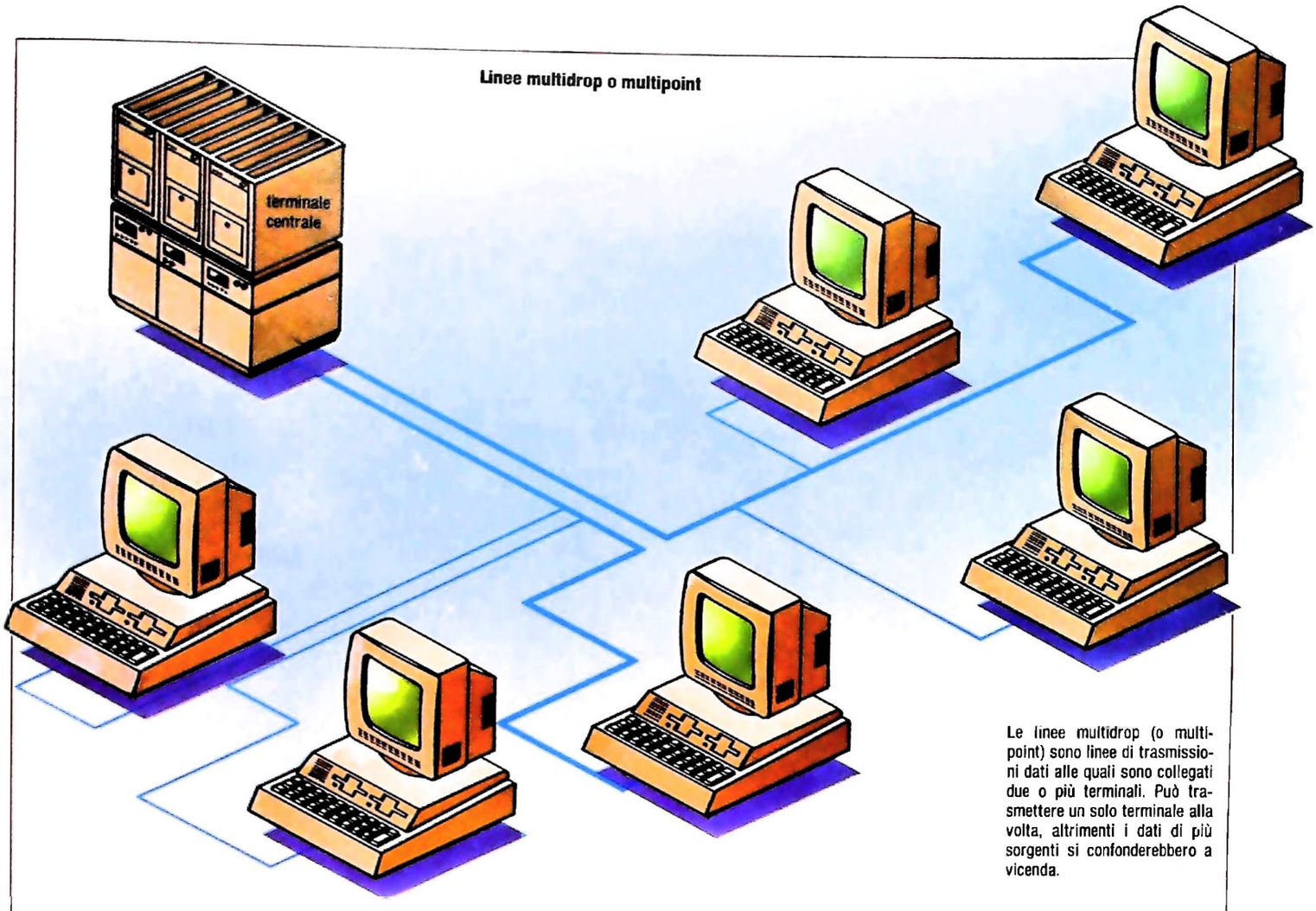
intrinseche dell'hardware e del software impiegati; quantità di traffico generata dai terminali (calcolata in base alla frequenza con cui sono generati i messaggi e alla loro lunghezza); velocità della linea ed eventuali limitazioni imposte dall'azienda telefonica che fornisce la linea di trasmissione.

Reti commutate

In molti casi si può verificare che i terminali debbano trasmettere ogni giorno dati per un periodo di tempo relativamente breve. In tali situazioni non è economicamente conveniente dedicare al terminale una linea privatizzata visto che si impiega la linea per una frazione così piccola del tempo disponibile ed è preferibile far ricorso alle *reti commutate*.
Queste consentono di stabilire, a richiesta, un collegamento punto a punto tra due terminali e di mantenerlo per tutto il tempo desiderato. Come regola generale, il costo della linea viene addebitato solo per la durata della chiamata e per il volume dei dati trasmessi.
Vi sono quattro tipi fondamentali di reti commutate:
— reti telefoniche
— reti telex/TWX
— reti a commutazione di pacchetto
— reti digitali specializzate.
Anche se nei dettagli la gestione delle varie linee è diversa, per stabilire un collegamento tra terminali si segue una serie di operazioni assai simili a quelle svolte per effettuare una chiamata telefonica. Per usare la rete telefonica, si alza il microtelefono, si forma il numero: a questo punto la rete telefonica stabilisce il collegamento automaticamente tra il telefono chiamante e quello ricevente e, alla fine della conversazione, si libera la linea riponendo il microtelefono. A questo punto è possibile, volendo, stabilire un altro collegamento punto a punto con un altro telefono qualsiasi della rete.
La maggior parte dei paesi del mondo ha reti telefoniche capillari e ben strutturate che sono tra loro interconnesse; formando reti internazionali è possibile quindi stabilire virtualmente un collegamento punto a punto tra due telefoni qualsiasi sul nostro pianeta.

La commutazione di pacchetto

Nei sistemi di commutazione tradizionali alla fase di indirizzamento, che si svolge in linguaggio numerico, segue la fase di conversazione che usa linee fisiche che vengono assegnate in modo rigido ai corrispondenti dal centro di smistamento.



Le linee multidrop (o multipoint) sono linee di trasmissioni dati alle quali sono collegati due o più terminali. Può trasmettere un solo terminale alla volta, altrimenti i dati di più sorgenti si confonderebbero a vicenda.

Questo si rende necessario per varie ragioni fra cui pesa in modo rilevante il fatto che la conversazione avviene in linguaggio analogico e quindi necessita di appropriate vie su cui poter transitare.

Nel caso di messaggi digitali, come quelli della trasmissione dati, essendo il messaggio intrinsecamente omogeneo nella parte di indirizzamento e nella parte di conversazione vera e propria, non è più rigorosamente necessario assegnare delle apposite vie agli utenti durante la fase di conversazione. Il centro di smistamento può prendere in consegna il messaggio con relativo indirizzo e provvedere direttamente all'inoltro, usando le vie che di volta in volta ritiene più opportune; in questo modo si evita di creare collegamenti rigidi fra gli utenti.

Nasce così un modo di gestione dei messaggi completamente diverso. Gli utenti non sono più connessi in forma rigida con dei percorsi su cui si scambiano le informazioni; il compito del centro di smistamento non è più quello di creare una via di connessione fra i due corrispondenti, ma semplicemente quello di leggere l'indirizzo di destinazione di un messaggio e di volta in volta avviarlo verso il corrispondente indicato.

Le due situazioni sono analoghe al caso in cui si voglia intraprendere un viaggio utilizzando le ferrovie oppure utilizzando un taxi.

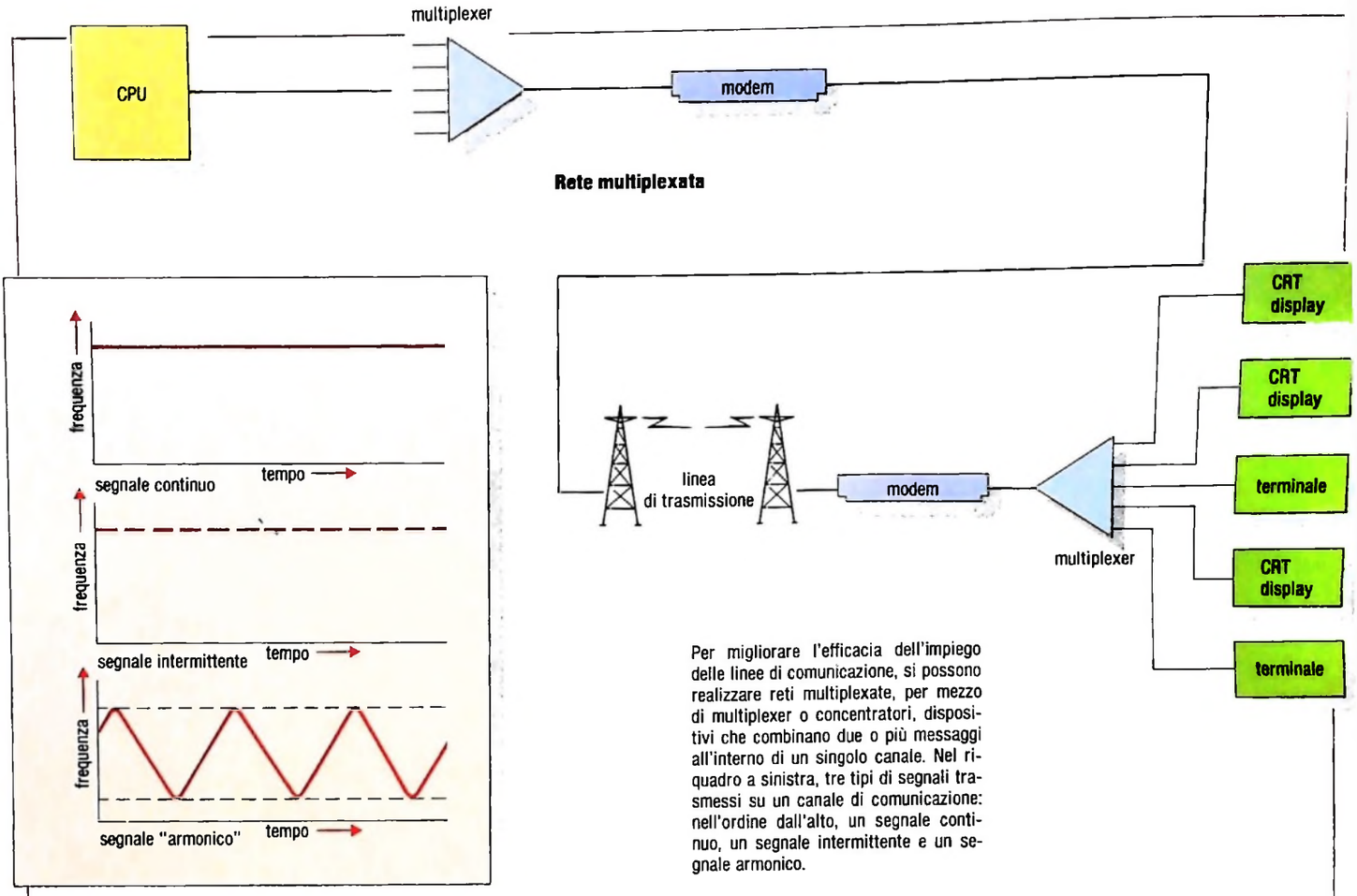
Nel caso delle ferrovie sarà sufficiente salire sul treno che viaggia verso la corretta destinazione e non sussistono ulteriori problemi. Sarà infatti il personale di terra che, azionando opportunamente gli scambi, preparerà una via fisica che, congiungendo il punto di partenza X con quello di destinazione Y, assicurerà che il treno transiti inequivocabilmente da X a Y.

Nel caso del taxi, nessuno prepara a terra una via fisica tra X e Y, ma l'autista sceglie di volta in volta il percorso che più gli conviene e, giunto, per esempio, a una tappa intermedia, può incaricare un collega, che si diriga più velocemente verso la destinazione indicata, di trasportargli il passeggero.

In un caso dunque la via è predeterminata sulla base dell'indirizzo e successivamente sul percorso così costruito si avviano i messaggi, nell'altro caso nessuna via viene predeterminata, ma i messaggi muniti di indirizzo trovano la via verso la destinazione scegliendosi i percorsi entro la rete di comunicazione.

Il traffico dati inserito nella tecnica di commutazione convenzionale presenta quindi lo svantaggio di impegnare e disimpegnare in continuazione gli organi di indirizzamento, creando una situazione di impegno artificiale non adeguata al mezzo di comunicazione impiegato.

Ecco quindi che per il servizio dati è più logico affidare alla



Per migliorare l'efficacia dell'impiego delle linee di comunicazione, si possono realizzare reti multiplexate, per mezzo di multiplexer o concentratori, dispositivi che combinano due o più messaggi all'interno di un singolo canale. Nel riquadro a sinistra, tre tipi di segnali trasmessi su un canale di comunicazione: nell'ordine dall'alto, un segnale continuo, un segnale intermittente e un segnale armonico.

rete i messaggi con il relativo indirizzo, e lasciare che di volta in volta sia la rete a gestire la situazione destinando i messaggi secondo i percorsi a suo giudizio più convenienti.
Il modo convenzionale di commutazione viene detto "commutazione di circuito", mentre il modo di smistare i messaggi sulla base del loro indirizzo viene detto "commutazione di messaggio".
La commutazione di messaggio può venire "perfezionata" dando ai messaggi un formato standard; ovvero tutti i messaggi che si presentano alla rete possono essere sezionati in moduli standard di assegnata lunghezza prima di essere inoltrati per la trasmissione. Questo metodo di funzionamento viene detto "commutazione di pacchetto" (Packet switching).
La commutazione di pacchetto tratta messaggi di lunghezza ottimale standard (1000-2000 bit), destinandoli secondo il loro indirizzo analogamente alla commutazione di messaggio, ma sfruttando la maggior flessibilità offerta dalla omogeneità e contenuta lunghezza del formato.

Multiplexer e concentratori

Per migliorare l'efficacia dell'impiego delle linee di comunicazione che sono assai costose è possibile utilizzare apparecchiature dette *multiplexer* o *concentratori*
Un multiplexer (chiamato anche multiplatore) è un dispositivo utilizzato per combinare due o più messaggi all'interno di un singolo canale di comunicazione e si comporta in modo trasparente, non è, cioè, rilevante ai fini della gestione del messaggio, in quanto non fa nulla sui dati che passano attraverso di esso (figura in alto a destra).
Ci sono due modi fondamentali con cui viene fatto il multiplexing: il multiplexing a divisione di frequenza (Frequency Division Multiplexing o FDM) e il multiplexing a divisione di tempo (Time Division Multiplexing o TDM).
Per comprenderne la differenza, si pensi ad un canale di comunicazione avente due dimensioni, che sono la frequenza e il tempo.
Per esempio, i tre grafici in alto a sinistra mostrano differenti tipi di segnali, trasportati da un canale di comunicazione; essi sono rappresentati da diagrammi frequenza-tempo.
Il primo mostra un segnale continuo di frequenza f_1 rappresentato da una riga continua parallela all'asse del tempo al valore f_1.
Il secondo mostra un segnale intermittente della stessa frequenza. Come nella figura precedente la linea è parallela all'asse dei tempi al valore f_1, ma è discontinua; rappresenta infatti la natura intermittente (on/off) del segnale.
Il terzo mostra un segnale "armonico", tipo quello prodotto dalla sirena di un'autoambulanza. La rappresentazione dell'escursione tra le frequenze f_1 e f_2 è fornita da una linea oscillante; questa rappresentazione sarà quella utilizzata per le future spiegazioni.

Lezione 32

Un po' di grafica (II)

La volta scorsa abbiamo iniziato a esaminare un orogramma per tracciare i bioritmi di una persona. Il nostro obiettivo è quello di avere effettivamente i grafici relativi ai tre ritmi, fisico, emotivo e intellettivo, disegnati sullo schermo, in modo da mostrare alti e bassi del loro andamento, ma soprattutto la loro intersezione con un asse "neutro", quando si passa da un andamento basso a uno alto e viceversa, che è proprio il punto che gli esperti di bioritmo considerano critico.
Per fare ciò, indirizzeremo lo schermo sul singolo PIXEL, ovvero sul singolo quadratino di quelli che compongono le lettere, che può essere individuato da una coppia di coordinate.
Infatti:

- lo schermo, pur avendo 8 righe di scrittura, in realtà corrisponde a 64 linee formate dai sottili quadratini;
- analogamente, pur avendo 40 colonne, queste corrispondono a un totale di 240 linee formate dai piccoli quadratini.

L'istruzione

PSET (x,y)

fa scurire il pixel di colonna x e riga y; di fatto, le righe sono numerate da 0 a 63 (dall'alto in basso) e le colonne da 0 a 239 (da sinistra a destra).

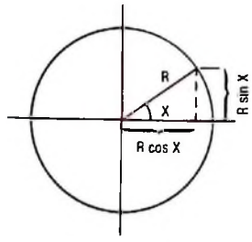
Così il programma

```
5 CLS
10 FOR I=0 TO 63
20 PSET(239,I)
30 NEXT I
```

fa annerire l'ultima colonna di pixel dello schermo, mentre con il programma

```
100 CLS
110 FOR I=0 TO 239
115 LET J=I
120 IF I>63 AND I<127 THEN J=127-I
130 IF I>=127 AND I<190 THEN J=I MOD 64
140 IF I>=190 THEN J=253-I
150 PSET(I,J)
160 NEXT I
```

otteniamo una linea che "rimbalza" sullo schermo:



Osserviamo ancora la nostra istruzione PSET: chi sa un po' di trigonometria conosce l'esistenza di due funzioni

sin(x) e cos(x)

che permettono di ottenere le coordinate di tutti i punti di una circonferenza; come si vede infatti nella figura a lato un punto sulla circonferenza di raggio R può essere individuato dalle coordinate

R*cos(x) e R*sin(x)

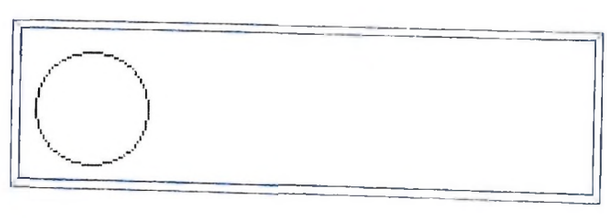
ove x è l'angolo individuato da uno degli assi e dal raggio passante per il punto. Così, se noi facciamo variare x da 0 a 2 pi greco (che equivale ad assumere tutti gli angoli da 0 a 360 gradi), le due coordinate descrivono l'intera circonferenza. Se ad esse noi sommiamo rispettivamente P e Q, spostiamo il centro del cerchio in una posizione (P, Q). Così il programma:

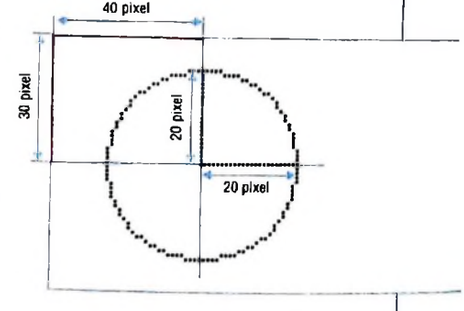
```
200 INPUT"RAGGIO,COORD.CENTRO";R,P,Q
205 CLS
210 FOR X=0 TO 6.28 STEP .1
220 PSET(R*SIN(X)+Q,R*COS(X)+P)
230 NEXT X
```

traccia una circonferenza di raggio R e centro di coordinate P,Q.
Per esempio, fornendo i valori

```
Ok
RUN
RAGGIO,COORD.CENTRO? 20,40,30
```

otteniamo l'immagine:





Attenzione che i valori delle coordinate devono sempre essere compresi tra 0,63 e tra 0 e 239; se noi poniamo, per esempio, il centro del cerchio in 10,10, dopo un certo tratto di curva compare il messaggio

```
?FC Error in 220
```

a segnalarci il problema.
Infatti, se ora chiediamo la lista dell'istruzione in errore, otteniamo:

```
LIST 220
220 PSET(R*COS(X)+P,R*SIN(X)+Q)
```

che ci indica un errore negli argomenti: con la richiesta di stampa "diretta":

```
PRINT R*SIN(X)+Q
```

otteniamo la risposta

```
 26.169928076391
Ok
```

che è legale, mentre, chiedendo il valore dell'altro parametro, otteniamo:

```
PRINT R*COS(X)+P
-1.770022345108
Ok
```

che non è nei limiti ammessi per la PSET.
Approfittiamo dell'occasione per osservare come abbiamo cercato l'errore: quando l'M10 si interrompe per errore, mantiene completamente inalterata la sua configurazione di memoria; con ciò è possibile usare le istruzioni come se fossero dei COMANDI, che vengono eseguiti direttamente, in modo da poter raccogliere informazioni sul contenuto delle variabili, e quindi poter individuare l'errore.
Torniamo al nostro problema del bioritmo: cancellando le istruzioni che avevamo inserito per ottenere le stampe di prova (cioè quelle da 106 a 108), possiamo ora occuparci dei grafici.
Poiché l'istruzione PSET indirizza lo schermo direttamente, essa non altera il contenuto dello stesso nelle parti che non sono interessate dall'istruzione stessa: sarebbe come dire che, mentre le istruzioni PRINT effettuano l'indirizzamento di una linea dopo l'altra, comportando uno "scorrimento" dello schermo dopo le otto linee (si parla di SCROLL), nel caso della PSET, l'indirizzamento del singolo pixel non influisce assolutamente sul contenuto di altre parti dello schermo.
Possiamo quindi inserire scritte, commenti, didascalie che ci interessino per illustrare i nostri grafici, sicuri che le successive PSET non ne modificheranno la struttura; se invece scrivessimo tali scritte con PRINT dopo avere effettuato delle PSET, lo scroll dello schermo sposterebbe completamente l'immagine.
Cominciamo allora con il porre nell'angolo in alto a destra dello schermo tre scritte delle tre iniziali dei grafici, che affiancheremo con il modo con cui denoteremo i grafici stessi, per poterli riconoscere:

```
111 PRINT"                                                F"
112 PRINT"                                                S"
113 PRINT"                                                I"
```

A fianco di queste scritte inseriremo tre campioni delle curve:
- a pixel contigui per la prima
- un pixel sì e uno no per la seconda
- un pixel sì e due no per la terza

in modo da distinguere le tre curve.

Prima però, è necessario stampare altre eventuali scritte con PRINT, e di fatto ce ne

serve una per inserire una scala di giorni del mese a fianco di una retta che rappresenta lo scorrere del tempo:

```
114 PRINT
115 PRINT"      5     10      15      20      25      30"
```

Ora, a fianco delle lettere che identificano le tre curve, nell'angolo destro, inseriamo i campioni delle curve:

```
116 FOR L=225 TO 235
117 PSET(L,4)
118 IF L MOD 2 = 0 THEN PSET(L,12)
119 IF L MOD 3 = 0 THEN PSET(L,20)
120 NEXT L
```

- tutti e tre i campioni vanno da colonna 225 a colonna 235
- il primo si trova alla riga 4
- il secondo si trova a riga 12 ma "accendiamo" solo i pixel pari
- il terzo si trova a riga 20, ma "accendiamo" solo i pixel multipli di 3

Quindi:

```
122 FOR K=1 TO 210
130 PSET(K,30)
135 IF K MOD 7 = 0 THEN PSET(K,31)
138 IF K MOD 35= 0 THEN PSET(K,32)
140 NEXT K
```

Ora possiamo tracciare l'asse dei tempi, circa a metà dello schermo (alla riga 30); poiché tracceremo la linea per la lunghezza di 210 pixel, possiamo considerare che, considerando 30 giorni, ogni giorno corrisponda a 7 pixel; allora aggiungiamo un pixel ogni 7 alla linea 31 per evidenziare i giorni; poiché inoltre abbiamo costruito la scala ogni 5 giorni, aggiungiamo un altro pixel a fronte di ogni 5*7=35 dei precedenti: in tal modo avremo evidenziato la scala sull'asse.
A questo punto dovremmo tracciare le curve: lo faremo nella prossima lezione.

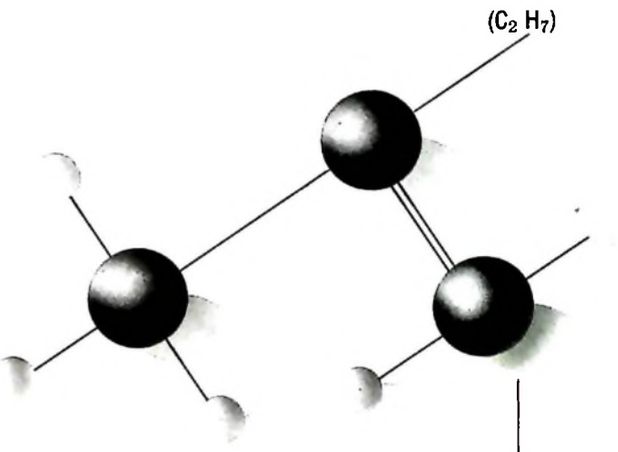
Che cosa abbiamo imparato

In questa lezione abbiamo appreso:

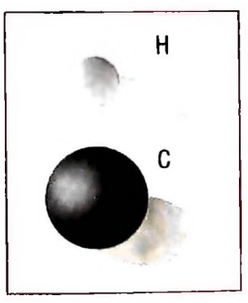
- come usare l'istruzione PRINT in "command mode", come se fosse un comando direttamente eseguibile, per poter cercare errori nei programmi;
- che cosa è un PIXEL, ovvero uno di quei quadratini che compongono lo schermo e che sono usati, in opportune configurazioni, per rappresentare i caratteri sullo schermo;
- l'istruzione PSET (x, y), che "accende" il pixel che si trova in colonna x e colonna y, con x che varia da 0 a 239 e y che varia da 0 a 63.

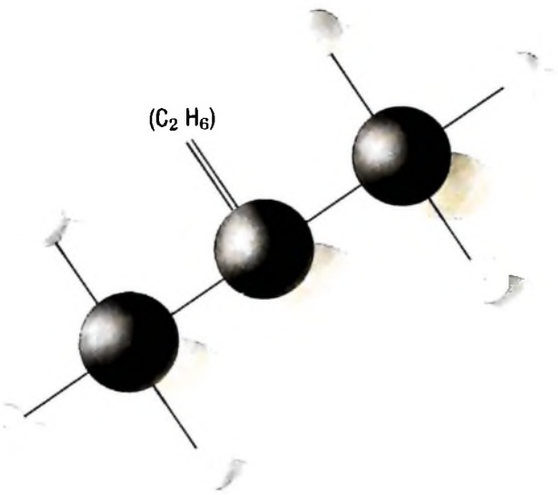
SISTEMI ESPERTI: REALIZZAZIONI PRATICHE

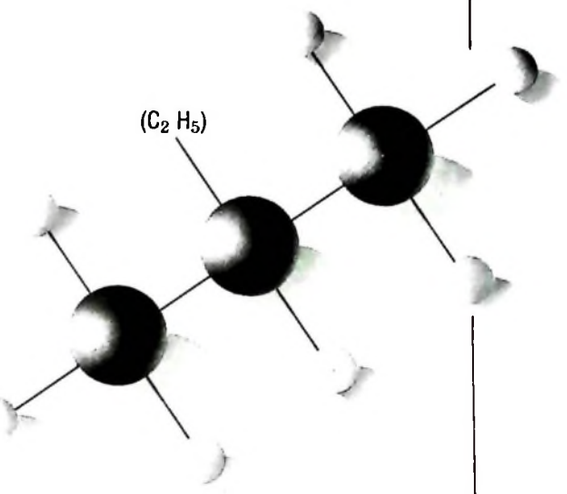
Un settore in rapida espansione, che offre continuamente nuovi progetti e nuove idee.



Concludiamo con questo articolo la panoramica sui sistemi esperti esaminando alcuni prodotti effettivamente realizzati e sperimentati. È bene avere presente che:
1) soltanto alcuni dei sistemi presentati possono definirsi dei veri prodotti disponibili e acquistabili sul mercato informatico; altri sono dei prototipi sviluppati in ambiente universitario o industriale per scopi di ricerca o di utilizzo interno;
2) la rassegna è largamente incompleta e soltanto indicativa dei campi di applicazione: ciò è dovuto a motivi di spazio e al fatto che l'estendersi dell'interesse per i sistemi esperti risale agli ultimi anni o addirittura mesi, e quindi diversi prodotti non sono ancora documentati;



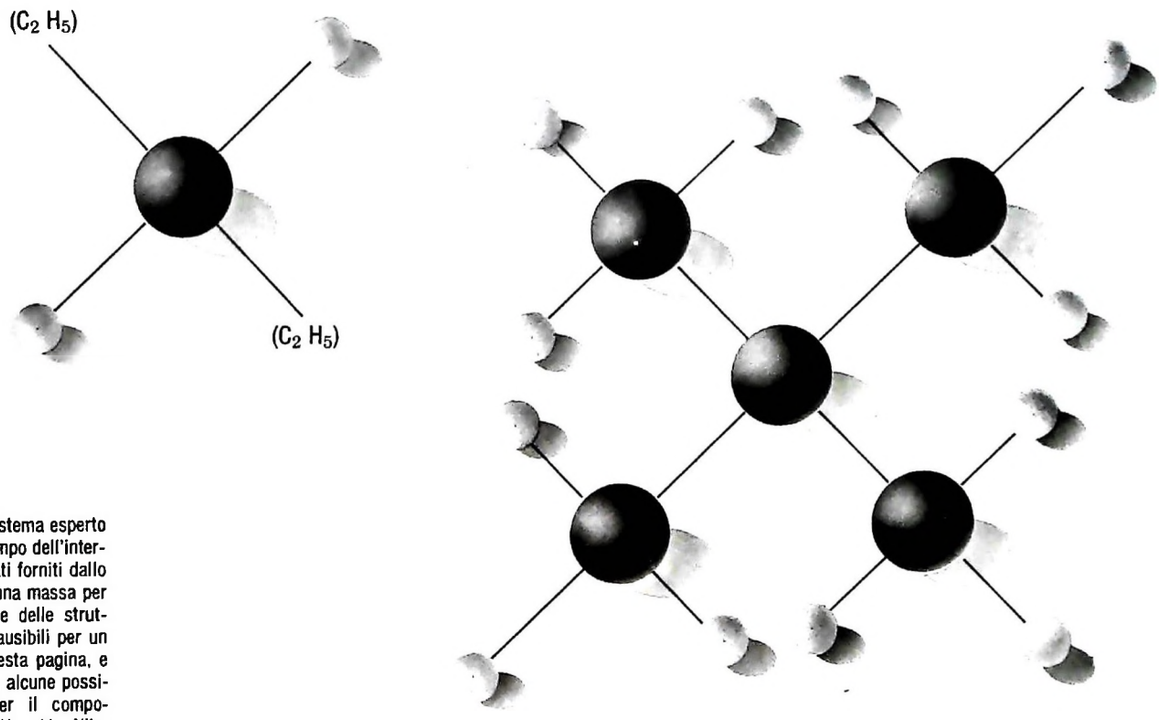




3) non si considerano tutti gli elementi (rappresentazione della base di conoscenza, strategia di controllo, interazione con l'utente) di un sistema, ma soltanto quelli utili per essere collegati a ciò che è stato visto nei due precedenti articoli.

Una rassegna

Il primo progetto di realizzazione di un sistema esperto risale alla seconda metà degli anni Sessanta presso l'università



DENDRAL è un sistema esperto che opera nel campo dell'interpretazione dei dati forniti dallo spettrometro di una massa per la determinazione delle strutture chimiche plausibili per un composto. In questa pagina, e nella precedente, alcune possibili strutture per il composto organico C_5H_{12} (da Nilsson, Principles of Artificial Intelligence).

americana di Stanford, e ancora oggi non si può considerare concluso: si tratta di DENDRAL, un sistema che opera nel campo dell'interpretazione, cioè nell'analisi di dati per determinarne il significato. In questo caso i dati provengono da uno strumento chiamato spettrografo di massa (sono misure della massa di frammenti molecolari di un composto organico sconosciuto) e da altri esperimenti chimici, e scopo del sistema è la determinazione di una o più strutture chimiche plausibili per il composto (figure in alto e nella pagina precedente).

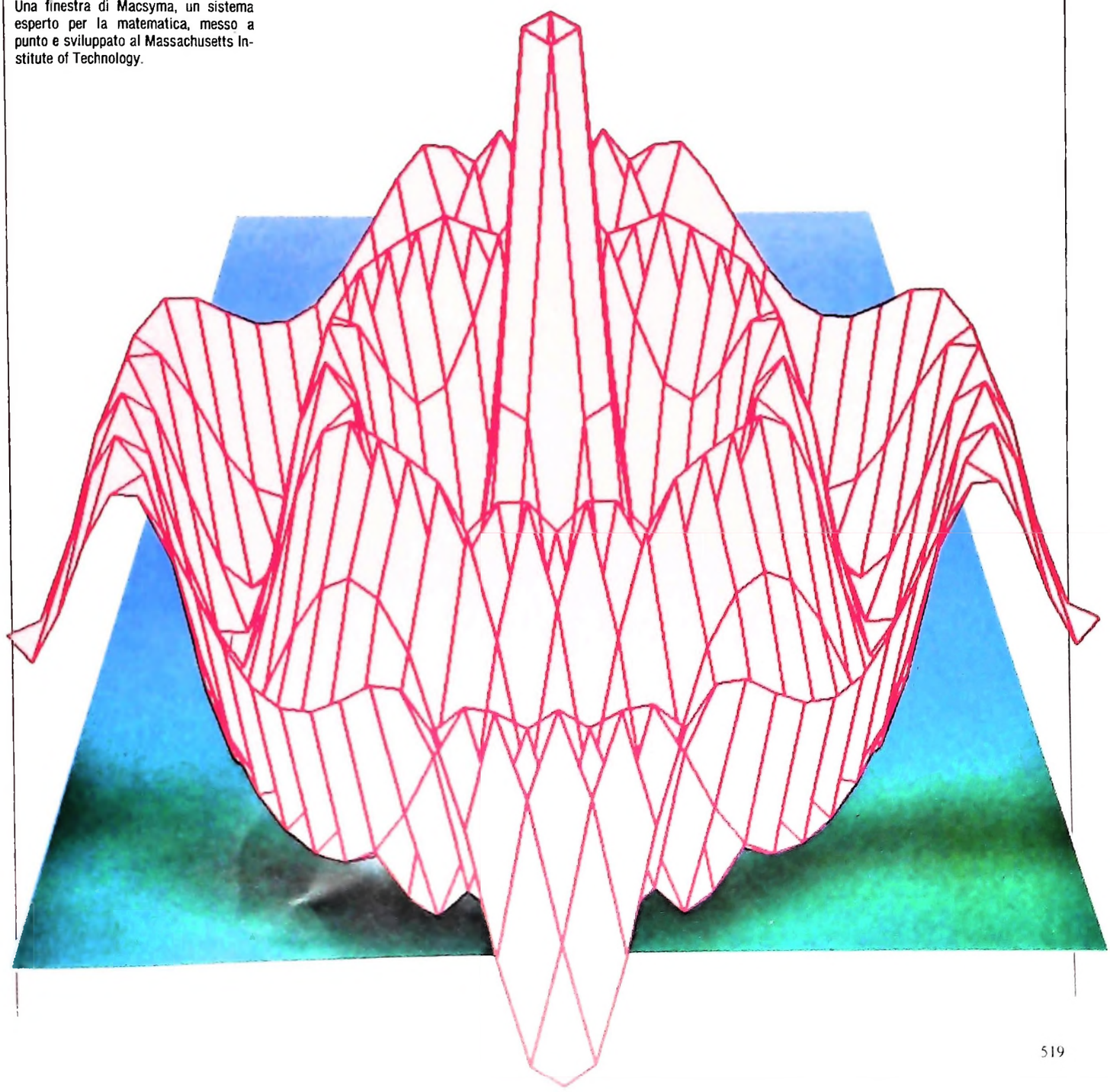
Si immagina facilmente come lo spazio delle strutture chimiche deducibili a partire dai dati sperimentali sia enorme: siamo perciò in un caso in cui non è possibile una ricerca completa (esaustiva) di tutti gli eventuali candidati, anche se sulla carta esiste un preciso algoritmo per generarli tutti. Per orientare la ricerca verso le strutture corrette il sistema possiede una vasta base di conoscenza specifica, parte della quale è codificata in forma di regole. Queste regole permettono di dedurre dei vincoli a partire dai dati sperimentali che servono a guidare un generatore di strutture plausibili, riducendo così l'esplosione combinatoriale. Su queste strutture operano altre regole che rappresentano precise conoscenze chimiche e verificano la bontà delle strutture proposte, restringendo progressivamente il numero di possibilità, ed eliminando così intere classi di soluzioni.

Le capacità di DENDRAL allo stato attuale, frutto di oltre quindici anni di estensioni e revisioni, sono giudicate quasi impressionanti da chimici esperti e il sistema sta per essere utilizzato in ambienti industriali e non più soltanto universitari. Un sottoprodotto di DENDRAL, se così si può chiamare visto che è costato uno sforzo quasi superiore, è META-DENDRAL. Il suo scopo, come dovrebbe suggerire il prefisso META-, è di produrre nuove regole che vengono utilizzate da DENDRAL per l'identificazione delle strutture chimiche, costituendo perciò un modulo di apprendimento per il sistema principale. L'idea di base è molto intuitiva: si parte dall'analisi dei casi che si presentano cercando di generalizzare a regole comunque valide; le ipotesi prodotte sono quindi di nuovo controllate rispetto alla casistica esistente. Al di là dei dettagli specifici, interessa il fatto, riconosciuto pubblicamente, che tutte le "scoperte" sinora operate da META-DENDRAL, che hanno incrementato la base di conoscenza di DENDRAL, potevano essere acquisite in tempo decisamente minore da parte di un bravo chimico attraverso l'analisi manuale dei dati; questo non è certo dovuto a incapacità dei realizzatori del progetto, ma ai problemi di complessità accennati nel primo articolo. Il risultato è tuttavia rilevante, poiché dimostra come il sistema sia in grado di evolvere in maniera autonoma.

Un altro progetto di fondamentale importanza nella storia dei sistemi esperti è MYCIN, sviluppato anch'esso a Stanford, vera fucina di produzione in questo campo, a partire dalla metà degli anni Settanta. Il campo di applicazione del sistema è la diagnosi delle malattie infettive del sangue, dove raggiunge risultati considerati di buon livello medico; il sistema è utilizzato attualmente per addestrare gli studenti di

medicina dell'università americana.
La produzione della diagnosi e dei suggerimenti per una terapia avviene grazie all'interazione con l'utente (un medico) che risponde alle richieste del sistema fornendo i dati opportuni. Il sistema genera un'ipotesi di diagnosi che viene successivamente verificata attraverso l'esame delle regole della BdC; in questo caso si tratta di regole della forma SE ... ALLORA ... esaminata nei precedenti articoli, con in più una sofisticazione che considereremo tra poco.
MYCIN corrisponde perfettamente alle caratteristiche generali che abbiamo esposto. La base di conoscenza è interamente costituita da regole, attualmente oltre 500, ciascuna delle quali corrisponde a una parte di conoscenza altamente specializzata: le capacità del sistema possono essere allargate introducendo nuove regole. Il motore di inferenza è piuttosto semplice; la verifica di un'ipotesi è effettuata tramite il *concatenamento all'indietro* descritto precedentemente: alcuni antecedenti di una regola sono dati già presenti o richiesti all'utente, altri diventano ipotesi che devono essere a loro volta verificate procedendo all'indietro sino a che è possibile giudicare la bontà dell'ipotesi iniziale.
MYCIN opera una ricerca esaustiva del proprio spazio di possibilità, cioè applica, quando diverse regole conducono alle stesse conclusioni, l'intero insieme di possibilità. In altri

Una finestra di Macsyma, un sistema esperto per la matematica, messo a punto e sviluppato al Massachusetts Institute of Technology.



termini ancora, percorre tutte le strade che si dipartono da un nodo del grafo considerato. Questo è possibile perché lo spazio di ricerca del problema è abbastanza limitato, quindi l'esplosione combinatoriale è ridotta al punto che si possono esaminare tutte le possibilità in un tempo ragionevole. Inoltre sembrerebbe inutile cercare nuove diagnosi quando il sistema ne ha già individuato una buona. Il fatto importante introdotto da MYCIN è che le regole non deducono quasi mai certezze assolute, ma soltanto con una certa probabilità. Un esempio è: "SE è plausibile almeno al 65 per cento che il valore di A sia compreso tra 50 e 60 ALLORA il valore di B è positivo con una probabilità dell'85 per cento". È perciò ragionevole verificare tutte le ipotesi per ritornare a quella che ha la maggiore probabilità di essere corretta. Soltanto nei rari casi in cui sono raggiunte conclusioni con una certezza assoluta il sistema evita di proseguire la ricerca.

Un problema che è stato affrontato e risolto in modo abbastanza brillante da MYCIN è quello della spiegazione del ragionamento seguito per dedurre il particolare risultato. Si tratta di un problema molto importante che non è stato ancora sufficientemente trattato negli studi svolti. In questo caso è un sistema apposito che svolge questa e altre funzioni, denominato TEIRESIAS; esso è ugualmente costituito da regole SE ... ALLORA ... che hanno come oggetto le regole stesse di MYCIN (si veda anche il precedente articolo).

Un primo metodo piuttosto rozzo di giustificazione di un risultato raggiunto consiste nel mostrare tutte le regole che sono state utilizzate per la deduzione; questo tipo di spiegazione può essere tuttavia insufficiente, perché non rende conto di possibili domande dell'utilizzatore: perché è stata scelta questa particolare regola e non un'altra, quali conseguenze avrebbe avuto la scelta della regola 65? La risposta a questo tipo di domande è possibile soltanto avendo a disposizione informazioni esplicite (quindi ad esempio in forma di regole) sulla conoscenza di base del sistema esperto.

L'esempio di MYCIN è stato seguito con successo da altri sistemi sviluppati nelle università americane. Tra questi ha conseguito un buon successo PUFF (uscito anch'esso da Stanford) che opera nel campo delle malattie polmonari e viene quotidianamente usato presso il Pacific Medical Center di San Francisco in California.

Avendo considerato due esempi ormai "storici" in questo campo, merita ora un accenno il sistema che probabilmente ha avuto l'evoluzione in assoluto meno travagliata e più veloce, conseguendo anche interamente gli scopi originali.

Si tratta del sistema R1, sviluppato presso l'università americana di Carnegie Mellon, un altro dei santuari dell'Intelligenza Artificiale.

È questo il sistema cui si è velatamente accennato nell'articolo precedente: suo scopo è la configurazione dei calcolatori VAX 11/780 prodotti dalla Digital Equipment Corporation, uno dei colossi mondiali nella produzione di calcolatori. I VAX 11/780 sono uno dei modelli di punta della casa americana, e la loro configurazione richiede (o meglio possiamo dire richiedeva) l'attenzione di diversi tecnici, che attualmente si limitano a controllare che il risultato prodotto da R1 non contenga clamorosi errori dovuti alla incompletezza della base di conoscenza.

Questa è nuovamente costituita da regole SE ... ALLORA ... la maggior parte delle quali specifica il posizionamento di un nuovo pezzo se sono verificate le condizioni opportune. Oltre alle regole, R1 possiede una base di conoscenza specifica sugli elementi necessari alla configurazione (processori, memorie, bus; si tratta complessivamente di circa 400 pezzi) alla quale attinge nel corso della configurazione per verificare il rispetto dei vincoli imposti dalle regole. Le regole sono circa un migliaio, numero che va aumentando quando nuovi componenti sono messi in catalogo. È stato calcolato che in media da ogni nodo dello spazio di ricerca (quindi da ogni configurazione parziale) si dipartono 3 strade distinte; la particolare natura del problema e il motore di inferenza di R1 permettono tuttavia una scelta che soltanto raramente deve essere ritrattata, e la configurazione viene quindi svolta in tempi decisamente brevi.

R1 è un caso interessante in quanto il suo sviluppo completo, dall'idea iniziale alla fase di utilizzo sperimentale presso la Digital, ha richiesto complessivamente 8 mesi. Il sistema è stato sviluppato principalmente da una sola persona aiutata da tecnici esperti nella configurazione. Attualmente R1 è utilizzato presso la Digital, e il suo operato viene considerato quasi sempre corretto.

Conclusioni

Questa breve rassegna, anche se largamente incompleta, dovrebbe dare un'idea della situazione attuale per quanto riguarda i sistemi esperti. È evidente che siamo ancora lontani dal momento in cui questi prodotti avranno una diffusione a livello industriale (confrontabile con quella dei prodotti EDP standard) e tantomeno domestica. Tuttavia si deve considerare che nel settore vengono investiti energie e sforzi ogni giorno maggiori, e in questi stessi ultimi mesi vengono alla luce nuovi progetti e nuove idee che contribuiscono grandemente ad accelerare la disponibilità di prodotti. In definitiva si prevede che nei prossimi anni i sistemi esperti acquisiranno una certa importanza come strumenti realmente utili a livello industriale e di ricerca, grazie anche all'evoluzione dell'hardware che renderà disponibili calcolatori sufficientemente potenti da supportare queste applicazioni a costi accettabili. Diverso è il discorso per quanto riguarda le applicazioni nel campo dei calcolatori personali. L'evoluzione, pure vertiginosa, di questi ultimi non è ancora tale da permettere la disponibilità di macchine adeguate a reggere sistemi così complessi: è vero che esistono alcuni prodotti per piccole macchine fatti passare sotto il nome di sistemi esperti, ma questa ci sembra onestamente un'operazione di contrabbando. Inoltre i prodotti sinora apparsi sono sistemi esperti che interagiscono con un altro esperto (eventualmente meno preparato ma sicuramente non estraneo alla materia); il problema dell'interazione con un utente non esperto aggiunge un altro ordine di complessità che richiede ulteriori studi e allontana, almeno per ora, questi prodotti dal mercato dei calcolatori personali.

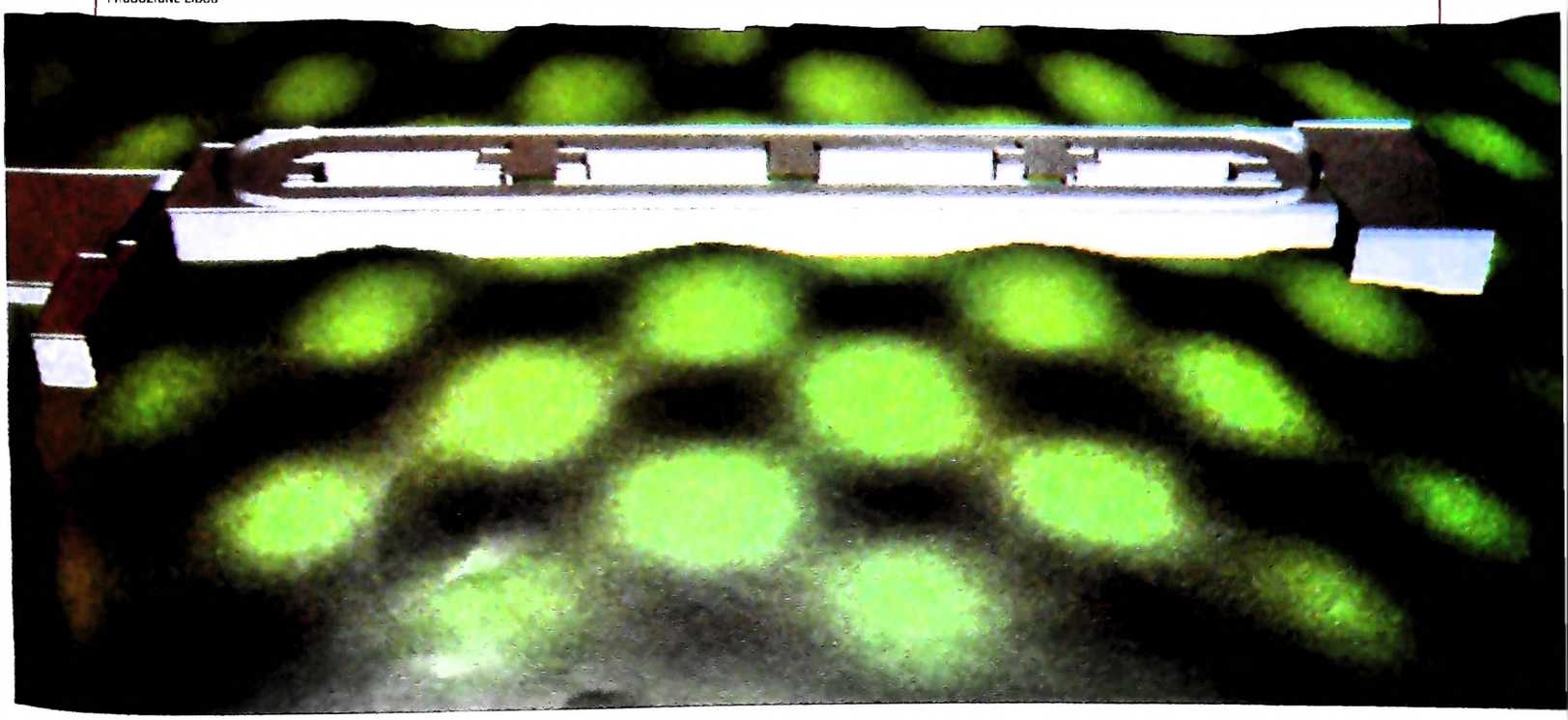
L'INTERPOLAZIONE

Un problema che ha sempre destato notevole interesse in svariati campi d'applicazione e a vari livelli di difficoltà.

Con la presente lezione si vuole introdurre uno dei più noti problemi dell'analisi numerica: l'interpolazione di dati. La necessità di dover interpolare un insieme di misurazioni affiora praticamente in tutte le discipline scientifiche; in particolare poi la teoria dell'interpolazione ha avuto notevoli sviluppi negli ultimi anni, trovando ausilio nell'utilizzo di sistemi di calcolo automatici; infatti la risoluzione di problematiche inerenti l'interpolazione richiede, anche in casi relativamente semplici, un elevato tempo di calcolo.
Vediamo ora cosa s'intende con interpolazione e come viene applicata. Dato un generico insieme di punti (appartenenti per esempio al piano cartesiano xy oppure allo spazio tridimensionale xyz) per interpolazione s'intende la determinazione di una funzione (che in dipendenza dagli spazi in esame sarà del tipo $y=f(x)$ o $z=f(x,y)$) che "approssimi" con una certa precisione tutti i punti appartenenti all'insieme dato. Se l'insieme è composto dai punti (x_1, y_1), (x_2, y_2) e (x_3, y_3), la funzione deve essere tale da fornire, se calcolata in x_1 oppure in x_2 o x_3 (ascisse dei tre punti), dei valori che tendono a rendere minima la distanza fra l'ordinata rilevata dal calcolo analitico della funzione approssimante e l'ordinata effettiva del punto esaminato (nel nostro esempio le ordinate effettive sono y_1, y_2, y_3).
Questo naturalmente non è il solo modo di pensare a un'interpolazione; in realtà è possibile chiedere alla funzione interpolante ulteriori caratteristiche, o far sì che soddisfi particolari condizioni di precisione; tutto ciò esula però da questa breve introduzione, che dovrebbe tuttavia far intuire l'im-

In questa immagine il terreno è stato ottenuto utilizzando un processo di interpolazione di funzioni bispline. Inoltre il valore della funzione interpolante controlla il grado di intensità luminosa dell'ombreggiatura sul terreno. (Estratto di una animazione da "Venti progetti per il lingotto").

PRODUZIONE EIDOS



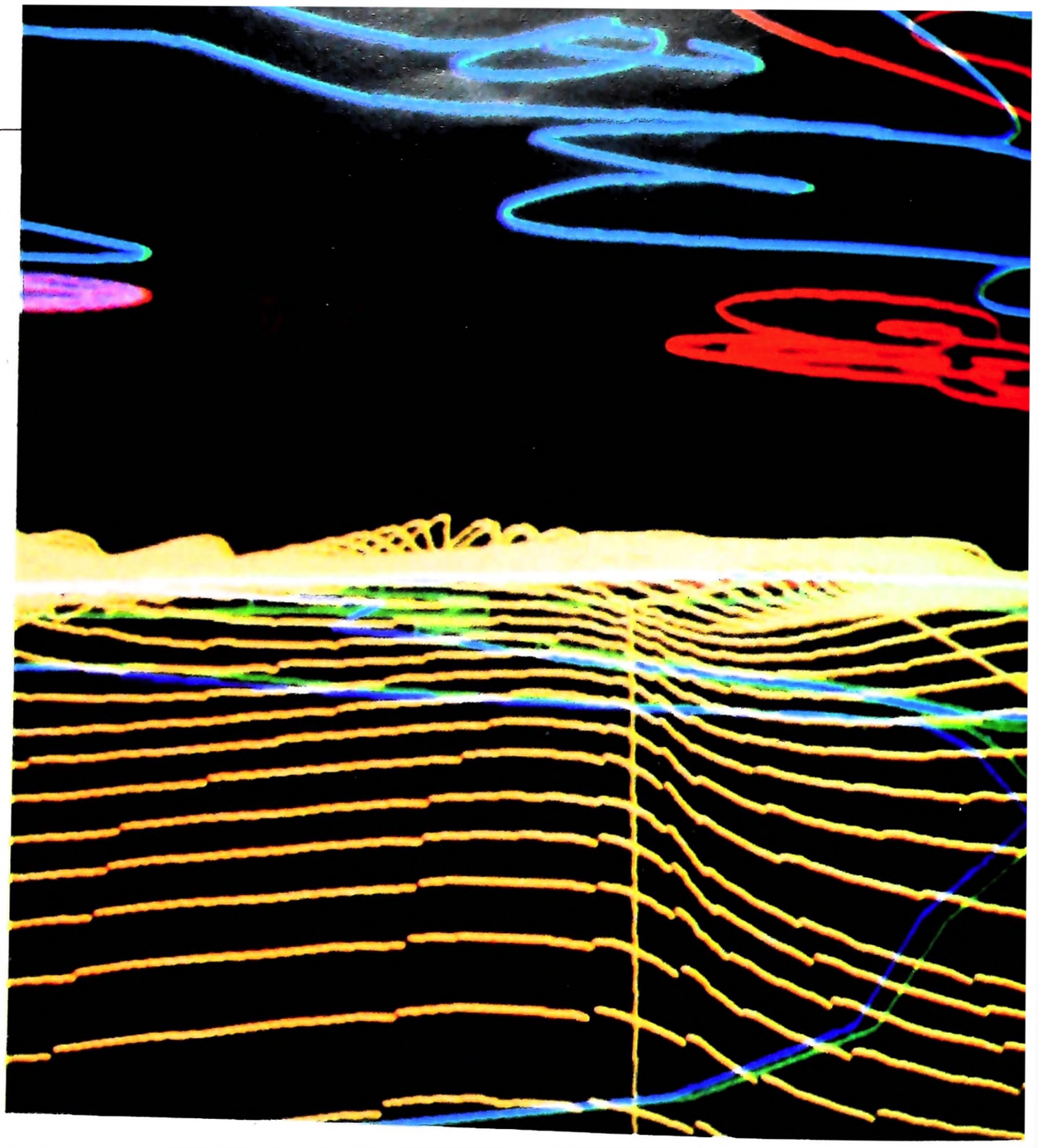
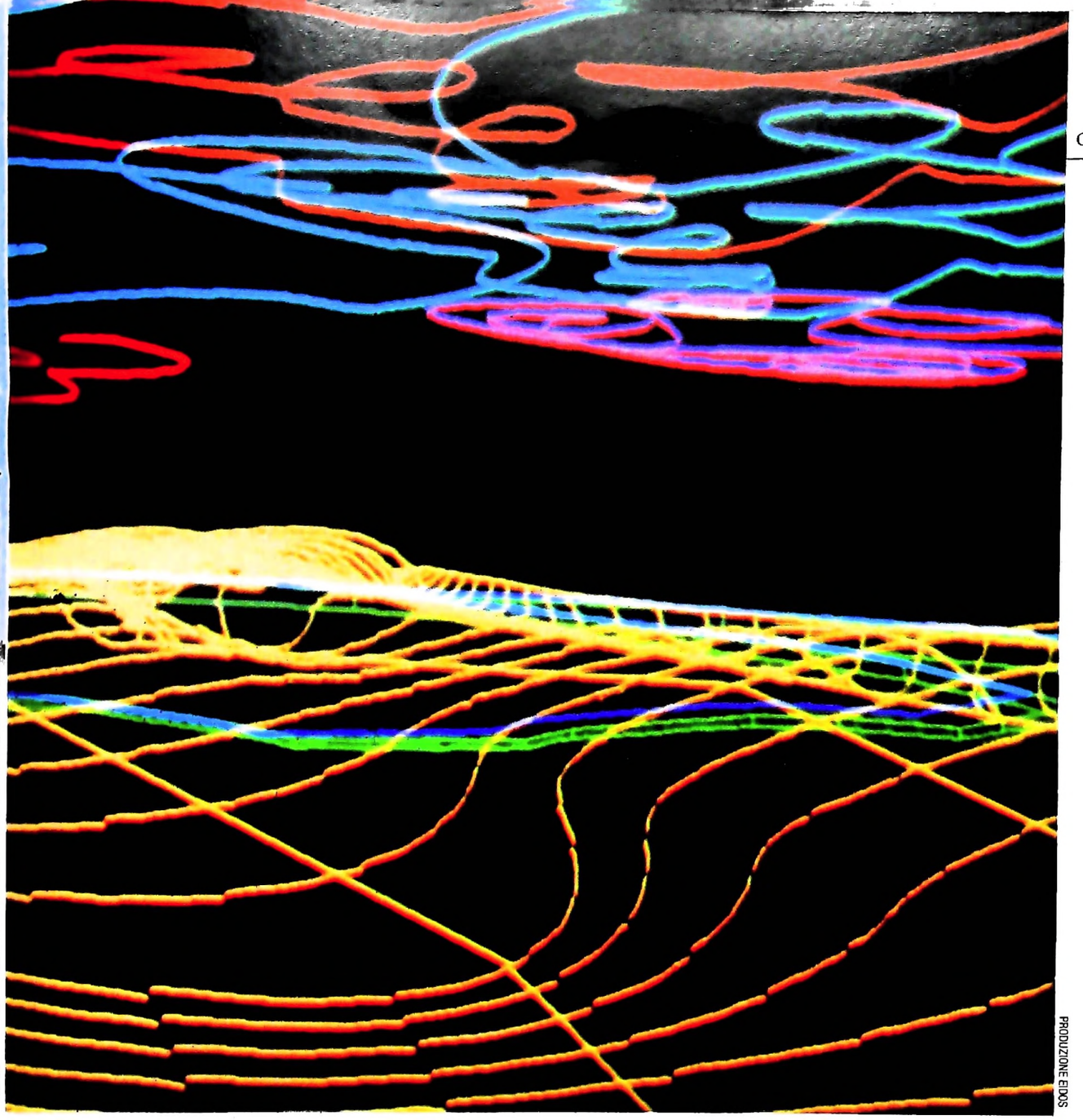


Immagine ottenuta come interpolazione di funzioni bispline. Tali funzioni sono costituite da un polinomio che consente di generare la linea curva che meglio approssima le coordinate dei punti dati in input, che costituiscono l'insieme dei dati da interpolare.

portanza che l'interpolazione riveste nell'ambito scientifico o commerciale.
Un'applicazione tipica in campo grafico è invece quella della modellazione di curve o superfici tridimensionali che viene ottenuta generalmente mediante l'interpolazione con il metodo delle "spline cubiche" derivante dalla teoria dell'elasticità. Essa ha lo scopo di unire più punti con un tratto di curva il più regolare possibile, permettendo inoltre di trovare un valore numerico che rappresenti la "curvatura globale".
È importante notare che tutto ciò è molto attendibile quando si valuta la funzione ottenuta all'interno dell'intervallo di interpolazione, effettuando quindi calcoli con valori che sono compresi fra il valore massimo e il minimo fra quelli inseriti.
È invece sconsigliato, di solito, effettuare l'estrapolazione, cioè la valutazione della funzione interpolante all'esterno dell'intervallo di interpolazione; questo perché la funzione è stata calcolata basandosi interamente sull'insieme dei valori inseriti.

Il programma

Per concretizzare questo discorso viene ora fornito il listato di un semplice programma che realizza uno tra i più comuni metodi di interpolazione: l'*interpolazione lineare.*
Interpolazioni di questo tipo vengono effettuate solitamente quando si suppone che i punti noti siano legati da una relazione di tipo lineare (del tipo y=ax+b). Il programma infatti traducendo in BASIC un metodo matematico assai noto (metodo dei minimi quadrati) calcola l'interpolazione lineare su un dato insieme numerico, che viene fornito in ingresso, ottenendo come uscita l'equazione della retta desiderata, nella già citata forma y=ax+b. Viene inoltre fornita la rappresentazione grafica della retta interpolante.
È da notare che l'applicazione qui indicata non è in pratica utilizzata per applicazioni grafiche a causa della semplicità di rappresentazione che si ottiene; essa costituisce comunque un buon approccio a questo tipo di problematiche e un vali-



PRODUZIONE EIDOS

do spunto per acquisire maggiore conoscenza sulle applicazioni di tipo prettamente grafico (metodo delle spline, curve di Bezier).
Il listato è stato suddiviso in routine alle quali si accede mediante chiamate dal menù principale (è questo un tipico esempio di stesura di programma con sviluppo top-down).
La parte grafica ha inizio dalla linea 2600 dove vengono fissati i valori relativi alla risoluzione dello schermo utilizzato. Poiché la finestra di visualizzazione è dimensionata in base ai valori massimi e minimi di x e y, si devono calcolare questi valori con un semplice algoritmo (linee 2620-2690).
Successivamente viene applicata una trasformazione (simile a quella di windowing) che ha lo scopo di creare un vettore, contenente le coordinate convertite dei punti inseriti (linee 2700-2770).
È ora possibile visualizzare i punti, che vengono rappresentati mediante crocette la cui dimensione fisica (in pixel) è indicata nella costante alla linea 2610.

Alla linea 2950 ha inizio la routine di visualizzazione della retta calcolata. Essa viene ottenuta calcolando il valore della funzione ottenuta mediante l'algoritmo di cui alle linee 790-880 nei punti di ascissa massima e minima; mediante l'istruzione LINE viene poi tracciata la retta che unisce questi due punti.
Una procedura di "fine programma" è stata inserita alle linee 3500-3550 per permettere l'osservazione della funzione fino alla pressione di un qualsiasi tasto, che provoca il rientro al menù.

Ampliamenti

Sfruttando la struttura principale è possibile introdurre altre interpolazioni semplicemente inserendo gli opportuni algoritmi di calcolo.
Si segnala a proposito il seguente algoritmo già compatibile

con il programma e dedicato al calcolo di un'interpolazione geometrica:

```
FOR I=1 TO NP
X=A1(I):Y=LOG(A2(I))
SY=SY+Y:SX=SX+X:QY=QY+Y↑2:QX=QX+X↑2:PC
=PC+X*Y
NEXT I
REM**calcola i due coefficienti dell'equazione
A1=(NP*PC-SX*SY)/(NP*QX-SX↑2)
AO=(SY-SX*A1)/NP
```

Tale algoritmo, con l'esclusione delle ultime tre linee che vanno eliminate in quanto già presenti nel listato, deve essere introdotto nel programma fra la linea 860 e la linea 870 prevedendo ovviamente le opportune opzioni di scelta nel menù principale sul tipo di operazione da effettuare (interpolazione lineare o geometrica con conseguente ridirezione delle chiamate alle parti di programma relative). Inoltre sarà necessario introdurre un'ulteriore linea di programma in modo da visualizzare l'interpolazione geometrica:

```
xxx CLS:PRINT "INTERPOLAZIONE GEOMETRICA":
PRINT "FUNZIONE INTERPOLANTE : Y =";EXP(AO);
"X↑"; A1
```

dove xxx è un numero di linea da inserire in una apposita procedura da creare seguendo quella di cui alla linea 930.
Si noti infine che, a causa della struttura dell'algoritmo, non possono essere calcolate funzioni interpolanti geometriche su gruppi di dati che abbiano anche un solo valore negativo.
Utile espansione sarebbe progettare opportuni controlli sui valori inseriti dall'utente (valori numerici troppo grandi, valori negativi per interpolazione geometrica).

```
60 '**PROGRAMMA CALCOLO INTERPOLAZIONE*
70 DIM A1(80),A2(80)
90 '   ***SALTA AL MENU PRINCIPALE***
100 GOSUB 430
110 '**SELEZIONA LA SCELTA DEL MENU'***
120 ON B1 GOSUB 180,780,150
130 GOTO 100
140 '   ***FINE PROGRAMMA***
150 CLS:PRINT"FINE LAVORO":END
160 '***********************************
170 '   ***ROUTINE DI INPUT***
180 FOR I = 1 TO 80:A1(I)=0:A2(I)=0:NEXT:I=1
190 NP=0
200 CLS
210 PRINT"INS.PUNTI: 'F' PER TERMINARE"
220 PRINT I-1;" PUNTI INSERITI"
230 INPUT"     ASCISSA: ";A1$
240 IF A1$<>"F" THEN A1(I)=VAL(A1$) ELSE RETURN
270 INPUT"    ORDINATA: ";A2(I)
310 NP=I
320 I=I+1
330 IF I<81 THEN 200 ELSE NP= 80:RETURN
340 '   ***********************************
430 '   ***MENU' PRINCIPALE***
440 CLS
450 PRINT"MENU DELLE OPERAZIONI"
460 PRINT:PRINT"1) INSERIMENTO DATI"
510 PRINT"2) GRAFICO FUNZIONE INTERPOLANTE"
520 PRINT"3) FINE LAVORO"
525 PRINT
530 PRINT"   SELEZIONE :"
540 B1$=INKEY$:IF B1$="" THEN 540
555 PRINT@255,B1$
560 B1 =VAL(B1$)
565 IF B1=2 AND NP<2 THEN B1=1
570 RETURN
780 '***********************************
790 '**CALCOLI INTERPOLAZIONE LINEARE**
800 FOR I=1 TO NP
820 X=A1(I):Y=A2(I)
850 SY=SY+Y:SX=SX+X:QY=QY+Y ^ 2:QX=QX+X ^ 2:PC=PC
+(X*Y)
860 NEXT I
870 A1=(NP*PC-SX*SY)/(NP*QX-SX ^ 2)
880 A0=(SY-SX*A1)/NP
920 '*** STAMPA FUNZIONE LINEARE    ***
930 CLS:PRINT"INTERPOLAZIONE LINEARE":PRINT"RETTA
INTERPOLANTE: Y= ";A1;"X +(";A0;")"
935 A$=" ":PS=0
940 PRINT @200,"PREMERE<UN TASTO>PER CONTINUARE"
950 A$=INKEY$:IF A$="" THEN 950
960 IF PS=1 THEN RETURN
2590 '******PLOTTAGGIO INTERPOLANTI****
2600 RXS=239:RYS=55'****RIS. SCHERMO***
2610 OF=1 '**OFFSET PLOT PUNTO*********
2620   '***CALCOLO MASSIMI MINIMI***
2630 MX=A1(1):MY=A2(1):LX=A1(1):LY=A2(1)
2640 FOR CR=1 TO NP
2650 IF MX<=A1(CR) THEN MX=A1(CR)
2660 IF MY<=A2(CR) THEN MY=A2(CR)
2670 IF LX>=A1(CR) THEN LX =A1(CR)
2680 IF LY>=A2(CR) THEN LY=A2(CR)
2690 NEXT CR
2700 '**CALCOLO VETTORE NORMALIZZATO***
2710 IF MX=LX OR MY=LY THEN CLS:PRINT"NON
E' POSSIBILE DISEGNARE IL GRAFICO":PS=1:GOTO 940
2720 CX=(MX-LX)
2730 CY=(MY-LY)
2740 FOR V=1 TO NP
2750 PX(V)=INT((A1(V)-LX)*(RXS-10)/ABS(CX))+5
2760 PY(V)=RYS-INT((A2(V)-LY)*(RYS-10)/ABS(CY))+5
2770 NEXT V
2830 CLS
2860 '*PLOTTAGGIO VETTORE NORMALIZZATO*
2870 FOR V=1 TO NP
2880 LINE(PX(V)-OF,PY(V))-(PX(V)+OF,PY(V))
2890 LINE(PX(V),PY(V)-OF)-(PX(V),PY(V)+OF)
2900 NEXT V
2950 '****PLOT INTERPOLANTE LINEARE****
2960 YIN=INT(A1*LX+A0)
2970 YFIN=INT(A1*MX+A0)
2980 YIN=RYS-INT(((YIN)-LY)*(RYS-10)/ABS(CY))+5
2990 XIN=5
3000 YFIN=RYS-INT((YFIN-LY)*(RYS-10)/ABS(CY))+5
3010 XFIN=RXS-5
3020 LINE(XIN,YIN)-(XFIN,YFIN)
3500 '**COMMENTO FINE PLOT GRAFICO***
3510 PRINT@0,"F" BEEP BEEP
3520 L$=""
3530 L$=INKEY$ IF L$="" THEN 3530
3540 RETURN
3550 ***********************************
```

Anche in leasing con Olivetti Leasing

LA FAMIGLIA DEI PERSONAL COMPUTER OLIVETTI

FRIENDLY & COMPATIBLE

Questa famiglia di personal compatibili tra loro e con i più diffusi standard internazionali, non ha rivali per espandibilità e flessibilità. Prestazioni che su altri diventano opzionali, sui personal computer Olivetti sono di serie. Per esempio M24 offre uno schermo ad alta definizione grafica, ricco di 16 toni o di 16 colori e con una risoluzione di 600x400 pixel; mentre la sua unità base dispone di 7 slots di espansione, fatto questo che gli consente di accettare schede di espansione standard anche se utilizza un microprocessore a 16 bit reali (INTEL 8086).
Ma ricchi vantaggi offrono anche tutti gli altri modelli.

Basti pensare che tutte le unità base includono sia l'interfaccia seriale che quella parallela. Oppure basti pensare all'ampia gamma di supporti magnetici: floppy da 360 a 720 KB o un'unità hard disk (incorporata o esterna) da 10 MB.
La loro compatibilità, inoltre, fa sì che si possa far uso di una grande varietà di software disponibile sul mercato. Come, ad esempio, la libreria PCOS utilizzabile anche su M24. Come le librerie MS-DOS®, CP/M-86® e UCSD-P System®, utilizzabili sia da M20 che da M21 e M24.

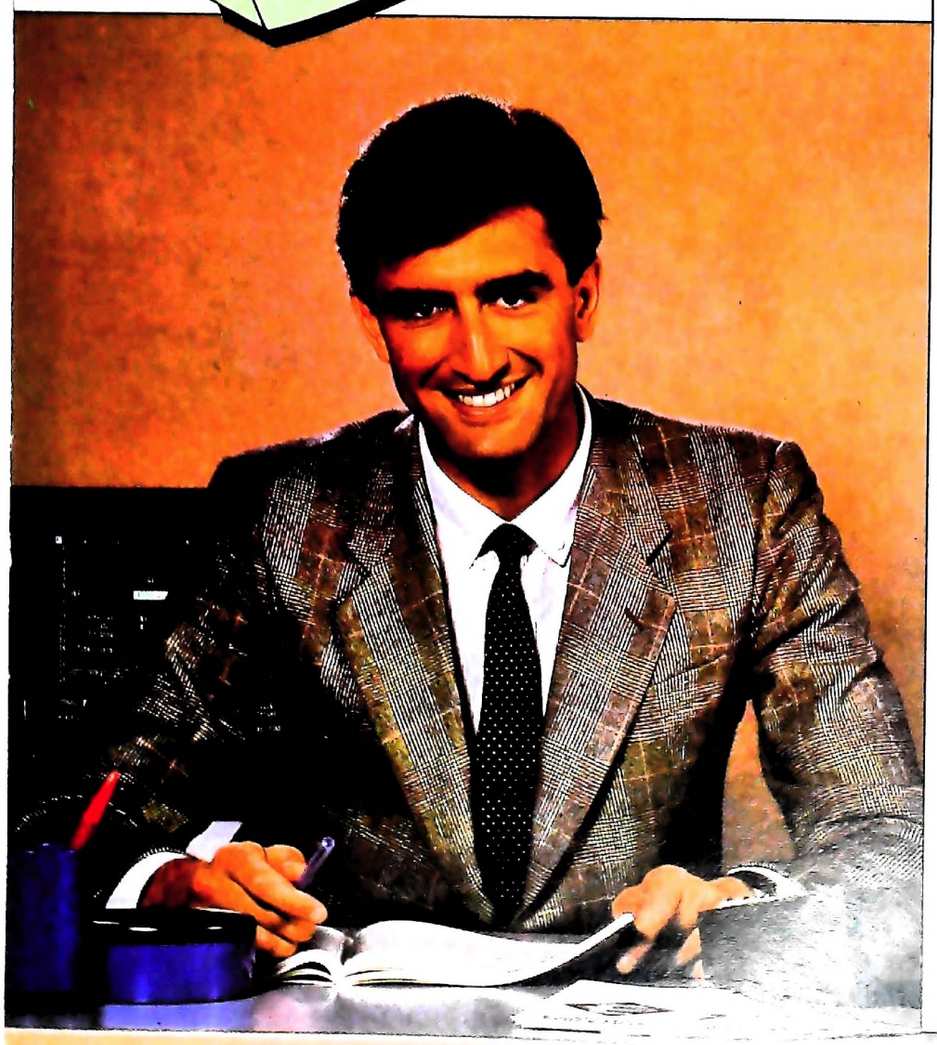
olivetti

Per maggiori informazioni inviare il coupon a Olivetti,
Divisione Personal Computer Via Meravigli 12 20123 Milano

NOME
INDIRIZZO
CITTÀ
TELEFONO

— UN NUOVO MODO DI USARE LA BANCA.—

Consulenza



GLI INVESTIMENTI CON VOI E PER VOI DEL BANCO DI ROMA.

Il Banco di Roma non si limita a custodire i vostri risparmi. Vi aiuta anche a farli meglio fruttare. Come? Mettendovi a disposizione tecnici e analisti in grado di offrirvi una consulenza di prim'ordine e di consigliarvi le forme di investimento piú giuste. Dai certificati di deposito ai titoli di stato, dalle obbligazioni alle azioni, il Banco di Roma vi propone professionalmente le varie opportunità del mercato finanziario. E grazie ai suoi "borsini", vi permette anche di seguire, su speciali video, l'andamento della Borsa minuto per minuto.

Se desiderate avvalervi di una gestione qualificata per investire sui più importanti mercati mobiliari del mondo, i fondi comuni del Banco di Roma, per titoli italiani ed esteri, vi garantiscono una ampia diversificazione.

Inoltre le nostre consociate Figeroma e Finroma forniscono consulenze per una gestione personalizzata del portafoglio e per ogni altra esigenza di carattere finanziario.

Veniteci a trovare, ci conosceremo meglio.

BANCO DI ROMA
CONOSCIAMOCI MEGLIO.