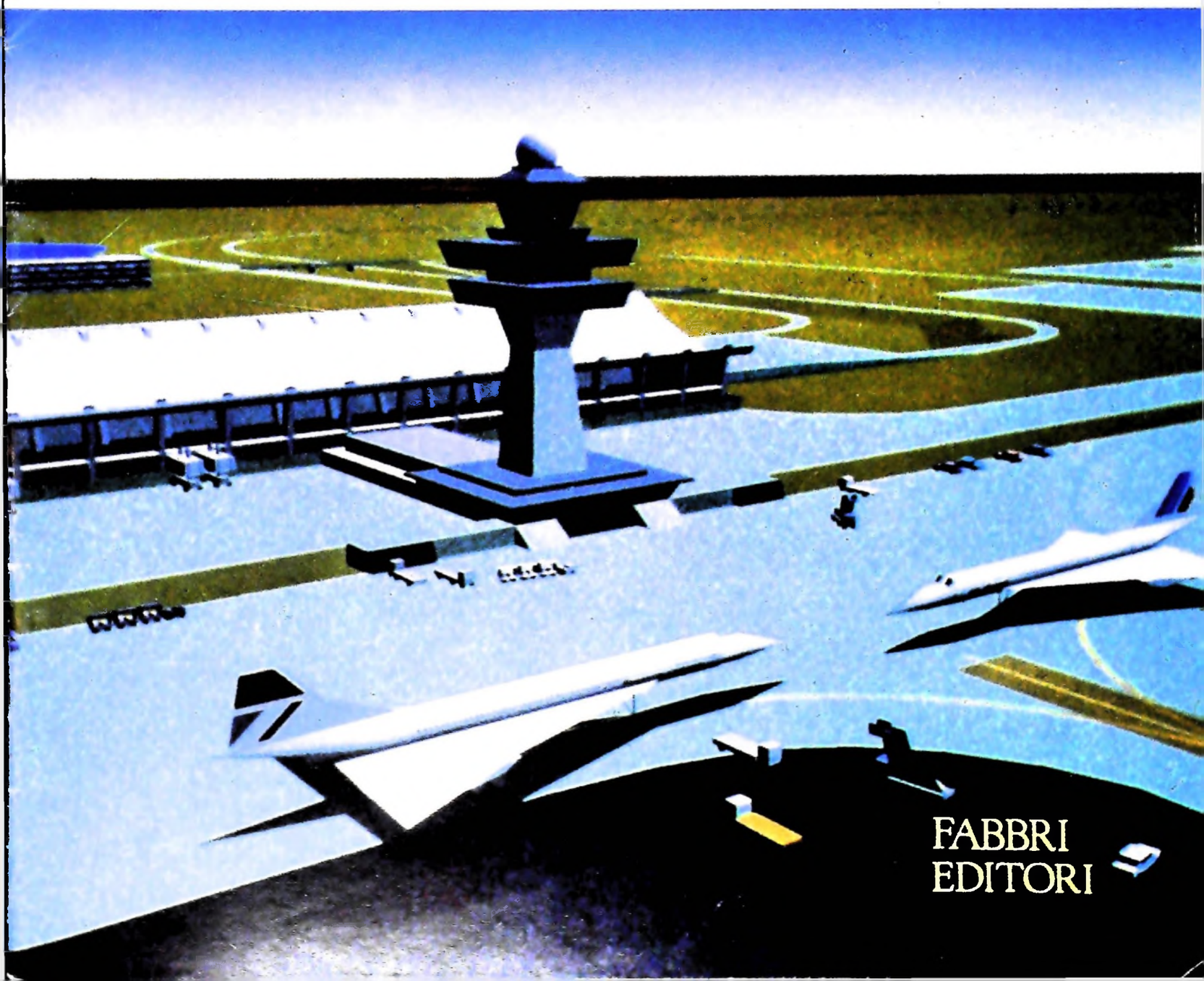
Spediz. in abbonamento postale GR II/70 L. 2.200
(...)

51 CORSO PRATICO COL COMPUTER

422030

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

diretto da **GIANNI DEGLI ANTONI**



FABBRI EDITORI

IL BANCO DI ROMA FINANZIA IL VOSTRO ACQUISTO DI M 10 e M 20

Acquisto per contanti

È la formula di acquisto tradizionale.
Non vi sono particolari commenti da fare, se non sottolineare che troverete ampia disponibilità presso i punti di vendita Olivetti, poiché, grazie al "Corso pratico col computer", godrete di un rapporto di privilegio.

Il servizio di finanziamento bancario

Le seguenti norme descrivono dettagliatamente il servizio di finanziamento offerto dal Banco di Roma e dagli Istituti bancari a esso collegati:

Banca Centro Sud
Banco di Perugia

Le agenzie e/o sportelli di questi istituti sono presenti in 216 località italiane.

Come si accede al credito e come si entra in possesso del computer

1) Il Banco di Roma produce una modulistica che è stata distribuita a tutti i punti di vendita dei computer M 10 e M 20 caratterizzati dalla vetrofania M 10.
2) L'accesso al servizio bancario è limitato solo a coloro che si presenteranno al punto di vendita Olivetti.
3) Il punto di vendita Olivetti provvederà a istruire la pratica con la più vicina agenzia del Banco di Roma, a comunicare al cliente entro pochi gorni l'avvenuta concessione del credito e a consegnare il computer.

I valori del credito

Le convenzioni messe a punto con il Banco di Roma, valide anche per le banche collegate, prevedono:
1) Il credito non ha un limite minimo, purché tra le parti acquistate vi sia l'unità computer base.
2) Il valore massimo unitario per il credito è fissato nei seguenti termini:
   — valore massimo unitario per M 10 = L. 3.000.000
   — valore massino unitario per M 20 = L. 15.000.000
3) Il tasso passivo applicato al cliente è pari al "prime rate ABI (Associazione Bancaria Italiana) + 1,5 punti percentuali".
4) La convenzione prevede anche l'adeguamento del tasso passivo applicato al cliente a ogni variazione del "prime rate ABI"; tale adeguamento avverrà fin dal mese successivo a quello a cui è avvenuta la variazione.
5) La capitalizzazione degli interessi è annuale con rate di rimborso costanti, mensili, posticipate; il periodo del prestito è fissato in 18 mesi.
6) Al cliente è richiesto, a titolo di impegno, un deposito cauzionale pari al 10% del valore del prodotto acquistato, IVA inclusa; di tale 10% L. 50.000 saranno trattenute dal Banco di Roma a titolo di rimborso spese per l'istruttoria, il rimanente valore sarà vincolato come deposito fruttifero a un tasso annuo pari all'11%, per tutta la durata del prestito e verrà utilizzato quale rimborso delle ultime rate.
7) Nel caso in cui il cliente acquisti in un momento successivo altre parti del computer (esempio, stampante) con la formula del finanziamento bancario, tale nuovo prestito attiverà un nuovo contratto con gli stessi termini temporali e finanziari del precedente.

Le diverse forme di pagamento del finanziamento bancario

Il pagamento potrà avvenire:
- ☐ presso l'agenzia del Banco di Roma, o Istituti bancari a esso collegati, più vicina al punto di vendita Olivetti;
- ☐ presso qualsiasi altra agenzia del Banco di Roma, o Istituto a esso collegati;
- ☐ presso qualsiasi sportello di qualsiasi Istituto bancario, tramite ordine di bonifico (che potrà essere fatto una volta e avrà valore per tutte le rate);
- ☐ presso qualsiasi Ufficio Postale, tramite vaglia o conto corrente postale. Il numero di conto corrente postale sul quale effettuare il versamento verrà fornito dall'agenzia del Banco di Roma, o da Istituti a esso collegati.

BANCO DI ROMA
CONOSCIAMOCI MEGLIO.

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
MARCO ANELLI, DIEGO BIASI, ANDREA GRANELLI, ALDO GRASSO, MARCO MAIOCCHI, DANIELE MARINI, GIANCARLO MAURI, CLAUDIO PARMELLI

Testi
MARCO ANELLI, DIEGO BIASI, VIRGINIO SALA,
Eidos (TIZIANO BRUGNETTI), 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

AVVISO AI LETTORI

Con il n. 52 sarà in edicola la copertina per rilegare il quarto volume del "Corso pratico col computer". Prenotatela dal vostro edicolante.

Corso Pratico col Computer - Copyright © sul fascicolo 1985 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 - Distribuzione per l'Italia Gruppo Editoriale Fabbri S.p.A., via Mecenate, 91 - Milano - tel. 02/50951 - Pubblicazione periodica settimanale - Anno II - n. 51 - 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.

GRAMMATICHE E MACCHINE (II)

A una gerarchia di linguaggi di complessità crescente corrisponde una gerarchia di macchine, finite e infinite, in grado di riconoscerli.

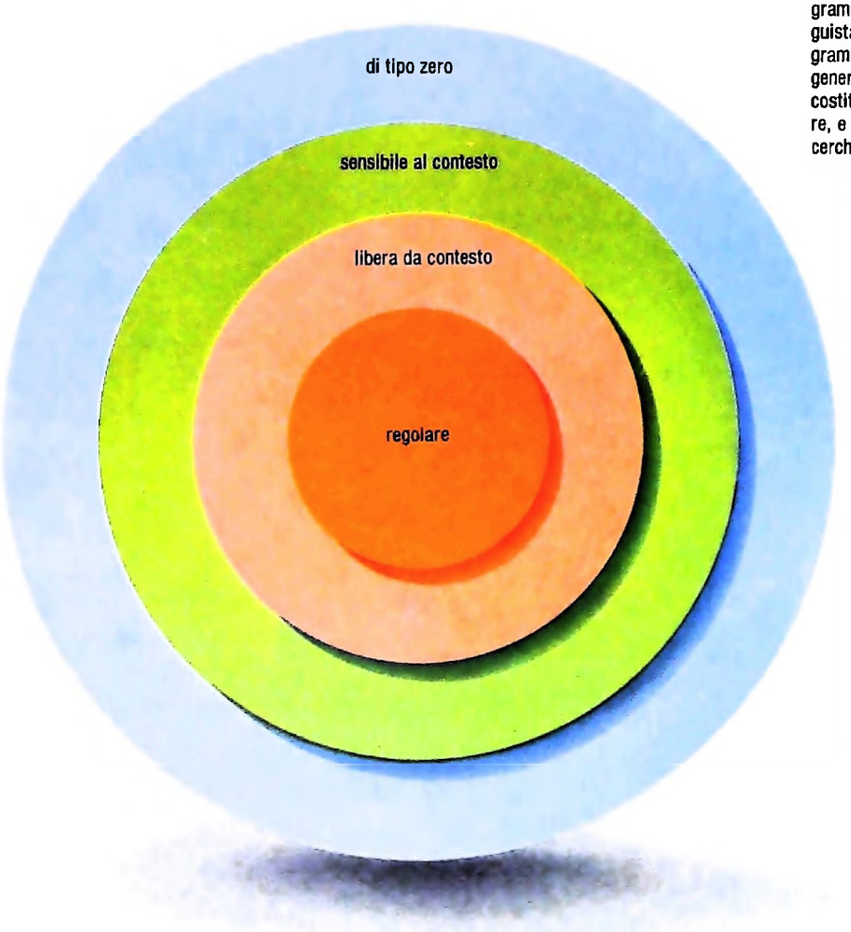
Le grammatiche sensibili al contesto, le grammatiche libere dal contesto, le grammatiche regolari si dicono anche, rispettivamente, grammatiche di tipo 1, 2 e 3 (le grammatiche a struttura sintagmatica, come ricorderete, si dicono anche di tipo 0). Le grammatiche a struttura sintagmatica sono quelle più generali, le altre ne sono casi particolari, ma non solo: le grammatiche libere dal contesto sono casi particolari delle grammatiche sensibili al contesto, e le grammatiche regolari sono casi particolari delle grammatiche libere dal contesto. L'illustrazione di questa pagina presenta visivamente la relazione che sussiste dunque fra i vari tipi di grammatiche.

Il nome "regolare" attribuito alle grammatiche di tipo 3 forse vi avrà già risvegliato qualche ricordo: in effetti sì, una grammatica regolare descrive un linguaggio regolare, che altro non è se non un insieme regolare di parole su un alfabeto (o vocabolario) di partenza.

E, come abbiamo visto, un insieme regolare può essere riconosciuto da una macchina a stati finiti: quindi una macchina a stati finiti è un riconoscitore per un linguaggio descritto da una grammatica regolare e, per il teorema di Kleene, viceversa, cioè per qualunque linguaggio descritto da una grammatica regolare esiste una macchina a stati finiti che è in grado di riconoscerlo.

All'altro estremo, abbiamo visto, stanno le grammatiche a struttura sintagmatica, che permettono di descrivere linguaggi ricorsivamente enumerabili: esistono macchine che possano riconoscere linguaggi di questo genere?

Sì, e le abbiamo già incontrate: sono le macchine di Turing. Le macchine di Turing, i modelli più astratti e più potenti delle macchine da calcolo, possono essere viste come automi finiti che possono muoversi liberamente su una matrice di memoria infinita.

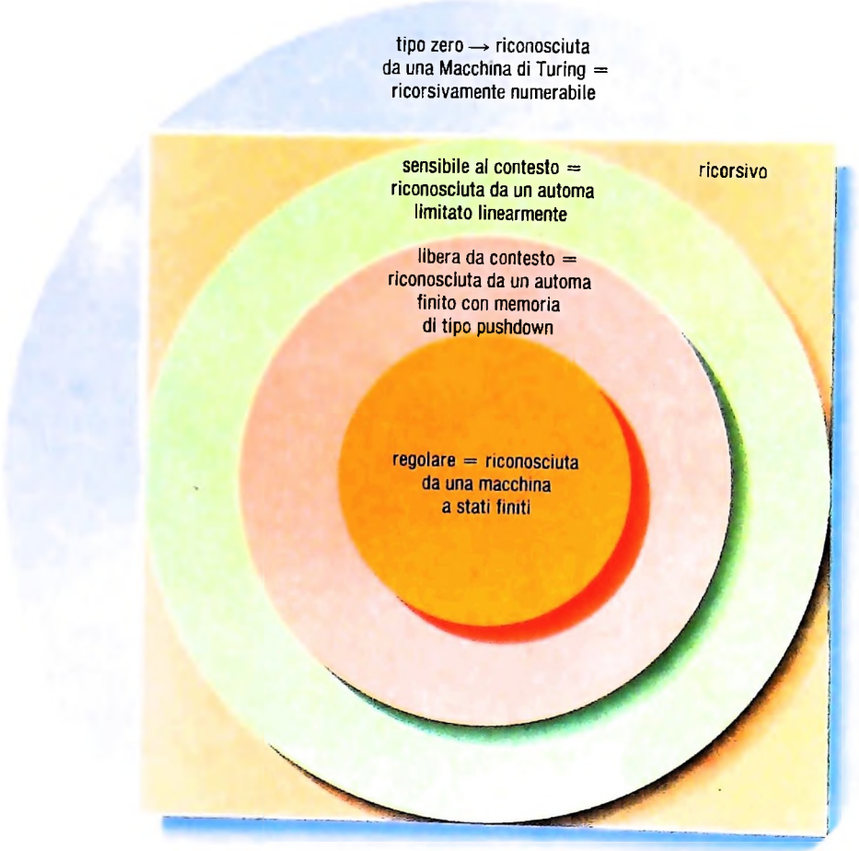


Il disegno visualizza la gerarchia delle grammatiche, secondo l'analisi del linguista americano Noam Chomsky. Le grammatiche di tipo zero sono le più generali: quelle sensibili al contesto ne costituiscono un sottoinsieme particolare, e via di seguito come è indicato dai cerchi concentrici.

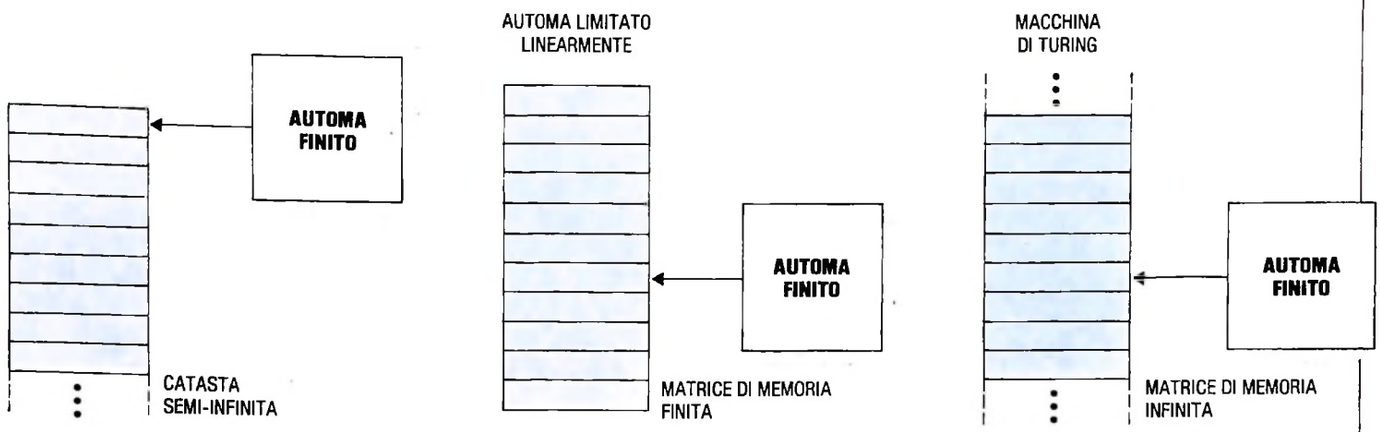
Esistono macchine in un certo senso "intermedie" fra le macchine a stati finiti e le macchine di Turing, che corrispondano agli altri due livelli nella gerarchia delle grammatiche (e dei linguaggi)?
Anche in questo caso la risposta è positiva. Una macchina a stati finiti (o automa finito) permette sostanzialmente di analizzare linguaggi finiti o linguaggi infiniti per i quali la correttezza (o l'accettabilità) di una parola possa essere stabilita semplicemente andandola a "leggere" da sinistra verso destra, un simbolo alla volta. È questo il significato della restrizione che avevamo posto nella definizione di una grammatica di tipo regolare (che cioè in ogni regola di riscrittura il simbolo a destra della freccia potesse essere solo o un simbolo terminale o un simbolo terminale seguito da un simbolo non terminale).
Nella sua attività, un automa finito non può tornare indietro a vedere un simbolo che ha già ricevuto in ingresso: per questo non è in grado di riconoscere se in una configurazione, per esempio, il numero di 1 è uguale al numero di 0.

Estensioni delle macchine finite

Il tipo di estensione possibile per una macchina finita è una qualche forma di "memoria", un dispositivo in cui i simboli possono essere immagazzinati e quindi recuperati dalla macchina, in modo da estenderne le capacità di confronto. Il primo tipo di memoria che si può aggiungere è uno stack (pila o catasta), una "sequenza" di elementi di memoria organizzata in modo particolare, cioè come una pila di piatti: i piatti si mettono uno sopra l'altro, ma dalla pila il primo piatto che si può togliere è quello più in alto, cioè l'ultimo accatastato. Lo stack è una struttura di memoria del tipo "ultimo dentro, primo fuori". L'automa finito può immagazzinare un elemento di informazione solo ponendolo in cima alla pila; quando poi deve andare a recuperare un elemento di informazione memorizzato, deve prima rimuovere dalla pila tutti gli elementi che sono stati accatastati al di sopra di esso.
Un automa finito dotato di una memoria di questo genere (detto anche macchina finita con memoria di tipo push-



Il rapporto fra la gerarchia delle grammatiche secondo Chomsky e i vari tipi di macchine che sono in grado di riconoscerle.



La gerarchia delle macchine al di sopra della semplice macchina a stati finiti che riconosce grammatiche regolari. La macchina con memoria di tipo pushdown è una macchina a stati finiti con una memoria a catasta semi-infinita. Un automa limitato linearmente ha come struttura di memoria una matrice finita, ma "ad accesso casuale". Nelle macchine di Turing si ha anche l'eliminazione del vincolo di finitezza della matrice di memoria.

down) permette il riconoscimento di un linguaggio generato da una grammatica libera da contesto: il riconoscimento di un simbolo, la sua accettabilità, può dipendere dal simbolo immediatamente precedente e da quello immediatamente successivo, perché questo è il margine che lascia il tipo di memoria di cui l'automa è fornito.

Una macchina finita dotata di una struttura di memoria a stack è in grado di stabilire se in una configurazione di simboli in ingresso due simboli particolari sono contenuti lo stesso numero di volte.

La pila che costituisce la memoria di una macchina in grado di riconoscere linguaggi liberi da contesto è pensata infinita (sia pure, ovviamente, in una sola direzione). Il concetto di pila è molto diffuso nei linguaggi di programmazione: il Forth addirittura fa della pila il concetto chiave nel suo modo tipico di organizzare la memoria. Sono frequenti anche i dispositivi fisici (hardware) che si comportano come pile di memoria (e cioè del tipo "ultimo dentro, primo fuori").

La possibilità di considerare solo i due simboli adiacenti al simbolo da analizzare è una limitazione abbastanza pesante: se la si toglie, si ottengono i linguaggi sensibili al contesto, e il tipo di macchina che è in grado di riconoscere linguaggi simili deve avere una memoria organizzata in modo molto più versatile.

Si tratta sostanzialmente di automi finiti con una struttura di memoria "ad accesso casuale", che permette cioè di accedere liberamente a qualunque posizione senza seguire lo schema rigorosamente sequenziale della pila.

La matrice di memoria è in questo caso finita, ma si suppone sufficientemente grande per contenere qualunque stringa di simboli sia fornita in ingresso alla macchina. La macchina stessa si definisce "automa limitato linearmente". L'eliminazione del vincolo di finitezza della matrice di memoria ci porta alla macchina di Turing: la matrice di memoria è il nastro, che si suppone sia infinito in ambedue le direzioni.

Le relazioni fra i quattro tipi di macchine possono essere viste anche a ritroso: se una macchina di Turing è un automa con un nastro infinito in ambedue le direzioni su cui una testina può leggere, scrivere e cancellare simboli, un automa limitato linearmente è una macchina di Turing con un nastro finito; la macchina con memoria a pila è una macchina di Turing che possiede un nastro infinito in una direzione, ma con una testina che deve rimanere fissa sull'ultima casella non vuota del nastro, e non può muoversi liberamente; l'automa finito, infine, è una macchina di Turing che non possiede nastro.

Automi e compilatori

Non è certo un compito facile scrivere programmi che traducano programmi scritti in linguaggi di alto livello in programmi scritti in linguaggio macchina: per questo la teoria degli automi, le macchine finite, riveste un interesse teorico notevole. L'attività di un compilatore si svolge attraverso varie fasi: la prima è l'analisi lessicale, che analizza le stringhe in ingresso per identificare i singoli elementi linguistici; la seconda è l'analisi sintattica, che opera sugli elementi identificati dall'analisi lessicale e ne stabilisce le relazioni sintattiche; la terza, infine, è la generazione del codice di macchina, in funzione delle due analisi precedenti.

L'attività dell'analizzatore lessicale è esemplificata dalla macchina a stati finiti che identificava numeri romani, che abbiamo visto in precedenza. Per l'analisi sintattica, nella maggior parte dei linguaggi di programmazione è sufficiente un automa finito con memoria di tipo pushdown: quasi tutti i linguaggi di programmazione sono governati, cioè, da grammatiche libere dal contesto. Evidentemente, grammatiche di questo tipo sono già in grado di generare linguaggi dotati di notevole potere espressivo.

INTERPRETI E LINKER

Che cosa sono e come operano.

Che cos'è un interprete

È stato illustrato come il processo di traduzione di un linguaggio, solitamente con un alto grado di simbolismo, in una forma eseguibile da una macchina, sia realizzato attraverso un compilatore.
Un compilatore è quindi un traduttore guidato dalle grammatiche di descrizione dei due linguaggi su cui opera: il linguaggio sorgente, quello cioè in cui viene espresso il programma, e il linguaggio oggetto, cioè quello che una macchina è in grado di eseguire.
I compilatori sono peraltro dei programmi di grosse dimensioni (a seconda del linguaggio che compilano e di quello in cui sono scritti vanno da migliaia di linee di codice in su), piuttosto complicati, con un grado di interazione con l'utente piuttosto basso e con un forte legame con la CPU per la quale generano codice. Ciò non significa che i compilatori abbiano solo svantaggi insiti nel loro uso e nelle loro modalità di funzionamento, altrimenti non sarebbe giustificabile la loro diffusione e il loro ruolo centrale nella fase di sviluppo di programmi, ma è senz'altro vero che, per piccoli sistemi di facile uso, non rivolti a utenti specializzati, nonché per certe classi di linguaggi, essi possano essere sostituiti da un'altra classe di strumenti che ne annullano in parte gli svantaggi. Questi strumenti si chiamano *interpreti*.
Gli interpreti non sono esattamente dei traduttori, anzi il processo di traduzione rappresenta solo la fase iniziale dell'interpretazione, spesso nemmeno percepita dall'utilizzatore. Essi sono piuttosto degli esecutori di programmi. La figura alla pagina successiva mostra come differiscano due catene di programmazione che usino, l'una un compilatore e l'altra un interprete, per trasformare un programma dalla sua forma sorgente in una forma funzionalmente equivalente ma eseguibile.
Nel caso di una catena di programmazione che utilizzi un compilatore, sono chiaramente distinguibili le fasi che vengono eseguite in sequenza per effettuare le necessarie trasformazioni. Nel caso di uso di un interprete il programma subisce delle trasformazioni all'interno dell'interprete stesso, al termine delle quali il programma viene eseguito direttamente, ma non dalla CPU bensì dall'interprete o, meglio, dalla parte esecutiva dell'interprete. A un livello di astrazione più alto è possibile pensare a un interprete come ad una estensione della CPU che attraverso di esso è in grado di eseguire direttamente il programma sorgente, come se il linguaggio macchina fosse proprio il linguaggio in cui il programma è scritto. Si dice, in questo caso, che l'interprete ha definito una *macchina astratta*, o *virtuale*, il cui linguaggio macchina è il linguaggio di programmazione utilizzato.
Proviamo ora ad analizzare quali vantaggi e svantaggi derivino dall'uso di un interprete piuttosto che di un compilatore.

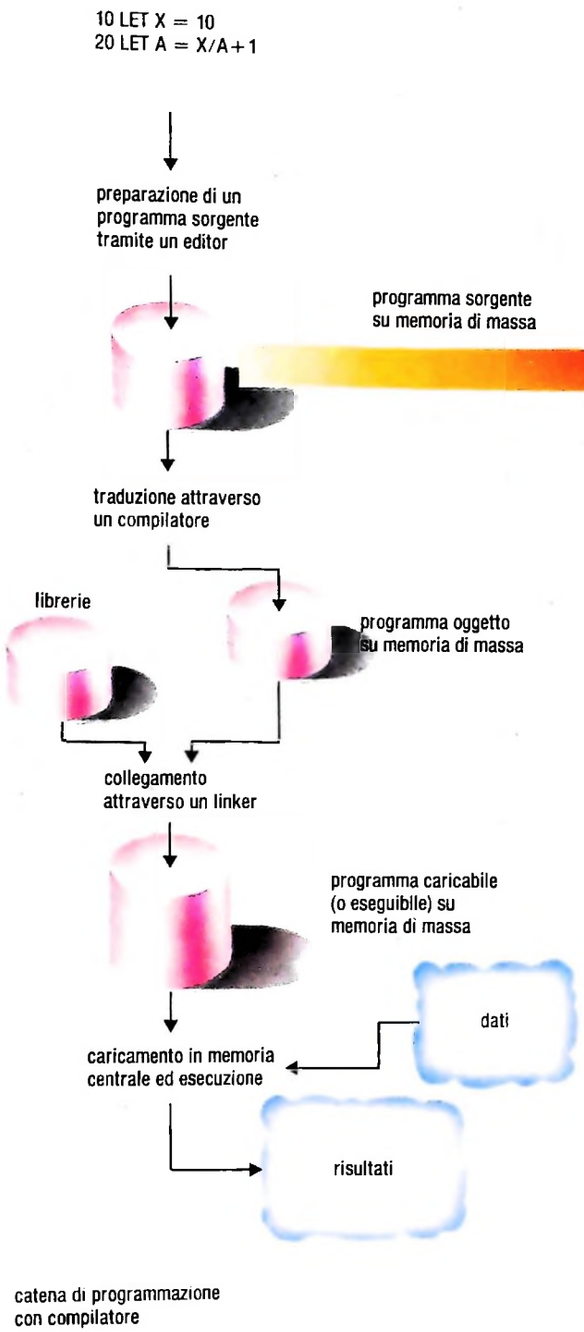
Vantaggi di un interprete

Semplicità d'uso: il processo di interpretazione permette una più stretta interazione con l'utente durante la fase di messa a punto del programma nonché la eliminazione di molti componenti della catena di programmazione (spesso, come per esempio nel BASIC, la preparazione del sorgente, cioè la fase di editing, è integrata nell'interprete stesso; non è richiesta, inoltre, la fase di collegamento).
Compattezza del sistema: su sistemi molto piccoli l'interprete costituisce l'intero ambiente di programmazione.
Indipendenza dalla CPU: l'interprete definisce una macchina astratta, indipendente dalla CPU, che eseguirà il programma. Inoltre, se l'interprete stesso è scritto in un linguaggio ad alto livello, e quindi indipendente da una particolare macchina, lo stesso interprete può essere eseguito da CPU differenti, in quanto, avendo a disposizione su diverse CPU il compilatore dello stesso linguaggio, sarà possibile compilare l'interprete stesso sulle varie macchine, ottenendo un programma eseguibile su ciascuna di esse.

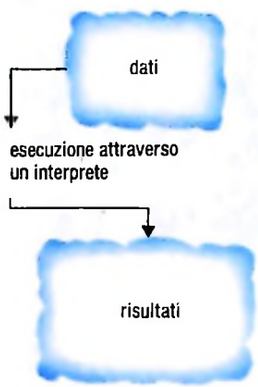
Svantaggi di un interprete

Minore flessibilità: una catena di programmazione come quella richiesta da un compilatore permette di integrare fra loro le componenti di un programma in un momento distinto dalla fase di traduzione. Ciò è molto importante in un ambiente di sviluppo a cui partecipano più persone.
Prestazioni peggiori: un programma interpretato è più lento di un programma eseguito direttamente dalla CPU. Infatti, nel caso dell'interpretazione, la CPU esegue l'interprete, il quale a sua volta esegue il programma dell'utente.
Ripetizione della traduzione: contrariamente all'approccio compilativo, la forma tradotta non viene in generale conservata. Il programma subisce quindi una traduzione prima di ogni esecuzione.

A sinistra, una catena di programmazione con compilatore. Qui sotto, invece, una che usa un interprete. Il fine è, ovviamente, il medesimo: trasformare un programma dalla sua forma sorgente in una forma eseguibile dalla macchina.



catena di programmazione con compilatore



catena di programmazione con interprete

Struttura di un interprete

Illustreremo ora un modello di struttura di un interprete (figura alla pagina seguente); interpreti reali possono differire dal modello illustrato anche in funzione del linguaggio che essi accettano, tuttavia l'architettura illustrata è generale e valida per la gran parte dei casi reali.

Modulo di comunicazione: gestisce le operazioni di I/O con l'utente, con i file di disco e con altre periferiche. L'elemento di comunicazione scambiato è la linea di testo.

Modulo di editing: definisce le funzioni di modifica del testo per l'aggiornamento del programma. Spesso effettua una trasformazione dal sorgente introdotto dall'utente in una forma più compatta, in cui vengono rimossi i separatori inutili mentre le parole chiave vengono rimpiazzate con codici; questa fase di comprensione prende il nome di *riduzione a token* (o, in gergo, "tokenizzazione"). Lo scopo di questo cambiamento di forma è duplice: da una parte si risparmia spazio per la memorizzazione del programma e dall'altra si rende più veloce la traduzione.

Se il testo del programma viene memorizzato in forma compressa l'editor deve fornire anche una funzione inversa di ricostruzione del programma sorgente a partire da questa.

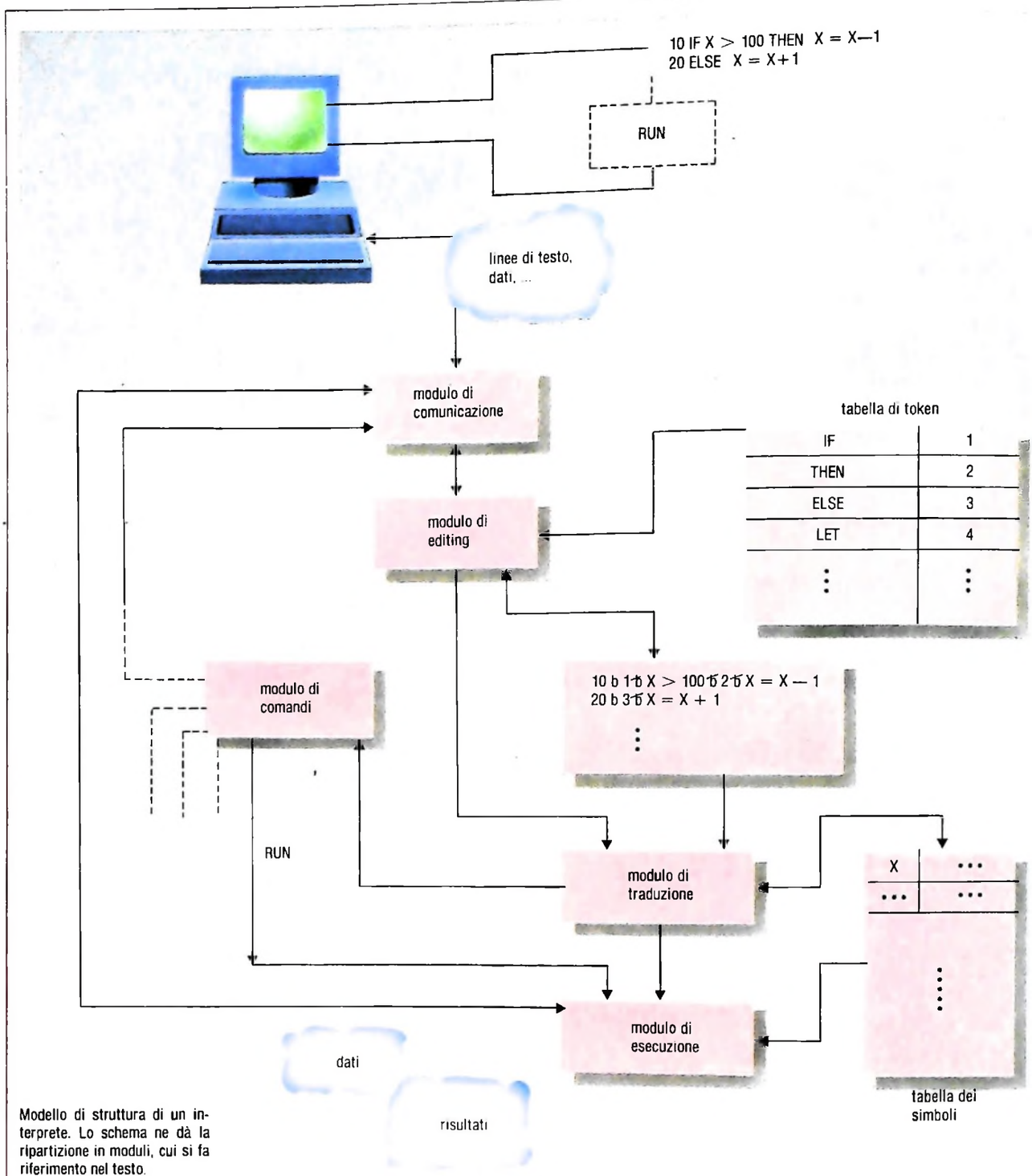
Modulo di traduzione: acquisisce le linee di testo dal modulo di comunicazione. La provenienza delle linee di testo è mascherata dal modulo di editing. Il traduttore distingue tra testo del programma e comandi. Ad esempio, frasi BASIC del tipo

```
10 LET X = A+B/C
```

vengono riconosciute come appartenenti al linguaggio, mentre comandi quali

```
LIST, RUN, SAVE
```

vengono individuati come non appartenenti al linguaggio da tradurre. Nel caso una linea di testo appartenga al linguaggio essa viene analizzata con un processo del tutto identico a quello di un compilatore: viene verificata la correttezza lessicale, sintattica e semantica della linea e l'eventuale presenza



Modello di struttura di un interprete. Lo schema ne dà la ripartizione in moduli, cui si fa riferimento nel testo.

di errori viene scanalata. Terminata la fase di analisi ha luogo la fase di sintesi per cui la linea di programma viene tradotta in una forma interna di più facile interpretazione.
Se, al contrario, la linea non contiene frasi appartenenti al linguaggio riconosciuto dal traduttore, questa viene passata a un altro modulo di interpretazione comandi senza alcuna analisi.
Modulo di comandi: interpreta i comandi che gli vengono passati dal traduttore. Esegue direttamente le funzioni che gli vengono richieste o ricorre ad altri moduli per ottenere i servizi necessari.
Modulo run time: è l'elemento esecutore dell'interprete. Esegue il programma trasformato nella forma interna dal modulo di traduzione, come se questa fosse il linguaggio macchina della macchina astratta definita dall'interprete. Acquisisce i dati necessari all'esecuzione.

Che cos'è un linker

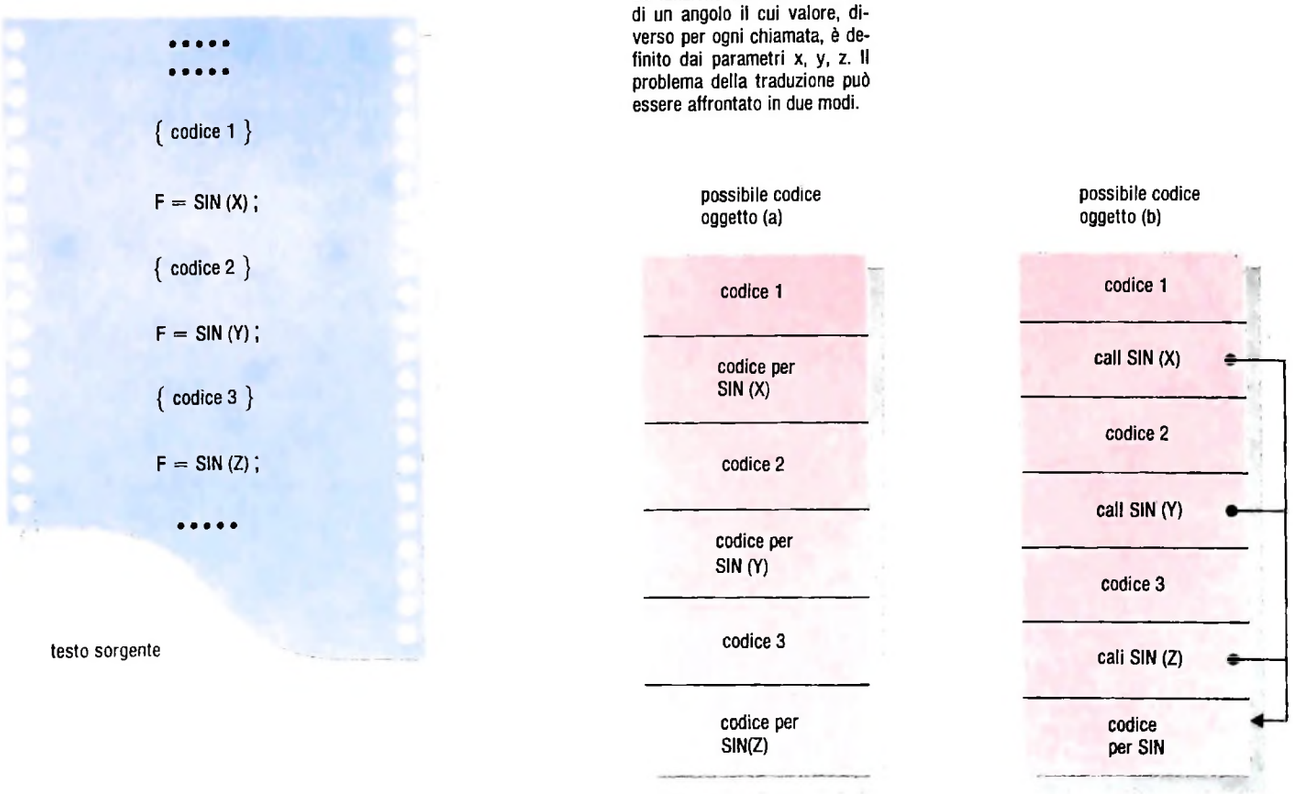
Si è già parlato in precedenza della funzione del linker e si è anche fatto cenno alla sua modalità di funzionamento.
Vediamo ora più precisamente sia le esigenze che portano alla necessità di un tale programma sia il procedimento seguito dal linker per costruire il programma eseguibile.
Da che cosa nasce la necessità di un passo addizionale dopo che la fase di compilazione ha già prodotto codice macchina?
In primo luogo, un programma è spesso costituito da più moduli compilati separatamente, contenenti sottoprogrammi che si richiamano a vicenda.
In secondo luogo, in maniera del tutto analoga alla precedente, quasi tutti i linguaggi di programmazione definiscono dei sottoprogrammi di utilità richiamabili dai programmi utente (funzioni matematiche, procedure di I/O) che, per ragioni di uso ottimale della memoria non vengono espansi nel codice nel punto in cui vengono richiamati.
Infine occorre valutare, relativamente alle aree dati su cui opera un programma, qual è lo spazio totale richiesto per la sua esecuzione, ottenuto come somma delle richieste dei singoli moduli.
Le funzioni di un linker riguardano pertanto:

- la determinazione delle dimensioni delle aree dati;
- la inclusione dei moduli di libreria;
- la risoluzione dei riferimenti esterni di ciascun modulo;
- la generazione di un programma in forma eseguibile (o caricabile).

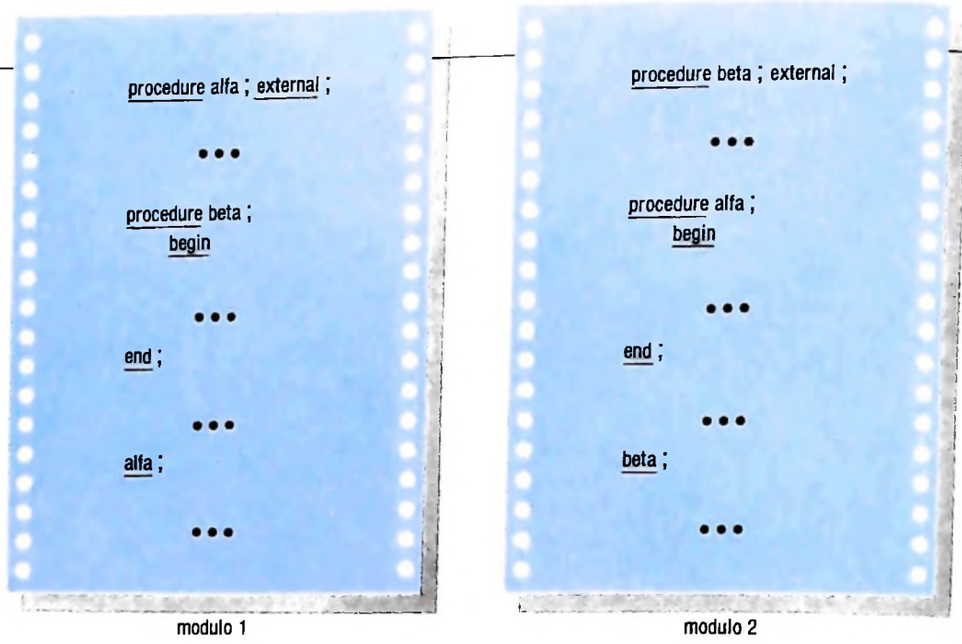
Come funziona un linker

La necessità di utilizzare un linker è legata a due considerazioni: la prima, di natura architetturale, suggerisce di assegnare a strumenti diversi funzioni logicamente distinte; la seconda è invece legata a una effettiva mancanza di informazioni durante la fase di compilazione dei singoli moduli.
Consideriamo ora l'esempio illustrato nella figura in basso.
Un programma effettua, in punti diversi del suo testo sorgente, tre chiamate alla funzione *sin* che valuta il seno di un angolo il cui valore, diverso per ogni chiamata, è definito dai parametri *x*, *y*, *z* e ritorna il valore della funzione in una variabile *f*. Il compilatore potrebbe espandere il codice relativo alle operazioni necessarie alla valutazione della funzione *sin* al posto dei riferimenti alla stessa. Ciò determinerebbe, naturalmente, uno spreco di spazio, tanto maggiore quanto più numerosi sono i riferimenti alla funzione *sin* nel testo del programma. È possibile invece affrontare il problema in modo diverso. Supponiamo di disporre di una libreria di sottoprogrammi, tipo la funzione *sin*, scritta da qualcuno (solitamente dallo stesso gruppo di progetto che ha realizzato il compilatore) e residente, sotto forma di codice oggetto, su disco. Il compilatore può ora ignorare i dettagli delle operazioni necessarie a valutare la funzione *sin* e generare una chiamata a essa una volta che ne conosca il nome.
In questo modo, il codice oggetto della funzione richiesta viene incluso una sola volta nel codice del programma, indipendentemente dal numero di riferimenti che a esso vengono

La funzione sin valuta il seno di un angolo il cui valore, diverso per ogni chiamata, è definito dai parametri x, y, z. Il problema della traduzione può essere affrontato in due modi.

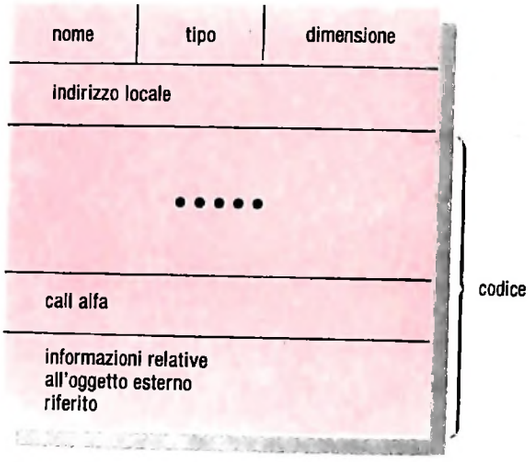


Il linker risolverà i riferimenti quando essi avvengono in modo incrociato tra due moduli, compilati separatamente, del programma utente. In basso: questa sarà la forma di ogni modulo oggetto prodotto dal compilatore.



fatti. Sorge però una difficoltà: il codice della funzione richiesta deve essere presente nel codice del programma almeno una volta e tutti i riferimenti a esso devono sapere dove è stato collocato, per potergli trasferire correttamente il controllo durante l'esecuzione. Sarà pertanto compito del linker individuare la libreria in cui la funzione richiesta è contenuta, estrarne il codice, fonderlo con quello del programma utente e infine scandire quest'ultimo per individuare tutti i riferimenti alla funzione richiesta e aggiornarli con l'indirizzo di caricamento del codice della funzione. Un'operazione identica è necessaria quando i riferimenti avvengono in modo incrociato tra due moduli, compilati separatamente, del programma utente. Si consideri l'esempio della figura in alto. Due moduli definiscono dei sottoprogrammi, beta nel modulo 2 e alfa nel modulo 1, e dei riferimenti incrociati, poiché il modulo 2 richiama il sottoprogramma alfa contenuto nel modulo 1, mentre quest'ultimo richiama il sottoprogramma beta contenuto nel modulo 2. Poiché i due moduli possono venire compilati separatamente, in momenti diversi e in modo del tutto indipendente, il compilatore non è in grado di stabilire, mentre compila modulo 2 dove si trova alfa e, viceversa, dove stia beta durante la compilazione di modulo 1. Di nuovo sarà il linker a risolvere questi riferimenti ma è necessario che il compilatore generi delle informazioni di supporto. Ogni modulo oggetto prodotto dal compilatore avrà dunque una forma come quella illustrata nell'ultima figura. Dovrà essere presente una intestazione contenente tutti i nomi dei simboli esportati e le relative caratteristiche (sottoprogrammi, variabili, ...) con le relative dimensioni (per le variabili) e gli eventuali valori di inizializzazione (sempre per le variabili).

Sparse nel codice si troveranno altre informazioni relative all'uso di oggetti (sottoprogrammi e variabili) non definiti localmente al modulo corrente. Il linker legge in sequenza tutti i moduli da aggregare ed estrae le seguenti informazioni, organizzandole nelle sue tabelle interne:



- nomi esportati e indirizzo di ciascun oggetto a cui il nome si riferisce nell'ambito di ciascun modulo;
- indirizzi dei punti in cui si riferiscono oggetti non locali al modulo;
- dimensioni dei vari oggetti nelle aree dati;
- criteri di sovrapposizione di alcune aree dati (COMMON del FORTRAN);

In una seconda fase il linker rilegge tutti i moduli da cui ha estratto le informazioni di cui sopra, li organizza in un unico file accodandoli e risolve tutti i riferimenti attraverso l'interrogazione delle tabelle costruite nel corso della prima passata. Può accadere però che non tutti i riferimenti vengano risolti nel corso di questa seconda passata nell'ambito dei moduli esaminati, per esempio per la presenza di richiami di funzioni di libreria; è a questo punto che vengono interrogate le librerie opportune da cui viene estratto il codice mancante. Se anche al termine di questa fase ci fossero dei riferimenti non risolti ciò sarebbe dovuto a un errore nel programma sorgente, dovuto a un riferimento a un oggetto inesistente. In assenza di errori il file generato dal linker può essere caricato in memoria per l'esecuzione.

Lezione 50

Altri comandi del microplotter

Esaminiamo ora, con piccole esemplificazioni, tutti gli altri comandi disponibili sul microplotter che non sono stati esaminati negli esempi precedenti.

I comandi in modalità "testo"

La maggior parte dei comandi che abbiamo fin qui esaminato erano eseguibili con il microplotter in modalità grafica, dopo cioè l'esecuzione di un'istruzione del tipo:

```
10 LPRINT CHR$(18)
```

Di fatto il microplotter è manovrabile anche restando in modalità testo, con un certo numero, pur limitato, di comandi.
Così, è possibile usare codici per andare a capo, risalire di una linea, spostarsi all'inizio della linea, tornare indietro di uno spazio. I rispettivi codici sono:

CHR$(08): fa retrocedere il pennino di una posizione di stampa;

CHR$(10): avanza di una linea, senza posizionare il pennino all'inizio della nuova riga;

CHR$(11): si comporta come il precedente, ma fa ruotare il rullo della carta in modo che il pennino si trovi sulla linea precedente a quella di partenza;

CHR$(13): sposta il pennino al margine sinistro nella riga in cui si trova.

Il seguente programma illustra l'uso di tali comandi:

```
10 REM Uso di CHR$ per sottolineare una
parola
20 LPRINT "Sottolinea l'ultima parola";
30 FOR I=1 TO LEN("parola")
40 LPRINT CHR$(08);
50 NEXT I
60 LPRINT "______"
70 REM Uso di CHR$(13) per sottolineare
una riga
80 LPRINT "Sottolinea l'intera riga";CHR
$(13);CHR$(11);"_____________________
"
110 REM CHR$(10) e CHR$(11) per spostars
i in su e in giu' di una riga
120 FOR I=1 TO 20
130 LPRINT"*";
140 LPRINT CHR$(10);
```

```
150 LPRINT"*";
160 LPRINT CHR$(11);
170 NEXT I
```

L'esecuzione del programma fornisce il seguente risultato:

```
Sottolinea l'ultima parola
Sottolinea l'intera riga
* * * * * * * * * * * * * * * * * * * *
 * * * * * * * * * * * * * * * * * * * *
```

Oltre ai codici indicati esiste anche

CHR$(17)

che, usato nella LPRINT quando il microplotter è in modo grafico, lo riporta in modo testo. È l'unico comando di questo tipo che si possa usare quando il microplotter è in modo grafico.
Per passare da modo grafico a modo testo possiamo anche avvalerci del comando "A", nella forma

```
10 LPRINT "A"
```

Cambiamento di direzione

Un comando particolarmente interessante quando si vogliano scrivere parole in verticale, per esempio per affiancare su una retta verticale, che funge da asse, il nome della variabile associata, è il comando "Q".
Esso va, come sempre, usato in modo grafico, e ha l'effetto di cambiare il sistema di riferimento delle scritte fornite dal microplotter.
Il suo formato è il seguente:

Qn

ove

n=0 comporta la normale scrittura sulla riga da sinistra a destra, come siamo soliti fare;

n=1 comporta l'interpretazione della linea di scrittura in verticale, dal basso verso l'alto;

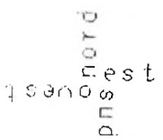
n=2 comporta l'interpretazione della linea di scrittura capovolta, e quindi da destra a sinistra;

n=3 comporta l'interpretazione della linea di stampa in verticale, dall'alto verso il basso.

Così, il programma seguente:

```
10 LPRINT CHR$(18)
20 LPRINT "M200,-400"
30 LPRINT "Q0" 'Da sinistra a destra
40 LPRINT "Pest"
50 LPRINT "M195,-395"
60 LPRINT "Q1" 'Dal basso in alto
70 LPRINT "Pnord"
80 LPRINT "M190,-400"
90 LPRINT "Q2" 'Da destra a sinistra
100 LPRINT"Povest"
110 LPRINT"M195,-405"
120 LPRINT"Q3" 'Dall'Alto in basso
130 LPRINT"Psud"
```

posiziona il pennino del microplotter indicativamente verso il centro del foglio, quindi stampa "est" normalmente, da sinistra a destra, poi posiziona il pennino in una posizione vicina a quella precedente, cambia la direzione di stampa in "da basso in alto", stampa "nord", e così via con gli altri punti cardinali.
Il risultato che si ottiene è illustrato nella figura seguente:



Le linee tratteggiate

Talvolta abbiamo interesse che una linea venga tracciata non come linea continua, ma tratteggiata. Il comando

Ln

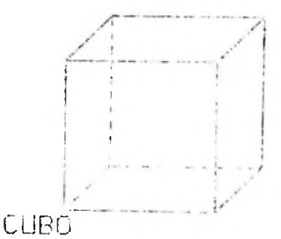
è ciò che ci serve:
n=0 indica che la linea sarà continua
n<>0 indica che la linea sarà tratteggiata: il valore 1 fornisce il tratteggio più piccolo, mentre il valore 15, che è il massimo possibile, fornisce il tratteggio più grande.

Il seguente esempio sfrutta tale istruzione per tracciare l'assonometria di un cubo, con il tratteggio degli spigoli nascosti:

```
10 LPRINT CHR$(18)
15 LPRINT "L0" 'Linea continua
```

```
20 REM Traccia un cubo con linee nascoste
30 LPRINT "M100,-100" 'In mezzo
35 REM Piano anteriore
40 LPRINT "D100,-200,200,-200,200,-100,
100,-100"
45 REM Spigoli e piano posteriore
50 LPRINT"D130,-70,230,-70,230,-170,200,
-200"
55 REM Ultimo spigolo
60 LPRINT "M200,-100"
70 LPRINT "D230,-70"
80 LPRINT "L5" 'Linee tratteggiate
90 REM Linee nascoste
100 LPRINT "M130,-70"
110 LPRINT "D130,-170,100,-200"
120 LPRINT "M130,-170"
130 LPRINT "D230,-170"
140 LPRINT "A" 'Modo testo
160 LPRINT CHR$(13);"      CUBO"
```

il risultato della sua esecuzione è il seguente:



Cosa abbiamo imparato

In questa lezione abbiamo visto:

- il comando CHR$(08), da usare in modo testo, che fa retrocedere il pennino di una posizione di stampa;
- il comando CHR$(10), da usare in modo testo, che avanza di una linea;
- il comando CHR$(11), da usare in modo testo, che fa risalire il pennino di una linea;
- il comando CHR$(13), da usare in modo testo, che sposta il pennino a capo;
- il comando CHR$(17), da usare in modo grafico, che fa ritornare al modo testo;
- il comando A, da usare in modo grafico, che fa ritornare in modo testo;
- il comando Q, da usare in modo grafico, che permette di capovolgere la stampa o di scrivere in verticale;
- il comando L, da usare in modo grafico, per definire che le linee tracciate saranno continue o tratteggiate.

LE STAMPANTI (I)

Una periferica tanto importante da condizionare la funzionalità dell'intero sistema computerizzato.

La diffusione capillare dell'automazione e delle apparecchiature elettroniche per il trattamento dell'informazione ha portato a una rivoluzione nei sistemi di comunicazione fra gli uomini, e questo spesso genera timori infondati sul futuro della comunicazione scritta. Nonostante tutto la carta non scomparirà mai. Anzi, assumerà un'importanza ancora maggiore, perché fungerà comunque da supporto permanente delle elaborazioni di dati e informazioni acquisite e trattate nella maniera più efficiente e precisa possibile.
In molti casi, acquistare un sistema computerizzato privo di una stampante è come comperare un'automobile di lusso senza accessori e optional di innegabile utilità. Nella maggioranza delle applicazioni, infatti, avere una copia su carta dei dati che compaiono sul video del computer è una necessità di fatto, e risolve non pochi problemi. Senza contare che in molti casi lo scopo principale dell'impiego del personal è proprio quello di fornire una razionalizzazione e una raccolta su carta delle informazioni.
Per forza di cose, dunque, la stampante diventa una periferica di primaria importanza, e va scelta con attenzione. Il criterio di base deve essere, come sempre, la funzionalità all'interno del sistema computerizzato. Una stampante lenta o po-

La stampante costituisce una parte irrinunciabile di un moderno sistema computerizzato. A seconda delle diverse esigenze, si può operare una scelta fra stampanti a matrici di punti e stampanti a margherita.

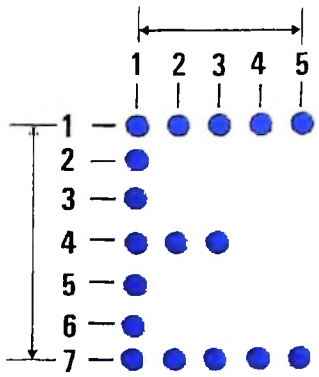
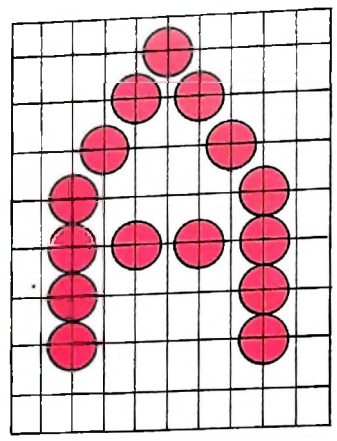
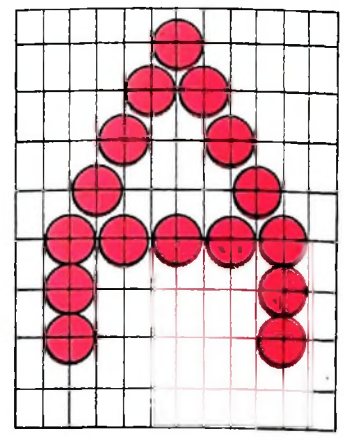


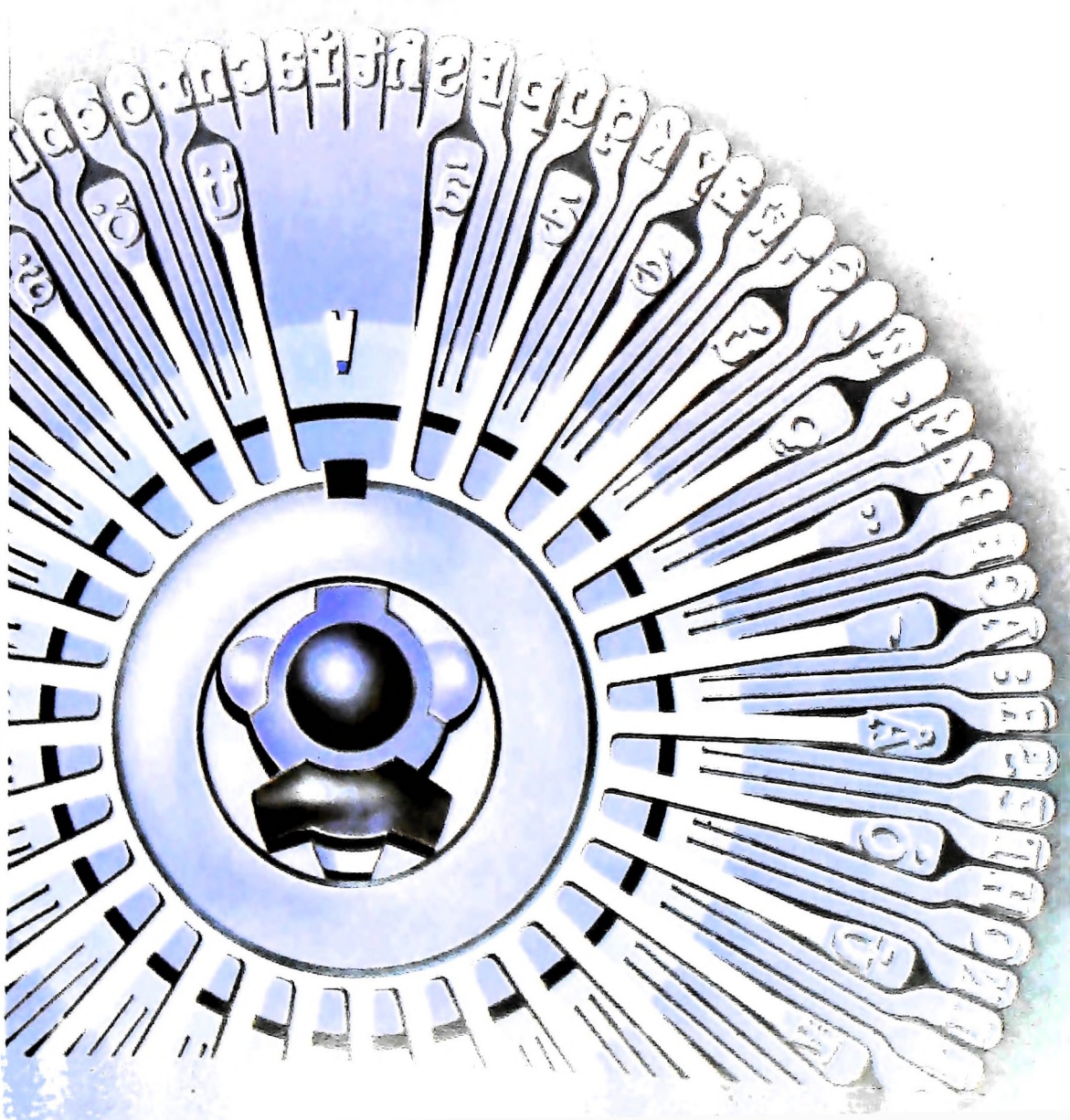
Immagine a punti con matrice 5x7.





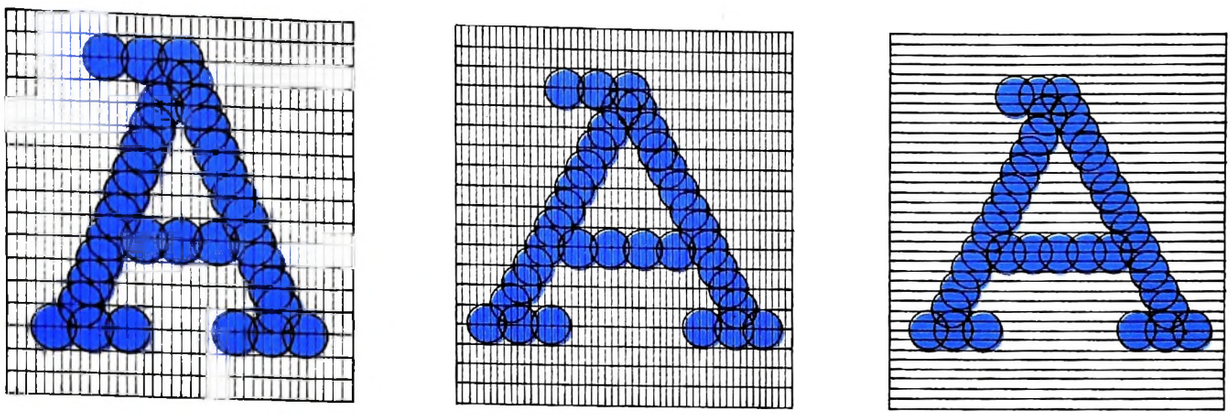
co efficiente rischia di rendere inutile un computer superveloce del costo di parecchi milioni, perché la sua inadeguatezza al sistema crea un "collo di bottiglia" in fase di stampa, provocando un rallentamento generale delle operazioni trattate in forma elettronica. Di contro, una stampante a margherita da quattro milioni collegata a un piccolo home computer non potrà mai essere sfruttata a fondo. È bene quindi avere le idee chiare sulle varie tecnologie disponibili, e sui rispettivi vantaggi e svantaggi.

La suddivisione principale fra i diversi tipi di stampanti è fatta in base alla tecnica generale di stampa, che può essere a impatto (contatto meccanico tra la carta e il meccanismo di stampa) o senza impatto (trasferimento d'inchiostro o alterazione controllata del colore della carta di supporto). In termini pratici, le stampanti a impatto sono generalmente più rumorose di quelle senza impatto, ma possiedono caratteristiche aggiuntive che spesso le fanno preferire. Vediamo in sintesi una panoramica sulle varie tecnologie disponibili.



A

In questa pagina: esempi di caratteri ottenuti con stampanti a matrici di punti, insieme a un'immagine di "margherita" per stampante. Una stampante a margherita fa uso di un disco di plastica, attorno al bordo del quale è fissata una serie completa di caratteri da stampa.

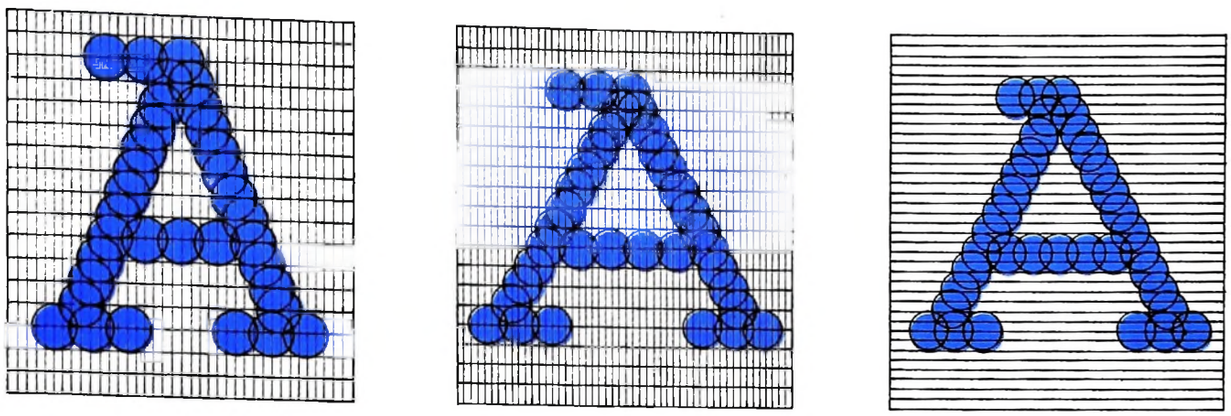


Stampanti ad aghi

Sono chiamate anche "a matrice di punti" (dall'inglese "dot matrix") perché il carattere è formato da una serie di piccoli punti, con un diametro medio di qualche decimo di millimetro, disposti in una matrice 5x7 o 9x11. La dimensione della matrice, all'interno della quale si ricava il carattere, è importante ai fini della qualità di stampa: più la matrice è densa, più complesso e raffinato è il carattere.
La stampa avviene per mezzo di piccoli aghi inseriti in una testina mobile montata su un carrellino portanastro che scorre parallelamente alla linea di stampa, a velocità costante. Ciascun ago è comandato da un elettromagnete che, a seconda dei segnali ricevuti dall'elaboratore, spinge l'ago contro il nastro inchiostrato a ridosso della carta, stampando un punto. Il carattere completo è formato da più punti stampati da più aghi che lavorano in sincronismo. (Nella figura sono rappresentati esempi di scrittura a punti costruiti con densità crescente.)
Questa tecnica di stampa, pilotata da un microprocessore, permette di utilizzare più set di caratteri di diversa forma e dimensioni, i cui parametri sono contenuti in una memoria ROM inserita nella stampante.
I caratteri possono essere compressi o espansi per ottenere stampe in formati diversi, e per realizzare intestazioni e titoli migliorando la leggibilità dei documenti. Alcune stampanti a matrice montano ROM di caratteri intercambiabili, che possono essere sostituite quando si voglia stampare documenti con caratteri speciali, come le lettere accentate dei vari alfabeti nazionali. Le stampanti dell'ultima generazione sono dotate di ROM multipla, che contiene diversi set di caratteri nazionali selezionabili via software al momento dell'installazione del sistema operativo.
La velocità delle stampanti a matrice varia tra i 20 e i 400 caratteri al secondo. La tecnica di stampa a matrice di punti è la più adatta per produrre listati di programmi, bozze di lettere e tabulati. Inoltre è l'ideale per riprodurre sulla carta immagini costruite in alta risoluzione sul video del computer. Alcuni modelli molto evoluti sono in grado di produrre stampe di qualità di testi e tabelle, passando la testina due o tre volte sulla stessa riga, per rendere più inciso e definito il carattere. Questa tecnica si chiama "letter quality" quando offre risultati perfetti, o "near-letter quality" quando il prodotto, pur non essendo perfetto, è comunque di ottima leggibilità e più che accettabile per documenti ufficiali.

Stampanti a margherita

Il termine deriva dall'inglese "daisy wheel", che si riferisce alla forma a raggiera del supporto di plastica che contiene il set di caratteri. Le margherite per le stampanti e la tecnica di impressione del carattere sono identiche a quelle utilizzate dalle moderne macchine per scrivere elettroniche. Il trasferimento dell'inchiostro sul foglio di carta avviene a opera di un martelletto che spinge contro la carta uno dei "petali" della margherita di plastica. Sui petali sono riportati tutti i caratteri alfabetici e numerici che compongono il set. Ruotando velocemente la margherita davanti al martelletto in una sequenza opportuna, si ottiene la composizione delle parole e delle frasi del testo.
Il vantaggio principale delle stampanti a margherita è la possibilità di sostituire rapidamente l'intero set di caratteri, scegliendolo fra una vasta gamma di combinazioni. La sostituzione è manuale, e va fatta a macchina ferma smontando la margherita vecchia e montandone una nuova. Questo purtroppo impedisce di stampare su un singolo foglio un testo con diversi stili e caratteri, come avviene nelle stampanti a matrice, che formano il carattere punto per punto, pilotando gli aghi uno per uno secondo le istruzioni contenute nella ROM dei caratteri.
Tuttavia, la qualità di stampa di una margherita è molto elevata, e solitamente è impossibile distinguere una lettera battuta con una macchina per scrivere elettronica da quella stampata con una stampante a margherita.
Gli svantaggi della stampante a margherita sono due: la lentezza della stampa, e il costo elevato dell'apparecchiatura. La velocità operativa media delle stampanti a margherita si ag-



Stampanti ad aghi

Sono chiamate anche "a matrice di punti" (dall'inglese "dot matrix") perché il carattere è formato da una serie di piccoli punti, con un diametro medio di qualche decimo di millimetro, disposti in una matrice 5x7 o 9x11. La dimensione della matrice, all'interno della quale si ricava il carattere, è importante ai fini della qualità di stampa: più la matrice è densa, più complesso e raffinato è il carattere.
La stampa avviene per mezzo di piccoli aghi inseriti in una testina mobile montata su un carrellino portanastro che scorre parallelamente alla linea di stampa, a velocità costante. Ciascun ago è comandato da un elettromagnete che, a seconda dei segnali ricevuti dall'elaboratore, spinge l'ago contro il nastro inchiostrato a ridosso della carta, stampando un punto. Il carattere completo è formato da più punti stampati da più aghi che lavorano in sincronismo. (Nella figura sono rappresentati esempi di scrittura a punti costruiti con densità crescente.)
Questa tecnica di stampa, pilotata da un microprocessore, permette di utilizzare più set di caratteri di diversa forma e dimensioni, i cui parametri sono contenuti in una memoria ROM inserita nella stampante.
I caratteri possono essere compressi o espansi per ottenere stampe in formati diversi, e per realizzare intestazioni e titoli migliorando la leggibilità dei documenti. Alcune stampanti a matrice montano ROM di caratteri intercambiabili, che possono essere sostituite quando si voglia stampare documenti con caratteri speciali, come le lettere accentate dei vari alfabeti nazionali. Le stampanti dell'ultima generazione sono dotate di ROM multipla, che contiene diversi set di caratteri nazionali selezionabili via software al momento dell'installazione del sistema operativo.
La velocità delle stampanti a matrice varia tra i 20 e i 400 caratteri al secondo. La tecnica di stampa a matrice di punti è la più adatta per produrre listati di programmi, bozze di lettere e tabulati. Inoltre è l'ideale per riprodurre sulla carta immagini costruite in alta risoluzione sul video del computer. Alcuni modelli molto evoluti sono in grado di produrre stampe di qualità di testi e tabelle, passando la testina due o tre volte sulla stessa riga, per rendere più inciso e definito il carattere. Questa tecnica si chiama "letter quality" quando offre risultati perfetti, o "near-letter quality" quando il prodotto, pur non essendo perfetto, è comunque di ottima leggibilità e più che accettabile per documenti ufficiali.

Stampanti a margherita

Il termine deriva dall'inglese "daisy wheel", che si riferisce alla forma a raggiera del supporto di plastica che contiene il set di caratteri. Le margherite per le stampanti e la tecnica di impressione del carattere sono identiche a quelle utilizzate dalle moderne macchine per scrivere elettroniche. Il trasferimento dell'inchiostro sul foglio di carta avviene a opera di un martelletto che spinge contro la carta uno dei "petali" della margherita di plastica. Sui petali sono riportati tutti i caratteri alfabetici e numerici che compongono il set. Ruotando velocemente la margherita davanti al martelletto in una sequenza opportuna, si ottiene la composizione delle parole e delle frasi del testo.
Il vantaggio principale delle stampanti a margherita è la possibilità di sostituire rapidamente l'intero set di caratteri, scegliendolo fra una vasta gamma di combinazioni. La sostituzione è manuale, e va fatta a macchina ferma smontando la margherita vecchia e montandone una nuova. Questo purtroppo impedisce di stampare su un singolo foglio un testo con diversi stili e caratteri, come avviene nelle stampanti a matrice, che formano il carattere punto per punto, pilotando gli aghi uno per uno secondo le istruzioni contenute nella ROM dei caratteri.
Tuttavia, la qualità di stampa di una margherita è molto elevata, e solitamente è impossibile distinguere una lettera battuta con una macchina per scrivere elettronica da quella stampata con una stampante a margherita.
Gli svantaggi della stampante a margherita sono due: la lentezza della stampa, e il costo elevato dell'apparecchiatura. La velocità operativa media delle stampanti a margherita si ag-

gira intorno ai 20 caratteri al secondo, ed è accettabile per scrivere lettere commerciali, ma non per testi di notevole lunghezza. Il costo supera il milione e mezzo di lire, e può arrivare anche ai quattro milioni per i modelli dotati di dispositivo per il caricamento automatico dei fogli singoli.

Il buffer

Poiché l'elaboratore invia i dati alla stampante a velocità elevata, e la stampante non è in grado di tenere il passo, è necessario che quest'ultima disponga di una memoria transitoria, chiamata memoria tampone o "buffer". Il buffer è in grado di accogliere un certo numero di caratteri che provengono dal computer, e di avviarli alla stampa non appena la testina ha completato il suo ciclo di scrittura. Quando il buffer è pieno, avverte il computer di attendere, e invia i dati al microprocessore interno che pilota la testina di stampa alla massima velocità possibile. Normalmente il buffer contiene una o due linee di caratteri (una trentina di parole). Terminata la scrittura di una o più linee, la stampante comunica all'elaboratore di essere pronta a ricevere altri dati.
Alcune stampanti, solitamente le più perfezionate e costose fra quelle a margherita, dispongono di buffer con una capacità di qualche migliaio di caratteri. Questo permette al computer di scaricare istantaneamente il testo da stampare, ed essere pronto per altre operazioni. La tecnica, chiamata "spooling", si rivela particolarmente utile nelle applicazioni di word processing, quando si desidera proseguire nelle operazioni di scrittura mentre la stampante lavora.

Glossario

BASIC - *sigla di Beginner's All-purpose Symbolic Instruction Code (codice d'istruzioni simboliche di uso generale per principianti), indica un linguaggio di programmazione di alto livello. Il BASIC è il linguaggio più diffuso sui personal computer; per la sua relativa semplicità può essere appreso rapidamente, e, per il fatto di essere un linguaggio interpretato, permette un'interazione diretta fra l'utente e la macchina che, soprattutto per i principianti, risulta molto efficace. Come dice il nome, è un linguaggio di uso generale. Esistono del BASIC numerosi "dialetti", che coincidono negli elementi fondamentali, ma differiscono profondamente per altri, in particolare per caratteristiche dipendenti dalla macchina come la grafica, il colore, la gestione dei suoni.*

BCD - *sigla di Binary Coded Decimal (decimale codificato in binario), indica un metodo di codificazione dei numeri decimali, utilizzato per la memorizzazione in un computer. Anziché i numeri, vengono codificate le cifre: una sequenza di 0 e 1 corrisponde all'1 decimale, un'altra al 2, una al tre, e via dicendo. È una codificazione meno efficiente rispetto al sistema di numerazione binario puro, e in particolare richiede un maggiore spazio di memoria e rende più lenta l'esecuzione delle operazioni aritmetiche, tuttavia permette di ottenere risultati di grande precisione.*

Compilatore - *un programma "traduttore" per linguaggi di alto livello: traduce programmi scritti in linguaggi di alto livello (programmi sorgente) in programmi scritti in codice di macchina (programmi oggetto). La traduzione viene effettuata una sola volta (fase di compilazione) su tutto il programma: quella che viene eseguita è poi sempre la versione compilata. I linguaggi per cui esiste un compilatore si dicono linguaggi compilati (in contrapposizione ai linguaggi interpretati, tradotti da un interprete). A differenza di un interprete, un compilatore può rendere faticosa la costruzione di un programma: se in fase di compilazione viene identificato un errore, il programmatore deve riprendere in considerazione (usando un apposito programma per la stesura di programmi sorgente, un text editor tipicamente) il programma sorgente nel suo complesso, poi tornare a compilarlo interamente. In compenso, l'esecuzione dei programmi, una volta compilati, è molto più veloce rispetto all'esecuzione dei programmi scritti in linguaggi interpretati.*

Flip-flop - *un elemento circuitale elettronico, che è in grado di assumere due soli stati, di commutare dall'uno all'altro secondo modalità ben precise e riproducibili e, in mancanza di impulsi di commutazione, rimane stabilmente nello stato in cui si trova. Il tipo più semplice di flip-flop può essere realizzato con due sole porte logiche di tipo NAND: il comportamento del circuito (indicato come flip-flop RS) dipende sia dall'ingresso, sia dall'uscita delle due porte. I flip-flop, che sono chiamati anche bistabili, costituiscono gli elementi fondamentali per la costruzione dei circuiti di memoria.*

Interprete - *un programma "traduttore" per linguaggi di alto livello: prende una riga di programma alla volta e sovrintende immediatamente alla sua traduzione in codice macchina e alla sua esecuzione. Linguaggi tradotti da un interprete sono detti "interpretati", in contrapposizione ai linguaggi "compilati" (tradotti da un compilatore). Linguaggi interpretati sono tipicamente il BASIC, l'APL, il LISP, il PROLOG: tuttavia un linguaggio interpretato può essere anche compilato (esistono, per esempio, anche compilatori per il BASIC). L'interprete permette una forma veloce di interazione fra l'uomo e la macchina, in fase di realizzazione dei programmi, ma in fase esecutiva è molto più lento di un compilatore, perché procede esaminando una sola riga di programma alla volta; se la medesima istruzione si presenta decine o centinaia di volte nel corso di un programma, in righe diverse, deve venir tradotta dall'interprete ogni volta come se fosse un'istruzione mai incontrata.*

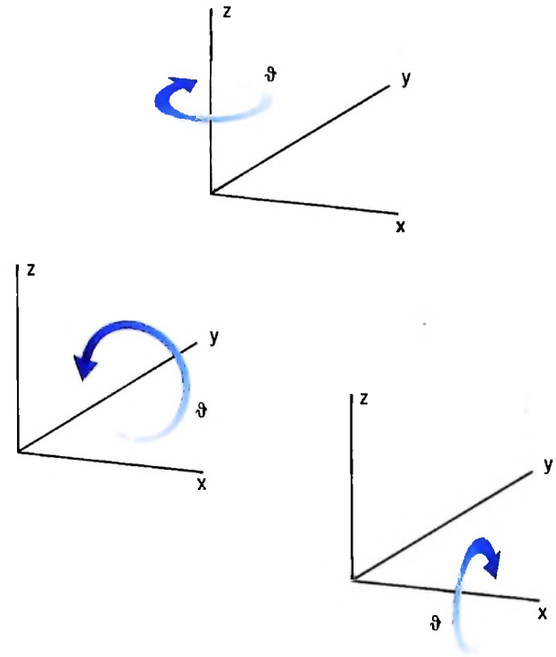
LISP - *acronimo di LISt Processing, indica un linguaggio di programmazione di alto livello, formulato verso la fine degli anni Cinquanta da John McCarthy, orientato in particolare modo all'elaborazione simbolica. È il linguaggio tipico degli studi sull'Intelligenza Artificiale. Linguaggio tipicamente interpretato, è anche un linguaggio "funzionale": tutte le costruzioni del LISP assumono la forma di funzioni. LISP è anche un linguaggio fortemente ricorsivo.*

Macchina di Turing - *il modello più astratto e più potente di macchina calcolatrice, formulato nel 1936 dal matematico inglese Alan M. Turing per dare una definizione rigorosa dei concetti di algoritmo e procedura effettiva. Una macchina di Turing (che, evidentemente, è una macchina ideale), è costituita da un nastro infinito diviso in celle, su cui possono essere scritti simboli tratti da un insieme finito, e da una testina di lettura/scrittura, che può muoversi lungo le due direzioni del nastro di una sola cella alla volta. L'unità di elaborazione di una macchina di Turing è un automa finito, che può assumere un numero finito di stati interni e fornire un'uscita ben determinata (lettura, scrittura, cancellazione di simboli) in funzione dell'ingresso e dello stato in cui si trova.*

LE TRASFORMAZIONI TRIDIMENSIONALI

Analizziamo in dettaglio il funzionamento delle singole matrici che eseguono i vari tipi di trasformazioni.

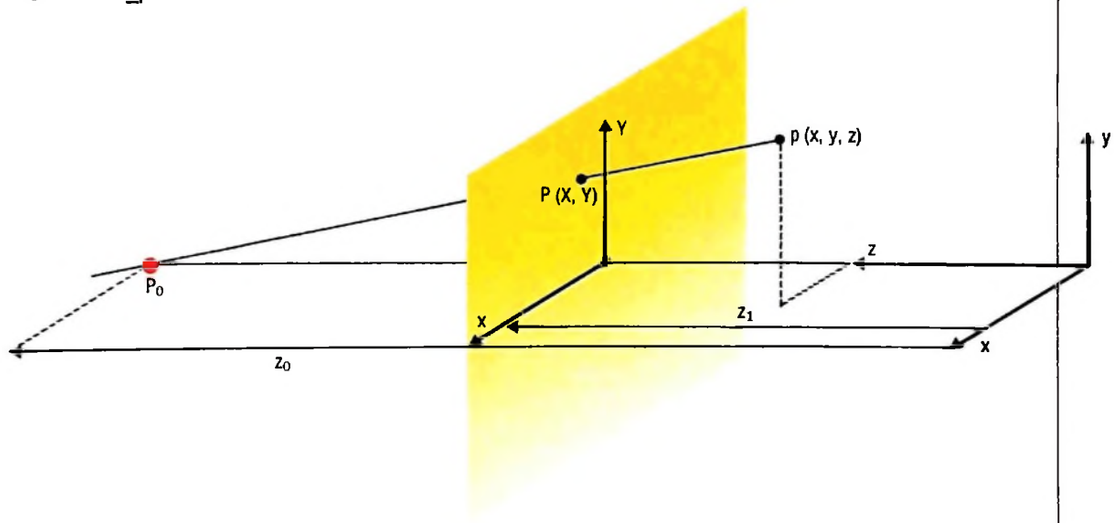
Nell'articolo precedente abbiamo visto il significato geometrico delle coordinate omogenee e abbiamo illustrato come sia geometricamente più completa la trattazione delle problematiche relative a uno spazio a n dimensioni ambientandole in uno spazio a n+1 dimensioni, che viene detto "spazio proiettivo" e le cui coordinate vengono chiamate omogenee. Questo artificio ci consente infatti di trattare anche i punti posti all'infinito mediante la semplice aggiunta di una coordinata omogenea. Abbiamo inoltre anticipato come l'insieme di tutte le possibili trasformazioni tridimensionali sia rappresentabile matematicamente mediante una matrice 4x4. In questo numero affronteremo più in dettaglio i principali tipi di trasformazione.



Le traslazioni tridimensionali

La trasformazione che trasla un punto (x, y, z) in un nuovo punto (x', y', z') è data da:

$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix}$$



Sopra: tre rotazioni primitive. Gli angoli sono misurati in senso orario guardando verso l'origine da un punto qualsiasi posto lungo il semiasse positivo dell'asse di rotazione. Sotto: costruzione della proiezione da cui sono ricavabili le relazioni algebriche fondamentali che legano le coordinate dei punti di un oggetto.

dove Tx, Ty, Tz sono gli incrementi da assegnare alle componenti lungo i tre assi coordinati.

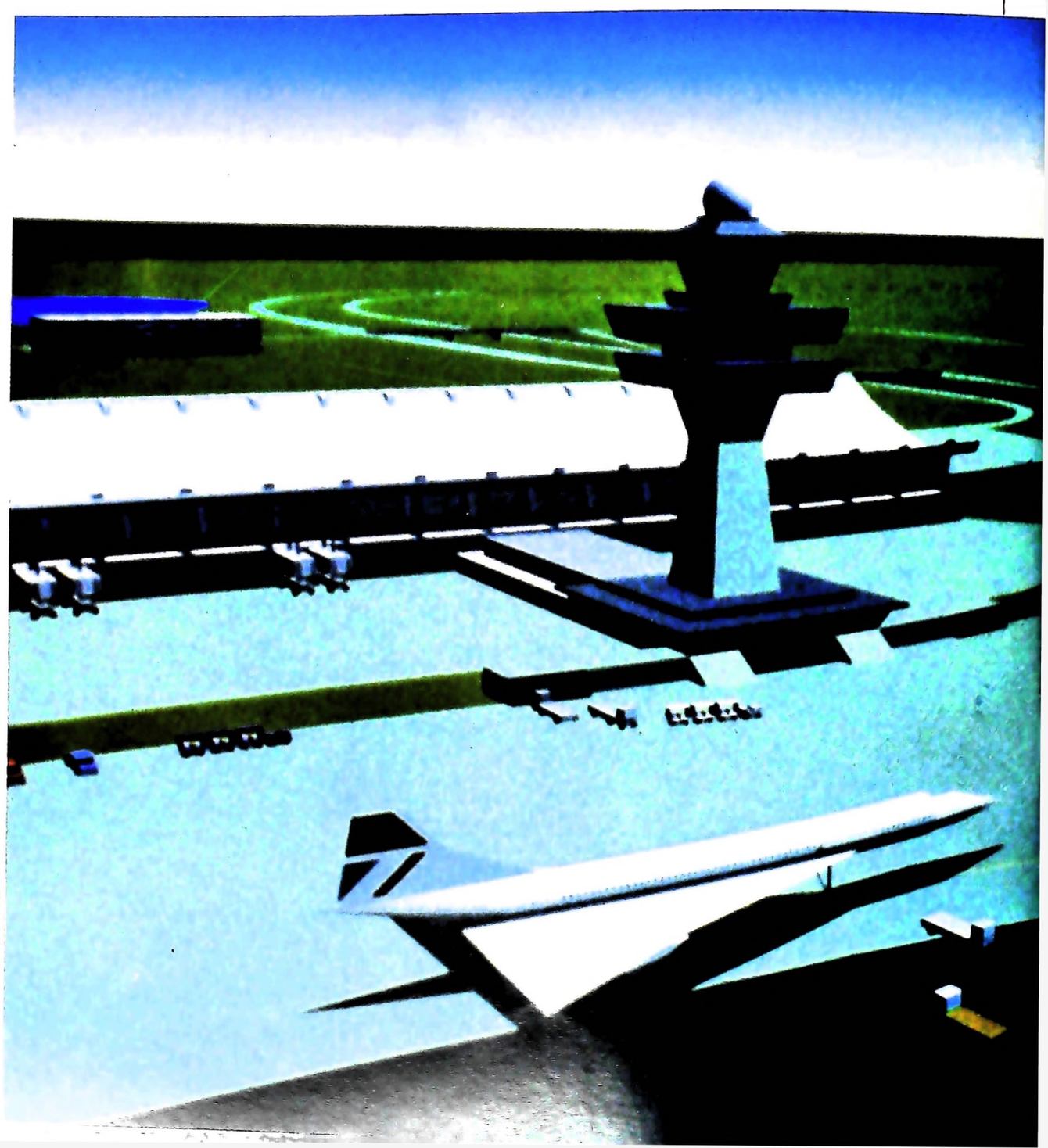
Trasformazioni di scala tridimensionali

I termini diagonali della generica matrice 4x4 di trasformazione producono una scalatura dell'immagine. Una tale trasformazione può essere usata per modificare le dimensioni dell'immagine in modo differenziato lungo ogni direzione:

$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1] \cdot \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

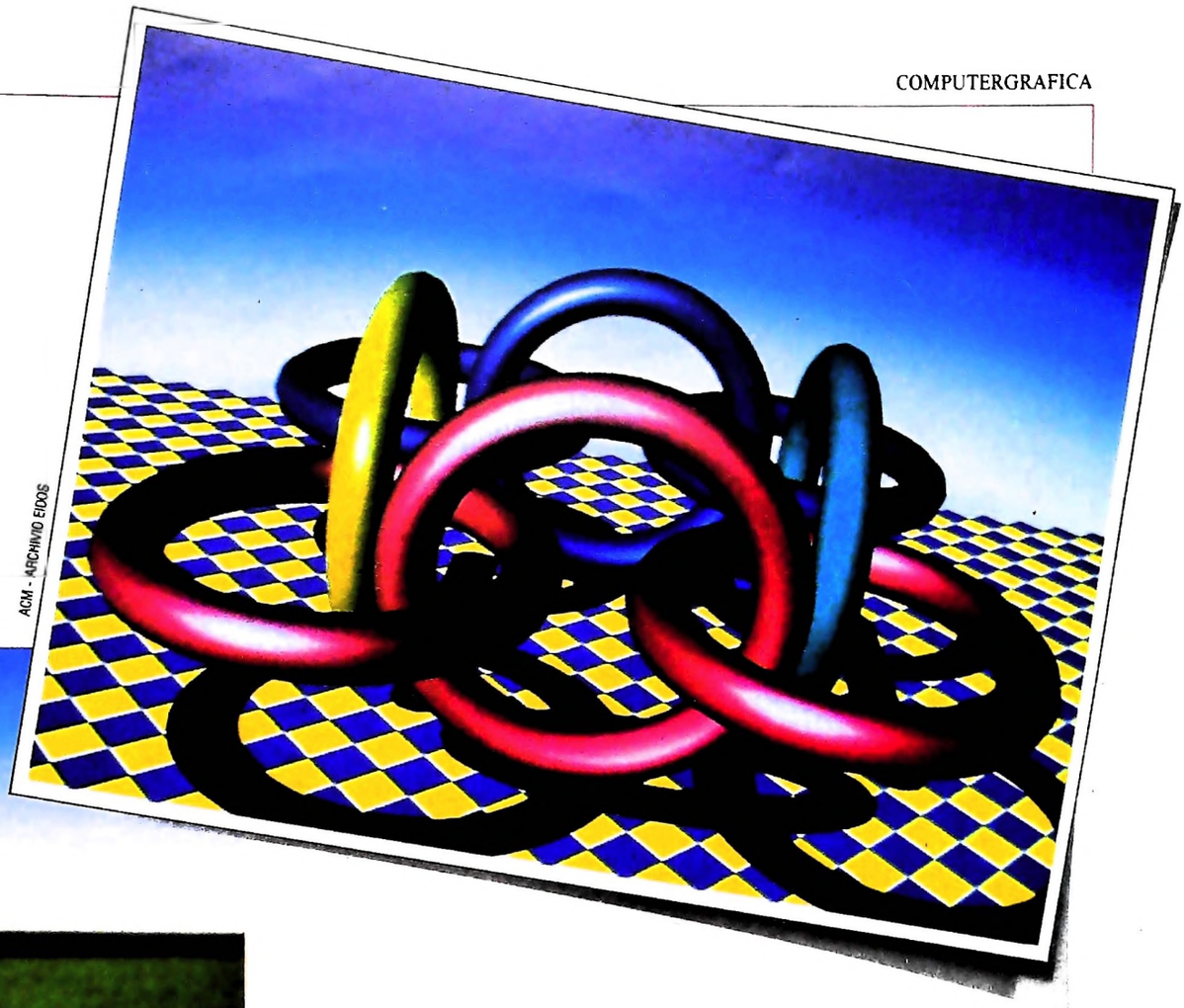
Le rotazioni tridimensionali

Nel caso delle trasformazioni di rotazione in tre dimensioni occorre determinare un asse di rotazione. Come per il caso bidimensionale, la situazione più semplice è quella in cui l'asse di rotazione passa per l'origine e coincide con uno dei tre assi coordinati x, y o z. Nel piano, una rotazione attorno all'origine corrisponde a una rotazione attorno all'ipotetico asse z passante per l'origine del sistema di riferimento ed emergente in direzione perpendicolare al piano stesso: ossia, la rotazione bidimensionale risulta essere un particolare tipo di rotazione tridimensionale.
Per effettuare una rotazione attorno a un generico asse non coincidente con uno degli assi coordinati bisognerà invece concatenare due o tre rotazioni primitive (cioè trasformazio-

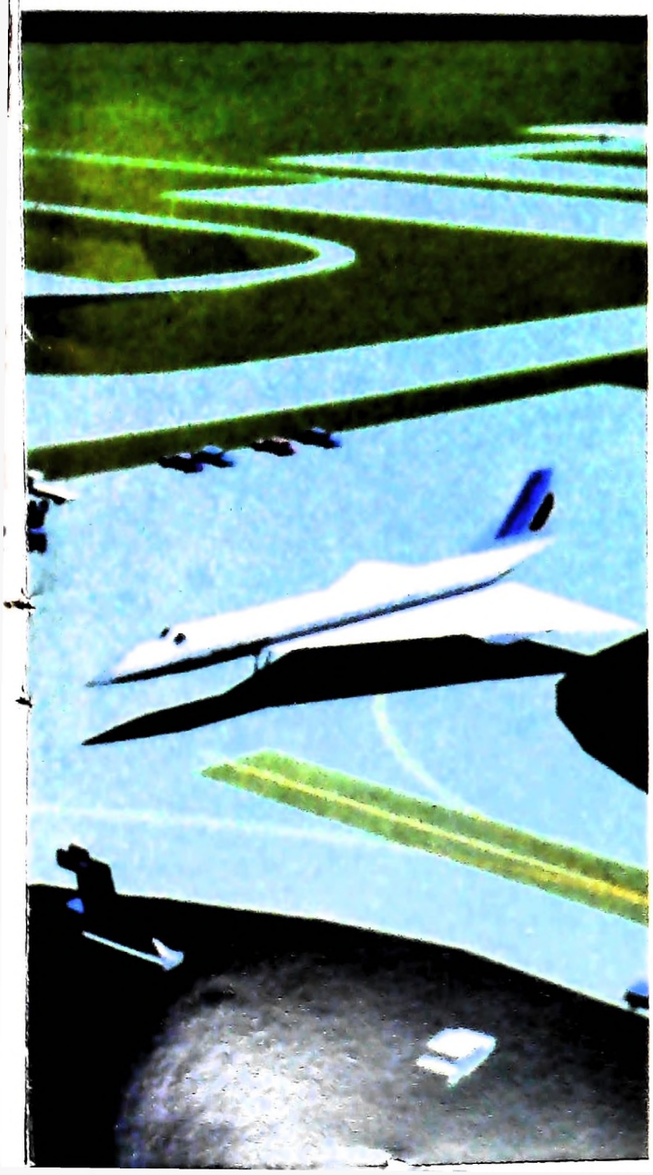


ADM - ARCHIVIO EIDOS

Sotto: la rappresentazione al computer di un aeroporto. In questa immagine risultano evidenti gli studi di prospettiva e di ombre dirette. Il calcolatore consente di ottenere sovrapposizioni di uno stesso modello tridimensionale (come nel caso dell'immagine a lato) dando origine a forme di grande suggestione. L'effetto del realismo, anche in questo caso, viene evidenziato dalla presenza di sorgenti luminose e di ombreggiature.



ACM - ARCHIVIO EIDOS



ni i cui assi di rotazione sono o x o y oppure z) al fine di ottenere la matrice che effettua la rotazione adeguata attorno all'asse desiderato. Per semplicità, limiteremo qui la nostra trattazione ai soli casi relativi alle rotazioni primitive.
La rotazione di un angolo α attorno all'asse coordinato delle z è generata dalla seguente trasformazione, che è del tutto analoga a quella vista per il caso bidimensionale:

$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1] \cdot \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

L'angolo α di rotazione va inteso calcolato in senso orario attorno all'origine quando si guardi l'origine stessa da un punto appartenente al semiasse positivo delle z (figura in alto di pagina 805). In questa trasformazione la componente del vettore relativa alla z rimane inalterata, mentre si modificano quelle lungo le direzioni x e y.
La rotazione attorno all'asse coordinato delle y è dato da:

$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1] \cdot \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La presenza degli zeri nella seconda riga e nella seconda colonna, con l'eccezione del termine 1 sulla diagonale principale, consente di mantenere inalterato il valore della componente y del vettore [x y z 1].
La rotazione attorno all'asse coordinato delle x è data da:

$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Le tre trasformazioni possono essere composte successivamente a formare un'unica matrice che caratterizza la rotazione simultanea attorno ai tre assi. Va però ricordato che la concatenazione di due o tre di queste rotazioni primitive è frutto della moltiplicazione righe per colonne fra le relative matrici. Non essendo tale prodotto commutativo, come già visto in una precedente lezione, è quindi di fondamentale importanza l'ordine con cui si effettuano le moltiplicazioni, dato che la sequenza seguita può alterare il risultato finale. A dimostrazione di ciò, consideriamo una rotazione attorno all'asse delle x seguita da una uguale rotazione attorno all'asse delle y. Utilizzando le matrici sopra esposte, avremo che la matrice T, risultato della concatenazione delle due, è data da:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ \sin^2\alpha & \cos\alpha & -\sin\alpha\cos\alpha & 0 \\ -\sin\alpha\cos\alpha & \sin\alpha & \cos^2\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Viceversa, se consideriamo dapprima la rotazione attorno all'asse delle y e poi quella attorno all'asse delle x, la matrice T' concatenazione delle due diviene:

$$T' = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\alpha & \sin^2\alpha & \sin\alpha\cos\alpha & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ -\sin\alpha & \sin\alpha\cos\alpha & \cos^2\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Confrontando ora le due matrici T e T' si può verificare che esse sono diverse. Quindi il fatto che le rotazioni tridimensionali non sono commutative deve sempre essere tenuto ben presente quando si devono effettuare più rotazioni concatenate fra loro.

Le proiezioni geometriche piane

Le trasformazioni che abbiamo considerato fino a questo punto, come abbiamo già avuto modo di rilevare per il caso bidimensionale, possiedono la proprietà di conservare alcune importanti relazioni delle figure su cui vengono applicate. In particolare la traslazione e la rotazione sono trasformazioni isometriche, ovvero conservano le misure associate alla figura, ossia le lunghezze e gli angoli; la trasformazione di scala non è invece isometrica ma isogonica, dato che conserva solo gli angoli; mentre non isometriche sono le trasformazioni proiettive, che consentono di generare una rappresentazione prospettica di una figura solida su di un piano.
Al problema della costruzione di una rappresentazione di un oggetto solido su un piano sono state date soluzioni di tipo geometrico fin dal Cinquecento; a noi tuttavia interessa dare una soluzione di tipo algebrico, per poter trattare con l'elaboratore la restituzione prospettica. Nella figura in basso di pagina 805 è illustrato il principio generale della proiezione, da cui si ricavano le relazioni algebriche fondamentali che legano le coordinate dei punti di un oggetto. In questa figura si considera il caso in cui la direzione di osservazione, che coincide con il centro di proiezione, è parallela all'asse delle z e quindi perpendicolare al piano coordinato xy del sistema di riferimento. Inoltre il piano di proiezione indicato con XY, per generalizzare, non viene fatto coincidere con il piano coordinato xy stesso. Le coordinate X,Y del punto sul piano di proiezione si ottengono semplicemente dalla relazione di similitudine che lega i vari segmenti $z_0\ z_1$ e le coordinate x, y, z del punto originale. Si osservi infatti che, per le relazioni fra triangoli simili, valgono le seguenti uguaglianze:

$$X/(z_0-z_1) = x/(z_0-z)$$
$$Y/(z_0-z_1) = y/(z_0-z)$$

da cui si possono ricavare le espressioni:

$$X = x(z_0-z_1) / (z_0-z)$$
$$Y = y(z_0-z_1) / (z_0-z)$$

Poiché z_0 rappresenta la distanza del centro di proiezione (punto di vista) dall'origine del sistema di riferimento associato all'oggetto, al tendere di z_0 all'infinito i coefficienti delle equazioni (1) tendono al valore 1, perciò in questo caso la trasformazione si riduce a: X=x, Y=y. In altre parole, quando il centro di proiezione è portato all'infinito, la proiezione è parallela, e le coordinate X, Y trasformate coincidono con le coordinate originali. Un caso tipico di proiezione parallela è la proiezione assonometrica, la quale consiste nello "schiacciare" la figura sul piano di proiezione, azzerando la coordinata z, e considerando solo le coordinate x e y. Qui ci interessa però discutere le proiezioni prospettive per l'interesse che hanno nella visualizzazione delle figure reali.
Vedremo in seguito la matrice di trasformazione per rappresentare in forma prospettica un oggetto tridimensionale.

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.

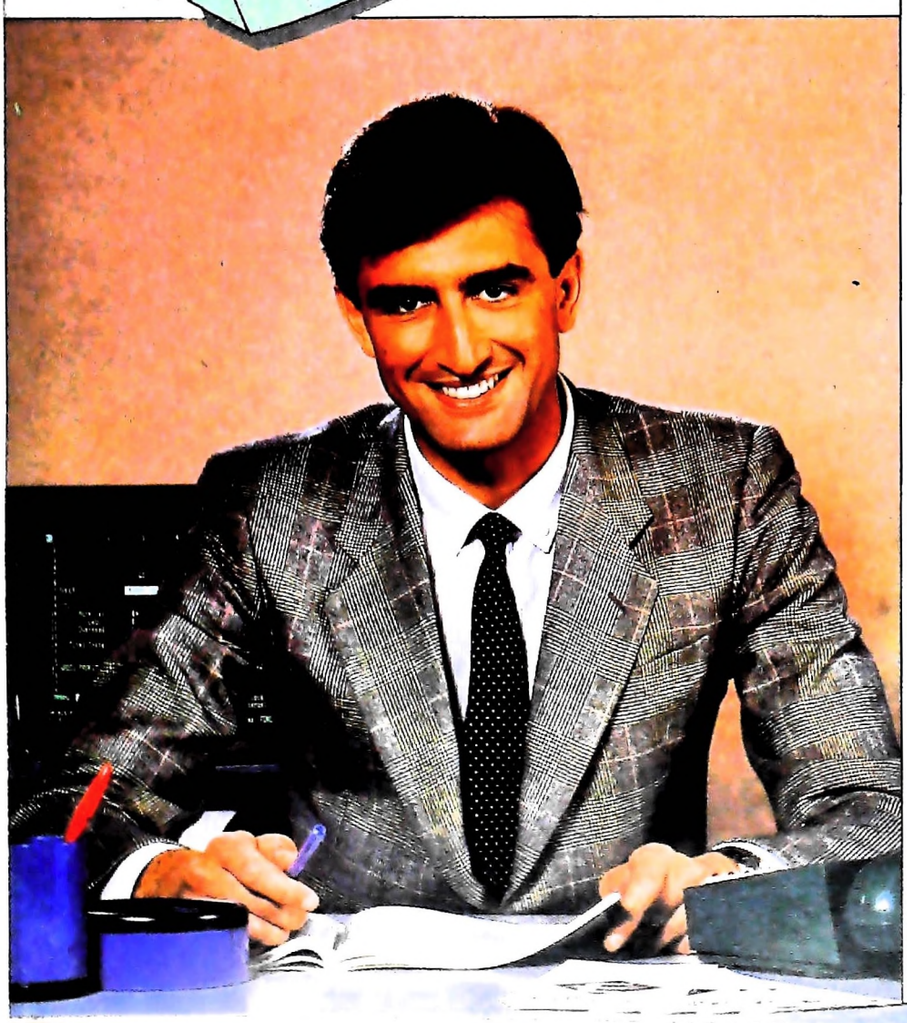
MS-DOS è un marchio Microsoft Corporation
CP/M-86 è un marchio Digital Research Inc.
UCSD-P System è un marchio
Regents of the University of California

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.