CADEL

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

17 CORSO PRATICO COL COMPUTER

421693

è una iniziativa
FABBRI EDITORI

in collaborazione con
BANCO DI ROMA

e **OLIVETTI**

F4 F5 F6 F7 F8 BATTERY LOW

diretto da GIANNI DE GLI ANTONI

IN OMAGGIO IL SETTIMO POSTER "LA STORIA DELL'INFORMATICA"

FABBRI EDITORI

ELENCO DEI VINCITORI DEL PERSONAL COMPUTER OLIVETTI M 10

Il concorso annunciato con la pubblicazione del primo numero di *Corso Pratico col Computer* si è concluso con l'estrazione di 10 Personal Computer Olivetti M10 avvenuta il 15/6/1984.
A esso hanno partecipato tutte le Personal Card pervenute alla Fabbri entro il 30/5/84.
Pubblichiamo qui di seguito i nomi dei vincitori:

Diego Aqueci - Via Dei Cordai, 6 - CALTAGIRONE (CT)
Giovanni Basco - Via Roma, 75 - PORTICI (NA)
Giovanni Caroppo - Via De Vito, 7/B - BARI
Angelo Conti - Via S. Leonardo, 20 - GUARDIA S. (BN)
Bruno Dagonese - Via Rattazzi, 34 - CAMPOBELLO DI LICATA (AG)
Fabrizio Diamantini - Via Romagna, 60 - FANO (PS)
Giuliana Galimberti - Via Trieste, 54 - MOZZATE (CO)
Luigi Lesquier - Via C. Nigra, 23 - TORINO
Marco Scipioni - Via Degli Alpini - AVEZZANO (AQ)
Roberto Ticci - Via Celle - DICOMANO (FI)

Aut. Min. Conc.

Corso Pratico col Computer - Copyright © sul fascicolo 1984 Gruppo Editoriale Fabbri, Bompiani, Sonzogno, Etas S.p.A., Milano - Copyright © sull'opera 1984 Gruppo Editoriale Fabbri, Bompiani, Sonzogno, Etas S.p.A., Milano - Prima Edizione 1984 - Direttore responsabile GIOVANNI GIOVANNINI - Registrazione presso il Tribunale di Milano n. 135 del 10 marzo 1984. - Iscrizione al Registro Nazionale della Stampa n. 00262, vol. 3, Foglio 489 del 20.9.1982 - Stampato presso lo Stabilimento Grafico del Gruppo Editoriale Fabbri S.p.A., Milano - Diffusione Gruppo Editoriale Fabbri S.p.A. via Mecenate, 91 - tel. 50951 - Milano - Distribuzione per l'Italia: A. & G. Marco s.a.s., via Fortezza 27 - tel. 2526 - Milano - Pubblicazione periodica settimanale - Anno I - n. 17 - esce il giovedì - Spedizione in abb. postale - Gruppo II/70. L'Editore si riserva la facoltà di modificare il prezzo nel corso della pubblicazione, se costretto da mutate condizioni di mercato.

IL MODO MINORE

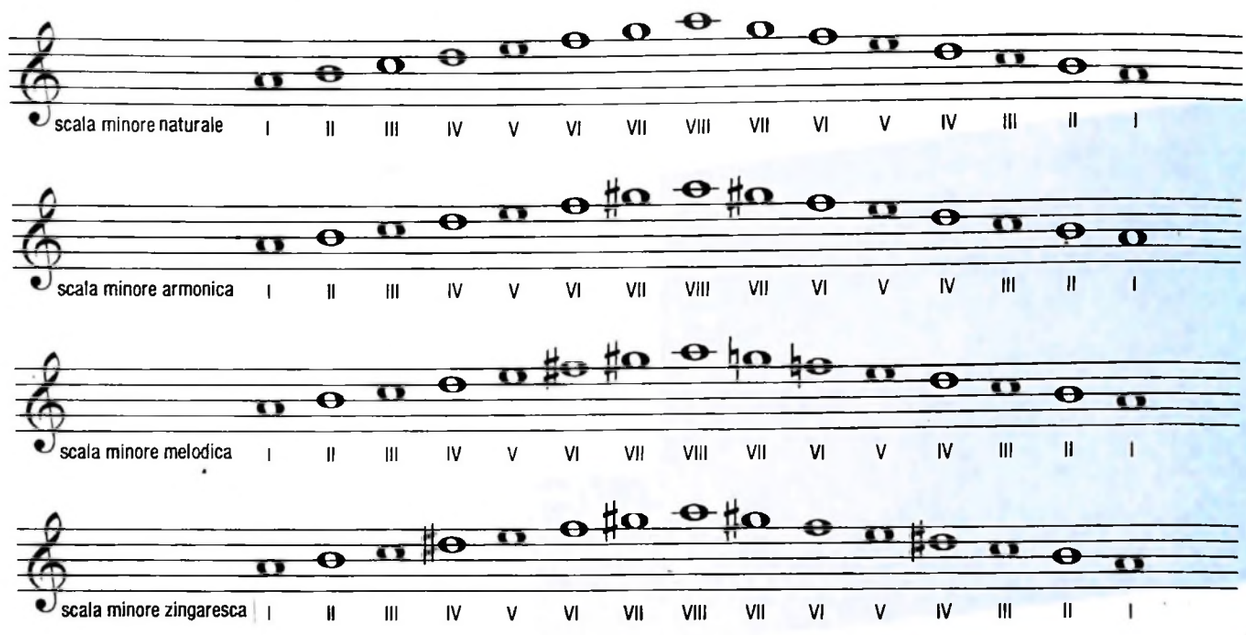
L'esame di un'altra modalità di elaborazione della musica contemporanea derivata dal modo eolico gregoriano.

Due esempi di notazione neumatica: a sinistra una pagina del Responsorio della Resurrezione in notazione del XII secolo (Lucca, Biblioteca Capitolare) e a destra una pagina di un antifonario in notazione del XIII secolo (Carpentras, Bibliothèque Inguimbertine).

Le scale del modo minore

Il modo minore deriva dal modo eolico (uno dei modi gregoriani): la sequenza tipo di intervalli espressa in semitoni è *2-1-2-2-1-2-2*. Rispetto alla scala diatonica possiamo considerare il modo minore come la scala che otteniamo partendo dal sesto grado della scala diatonica; ad esempio, partendo dal *la* si ottiene la scala di *la minore*; questa versione viene detta *naturale* ed è uguale sia quando è ascendente che quando è discendente; nella prassi musicale incontriamo però anche altre scale minori ottenute mediante l'introduzione di alterazioni nella scala minore naturale, come si vede nella figura della pagina seguente: la *scala minore armonica* (*2-1-2-2-1-3-1*, ascendente uguale alla discendente), la *scala minore melodica* (ascendente: *2-1-2-2-2-2-1*; discendente: come la scala minore naturale) e la *scala minore zingaresca* (*2-1-3-1-1-3-1*, ascendente uguale alla discendente).
Per ognuna di queste scale possiamo, come per il modo maggiore, operare delle trasposizioni così da ottenere dodici *tonalità minori* (*do minore, do diesis minore* ecc.).

Notiamo quindi che ad ogni tonalità maggiore corrisponde una tonalità minore la cui tonica è posta tre semitoni (cioè un intervallo di terza minore) sopra quella della tonalità maggiore corrispondente. Ciò che caratterizza una tonalità minore è proprio questo intervallo di terza minore della *mediante*, in contrapposizione all'intervallo di terza maggiore della mediante in una tonalità maggiore; ad esempio, il *la minore* ha come mediante il *do*, mentre la corrispondente tonalità maggiore (il *do maggiore*) ha come mediante il *mi*.
Notiamo inoltre una peculiarità della scala minore melodica: percorsa in senso ascendente è diversa rispetto a quando è percorsa in senso discendente; ciò nasce dal fatto che in senso ascendente si è voluto rendere sensibile l'intervallo tra il settimo e il primo grado (cioè un solo semitono) e per addolcire l'intervallo di seconda aumentata (tre semitoni), presente nella scala minore armonica, è stato alterato anche il sesto grado innalzandolo di un semitono; nel senso discendente si è invece lasciato l'intervallo di un tono intero tra la tonica e la settima appena più bassa e di conseguenza la scala è rimasta uguale alla scala minore naturale.

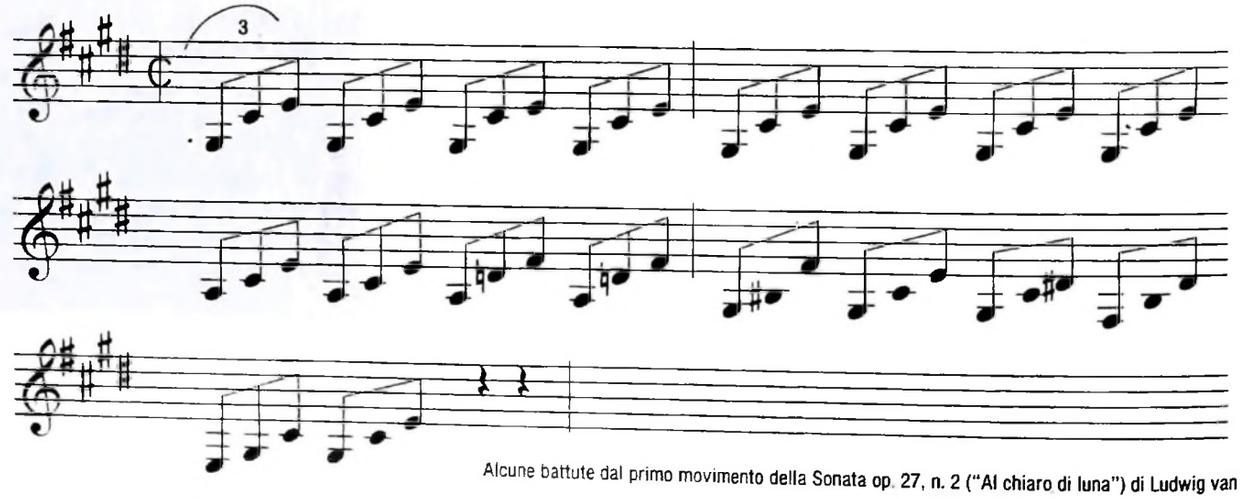


Scale del modo minore nella tonalità di "la".

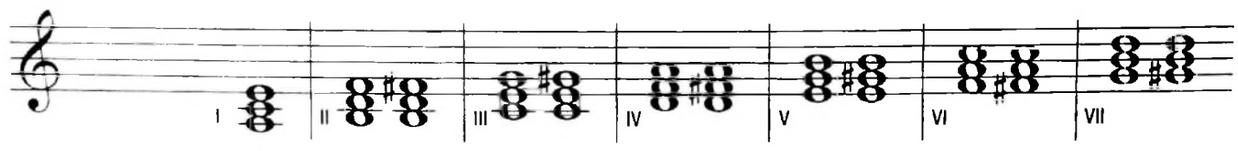
La scala minore melodica ha quindi richiesto l'introduzione di quattro regole (chiamate *leggi dei quattro punti di volta della scala minore*) per usare correttamente il sesto ed il settimo grado della scala e di cui dobbiamo tenere conto nella preparazione di programmi per l'elaborazione di testi musicali in modo minore. Descriviamo brevemente queste leggi come segue (per semplicità espositiva facciamo riferimento alla tonalità di *la minore*):
1° punto di volta (*sol diesis*): il *sol diesis* deve essere seguito dal *la*; non può seguire il *fa*;
2° punto di volta (*fa diesis*): il *fa diesis* deve essere seguito dal *sol diesis*; non può seguire il *fa* o il *sol*;
3° punto di volta (*sol*): il *sol* deve essere seguito dal *fa*; non può seguire il *fa diesis* o il *sol diesis*;
4° punto di volta (*fa*): il *fa* deve essere seguito dal *mi*; non può seguire il *fa diesis*.

Il circolo delle quinte e le triadi nel modo minore

Analogamente a quanto abbiamo visto per il modo maggiore, possiamo tracciare un *circolo delle quinte* anche per il modo minore, stabilendo così i legami di parentela tra le tonalità minori. Si tratta di un circolo del tutto identico a quello del modo maggiore tranne che per il punto di partenza che, invece del *do*, è il *la*. Sulla base delle scale che abbiamo intro-



Alcune battute dal primo movimento della Sonata op. 27, n. 2 ("Al chiaro di luna") di Ludwig van Beethoven.



Triadi costruibili sui sette gradi nel modo minore.

dotto, possiamo costruire le triadi sui sette gradi e vediamo che, a seconda delle alterazioni usate, su alcuni gradi abbiamo più di una triade (illustrazione qui sopra).
Rispetto al modo maggiore abbiamo sei triadi in più; trattiamo gli accordi senza note alterate senza difficoltà come nel modo maggiore; avremo solo delle preferenze nelle successioni delle triadi; per le triadi contenenti note alterate dovremo invece tener conto delle leggi dei punti di volta da cui conseguirà l'impossibilità di alcune successioni (per una trattazione completa dell'argomento si veda il *Manuale di armonia*, di A. Schönberg, Il Saggiatore).

Un esempio

Procedendo nella definizione di concetti musicali è bene che introduciamo parallelamente alcuni concetti informatici fondamentali.
Prendiamo quindi un esempio di brano ben conosciuto in una tonalità minore: l'arpeggio iniziale dal 1° Movimento della Sonata op. 27, n. 2 ("*Al chiaro di luna*") in do diesis minore di Ludwig van Beethoven. Potremmo codificarlo come una sequenza di note con una successione di istruzioni SOUND, una per ogni nota; balza però all'occhio che ci sono delle *ridondanze*, dal punto di vista dell'informazione contenuta nella sequenza dell'arpeggio. Per prima cosa le durate: sono tutte uguali, essendo tutte raggruppate in terzine di tre figure di un ottavo.
Se osserviamo le altezze vediamo che:
a) la prima terzina, *A*, è ripetuta otto volte (prime due battute);
b) la terza battuta è costituita da una seconda terzina, *B*, ripetuta due volte e da una terza terzina, *C*, ripetuta due volte;
c) la quarta battuta è costituita da quattro terzine differenti, *D*, *E*, *F* e *G*;
d) la quinta battuta da due terzine differenti, *H* e *I*.
Allora, possiamo scrivere un programma BASIC come il seguente:

```
010 clear
020 REM codifica altezze
030 G3D=11836
040 C4D=8866
050 E4=7456
060 A3=11172
070 DQ4=8368
080 F4D=6642
090 B3D=9394
100 D4D=7900
110 F3D=13284
120 E3=14912
130 G4D=5918
135 C5D=4433
139 REM impostazione del tempo
140 input "tempo (da 0 a 7)";tmp
150 y=128/2^tmp
160 REM esecuzione del ritornello
170 input "quante ripetizioni";rip
175 if rip=0 then gosub 250 else
    gosub 180
177 gosub 1000
178 end
179 REM struttura ripetitiva
180 for j=1 to rip
190 gosub 250
200 gosub 500
210 gosub 500
220 next j
230 gosub 250
240 return
250 REM struttura del ritornello
260 for i=1 to 8
263 gosub 500
266 next i
270 for i=1 to 2
273 gosub 600
276 next i
280 for i=1 to 2
283 gosub 700
286 next i
290 gosub 800
300 return
499 REM melodia battute 1 e 2
500 sound G3D,y
510 sound C4D,y
520 sound E4,y
530 return
```

```
599 REM melodia battuta 3
    (prima meta')
600 sound A3,y
610 sound C4D,y
620 sound E4,y
630 return
699 REM melodia battuta 3
    (seconda meta')
700 sound A3,y
710 sound DQ4,y
720 sound F4D,y
730 return
799 REM melodia battute 4 e 5
800 sound G3D,y
810 sound B3D,y
820 sound F4D,y
829 REM
830 sound G3D,y
840 sound C4D,y
850 sound E4,y
859 REM
860 sound G3D,y
870 sound C4D,y
880 sound D4D,y
889 REM
890 sound F3D,y
900 sound B3D,y
910 sound D4D,y
919 REM
920 sound E3,y
930 sound G3D,y
940 sound C4D,y
949 REM
950 sound G3D,y
960 sound C4D,y
970 sound E4,y
980 return
1000 REM finale
1010 sound C4D,y
1020 sound E4,y
1030 sound G4D,y
1040 sound E4,y
1050 sound G4D,y
1060 sound C5D,y*3
1070 return
```

Le istruzioni 499-530 sono la terzina *A*; le istruzioni 599-630 sono la terzina *B*; e così via: il blocco 699-730 la terzina *C*, il blocco 799-820 la *D*, fino al blocco 949-980 che costituisce la terzina *I*.
Possiamo poi ricostruire l'intera sequenza con il blocco 250-300, costituito da richiami alle subroutines delle terzine e cicli FOR per le ripetizioni.
Se infine volessimo iterare l'esecuzione di questa sequenza melodica potremmo usare una struttura FOR controllata da un parametro da noi definito (RIP) come nel blocco di istruzioni 179-240; il richiamo al blocco 250-300 produce l'esecuzione del testo originale di Beethoven; per congiungere una esecuzione del ritornello con la successiva abbiamo aggiunto l'esecuzione per due volte della terzina 499-530. Alla fine delle ripetizioni eseguiamo comunque una volta il testo originale; per chiudere l'esecuzione abbiamo aggiunto una chiusura (ci perdoni Beethoven!!) costituita dal blocco 1000-1070.

Un particolare del pianoforte a coda di Beethoven, costruito da Conrad Graf e conservato a Bonn nella casa del musicista.

INFORMATICA E MATEMATICA

Dopo aver presentato le diverse metodologie dell'informatica applicata alla didattica, cominciamo a esaminare le singole discipline a cui l'informatica è strettamente connessa.

È naturale, in una panoramica di questo genere, partire dalla matematica. Diciamo subito che la matematica ha un rapporto speciale con l'informatica, un rapporto che potremmo chiamare privilegiato, dal momento che l'informatica stessa, come disciplina, impiega spesso metodi presi dalla matematica, tipicamente gli algoritmi. Esiste anzi una tendenza che giunge perfino a negare all'informatica la dignità di disciplina autonoma, poiché la vede soltanto come un ramo della matematica. A parte questa posizione estrema, noi crediamo che l'insegnamento dell'informatica sia in se stesso di grande aiuto nei confronti della matematica, e il motivo è semplice. Stendere un programma significa in sostanza tradurre la soluzione di un problema, attraverso la scoperta dell'algoritmo adatto, in un linguaggio riconosciuto da una macchina. È essenziale perciò alla conoscenza dell'informatica la capacità di trovare algoritmi risolutivi dei problemi e insieme la capacità di tradurli in modo che siano accettati dal sistema che dovrà eseguirli. La scoperta degli algoritmi non è certo dovuta all'informatica, eppure all'informatica va il merito di costringere alla loro stesura assolutamente corretta in modo che possano essere riconosciuti ed eseguiti in forma del tutto automatica.

```
INIZIO
   Introduzione dei numeri N0 e N1,
        valori limite entro cui calcolare
        i numeri primi
   Controllo accettabilità dati (N0>=5;
        N0 intero; N0 dispari; N0<N1)
   ESEGUI VARIANDO N DA N0 A N1 PASSO 2
      ESEGUI VARIANDO D DA 3 A √N PASSO 2
         SE N è divisibile per D
            ALLORA Uscita dal ciclo su D e
                   passaggio al valore
                   successivo di N
         FINESE
      RIPETI
      Visualizzazione N
   RIPETI
FINE
```

Generazione dei numeri primi

Vediamo per esempio, qui sopra a destra, un algoritmo piuttosto semplice, che genera i numeri primi, entro un intervallo qualunque, stabilito a piacere dall'utente. Lo abbiamo scritto anzitutto nel linguaggio della programmazione strutturata, di cui si è parlato precedentemente.
Come si vede si tratta di un algoritmo abbastanza semplice e facile da scrivere. Notiamo comunque ancora una volta che l'uso della programmazione strutturata permette la stesura dell'algoritmo con una notevole chiarezza di impostazione. Risulta facile ora tradurlo in BASIC, in un programma scritto per l'elaboratore M10 della Olivetti, come possiamo vedere nella pagina seguente.
Notiamo che, per ottenere la stampa dei numeri primi richiesti anziché la semplice visualizzazione, è sufficiente alla riga 200 scrivere LPRINT invece di PRINT.
A questo punto è quasi immediato cogliere la conseguenza più evidente dell'uso dell'informatica per la matematica: occorre cambiare il tipo di insegnamento. Fino ad oggi la matematica è stata vista soprattutto come complesso di sistemi ipotetico-deduttivi e quindi l'insistenza maggiore è riservata solitamente agli aspetti logico-formali, ad esempio la dimostrazione dei teoremi. L'informatica richiede invece di riservare molta più attenzione agli aspetti algoritmico-costruttivi da impiegare per la soluzione dei problemi. E non si tratta di una osservazione banale: è tutta una mentalità che deve essere cambiata, soprattutto per lasciare spazio alla creatività degli allievi nello scoprire l'algoritmo risolutivo, nel modificarlo adattandolo alle esigenze del problema concreto a cui è riferito, nello stenderlo in modo corretto e nel verificarne il funzionamento sulla macchina.

Programma per generare i numeri primi

```
10 REM Programma NUMERI PRIMI
20 CLS
30 PRINT "INTRODURRE I VALORI LIMITE ENTRO CUI"
40 PRINT "CALCOLARE I NUMERI PRIMI"
50 PRINT "IL VALORE MINIMO DEVE ESSERE MAGGIORE"
60 PRINT "O UGUALE A 5 E DISPARI"
70 INPUT NO,N1
80 IF NO<5 OR INT(NO)<NO OR INT(NO/2)=NO/2 THEN 90
   ELSE 120
90 PRINT "IL PRIMO NUMERO DEVE ESSERE INTERO"
100 PRINT "MAGGIORE O UGUALE A 5 E DISPARI"
110 GOTO 70
120 IF N1<=NO THEN 130 ELSE 150
130 PRINT "IL SECONDO NUMERO DEVE ESSERE MAGGIORE
    DEL PRIMO"
140 GOTO 70
150 CLS
160 FOR N=NO TO N1 STEP 2
170 FOR D=3 TO SQR(N) STEP 2
180 IF INT(N/D)=N/D THEN 210
190 NEXT D
200 PRINT N;
210 NEXT N
220 END
```

Calcolo della derivata in un punto

Vediamo un altro programma scritto in questa prospettiva e precisamente per il calcolo del valore della derivata in un punto come limite del rapporto incrementale. Presentiamo anzitutto, qui a lato, la struttura dell'algoritmo scritta ancora con gli strumenti della programmazione strutturata.
Notiamo che F(h) significa $(f(x_0+h) - f(x_0))/h$.
Anche in questo caso, la programmazione strutturata rende immediatamente leggibile l'algoritmo, il quale è basato sulla replicazione del calcolo di F(h) con h decrescente fino al punto in cui si ha una differenza minima tra i due valori consecutivi di F(h). Si prende allora come valore della derivata il penultimo valore calcolato di F(h) e si visualizza.
Vediamo in basso in questa pagina il programma scritto per l'elaboratore M10 della Olivetti.
Notiamo che anche in questo caso, volendo la stampa dei risultati anziché la semplice visualizzazione, è sufficiente, alla riga 170, sostituire PRINT con LPRINT.
Abbiamo inserito come funzione la seguente: $y = e^x$ a scopo di verifica. Com'è noto, infatti, essa è in ogni punto esattamente uguale alla propria derivata.
È chiaro che questo programma può essere ulteriormente arricchito, anzitutto con un ciclo che esegua il calcolo della derivata per una serie di valori entro un intervallo fissato. Inoltre si potrebbe far intervenire la grafica per visualizzare, entro un dato intervallo, tanto la funzione data quanto la funzione derivata.

```
INIZIO
   Introduzione valore x0
   Calcolo del rapporto incrementale F(h0)
        (con h0=1)
   Calcolo del rapporto incrementale F(h1)
        (con h1=1/10)
   Calcolo di D1= ||F(h1)| - |F(h0)||
   N=1
   ESEGUI
      N=N+1
      Calcolo F(hn)
      Calcolo Dn= ||F(hn)| - |F(hn-1)||
   RIPETI MENTREVAL Dn < Dn-1
   f'(x0) = F(hn-1)
   Calcolo di f(x0)
   Visualizzazione di x0 , f(x0) , f'(x0)
FINE
```

Programma per calcolare la derivata in un punto

```
10 REM Programma CALCOLO DERIVATA IN UN PUNTO
20 PRINT "INTRODUZIONE VALORE DELLA VARIABILE"
30 PRINT "INDIPENDENTE"
40 INPUT X
50 RO=EXP(X+1)-EXP(X)
60 R1=(EXP(X+1/10)-EXP(X))/10
70 D1=ABS(ABS(R1)-ABS(RO))
80 N=1
90 RO=R1
100 DO=D1
110 N=N+1
120 H=1/10^N
130 R1=(EXP(X+H)-EXP(X))/H
140 D1=ABS(ABS(R1)-ABS(RO))
150 IF D1<DO THEN 90
160 CLS
170 PRINT X,EXP(X);RO
180 END
```

Programma per generare i numeri primi

```
10 REM Programma NUMERI PRIMI
20 CLS
30 PRINT "INTRODURRE I VALORI LIMITE ENTRO CUI"
40 PRINT "CALCOLARE I NUMERI PRIMI"
50 PRINT "IL VALORE MINIMO DEVE ESSERE MAGGIORE"
60 PRINT "O UGUALE A 5 E DISPARI"
70 INPUT NO,N1
80 IF NO<5 OR INT(NO)<>NO OR INT(NO/2)=NO/2 THEN 90
   ELSE 120
90 PRINT "IL PRIMO NUMERO DEVE ESSERE INTERO"
100 PRINT "MAGGIORE O UGUALE A 5 E DISPARI"
110 GOTO 70
120 IF N1<=NO THEN 130 ELSE 150
130 PRINT "IL SECONDO NUMERO DEVE ESSERE MAGGIORE
    DEL PRIMO"
140 GOTO 70
150 CLS
160 FOR N=NO TO N1 STEP 2
170 FOR D=3 TO SQR(N) STEP 2
180 IF INT(N/D)=N/D THEN 210
190 NEXT D
200 PRINT N;
210 NEXT N
220 END
```

Calcolo della derivata in un punto

Vediamo un altro programma scritto in questa prospettiva e precisamente per il calcolo del valore della derivata in un punto come limite del rapporto incrementale. Presentiamo anzitutto, qui a lato, la struttura dell'algoritmo scritta ancora con gli strumenti della programmazione strutturata.
Notiamo che F(h) significa $(f(x_0+h) - f(x_0))/h$.
Anche in questo caso, la programmazione strutturata rende immediatamente leggibile l'algoritmo, il quale è basato sulla replicazione del calcolo di F(h) con h decrescente fino al punto in cui si ha una differenza minima tra i due valori consecutivi di F(h). Si prende allora come valore della derivata il penultimo valore calcolato di F(h) e si visualizza.
Vediamo in basso in questa pagina il programma scritto per l'elaboratore M10 della Olivetti.
Notiamo che anche in questo caso, volendo la stampa dei risultati anziché la semplice visualizzazione, è sufficiente, alla riga 170, sostituire PRINT con LPRINT.
Abbiamo inserito come funzione la seguente: $y = e^x$ a scopo di verifica. Com'è noto, infatti, essa è in ogni punto esattamente uguale alla propria derivata.
È chiaro che questo programma può essere ulteriormente arricchito, anzitutto con un ciclo che esegua il calcolo della derivata per una serie di valori entro un intervallo fissato. Inoltre si potrebbe far intervenire la grafica per visualizzare, entro un dato intervallo, tanto la funzione data quanto la funzione derivata.

```
INIZIO
    Introduzione valore x0
    Calcolo del rapporto incrementale F(h0)
        (con h0≠1)
    Calcolo del rapporto incrementale F(h1)
        (con h1=1/10)
    Calcolo di D1= ||F(h1)| - |F(h0)||
    N=1
    ESEGUI
        N=N+1
        Calcolo F(hn)
        Calcolo Dn= ||F(hn)| - |F(hn-1)||
    RIPETI MENTREVAL Dn < Dn-1
    f'(x0) = F(hn-1)
    Calcolo di f(x0)
    Visualizzazione di x0 , f(x0) , f'(x0)
FINE
```

Programma per calcolare la derivata in un punto

```
10 REM Programma CALCOLO DERIVATA IN UN PUNTO
20 PRINT "INTRODUZIONE VALORE DELLA VARIABILE"
30 PRINT "INDIPENDENTE"
40 INPUT X
50 R0=EXP(X+1)-EXP(X)
60 R1=(EXP(X+1/10)-EXP(X))/10
70 D1=ABS(ABS(R1)-ABS(R0))
80 N=1
90 R0=R1
100 D0=D1
110 N=N+1
120 H=1/10^N
130 R1=(EXP(X+H)-EXP(X))/H
140 D1=ABS(ABS(R1)-ABS(R0))
150 IF D1<D0 THEN 90
160 CLS
170 PRINT X,EXP(X);R0
180 END
```

Programma per generare i numeri primi

```
10 REM Programma NUMERI PRIMI
20 CLS
30 PRINT "INTRODURRE I VALORI LIMITE ENTRO CUI"
40 PRINT "CALCOLARE I NUMERI PRIMI"
50 PRINT "IL VALORE MINIMO DEVE ESSERE MAGGIORE"
60 PRINT "O UGUALE A 5 E DISPARI"
70 INPUT NO,N1
80 IF NO<5 OR INT(NO)<>NO OR INT(NO/2)=NO/2 THEN 90
   ELSE 120
90 PRINT "IL PRIMO NUMERO DEVE ESSERE INTERO"
100 PRINT "MAGGIORE O UGUALE A 5 E DISPARI"
110 GOTO 70
120 IF N1<=NO THEN 130 ELSE 150
130 PRINT "IL SECONDO NUMERO DEVE ESSERE MAGGIORE
    DEL PRIMO"
140 GOTO 70
150 CLS
160 FOR N=NO TO N1 STEP 2
170 FOR D=3 TO SQR(N) STEP 2
180 IF INT(N/D)=N/D THEN 210
190 NEXT D
200 PRINT N;
210 NEXT N
220 END
```

Calcolo della derivata in un punto

Vediamo un altro programma scritto in questa prospettiva e precisamente per il calcolo del valore della derivata in un punto come limite del rapporto incrementale. Presentiamo anzitutto, qui a lato, la struttura dell'algoritmo scritta ancora con gli strumenti della programmazione strutturata.
Notiamo che F(h) significa $(f(x_0+h) - f(x_0))/h$.
Anche in questo caso, la programmazione strutturata rende immediatamente leggibile l'algoritmo, il quale è basato sulla replicazione del calcolo di F(h) con h decrescente fino al punto in cui si ha una differenza minima tra i due valori consecutivi di F(h). Si prende allora come valore della derivata il penultimo valore calcolato di F(h) e si visualizza.
Vediamo in basso in questa pagina il programma scritto per l'elaboratore M10 della Olivetti.
Notiamo che anche in questo caso, volendo la stampa dei risultati anziché la semplice visualizzazione, è sufficiente, alla riga 170, sostituire PRINT con LPRINT.
Abbiamo inserito come funzione la seguente: $y=e^x$ a scopo di verifica. Com'è noto, infatti, essa è in ogni punto esattamente uguale alla propria derivata.
È chiaro che questo programma può essere ulteriormente arricchito, anzitutto con un ciclo che esegua il calcolo della derivata per una serie di valori entro un intervallo fissato. Inoltre si potrebbe far intervenire la grafica per visualizzare, entro un dato intervallo, tanto la funzione data quanto la funzione derivata.

```
INIZIO
    Introduzione valore x0
    Calcolo del rapporto incrementale F(h0)
        (con h0=1)
    Calcolo del rapporto incrementale F(h1)
        (con h1=1/10)
    Calcolo di D1= ||F(h1)| - |F(h0)||
    N=1
    ESEGUI
        N=N+1
        Calcolo F(hn)
        Calcolo Dn= ||F(hn)| - |F(hn-1)||
    RIPETI MENTREVAL Dn < Dn-1
    f'(x0) = F(hn-1)
    Calcolo di f(x0)
    Visualizzazione di x0 , f(x0) , f'(x0)
FINE
```

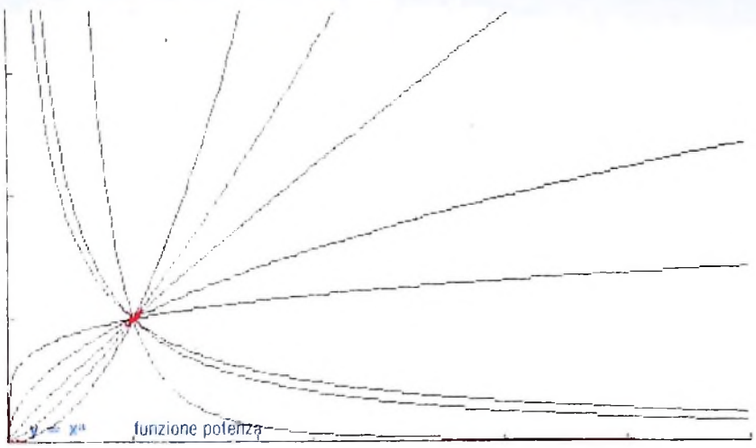
Programma per calcolare la derivata in un punto

```
10 REM Programma CALCOLO DERIVATA IN UN PUNTO
20 PRINT "INTRODUZIONE VALORE DELLA VARIABILE"
30 PRINT "INDIPENDENTE"
40 INPUT X
50 R0=EXP(X+1)-EXP(X)
60 R1=(EXP(X+1/10)-EXP(X))/10
70 D1=ABS(ABS(R1)-ABS(R0))
80 N=1
90 R0=R1
100 D0=D1
110 N=N+1
120 H=1/10^N
130 R1=(EXP(X+H)-EXP(X))/H
140 D1=ABS(ABS(R1)-ABS(R0))
150 IF D1<D0 THEN 90
160 CLS
170 PRINT X;EXP(X);R0
180 END
```

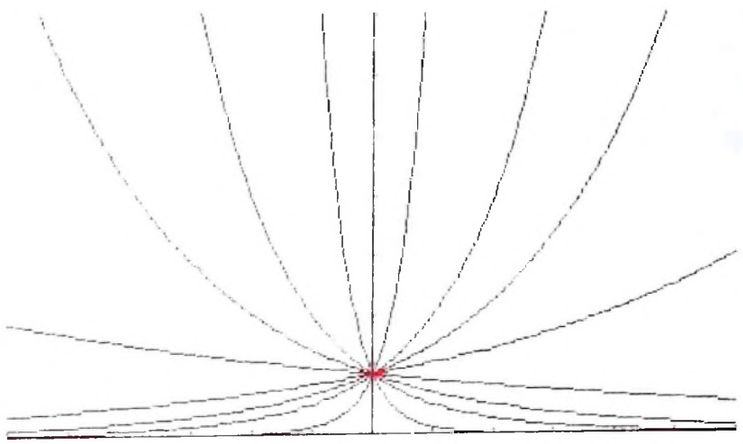
In tal caso lo studente potrebbe eseguire una analisi precisa dei rapporti tra le due funzioni: funzione crescente e derivata positiva, funzione decrescente e derivata negativa, massimi, minimi o flessi a tangente orizzontale e derivata nulla eccetera. Abbiamo citato la grafica: è tutto un campo aperto alle applicazioni della didattica della matematica, soprattutto per la visualizzazione dell'andamento delle funzioni e il relativo studio, come è stato illustrato in uno dei numeri precedenti.

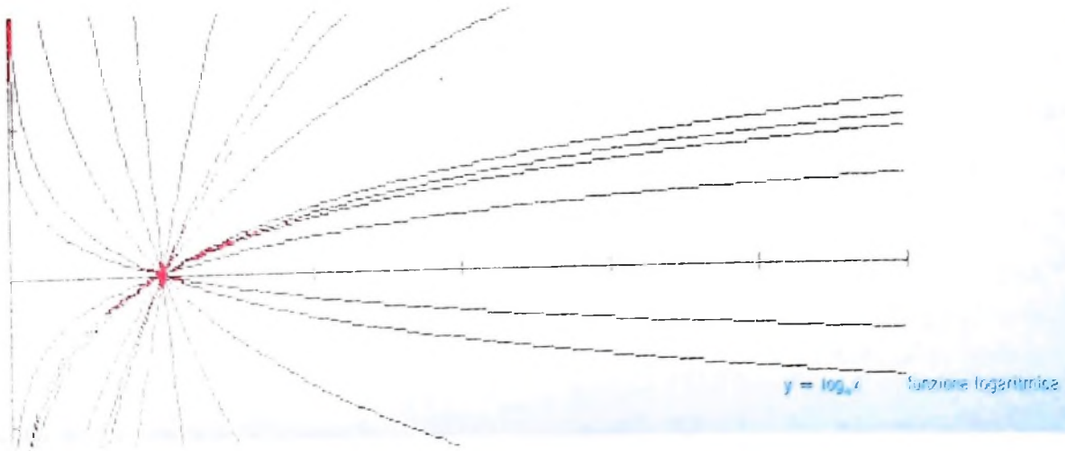
Analisi delle funzioni più usate in matematica

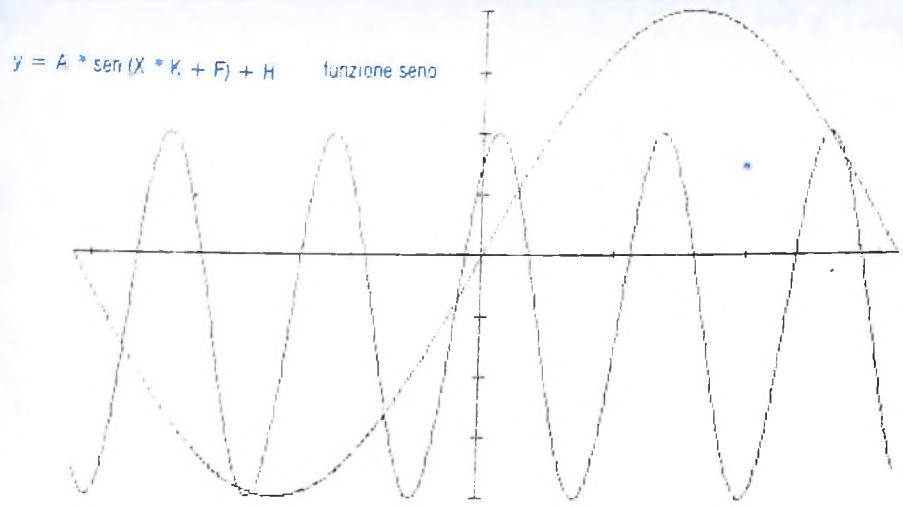
A questo proposito vogliamo presentare un altro programma, che serve a visualizzare le funzioni più usate nella matematica e precisamente la funzione potenza, l'esponenziale, il logaritmo, il seno e il coseno.
Del programma, che funziona sull'elaboratore M20 della Olivetti, mostriamo i risultati.



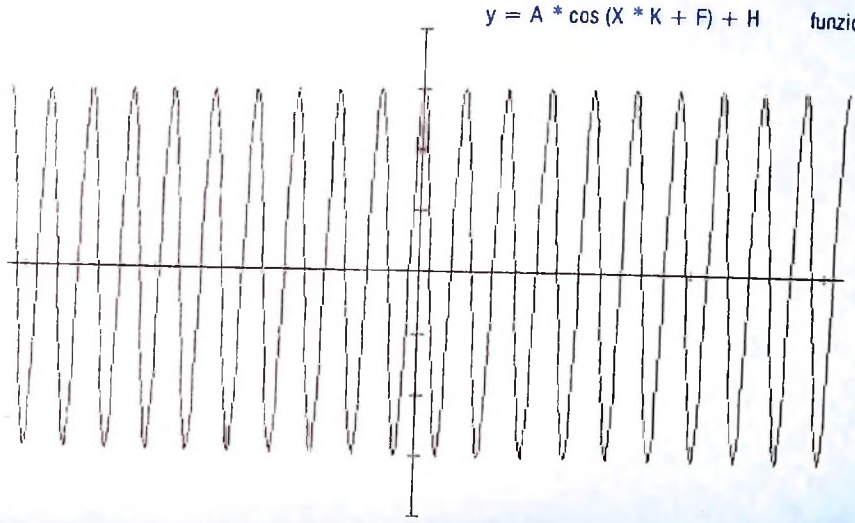
I risultati di un programma per M20, finalizzato alla visualizzazione delle funzioni più usate in matematica. Le funzioni rappresentate qui sono la funzione potenza, l'esponenziale e il logaritmo (dall'alto in basso).







Ancora due funzioni visualizzate dal programma matematico per M20: sono la funzione seno (in alto) e la funzione coseno (in basso).



Nel caso delle prime tre funzioni l'utente deve scegliere la scala e quindi il valore rispettivamente dell'esponente della potenza oppure della base dell'esponenziale o del logaritmo. L'elaboratore visualizza l'andamento della funzione. A sua scelta l'utente può cancellare lo schermo e ripartire da capo oppure può visualizzare altre funzioni dello stesso tipo cambiando i rispettivi parametri.
I risultati sono visualizzati dai grafici che riportano più funzioni contemporaneamente e riproducono esattamente quello che appare sullo schermo.
Nel caso delle altre due funzioni, seno e coseno, la scala è già fissata automaticamente; l'utente deve solo scegliere i quattro parametri che intervengono nella definizione.
Più precisamente, date le funzioni:

$Y = A * \text{sen}(X * K + F) + H$

$Y = A * \cos(X * K + F) + H$

i parametri da fissare di volta in volta sono A, K, F, H.
Nel primo esempio abbiamo riportato due funzioni, nel secondo una sola.

Le due funzioni del primo grafico sono definite dai seguenti parametri: A = 1; K = 1; F = 0; H = 0, che rappresentano una normale funzione seno.
L'altra funzione è definita dai seguenti parametri: A = 3; K = 5; F = 1; H = –1.
Nel secondo grafico abbiamo la funzione coseno rappresentata con i seguenti parametri: A = 3; K = 20; F = 0; H = 0.
Abbiamo volutamente scelto esempi piuttosto semplici e facilmente comprensibili; il campo comunque resta aperto ad applicazioni di qualunque livello, fino alle più complesse.
Riteniamo in ogni caso essenziale che per la completa formazione dell'allievo non vengano proposti soltanto programmi preconfezionati e pronti per l'uso, ma che l'allievo abbia una parte non secondaria anche nella preparazione dei programmi stessi, almeno di quelli semplici. L'allievo, imparando matematica con l'elaboratore, deve rendersi conto dei meccanismi che entrano in gioco nella preparazione dei programmi, soprattutto deve imparare a trovare di volta in volta l'algoritmo necessario a risolvere il problema che gli viene posto. Soltanto dopo può anche usare altri programmi, più complessi, già pronti.

Lezione 16

Sul concetto di tipo

Abbiamo visto fin qui variabili di tipo numerico e di tipo stringa. Questi non sono però gli unici tipi di dati con cui ci troveremo a trattare; pensiamo per esempio di dover ordinare un mazzo di carte: le variabili su cui operare in tal caso assumerebbero i valori "asso", "due", "tre", ... "jack", "donna", "re", su cui è definita una relazione che permette di stabilire, per esempio, che il re è "maggiore" della donna e, conseguentemente, la segue in un mazzo ordinato. Immaginiamo anche il caso di un programma che fornisca i prezzi di alcune vernici sulla base del loro colore. Un prezziario del genere potrebbe essere realizzato dalla seguente tabella:

Rosso	10 000
Blu	12 000
Verde	8 000
Giallo	15 000
Nero	12 000

che ha come indici i "nomi" dei colori e contiene in ogni elemento il prezzo della vernice del colore corrispondente. Così, volendo conoscere il prezzo della vernice gialla, basta leggere l'elemento della tabella di indice "giallo" e si otterrà immediatamente il corrispondente prezzo.
Abbiamo già visto esempi di programmi in cui l'esecuzione è guidata da condizioni espresse tramite variabili che assumono il valore "sì" o "no": per esempio, programmi di ricerca in tabella, dove l'iterazione della ricerca viene interrotta quando una variabile di nome TROVATO assume il valore "sì". Molto spesso è utile introdurre variabili come questa per controllare l'esecuzione: queste variabili assumono uno tra due valori, che possono anche essere espressi come "vero" e "falso" o, in inglese, "true" e "false". Facendo riferimento all'algebra booleana, che è appunto basata su due valori, 0 e 1, vengono normalmente dette variabili booleane.
Se, ancora, pensiamo a un programma che calcoli l'area di alcune figure geometriche (quadrato, rettangolo, trapezio, pentagono, cerchio) potremmo pensare di utilizzare una variabile che assuma i valori quadrato, rettangolo, trapezio ecc. in modo da costruire un programma come il seguente:

```
SE F=QUADRATO ALLORA
   calcola l'area del quadrato
SE F=RETTANGOLO ALLORA
   calcola l'area del rettangolo
SE F=TRAPEZIO ALLORA
   calcola l'area del trapezio
```

e così via.
Un programma così fatto userebbe quindi una variabile i cui valori sono appunto quadrato, rettangolo, trapezio, pentagono e cerchio.
Poiché questi sono i valori della variabile F sarebbero anche possibili operazioni di

Completata questa sedicesima lezione del Corso di Programmazione e BASIC, siete in grado di eseguire gli esercizi

PRIMIT. DO
PRIMIP. BA

contenuti nella cassetta "5 esercizi di programmazione", lato A.

I titoli seguiti dal suffisso DO corrispondono a testi, quelli seguiti da BA a programmi in BASIC.
Caricateli secondo le modalità che avete appreso.

assegnamento come per esempio la seguente:

```
F:=RETTANGOLO
```

In generale, diremo che la variabile F può assumere i seguenti valori: quadrato, rettangolo, trapezio, pentagono, cerchio.
Si noti che non si tratta di stringhe di caratteri, ma di veri e propri simboli, così come "1" (carattere) è diverso da 1 intero.
In sostanza abbiamo elencato i valori ammessi per la variabile F e consideriamo che questi e solo questi siano i valori che possono essere correttamente assegnati a tale variabile. È facile capire quanto possa essere comodo poter attribuire esattamente i valori reali alle variabili che usiamo nei programmi: per questo i linguaggi di programmazione più avanzati forniscono la possibilità di definire nuovi tipi di dati secondo le esigenze specifiche del problema che si vuole risolvere.

I valori interi

Una variabile intera, denotata con il simbolo % come terminatore del nome, ha le seguenti caratteristiche:

massimo valore rappresentabile: 32767
minimo valore rappresentabile: —32768
occupazione di memoria: 2 byte per un totale di 16 bit.

Si tratta quindi di una rappresentazione numerica che occupa, rispetto agli altri tipi, molto poco.

La definizione di tipi enumerativi in PASCAL

In PASCAL è possibile definire nuovi tipi di dati, "enumerando" i valori corrispondenti. Così, nel caso del programma che costruisce il prezziario delle vernici, l'indice della tabella verrebbe descritto nel modo seguente:

```
VAR INDEX: (ROSSO, BLU, VERDE, GIALLO, NERO)
```

supposto che questi fossero i colori disponibili. La dichiarativa riportata indica al compilatore PASCAL che nel programma verrà usata una variabile che può assumere uno tra i valori riportati tra parentesi. Allo stesso modo la variabile booleana verrebbe descritta nel modo seguente:

```
VAR TROVATO: (VERO, FALSO)
```

e la variabile CARTADAGIOCO nel modo seguente:

```
VAR CARTADAGIOCO: (ASSO, DUE, TRE, QUATTRO, CINQUE, SEI, SETTE,
OTTO, NOVE, DIECI, JACK, DONNA, RE)
```

I valori reali in singola precisione

Le variabili che contengono tali tipi di valori sono denotate dal carattere "!" come terminatore dell'identificatore.

Le caratteristiche di tali valori sono:

massimo valore assoluto rappresentabile:
ordine di grandezza di 10^62
(cioè numeri seguiti da 62 zeri, sia che siano positivi o negativi)

ES. 3.2*10^62
-.2*10^62

minimo valore assoluto rappresentabile:
ordine di grandezza di 10^-64
(positivi o negativi)

ES. 3.2*10^-64
-1.7*10^-64

precisione:
riesce a rappresentare 7 cifre significative, ma richiedendone la visualizzazione ne vengono fornite solo 6, essendo la sesta arrotondata sulla base del valore della settima

occupazione:
una variabile di questo tipo occupa 4 byte, per un totale di 32 bit.

Valori reali in doppia precisione

Si tratta di valori che sono correlati a variabili che non hanno uno specifico terminatore nel nome, o che hanno il terminatore "III".

Le loro caratteristiche sono:

massimo valore assoluto rappresentabile:
ordine di grandezza di 10^62
(come per la precisione semplice)

minimo valore assoluto rappresentabile:
ordine di grandezza di 10^-64
(come per la precisione semplice)

precisione:
riesce a rappresentare 16 cifre significative, ma, a una richiesta di visualizzazione, ne mostra solo le prime 15, arrotondando la quindicesima sulla base del valore dell'ultima

occupazione:
questo tipo di dato occupa 8 byte per un totale di 64 bit.

Si noti che è anche il tipo che richiede più tempo di esecuzione per i calcoli.

Poiché possiamo immaginare un numero pressoché infinito di differenti tipi dipendenti dal problema specifico, i linguaggi di programmazione non li mettono direttamente a disposizione, ma possono soltanto mettere l'utente in condizione di costruire i tipi di dati che di volta in volta gli sono necessari.
In qualche caso può succedere che in uno stesso programma si usino più variabili dello stesso tipo definito dall'utente.
Per esempio, possiamo immaginare che il programma per il listino prezzi usi una variabile per scorrere la tabella e una per memorizzare il colore della vernice di cui si vuole il prezzo.
In tal caso entrambe le variabili dovrebbero essere descritte enumerando per ciascuna i valori ammessi.
Ciò potrebbe risultare scomodo e appesantire la lista del programma. Per questo motivo il PASCAL consente di attribuire un nome ai tipi di dati definiti dall'utente nel modo seguente:

```
TYPE COLORE = (ROSSO, BLU, VERDE, GIALLO, NERO)
```

In questo modo si è reso disponibile un nuovo tipo, che da questo momento in avanti potrà essere usato per descrivere le variabili del programma esattamente come i tipi resi disponibili dal linguaggio. Potremo pertanto dichiarare le variabili nel modo seguente:

```
VAR INDEX, VERNICE: COLORE
```

che indica al compilatore che verranno usate due variabili, rispettivamente di nome INDEX e VERNICE, di tipo COLORE, che possono cioè assumere i valori definiti dal tipo COLORE.
In sostanza non abbiamo fatto altro che attribuire un nome mnemonico a un insieme di valori, assicurandoci così la possibilità di indicarli nel loro complesso senza per altro doverli ogni volta ripetere uno per uno.
Non è diverso da quello che facciamo quando diciamo che una variabile è di tipo intero: con questo termine, infatti, identifichiamo tutti i valori interi, che dovremmo altrimenti enumerare.

Cosa abbiamo imparato

In questa lezione abbiamo visto:

- Il concetto di tipo enumerativo;
- Il concetto di tipo definito dall'utente;
- La definizione di tipo e la dichiarazione di variabili in Pascal;
- I tipi intero, reale in singola precisione, reale in doppia precisione e in BASIC.

SULLA PERCEZIONE VISIVA

Come trasferire nelle applicazioni informatiche i risultati degli studi sulle capacità di percepire e memorizzare immagini e dati.

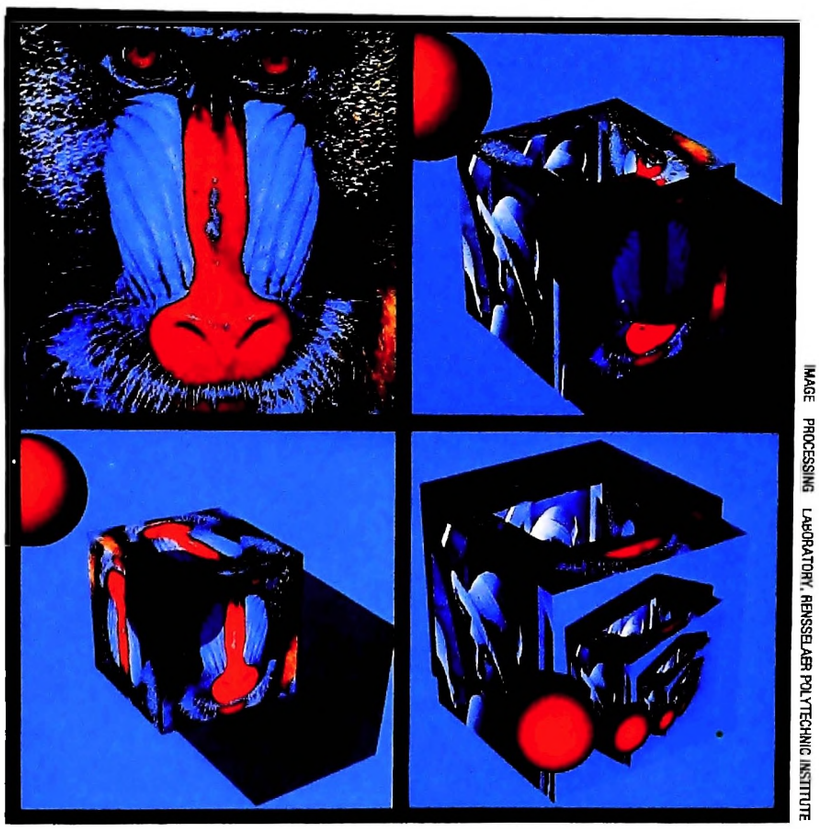


IMAGE PROCESSING LABORATORY, RENSSELAER POLYTECHNIC INSTITUTE

Un'immagine paradossale, e quindi difficile da percepire: la rigidità geometrica del cubo contrasta con la ricchezza di particolari contenuta nell'immagine scimmiesca su ogni sua faccia. Per di più, il cervello, agendo in base al contesto, tende ad attribuirvi una profondità spaziale che qui non esiste.

Come si è visto precedentemente, uno dei maggiori problemi incontrati nel campo della Intelligenza Artificiale è stato quello della comunicazione di concetti, idee e informazioni. Oltre alla comunicazione linguistica, il principale strumento di acquisizione e trasmissione di informazione, per un sistema intelligente, è quello visuale.

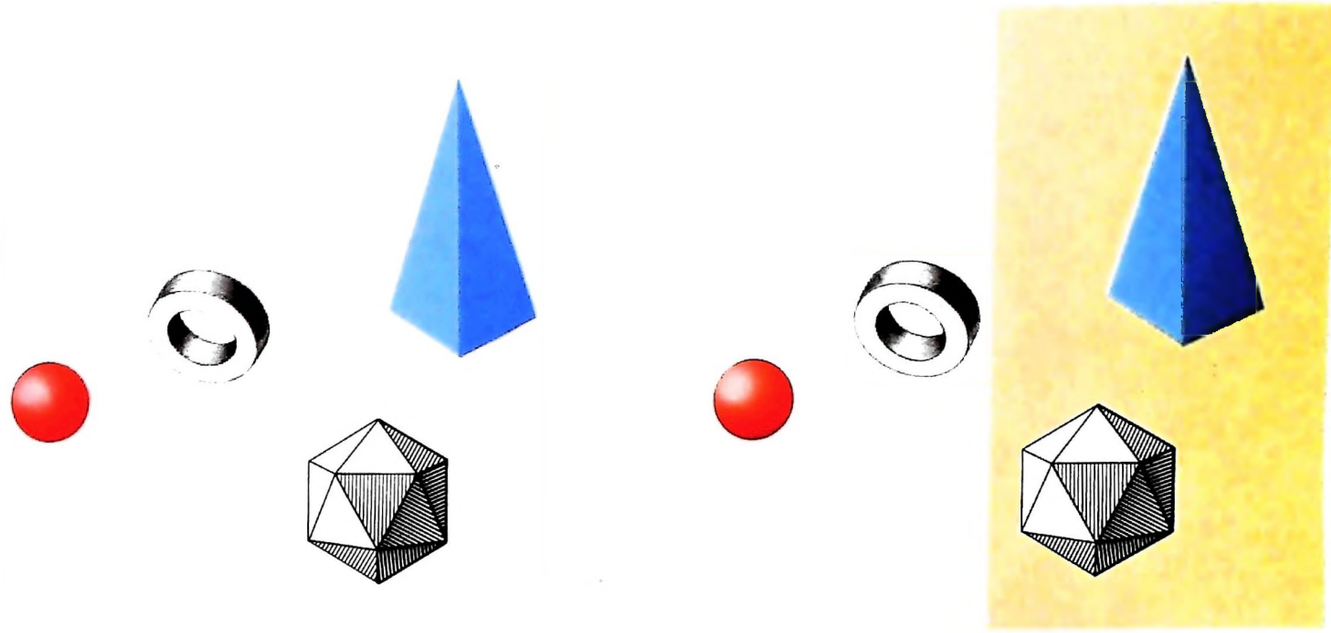
La ricerca nel campo dell'interfaccia uomo-macchina, supportata da scoperte in altri settori, ha portato alla comprensione dei vasti confini del problema ed alla proposta di soluzioni rilevanti.

A partire da questi studi si è compreso in che modo la comunicazione stessa con il calcolatore dovesse essere migliorata e si è quindi presentata l'esigenza di definire come la rappresentazione dell'informazione sui display dovesse essere organizzata.

I computer hanno architetture e sistemi operativi che determinano il loro modo di elaborare ed organizzare le informazioni. Gli uomini hanno essi stessi un modo di usare ed organizzare le medesime. L'interazione attraverso questi due modi distinti e diversi di vedere le informazioni avviene attraverso i display dei computer, che devono quindi essere riprogettati e riorganizzati tenendo conto sia dei risultati degli studi di A.I., sia dei risultati ottenuti dalla psicologia, e anche dalle tecniche di organizzazione delle immagini che sono tipiche del mondo pubblicitario.

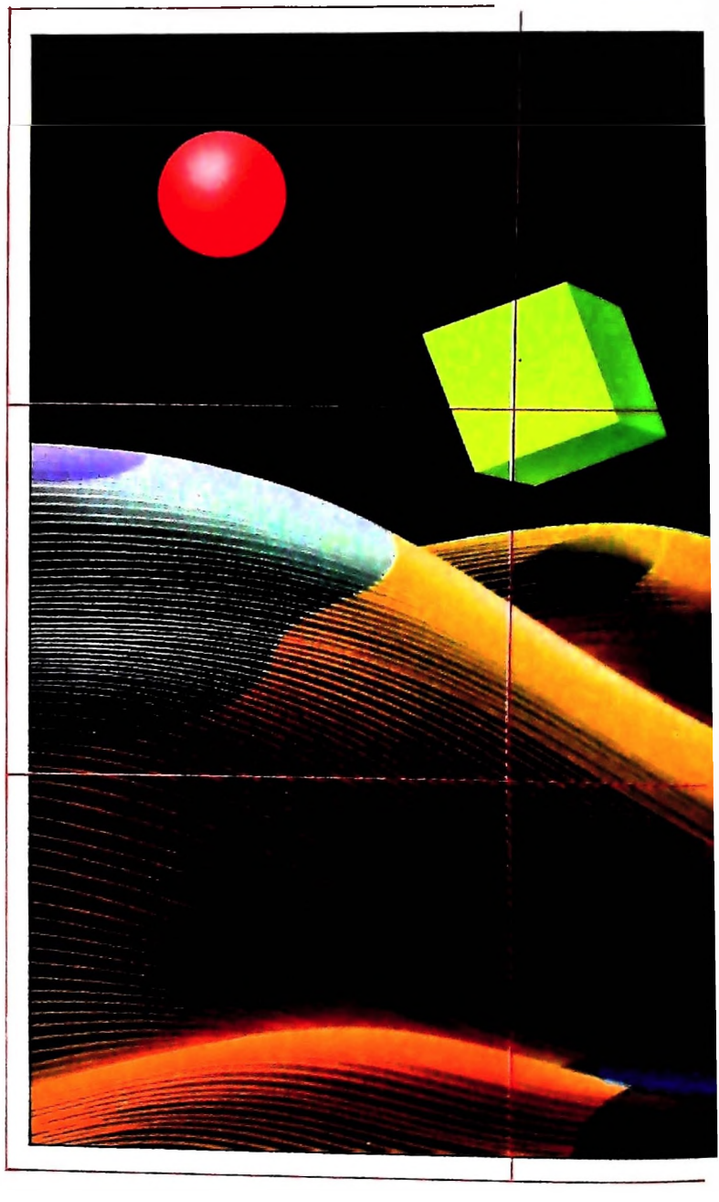
I sistemi operativi ed i linguaggi di programmazione devono inoltre evolversi per permettere agli utenti uno stile di comunicazione con i calcolatori più vicino al loro modo di pensare ed agire.

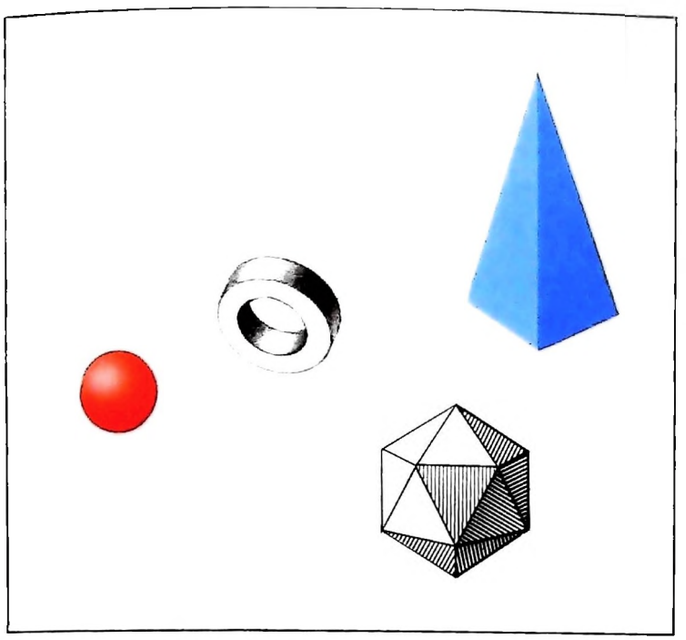
Le pagine seguenti saranno rivolte principalmente all'interazione visuale con il calcolatore, così come è resa possibile dall'utilizzo di tecniche proprie della Computer Graphics.



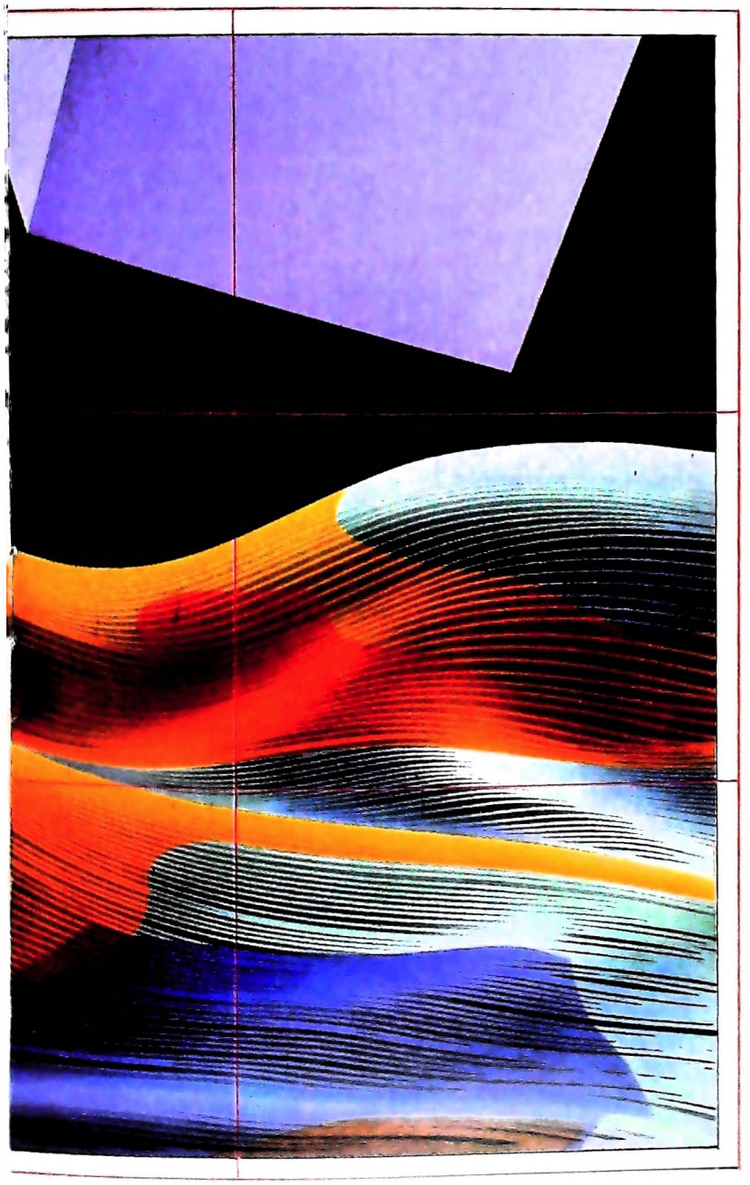
La struttura e l'organizzazione delle informazioni nella memoria

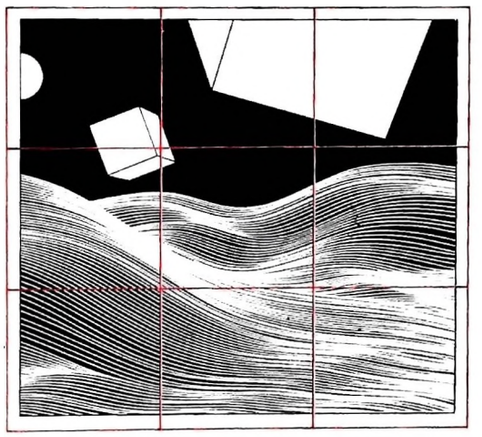
È evidente, da semplici considerazioni empiriche e da più rigorosi studi sulla percezione, che informazioni organizzate o potenzialmente organizzabili sono percepite, comprese e ritenute in maniera migliore di informazioni comparabili ma disorganizzate. Un numero di telefono di sette cifre, ad esempio, è raramente ricordato se appare come una sequenza di numeri casuali; se questo è invece organizzato come sequenze numeriche (22-435-24) è molto più facilmente ricordabile. La stessa cosa accade per sequenze di caratteri, dove sembra esistere un limite superiore di memorizzazione di sette elementi; se questi caratteri sono organizzati in una parola con un senso compiuto tale limite può essere elevato di molto. Questo principio, che stabilisce in sette il limite massimo al numero di unità informative che può essere ritenuto senza difficoltà, è stato definito da G. Miller nell'articolo "*The Magical Number Seven, Plus or Minus Two*". Questo tipo di limite vale anche per informazioni composite; le unità informative possono essere composte sia di parole che di frasi, così come di periodi, quindi senza riguardo alla lunghezza delle unità stesse (a cui la regola si applica ricorrentemente). Per ricordare un numero più alto di unità si sviluppano, in generale, delle connessioni fra i gruppi di informazioni in modo da ridurne il numero. Un tipico esempio di ciò è dato dalla famosa filastrocca "Trenta giorni ha Novembre con April ...." con cui si ricorda la lunghezza dei singoli mesi dell'anno riducendo così il numero di unità da ricordare da dodici a quattro. Queste regole si applicano specificatamente alla memoria di breve termine, quella che dura pochi giorni, mentre per la memoria a lungo termine, permanente, non sembra esistere un limite superiore al numero di informazioni che possono essere ritenute; vale ancora invece la regola del raggruppamento in blocchi delle informazioni.

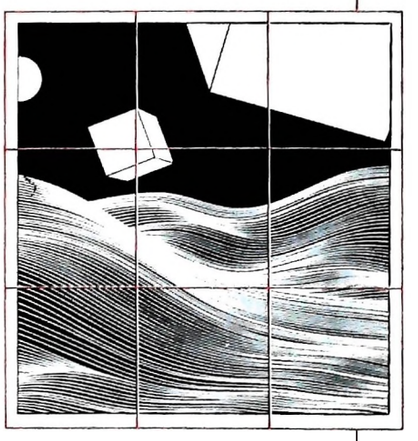




Ulteriori studi nel campo delle informazioni visuali, immagini e scene hanno dimostrato che il limite superiore di sette unità non può essere ritenuto valido in questo campo e numerosi esperimenti hanno portato a credere nella possibilità di ricordare e distinguere fino a diecimila immagini diverse (*L. Standing: "Learning 10.000 pictures"*). Le immagini sembrano dunque avere un accesso diretto alla memoria a lungo termine e le loro caratteristiche di forma, colore, dimensioni e struttura le rendono percepibili come unità direttamente inseribili negli schemi globali di questa. Informazioni comprendenti caratteristiche quali dimensioni, forme, distanze relative sono strutturate dalla mente per rappresentare scene visive. Le loro caratteristiche spaziali sono percepite come caratteristiche degli oggetti e non come proprietà astratte della scena stessa.
Tutte le parti di un oggetto sono percepite globalmente e non come componenti separate dello stesso e tutti gli oggetti sono percepiti come correlati l'uno all'altro. Ciò significa che la comunicazione visuale convoglia la struttura stessa di una







Due esempi di come immagini sostanzialmente identiche (spazialmente congruenti) siano percepite in maniera diversa variando opportunamente alcune condizioni (colori, sfondi, posizione degli oggetti rispetto alla cornice ideale della figura). Nella pagina accanto, in alto, un esempio di come l'uso di sfondi colorati e di cornici alteri l'equilibrio (bilanciamento) di una figura, dando più o meno peso a una parte o all'insieme degli oggetti che vi compaiono. Qui a sinistra, sopra e a destra, un esempio di proporzionamento: secondo la "regola dei terzi", i punti di interesse in cui dovrebbero essere collocati i soggetti componenti di una figura corrispondono alla intersezione delle coppie di linee verticali e orizzontali con cui una figura viene divisa in nove riquadri; le tre sequenze mostrano come variano i rapporti tra gli oggetti a seconda di come viene "tagliata" la superficie di un'immagine.

scena direttamente, senza necessità di mediazioni verbali. È proprio questa capacità della comunicazione visuale di far percepire la struttura che la rende estremamente comprensibile se confrontata alla comunicazione non-visuale. Per queste caratteristiche le immagini sono particolarmente adatte all'uso su display di calcolatori, essendo in grado sia di alleviare il carico sulla memoria dell'operatore, sia di diminuire il suo sforzo nel ricordarne e apprenderne il significato.

Come organizzare l'informazione presentata sui display dei computer

Attraverso lo sviluppo delle tecniche di Computer Graphics i display dei computer sono diventati un mezzo potenzialmente in grado di comunicare informazioni con l'immediatezza visiva di uno spot o di una pubblicità su rivista. A causa della scarsa conoscenza media dei programmatori sulle tecniche di comunicazione visuale, esiste la necessità di riconoscere l'importanza di una attenta definizione dell'interfaccia utente e di modificare quindi la metodologia di sviluppo del software in modo da ottenere una significativa comunicatività. I pubblicitari hanno da tempo sviluppato delle tecniche che garantissero il successo delle loro campagne attraverso lo studio dei componenti percettivi di un'immagine. L'uso di tecniche che uniscano la semplicità, la chiarezza e l'ordine nella presentazione delle informazioni porterebbe all'utente grandi benefici e, anche se l'intento della pubblicità è distinto da quello di un programma, questi ultimi potrebbero usare gli stessi metodi di presentazione. I principi fondamentali su cui si basa la creazione di una immagine pubblicitaria sono: a) Bilanciamento, b) Proporzione, c) Sequenzialità, d) Unità, e) Enfasi.

Bilanciamento

Il *bilanciamento* può essere definito come la distribuzione del peso ottico all'interno di una figura: con peso ottico ci si riferisce al fatto che un oggetto, nella composizione di una figura, possa influire, pesare, più o meno a seconda dei suoi parametri. Si noti che una figura bilanciata non significa una figura simmetrica, in cui due composizioni occupano i due lati di questa, ma una figura la cui composizione, con l'uso opportuno di forme e colori, dia una sensazione di stabilità e sicurezza. Gli elementi che si possono manipolare per ottenere una figura bilanciata sono molti, e fra questi: il colore, oggetti scuri pesano più di oggetti chiari ed il colore pesa più del bianco-nero; la dimensione degli oggetti, grossi oggetti pesano più di piccoli oggetti; la forma degli oggetti, una forma irregolare è più pesante di una regolare e definita.

Proporzione

La *proporzione* consiste nelle relazioni di dimensione, peso e massa che intercorrono in una figura. Un arrangiamento ben proporzionato dei componenti di una figura contribuisce in maniera significativa all'interesse di chi si pone di fronte a questa. In particolare esiste una regola, detta dei terzi, che stabilisce che i punti di interesse, in cui andrebbero quindi collocati gli oggetti, di una figura sono quelli che si trovano all'intersezione delle linee di divisione verticale ed orizzontale in terzi di questa.

Sequenzialità

La *sequenzialità* si riferisce all'arrangiamento degli oggetti, in una composizione, in modo che l'occhio, per vedere e percepire l'intera figura, possa scorrere da un oggetto all'altro senza fratture di continuità e salti di attenzione. Gli oggetti dovrebbero essere disposti a partire da sinistra verso destra e dall'alto in basso (nelle culture occidentali). Gli psicologi hanno inoltre dimostrato che l'interesse visivo delle persone "scorre" dagli oggetti più grandi a quelli più piccoli, dai colori chiari a quelli scuri, da oggetti di forma irregolare ad oggetti con caratteristiche di regolarità ed infine al testo.

Unità

Unità significa che tutti gli oggetti, all'interno di una composizione, dovrebbero apparire come appartenenti ad una stessa famiglia, ovvero possedere caratteristiche simili: ad esempio i caratteri che compaiono in una immagine dovrebbero non avere lo stesso corpo ma avere lo stesso stile. Oltre all'unità di componenti anche l'intera figura dovrebbe apparire come una unità: ad esempio l'uso di bordi e di spazi bianchi attorno a gruppi di oggetti conferisce unità agli stessi.

Enfasi

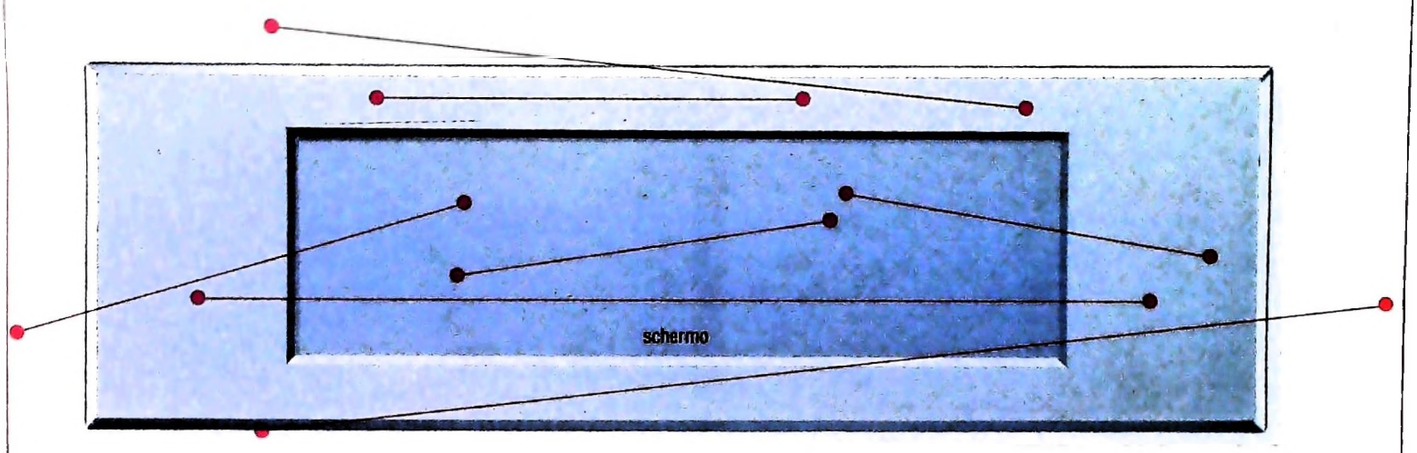
Da ultimo l'*enfasi*: in una composizione esiste un soggetto principale che deve essere messo in rilievo rispetto agli altri componenti di questa. Si è visto come la presenza di più soggetti principali distraggano e confondano e addirittura possano generare nell'osservatore instabilità e nervosismo. L'enfasi andrebbe posta sul soggetto che si vuole evidenziato.

Conclusioni

È possibile aumentare enormemente la facilità di apprendimento e di uso del calcolatore, alleggerendo così la fatica fisica di un operatore, attraverso l'utilizzo di principi della psicologia e della pubblicità; i programmatori dovrebbero essere messi in grado di sfruttare questi principi nel loro software.
In un prossimo articolo, vedremo quali tecniche si possono usare per raggiungere più alti livelli di comunicatività visuale e quali strumenti sono stati sviluppati, linguaggi e metodologie, per soddisfare queste esigenze.

IL CLIPPING

L'algoritmo di "clipping" è un nuovo strumento per arricchire il nostro pacchetto di software grafico.



I possibili modi in cui un segmento di retta può porsi rispetto allo schermo.

A cosa serve il clipping? Supponiamo di dover trattare una qualsiasi figura le cui dimensioni siano maggiori di quelle dello schermo di M10. Abbiamo visto nelle precedenti lezioni che potremmo scegliere una window di dimensioni tali da contenere l'intera figura e poi, definita una viewport sullo schermo, effettuare una trasformazione di scala. Potrebbe però interessarci riprodurre soltanto le parti della figura in questione che possono essere visibili sullo schermo, senza dover effettuare alcuna trasformazione. Il problema è allora quello di selezionarne le parti che risulteranno visibili. Il metodo utilizzato per ottenere queste informazioni è detto "clipping", ossia un processo in grado di dividere ogni elemento della figura nelle sue parti visibili e invisibili attraverso la finestra costituita dallo schermo.
Il clipping può essere applicato a differenti tipi di figure: punti, linee, curve di vario genere, testi di caratteri, poligoni. La base delle operazioni è una coppia di diseguaglianze in grado di determinare se un punto è visibile:

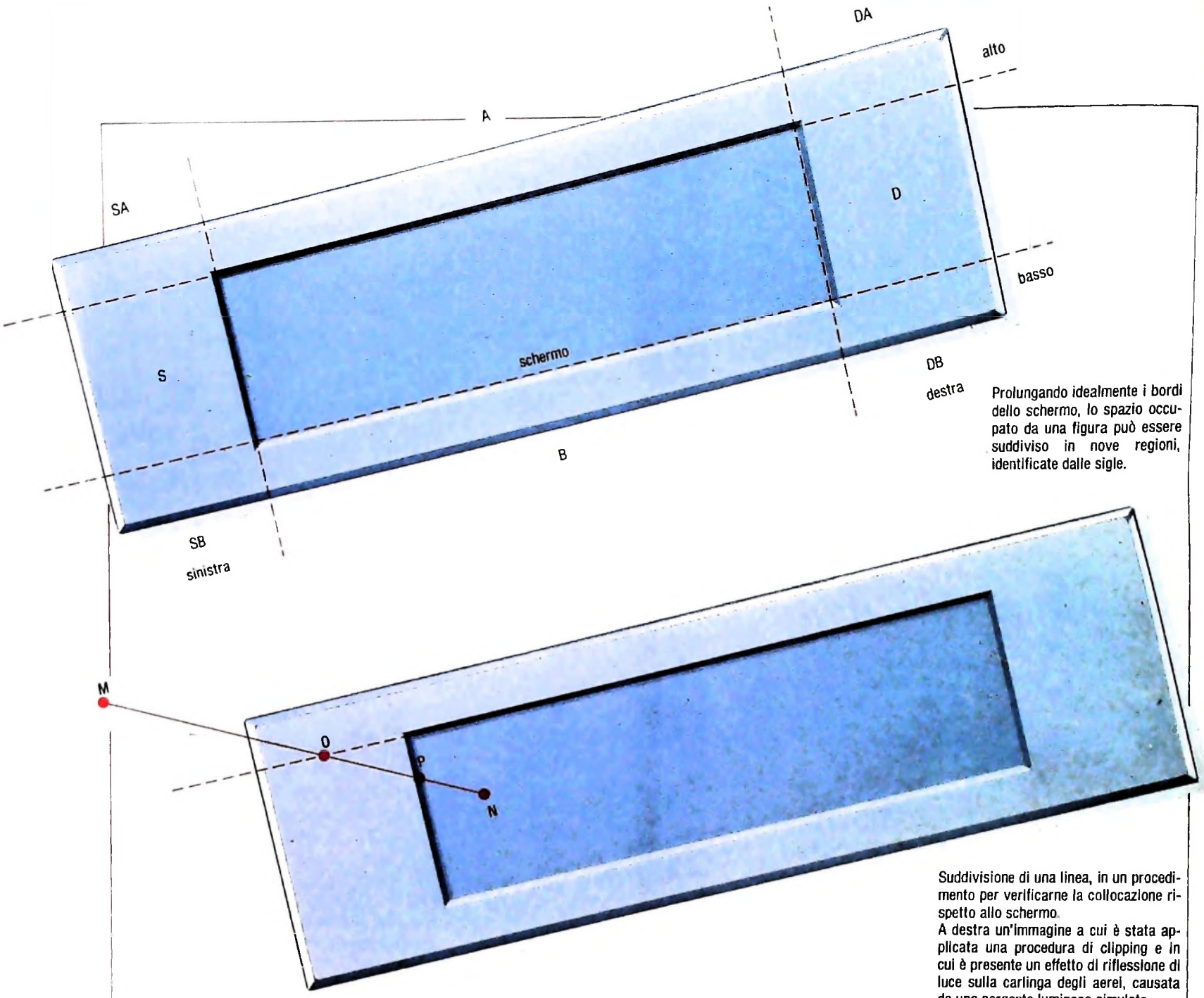
$$Xs \leqslant x \leqslant Xd$$
$$Yb \leqslant y \leqslant Ya$$

Xs,Xd,Yb,Ya sono le posizioni dei bordi dei lati dello schermo, con lo stesso significato definito per le window, mentre x e y sono le coordinate del punto considerato. Per M10 valgono quindi i seguenti valori: Xs=0, Xd=239, Yb=0, Ya=63.

Queste due diseguaglianze costituiscono un metodo molto semplice per effettuare il clipping di figure punto per punto: basta infatti sostituire le coordinate x e y di ogni punto nelle diseguaglianze e, se il loro valore non le verifica entrambe, esso è invisibile. Pur essendo questo metodo concettualmente il più semplice, è pur vero che, dovendo trattare una figura composta da molti punti, esso diventa enormemente oneroso in termini di tempo. Si rende perciò necessaria la definizione di un algoritmo di clipping che sia in grado di trattare non più solo singoli punti, ma insiemi di punti: le linee rette.

Un algoritmo di clipping per segmenti di linee

La figura in alto mostra tutti i possibili modi con cui un segmento di una linea retta può porsi rispetto allo schermo.
Si può verificare che tutte le linee solo parzialmente visibili vengono suddivise dai bordi dello schermo in una o due parti invisibili, ma con un solo segmento visibile. Quest'osservazione è estremamente utile, perché ci permette di affermare che il segmento visibile di una linea retta si può semplicemente determinare calcolando i suoi due estremi. Un algoritmo che utilizza questa osservazione, e che è particolarmente veloce nella ricerca dei vertici dei segmenti visibili, è quello che venne inventato da Dan Cohen ed Ivan Sutherland negli anni Sessanta. L'algoritmo è concettualmente suddiviso in due parti: nella prima si verifica se una linea è interamente



Prolungando idealmente i bordi dello schermo, lo spazio occupato da una figura può essere suddiviso in nove regioni, identificate dalle sigle.

Suddivisione di una linea, in un procedimento per verificarne la collocazione rispetto allo schermo.
A destra un'immagine a cui è stata applicata una procedura di clipping e in cui è presente un effetto di riflessione di luce sulla carlinga degli aerei, causata da una sorgente luminosa simulata.

interna o esterna allo schermo; nella seconda si seleziona il segmento visibile delle linee che sono parte interne e parte esterne allo schermo.
Descriviamo per ora a parole come si sviluppa questo algoritmo. Per prima cosa, si effettuano due test per determinare se la linea considerata giace interamente nello schermo o, altrimenti, se può essere scartata come completamente esterna. Se invece la linea risulta essere parzialmente contenuta nello schermo, viene suddivisa in due parti, che vengono sottoposte di nuovo ai due test precedenti. Il concetto sul quale si basa l'algoritmo è infatti che ogni linea o è interamente contenuta nello schermo o può essere suddivisa in due parti, in modo tale da poter escludere, come esterna e quindi invisibile, una delle due.
Per effettuare il test di esclusione, si prolungano idealmente i bordi dello schermo. Lo spazio occupato dalla figura che deve essere sottoposta al clipping risulta così suddiviso in nove regioni, ognuna delle quali viene identificata con una sigla costituita da uno o due caratteri, come mostrato in figura.

I vertici di ogni linea vengono poi a loro volta identificati con le sigle delle regioni nelle quali cadono. Il significato dei quattro caratteri utilizzati per la codifica è il seguente:

S : il punto è alla sinistra del lato sinistro dello schermo.
D : il punto è alla destra del lato destro dello schermo.
B : il punto è in basso rispetto al lato inferiore dello schermo.
A : il punto è in alto rispetto al lato superiore dello schermo.

Per esempio, se i vertici di una linea sono identificabili con le sigle SA e DB (ossia se cadono nelle rispettive regioni), significa che il primo vertice è alla sinistra del lato sinistro (dello schermo) e in alto rispetto a quello superiore, mentre il secondo è alla destra del lato destro e in basso rispetto a quello inferiore. In questo caso esisterà una parte della linea che è visibile. Possiamo anche qui fare una osservazione: le linee che sono visibili in parte sullo schermo sono tutte quelle i cui due vertici risultano identificati con due sigle contenenti cia-

scuna caratteri diversi dall'altra. Questo concetto si esprime dicendo che l'intersezione logica delle due sigle deve essere uguale a zero. Verificate questa asserzione nel disegno rappresentante le nove regioni dello spazio.
Chiaramente, se le sigle di entrambi i codici dei vertici sono vuote, allora la linea stessa giace interamente nello schermo.
Allora, se la linea non può essere interamente visibile o invisibile, deve essere suddivisa in due parti. Un buon metodo di suddivisione è quello di determinare un punto di intersezione della linea con un bordo dello schermo e scartare la parte che cade all'esterno dello stesso. Per esempio, la linea MN della figura può essere divisa nel punto O, e si può così scartare la parte MO come esterna. Il segmento ON deve essere ancora sottoposto a una suddivisione, che avviene in P. Si scarta OP e rimane, come parte totalmente interna allo schermo, il segmento PN (figura in basso della pagina a fronte).
Avrete osservato che sarebbe stato più conveniente effettuare la prima suddivisione direttamente in P, ma abbiamo così visto come il risultato finale dell'algoritmo non cambi, sia che si parta da un vertice o dall'altro.
Diamo, nella pagina seguente, una versione BASIC dell'algoritmo per segmenti di linee di Cohen-Sutherland, rimandando al riquadro per un suo commento più specifico.
In questo listato abbiamo utilizzato alcune possibilità di compattamento delle istruzioni consentite dal BASIC. Per esempio, nella istruzione 20, in cui si assegnano i lati di una window sullo schermo, sono state raggruppate quattro istruzioni diverse, separate fra loro dal simbolo dei due punti (:). Questo è un modo di procedere lecito e che consente di rendere più compatto e comprensibile il listato.

ACM ASSOCIATION - ARCHIVIO EIDOS

```
10 CLS
20 XS=10 : XD=229 : YB=10 : YA=53
30 INPUT "DAMMI LE COORDINATE DEI DUE
   VERTICI";X1,Y1,X2,Y2
40 CLS
50 GOSUB 2005
60 END
2005 REM SUBROUTINE CLIPLINE
2030 REM DISEGNO LA WINDOW
2040 LINE (XS,YB)-(XD,YA),1,B
2050 GOSUB 3050 :REM CODIFICA ESTREMI
2100 REM PROCEDURA PRINCIPALE
2110 IF C1$="" AND C2$="" THEN 2700
2120 IF LEFT$(C1$,1)=LEFT$(C2$,1) OR RI
     GHT$(C1$,1)=RIGHT$(C2$,1) THEN 2710
2130 C$=C1$
2140 IF C$="" THEN C$=C2$
2150 IF LEFT$(C$,1)="S" THEN 2500 ELSE
     IF LEFT$(C$,1)="D" THEN 2550 ELSE
     IF RIGHT$(C$,1)="B" THEN 2600 ELSE
     IF RIGHT$(C$,1)="A" THEN 2650
2160 IF C$=C1$ THEN X1=X : Y1=Y ELSE X2
     =X : Y2=Y
2170 GOSUB 3050
2180 GOTO 2100
2490 REM CALCOLO DELLE INTERSEZIONI DEL
2491 REM SEGMENTO DI LINEA CONSIDERATO
2492 REM CON I PROLUNGAMENTI DEI LATI
2493 REM DELLO SCHERMO.
2500 Y=Y1+(Y2-Y1)*(XS-X1)/(X2-X1)
2510 X=XS
2520 GOTO 2160
2550 Y=Y1+(Y2-Y1)*(XD-X1)/(X2-X1)
2560 X=XD
2570 GOTO 2160
2600 X=X1+(X2-X1)*(YB-Y1)/(Y2-Y1)
2610 Y=YB
2620 GOTO 2160
2650 X=X1+(X2-X1)*(YA-Y1)/(Y2-Y1)
2660 Y=YA
2670 GOTO 2160
2700 LINE (X1,Y1)-(X2,Y2)
2710 RETURN
3050 REM ASSEGNAMENTO CODICI DEI VERTICI
3055 C1$="" : C2$=""
3060 IF X1<XS THEN C1$="S" ELSE IF X1>
     XD THEN C1$="D"
3070 IF Y1<YB THEN C1$=C1$+"B" ELSE IF
     Y1>YA THEN C1$=C1$+"A"
3080 IF X2<XS THEN C2$="S" ELSE IF X2>
     XD THEN C2$="D"
3090 IF Y2<YB THEN C2$=C2$+"B" ELSE IF
     Y2>YA THEN C2$=C2$+"A"
3100 RETURN
```

Commento all'algoritmo di Cohen-Sutherland

La natura di questo algoritmo è quella di programma di utilizzo, ossia le istruzioni che vanno dalla 2005 alla 3100, e che costituiscono la subroutine CLIPLINE, possono essere accodate a un qualsiasi programma ove si renda necessario il clipping di linee. Vi consigliamo pertanto di salvare su cassetta questa subroutine, dato che vi potrà essere utile molte volte. La numerazione delle linee a partire dal numero 2005 è stata volutamente scelta per lasciare lo spazio sufficiente al programma al quale la subroutine andrà accodata e dal quale potrà essere richiamata.

Per esemplificare il suo funzionamento abbiamo qui introdotto la richiesta di dati per un solo segmento di linea (istr. 30). Una estensione dell'input può essere ottenuta creando due tipi di array, uno per la coordinata x e uno per la y dei vertici di una figura più complessa.

La struttura della CLIPLINE è costituita sostanzialmente da tre sezioni: la subroutine "Assegnamento codici dei vertici" (istr. 3050-3100), la procedura principale (istr. 2100-2180) e una sezione dedicata al calcolo delle intersezioni fra segmento e lati dello schermo (istr. 2490-2670). La sequenza logica utilizzata nell'algoritmo è già stata esposta; vediamo qui invece le tecniche e gli strumenti di programmazione utilizzati.

Le sigle con cui si indicano i vertici del segmento considerato vengono memorizzate in due variabili alfanumeriche (C1$ e C2$). Su queste variabili è possibile compiere una serie di test utilizzando le due funzioni LEFT$ e RIGHT$. Per vederne il funzionamento facciamo un esempio estraendolo dal listato.

Consideriamo l'istruzione 2150 e commentiamo il test:

IF LEFT$(C$,1)="S" THEN 2500

Ossia, se il primo carattere partendo dalla sinistra della stringa contenuta nella variabile alfanumerica C$ è uguale a S, allora il programma salta alla linea 2500. Pertanto il risultato della funzione LEFT$ è quello di estrarre un certo numero di caratteri da una stringa partendo da sinistra.

Il formato generale della funzione è il seguente:

LEFT$(X$,n)

ove X$ è una variabile alfanumerica

n è un numero intero compreso fra 0 e 255

Vengono cioè selezionati i primi n caratteri della stringa X$ a partire dalla sinistra. Se n è maggiore del numero dei caratteri contenuti da X$, viene considerata l'intera stringa, se invece n è uguale a zero viene considerata una stringa vuota.

Un analogo discorso vale per la funzione RIGHT$, ove tutto quanto detto precedentemente viene riferito al lato destro della stringa.

Ricordiamo che in inglese "left" e "right" significano rispettivamente sinistra e destra.

Le istruzioni 2500, 2550, 2600 e 2650 costituiscono ciascuna la formula matematica, utilizzata dalla geometria analitica, per calcolare le coordinate della intersezione di due linee.

Come ultima osservazione, utilizzando il programma, va ricordato che il sistema di riferimento di M10 è posto nell'angolo in alto a destra dello schermo e che le y aumentano scendendo verso il basso.

Olivetti M10 vuol dire disporre del proprio ufficio in una ventiquattrore. Perché M10 non solo produce, elabora, stampa e memorizza dati, testi e disegni, ma è anche capace di comunicare via telefono per spedire e ricevere informazioni. In grado di funzionare a batteria oppure collegato all'impianto elettrico, M10 mette ovunque a disposizione la sua potenza di memoria, il suo display orientabile a cristalli liquidi capace anche di elaborazioni grafiche, la sua tastiera professionale arricchita da 16 tasti funzione.

Ma M10 può utilizzare piccole periferiche portatili che ne ampliano ancora le capacità, come il microplotter per scrivere e disegnare a 4 colori, o il registratore a cassette per registrare dati e testi, o il lettore di codici a barre. E in ufficio può essere collegato con macchine per scrivere elettroniche, con computer, con stampanti. Qualunque professione sia la vostra, M10 è in grado, dovunque vi troviate, di offrirvi delle capacità di soluzione che sono davvero molto grandi. M10: il più piccolo di una grande famiglia di personal.

PERSONAL COMPUTER OLIVETTI M10

L'UFFICIO DA VIAGGIO

Anche in leasing con Olivetti Leasing.

olivetti

Per ulteriori informazioni rivolgersi a negozi convenzionati di Olivetti M10 Punto di Vendita
o inviare questo coupon a Olivetti, Divisione Personal Computer, Via Meravigli 12
20123 Milano

NOME COGNOME

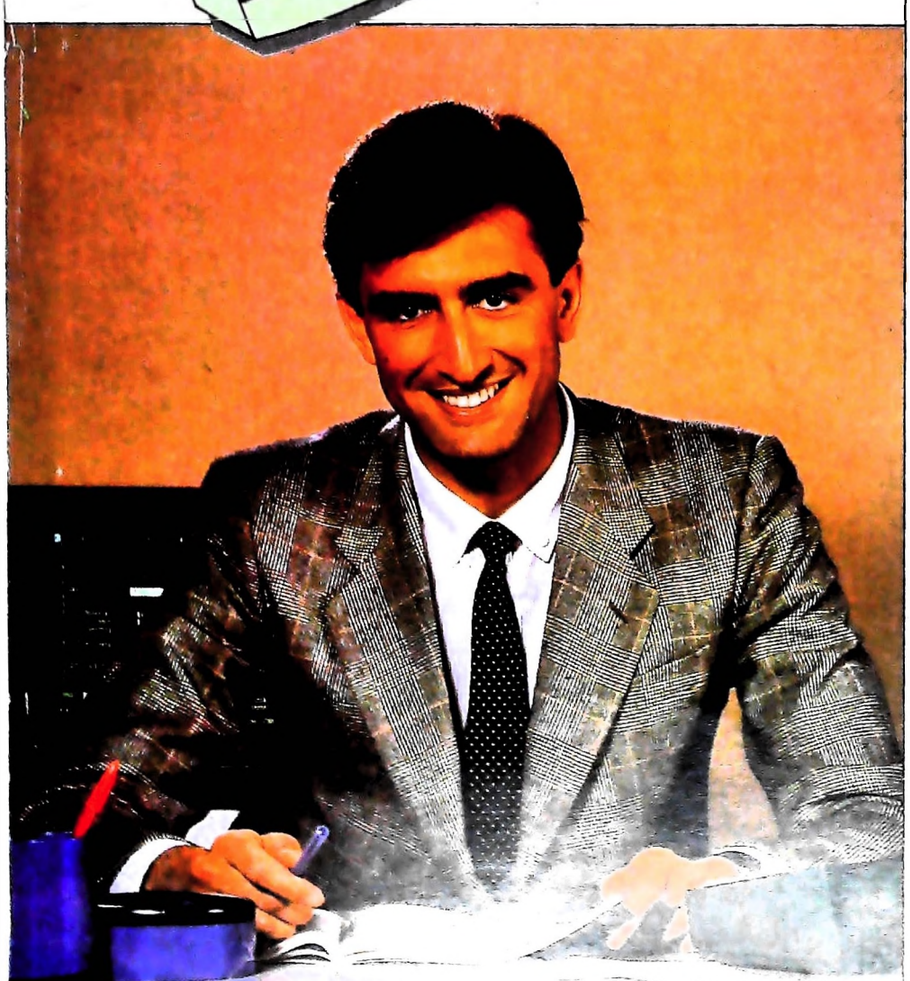
VIA

CAP 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.